
UNIVERSIDAD DE SEVILLA
Departamento de Arquitectura y Tecnología de Computadores

# MÁQUINAS DE ESTADOS FINITOS CON MULTIPLEXIÓN DE ENTRADAS: UNA CONTRIBUCIÓN AL DISEÑO E IMPLEMENTACIÓN ELECTRÓNICA DE MÁQUINAS DE ESTADOS

**Memoria para optar al grado de Doctor**
**por la Universidad de Sevilla**
presentada por
**Ignacio García Vargas**

Director:
**Raouf Senhadji Navarro**

Sevilla, 2015

*A Marta y Alicia.*

*A la memoria de mi madre, cuyo recuerdo me ha acompañado durante la escritura de esta tesis.*

# Agradecimientos

Deseo expresar mi profundo agradecimiento a Raouf Senhadji Navarro, director de esta tesis doctoral y, sobre todo, mi compañero y amigo. Estoy convencido de que no solo esta tesis, sino toda mi actividad investigadora en la Universidad de Sevilla no hubiera sido posible sin su ayuda.

# Índice general

# Índice de Tablas

# Índice de Figuras

# Introducción

## Motivaciones

Una *Máquina de Estados Finitos* (FSM, sigla del inglés Finite State Machine) es un modelo de comportamiento de un sistema con entradas y salidas en el que el valor de las salidas viene determinado por la evolución de los valores de las entradas a lo largo del tiempo. Se trata de un modelo muy versátil que ha sido utilizado en áreas de conocimiento tan diversas como la ingeniería [Jezernik et al., 2012; Driesse et al., 2008; Serhan et al., 2008], la biología [Gao et al., 2010; Sutterlin et al., 2009] o la lingüística [Jhonson et al., 2004]. En el contexto del diseño digital, una FSM permite definir el comportamiento de un circuito secuencial, por lo que se trata de un modelo fundamental en el proceso de diseño de cualquier sistema digital [Sklyarov, 2002b]. Uno de los ejemplos más característicos de estos circuitos son las unidades de control.

Desde hace décadas, la implementación electrónica de FSM es una de las áreas más activas en el campo de la *Automatización del Diseño Electrónico* (EDA, sigla del inglés Electronic Design Automation). En términos generales, la investigación se centra hoy día en: (a) aportar nuevas arquitecturas que permitan mejorar las prestaciones de las implementaciones aprovechando las características de los nuevos dispositivos y (b) proporcionar algoritmos y técnicas que mejoren las soluciones de los problemas NP que se plantean en el proceso de diseño, tales como la codificación de estados [Aly, 2009; El-Maleh et al., 2006].

En los últimos años, el espectacular incremento de la densidad y rendimiento de los dispositivos lógicos programables, como las *Field Programmable Logic Arrays* (FPGA), los ha convertido en un serio competidor de los complejos *Circuitos Integrados de Aplicación Específica* (ASIC, sigla del in-

glés Application Specific Integrated Circuit). Aunque inicialmente las FPGA fueron utilizadas exclusivamente para prototipado o como sustituto de la lógica discreta requerida para adaptar sistemas (denominada en inglés *glue-logic*), los dispositivos actuales permiten incluir sistemas digitales completos basados en procesadores empotrados [Das et al., 2011]. El bajo coste y alta flexibilidad que caracteriza a la lógica programable hace de las FPGA una plataforma ideal para el desarrollo de complejos sistemas empotrados, a los que se les denomina habitualmente *Systems-on-Programmable-Chips* (SoPC). Estos sistemas requieren grandes cantidades de memoria —según el ITRS (International Technology Roadmap for Semiconductor), la memoria de un System-on-Chip (SoC) ocupa el 94 % del chip [Huang, 2011]— lo que ha hecho que los fabricantes incluyan un mayor número de bloques de memoria empotrados en cada nueva generación de dispositivos.

Esto hecho ha despertado interés en la utilización de estos recursos de una forma no convencional. En diseños en los que los recursos de memoria no son críticos (o, simplemente, no se necesitan), estos bloques pueden utilizarse para implementar lógica [Chiu et al., 2006]. La integración eficiente de los bloques de memoria en el flujo de diseño de las herramientas de CAD sigue siendo hoy día una asignatura pendiente. La implementación de FSM basadas en memoria han supuesto la mayor parte de las contribuciones realizas en este ámbito [Frigerio y Salice, 2007; Borowik et al., 2007; Janarthanan et al., 2007; Tiwari y Tomko, 2004; Rawski et al., 2005; Senhadji-Navarro et al., 2012; Senhadji-Navaro y Garcia-Vargas, 2015a]. Las funciones de transición y salida de una FSM pueden implementarse de forma trivial utilizando memoria ROM en lugar de celdas lógicas[1]. Sin embargo, el número de bloques de memoria requeridos crece de manera lineal con el número de salidas y de estados de la FSM, y de forma exponencial con el número de entradas, lo que se traduce en un deterioro significativo de la velocidad. En [Senhadji-Navarro et al., 2004] propusimos una técnica que permite reducir el efecto que tienen estos parámetros en la degradación de las prestaciones. Esta técnica dio lugar al modelo de *Máquina de Estados Finitos con Multiplexión de Entradas* (FSMIM, sigla del inglés Finite State Machine with Input Multiplexing), que fue presentado en la memoria realizada por el autor de esta tesis para la obtención del Diploma de Estudios Avanzados [García-Vargas, 2006]. En [García-Vargas et al., 2007], publicamos un estudio experimental en el que se demostraba la efectividad de la técnica en las FPGA. Los resultados mostraban una reducción importante en el consumo de bloques de memoria, conseguida mediante el empleo de un número reducido de celdas

[1]Las celdas lógicas son los elementos configurables básicos de los dispositivos FPGA.

lógicas, lo que se traducía en mejoras en velocidad en un porcentaje significativo de casos. La motivación principal de esta tesis doctoral es profundizar en el modelo FSMIM, tanto en el ámbito de las arquitecturas como de los algoritmos empleados para mejorar su efectividad.

## Objetivos

Los objetivos principales de la tesis pueden resumirse en los siguientes puntos:

- Proponer nuevas arquitecturas que permitan implementaciones más eficientes desde el punto de vista del número de bloques utilizados, del número de celdas lógicas utilizadas o de la velocidad.
- Estudiar los problemas involucrados en la optimización de FSMIM desde el punto de vista de la complejidad computacional.
- Proponer técnicas y algoritmos que mejoren las soluciones a los problemas involucrados en la optimización de las FSMIM.
- Realizar un estudio experimental profundo que permita: (a) comparar las nuevas aportaciones en el campo de las FSMIM con las ya existentes, así como con otras técnicas para la implementación de FSM, y (b) caracterizar las distintas técnicas y arquitecturas del modelo FSMIM con objeto de determinar su idoneidad en función del número y tipo de recursos disponibles en el dispositivo y de las restricciones de diseño.

## Organización de la tesis

La presente memoria está organizada en siete capítulos. En el capítulo 1 se introducen diversos conceptos generales relacionados con las FSM y se describen los dos tipos de arquitecturas convencionales empleadas para su implementación electrónica: las arquitecturas basadas en celdas lógicas y las basadas en memoria. Además, se analizan las características que presentan las implementaciones en dispositivos FPGA de las arquitecturas convencionales basadas en memoria.

El capítulo 2 está dedicado al modelo de FSMIM. En este capítulo se define formalmente el modelo y sus propiedades, se describen las arquitecturas propuestas para su implementación y se estudia el problema de optimización de FSMIM asociado a cada arquitectura. El problema de optimización de

FSMIM se divide en dos subproblemas denominados *agrupación de estados* y *minimización de la selección de entradas*. Los capítulos 3 y 4 presentan los diferentes algoritmos propuestos para resolver cada subproblema. El primero de ellos está dedicado al subproblema de la agrupación de estados, mientras que el capítulo 4 está dedicado al subproblema de la minimización de la selección de entradas.

En el capítulo 5 se realiza un estudio experimental en el que se analizan las prestaciones de las dos arquitecturas de FSMIM y de las arquitecturas convencionales de FSM. Este estudio se divide en tres partes: en la primera se comparan las prestaciones de ambas arquitecturas de FSMIM, en la segunda se evalúan las mejoras en la optimización de FSMIM que se consiguen con los algoritmos propuestos y, finalmente, en la tercera se comparan las implementaciones de FSMIM con las implementaciones convencionales de FSM.

El capítulo 6 presenta una breve revisión de las diferentes técnicas propuestas en la literatura para la implementación de FSM con recursos de memoria. Finalmente, en el capítulo 7 se exponen las conclusiones principales del trabajo realizado y se proponen nuevas líneas de investigación que pueden surgir a raíz de este trabajo.

Además de los siete capítulos mencionados, la memoria consta de once apéndices. En el apéndice A se detallan las publicaciones a las que ha dado lugar el trabajo de investigación realizado. Los apéndices B al E contienen información relacionada con las diferentes baterías de pruebas utilizadas en el estudio experimental. Las descripciones VHDL empleadas en los experimentos y las herramientas desarrolladas para generar las implementaciones FSMIM se presentan en los apéndices F al J. Finalmente, el apéndice K contiene información sobre las características del dispositivo FPGA y la herramienta de síntesis empleados.

Capítulo 1

# Máquinas de Estados Finitos: Diseño e implementación

## 1.1. Diseño de máquinas de estados finitos

### 1.1.1. Definición de una máquina de estados finitos

Una *Máquina de Estados Finitos* (FSM, sigla del inglés Finite State Machine) es un modelo de comportamiento de un sistema en el que el valor de la salida depende del valor actual y de los valores anteriores de la entrada. El modelo se basa en la definición de un conjunto finito de estados que determinan el funcionamiento de la FSM, es decir, su respuesta ante los valores de entrada. El estado de la máquina viene determinado por la evolución de los valores de entrada; de esta forma, la salida depende únicamente de la entrada y estado actuales.

En el ámbito del diseño digital, las FSM permiten modelar circuitos secuenciales. En este contexto, los valores de entrada y salida son elementos de $\mathbb{B}^m$ y $\mathbb{B}^n$, respectivamente, donde $\mathbb{B} = \{0, 1\}$ y $m, n \in \mathbb{N}$; por tanto, puede considerarse que las FSM tienen $m$ entradas $(x_1, \ldots, x_m) \in \mathbb{B}^m$ y $n$ salidas $(y_1, \ldots, y_n) \in \mathbb{B}^n$. En cada ciclo de reloj, el circuito almacena el estado actual de la FSM (denominado *estado presente*) y genera tanto la salida de la máquina como el estado que tendrá en el ciclo siguiente (denominado *estado siguiente*). Al final de cada ciclo se produce una *transición* en la que el estado siguiente se convierte en el estado presente de la máquina. La salida y el estado siguiente vienen determinados por una función combinacional que depende del estado presente y de la entrada. Las unidades de control [Barka-

lov y Titarenko, 2009; Katz, 1994] y los reconocedores de patrones [Borowik y Luba, 2009; Katz, 1994] son ejemplos de aplicación característicos.

**Definición 1.1.** *Sea $\mathbb{B} = \{0, 1\}$. Se define una FSM de $m$ entradas y $n$ salidas como la 4-tupla $(E, t, h, e_0)$, donde:*

- *$E$ es el conjunto de estados.*
- *$t$ es la función de transición:*

$$t : E \times \mathbb{B}^m \to E. \tag{1.1}$$

- *$h$ es la función de salida, que en el caso de las* **máquinas de Mealy** *depende tanto del estado presente como de las entradas:*

$$h : E \times \mathbb{B}^m \to \mathbb{B}^n. \tag{1.2}$$

  *En el caso de las* **máquinas de Moore***, la función de salida depende sólo del estado presente:*

$$h : E \to \mathbb{B}^n. \tag{1.3}$$

- *$e_0 \in E$ es el estado inicial.*

Se dice que una FSM está *completamente especificada* cuando las funciones de transición y salida están definidas para cualquier $(e, X) \in E \times \mathbb{B}^m$. En caso contrario, se habla de FSM *incompletamente especificadas.*

Dado un valor de entrada $X = (x_1, \ldots, x_m) \in \mathbb{B}^m$ y un estado presente $e \in E$, el estado siguiente $e' \in E$ viene determinado por la expresión $e' = t(e, X)$. El valor de salida $Y \in \mathbb{B}^n$ viene dado por la expresión $Y = h(e, X)$ en el caso de una máquina de Mealy, y por la expresión $Y = h(e)$ en el caso de una máquina de Moore.

**Definición 1.2.** *Sean $(x_1, \ldots, x_m) \in \mathbb{B}^m$ las entradas de una FSM. En una máquina de Moore, una transición que va del estado $e$ al estado $e'$ es* **independiente** *de la entrada $x_k$ si existe algún valor de las entradas*

$$x_1, x_2 \ldots, x_{k-1}, x_{k+1}, \ldots, x_m$$

*para el que se cumple que*

$$e' = t(e, x_1, \ldots, x_{k-1}, 0, x_{k+1}, \ldots, x_m) = t(e, x_1, \ldots, x_{k-1}, 1, x_{k+1}, \ldots, x_m).$$

*En una máquina de Mealy, además debe cumplirse que*

$$h(e, x_1, \ldots, x_{k-1}, 0, x_{k+1}, \ldots, x_m) = h(e, x_1, \ldots, x_{k-1}, 1, x_{k+1}, \ldots, x_m).$$

*En caso contrario, se dice que la transición* **depende** *de la entrada $x_k$. Una entrada está* **indeterminada** *en una transición cuando dicha transición es independiente de la entrada.*

### 1.1.2. Representación de una máquina de estados finitos

Aunque existen múltiples alternativas para representar una FSM, en este apartado se describen dos de los sistemas de representación utilizados con más frecuencia en la literatura.

#### 1.1.2.1. Diagrama de transición de estados

Los *Diagramas de Transición de Estados* (DTE) describen de una manera gráfica el comportamiento de la FSM, por lo que son una representación muy intuitiva para las FSM pequeñas[1]; sin embargo, son menos adecuado para las FSM de cierta complejidad. Un DTE es un pseudografo[2] $G = (E, T)$ donde:

- $E$ es el conjunto de estados de la FSM, que constituyen los nodos del pseudografo.
- $T$ es un multiconjunto[3] de pares ordenados de $E \times E$, que constituyen los arcos del pseudografo. Cada arco $(e, e') \in T$ corresponde a una transición de la FSM, en la que $e$ es el estado presente y $e'$ el estado siguiente.
- El estado inicial (denominado también *estado de reset*) se indica con un arco sin origen que incide en dicho estado.

Existen algunas diferencias entre el DTE de una máquina de Mealy y el de una máquina de Moore. En una FSM de Mealy con $m$ entradas y $n$ salidas, los arcos del grafo se etiquetan

$$x_1 x_2 \dots x_m / y_1 y_2 \dots y_n,$$

donde $(x_1, \dots, x_m) \in \mathbb{B}^m$ es el valor de entrada que activa la transición y $(y_1, \dots, y_n) \in \mathbb{B}^n$ es el valor de salida asociado a dicha transición. El símbolo "–" representa una entrada indeterminada. La figura 1.1a muestra un ejemplo de un DTE correspondiente a una máquina de Mealy con tres entradas y una salidas en el que los arcos están etiquetados como $x_1 x_2 x_3 / y_1$. Expresado en términos de la ecuación 1.1 y la ecuación 1.2, un arco $(e, e')$ se etiqueta $x_1 x_2 \dots x_m / y_1 y_2 \dots y_n$, si y solo si $e' = t(e, x_1, \dots, x_m)$ y $(y_1, \dots, y_n) = h(e, x_1, \dots, x_m)$.

[1] FSM con un reducido número de estados y transiciones.

[2] Un pseudografo es un grafo no simple en el que bucles y arcos múltiples están permitidos.

[3] Un multiconjunto difiere de un conjunto en que cada miembro del mismo tiene asociada una multiplicidad que indica cuántas veces el elemento es miembro del conjunto.

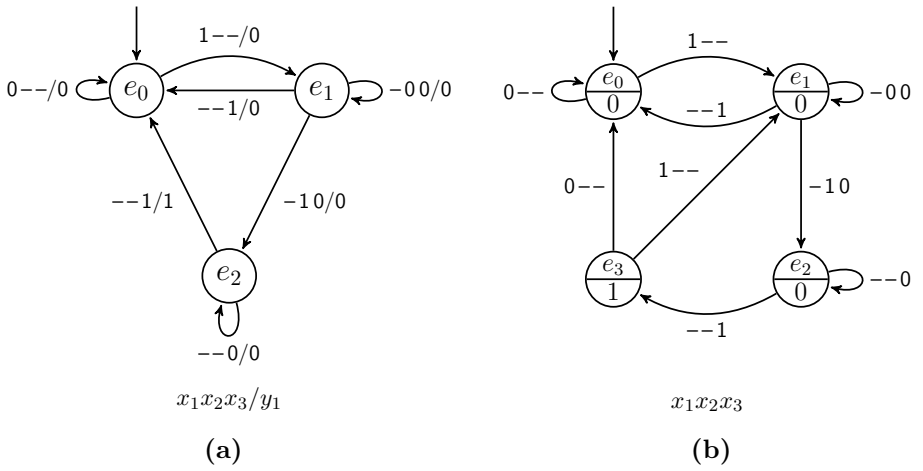



**Figura 1.1:** Ejemplos de Diagrama de Transición de Estados de una FSM con entradas $(x_1, x_2, x_3)$ y salida $y_1$: (a) máquina de Mealy y (b) máquina de Moore.

En el caso de una máquina de Moore con $m$ entradas y $n$ salidas, los nodos se etiquetan como $\frac{e}{y_1 y_2 \ldots y_n}$, donde $(y_1, \ldots, y_n) \in \mathbb{B}^n$ es el valor de salida asociado al estado $e$. Los arcos se etiquetan como $x_1 x_2 \ldots x_m$, donde $(x_1, \ldots, x_m) \in \mathbb{B}^m$ es el valor de entrada que activa la transición. La figura 1.1b muestra un ejemplo de un DTE correspondiente a una máquina de Moore con tres entradas y una salida en el que los arcos están etiquetados como $x_1x_2x_3$ y los nodos como $\frac{e}{y_1}$.

#### 1.1.2.2. Tabla de transición de estados

Las *Tablas de Transición de Estados* (TTE) son menos intuitivas que los DTE; sin embargo, son más adecuadas para representar FSM complejas. Por otra parte, las TTE permiten ilustrar convenientemente las técnicas que se estudian en esta tesis doctoral.

En una TTE el comportamiento de la FSM se especifica mediante una tabla. Cada fila de esta tabla corresponde a una transición de la FSM y está constituida por una 4-tupla $(en, ep, es, sal)$, donde $en$ es el valor de entrada; $sal$, el valor de salida; $ep$, el estado presente; y $es$, el estado siguiente. En la *Tabla de Transición de Estados Simbólica*, los estados de la FSM aparecen como símbolos (en adelante, se supondrá que todas las TTE son simbólicas excepto en aquellos casos en los que se indique lo contrario). La figura 1.2 muestra la TTE correspondiente al DTE mostrado en la figura 1.1a.

| **en** | | | **ep** | **es** | **sal** |
|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | | | $y$ |
| 1 | – | – | $s_0$ | $s_1$ | 0 |
| 0 | – | – | $s_0$ | $s_0$ | 0 |
| – | 1 | 0 | $s_1$ | $s_2$ | 0 |
| – | 0 | 0 | $s_1$ | $s_1$ | 0 |
| – | – | 1 | $s_1$ | $s_0$ | 0 |
| – | – | 1 | $s_2$ | $s_0$ | 1 |
| – | – | 0 | $s_2$ | $s_2$ | 0 |

**Figura 1.2:** Ejemplo de Tabla de Transición de Estados Simbólica

### 1.1.3. Estados equivalentes

Se dice que dos estados son *equivalentes* si para cualquier valor de entrada, sus salidas son las mismas y sus transiciones van a los mismos estados o a estados equivalentes.

**Definición 1.3.** *Sea $F = (E, t, h, e_0)$ una FSM de $m$ entradas. Dos estados $e, e' \in E$ son equivalentes, denotado por $e \approx e'$, si y solo si para todo $X \in \mathbb{B}^m$, se cumple que*

$$h(e, X) = h(e', X) \tag{1.4}$$

*y*

$$t(e, X) = t(e', X) \quad \vee \quad t(e, X) \approx t(e', X). \tag{1.5}$$

*Se dice que un estado es* **redundante** *cuando existe otro equivalente a él.*

Por ejemplo, en la FSM de la figura 1.3, $e_1$ y $e_2$ son estados equivalentes, por lo que podrían fusionarse en un único estado sin afectar al comportamiento de la FSM. Por otra parte, uno de los dos estados es redundante.

### 1.1.4. Máquinas de estados finitos equivalentes

Se dice que dos FSM son *equivalentes* si para cualquier secuencia de entradas se obtiene la misma salida en ambas, independientemente del número de estados y de las funciones de transición y salida de cada una de ellas.

**Definición 1.4.** *Sean $F = (E, h, t, e_0)$ y $F' = (E', h', t', e'_0)$ dos FSM de $m$ entradas. Se dice que $F$ y $F'$ son equivalentes (denotado por $F \sim F'$) si y*

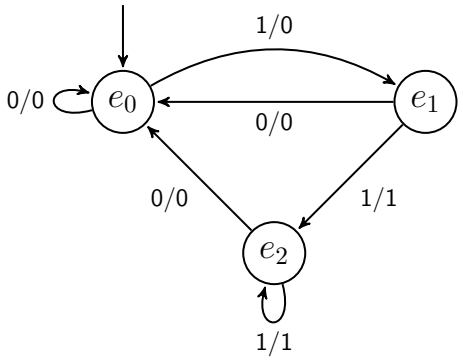



**Figura 1.3:** Ejemplo de FSM con estados equivalentes

*solo si para cualquier secuencia de entradas* $\langle X^{(1)}, \ldots, X^{(n)} \rangle$*, con* $X^{(i)} \in \mathbb{B}^m$*, se cumple que*

$$\begin{aligned} & h(t(\ldots t(t(e_0, X^{(1)}), X^{(2)}) \ldots, X^{(n-1)}), X^{(n)}) \\ & \quad = h'(t'(\ldots t'(t'(e'_0, X^{(1)}), X^{(2)}) \ldots, X^{(n-1)}), X^{(n)}). \end{aligned} \tag{1.6}$$

Por ejemplo, las dos FSM mostradas en la figura 1.1a y en la figura 1.1b son equivalentes.

### 1.1.5. Diseño e implementación electrónica de máquinas de estados finitos

El proceso de diseño e implementación electrónica de FSM incluye distintas etapas que van desde la descripción inicial de la FSM hasta su implementación final. Aunque el número y naturaleza de las etapas depende de la metodología, de las herramientas y de la tecnología utilizadas, en términos generales, el proceso consiste en las siguiente fases:

- **Especificación de la FSM**. El comportamiento de la FSM se describe utilizando alguna representación abstracta o lenguaje de especificación, cuya elección dependerá fundamentalmente de su disponibilidad en las herramientas empleadas. Es habitual que las herramientas de diseño incluyan herramientas gráficas que facilitan la descripción de la FSM, como *State Diagram Editor* (StateCAD), en el caso de Xilinx® [Xilinx, 2007], o *State Machine Editor*, en el caso de Altera® [Altera, 2010]. Sin embargo, la opción más extendida es utilizar lenguajes de

descripción de hardware como Verilog [Palnitkar, 2003] o VHDL [Ashenden, 2001]. Debido al creciente desarrollo de lenguajes de alto nivel para describir hardware, tales como Handel-C [Kamat et al., 2009] o SystemC [Grotker, 2002], cada vez es más habitual describir sistemas que incluyen FSM utilizando este tipo de lenguajes. Algunas herramientas, como Alliance [Greiner y Pcheux, 1992], utilizan el formato de descripción *KISS2* (descrito en el apéndice B.1). Este formato, que es utilizado por las herramientas desarrolladas para esta tesis doctoral, permite básicamente describir la TTE de las FSM.

- **Minimización de estados**. Es habitual que las descripciones de FSM realizadas por los diseñadores contengan más estados de los que son estrictamente necesarios, lo que afecta negativamente a las prestaciones de la implementación correspondiente. La minimización de estados consiste en eliminar los estados redundantes de una FSM para obtener otra equivalente con el mínimo número de estados. En términos generales, esto permite mejorar las prestaciones del circuito que la implementa. Para las FSM completamente especificadas, la tarea es relativamente sencilla, existiendo algoritmos de coste $O(n \log n)$ para llevarla a cabo [Avedillo et al., 1994, 1990]. Sin embargo, la minimización de estados en máquinas incompletamente especificadas es un problema *NP*-completo [Pfleeger, 1973; Avedillo et al., 1994, 1990].

- **Asignación o codificación de estados**. En esta fase se asigna a cada estado un código (o patrón de bits) único que lo identifica. Para la elección del código, existen diferentes sistemas de codificación generales tales como la codificación binaria, *one-hot*, *Gray* o *Jhonson* [Uma y Dhavachelvan, 2012]. Puesto que las funciones de transición y salida dependen del estado presente, la codificación de estados puede tener una influencia significativa en las prestaciones obtenidas por las implementaciones de FSM. Esto justifica el interés existente en el desarrollo de algoritmos para buscar la codificación de los estados de una FSM que optimice el coste de su posterior implementación en términos del consumo de recursos (área), de la frecuencia de operación máxima (velocidad) o de la potencia consumida. Aunque durante muchos años han sido propuestas numerosas técnicas, la codificación de estados sigue siendo en la actualidad un problema que suscita mucho interés [Grzes y Solov'ev, 2014]. Esto es debido a la diversidad de recursos disponibles para la implementación de FSM, que ha crecido con la aparición de los modernos dispositivos FPGA. Se trata

de un problema *NP*-completo [Aly, 2009] que ha sido abordado con múltiples estrategias, tales como la optimización combinatoria [Villa y Sangiovanni-Vincentelli, 1990], los algoritmos genéticos [Almaini et al., 1995], el enfriamiento simulado [Aly, 2009] o la lógica difusa [El-Maleh et al., 2006].

Si se sustituye cada estado de una TTE simbólica por el código correspondiente, se obtiene una *Tabla de Transición de Estados Codificada*. La figura 1.4 muestra un ejemplo de TTE codificada, que ha sido obtenida aplicando una codificación binaria a los estados de la FSM de la figura 1.1a, cuya TTE simbólica se muestra en la figura 1.2. Las TTE codificadas son el punto de partida de la fase de implementación.

- **Implementación**. La última fase consiste en la implementación de la máquina de estados. Respecto a las tecnologías disponibles, existen numerosas alternativas que van desde los *Circuitos Integrados de Aplicación Específica* (ASIC, sigla del inglés Application-Specific Integrated Circuit) hasta la implementación mediante dispositivos programables como las *Field Logic Programmable Gate Array* (FPGA) o *Complex Programmable Logic Device* (CPLD). En relación a esta fase, la investigación se centra en el desarrollo de nuevas arquitecturas y técnicas de optimización que permitan implementaciones más eficientes, en términos de velocidad, área y potencia consumida. En esta tesis doctoral, las arquitecturas y técnicas propuestas se circunscriben a los dispositivos FPGA.

| **en** | | | **ep** | **es** | **sal** |
|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | | | $y$ |
| 1 | – | – | 00 | 01 | 0 |
| 0 | – | – | 00 | 00 | 0 |
| – | 1 | 0 | 01 | 10 | 0 |
| – | 0 | 0 | 01 | 01 | 0 |
| – | – | 1 | 01 | 00 | 0 |
| – | – | 1 | 10 | 00 | 1 |
| – | – | 0 | 10 | 10 | 0 |

**Figura 1.4:** Ejemplo de Tabla de Transición de Estados Codificada

## 1.2. Implementación electrónica de máquinas de estados finitos

### 1.2.1. Arquitecturas convencionales para la implementación de máquinas de estados finitos

Existen distintas alternativas a la hora de implementar una FSM en un dispositivo electrónico. En el caso de dispositivos programables como las FPGA, la elección de una u otra alternativa va a venir determinada fundamentalmente por el tipo de recursos disponibles en la plataforma destino. La arquitectura más extendida está constituida por un registro que almacena el estado presente y lógica combinacional que implementa las funciones de transición y salida [Katz, 1994; Kubátová, 2005; Feske et al., 1997].

Las FSM de Mealy y de Moore presentan una diferencia importante desde el punto de vista de la implementación. La figura 1.5 muestra los diagramas de bloques de ambos tipos de FSM. En la implementación de una FSM de Moore (figura 1.5a), existen dos bloques combinacionales distintos. Uno de ellos implementa la función de transición, por lo que genera el estado siguiente a partir del estado presente y la entrada. El otro bloque implementa la función de salida, generando la salida de la FSM a partir del estado presente almacenado en el registro. Por tanto, como el estado es modificado exclusivamente en cada flanco activo de reloj, la salida está sincronizada con la señal de reloj.

En el caso de una máquina de Mealy (figura 1.5b), existe un único bloque combinacional que implementa tanto la función de transición como la de salida, ya que ambas dependen del estado presente y la entrada. Sin embargo, a diferencia del estado presente, la salida no está registrada; por tanto, si la entrada no está sincronizada con la señal de reloj, la salida tampoco lo estará. Debido a esta característica, las máquinas de Mealy con salidas asíncronas presentan dos inconvenientes. Por una parte, pueden aparecer *gliches* en la señal de salida, que son una causa importante de problemas en los sistemas de control real. Por otra parte, los flujos de diseño para los dispositivos FPGA (ofrecidos por fabricantes como Xilinx) están pensados para sistemas completamente síncronos [Xilinx, 2008], por lo que este tipo de máquinas de Mealy son más difíciles de testar [Govindarajalu, 2010].

Una posible solución a ambos problemas consiste en registrar también las señales de salida, tal como muestra la figura 1.6. De esta manera, la salida sólo cambia en los flancos activos de reloj. Sin embargo, la FSM de Mealy con salidas sincronizadas no tiene el mismo comportamiento entrada/salida

que la máquina original, pues los cambios en la salida tienen efecto con un ciclo de retraso respecto a la FSM original.

La implementación de los bloques combinacionales de las arquitecturas mostradas puede llevarse a cabo de muy distintas formas, en función de los recursos disponibles en el dispositivo FPGA y de las restricciones de área y velocidad impuestas [Barkalov y Titarenko, 2009]. Habitualmente,

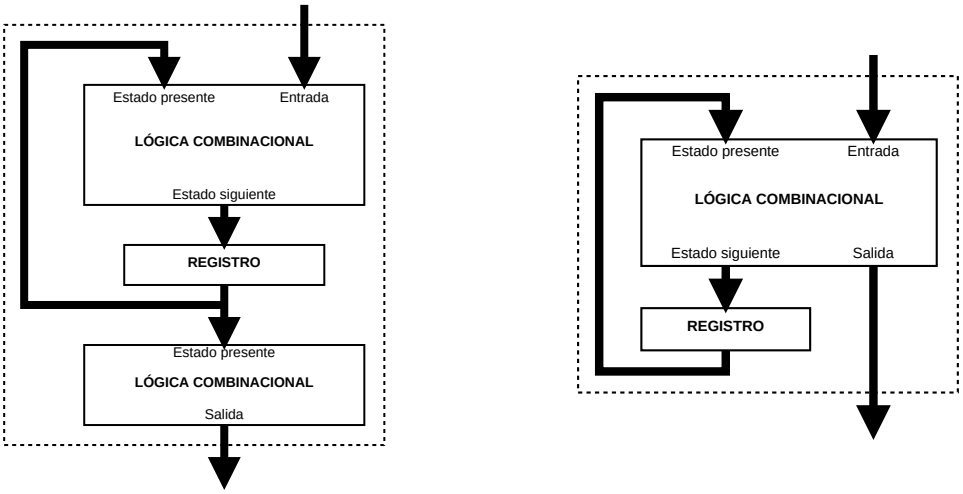



**Figura 1.5:** Arquitecturas para la implementación de FSM: (a) FSM de Moore y (b) FSM de Mealy

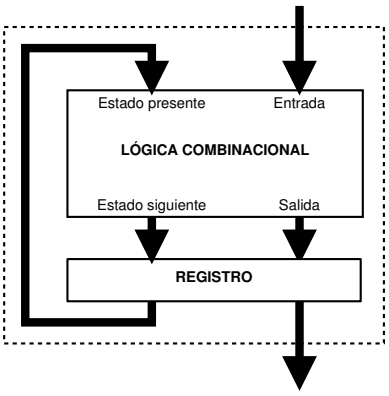



**Figura 1.6:** Arquitectura para la implementación de FSM de Mealy con salidas sincronizadas

estos bloques combinacionales se implementan utilizando *Tabla de Búsqueda* (LUT, sigla del inglés Look-Up Table). Sin embargo, en dispositivos FPGA con bloques de memoria empotrados, las funciones de transición y salida pueden ser implementadas como memoria ROM [Borowik y Luba, 2009; Rawski et al., 2005; Selvaraj et al., 2002; Le Gal et al., 2010]. El trabajo de investigación presentado en esta tesis doctoral está enmarcado en este tipo de implementaciones, que se describen con detalle en la sección siguiente.

Una arquitectura distinta consiste en utilizar un contador en vez de un registro, de manera que, controlando las señales de cuenta y carga convenientemente es posible simular el comportamiento de la FSM. En estas implementaciones es habitual el uso de decodificadores y multiplexores para reducir la lógica necesaria [Katz, 1994].

### 1.2.2. Implementaciones basadas en memoria

En los últimos años, el número de bloques de memoria empotrados disponibles en los dispositivos FPGA ha crecido espectacularmente. Como ejemplo, la figura 1.7 muestra la cantidad total de memoria de bloques disponible en el dispositivo FPGA de mayor tamaño de cada una de las familias de la serie Virtex de Xilinx. Este crecimiento viene motivado principalmente por los requerimientos de memoria de las actuales aplicaciones basadas en *System on Programable Chip* (SoPC). Según las estimaciones del *International Technology Roadmap for Semiconductors* (ITRS), la memoria empotrada de un SoPC ocupa un 94 % del área total del chip [Huang, 2011].

La posibilidad de usar estos bloques de memoria para implementar funciones lógicas en vez de ser utilizados como memoria convencional, ha permitido el surgimiento de numerosas técnicas que tienen como objeto integrar estos bloques en el flujo de diseño de las herramientas de CAD [Chiu et al., 2006; Borowik y Luba, 2009; Cong y Yan, 2000]. Especial interés han suscitado las técnicas que permiten la implementación eficiente de FSM en ROM [Borowik et al., 2007; Borowik y Luba, 2009; Rawski et al., 2003, 2005; Selvaraj et al., 2002; Le Gal et al., 2010; Tiwari y Tomko, 2004; Senhadji-Navarro et al., 2004]. Además, la posibilidad de configurar los bloques como RAM en vez de ROM ha abierto una prometedora vía para la implementación de FSM reconfigurables [Sklyarov, 2002b; Rafla y Davis, 2006; Senhadji-Navarro et al., 2007; Senhadji-Navaro y Garcia-Vargas, 2015b]. En este tipo de FSM, su comportamiento puede ser modificado en tiempo de ejecución realizado operaciones de escritura sobre la memoria que implementa las funciones de transición y salida.

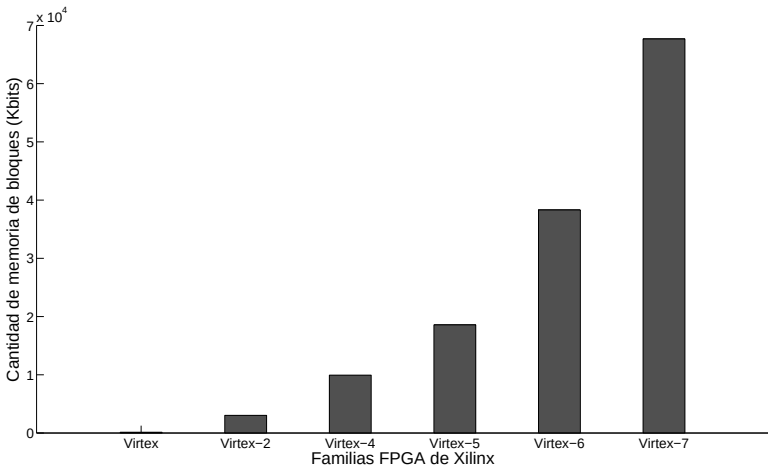



**Figura 1.7:** Cantidad de memoria de bloques en las familias Virtex de Xilinx

#### 1.2.2.1. Arquitecturas convencionales basadas en memoria

La figura 1.8 muestra los diagramas de bloques de las diferentes arquitecturas empleadas para la implementación convencional de FSM basada en memoria. La única diferencia entre estas arquitecturas y las mostradas en la sección 1.2.1 consiste en que los bloques combinacionales han sido implementados con memoria ROM asíncrona.

En el caso de una máquina de Mealy (figuras 1.8b y 1.8c), la dirección de la memoria ROM es una palabra constituida por los bits de codificación del estado presente y por las señales de entrada. Esta dirección determina la palabra de la ROM, que contiene los bits de codificación del estado siguiente (según la ecuación 1.1) y el valor de la salida de la FSM (según la ecuación 1.2); es decir, contiene las transiciones de la FSM. La figura 1.9 muestra un ejemplo de implementación de una máquina de Mealy: la figura 1.9a muestra la TTE simbólica; la figura 1.9b, la TTE codificada, y la figura 1.9c, el contenido de la memoria ROM.

En el caso de una máquina de Moore (figura 1.8a), la salida es generada por una segunda ROM (según la ecuación 1.3), que es direccionada por la codificación del estado presente.

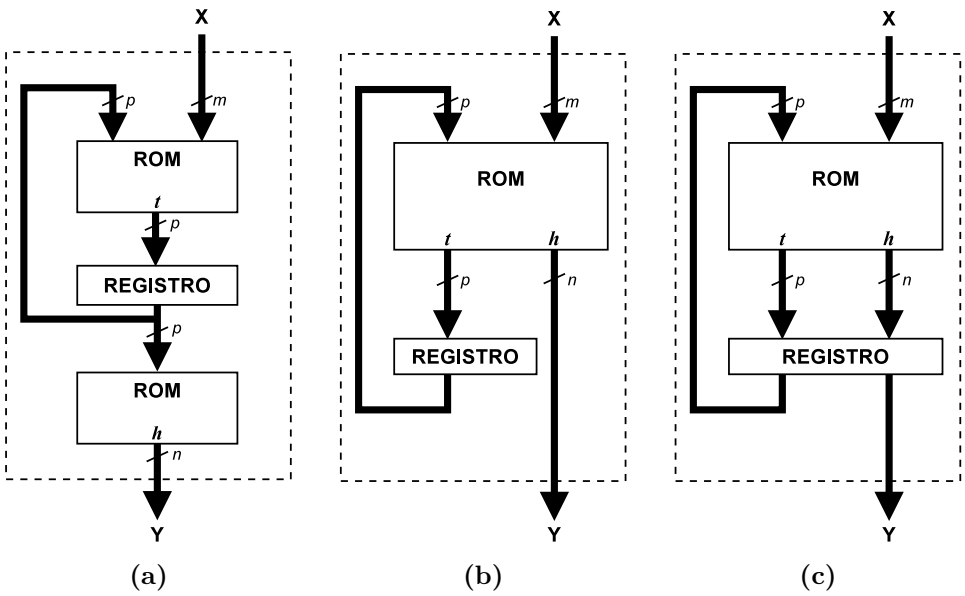



**Figura 1.8:** Arquitecturas para la implementación convencional de FSM basada en memoria: (a) máquina de Moore, (b) máquina de Mealy y (c) máquina de Mealy con salidas sincronizadas.

### 1.2.3. Análisis de prestaciones de las implementaciones convencionales basadas en memoria

En esta sección, se estudia la influencia del número de entradas, salidas y estados de una FSM sobre las prestaciones obtenidas con las implementaciones convencionales basadas en ROM. Para analizar el área ocupada independientemente del dispositivo utilizado, se empleará como métrica el tamaño en bits de la ROM requerida por la implementación. Las características específicas del dispositivo FPGA (tamaño y configuración de los bloques de memoria, número de niveles de multiplexores empotrados, número de entradas de la LUT, etc.) determinarán el consumo final final de LUT y bloques de memoria. Los resultados de velocidad dependen fundamentalmente del dispositivo utilizado; por tanto, no es posible realizar un estudio similar al del área ocupada. En cualquier caso, en la siguiente sección se resumen las conclusiones de un estudio publicado en [Senhadji-Navarro et al., 2012], en el que se analiza la influencia de las características específicas de diferentes familias de FPGA en la velocidad, número de LUT y número de bloques de memoria empotrados usados en las implementaciones de FSM basadas en memoria.

| **en** | | **ep** | **es** | **sal** |
|---|---|---|---|---|
| $x_1$ | $x_2$ | | | $y$ |
| 0 | – | $e_0$ | $e_0$ | 0 |
| 1 | – | $e_0$ | $e_1$ | 0 |
| 0 | 0 | $e_1$ | $e_1$ | 0 |
| 1 | 0 | $e_1$ | $e_2$ | 0 |
| – | 1 | $e_1$ | $e_0$ | 1 |
| – | 0 | $e_2$ | $e_2$ | 0 |
| – | 1 | $e_2$ | $e_0$ | 1 |

**(a)**

| **en** | | **ep** | **es** | **sal** |
|---|---|---|---|---|
| $x_1$ | $x_2$ | | | $y$ |
| 0 | – | 00 | 00 | 0 |
| 1 | – | 00 | 01 | 0 |
| 0 | 0 | 01 | 01 | 0 |
| 1 | 0 | 01 | 10 | 0 |
| – | 1 | 01 | 00 | 1 |
| – | 0 | 10 | 10 | 0 |
| – | 1 | 10 | 00 | 1 |

**(b)**

**ROM**

| Dirección | | | | Contenido | | |
|---|---|---|---|---|---|---|
| $\text{ep}_1$ | $\text{ep}_0$ | $x_1$ | $x_2$ | $\text{es}_1$ | $\text{es}_0$ | $y$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **0** | **0** | **0** | **1** | **0** | **0** | **0** |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| **0** | **0** | **1** | **1** | **0** | **1** | **0** |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| **0** | **1** | **1** | **1** | **0** | **0** | **1** |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| **1** | **0** | **1** | **0** | **1** | **0** | **0** |
| **1** | **0** | **1** | **1** | **0** | **0** | **1** |
| 1 | 1 | 0 | 0 | – | – | – |
| 1 | 1 | 0 | 1 | – | – | – |
| 1 | 1 | 1 | 0 | – | – | – |
| 1 | 1 | 1 | 1 | – | – | – |

**(c)**

**Figura 1.9:** Implementación en ROM de una máquina de Mealy: (a) TTE simbólica, (b) TTE codificada y (c) contenido de la ROM.

Sea $F = (E, t, h, e_0)$ una FSM con $m$ entradas y $n$ salidas. Se requieren, por tanto, $p = \lceil \log_2 |E| \rceil$ bits de codificación de estados (este cálculo se ha realizado suponiendo que se emplea la codificación binaria o la codificación *Gray*, otro tipo de codificación como *one-hot* o *Jhonson* darían lugar a un mayor número de bits de codificación).

En una máquina de Mealy (arquitecturas mostradas en las figuras 1.8b y 1.8c) cada palabra de la ROM contiene $n$ bits de salida y $p$ bits de codificación de estados; es decir, el ancho de la ROM es $n + p$. Por otra parte, la profundidad mínima de la ROM es

$$2^m |E| \text{ palabras.} \tag{1.7}$$

Para garantizar que se alcanza el valor de la ecuación 1.7 deben cumplirse dos condiciones: (a) que los códigos asignados a los estados sean códigos binarios con valores entre 0 y $|E| - 1$; y (b) que la parte alta de la dirección de la ROM esté formada por los bits de codificación de estados. Por ejemplo, en la ROM mostrada en la figura 1.9c, los 3 estados de la FSM se han codificado con los códigos binarios $\{0, 1, 2\}$ y la dirección de la ROM está formada por los bits "$\text{ep}_1\text{ep}_0x_1x_2$" (donde $\text{ep}_1$ y $\text{ep}_0$ son los bits de codificación del

estado presente); como resultado, sólo se acceden a las direcciones de la $0000_{(2}$ a la $1011_{(2}$. En el resto del capítulo se supondrá que se cumplen las condiciones anteriores en todas las implementaciones de FSM basadas en memoria. Generalmente, el número de estados de una FSM no es potencia de 2, es decir, $|E| \leq 2^p$; por tanto, $2^m|E| \leq 2^{m+p}$. El tamaño de la ROM de una FSM de Mealy viene dado por la expresión

$$T_{\text{FSM-ROM}}^{\text{Mealy}} = 2^m|E|(p+n) \leq 2^{m+p}(p+n) \text{ bits.} \tag{1.8}$$

En una máquina de Moore (figura 1.8a), la memoria que contiene las transiciones tiene la misma profundidad que en el caso de una máquina de Mealy, pero su ancho es $p$, ya que no contiene la salida. La ROM que implementa la función de salida tiene $|E|$ palabras de $n$ bits de ancho. Por lo tanto, la cantidad de memoria requerida es

$$T_{\text{FSM-ROM}}^{\text{Moore}} = 2^m|E|p + |E|n \leq 2^{m+p}p + 2^p n \text{ bits.} \tag{1.9}$$

Como muestran las ecuaciones anteriores, el crecimiento del tamaño de la memoria requerida es lineal respecto al número de estados y al número de salidas; sin embargo, es exponencial respecto al número de entradas. Por otra parte, en términos generales, el consumo de memoria de las implementaciones de FSM es ineficiente en estas arquitecturas, debido fundamentalmente a la existencia en la ROM de información redundante y de palabras no usadas.

La información redundante aparece como consecuencia de las indeterminaciones en las entradas de la FSM. Cada entrada indeterminada en una transición tiene el efecto de duplicar el número de palabras de la ROM en las que hay que almacenar el estado siguiente y las salidas de dicha transición. Por lo tanto, una transición con $k$ entradas indeterminadas da lugar a $2^k - 1$ palabras redundantes en la ROM. Por ejemplo, en la figura 1.9c la dirección y el contenido de la ROM correspondiente a las palabras redundantes se muestran resaltados en negrita.

En general, la memoria ROM utilizada para implementar una FSM contendrá palabras no usadas en los casos en los que el número de estados no sea potencia de dos. Sin embargo, el número de palabras desaprovechadas depende de los recursos empleados para implementar la ROM (este problema se estudiará con más detalle en el siguiente apartado). En el ejemplo de la figura 1.9c, las cuatro palabras de las direcciones altas de la ROM están desaprovechadas debido a que la FSM tiene tres estados (en el ejemplo, se supone una ROM con capacidad para $2^4$ palabras).

En los últimos años se han venido desarrollando una serie de técnicas encaminadas a reducir el tamaño de memoria necesario para las implementaciones de FSM [Borowik et al., 2007; Rawski et al., 2005, 2011; Bukowiec,

2008; Barkalov et al., 2012; Sklyarov, 2000] (véase el apéndice 6). Estas técnicas permiten, por una parte, mitigar el efecto del número de entradas en el área ocupada y, por otra, reducir la cantidad de información superflua almacenada en la ROM. Los trabajos propuestos en esta tesis doctoral se enmarcan en esta línea de investigación.

En una implementación que utilice exclusivamente ROM, la asignación de estados no influye en la velocidad o el área, pues la codificación sólo va a determinar en qué palabras de la ROM va a almacenarse la información relativa al estado[4] [Katz, 1994]. Sin embargo, cuando se utilizan técnicas de optimización para reducir el tamaño de la memoria que incluyen lógica combinacional adicional, la codificación sí puede tener un efecto importante. En la bibliografía existen algunos trabajos en este sentido [Borowik et al., 2007; Rawski et al., 2005].

### 1.2.4. Caracterización de las implementaciones convencionales basadas en memoria en dispositivos FPGA

En una FPGA, una memoria ROM se implementa mediante elementos de memoria (configurados como memoria ROM) cuyas salidas se encuentran conectadas a un banco de multiplexores, que será denominado *banco de multiplexores de la ROM*. En una operación de lectura, estos multiplexores permiten conectar la salida de los elementos de memoria que contienen la palabra que se va a leer con la señal de salida de la ROM. La figura 1.10 muestra la implementación de una ROM de geometría $2^r \times n$ constituida por elementos de memoria de $2^s \times t$. Los elementos se organizan en una matriz de $f \times c$ bloques, donde $f = 2^{r-s}$ y $c = \lceil \frac{n}{t} \rceil$. Se requieren $c$ multiplexores de $f$ entradas y $t$ bits de ancho (en el caso de que $n$ no sea múltiplo de $t$, el último multiplexor puede ser de $n$ mód $t$ bits de ancho). Por tanto, la complejidad de los multiplexores crece al aumentar la profundidad de la memoria ROM (ya que la complejidad de un multiplexor depende del número de entradas).

La mayoría de los fabricantes de FPGA (tanto Xilinx como Altera) incluyen dos tipos distintos de elementos de memoria en sus dispositivos: (a) LUT que pueden ser configuradas como pequeñas memorias y (b) bloques de memoria empotrados. Las memorias ROM construidas con LUT se denominan ROM distribuidas y se usan habitualmente para implementar memorias de tamaño reducido. Éstas memorias permiten lectura asíncrona, por lo que pueden utilizarse para implementar FSM de Moore y de Mealy utilizando las arquitecturas mostradas en la figura 1.8a y figura 1.8b, respectivamente.

[4]Esta afirmación es válida para implementaciones basadas en memoria de FSM que cumplen las condiciones para garantizar el valor de la ecuación 1.7 (página 18).

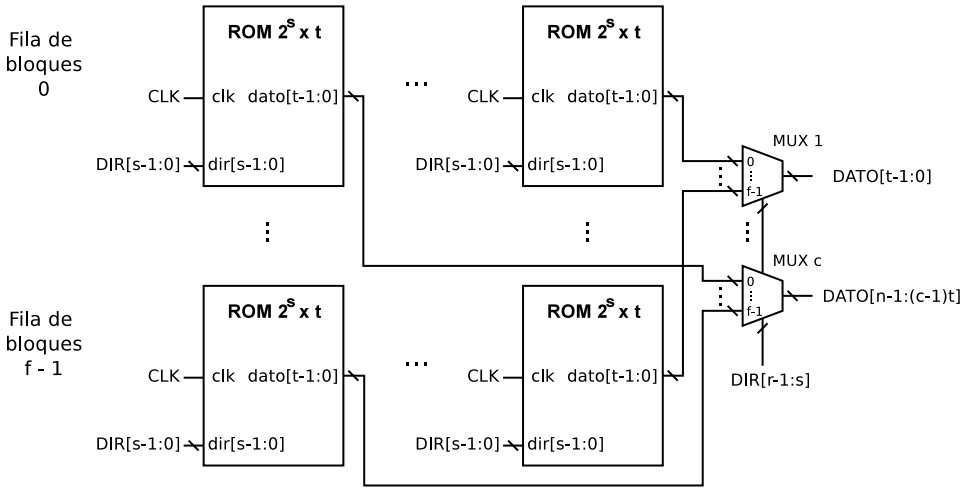



**Figura 1.10:** Implementación en FPGA de una ROM con geometría $2^r \times n$. En el diagrama $f = 2^{r-s}$ y $c = \lceil \frac{n}{t} \rceil$.

Las memorias ROM construidas con bloques de memoria empotrados se denominan ROM de bloques, este será el tipo de memoria empleada en esta tesis doctoral. El uso bloques de memoria empotrados para implementar las funciones de transición y salida de una FSM permiten liberar LUT, que pueden ser utilizadas para otros propósitos. De hecho, el objetivo principal de algunas de las técnicas propuestas en esta tesis es conseguir importantes reducciones en el consumo de bloques mediante el empleo de un número reducido de LUT. En la inmensa mayoría de los dispositivos FPGA existentes en el mercado, los bloques de memoria son síncronos; por lo tanto, sólo pueden utilizarse para la implementación de FSM de Mealy con salidas sincronizadas (figura 1.8c). Tal como se mostrará en la sección 1.2.5, esto no supone una limitación importante.

Cuando se implementa una ROM de bloques, es frecuente que un porcentaje de los bloques utilizados no se aprovechen completamente. Por ejemplo, en la implementación de la ROM de la figura 1.10, si la profundidad de la ROM no es múltiplo de $2^s$, entonces los bloques de la fila de bloques correspondiente a las direcciones altas de la ROM (fila $f - 1$ de la figura) estarán ocupados parcialmente, quedando desaprovechado el espacio libre que contienen. Este efecto se denominará *fraccionamiento interno* debido a que sólo se usa una fracción de la cantidad de memoria de los bloques afectados. Con el fin de paliar este problema, los fabricantes de dispositivos FPGA han dotado a los bloques de memoria empotrados de la capacidad de configurar

su ancho y su profundidad. Obviamente, cuando se implementa una memoria empleando bloques con diferentes *geometrías*[5], el esquema utilizado para construir la ROM difiere ligeramente del mostrado en la figura 1.10.

El número de entradas de los multiplexores y, por tanto, su complejidad, disminuye con el aumento de la profundidad de los bloques empleados para construir la ROM, lo que favorece el incremento de la frecuencia de operación máxima. En el caso extremo de que la profundidad de la ROM sea menor o igual que la profundidad máxima de los bloques, no se requiere ningún multiplexor (en Xilinx, por ejemplo, los bloques de 18 Kbits disponibles en los dispositivos de la familia Spartan-6 pueden configurarse con una profundidad de hasta 16K palabras). Como consecuencia, para memorias ROM de cierta profundidad, las implementaciones basadas en ROM de bloques son más rápidas que las basadas en ROM distribuida. A pesar de ello, la tendencia seguida por los fabricantes en los últimos años no ha sido aumentar el tamaño de los bloques, sino la cantidad de bloques disponibles en los dispositivos FPGA. Esto se debe a que el fraccionamiento interno de los bloques aumenta con el tamaño de estos, especialmente cuando se implementan memorias cuyo tamaño es poco significativo en comparación al de los bloques. Para reducir el fraccionamiento interno, los fabricantes han incluido en sus dispositivos bloques de memoria que pueden dividirse en dos bloques independientes de la mitad de tamaño [Xilinx, 2011d, 2014].

Desde un punto de vista teórico, debido a que elimina estados redundantes, la etapa de minimización de estados favorece la reducción del tamaño de memoria requerido para la implementación de la FSM. En un dispositivo FPGA, esto puede repercutir en el ahorro del consumo de bloques si se consigue eliminar la cantidad suficiente de estados. La cantidad mínima de estados que deben eliminarse para poder ahorrar bloques depende del número de estados contenidos en la fila de bloques correspondiente a las direcciones altas de la ROM (la fila $f-1$ en la figura 1.10). Dada una FSM con $m$ entradas, el número de direcciones diferentes asociadas a cada estado es $2^m$ en la memoria que contiene las transiciones. Por tanto, suponiendo que la profundidad de los bloques de la fila correspondiente a las direcciones altas de la ROM es de $2^s$ palabras y que $s > m$, el número máximo de estados que se pueden almacenar en dicha fila es $2^{s-m}$ en el caso de la memoria que contiene las transiciones (FSM de Moore o de Mealy). En el caso de la memoria que contiene las salidas (FSM de Moore), el número máximo de

[5]En este documento se empleará el término geometría para hacer referencia a la *profundidad* (medida en palabras) y al *ancho* (medido en bits) de una memoria o de un bloque de memoria empotrado. Los valores de la geometría de memoria se expresarán como *profundidad* × *ancho*.

estados que pueden almacenarse en dicha fila es $2^s$, ya que cada estado esta asociado a una dirección de esta memoria.

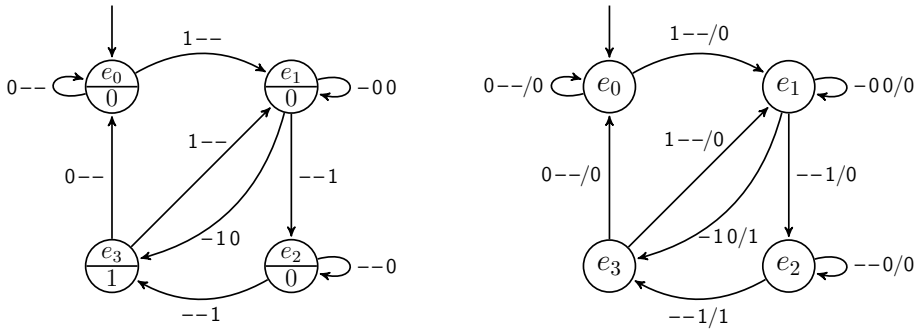
En [Senhadji-Navarro et al., 2012], presentamos un estudio de la influencia de los parámetros de una FSM (número de entradas, número de salidas y número de bits de codificación de estados) en las prestaciones de las implementaciones basadas en memoria. Se realizó un estudio tanto de velocidad como de área en distintas familias de FPGA con objeto de determinar la influencia que tiene el tamaño de LUT, el tamaño de los bloques de memoria y el número de niveles de multiplexores empotrados sobre las prestaciones obtenidas. Los resultados mostraron que el aumento del número de salidas provoca un crecimiento lineal del consumo de recursos y no afecta significativamente a la velocidad. Sin embargo, la influencia del crecimiento del número de estados y entradas sobre las prestaciones es mayor debido a que causan un aumento de la profundidad, que repercute negativamente en la complejidad de los multiplexores. Este efecto es especialmente importante en el caso del número de entradas, ya que su crecimiento provoca un incremento exponencial del numero de recursos utilizados. Como consecuencia, la frecuencia máxima de operación sufre una importante degradación, que es más significativa cuando se requiere el uso de LUT para implementar los multiplexores de la ROM. Sin embargo, incluso cuando éstos no son necesarios, se observa una reducción de velocidad debida exclusivamente al rutado de los bloques. Esto pone de manifiesto la necesidad de técnicas que reduzcan los recursos necesarios y aumenten la velocidad de las implementaciones basadas en ROM de FSM con muchas entradas o estados.

### 1.2.5. Arquitectura de referencia para implementaciones basadas en memoria en dispositivos FPGA

Tal como se ha mencionado anteriormente, debido a que los bloques de memoria empotrados disponibles en los dispositivos FPGA son síncronos, la arquitectura utilizada en el estudio experimental para las implementaciones basadas en memoria de FSM ha sido la correspondiente a las máquinas de Mealy con salidas sincronizadas (figura 1.8c). Esta arquitectura de referencia será denominada FSM-ROM.

Descartar el resto de arquitecturas para el estudio experimental no supone ninguna limitación importante. En primer lugar, tal como se estudió en la sección 1.2.1, las FSM de Mealy con salidas sincronizadas presentan las siguientes ventajas frente a las FSM de Mealy con salidas asíncronas: la señal de salida está libre de *gliches* y son máquinas más fáciles de diseñar y depurar [Govindarajalu, 2010]. En segundo lugar, siempre es posible ob-

tener una FSM de Mealy con salidas sincronizadas equivalente a una FSM de Moore. El procedimiento consiste en crear una FSM de Mealy con los mismos estados y transiciones que los de la FSM de Moore. Las salidas asociadas a cada transición de la FSM de Mealy son las salidas asociadas al estado destino de la transición correspondiente en la FSM de Moore. La figura 1.11 muestra un ejemplo de este procedimiento. Sin embargo, a pesar de que la arquitectura mostrada en figura 1.8c permite implementar FSM de Moore, la implementación correspondiente será más ineficiente debido a que requiere una ROM de mayor tamaño, tal como se puede comprobar si se compara la ecuación 1.8 con la ecuación 1.9.




**(a)** **(b)**

**Figura 1.11:** Ejemplo de conversión de una máquina de Moore en una máquina de Mealy equivalente: (a) Máquina de Moore y (b) máquina de Mealy.

Capítulo 2

# Máquinas de estados finitos con multiplexión de entradas

Tal como se ha estudiado en el capítulo anterior, la arquitectura convencional para las implementaciones basadas en memoria de FSM presenta varios inconvenientes. Por una parte, el tamaño de la ROM crece exponencialmente con el número de entradas de la FSM y, por otra, las entradas indeterminadas dan lugar a la aparición de información redundante en la ROM. En este capítulo se estudiarán un conjunto de técnicas que aprovechan la existencia de entradas indeterminadas para reducir tanto el número de entradas que forman parte de los bits de direcciones de la ROM como la cantidad de información redundante que ésta almacena.

En aquellas FSM en las que cada estado responde a un subconjunto de las entradas, se puede reducir el número de entradas que atacan a la ROM utilizando la arquitectura mostrada en la figura 2.1a. Esta arquitectura, inspirada en la técnica descrita en [Baranov, 1994], dispone de un banco de multiplexores que permite seleccionar un subconjunto de entradas diferentes para cada estado (compuesto por las entradas a las que el estado es sensible). El número de entradas que atacan a la ROM (el valor de $r$ en la figura 2.1a) viene determinado por el subconjunto de entradas con mayor cardinalidad. En aquellos casos en los que este valor es menor que el número de entradas totales de la FSM, esta arquitectura consigue reducir la profundidad de la ROM: reduce el valor de $m$ en las ecuaciones 1.8 y 1.9.

Puesto que la selección depende del estado, el banco de multiplexores está controlado por los bits de codificación de estado. Por lo tanto, esta arquitectura no requiere ampliar la información contenida en la ROM (es

decir, no requiere aumentar el ancho de palabra de la ROM). Sin embargo, esta técnica es efectiva solamente si ningún estado es sensible a todas las entradas de la FSM, ya que, en caso contrario, el banco de multiplexores no podrá reducir el número de entradas que atacan a la ROM.

Con el fin de evitar este inconveniente y para aumentar la capacidad de reducción del tamaño de la ROM, en [Senhadji-Navarro et al., 2004] propusimos una arquitectura alternativa en la que el banco de multiplexores está controlado por un conjunto de bits específicos almacenados en la ROM (véase la figura 2.1b). Esta arquitectura implementa un nuevo modelo de FSM, denominado *Máquina de Estados Finitos con Multiplexión de Entradas* (FSMIM, sigla del inglés Finite State Machine with Input Multiplexing), que fue presentado en [García-Vargas, 2006]. Aunque este esquema requiere aumentar el ancho de la ROM, permite la aplicación de las técnicas de optimización de FSMIM presentadas en [Senhadji-Navarro et al., 2004;

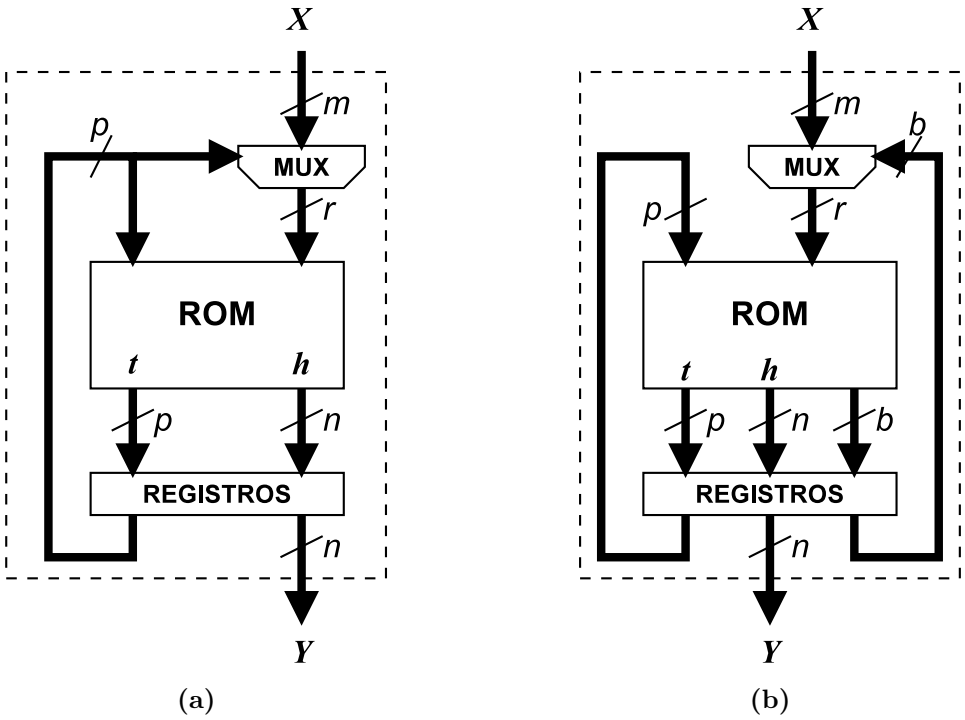



**Figura 2.1:** Arquitecturas de FSM basadas en memoria y en multiplexión de entradas: (a) multiplexores controlados por los bits de codificación de estado, (b) multiplexores controlados por bits de control específicos almacenados en ROM.

García-Vargas et al., 2007; García-Vargas y Senhadji-Navarro, 2015] y de las nuevas técnicas propuestas en esta tesis.

Por una parte, estas técnicas, además de reducir el valor de $m$ en las ecuaciones 1.8 y 1.9, también consiguen disminuir el valor de $|E|$. Esto permite mejorar los resultados, consiguiendo reducir el tamaño de la ROM incluso en casos en los que la arquitectura mostrada en la figura 2.1a no es efectiva (es decir, en los que la multiplexión no permite reducir el valor de $m$). Por otra parte, consiguen simplificar el banco de multiplexores, mejorando así las prestaciones de las implementaciones de FSMIM. Además, con el fin de solucionar el problema del aumento del ancho de la ROM, se propone en esta tesis una nueva arquitectura de FSMIM que no almacena en la ROM los bits de control de los multiplexores [García-Vargas y Senhadji-Navarro, 2015].

## 2.1. Definición de una máquina de estados finitos con multiplexión de entradas

El modelo de FSMIM propuesto surge como una generalización de la arquitectura mostrada en la figura 2.1b. La diferencia fundamental con el modelo de FSM convencional radica en que las transiciones de las FSMIM, además de determinar el estado siguiente y el valor de las salidas, seleccionan las entradas o los valores de las entradas a los que va a responder la máquina en el estado siguiente.

**Definición 2.1.** *Se define un* **conjunto de especificadores de $m$ entradas** *como el conjunto de símbolos $S_m = \{\lambda_1, \lambda_2, \ldots, \lambda_m\} \cup \{0_m, 1_m\}$, donde $0_m$ y $1_m$ se denominan* **constantes**. *Dado un especificador de entradas $\lambda \in S_m$ y una tupla $X = (x_1, \ldots, x_m) \in \mathbb{B}^m$, se define el* **operador de selección** *$X[\lambda] \in \mathbb{B}$ como*

$$X[\lambda] = \begin{cases} x_i & si \;\; \lambda = \lambda_i \\ 0 & si \;\; \lambda = 0_m \\ 1 & si \;\; \lambda = 1_m \end{cases}$$

*El conjunto $S_m^r \equiv (S_m)^r$ se denomina* **conjunto de selecciones de entradas de $r$ canales**, *y cada uno de sus elementos se denomina* **selección de entradas**. *Sea $s = (s_1, \ldots, s_r) \in S_m^r$ una selección de entradas, la posición (o índice) de cada componente de $s$ es un* **canal de selección** *y sus valores son* **entradas seleccionadas**.

*Dado $X \in \mathbb{B}^m$, el operador de selección se define para una selección de entradas $s$ como $X[s] = (X[s_1], \ldots, X[s_r])$.*

**Definición 2.2.** *Se define una FSMIM de $m$ entradas y $n$ salidas como la 5-tupla $F = (E, P, t, h, p_0)$, donde:*

- *$E$ es el conjunto finito de estados de la FSMIM.*
- *$P \subseteq E \times S_m^r$ es un conjunto finito de* **metaestados** *tal que para todo estado $e \in E$ existe al menos un metaestado $(e, s) \in P$. Cada metaestado $(e, s) \in P$ de la FSMIM se dice que está* **asociado al estado** *$e$.*
- *$t$ es la función de transición*

$$t : E \times \mathbb{B}^r \to P. \tag{2.1}$$

  *La función de transición de una FSMIM puede ser una función parcial.*

- *$h$ es la función de salida, que puede tomar la forma de la ecuación 2.2 en el caso de las FSMIM que son máquinas de Mealy (que serán denominadas* **FSMIM de Mealy***), o de la ecuación 2.3 en el caso de las que son máquinas de Moore (que serán denominadas* **FSMIM de Moore***):*

$$h : E \times \mathbb{B}^r \to \mathbb{B}^n, \tag{2.2}$$

$$h : P \to \mathbb{B}^n. \tag{2.3}$$

  *En el caso de la ecuación 2.2, la función de salida puede ser una función parcial.*

- *$p_0 = (e_0, s^{(0)}) \in P$ es el metaestado inicial, donde $e_0$ es el estado inicial y $s^{(0)}$ es la selección de entradas inicial.*

*Dado el conjunto de metaestados $P$, se define el* **conjunto de selecciones de entradas asociadas a un estado** *$e \in E$ como $P(e) = \{s : (e, s) \in P\}$. Por definición, $P(e) \neq \emptyset$. Si $|P(e)| = 1$ (es decir, con una única selección de entradas asociada), se denomina* **estado simple***; y si $|P(e)| > 1$ (es decir, si tiene varias selecciones de entradas), se denomina* **estado compuesto**.

Dado un valor de entrada $X = (x_1, \ldots, x_m) \in \mathbb{B}^m$ y un metaestado $(e, s) \in P$, el estado siguiente $e'$ y la selección de entradas siguiente $s'$ (es decir, el metaestado siguiente) vienen dados por la expresión $(e', s') = t(e, X[s])$. La salida viene dada por la expresión $h(e, X[s])$ en el caso de una FSMIM de Mealy, y por $h(e, s)$ en el caso de una FSMIM de Moore. Además, cada estado de una FSMIM puede tener asociadas diferentes selecciones de entradas, de las que depende la respuesta del estado ante

las entradas de la máquina. Por lo tanto, a diferencia de las FSM, en las que el comportamiento de la máquina en un instante determinado viene determinado por el estado presente y el valor de las entradas, en una FSMIM el comportamiento depende también de la selección de entradas presente.

**Definición 2.3.** *Sea* $F = (E, P, t, h, p_0)$ *una FSMIM de Mealy. Dado* $(e, s) \in P$*, la entrada seleccionada* $s_i$ *de la selección de entradas* $s = (s_1, \ldots, s_r)$ *es una* **Entrada Seleccionada Indeterminada** *(ESI) del estado* $e$ *si para todo* $X \in \mathbb{B}^m$*, se cumple que*

$$\begin{aligned} &t(e, (X[s_1], \ldots, X[s_{i-1}], 0, X[s_{i+1}], \ldots, X[s_r])) \\ &\quad = t(e, (X[s_1], \ldots, X[s_{i-1}], 1, X[s_{i+1}], \ldots, X[s_r])) \end{aligned} \tag{2.4}$$

*y*

$$\begin{aligned} &h(e, (X[s_1], \ldots, X[s_{i-1}], 0, X[s_{i+1}], \ldots, X[s_r])) \\ &\quad = h(e, (X[s_1], \ldots, X[s_{i-1}], 1, X[s_{i+1}], \ldots, X[s_r])). \end{aligned}$$

*En una FSMIM de Moore debe cumplirse la ecuación 2.4 y, además, debe cumplirse que* $h(e, s') = h(e, s)$ *para todo* $s' \in P(e)$ *tal que* $s_j = s'_j$ *para* $1 \leq j \leq r$ *y* $j \neq i$*.*

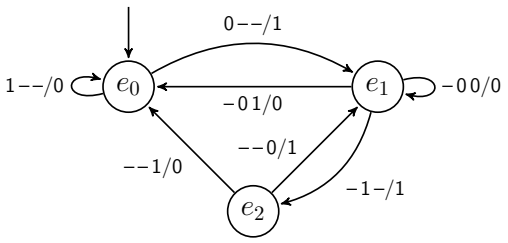
**Definición 2.4.** *Dada una FSM* $(E, t, h, e_0)$*, se dice que una entrada* $x_i$ *es una* **entrada efectiva** *del estado* $e \in E$ *si existe alguna transición de* $e$ *que dependa de* $x_i$ *(es decir, en la que* $x_i$ *no esté indeterminada). Por lo tanto, si una entrada no es efectiva para un estado, entonces estará indeterminada en todas sus transiciones.*

Por ejemplo, en la FSM de la figura 2.2, los estados $e_0$ y $e_2$ tienen una sola entrada efectiva ($x_1$ es efectiva para $e_0$, y $x_3$ para $e_2$). Por otra parte, las entradas efectivas del estado $e_1$ son $x_2$ y $x_3$, a pesar de que $x_2$ está indeterminada en la transición $(e_1, e_2)$.

Se dice que una FSMIM y una FSM son equivalentes si para cualquier secuencia de entradas, se obtiene en ambas la misma secuencia de salidas. A continuación se propone una definición formal.

**Definición 2.5.** *Dada un FSMIM* $F = (E,\ P,\ t,\ h,\ p_0)$ *y una FSM* $M = (E', t', h, e'_0)$*, se dice que ambas son* **equivalentes** *(representado* $F \sim M$*) sii para cualquier secuencia de entradas* $\langle X^{(0)}, X^{(1)}, \ldots, X^{(n)} \rangle$*, con* $X^{(i)} \in \mathbb{B}^m$*, se cumple que*

$$h(e^{(i)}, X^{(i)}[s^{(i)}]) = h'(e'^{(i)}, X^{(i)}) \quad \forall i = 0, 1, \ldots, n, \tag{2.5}$$




**(a)**

| **en** | | | **ep** | **es** | **sal** |
|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | | | $y$ |
| 0 | – | – | $e_0$ | $e_1$ | 1 |
| 1 | – | – | $e_0$ | $e_0$ | 0 |
| – | 0 | 0 | $e_1$ | $e_1$ | 0 |
| – | 0 | 1 | $e_1$ | $e_0$ | 0 |
| – | 1 | – | $e_1$ | $e_2$ | 1 |
| – | – | 0 | $e_2$ | $e_1$ | 1 |
| – | – | 1 | $e_2$ | $e_0$ | 0 |

**(b)**

**Figura 2.2:** Ejemplo de FSM con indeterminaciones en las entradas: (a) DTE, (b) TTE

*en el caso de las FSMIM de Mealy, o que*

$$h(e^{(i)}, s^{(i)}) = h'(e'^{(i)}) \quad \forall i = 0, 1, \ldots, n, \tag{2.6}$$

*en el caso de las FSMIM de Moore; donde*

$$\begin{aligned} (e^{(i)}, s^{(i)}) &= \begin{cases} p_0 & si\ i = 0 \\ t(e^{(i-1)}, X^{(i)}[s^{(i-1)}]) & si\ i > 1 \end{cases}, \\ e'^{(i)} &= \begin{cases} e'_0 & si\ i = 0 \\ t'(e'^{(i-1)}, X^{(i-1)}) & si\ i > 1 \end{cases}. \end{aligned} \tag{2.7}$$

Un procedimiento para obtener una FSMIM $F^* = (E,\ P,\ t^*,\ h^*,\ p_0)$ equivalente a una FSM $F = (E, t, h, e_0)$ de $m$ entradas, que será denominado *conversión trivial*, se describe a continuación:

1. Sea $S_m$ un conjunto de especificadores de $m$ entradas. Sea $\mathrm{EE}(e_i) \subset S_m$ la selección de entradas efectivas del estado $e_i \in E$, que se define como

   $$\mathrm{EE}(e_i) = \{\lambda_j \in S_m : x_j \text{ es una entrada efectiva del estado } e_i\}.$$

   El número de canales de la FSMIM es igual a $r = \mathrm{máx}_{e_i \in E} |\mathrm{EE}(e_i)|$.

2. Para cada estado $e_i \in E$ se define una selección de entradas $s^{(i)} = (s_1^{(i)}, \ldots, s_r^{(i)})$ donde $\{s_1^{(i)}, \ldots, s_r^{(i)}\}$ es una combinación sin repetición de $S_m \setminus \{0_m, 1_m\}$ tal que $\mathrm{EE}(e_i) \subseteq \{s_1^{(i)}, \ldots, s_r^{(i)}\}$.

3. $P = \{(e_i, s^{(i)}) : e_i \in E\}$.

4. Para todo $X \in \mathbb{B}^m$ y todo $e_i \in E$, se define $t^*(e_i, X[s^{(i)}]) = (e_j, s^{(j)})$, donde $e_j = t(e_i, X)$.

5. En el caso de una FSM de Mealy, para todo $X \in \mathbb{B}^m$ y todo $(e, s) \in P$, se define $h^*(e, X[s]) = h(e, X)$.

6. En el caso de una FSM de Moore, para todo $(e, s) \in P$ se define $h^*(e, s) = h(e)$.

El procedimiento de conversión trivial establece una relación biyectiva entre los estados de la FSM y los metaestados de la FSMIM.

**Proposición 2.6.** *Dada una FSM $F = (E, t, h, e_0)$, y una FSMIM $F^* = (E, P, t^*, h^*, p_0)$ obtenida a partir de $F$ mediante la conversión trivial, se cumple que $F \sim F^*$.*

***Demostración.*** En primer lugar, es necesario demostrar que las funciones $t^*$ y $h^*$ están bien definidas para todo $(e_i, s^{(i)}) \in P$. Es decir, que cumplen la condición de unicidad. Para una máquina de Moore, es trivial comprobar que la función $h^*$ cumple la condición de unicidad; por lo tanto, se demostrará sólo para las máquinas de Mealy.

Para todo $X, Y \in \mathbb{B}^m$ tal que $X \neq Y$, si $X[s^{(i)}] = Y[s^{(i)}]$ entonces $X$ e $Y$ se diferencian sólo en el valor de las entradas no seleccionadas, que por construcción son entradas indeterminadas del estado $e_i$. Por definición, esto implica que $t(e_i, X) = t(e_i, Y)$ y que $h(e_i, X) = h(e_i, Y)$. Por lo tanto, por una parte, $t^*(e_i, X[s^{(i)}]) = t^*(e_i, Y[s^{(i)}]) = (e_j, s^{(j)})$, donde $e_j = t(e_i, X)$; y por otra, $h^*(e_i, X[s^{(i)}]) = h^*(e_i, Y[s^{(i)}])$.

Una vez demostrada la unicidad de las funciones de transición y salida, es fácil comprobar que $F$ y $F^*$ son equivalentes, ya que, por construcción de la FSMIM, para todo estado $e_i \in E$ existe un metaestado $(e_i, s^{(i)}) \in P$ tal que para todo $X \in \mathbb{B}^m$ se cumple la ecuación 2.5 y la ecuación 2.7 para las máquinas de Mealy; o la ecuación 2.6 y la ecuación 2.7 para las máquinas de Moore. □

En las FSMIM obtenidas a partir del procedimiento de conversión trivial, el número de canales de la FSMIM (es decir, el número de entradas que se seleccionan) viene dado por el estado con el máximo número de entradas efectivas. Como consecuencia, para los estados que no alcanzan este máximo, parte de las entradas seleccionadas no son efectivas, sino indeterminadas, es decir, son ESI; el resto de entradas seleccionadas son las entradas efectivas del estado.

### 2.1.1. Representación de las FSMIM

Partiendo de las TTE utilizadas para representar las FSM, se ha definido un nuevo esquema de representación para las FSMIM, denominado *Tabla de Transición de Estados Extendida* (TTEE), que permite mostrar toda la información relativa a las entradas seleccionadas. Básicamente, una TTEE es una tabla de transición de estados simbólica a la que se ha añadido información sobre la selección de entradas.

Cada fila de la TTEE corresponde a una transición de la máquina de estados y contiene una 6-tupla $(sep, en, ep, ses, es, sal)$, donde $en$ y $sal$ son los valores de las entradas y salidas, respectivamente; $ep$ y $es$ son el estado presente y siguiente, respectivamente; y, finalmente, $sep$ y $ses$ son la selección de entradas presente[1] y siguiente, respectivamente.

En cada transición, $sep$ identifica a las entradas seleccionadas para el estado presente y $en$ contiene los valores de dichas entradas. Por otra parte, la información sobre las entradas que serán seleccionadas para el estado siguiente se encuentran en $ses$.

En una FSMIM con $r$ entradas seleccionadas, tanto $sep$ como $ses$ son $r$-tuplas $(se_0, se_1, \ldots, se_r)$ en la que $se_i$ identifica la entrada que se selecciona en la posición $i$ (es decir, en el canal de selección $i$). Las indeterminaciones se identifican con el símbolo '$-$'. Por motivos de claridad, las selecciones se representarán entre signos $\langle \ldots \rangle$.

La figura 2.3 muestra la TTEE correspondiente a la FSM de la figura 2.2. Las entradas seleccionadas para el estado $e_1$ son $\langle x_2, x_3 \rangle$, que se corresponden con las entradas efectivas de dicho estado. Por otra parte, la selección de entradas del estado $e_0$ es $\langle x_1, - \rangle$. Este estado tiene una sola entrada efectiva; por lo tanto, el resto de entradas seleccionadas serán indeterminadas.

[1] Siempre que no exista ambigüedad, la selección de entradas presente de un estado será denominada, simplemente, selección de entradas del estado

| **sep** | | **en** | | **ep** | **ses** | | **es** | **sal** |
|---|---|---|---|---|---|---|---|---|
| $sep_1$ | $sep_2$ | $en_1$ | $en_2$ | | $ses_1$ | $ses_2$ | | $y$ |
| $x_1$ | $-$ | 0 | $-$ | $e_0$ | $x_2$ | $x_3$ | $e_1$ | 1 |
| $x_1$ | $-$ | 1 | $-$ | $e_0$ | $x_1$ | $-$ | $e_0$ | 0 |
| $x_2$ | $x_3$ | 0 | 0 | $e_1$ | $x_2$ | $x_3$ | $e_1$ | 0 |
| $x_2$ | $x_3$ | 0 | 1 | $e_1$ | $x_1$ | $-$ | $e_0$ | 0 |
| $x_2$ | $x_3$ | 1 | $-$ | $e_1$ | $x_3$ | $-$ | $e_2$ | 1 |
| $x_3$ | $-$ | 0 | $-$ | $e_2$ | $x_2$ | $x_3$ | $e_1$ | 1 |
| $x_3$ | $-$ | 1 | $-$ | $e_2$ | $x_1$ | $-$ | $e_0$ | 0 |

**Figura 2.3:** Ejemplo de TTEE

## 2.2. Transformaciones elementales de una FSMIM

Dos FSMIM son equivalentes si para cualquier secuencia de entradas se obtiene la misma salida en ambas máquinas, independientemente del número de estados, de las selecciones de entradas y de sus funciones de transición y salida.

**Definición 2.7.** *Dos FSMIM de $m$ entradas $F = (E,\ P,\ t,\ h,\ p_0)$ y $F' = (E',\ P',\ t',\ g',\ p'_0)$ son equivalentes (denotado por $F \sim F'$) sii para cualquier secuencia de entradas $\langle X^{(0)}, X^{(1)}, \ldots, X^{(n)} \rangle$, con $X^{(i)} \in \mathbb{B}^m$, se cumple*

$$h(e^{(i)}, X^{(i)}[s^{(i)}]) = h'(e'^{(i)}, X^{(i)}[s'^{(i)}]) \quad \forall i = 0, 1, \ldots, n, \tag{2.8}$$

*en el caso de una FSMIM de Mealy, o*

$$h(e^{(i)}, s^{(i)}) = h'(e'^{(i)}, s'^{(i)}) \quad \forall i = 0, 1, \ldots, n, \tag{2.9}$$

*en el caso de una FSMIM de Moore; donde*

$$\begin{aligned} (e^{(i)}, s^{(i)}) &= \begin{cases} p_0 & si\ i = 0 \\ t(e^{(i-1)}, X^{(i-1)}[s^{(i-1)}]) & si\ i > 1 \end{cases}, \\ (e'^{(i)}, s'^{(i)}) &= \begin{cases} p'_0 & si\ i = 0 \\ t'(e'^{(i-1)}, X^{(i-1)}[s'^{(i-1)}]) & si\ i > 1 \end{cases}. \end{aligned} \tag{2.10}$$

Sobre las FSMIM se pueden realizar una serie de transformaciones que dan como resultado otra FSMIM equivalente. A continuación se estudiarán un conjunto de transformaciones elementales.

### 2.2.1. Agrupación de estados seudoequivalentes

**Definición 2.8.** *Sea $F = (E, P, t, h, p_0)$ una FSMIM de Mealy. Los estados $e, e' \in E$ son* **seudoequivalentes** *(denotado por $e \simeq e'$) sii para todo $X, Y \in \mathbb{B}^m$, todo $s \in P(e)$ y todo $s' \in P(e')$, tales que $X[s] = Y[s']$, se cumple*

$$t(e, X[s]) = t(e', Y[s']) \tag{2.11}$$

*y*

$$h(e, X[s]) = h(e', Y[s']). \tag{2.12}$$

*En el caso de una FSMIM de Moore, la ecuación 2.12 no es aplicable; en su lugar debe cumplirse la condición*

$$h(e, s) = h(e', s') \tag{2.13}$$

*para todo $s \in P(e)$ y todo $s' \in P(e')$, tales que $s = s'$.*

De manera intuitiva, la agrupación de dos estados seudoequivalentes $a$ y $b$ consiste en sustituir ambos estados por un nuevo estado $c$ (obviamente, el estado $c$ debe contener todas las transiciones de los estados $a$ y $b$). En una FSMIM, esto equivale a agrupar en el estado $c$ todos los metaestados asociados a los estados $a$ y $b$ (es decir, en la nueva FSMIM que se obtiene, todos los metaestados de $a$ y $b$ estarán asociados al estado $c$). La FSMIM obtenida es equivalente a la original. A continuación se proporciona una definición formal.

**Definición 2.9.** *Sea $F = (E, P, t, h, p_0)$ una FSMIM con dos estados $a, b \in E$ seudoequivalentes ($a \simeq b$). La agrupación de los estados $a$ y $b$ (representado como $a \oplus b$) genera una nueva FSMIM $F_{a\oplus b} = (E', P', t', g', p'_0)$ con las siguientes propiedades:*

(a) *Sea $\rho_{a\oplus b} : E \to E'$ una función definida como*

$$\rho_{a\oplus b}(e) = \begin{cases} a \oplus b & \text{si } e \in \{a, b\} \\ e & \text{en otro caso} \end{cases} .$$

(b) $E' = (E \setminus \{a, b\}) \cup \{a \oplus b\}$.

(c) $P' = \{(\rho_{a\oplus b}(e), s) : (e, s) \in P\}$.
*Obviamente, $|P'| < |P|$ sii $P(a) \cap P(b) \neq \emptyset$.*

(d) $p'_0 = (\rho_{a\oplus b}(e_0), s^{(0)})$, *donde* $(e_0, s^{(0)}) = p_0$.

(e) *Para todo* $X \in \mathbb{B}^m$ *y todo* $(e,s) \in P$, $t'(\rho_{a\oplus b}(e), X[s]) = (\rho_{a\oplus b}(c), u)$ *donde* $(c,u) = t(e, X[s])$.

(f) *En el caso de una FSMIM de Mealy,* $h'(\rho_{a\oplus b}(e), X[s]) = h(e, X[s])$ *para todo* $X \in \mathbb{B}^m$ *y todo* $(e,s) \in P$.

(g) *En el caso de una FSMIM de Moore,* $h'(\rho_{a\oplus b}(e), s) = h(e,s)$ *para todo* $(e,s) \in P$.

Resulta fácil comprobar a partir de la Definición 2.8 que, debido a que $a \simeq b$, las funciones $t'$ y $h'$ cumplen la condición de unicidad para todo $(a \oplus b, s) \in P'$. Es decir, que para todo $X \in \mathbb{B}^m$ se cumple que el valor de $t'(a \oplus b, X[s])$ es único y que también lo es el valor de $h'(a \oplus b, X[s])$, en el caso de una máquina de Mealy, o de $h'(a \oplus b, s)$, en el caso de una máquina de Moore. Por tanto, se puede concluir que la FSMIM $F_{a\oplus b}$ está bien construida.

A continuación se demostrará que la agrupación de estados seudoequivalentes da lugar a una FSMIM equivalente a la original. Sin embargo, en primer lugar, debe demostrarse el siguiente lema.

**Lema 2.10.** *Dada una FSMIM* $F = (E, P, t, h, p_0)$, *sea* $F_{a\oplus b} = (E', P', t', h', p'_0)$, *donde* $a$ *y* $b$ *son dos estados seudoequivalentes de* $F$. *Para cualquier secuencia de entradas* $\langle X^{(0)}, X^{(1)}, \ldots, X^{(n)}\rangle$, *con* $X^{(i)} \in \mathbb{B}^m$, *se cumple que* $(e'^{(i)}, s'^{(i)}) = (\rho_{a\oplus b}(e^{(i)}), s^{(i)})$ *para todo* $i = 0, \ldots, n$, *donde*

$$(e^{(i)}, s^{(i)}) = \begin{cases} p_0 & si\ i = 0 \\ t(e^{(i-1)}, X^{(i-1)}[s^{(i-1)}]) & si\ i > 1 \end{cases},$$

$$(e'^{(i)}, s'^{(i)}) = \begin{cases} p'_0 & si\ i = 0 \\ t'(e'^{(i-1)}, X^{(i-1)}[s'^{(i-1)}]) & si\ i > 1 \end{cases}.$$

***Demostración***. La demostración por inducción es trivial a partir de las propiedades (d) y (e) de la Definición 2.9. Supóngase, que $(e'^{(i)}, s'^{(i)}) = (\rho_{a\oplus b}(e^{(i)}), s^{(i)})$. Por (e) se tiene que $(e'^{(i+1)}, s'^{(i+1)}) = t'(e'^{(i)}, X^{(i)}[s'^{(i)}]) = t'(\rho_{a\oplus b}(e^{(i)}), X^{(i)}[s^{(i)}]) = (\rho_{a\oplus b}(c), u)$, donde $(c,u) = t(e^{(i)}, X^{(i)}[s^{(i)}]) = (e^{(i+1)}, s^{(i+1)})$. Por tanto, $(e'^{(i+1)}, s'^{(i+1)}) = (\rho_{a\oplus b}(e^{(i+1)}), s^{(i+1)})$. El caso para $i = 0$ se implica directamente de (d). □

**Proposición 2.11.** *Sea* $F = (E, P, t, h, p_0)$ *una FSMIM. Si* $a, b \in E$ *son estados seudoequivalentes, entonces* $F \sim F_{a\oplus b}$.

***Demostración.*** Sea $F_{a\oplus b} = (E', P', t', h', p'_0)$. Sea la secuencia de entradas $\langle X^{(0)}, X^{(1)}, \ldots, X^{(n)}\rangle$, con $X^{(i)} \in \mathbb{B}^m$, que da lugar a las secuencias de metaestados

$$(e^{(i)}, s^{(i)}) = \begin{cases} p_0 & \text{para } i = 0 \\ t(e^{(i-1)}, X^{(i-1)}[s^{(i-1)}]) & \text{para } i = 1, \ldots, n \end{cases},$$

$$(e'^{(i)}, s'^{(i)}) = \begin{cases} p'_0 & \text{para } i = 0 \\ t'(e'^{(i-1)}, X^{(i-1)}[s'^{(i-1)}]) & \text{para } i = 1, \ldots, n \end{cases}.$$

Por el Lema 2.10, se puede comprobar fácilmente que se cumplen las siguientes afirmaciones para todo $i = 0, \ldots, n$:

- En una máquina de Mealy,

  $$h'(e'^{(i)}, X^{(i)}[s'^{(i)}]) = h'(\rho_{a\oplus b}(e^{(i)}), X^{(i)}[s^{(i)}]),$$

  que es igual a $h(e^{(i)}, X^{(i)}[s^{(i)}])$ por la propiedad (f) de la Definición 2.9; quedando demostrada la condición de equivalencia de la ecuación 2.8.

- En una máquina de Moore, $h'(e'^{(i)}, s'^{(i)}) = h'(\rho_{a\oplus b}(e^{(i)}), s^{(i)})$, que es igual a $h(e^{(i)}, s^{(i)})$ por la propiedad (g) de la Definición 2.9; quedando demostrada la condición de equivalencia de la ecuación 2.9.

□

Los estados $e_0$ y $e_2$ de la FSMIM de la figura 2.3 son seudoequivalentes. Como puede observarse, para cualquier valor de las entradas seleccionadas (véanse las columnas $en_1$ y $en_2$), generan la misma salida y el mismo metaestado siguiente (véase la columna *sal* para la salida y las columnas *ses* y *es* para el metaestado siguiente). Por lo tanto, ambos estados cumplen las condiciones de la ecuación 2.11 y de la ecuación 2.12. Sea $e_{02} \equiv e_0 \oplus e_2$, la FSMIM que resulta de la agrupación se muestra en la figura 2.4. Los metaestados $(e_0, \langle x_1, -\rangle)$ y $(e_2, \langle x_3, -\rangle)$ de la máquina original están representados ahora por los metaestados $(e_{02}, \langle x_1, -\rangle)$ y $(e_{02}, \langle x_3, -\rangle)$, respectivamente; por lo tanto, ambos están asociados ahora al estado $e_{02}$. Este cambio afecta también a las transiciones originales cuyos estados siguientes eran $e_0$ y $e_2$ (véanse los campos *ses* y *es* de la TTEE del ejemplo).

### 2.2.2. Asignación de indeterminaciones

La *asignación de indeterminaciones* consiste en la selección de valores constantes $\{0, 1\}$ para las ESI. Se dirá que se ha asignado la constante 0 o 1 a una

| **sep** | | **en** | | **ep** | **ses** | | **es** | **sal** |
|---|---|---|---|---|---|---|---|---|
| $sep_1$ | $sep_2$ | $en_1$ | $en_2$ | | $ses_1$ | $ses_2$ | | $y$ |
| $x_1$ | $-$ | 0 | $-$ | $e_{02}$ | $x_2$ | $x_3$ | $e_1$ | 1 |
| $x_1$ | $-$ | 1 | $-$ | $e_{02}$ | $x_1$ | $-$ | $e_{02}$ | 0 |
| $x_2$ | $x_3$ | 0 | 0 | $e_1$ | $x_2$ | $x_3$ | $e_1$ | 0 |
| $x_2$ | $x_3$ | 0 | 1 | $e_1$ | $x_1$ | $-$ | $e_{02}$ | 0 |
| $x_2$ | $x_3$ | 1 | $-$ | $e_1$ | $x_3$ | $-$ | $e_{02}$ | 1 |
| $x_3$ | $-$ | 0 | $-$ | $e_{02}$ | $x_2$ | $x_3$ | $e_1$ | 1 |
| $x_3$ | $-$ | 1 | $-$ | $e_{02}$ | $x_1$ | $-$ | $e_{02}$ | 0 |

**Figura 2.4:** Agrupación de estados seudoequivalentes en una FSMIM

ESI si se selecciona dicho valor para la entrada correspondiente. El resultado de esta operación es una FSMIM equivalente a la original, ya que, por definición, el valor de las ESI no afecta al comportamiento de las FSMIM. Sin embargo, la asignación de indeterminaciones se puede aprovechar para obtener estados seudoequivalentes.

**Definición 2.12.** *Sea $F = (E, P, t, h, p_0)$ una FSMIM con $m$ entradas y $r$ canales de selección. Sea $e \in E$ tal que $s_j$ es una ESI para todo $s = (s_1, \ldots, s_j, \ldots, s_r) \in P(e)$, y sea $e' \in E$ tal que $s'_j$ es una ESI para todo $s' = (s'_1, \ldots, s'_j, \ldots, s'_r) \in P(e')$, siendo $1 \leq j \leq r$. Se denomina* **asignación exclusiva de indeterminaciones** *a los estados $e$ y $e'$ en el canal $j$ a la asignación de $s_j = 0_m$ para todo $s = (s_1, \ldots, s_j, \ldots, s_r) \in P(e)$ y $s'_j = 1_m$ para todo $s' = (s'_1, \ldots, s'_j, \ldots, s'_r) \in P(e')$. Los estados $e$ y $e'$ obtenidos tras la asignación serán denominados* **estados disjuntos en el canal** *$j$.*

**Proposición 2.13.** *Sea $F = (E, P, t, h, p_0)$ una FSMIM con $m$ entradas y $r$ canales de selección. Si $e, e' \in E$ son estados disjuntos en el canal $j$ $(1 \leq j \leq r)$, entonces se cumple que*

(a) *$P(e) \cap P(e') = \emptyset$.*

(b) *$X[s] \neq X[s']$ para todo $X \in \mathbb{B}^m$, todo $s \in P(e)$ y todo $s' \in P(e')$.*

(c) *Los estados $e$ y $e'$ son seudoequivalentes.*

***Demostración.*** (a): se implica directamente de la definición de estados disjuntos, ya que las selecciones de $e$ y $e'$ se diferencian en la constante seleccionada para el canal $j$.

(b): Es fácil comprobar que $X[s] \neq X[s']$ para todo $X \in \mathbb{B}^m$, todo $s \in P(e)$ y todo $s' \in P(e')$, ya que

$$X[s] = (x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_r)$$

y

$$X[s'] = (x'_1, \ldots, x'_{i-1}, 1, x'_{i+1}, \ldots, x'_r).$$

(c): Debido a (b), en una FSMIM de Mealy no existe ningún valor de $s \in P(e)$, $s' \in P(e')$ y $X \in \mathbb{B}^m$ que incumplan las ecuaciones 2.11 y 2.12. De manera similar, debido a (a), en el caso de una FSMIM de Moore no existe ningún valor de $s$ y $s'$ que incumplan la ecuación 2.13. □

La asignación de indeterminaciones junto con la agrupación de estados pueden ser utilizadas para minimizar el número de estados de la FSMIM. Una de las técnicas que se presentan en esta tesis para reducir el tamaño de la ROM de las implementaciones FSMIM se basa en estas dos operaciones. Aunque será estudiada en profundidad más adelante, en esta sección se mostrará un pequeño ejemplo para ilustrar los mecanismos básicos en los que se fundamenta.

La figura 2.5a muestra la TTEE de una FSMIM con dos entradas y dos canales de selección, obtenida a partir de la FSM de la figura 1.9 mediante el procedimiento de conversión trivial. Como puede observarse, en esta FSMIM no existen estados seudoequivalentes. Por ejemplo, para el valor de entrada $(x_1, x_2) = (1, 1)$, las entradas seleccionadas de los metaestados $(e_0, \langle x_1, - \rangle)$ y $(e_2, \langle x_2, - \rangle)$ coinciden, ya que ambas toman el valor $(1, -)$ (tal como se puede comprobar observando las columnas *sep*, *en* y *ep* de las filas segunda y última de la figura 2.5a). Sin embargo, no coinciden ni los valores de la función de transición ni las salidas asociadas a dichos valores de entradas: para $(e_0, \langle x_1, - \rangle)$ el metaestado siguiente es $(e_1, \langle x_1, x_2 \rangle)$ y la salida vale 0, mientras que para $(e_2, \langle x_2, - \rangle)$ el metaestado siguiente es $(e_0, \langle x_1, - \rangle)$ y la salida vale 1. Por lo tanto, no son estados seudoequivalentes, ya que no cumplen la ecuación 2.11 ni la ecuación 2.12. Sin embargo, puesto que ambos estados comparten una ESI en el canal de selección $sep_2$, se pueden transformar en estados seudoequivalentes si se asigna, por ejemplo, la constante 0 a la ESI del estado $e_0$ y la constante 1 a la del estado $e_2$. Esto permite agrupar ambos estados. Sea el estado $e_{02} \equiv e_0 \oplus e_2$, se dice que los estados $e_0$ y $e_2$ han sido agrupados en el estado $e_{02}$.

La figura 2.5b muestra la TTEE después de realizar la agrupación de estados. Los símbolos '*0*' y '*1*' en los campos *sep* y *ses* de la tabla indican que se deben seleccionar las constantes 0 y 1, respectivamente, para la entrada correspondiente (son equivalentes a los símbolos $0_m$ y $1_m$ del conjunto de especificadores de entrada de la FSMIM). Por motivos de claridad, las transiciones del estado $e_{02}$ no aparecen consecutivas en la tabla, sino en las mismas filas en las que aparecían las transiciones de los estados originales.

El metaestado $(e_0, \langle x_1, - \rangle)$ ha sido sustituido por $(e_{02}, \langle x_1, 0 \rangle)$ en todas las transiciones (veánse los campos *sep*, *ep*, *ses* y *es*). Igualmente, el metaestado $(e_2, \langle x_2, - \rangle)$ ha sido sustituido por $(e_{02}, \langle x_2, 1 \rangle)$ en todas las transiciones. Puede observarse cómo el metaestado $(e_{02}, \langle x_1, 0 \rangle)$ contiene las transiciones del estado $e_0$ y el metaestado $(e_{02}, \langle x_2, 1 \rangle)$, las del estado $e_2$. Cuando el estado y la selección de entradas presentes sean $(e_{02}, \langle x_1, 0 \rangle)$, independientemente del valor de $x_2$, siempre se seleccionarán una de las transiciones del estado $e_0$ original (ya que la entrada seleccionada $en_2$ siempre valdrá 0). De forma similar, cuando el estado y la selección de entradas presentes sean $(e_{02}, \langle x_2, 1 \rangle)$, independientemente del valor de $x_1$, siempre se seleccionarán una de las transiciones del estado $e_2$ original (ya que la entrada seleccionada $en_2$ siempre valdrá 1). Como se puede observar, la operación de agrupación de estados seudoequivalentes mediante la asignación de indeterminaciones ha reducido el número de estados de la FSMIM, pero no ha cambiado el número de metaestados.

Sea $F = (E, t, h, e_0)$ una FSM. Sea $F^* = (E, P, t^*, h^*, p_0^*)$ una FSMIM obtenida a partir de $F$ por el procedimiento de conversión trivial. Sea $F^*_{a \simeq b} = (E, P_{a \simeq b}, t', h', p_0')$ una FSMIM obtenida a partir de $F^*$ en la que se ha realizado una asignación exclusiva de indeterminaciones a los estados $a, b \in E$ para que sean seudoequivalentes. Sea $F^*_{a \oplus b} = (E_{a \oplus b}, P_{a \oplus b}, t'', h'', p_0'')$ la FSMIM obtenida a partir de $F^*_{a \simeq b}$ agrupando los estados $a, b \in E$. Por la Proposición 2.13, en $F^*_{a \simeq b}$, $P_{a \simeq b}(a) \cap P_{a \simeq b}(b) = \emptyset$, entonces $|P_{a \oplus b}| = |P_{a \simeq b}| = |P|$. Por lo tanto, se cumple que $|E_{a \oplus b}| < |E|$ y $|P_{a \oplus b}| = |E|$, donde $E$ es el conjunto de estados de la FSM original $F$.

Por otra parte, en el ejemplo, el número de canales de la FSMIM es igual al número de entradas de la FSM. Por lo tanto, la técnica de multiplexión de entradas por sí sola (véase la figura 2.1) no consigue reducir el tamaño de la ROM, ya que el número de entradas que atacan a la ROM es el mismo que en la implementación convencional basada en memoria de esta FSM. Sin embargo, con la agrupación de estados se ha conseguido reducir el número de estados de tres a dos, lo que permite reducir la profundidad de la ROM de la arquitectura mostrada en la figura 2.1b.

### 2.2.3. Permutación de entradas

La permutación de entradas de un estado de la FSMIM consiste en la permutación de los elementos de la selección de entradas asociada al estado (la permutación de entradas implica también la modificación de las funciones de transición y salida). Esta operación equivale a cambiar el canal de selección de la FSMIM en el que cada entrada es seleccionada por el estado. Dado un

| **sep** | | **en** | | **ep** | **ses** | | **es** | **sal** |
|---|---|---|---|---|---|---|---|---|
| $sep_1$ | $sep_2$ | $en_1$ | $en_2$ | | $ses_1$ | $ses_2$ | | $y$ |
| $x_1$ | – | 0 | – | $e_0$ | $x_1$ | – | $e_0$ | 0 |
| $x_1$ | – | 1 | – | $e_0$ | $x_1$ | $x_2$ | $e_1$ | 0 |
| $x_1$ | $x_2$ | 0 | 0 | $e_1$ | $x_1$ | $x_2$ | $e_1$ | 0 |
| $x_1$ | $x_2$ | 1 | 0 | $e_1$ | $x_2$ | – | $e_2$ | 0 |
| $x_1$ | $x_2$ | – | 1 | $e_1$ | $x_1$ | – | $e_0$ | 1 |
| $x_2$ | – | 0 | – | $e_2$ | $x_2$ | – | $e_2$ | 0 |
| $x_2$ | – | 1 | – | $e_2$ | $x_1$ | – | $e_0$ | 1 |

**(a)**

| **sep** | | **en** | | **ep** | **ses** | | **es** | **sal** |
|---|---|---|---|---|---|---|---|---|
| $sep_1$ | $sep_2$ | $en_1$ | $en_2$ | | $ses_1$ | $ses_2$ | | $y$ |
| $x_1$ | *0* | 0 | 0 | $e_{02}$ | $x_1$ | *0* | $e_{02}$ | 0 |
| $x_1$ | *0* | 1 | 0 | $e_{02}$ | $x_1$ | $x_2$ | $e_1$ | 0 |
| $x_1$ | $x_2$ | 0 | 0 | $e_1$ | $x_1$ | $x_2$ | $e_1$ | 0 |
| $x_1$ | $x_2$ | 1 | 0 | $e_1$ | $x_2$ | *1* | $e_{02}$ | 0 |
| $x_1$ | $x_2$ | – | 1 | $e_1$ | $x_1$ | *0* | $e_{02}$ | 1 |
| $x_2$ | *1* | 0 | 1 | $e_{02}$ | $x_2$ | *1* | $e_{02}$ | 0 |
| $x_2$ | *1* | 1 | 1 | $e_{02}$ | $x_1$ | *0* | $e_{02}$ | 1 |

**(b)**

**Figura 2.5:** Agrupación de estados en una FSMIM: (a) TTEE sin agrupación de estados, (b) TTEE con un nuevo estado $e_{02} \equiv e_0 \oplus e_2$.

conjunto de selecciones de entradas de $r$ canales $S_m^r$, cada permutación será representada por una función $\pi : \{1, \ldots, r\} \to \{1, \ldots, r\}$.

**Definición 2.14.** *Sea $F = (E, P, t, h, p_0)$ una FSMIM con $m$ entradas. Sea $\pi : \{1, \ldots, r\} \to \{1, \ldots, r\}$ una permutación. Dado un estado simple $e$, la permutación de la selección de entradas del metaestado $(e, s) \in P$ genera una nueva FSMIM $F_{\pi,(e,s)} = (E, P', t', g', p_0')$ con las siguientes propiedades:*

(a) *Sea $\rho_{\pi,(e,s)} : P \to S_m^r$ una función definida como*

$$\rho_{\pi,(e,s)}(d, u) = \begin{cases} (s_{\pi(1)}, \ldots, s_{\pi(r)}) & \text{si } (e, s) = (d, u), \\ u & \text{en otro caso.} \end{cases}$$

(b) $P' = (P \setminus \{(e, s)\}) \cup \{(e, \langle s_{\pi(1)}, \ldots, s_{\pi(r)} \rangle)\}$.

(c) $p_0' = (e_0, \rho_{\pi,(e,s)}(p_0))$, *donde* $(e_0, s^{(0)}) = p_0$.

(d) *Para todo* $X \in \mathbb{B}^m$ *y todo* $(d,u) \in P$, $t'(d, X[\rho_{\pi,(e,s)}(d,u)]) = (c, \rho_{\pi,(e,s)}(c,v))$ *donde* $(c,v) = t(d, X[u])$.

(e) *En el caso de una FSMIM de Mealy,* $h'(d, X[\rho_{\pi,(e,s)}(d,u)]) = h(d,u)$ *para todo* $X \in \mathbb{B}^m$ *y todo* $(d,u) \in P$.

(f) *En el caso de una FSMIM de Moore,* $h'(e, \rho_{\pi,(e,s)}(d,u)) = h(d,u)$ *para todo* $(d,u) \in P$.*:*

El estado $e$ debe cumplir la condición de ser un estado simple para garantizar que las funciones $t'$ y $h'$ cumplen la condición de unicidad para el metaestado $(e, \rho_{\pi,(e,s)}(e,s)) \in P'$. Es decir, que para todo $X \in \mathbb{B}^m$ se cumple que es único tanto el valor de $t'(e, X[\rho_{\pi,(e,s)}(e,s)])$ como el valor de $h'(e, X[\rho_{\pi,(e,s)}(e,s)])$, en el caso de una máquina de Mealy, o de $h'(e, \rho_{\pi,(e,s)}(e,s))$, en el caso de una máquina de Moore.

**Lema 2.15.** *Dada una FSMIM* $F = (E, P, t, h, p_0)$, *sea* $F_{\pi,(e,s)} = (E', P', t', h', p'_0)$, *donde* $e$ *es un estado simple y* $\pi$ *una permutación. Para cualquier secuencia de entradas* $\langle X^{(0)}, X^{(1)}, \ldots, X^{(n)} \rangle$, *con* $X^{(i)} \in \mathbb{B}^m$, *se cumple que* $(e'^{(i)}, s'^{(i)}) = (e^{(i)}, \rho_{\pi,(e,s)}(e^{(i)}, s^{(i)}))$ *para todo* $i = 0, \ldots, n$, *donde*

$$(e^{(i)}, s^{(i)}) = \begin{cases} p_0 & si\ i = 0 \\ t(e^{(i-1)}, X^{(i-1)}[s^{(i-1)}]) & si\ i > 1 \end{cases},$$

$$(e'^{(i)}, s'^{(i)}) = \begin{cases} p'_0 & si\ i = 0 \\ t'(e'^{(i-1)}, X^{(i-1)}[s'^{(i-1)}]) & si\ i > 1 \end{cases}.$$

***Demostración.*** La demostración por inducción es trivial a partir de las propiedades (c) y (d) de la Definición 2.14. Supóngase, que $(e'^{(i)}, s'^{(i)}) = (e^{(i)}, \rho_{\pi,(e,s)}(e^{(i)}, s^{(i)}))$. Por (d) se tiene que $(e'^{(i+1)}, s'^{(i+1)}) = t'(e'^{(i)}, X^{(i)}[s'^{(i)}]) = t'(e^{(i)}, X^{(i)}[\rho_{\pi,(e,s)}(e^{(i)}, s^{(i)})]) = (c, \rho_{\pi,(e,s)}(c,v))$, donde $(c,v) = t(e^{(i)}, X^{(i)}[s^{(i)}]) = (e^{(i+1)}, s^{(i+1)})$. Por tanto, $(e'^{(i+1)}, s'^{(i+1)}) = (e^{(i+1)}, \rho_{\pi,(e,s)}(e^{(i+1)}, s^{(i+1)}))$. El caso para $i = 0$ se implica directamente de (c). □

**Proposición 2.16.** *Dada una FSMIM* $F = (E, P, t, h, p_0)$, *una permutación* $\pi$ *y un metaestado* $(e,s) \in P$, *siendo* $e$ *un estado simple, entonces* $F \sim F_{\pi,(e,s)}$.

***Demostración.*** Sea $F_{\pi,(e,s)} = (E', P', t', h', p'_0)$. Sea $[X^{(0)}, X^{(1)}, \ldots, X^{(n)}]$, con $X^{(i)} \in \mathbb{B}^m$, una secuencia de entradas que da lugar a las secuencias de

metaestados

$$(e^{(i)}, s^{(i)}) = \begin{cases} p_0 & \text{para } i = 0 \\ t(e^{(i-1)}, X^{(i-1)}[s^{(i-1)}]) & \text{para } i = 1, \ldots, n \end{cases},$$

$$(e'^{(i)}, s'^{(i)}) = \begin{cases} p'_0 & \text{para } i = 0 \\ t'(e'^{(i-1)}, X^{(i-1)}[s'^{(i-1)}]) & \text{para } i = 1, \ldots, n \end{cases}.$$

Por el Lema 2.15, se puede comprobar fácilmente que se cumplen las siguientes afirmaciones para todo $i = 0, \ldots, n$:

- En una máquina de Mealy,

  $$h'(e'^{(i)}, X^{(i)}[s'^{(i)}]) = h'(e^{(i)}, X^{(i)}[\rho_{\pi,(e,s)}(e^{(i)}, s^{(i)})]),$$

  que es igual a $h(e^{(i)}, X^{(i)}[s^{(i)}])$ por la propiedad (e) de la Definición 2.14; quedando demostrada la condición de equivalencia de la ecuación 2.8.

- En una máquina de Moore, $h'(e'^{(i)}, s'^{(i)}) = h'(e^{(i)}, \rho_{\pi,(e,s)}(e^{(i)}, s^{(i)}))$, que es igual a $h(e^{(i)}, s^{(i)})$ por la propiedad (f) de la Definición 2.14; quedando demostrada la condición de equivalencia de la ecuación 2.9.

□

En el contexto del diseño digital de FSMIM, la selección de entradas se implementa con elementos combinacionales. Cada canal de selección de la FSMIM define una función lógica diferente, ya que el subconjunto de entradas que selecciona (denominado *entradas del canal de selección*) no es el mismo en todos los canales. Por ejemplo, en la FSMIM de la figura 2.5b, las entradas del canal de selección $sep_1$ son $\{x_1, x_2\}$, mientras que las del canal de selección $sep_2$ son $\{x_2, 0, 1\}$.

La complejidad de la función lógica asociada a cada canal y, por tanto, la del bloque combinacional que la implementa, depende en parte del número de entradas del canal[2]. Como consecuencia, las funciones lógicas pueden ser simplificadas buscando un conjunto de permutaciones de entradas (una permutación para cada metaestado) que minimicen el número de entradas de cada canal de selección.

[2]Tal como se estudiará en secciones posteriores, dependiendo del tipo de arquitectura que se utilice para implementar la función de selección de entradas, pueden existir otros factores que influyen en la complejidad del bloque combinacional.

Por ejemplo, en las arquitecturas mostradas en la figura 2.1, la selección de entradas se realiza usando un banco de multiplexores, que está compuesto por un multiplexor diferente para cada canal de selección de la FSMIM. En este caso, puesto que la complejidad de un multiplexor depende principalmente del número de entradas, la influencia de las permutaciones de entradas en las prestaciones del banco de multiplexores (en términos de área y velocidad) es muy significativa. La figura 2.6 muestra diferentes ejemplos de permutaciones de entradas que se han realizado sobre una FSMIM obtenida a partir de la FSM de la figura 2.6a. En cada caso, se muestran solamente las columnas *sep* y *ep* de las TTEE correspondientes (sin filas repetidas). Cada ejemplo incluye, además, el banco de multiplexores requerido para implementar la selección de entradas correspondiente con la arquitectura mostrada en la figura 2.1b.

Como puede observarse, el peor resultado se obtiene con las permutaciones de la figura 2.6b, que producen una FSMIM en la que el canal de selección $sep_1$ tiene tres entradas. Este resultado mejora con las permutaciones mostradas en las figuras 2.6c y 2.6d. De estas dos, la segunda es la mejor opción, ya que también permitiría realizar la agrupación de los estados $e_0$ y $e_2$. Por otra parte, en la arquitectura de la figura 2.1b, además de simplificar el banco de multiplexores, se consigue reducir el ancho de la ROM, ya que la ROM contiene los bits de control que atacan a los multiplexores (el número de bits de control se reduce de dos a uno).

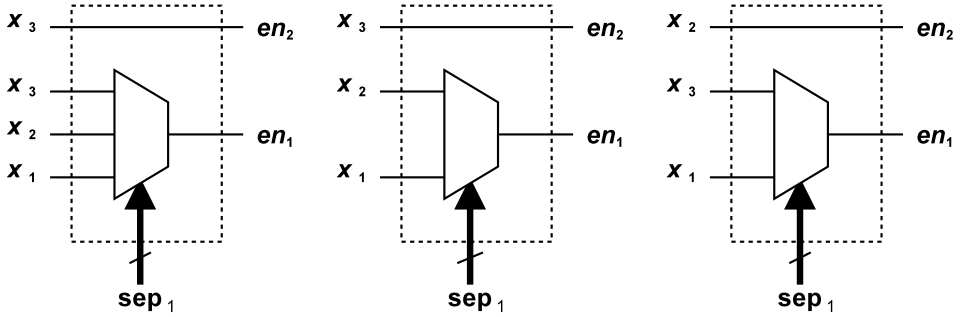
| **en** | | | **ep** | **es** | **sal** |
|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | | | $y$ |
| 0 | – | – | $s_0$ | $s_1$ | 1 |
| 1 | – | – | $s_0$ | $s_0$ | 0 |
| – | 0 | 0 | $s_1$ | $s_1$ | 0 |
| – | 0 | 1 | $s_1$ | $s_0$ | 0 |
| – | 1 | – | $s_1$ | $s_2$ | 1 |
| – | – | 0 | $s_2$ | $s_1$ | 1 |
| – | – | 1 | $s_2$ | $s_0$ | 0 |

(a)

| **sep** | | **ep** |
|---|---|---|
| $sep_1$ | $sep_2$ | |
| $x_1$ | – | $s_0$ |
| $x_2$ | $x_3$ | $s_1$ |
| $x_3$ | – | $s_2$ |

| **sep** | | **ep** |
|---|---|---|
| $sep_1$ | $sep_2$ | |
| $x_1$ | – | $s_0$ |
| $x_2$ | $x_3$ | $s_1$ |
| – | $x_3$ | $s_2$ |

| **sep** | | **ep** |
|---|---|---|
| $sep_1$ | $sep_2$ | |
| $x_1$ | – | $s_0$ |
| $x_3$ | $x_2$ | $s_1$ |
| $x_3$ | – | $s_2$ |




(b) (c) (d)

**Figura 2.6:** Efecto de la permutación de entradas sobre la complejidad de la función de selección de entradas: (a) ejemplo de FSM; (b), (c) y (d) diferentes selecciones de entradas junto con el banco de multiplexores correspondiente.

## 2.3. Arquitecturas para la implementación de FSMIM

La diferencia fundamental entre las arquitecturas que se presentan en esta tesis para la implementación de las FSMIM radica en la implementación de la función de transición (o más concretamente, en la forma de codificar los metaestados en la ROM). El tipo de codificación utilizada determina

la arquitectura interna del bloque combinacional que implementa la selección de entradas (denominado *banco de selectores de entradas*). Puede ser necesario, además, un bloque combinacional adicional para decodificar los metaestados. Ambos bloques serán denominados *modificadores de dirección* de la FSMIM.

Tal como se mostró en el capítulo anterior, el tamaño de la ROM de una FSM $F = (E, t, h, e_0)$ con $m$ entradas y $n$ salidas viene dado por la ecuación 1.8 para las máquinas de Mealy, y por la ecuación 1.9 para las máquinas de Moore.

En el caso de las máquinas de Mealy, la profundidad de la ROM viene dada por la expresión

$$2^m|E| \text{ palabras.}$$

Sea $F' = (E', P, t', h', p_0)$ una FSMIM con $r$ canales de selección, obtenida a partir de $F$. Independientemente de la arquitectura utilizada (véase la figura 2.1), la profundidad de la ROM para las FSMIM de Mealy es igual a

$$2^r|E'| \text{ palabras.}$$

Uno de los objetivos de las implementaciones de FSMIM es la reducción de la profundidad de la ROM[3]. Para conseguirlo, aprovechan el mecanismo de selección de entradas de dos maneras diferentes. Por una parte, la selección de las entradas efectivas de la FSM permite reducir la profundidad en aquellos casos en los que $r < m$. Por otra parte, la agrupación de estados permite aumentar la tasa de reducción, ya que $|E'| \leq |E|$, permitiendo reducir la profundidad incluso en casos en los que $r = m$ (de las dos arquitecturas mostradas en la figura 2.1, esto sólo es posible en la correspondiente a la figura 2.1b).

En las máquinas de Moore, la ROM que implementa la función de transición tiene las mismas características que la de las máquinas de Mealy respecto a la profundidad. Por lo tanto, se pueden aplicar los mecanismos anteriores.

### 2.3.1. Arquitectura FSMIM con selección de entradas basada en Transiciones

Tal como se mostró en la introducción del capítulo, en la primera arquitectura que se propuso para la implementación de FSMIM (véase figura 2.1b),

[3]Como se estudiará más adelante, el otro objetivo es reducir el consumo de recursos combinacionales necesarios para implementar la FSMIM.

el banco de selectores de entradas se implementa con un banco de multiplexores [Senhadji-Navarro et al., 2004]. En esta arquitectura, el banco de selectores de entradas (denominado *banco de multiplexores de entradas*) está controlado por un conjunto de bits específicos almacenados en la ROM (denominados *bits de selección*). Debido a que las transiciones almacenadas en la ROM incluyen los bits de selección de cada multiplexor, esta arquitectura será denominada *FSMIM con selección de entradas basada en transiciones* (FSMIM-T, sigla del inglés **FSMIM** with **T**ransition-based input selection).

Puesto que la función de selección de una FSMIM permite seleccionar valores constantes $\{0, 1\}$ además de las entradas de la FSM, el banco de multiplexores de entradas debe ser ampliado para que sea posible la selección de las señales 0 y 1. La figura 2.7 muestra la descripción de alto nivel de diferentes variantes de la arquitectura FSMIM-T, utilizadas para la implementación de máquinas de Mealy y de Moore con memoria asíncrona (figuras 2.7a y 2.7b, respectivamente) y de máquinas de Mealy sincronizadas (figura 2.7c).

En la arquitectura FSMIM-T los metaestados de la FSMIM se identifican codificando por separado todos sus componentes. Es decir, para cada metaestado, se codifican por separado tanto el estado como la selección de entradas asociada a cada canal de selección (los bits de selección son la codificación de la selección de entradas).

En el caso de una máquina de Mealy (figuras 2.7a y 2.7c), la dirección de acceso a la ROM viene determinada por los bits de codificación del estado presente y por el valor de las entradas seleccionadas por la selección de entradas presente. Esta dirección determina la palabra de la ROM que contiene el valor de la salida de la máquina (según la ecuación 2.2) y la codificación del metaestado siguiente (según la ecuación 2.1).

En el caso de una máquina de Moore (figura 2.7a), la salida es generada por una segunda ROM (según la ecuación 2.3), que es direccionada por la codificación del metaestado presente (es decir, por los bits de codificación del estado presente y por la codificación de la selección de entradas presente de cada canal de selección).

La figura 2.9 muestra un ejemplo de implementación de la arquitectura FSMIM-T. La TTEE de la FSMIM implementada se muestra en la figura 2.8 (que es una copia de la TTEE de la figura 2.5 con las filas reordenadas para facilitar la lectura del contenido de la ROM). Se trata de una máquina de Mealy. La figura 2.9b muestra una posible codificación de la selección de entradas del canal de selección 2; y las figuras 2.9d y 2.9e, dos posibles implementaciones del multiplexor de dicho canal. Para que la implementación de la figura 2.9d sea correcta, la indeterminación de la codificación de entradas

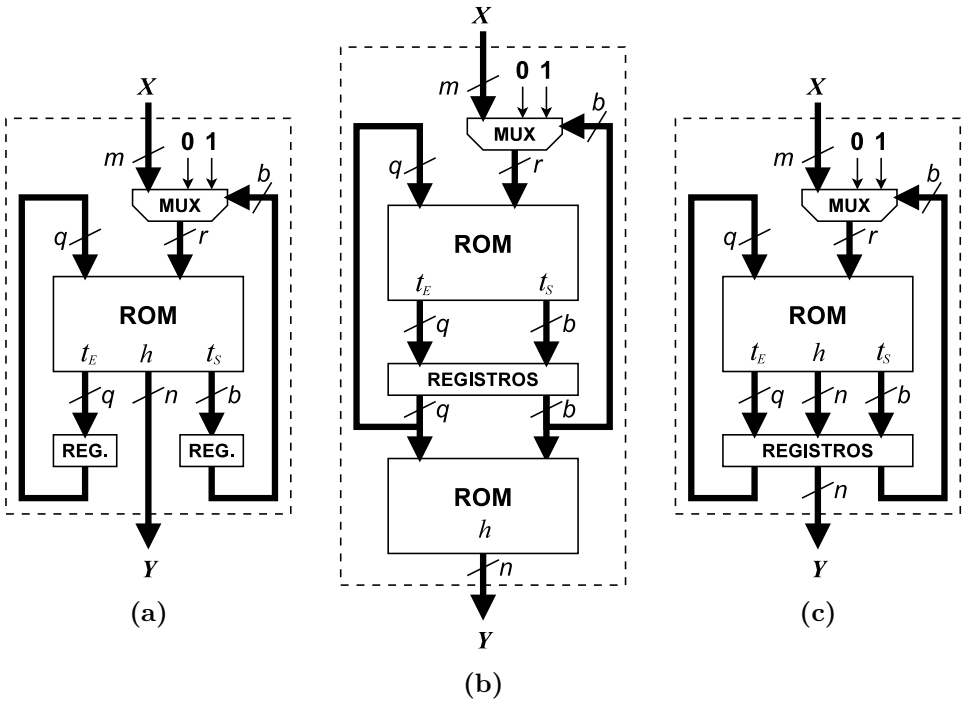



**Figura 2.7:** Arquitecturas para la implementación de FSMIM-T: (a) implementación basada en memoria asíncrona de una FSMIM de Mealy, (b) implementación basada en memoria asíncrona de una FSMIM de Moore y (c) implementación de una FSMIM de Mealy sincronizada. En la ROM, $t_E$ es el estado siguiente generado por la función de transición, y $t_S$, la selección de entradas siguientes.

debe sustituirse por el valor 0. La figura 2.9f muestra la TTEE codificada, en la que se han sustituidos todos los símbolos de la TTEE de la figura 2.8 por su correspondiente codificación. Finalmente, la figura 2.9g muestra el contenido de la ROM. Puede observarse como el ancho de la ROM crece debido a los bits de selección, aunque en este ejemplo, a causa de la extrema simplicidad de la FSM, la columnas de la ROM $sep_2^{(1)}$ y $sep_2^{(0)}$ pueden eliminarse (la columna $sep_2^{(1)}$ es equivalente a la columna $sep_1^{(0)}$, mientras que la columna $sep_2^{(0)}$ es igual que la columna *es*). Sin embargo, se ha optado por no eliminar las columnas redundantes para que sea patente la estructura de la información que contiene la ROM (esto permite observar la codificación de los metaestados en la arquitectura FSMIM-T). Por otra parte, en este ejemplo se implementa la misma FSM que en la figura 1.9. Si se compara

el contenido de ambas memorias, se observa cómo las transiciones repetidas en la implementación FSM-ROM han desaparecido.

#### 2.3.1.1. Prestaciones de la arquitectura FSMIM-T

Sea $F = (E, P, t, h, p_0)$ una FSMIM con $m$ entradas, $n$ salidas y $r$ canales de selección, equivalente a una FSM $F_{\text{FSM}} = (E', t', h', e'_0)$. El banco de multiplexores está formado por $r$ multiplexores como máximo[4], controlados por los bits de selección. Sea $m_i$ el número de entradas, incluyendo las constantes $\{0, 1\}$, del canal de selección $i$ $(1 \leq i \leq r)$. El número de bits de selección de la FSMIM-T es

$$b = \sum_{i=1}^{r} \lceil \log_2 m_i \rceil. \tag{2.14}$$

En una máquina de Mealy (arquitecturas mostradas en las figuras 2.7a y 2.7c), cada palabra de la ROM contiene los bits de salida $n$, los bits de codificación de estados de la FSMIM $q = \lceil \log_2 |E| \rceil$ y los bits de selección $b$; es decir, el ancho de la ROM es

$$n + q + b \quad \text{bits.} \tag{2.15}$$

La profundidad de la ROM es

$$2^r |E| \quad \text{palabras} \tag{2.16}$$

| **sep** | | **en** | | **ep** | **ses** | | **es** | **sal** |
|---|---|---|---|---|---|---|---|---|
| $sep_1$ | $sep_2$ | $en_1$ | $en_2$ | | $ses_1$ | $ses_2$ | | $y$ |
| $x_1$ | *0* | 0 | 0 | $e_{02}$ | $x_1$ | *0* | $e_{02}$ | 0 |
| $x_2$ | *1* | 0 | 1 | $e_{02}$ | $x_2$ | *1* | $e_{02}$ | 0 |
| $x_1$ | *0* | 1 | 0 | $e_{02}$ | $x_1$ | $x_2$ | $e_1$ | 0 |
| $x_2$ | *1* | 1 | 1 | $e_{02}$ | $x_1$ | *0* | $e_{02}$ | 1 |
| $x_1$ | $x_2$ | 0 | 0 | $e_1$ | $x_1$ | $x_2$ | $e_1$ | 0 |
| $x_1$ | $x_2$ | – | 1 | $e_1$ | $x_1$ | *0* | $e_{02}$ | 1 |
| $x_1$ | $x_2$ | 1 | 0 | $e_1$ | $x_2$ | *1* | $e_{02}$ | 0 |

**(a)**

**Figura 2.8:** Ejemplo de TTEE. Es una copia de la TTEE de la figura 2.5b con las filas reordenadas.

[4]Recuérdese que los canales de selección con una sola entrada no requieren ningún multiplexor; por tanto, el número de multiplexores puede ser menor que $r$.

(para garantizar este valor, deben cumplirse las mismas condiciones que para la ecuación 1.7, en la página 18). Por lo tanto, el tamaño de la ROM viene dado por la expresión

$$T^{\text{Mealy}}_{\text{FSMIM-T}} = 2^r|E|(n+q+b) \leq 2^{r+q}(n+q+b) \text{ bits.} \tag{2.17}$$

En una máquina de Moore (arquitectura mostrada en la figura 2.7b), la memoria que contiene las transiciones se diferencia de la memoria de la máquina de Mealy en el ancho (que es $q+b$, ya que no contiene la salida). La ROM que implementa la función de salida tiene $2^b|E|$ palabras de $n$ bits de ancho. Por lo tanto, la cantidad de memoria requerida es

$$T^{\text{Moore}}_{\text{FSMIM-T}} = 2^r|E|(q+b) + 2^b|E|n \leq 2^{r+q}(q+b) + 2^{b+q}n \text{ bits.} \tag{2.18}$$

De las ecuaciones 2.17 y 2.18 se deduce que, para que la arquitectura FSMIM-T de Moore tenga interés práctico frente a la arquitectura FSMIM-T de Mealy, debe cumplirse que el número de bits de selección ($b$) sea menor que el número de canales ($r$).

El objetivo de la arquitectura FSMIM-T es obtener implementaciones FSMIM en FPGA que reduzcan el consumo de los bloques de memoria empotrados utilizando el menor número posible de celdas lógicas y con altas prestaciones en velocidad. El empleo de un banco de multiplexores en el que cada multiplexor está controlado por bits de selección específicos permite alcanzar este objetivo. Por una parte, las FPGA actuales disponen de recursos para la implementación eficiente de multiplexores [Xilinx, 2011e]. Por otra parte, en esta arquitectura el tamaño de cada multiplexor depende exclusivamente del número de entradas del canal de selección correspondiente. Esto permite reducir el consumo de celdas lógicas necesarias para su implementación y, puesto que el banco de multiplexores se encuentra en el camino crítico, aumentar la frecuencia máxima de operación

Sin embargo, el uso de los bits de selección presenta el inconveniente de incrementar el ancho de la ROM de la FSMIM-T. Esto significa que la arquitectura FSMIM-T reduce la profundidad de la ROM respecto a las implementaciones FSM-ROM a expensas de aumentar su ancho. A pesar de ello, la reducción de la profundidad tiene mayor influencia en la velocidad y en el consumo de recursos que el incremento del ancho [Senhadji-Navarro et al., 2012]. Este hecho unido a la reducida complejidad del banco de multiplexores, hace que las implementaciones FSMIM-T presenten mejores prestaciones que las implementaciones FSM-ROM tanto en velocidad como en consumo de bloques de memoria, consiguiendo reducir incluso el consumo de celdas lógicas en muchos casos, tal como mostrarán los resultados experimentales.

### 2.3.2. Arquitectura FSMIM con selección de entradas basada en estados

El tamaño de la memoria requerida por las implementaciones FSMIM basadas en la arquitectura FSMIM-T puede reducirse eliminando de la ROM los bits de selección. Basándonos en esta idea, presentamos en [García-Vargas y Senhadji-Navarro, 2015] una nueva arquitectura para la implementación de FSMIM que codifica los metaestados empleando una codificación binaria. Esto obliga a descomponer la función de transición de la FSMIM (ecuación 2.1) en varias funciones.

Sea $F = (E, P, t, h, p_0)$ una FSMIM con $m$ entradas y $r$ canales de selección, y sea $C = \{c_1, \ldots, c_{|P|}\}$ el conjunto de códigos binarios asignados a los metaestados de $P$. La función de transición $t : E \times \mathbb{B}^r \to P$ se descompone en tres funciones: la función

$$\tau : E \times \mathbb{B}^r \to C, \tag{2.19}$$

que genera el código del metaestado siguiente; la función $\sigma : C \to S_m^r$, que relaciona cada código de metaestado con la selección de entradas correspondiente; y la función

$$\gamma : C \to E, \tag{2.20}$$

que relaciona cada código de metaestado con el estado correspondiente de la FSMIM.

El banco de selectores de entradas implementa la operación de selección $X[c_i] = X[\sigma(c_i)]$, donde $X \in \mathbb{B}^m$ y $c_i \in C$. Por lo tanto, en esta arquitectura el banco de selectores de entradas está controlado por los bits de codificación de metaestados, o lo que es igual, por los bits de codificación de estados de la FSM (ya que existe una correspondencia biunívoca entre los estados de una FSM y los metaestados de la FSMIM equivalente). Por esta razón, la arquitectura ha sido denominada *FSMIM con selección de entradas basada en estados* (FSMIM-S, sigla del inglés **FSMIM** with **S**tate-based input selection)[5].

Por otra parte, se requiere un segundo bloque combinacional que implemente la función $\gamma$; es decir, que genere el código del estado presente de la FSMIM a partir del código del metaestado presente. Este bloque se denomi-

[5] Aunque en el nombre de la arquitectura el término *estado* hace referencia a los estados de la FSM, en esta sección dicho término se usará para denominar a los estados de la FSMIM salvo cuando se especifique lo contrario.

na *codificador de grupos* debido a que un mismo estado de la FSMIM puede estar relacionado con un grupo de metaestados diferentes[6].

La figura 2.10 muestra diferentes variantes de la arquitectura FSMIM-S, utilizadas para la implementación de máquinas de Mealy y de Moore con memoria asíncrona (figura 2.10a y figura 2.10b, respectivamente), y de máquinas de Mealy sincronizadas (figura 2.10c).

En el caso de una máquina de Mealy (figuras 2.10a y 2.10c), la dirección de acceso a la ROM viene determinada, al igual que en la arquitectura FSMIM-T, por los bits de codificación del estado presente y por el valor de las entradas seleccionadas por la selección de entradas presente. Esta dirección determina la palabra de la ROM que contiene el valor de la salida de la máquina (según la ecuación 2.2) y la codificación del metaestado siguiente (según la ecuación 2.19).

En el caso de una máquina de Moore (figura 2.7a), la salida es generada por una segunda ROM (según la ecuación 2.3), que es direccionada por la codificación del metaestado presente.

La figura 2.11 muestra un ejemplo de implementación en ROM de la FSMIM de la figura 2.8 utilizando la arquitectura FSMIM-S. Se trata de una FSMIM de Mealy. La codificación de los tres metaestados se muestra en la figura 2.11b, donde $c^{(1)}c^{(0)}$ son los dos bits que forman el código. Las tablas de verdad del codificador de grupos y de los selectores de entradas de los canales de selección 1 y 2 se muestran en las figuras 2.11c, 2.11d y 2.11e, respectivamente. Aunque en este ejemplo las dos primeras tablas se pueden simplificar, se ha optado por no hacerlo para ilustrar el hecho de que los modificadores de dirección de las FSMIM dependen, en general, de todos los bits de codificación de metaestados. Finalmente, la figura 2.11f muestra el contenido de la ROM.

#### 2.3.2.1. Prestaciones de la arquitectura FSMIM-S

Sea $F = (E, P, t, h, p_0)$ una FSMIM con $m$ entradas, $n$ salidas y $r$ canales de selección, equivalente a una FSM $F_{\text{FSM}} = (E', t', h', e'_0)$. Sea $p = \lceil \log_2 |P| \rceil$ el número de bits de codificación de metaestados (igual al número de bits de codificación de estados de la FSM, ya que $|P| = |E'|$), y sea $q = \lceil \log_2 |E| \rceil$ el número de bits de codificación de estados de la FSMIM.

En una máquina de Mealy (arquitecturas mostradas en las figuras 2.10a y 2.10c), cada palabra de la ROM contiene los $n$ bits de salida y los $p$ bits

[6] Recuérdese que la agrupación de estados en una FSMIM equivale a agrupar múltiples metaestados en un único estado.

de codificación de metaestados; por lo tanto, el ancho de la ROM es

$$n + p \quad \text{bits.} \tag{2.21}$$

Por otra parte, la profundidad de la ROM es

$$2^r|E| \quad \text{palabras} \tag{2.22}$$

(para garantizar este valor, deben cumplirse las mismas condiciones que para la ecuación 1.7, en la página 18). Por lo tanto, el tamaño de la ROM viene dado por la expresión

$$T^{\text{Mealy}}_{\text{FSMIM-S}} = 2^r|E|(n+p) \leq 2^{r+q}(n+p) \quad \text{bits.} \tag{2.23}$$

Esta ROM tiene el mismo ancho que la de la arquitectura FSM-ROM y la misma profundidad que la de la arquitectura FSMIM-T.

En una máquina de Moore (arquitectura mostrada en la figura 2.10b), cada palabra de la memoria que implementa la función de transición tan solo contiene los $p$ bits de codificación de metaestados. Compara con el resto de arquitecturas para máquinas de Moore, esta memoria tiene el ancho de la ROM de la arquitectura FSM-ROM y la profundidad de la ROM de la arquitectura FSMIM-T (existe, por tanto, la misma relación que entre las arquitecturas para máquinas de Mealy). La ROM que implementa la función de salida tiene $|E|$ palabras de $n$ bits de ancho (tiene menor profundidad que la ROM de la arquitectura FSMIM-T y el mismo ancho que la de la arquitectura FSM-ROM). La cantidad total de memoria requerida es

$$T^{\text{Moore}}_{\text{FSMIM-S}} = 2^r|E|p + |E|n \leq 2^{r+q}p + 2^q n \quad \text{bits.} \tag{2.24}$$

Si se comparan, por una parte, las ecuaciones 2.23, 2.17 y 1.8, y, por otra, las ecuaciones 2.24, 2.18 y 1.9, se puede comprobar que

$$\begin{aligned} T^{\text{Mealy}}_{\text{FSMIM-S}} &\leq \text{mín}\{T^{\text{Mealy}}_{\text{FSMIM-T}}, T^{\text{Mealy}}_{\text{FSM-ROM}}\}, \\ T^{\text{Moore}}_{\text{FSMIM-S}} &\leq \text{mín}\{T^{\text{Moore}}_{\text{FSMIM-T}}, T^{\text{Moore}}_{\text{FSM-ROM}}\}. \end{aligned} \tag{2.25}$$

Además, si se cumple que $|E| < |E'|$ o que $r < m$, entonces el tamaño de la FSMIM-S es estrictamente menor que el mínimo calculado en la ecuación 2.25.

Esta reducción en el tamaño de la memoria requerida se consigue a expensas de un incremento en el consumo de recursos lógicos adicionales, necesarios para implementar el banco de selectores de entradas y el codificador

de grupos. Puesto que el camino crítico siempre incluye a uno de los modificadores de dirección, se produce también una degradación de la frecuencia máxima de operación de las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T. Para cada FSM, el modificador de dirección más lento será el que determine la frecuencia máxima de operación.

El banco de selectores de entradas está formado por $r$ selectores de entradas como máximo[7], controlados por los $p$ bits de codificación de metaestados. Mientras que la complejidad de cada multiplexor de entradas de la FSMIM-T depende exclusivamente del número de entradas del canal de selección, la complejidad de los selectores de entradas depende, además, de la codificación de los metaestados (el valor de $p$ no debe ser considerado debido a que no se puede ajustar durante el diseño de la FSMIM). Por lo tanto, el banco de selectores de entradas es un bloque combinacional de mayor complejidad que el banco de multiplexores de la FSMIM-T.

El codificador de grupos es una función lógica de $p$ entradas y $q$ salidas. Su complejidad depende, por una parte, del número de estados de la FSMIM (ya que este determina el valor de $q$) y, por otra, de la codificación de los metaestados y de los estados de la FSMIM.

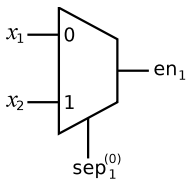
[7]Los canales de selección con una sola entrada no requieren ninguna función de selección.

| ep | Codificación |
|---|---|
| $e_{02}$ | 0 |
| $e_1$ | 1 |

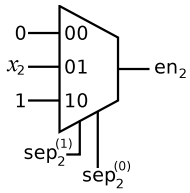
**(a)**

| ***sep₂*** | Codificación | |
|---|---|---|
| | $sep_2^{(1)}$ | $sep_2^{(0)}$ |
| *0* | 0 | 0 |
| *1* | 1 | 0 |
| $x_2$ | – | 1 |

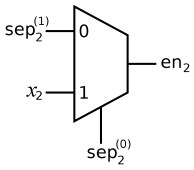
**(b)**




**(c)**




**(d)**




**(e)**

| **sep** | | **en** | | **ep** | **ses** | | **es** | **sal** |
|---|---|---|---|---|---|---|---|---|
| $sep_1$ | $sep_2$ | $en_1$ | $en_2$ | | $ses_1$ | $ses_2$ | | $y$ |
| 0 | 00 | 0 | 0 | 0 | 0 | 00 | 0 | 0 |
| 1 | 10 | 0 | 1 | 0 | 1 | 10 | 0 | 0 |
| 0 | 00 | 1 | 0 | 0 | 0 | –1 | 1 | 0 |
| 1 | 10 | 1 | 1 | 0 | 0 | 00 | 0 | 1 |
| 0 | –1 | 0 | 0 | 1 | 0 | –1 | 1 | 0 |
| 0 | –1 | – | 1 | 1 | 0 | 00 | 0 | 1 |
| 0 | –1 | 1 | 0 | 1 | 1 | 10 | 0 | 0 |

**(f)**

**ROM**

| | **Dirección** | | | **Contenido** | | | | |
|---|---|---|---|---|---|---|---|---|
| | ep | en$_1$ | en$_2$ | sep$_1^{(0)}$ | sep$_2^{(1)}$ | sep$_2^{(0)}$ | es | $y$ |
| $e_{02}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $e_{02}$ | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| $e_{02}$ | 0 | 1 | 0 | 0 | – | 1 | 1 | 0 |
| $e_{02}$ | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| $e_1$ | 1 | 0 | 0 | 0 | – | 1 | 1 | 0 |
| $e_1$ | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| $e_1$ | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| $e_1$ | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

**(g)**

**Figura 2.9:** Implementación en ROM de la FSMIM de la figura 2.8 usando la arquitectura FSMIM-T: (a) Codificación de estados de la FSMIM, (b) Codificación de la selección de entradas del canal de selección 2, (c) Multiplexor del canal de selección 1, (d) Multiplexor del canal de selección 2, (e) Implementación alternativa del multiplexor del canal de selección 2, (f) TTEE codificada y (g) Contenido de la ROM.

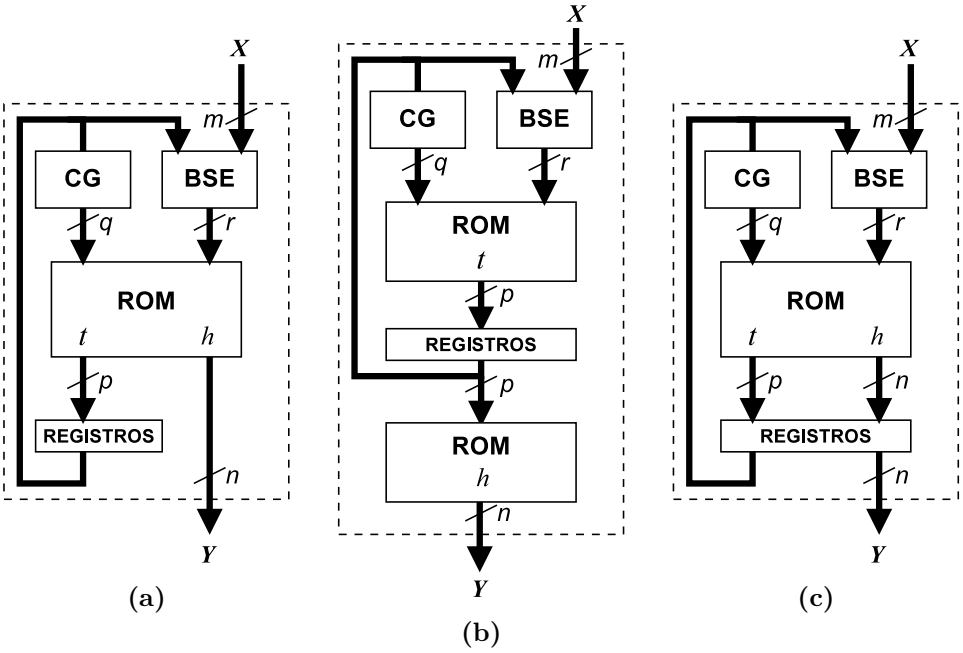



**Figura 2.10:** Arquitecturas para la implementación de FSMIM-S: (a) implementación basada en memoria asíncrona de una FSMIM de Mealy, (b) implementación basada en memoria asíncrona de una FSMIM de Moore y (c) implementación de una FSMIM de Mealy sincronizada. El bloque CG es el Codificador de Grupos y el bloque BSE, el Banco de Selectores de Entradas.

| **ep** | **Codificación** |
|---|---|
| $e_{02}$ | 0 |
| $e_1$ | 1 |

**(a)**

| **Metaestado** | **Codificación** | |
|---|---|---|
| | $c^{(1)}$ | $c^{(0)}$ |
| $(e_{02}, \langle x_1, 0\rangle)$ | 0 | 0 |
| $(e_{02}, \langle x_2, 1\rangle)$ | 0 | 1 |
| $(e_1, \langle x_1, x_2\rangle)$ | 1 | 0 |

**(b)**

| $c^{(1)}$ | $c^{(0)}$ | ep |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

**(c)**

| $c^{(1)}$ | $c^{(0)}$ | $en_1$ |
|---|---|---|
| 0 | 0 | $x_1$ |
| 0 | 1 | $x_2$ |
| 1 | 0 | $x_1$ |

**(d)**

| $c^{(1)}$ | $c^{(0)}$ | $en_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | $x_2$ |

**(e)**

**ROM**

| | **Dirección** | | | **Contenido** | | |
|---|---|---|---|---|---|---|
| | ep | $\text{en}_1$ | $\text{en}_2$ | $c^{(1)}$ | $c^{(0)}$ | $y$ |
| $e_{02}$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $e_{02}$ | 0 | 0 | 1 | 0 | 1 | 0 |
| $e_{02}$ | 0 | 1 | 0 | 1 | 0 | 0 |
| $e_{02}$ | 0 | 1 | 1 | 0 | 0 | 1 |
| $e_1$ | 1 | 0 | 0 | 1 | 0 | 0 |
| $e_1$ | 1 | 0 | 1 | 0 | 0 | 1 |
| $e_1$ | 1 | 1 | 0 | 0 | 1 | 0 |
| $e_1$ | 1 | 1 | 1 | 0 | 0 | 1 |

**(f)**

**Figura 2.11:** Implementación en ROM de la FSMIM de la figura 2.8 usando la arquitectura FSMIM-S: (a) Codificación de estados de la FSMIM, (b) Codificación de metaestados, (c) Tabla de verdad del codificador de grupos, (d) Tabla de verdad del selector de entradas del canal de selección 1, (e) Tabla de verdad del selector de entradas del canal de selección 2 y (f) Contenido de la ROM.

## 2.4. Optimización de FSMIM

Tal como se estudió en la sección 2.2, la agrupación de estados y la permutación de entradas permiten obtener FSMIM equivalentes con menor número de estados o con menor número de entradas en cada canal. Estas transformaciones permiten reducir el tamaño de la ROM y la complejidad de los modificadores de dirección. En términos generales, esto permite reducir, por una parte, el consumo de recursos de memoria necesarios para implementar la ROM en los dispositivos FPGA y, por otra, el número de recursos lógicos necesarios para implementar los modificadores de dirección. Se denominará *optimización de FSMIM* a la búsqueda de una FSMIM equivalente a otra dada, que minimice el valor de los siguientes objetivos: el tamaño de la ROM y la complejidad de los modificadores de dirección.

La arquitectura elegida para la implementación de la FSMIM determina algunas de las características del problema de optimización; sin embargo, las técnicas utilizadas para obtener soluciones son independientes de la arquitectura. Debido a que los bloques de memoria disponibles en los dispositivos FPGA actuales son síncronos [Xilinx, 2011c], las únicas arquitecturas adecuadas para la implementación de FSMIM en FPGA utilizando bloques de memoria son las correspondientes a las máquinas de Mealy sincronizadas (figuras 2.7c y 2.10c). Por lo tanto, estas serán las arquitecturas usadas para estudiar el problema de optimización de FSMIM presentado en esta tesis.

Se denominará *función tamaño de ROM* a la función $R : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_+$, donde $R(p, b)$ es el tamaño de una ROM de $p$ palabras de profundidad y $b$ bits de ancho. Obviamente, en el caso de que el tamaño de la ROM se mida en bits, $R(p, b) = pb$. Sin embargo, en el caso de que el tamaño de la ROM se mida como el número de bloques de memoria empotrados del dispositivo FPGA, la función $R$ es una función escalonada que depende de las características de los recursos de memoria (tamaño de los bloques, conjunto de geometrías de memoria admitidas por los bloques, etc.). Los detalles sobre la función $R$ se estudian en la sección G.2.1.

En cualquier caso, el tamaño de la ROM es una función creciente con el ancho y la profundidad, por lo que el análisis que se realizará en los diferentes apartados de esta sección es independiente de la unidad de medida utilizada.

### 2.4.1. Matriz de selección de entradas

A pesar de que una TTEE permite representar convenientemente a las FSMIM, la información relativa a las transiciones y las salidas de la máquina no es relevante para el estudio de las transformaciones que se aplican

a las FSMIM. Por lo tanto, es conveniente una forma de representación más compacta para estudiar el problema de optimización.

**Definición 2.17.** *Sea $F = (E, P, t, h, p_0)$ una FSMIM con $m$ entradas y $r$ canales de selección. Sean $p^{(1)}, \ldots, p^{(k)}$ los elementos de $P$ ordenados arbitrariamente, siendo $k = |P|$. Se define una* **Matriz de Selección de Entradas** *(MSE) de $F$ como una matriz $A = (a_{ij}) \in \mathcal{M}_{k\times r}$, en la que cada elemento $a_{ij} \in \{x_1, \ldots, x_m\} \cup \{0, 1, -\}$ es un símbolo que identifica a la entrada seleccionada en el canal de selección $j$ por el metaestado $p^{(i)}$ de $F$; donde $x_l$ $(1 \leq l \leq m)$ representa a las entradas de $F$, $\{0, 1\}$ representan valores constantes, y '$-$' representa una ESI:*

$$\begin{array}{c} \\ p_1 \\ \vdots \\ p_i \\ \vdots \\ p_k \end{array} \begin{array}{c} \begin{array}{cccccc} \mathit{canal}_1 & \cdots & \mathit{canal}_j & \cdots & \mathit{canal}_r \end{array} \\ \begin{pmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1r} \\ \vdots & \ddots & \vdots & & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{ir} \\ \vdots & & \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kj} & \cdots & a_{kr} \end{pmatrix} \end{array}.$$

*El conjunto de todas las MSE de orden $n \times m$ se representa como $MSE^{n\times m}$.*

La MSE sólo contiene información de la selección de entradas de cada metaestado de la FSMIM; sin embargo, en algunos casos puede ser necesaria también la información del estado correspondiente.

**Definición 2.18.** *Sea $F$ una FSMIM. Se denomina* **Matriz de Selección de Entradas Extendida** *(MSEE) de $F$ a una MSE de $F$ en la que se ha añadido información del estado de la FSMIM asociado a cada selección de entradas. Se representa como*

$$\left[\begin{array}{cccccc|c} a_{11} & \cdots & a_{1j} & \cdots & a_{1r} & & e^{(1)} \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{ir} & & e^{(i)} \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ a_{k1} & \cdots & a_{kj} & \cdots & a_{kr} & & e^{(k)} \end{array}\right],$$

*donde $e^{(i)} \in E$ es el estado asociado al metaestado $p^{(i)}$ $(1 \leq i \leq k)$. Por lo tanto, la fila $i$ de una MSEE representa al metaestado*

$$p^{(i)} = (e^{(i)}, \langle a_{i1}, \ldots, a_{ij}, \ldots, a_{ir}\rangle).$$

*El conjunto de todas las MSEE de orden $n \times m$ se representa como $MSEE^{n\times m}$.*

La figura 2.12 muestra una TTEE (figura 2.12a) y su correspondiente MSEE (figura 2.12b). Las MSEE contienen toda la información necesaria para representar las transformaciones elementales de una FSMIM: permutación de entradas, asignación de indeterminaciones y agrupación de estados. Por lo tanto, las MSEE permiten especificar de forma compacta los conjuntos de transformaciones elementales que se pueden aplicar a una FSMIM para obtener una nueva FSMIM, aunque el resultado no es necesariamente una FSMIM equivalente a la original.

**Definición 2.19.** *Sea $F$ una FSMIM y $A_F$ una MSEE de $F$. Se dice que una MSEE $A$* **es aplicable** *a $F$ si $A$ se puede obtener aplicando transformaciones elementales sobre $A_F$. Si $A$ es aplicable a $F$, entonces la FSMIM* **generada por** *$A$ es la que se obtiene al aplicar a $F$ todas las transformaciones elementales especificadas por $A$.*

*Se define el* **conjunto de MSEE equivalentes** *de $F$, denotado por $MSEE[F]$, como el conjunto de todas las MSEE aplicables a $F$ que generan FSMIM equivalentes a $F$. Se dirá que dos MSE* **son equivalentes** *si ambas pertenecen a $MSEE[F]$. Se cumple que $A_F \in MSEE[F]$.*

La figura 2.12c muestra la MSEE equivalente a la de la figura 2.12b que se obtiene al asignar constantes, agrupar estados y permutar entradas (los cambios se muestran en negrita).

En algunos casos, realizar permutaciones de filas o columnas completas puede ser útil para facilitar el estudio de las transformaciones que se pueden realizar sobre una FSMIM dada. Esta modificación de la MSEE no afecta a la implementación de la FSMIM. Por una parte, puesto que cada fila de una MSEE contiene información tanto de la selección de entradas como del estado en el que se aplica, el orden de las filas no tiene relevancia sobre la FSMIM que representa. Por otra parte, se puede considerar la permutación de las columnas simplemente como un cambio en la numeración de los canales de selección (tras el cual, al canal de selección $i$ no le corresponde la columna $i$ de la MSEE, sino otra diferente).

Dada una MSEE, se puede redefinir el problema de optimización de FSMIM como la búsqueda de una MSEE equivalente que minimice el tamaño de la ROM y la complejidad de los modificadores de dirección. Por ejemplo, a partir de la MSEE de la figura 2.12b, se puede obtener de forma trivial la matriz equivalente de la figura 2.12c. Esta MSEE corresponde a una FSMIM equivalente a la original, en la que se han asignado valores constantes a las indeterminaciones de los estados $e_0$ y $e_2$ para permitir su agrupación en el estado $e_0 \oplus e_2$, lo que reduce el número de estados de tres a dos. Además,

se han permutado las entradas seleccionadas del estado $e_1$ para reducir el número de entradas del canal 1 (de tres a dos entradas).

**Definición 2.20.** *Sea $A \in MSEE^{n\times r}$. Se define $E(A)$ como el conjunto de estados de $A$, siendo $|E(A)| \leq n$. Se define el* **coste de selección de entradas del canal (o columna)** *$j$ de $A$ como $CSE_j(A) = |\{a_{ij} : 1 \leq i \leq k\} \setminus \{-\}|$, que es igual al número de entradas del canal de selección $j$ de la FSMIM. Se define el* **coste de selección de entradas** *de $A$ como $CSE(A) = \sum_{j=1}^{r} CSE_j(A)$.*

Sea $F$ una FSMIM. Se denominará *función tamaño de FSMIM* definida para $F$ (o simplemente *función tamaño de $F$*) a la función $TAM : MSEE[F] \rightarrow \mathbb{R}_+$, tal que $TAM(A)$ es el tamaño de la ROM de la FSMIM generada por $A \in MSEE[F]$ (que, por definición, es equivalente a $F$). La definición de la función $TAM$ depende en primer lugar de la arquitectura utilizada para implementar la FSMIM. De hecho, si el tamaño de la ROM se mide en bits, el valor de la función se obtiene a partir de la ecuación 2.17, en el caso de la arquitectura FSMIM-T, o de la ecuación 2.23, en el caso de la arquitectura FSMIM-S. Por otra parte, si el tamaño de la ROM se mide en número de bloques de memoria, entonces la definición de $TAM$ depende también de las características de los recursos de memoria del dispositivo FPGA.

| **sep** | | **en** | | **ep** | **ses** | | **es** | **sal** |
|---|---|---|---|---|---|---|---|---|
| $sep_1$ | $sep_2$ | $en_1$ | $en_2$ | | $ses_1$ | $ses_2$ | | $y$ |
| $x_1$ | $-$ | 0 | $-$ | $e_0$ | $x_2$ | $x_3$ | $e_1$ | 1 |
| $x_1$ | $-$ | 1 | $-$ | $e_0$ | $x_1$ | $-$ | $e_0$ | 0 |
| $x_2$ | $x_3$ | 0 | 0 | $e_1$ | $x_2$ | $x_3$ | $e_1$ | 0 |
| $x_2$ | $x_3$ | 0 | 1 | $e_1$ | $x_1$ | $-$ | $e_0$ | 0 |
| $x_2$ | $x_3$ | 1 | $-$ | $e_1$ | $x_3$ | $-$ | $e_2$ | 1 |
| $x_3$ | $-$ | 0 | $-$ | $e_2$ | $x_2$ | $x_3$ | $e_1$ | 1 |
| $x_3$ | $-$ | 1 | $-$ | $e_2$ | $x_1$ | $-$ | $e_0$ | 0 |

**(a)**

$$\left[\begin{array}{cc|c} x_1 & - & e_0 \\ x_2 & x_3 & e_1 \\ x_3 & - & e_2 \end{array}\right] \qquad \left[\begin{array}{cc|c} x_1 & \mathbf{0} & \boldsymbol{e_0 \oplus e_2} \\ \boldsymbol{x_3} & \boldsymbol{x_2} & e_1 \\ x_3 & \mathbf{1} & \boldsymbol{e_0 \oplus e_2} \end{array}\right]$$

**(b)** **(c)**

**Figura 2.12:** Ejemplo de MSEE ampliada: (a) TTEE, (b) MSEE y (c) MSEE equivalente (los cambios están marcados en negrita).

**Definición 2.21.** *Dada una FSMIM $F$, la optimización de $F$ puede ser formulada como el siguiente problema de optimización multiobjetivo:*

$$\begin{gathered} \textit{minimizar } \{\ TAM(A),\ CSE(A)\ \} \\ \textit{sujeto a } A \in MSEE[F]. \end{gathered}$$

### 2.4.2. Estrategias de optimización de FSMIM

Debido a la complejidad del problema de optimización de FSMIM, se ha optado por dividirlo en dos subproblemas diferentes denominados *minimización de la selección de entradas* y *agrupación de estados*.

El objetivo principal de la minimización de la selección de entradas es reducir la complejidad del banco de multiplexores. Este subproblema consiste en buscar las permutaciones de entradas de la MSEE que minimizan el coste de selección de entradas o el número de bits de selección. En [García-Vargas y Senhadji-Navarro, 2012] demostramos que este problema es *NP*-completo.

La agrupación de estados tiene como objetivo la reducción del tamaño de la ROM. Consiste en buscar la asignación de indeterminaciones y la agrupación de estados que minimiza el tamaño de la ROM (tal como se estudiará más adelante, dependiendo de la arquitectura utilizada, el número de estados óptimo para alcanzar el tamaño mínimo de la ROM puede corresponder al mínimo número de estados posible o a un valor mayor).

Ambos subproblemas tienen objetivos en conflicto, lo que significa, por una parte, que las soluciones encontradas por cualquiera de ellos pueden empeorar las soluciones del otro. Por ejemplo, en la MSEE de la figura 2.12c, la agrupación de estados ha aumentado el número de entradas del canal 2, que ha pasado de tener una sola entrada ($x_2$) a tener tres ($x_2$, 0 y 1).

Por otra parte, el orden en el que se resuelven los subproblemas influye en la solución final. Es decir, solucionar uno de los subproblemas primero, restringe el espacio de búsqueda del otro, lo que degrada la calidad de las soluciones encontradas para el problema de optimización completo. Resolver la agrupación de estados restringe el número de permutaciones de entradas que puede realizar la minimización de la selección de entradas. De forma similar, las soluciones de la minimización de selección de entradas fijan las posiciones de las ESI, limitando el número de estados que pueden agruparse.

**Ejemplo 2.22.** *La figura 2.13 ilustra el conflicto entre ambos subproblemas. La MSE inicial $A$ tiene tres estados y el coste de selección de entradas es $CSE(A) = 2 + 3 + 1$. En primer lugar, si se comienza minimizando la selección de entradas de $A$ (figura 2.13b), se obtiene la MSEE $A^M$, con tres estados y $CSE(A^M) = 1+2+1$. Sin embargo, no es posible agrupar estados a*

*partir de $A^M$ sin alterar la permutación de entradas obtenida como solución. En segundo lugar, si se comienza agrupando estados, se obtiene la matriz $A^A$ (figura 2.13b), con dos estados y CSE($A^A$) = 2+3+3. Minimizando después la selección de entradas de $A^A$, se obtiene la MSEE $A^{AM}$, con dos estados y CSE($A^{AM}$) = 2 + 2 + 3. En esta minimización, el número de permutaciones posibles se ha visto limitado debido a que las constantes deben permanecer en la misma columna para que se cumpla que $e_0 \simeq e_2$.*

Este efecto mutuo que tiene cada subproblema en el espacio de búsqueda del otro es general para el problema de optimización de FSMIM. Sin embargo, existen otras influencias cruzadas entre las soluciones de ambos subproblemas que dependen de la arquitectura FSMIM utilizada. Estas serán estudiadas en los próximos apartados.

A pesar del inconveniente de que las soluciones obtenidas no son necesariamente óptimas, la división en subproblemas permite simplificar el problema general para que pueda ser abordado más fácilmente. Las estrategias que se presentan en esta tesis para optimizar las FSMIM consisten en dividir el proceso de optimización en varias fases, denominadas *fases*

$$A = \left[\begin{array}{ccc|c} x_1 & x_4 & - & e_0 \\ x_1 & x_2 & x_3 & e_1 \\ x_2 & x_3 & - & e_2 \end{array}\right] \qquad A^M = \left[\begin{array}{ccc|c} x_1 & x_4 & - & e_0 \\ x_1 & x_2 & x_3 & e_1 \\ \boldsymbol{-} & \boldsymbol{x_2} & \boldsymbol{x_3} & e_2 \end{array}\right]$$

**(a)** **(b)**

$$A^A = \left[\begin{array}{ccc|c} x_1 & x_4 & \mathbf{0} & \boldsymbol{e_{02}} \\ x_1 & x_2 & x_3 & e_1 \\ x_2 & x_3 & \mathbf{1} & \boldsymbol{e_{02}} \end{array}\right] \quad A^{AM} = \left[\begin{array}{ccc|c} x_1 & x_4 & 0 & e_{02} \\ x_1 & x_2 & x_3 & e_1 \\ \boldsymbol{x_3} & \boldsymbol{x_2} & 1 & e_{02} \end{array}\right]$$

$$e_{02} \equiv e_0 \oplus e_2$$

**(c)**

**Figura 2.13:** Ejemplo de minimización de selección de entradas y agrupación de estados de una MSEE: (a) MSEE inicial, (b) minimización de la selección de entradas de $A$ y (c) agrupación de los estados de $A$ seguido de minimización de la selección de entradas de $A^A$. Los cambios están marcados en negrita.

*de optimización*[8], alternándose en cada fase el subproblema que se intenta resolver.

La estrategia propuesta en [Senhadji-Navarro et al., 2004] y utilizada en [García-Vargas et al., 2007; García-Vargas y Senhadji-Navarro, 2015] divide el problema de optimización en dos fases: una fase de minimización de selección de entradas seguida de una fase de agrupación de estados [García-Vargas, 2006]. En esta tesis se proponen nuevas estrategias de dos o más fases, que pueden comenzar con la minimización de selección de entradas o con la agrupación de estados. Además se proponen nuevas técnicas para resolver problemas de minimización de selección de entradas [García-Vargas y Senhadji-Navarro, 2012] y de agrupación de estados.

#### 2.4.2.1. Optimización de FSMIM-T

Tal como se mencionó en la introducción de la sección, algunas características del problema de optimización de FSMIM depende de la arquitectura elegida para su implementación. En este apartado se analiza este problema cuando se aplica a implementaciones FSMIM-T de máquinas de Mealy sincronizadas (que será denominado *optimización de FSMIM-T*).

Sea $R : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_+$ la función tamaño de ROM. Dada una FSMIM $F = (E, P, t, h, p_0)$ con $m$ entradas, $n$ salidas y $r$ canales de selección, la optimización de la FSMIM-T puede ser formulada como el siguiente problema de optimización multiobjetivo:

$$\text{minimizar } \{\ R(\,2^r|E(A)|,\ n+b+q\,), \tag{2.26}$$

$$CSE(A)\ \}, \tag{2.27}$$

$$\text{donde } q = \lceil \log_2 |E(A)| \rceil, \tag{2.28}$$

$$b = \sum_{j=1}^{r} \lceil \log_2 CSE_j(A) \rceil, \tag{2.29}$$

$$\text{sujeto a } A \in MSEE[F].$$

Los valores de $r$ y $n$ son constantes, las soluciones sólo dependen de $A$. La ecuación 2.26 define el tamaño de la ROM para esta arquitectura y se obtiene a partir de las ecuaciones 2.16 y 2.15. En el caso de que el tamaño de la ROM se mida en bits, el valor de la ecuación 2.26 coincide con el de la ecuación 2.17. El tamaño de la ROM depende del número de estados ($|E(A)|$), que es minimizado por la fase de agrupación de estados, y del coste

[8]Las fases de optimización serán denominadas simplemente *fases* excepto en aquellos casos en los que el contexto no permita evitar la ambigüedad.

de selección de entradas de cada canal ($CSE_j(A)$ en la ecuación 2.29), que es minimizado por la fase de minimización de la selección de entradas.

Por otra parte, la ecuación 2.27 permite minimizar la complejidad del banco de multiplexores y depende exclusivamente del coste de selección de entradas, que es minimizado por la fase de minimización de la selección de entradas. Puesto que ambos objetivos están relacionados por el coste de selección de entradas, la minimización de la selección de entradas, además de simplificar el banco de multiplexores, también influye en la reducción del tamaño de la ROM.

La agrupación de estados reduce la profundidad de la ROM, pero puede tener dos efectos negativos: además de aumentar la complejidad del banco de multiplexores, puede incrementar el ancho de la ROM. Esto es debido a que la asignación de constantes a un canal de selección incrementa en dos el número de entradas del canal, provocando un incremento en el número de entradas de los multiplexores correspondientes y, en algunos casos, un incremento del número de bits de selección (valor de $b$); aunque, debido al redondeo por exceso de la ecuación 2.29, el aumento del número de bits de selección no se produce siempre. Como consecuencia del incremento del ancho de la ROM, la solución óptima de la ecuación 2.26 (es decir, el tamaño mínimo de la ROM) no se obtiene necesariamente para el número mínimo de estados. Como contrapartida, una disminución del número de estados puede reducir el valor de $q$, favoreciendo la reducción del ancho de la ROM.

#### 2.4.2.2. Optimización de FSMIM-S

En este apartado se analizará el problema de optimización de FSMIM cuando se aplica a implementaciones FSMIM-S de máquinas de Mealy sincronizadas (que será denominado *optimización de FSMIM-S*).

Sea $R : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_+$ la función tamaño de ROM. Dada una FSMIM $F = (E, P, t, h, p_0)$ con $m$ entradas, $n$ salidas y $r$ canales de selección, la optimización de la FSMIM-S puede ser formulado como el siguiente problema de optimización multiobjetivo:

$$\begin{aligned} \text{minimizar } \{\ & R(\,2^r|E(A)|,\ n + \lceil \log_2|P| \rceil\,), && (2.30)\\ & CSE(A)\ \} && (2.31)\\ \text{sujeto a } & A \in MSEE[F]. \end{aligned}$$

Los valores de $r$, $n$ y $|P|$ son constantes, por lo que cada objetivo de optimización depende exclusivamente de una propiedad diferente de $A$ (el número de estados en un caso y el coste de selección de entradas en el otro). La

ecuación 2.30 define el tamaño de la ROM para esta arquitectura y se obtiene a partir de las ecuaciones 2.22 y 2.21. En el caso de que el tamaño de la ROM se mida en bits, el valor coincide con el de la ecuación 2.23. El tamaño de la ROM depende exclusivamente del número de estados ($|E(A)|$), que es minimizado por la agrupación de estados. La complejidad del banco de selectores de entradas (ecuación 2.31), depende del coste de selección de entradas ($CSE(A)$), que es minimizado por la minimización de la selección de entradas.

A diferencia de la arquitectura FSMIM-T, el ancho de la ROM de la arquitectura FSMIM-S es constante; es decir, el tamaño de la ROM sólo depende de su profundidad. Por lo tanto, el valor óptimo de la ecuación 2.30 se obtiene para el número de estados mínimo. Por otra parte, al igual que en la arquitectura FSMIM-T, la agrupación de estados aumenta la complejidad del banco de selectores de entrada. En general, la asignación de constantes en un canal de selección aumenta la complejidad de la función lógica del selector de entradas correspondiente debido a que disminuye el número de indeterminaciones.

Aunque la complejidad del banco de selectores de entradas está relacionada con el coste de selección de entradas, en la arquitectura FSMIM-S también depende de otros factores, como la codificación de metaestados, que es independiente de la MSEE y se realiza tras la optimización de la FSMIM-S. Como consecuencia, la influencia de la minimización de la selección de entradas en la simplificación de los selectores de entradas es menor en esta arquitectura que en la arquitectura FSMIM-T. Por otra parte, la minimización de la selección de entradas no afecta al tamaño de la ROM.

El número de entradas del codificador de grupos es constante, ya que depende del número de bits de codificación de metaestados ($\lceil \log_2 |P| \rceil$). Por otra parte, el número de salidas depende del número de bits de codificación de estados ($\lceil \log_2 |E(A)| \rceil$); por lo tanto, la agrupación de estados puede reducir el número de salidas del codificador de grupos. Sin embargo, la complejidad de la función lógica asociada a cada salida depende de dos factores: (a) de la relación que asocia los metaestados con los estados y (b) de la codificación de los metaestados y estados de la FSMIM. De estos factores, el primero se establece durante la agrupación de estados, que determina qué metaestados se agruparán en un mismo estado de la FSMIM. Sin embargo, el estudio de la influencia de la agrupación de estados en el codificador de grupos no se ha realizado en esta tesis, por lo que los algoritmos de agrupación propuestos no incluyen ningún criterio para reducir su complejidad. Dicho estudio, que formará parte de los trabajos futuros, podría dar lugar a nuevos algoritmos de agrupación que favorezcan la simplificación del codifi-

cador de grupos. Por otra parte, la codificación de los metaestados y estados es realizada tras la optimización de la FSMIM.

Capítulo 3

# Agrupación de estados

En el presente capítulo se estudiarán los algoritmos empleados en las fases de agrupación de estados de las diferentes estrategias de optimización de FSMIM y que han sido implementadas en las herramientas FSMIM-Gen 1.2 y FSMIM-Gen 2.0 (véase el apéndice H). El estudio comenzará con el algoritmo presentado en [García-Vargas, 2006], que ha servido como punto de partida para el resto de algoritmos que se proponen en este capítulo.

## 3.1. Definiciones y conceptos previos

Denominaremos *grupo de estados* (o simplemente grupo) a un conjunto de estados seudoequivalentes. La fase de agrupación de estados se aplica a una FSMIM en la que todos los estados son simples (obtenida a partir de una FSM mediante el procedimiento de conversión trivial), y se obtiene como resultado una partición del conjunto de estados en la que cada subconjunto es un grupo de estados (de cardinalidad mayor o igual a uno). Puesto que cada grupo da lugar a un único estado en la FSMIM generada por la fase de agrupación de estados, el número final de estados será igual al número de grupos obtenido. Por lo tanto, se puede redefinir el objetivo de la fase de agrupación de estados como la búsqueda de una partición del conjunto de estados que minimice el tamaño de la ROM de la implementación FSMIM.

Aunque en los ejemplos se utilizarán las MSEE para ilustrar la agrupación de estados, los algoritmos presentados en esta sección operan con una MSE, que corresponde a una FSMIM en la que todos los estados son simples; es decir, en la que cada estado está asociado a una única fila de la MSE. Esto implica que cada estado de la FSMIM puede ser identificado

por el índice de la fila correspondiente, y que los grupos de estados pueden ser representados como conjuntos de índices. Dada una MSE $A \in MSE^{n\times r}$, el resultado de los algoritmos de agrupación es una partición de $\{1,\ldots,n\}$ que determina los grupos de estados, y una MSE con la asignación de indeterminaciones necesaria para que los estados de los diferentes grupos sean seudoequivalentes.

Todos los algoritmos de agrupación presentados en esta tesis realizan las siguientes operaciones básicas. Sea $F = (E, P, t, h, p_0)$ una FSMIM formada por estados simples y con $r$ canales de selección. Sea $A \in MSE^{n\times r}$ una MSE de $F$. Inicialmente, cada índice de fila constituye un grupo diferente. En cada paso, los algoritmos de agrupación seleccionan un par de grupos que serán agrupados (dando lugar a un nuevo grupo) y realizan la asignación de indeterminaciones necesaria en la MSE. Sean $G^i$ y $A^i$ el conjunto de grupos y la MSE obtenidos en el paso $i$ del algoritmo; ambos determinan una FSMIM $F^i = (E^i, P^i, t^i, h^i, p_0^i)$ en la que cada grupo $g \in G^i$ representa a un único estado $e \in E^i$ tal que para todo $k \in g$ existe una y sólo una selección de entradas $s \in P^i(e)$ representada por la fila $k$ de $A^i$.

Sean $g_1^{i-1}, g_2^{i-1} \in G^{i-1}$ los dos grupos que se han agrupado en $g^i$, que representan a los estados $e_1^{i-1}$ y $e_2^{i-1}$, respectivamente, de la FSMIM $F^{i-1} = (E^{i-1}, P^{i-1}, t^{i-1}, h^{i-1}, p_0^{i-1})$. Para que los estados sean seudoequivalentes y se puedan agrupar, debe realizarse una asignación exclusiva de indeterminaciones a $e_1^{i-1}$ y $e_2^{i-1}$ en un canal de selección $j$ (véase la Definición 2.12). Esto equivale a realizar en la columna $j$ de $A^{i-1}$ una asignación de indeterminaciones a todas las filas de los grupos $g_1^{i-1}$ y $g_2^{i-1}$, dando lugar a $A^i$. La asignación de indeterminaciones permite obtener un nuevo conjunto de grupos $G^i = (G^{i-1} \setminus \{g_1^{i-1}, g_2^{i-1}\}) \cup g^i$, donde $g^i = g_1^{i-1} \cup g_2^{i-1}$ (que representa al estado $e^i = e_1^{i-1} \oplus e_2^{i-1}$). $G^i$ y $A^i$ determinan la nueva FSMIM $F^i = F^{i-1}_{e_1^{i-1}\oplus e_2^{i-1}}$.

**Definición 3.1.** *Se define la función* $I_j : MSE^{n\times r} \times 2^{\{1,\ldots,n\}} \to \mathbb{B}$ *para* $1 \le j \le r$ *como*

$$I_j(A, g) = \begin{cases} 1 & \text{si } a_{k,j} \text{ es una ESI para todo } k \in g \\ 0 & \text{en otro caso} \end{cases}, \tag{3.1}$$

*donde* $A = (a_{i,j}) \in MSE^{n\times r}$. *Se dirá que la columna* $j$ *de* $A$ **está indeterminada en un grupo** $g \subseteq \{1,\ldots,n\}$ *si* $I_j(A, g) = 1$. *Si* $I_j(A, g) = 0$ *y* $a_{k,j}$ *es una ESI, donde* $k \in g$, *entonces se dirá que* $a_{k,j}$ *es una* **indeterminación anulada**. *Es decir, las indeterminaciones anuladas del grupo* $g$ *son las ESI pertenecientes a* $g$ *que no se encuentran en columnas indeterminadas.*

Por lo tanto, para que dos grupos puedan agruparse mediante la asignación exclusiva de indeterminaciones en una columna dada, es necesario que dicha columna esté indeterminada en ambos grupos.

Aunque el número de grupos que se obtiene en la fase de agrupación viene determinado por el algoritmo utilizado, se puede establecer una cota inferior que depende exclusivamente del número de indeterminaciones de cada fila de la MSE. En la sección 3.5 se demostrará que esta cota es realmente un mínimo y se presentará un algoritmo que permite encontrar dicho valor. Sin embargo el resto de algoritmos que se estudiarán no garantizan la obtención del valor mínimo.

**Lema 3.2.** *Sea $F = (E, P, t, h, p_0)$ una FSMIM con $r$ canales de selección y $m$ entradas, formada por estados simples y sin constantes asignadas a indeterminaciones (estas son las características de las FSMIM generadas a partir de una FSM por el procedimiento de conversión trivial). Sea $F^i = (E^i, P^i, t^i, h^i, p_0^i)$ la FSMIM obtenida por un algoritmo de agrupación en el paso $i$. Para cada $e \in E^i$ se cumple que*

$$X[s] \neq X[s'] \quad \forall X \in \mathbb{B}^m \text{ y } \forall s, s' \in P^i(e) \text{ tal que } s \neq s'. \tag{3.2}$$

***Demostración.*** Se demuestra por inducción. El caso $F^0$ es trivial, ya que $F^0 = F$ está formada por estados simples; es obvio que se satisface la ecuación 3.2 para todo $e \in E$ debido que $|P(e)| = 1$.

Supóngase ahora que la ecuación 3.2 se cumple para todo $e \in F^i$. Sean $g_1^i$ y $g_2^i$ los dos grupos que son agrupados en el paso $i+1$ realizando asignación de indeterminaciones en la columna $j$ $(1 \leq j \leq r)$, dando lugar al grupo $g^{i+1}$. Sean $e_1^{i'}$ y $e_2^{i'}$ los estados representados por los grupos $g_1^i$ y $g_2^i$, respectivamente. Puesto que en $e_1^{i'}$ y $e_2^{i'}$ se ha realizado la asignación exclusiva de indeterminaciones en el canal de selección $j$, por la Proposición 2.13 se tiene que $X[s] \neq X[s']$ para todo $X \in \mathbb{B}^m$, $s \in P^i(e_1^{i'})$ y $s' \in P^i(e_2^{i'})$. Esto, junto a la ecuación 3.2, implica que $X[s] \neq X[s']$ para todo $X \in \mathbb{B}^m$ y todo $s, s' \in P^i(e_1^{i'}) \cup P^i(e_2^{i'})$ tal que $s \neq s'$. Por lo tanto, el nuevo estado $e^{i+1} = e_1^{i'} \oplus e_2^{i'}$, representado por el grupo $g^{i+1}$ satisface la ecuación 3.2. Puesto que el resto de estados de $F^{i+1} = F^i_{e_1^{i'} \oplus e_2^{i'}}$ son iguales a los estados de $F^i$ (es decir, $E^{i+1} \setminus \{e^{i+1}\} \subset E^i$), se puede concluir que la ecuación 3.2 se satisface para todos los estados de $F^{i+1}$. □

**Proposición 3.3.** *Sea $F = (E, P, t, h, p_0)$ una FSMIM con $r$ canales de selección y $m$ entradas, formada por estados simples. Entonces, la expresión*

$$\left\lceil \frac{\sum_{e \in E} 2^{w_e}}{2^r} \right\rceil, \tag{3.3}$$

*donde $w_e$ es el número de entradas efectivas de e, es una cota inferior del número de grupos que se puede obtener en la fase de agrupación de estados.*

***Demostración.*** Sea $S_m^r$ un conjunto de selecciones de entradas de $r$ canales de selección. Se define la función $\delta : S_m^r \to 2^{\mathbb{B}^r}$ como $\delta(s) = \{X[s] \in \mathbb{B}^r : X \in \mathbb{B}^m\}$; es decir, el conjunto de tuplas $X \in \mathbb{B}^r$ que genera la selección de entradas $s$. Es fácil comprobar que $|\delta(s)| = 2^{r-c}$, donde $c$ es el número de valores constantes $\{0, 1\}$ seleccionados por $s$. Por lo tanto, $1 \leq |\delta(s)| \leq 2^r$.

Puesto que la FSMIM $F$ está formada por estados simples, la selección de entradas de $e \in E$ será denotada $s_e$; es decir, $P(e) = \{s_e\}$ para todo $e \in E$. El valor de $|\delta(s_e)|$ depende de la asignación de constantes a las ESI del estado $e$, obteniéndose el mínimo cuando se asignan constantes a todas las ESI; es decir, cuando las únicas entradas no constantes son las entradas efectivas. Por lo tanto, $|\delta(s_e)| \geq 2^{w_e}$, donde $w_e$ es el número de entradas efectivas del estado $e$.

Sea $G$ una partición de $E$ obtenida en la fase de agrupación de estados y $F^G = (E^G, P^G, t^G, h^G, p_0^G)$ la FSMIM generada a partir de $G$, en la que se ha realizado la asignación de indeterminaciones necesaria para la agrupación. Sea $E^G = \{e_g^G : g \in G\}$, donde $e_g^G$ es el estado obtenido al agrupar los estados del grupo $g \in G$. Por lo tanto, $P^G(e_g^G) = \{s_e^G : e \in g\}$, donde $s_e^G$ denota a la selección de entradas $s_e$ con la asignación de indeterminaciones realizadas durante la agrupación. Se cumple que $|\delta(s_e^G)| \geq 2^{w_e}$ para todo $e \in E$.

Se define el número de direcciones del estado $e_g^G \in E^G$ como

$$\Delta(e_g^G) = \left| \bigcup_{s \in P^G(e_g^G)} \delta(s) \right|,$$

que verifica que

$$\Delta(e_g^G) \leq 2^r. \tag{3.4}$$

Dado cualquier $e_g^G \in E^G$, por el Lema 3.2 se tiene que $\delta(s) \cap \delta(s') = \emptyset$ para todo $s, s' \in P^G(e_g^G)$ tal que $s \neq s'$. Por lo tanto,

$$\Delta(e_g^G) = \left| \bigcup_{s \in P^G(e_g^G)} \delta(s) \right| = \sum_{s \in P^G(e_g^G)} |\delta(s)| \geq \sum_{e \in g} 2^{w_e}.$$

Por reducción al absurdo, supóngase que

$$|G| = n < \left\lceil \frac{\sum_{e \in E} 2^{w_e}}{2^r} \right\rceil,$$

entonces

$$n < \frac{\sum_{e \in E} 2^{w_e}}{2^r} = \frac{\sum_{e \in g_1} 2^{w_e} + \cdots + \sum_{e \in g_n} 2^{w_e}}{2^r} \leq \frac{\Delta(e_{g_1}^G) + \cdots + \Delta(e_{g_n}^G)}{2^r}.$$

Esto implica que existe un $e_g^G \in E^G$ tal que $\Delta(e_g^G) > 2^r$, lo que contradice la ecuación 3.4. Por lo tanto, se puede concluir que

$$|G| \geq \left\lceil \frac{\sum_{e \in E} 2^{w_e}}{2^r} \right\rceil.$$

□

Debido a que los algoritmos de agrupación de estados trabajan con una MSE (en lugar de una MSEE), es necesario redefinir la función tamaño de FSMIM. En el caso de que el tamaño de la ROM se mida en bits, para cada arquitectura y para cada FSMIM $F = (E, P, t, h, p_0)$ con $r$ canales de selección, se define una función $TAM : MSE^{|P| \times r} \times \{1, \ldots, |P|\} \to \mathbb{R}_+$, donde $TAM(A, k)$ es el tamaño de la ROM necesaria para implementar $F$ con la selección de entradas especificada en $A \in MSE^{|P| \times r}$ y con $k$ grupos (es decir, con $k$ estados). Esta función determina el coste del objetivo de optimización que debe ser minimizado por los algoritmos de agrupación. Y se define a partir de las ecuaciones 2.26, 2.28 y 2.29, o de la ecuación 2.30, donde $|E(A)|$ es el valor $k$. En el caso de que el tamaño de la ROM se mida en número de bloques de memoria, las características de los bloques de memoria también intervienen en la definición la función $TAM$.

## 3.2. Algoritmo de agrupación voraz básico

Se denominará *algoritmo de agrupación voraz básico* al algoritmo de agrupación utilizado para el estudio experimental en [Senhadji-Navarro et al., 2004; García-Vargas et al., 2007] y presentado en [García-Vargas, 2006].

Este algoritmo se diseñó para ser aplicado en una estrategia de optimización de FSMIM de dos fases compuesta por una fase inicial de minimización de la selección de entradas y una fase final de agrupación de estados. Con el fin de minimizar el coste de selección de entradas, la fase inicial determina la posición definitiva de las entradas de la MSE. La fase de agrupación final no debe realizar ninguna permutación de entradas para no alterar la solución encontrada por la fase previa. Por lo tanto, este algoritmo sólo realiza operaciones de asignación de indeterminaciones y agrupación de estados sobre la MSEE de partida (es decir, sobre la FSMIM).

El algoritmo de agrupación voraz básico obtiene soluciones al subproblema de agrupación de estados utilizando el procedimiento descrito en el algoritmo 1. Este procedimiento es un algoritmo voraz que procesa los elementos de la MSE en un orden preestablecido para determinar los grupos de estados.

**Algoritmo 1** Procedimiento de Agrupación Voraz con Permutación Fija (AVPF)

**Entrada:** $A = (a_{i,j}) \in MSE^{n \times r}$
$\pi_f : \{1, \ldots, n\} \to \{1, \ldots, n\}$ ▷ *Ordenación de filas.*
$\pi_c : \{1, \ldots, r\} \to \{1, \ldots, r\}$ ▷ *Ordenación de columnas.*
**Salida:** Una tupla $(A^H, G^H)$ donde $G^H$ es una partición de $\{1, \ldots, n\}$

**procedimiento** AVPF($A, \pi_f, \pi_c$)
  $G^H \leftarrow \{\{1\}, \ldots, \{n\}\}$ ▷ *Inicialmente cada fila es un grupo.*
  $A^H \leftarrow A$
  **para** $j \leftarrow 1$ **hasta** $r$ **hacer**
    $k \leftarrow \pi_c(j)$ ▷ *Procesamos la columna $k$.*
    $G \leftarrow \{g \in G^H : I_k(A^H, \{\pi_f(e) : e \in g\}) = 1\}$
    Sean $g_1, \ldots, g_{|G|}$ los elementos de $G$ en orden creciente de mín $g_i$.
    **para** $i \leftarrow 1$ **hasta** $\lfloor \frac{|G|}{2} \rfloor$ **hacer**
      $a^H_{\pi_f(e),k} \leftarrow 0$ para todo $e \in g_{2i-1}$
      $a^H_{\pi_f(e),k} \leftarrow 1$ para todo $e \in g_{2i}$
      $G^H \leftarrow (G^H \setminus \{g_{2i-1}, g_{2i}\}) \cup \{g_{2i-1} \cup g_{2i}\}$
    **fin para**
  **fin para**
  **devolver** $(A^H, \{\{\pi_f(e) : e \in g\} : g \in G^H\})$
**fin procedimiento**

Se denominará *ordenación de filas o de columnas* de una MSE a una permutación de sus índices de filas o de columnas, es decir, a una función biyectiva $\pi : \{1 \ldots, n\} \to \{1, \ldots, n\}$, donde $n$ es el número de filas o columnas. Dada una MSE, una ordenación de filas y una ordenación de columnas, el algoritmo 1 devuelve como resultado, por una parte, una nueva MSE en la que se ha realizado una asignación de indeterminaciones y, por otra, los grupos de estados que dicha asignación ha transformado en seudoequivalentes. En este y en el resto de algoritmos presentados en esta sección, $A^H$ y $G^H$ denotan la MSE y el conjunto de grupos, respectivamente, obtenidos por la heurística que implementa el algoritmo.

Dada una MSE, una ordenación de filas $\pi_f$ y una ordenación de columnas

$\pi_c$, el algoritmo puede ser resumido como sigue. Inicialmente, cada estado (o índice de la MSE) forma un único grupo (es decir, el número de grupos inicial es igual al número de estados de la FSMIM). El algoritmo procesa las columnas de la MSE de una en una en el orden establecido por $\pi_c$, obteniendo un nuevo conjunto de grupos tras procesar cada columna. Para cada columna, se ordenan los grupos en un orden que depende de $\pi_f$ y se seleccionan pares de grupos que pueden ser agrupados aprovechando las indeterminaciones de la columna.

**Ejemplo 3.4.** *Con el fin de ilustrar el algoritmo 1 se considerará la MSE*

$$A = \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \left[\begin{array}{ccc} - & \bullet & \bullet \\ - & \bullet & - \\ - & - & - \\ - & - & \bullet \\ \bullet & \bullet & \bullet \\ - & \bullet & - \\ \bullet & \bullet & - \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array},$$

*donde el símbolo $\bullet$ representa a los elementos no indeterminados de A (es decir, a las entradas efectivas); la ordenación de filas*[1]

$$\pi_f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 5 & 7 & 1 & 2 & 4 & 6 & 3 \end{pmatrix},$$

*que ordena las filas de A en orden creciente del número de indeterminaciones; y la ordenación de columnas*

$$\pi_c = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix},$$

*que ordena las columnas de A en orden creciente del numero de indetermi-*

[1] En la representación matricial utilizada para las permutaciones, la primera fila representa los elementos del dominio de la función, y la segunda fila, los elementos de la imagen.

*naciones. Cuando se aplican ambas ordenaciones a A, se obtiene la MSE*

$$A' = \begin{array}{c} \begin{array}{ccc} 2 & 3 & 1 \end{array} \\ \left[\begin{array}{ccc} \bullet & \bullet & \bullet \\ \bullet & - & \bullet \\ \bullet & \bullet & - \\ \bullet & - & - \\ - & \bullet & - \\ \bullet & - & - \\ - & - & - \end{array}\right] \end{array} \begin{array}{c} 5 \\ 7 \\ 1 \\ 2 \\ 4 \\ 6 \\ 3 \end{array},$$

*que muestra el orden en el que serán procesados los elementos de A (las columnas serán procesadas de izquierda a derecha, y cada columna se recorrerá de arriba abajo).*

*Para facilitar la descripción del algoritmo, la MSE A será representada por la MSEE*

$$A_0^H = \left[\begin{array}{ccc|c} \bullet & \bullet & \bullet & 1 \\ \bullet & - & \bullet & 2 \\ \bullet & \bullet & - & 3 \\ \bullet & - & - & 4 \\ - & \bullet & - & 5 \\ \bullet & - & - & 6 \\ - & - & - & 7 \end{array}\right],$$

*en la que las filas y columnas se han ordenado como en $A'$, y en la que los estados están representados por los índices de filas de $A_0^H$. Por lo tanto, la fila $k$ de $A_0^H$ representa, por una parte, a la fila $\pi_f(k)$ de A y, por otra, al estado asociado a dicha fila. La columna $j$ representa a la columna $\pi_c(j)$ de A.*

*Inicialmente, cada estado de la MSE forma un grupo de un sólo elemento, por lo tanto, inicialmente $G^H = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\}\}$.*

*El algoritmo procesa las columnas en el orden establecido por $\pi_c$ (línea 5 del algoritmo 1), seleccionando para cada columna el conjunto de grupos que tienen dicha columna indeterminada (línea 6); es decir, los grupos en los que todos los estados tienen una ESI en dicha columna. Para la columna 1, se seleccionan los grupos $\{5\}$ y $\{7\}$; por lo tanto $G = \{\{5\},\{7\}\}$.*

*Los grupos seleccionados en G se ordenan en el orden establecido por $\pi_f$ (ya que se ordenan por el índice de fila mínimo de cada grupo) y se agrupan por pares hasta que quedan menos de dos grupos (líneas de la 8 a la 12). Para*

*cada par de grupos se actualiza la solución* $(A^H, G^H)$*: (a) en la columna de* $A^H$ *correspondiente se asigna el valor 0 a la ESI de todos los estados de uno de los grupos y el valor 1 a la de todos los estados del otro grupo (líneas 9 y 10); y (b) se sustituye el par de grupos de* $G^H$ *por un nuevo grupo que es la unión de ambos (línea 11).*

*Tras la agrupación de los estados 5 y 7, seleccionados en la columna 1, se obtienen los grupos* $G_1^H = \{\{1\},\{2\},\{3\},\{4\},\{5,7\},\{6\}\}$ *y la MSEE*

$$A_1^H = \left[\begin{array}{ccc|c} \bullet & \bullet & \bullet & 1 \\ \bullet & - & \bullet & 2 \\ \bullet & \bullet & - & 3 \\ \bullet & - & - & 4 \\ 0 & \bullet & - & 5 \oplus 7 \\ \bullet & - & - & 6 \\ 1 & \times & - & 5 \oplus 7 \end{array}\right],$$

*en la que se la asignación de constantes en la columna 1 permite agrupar los estados 5 y 7 en el estado* $5 \oplus 7$*. El símbolo* $\times$ *en la última fila de* $A_1^H$ *indica que la ESI correspondiente es una indeterminación anulada (nótese que ya no puede ser aprovechada por el algoritmo debido a que la columna 2 no está indeterminada en todas las filas del estado* $5 \oplus 7$*)*

*Para la columna 2 se seleccionan los grupos* $G = \{\{2\},\{4\},\{6\}\}$ *y, tras la agrupación de* $\{2\}$ *y* $\{4\}$ *se obtienen los grupos* $G_2^H = \{\{1\},\{2,4\},\{3\},\{5,7\},\{6\}\}$ *y la MSEE*

$$A_2^H = \left[\begin{array}{ccc|c} \bullet & \bullet & \bullet & 1 \\ \bullet & 0 & \bullet & 2 \oplus 4 \\ \bullet & \bullet & - & 3 \\ \bullet & 1 & \times & 2 \oplus 4 \\ 0 & \bullet & - & 5 \oplus 7 \\ \bullet & - & - & 6 \\ 1 & \times & - & 5 \oplus 7 \end{array}\right].$$

*Puesto que, tras esta agrupación, sólo queda el grupo* $\{6\}$ *en* $G$*, se termina el procesamiento de la columna 2.*

*Para la columna 3 se seleccionan los grupos* $G = \{\{3\},\{5,7\},\{6\}\}$ *(línea 6) y se ordenan por su valor mínimo, dando como resultado* $\text{mín}\{3\} < \text{mín}\{5,7\} < \text{mín}\{6\}$ *(línea 7). Esta ordenación permite que los grupos se procesen en el orden establecido por la permutación* $\pi_f$*. Por tanto, en la primera iteración del bucle se agrupan* $\{3\}$ *y* $\{5,7\}$*. El algoritmo asigna en la columna 3 un 0 a la fila 3 (es decir, al estado 3) y un 1 a las filas 5 y 7 (es decir, a todas las selecciones de entradas del estado* $5 \oplus 7$*).*

*Puesto que sólo queda el grupo* $\{6\}$ *en* $G$ *tras esta agrupación, se termina el procesamiento de la columna 3. Como resultado se obtienen los grupos* $G_3^H = \{\{1\},\{2,4\},\{3,5,7\},\{6\}\}$ *y la MSEE*

$$A_3^H = \left[\begin{array}{ccc|c} \bullet & \bullet & \bullet & 1 \\ \bullet & 0 & \bullet & 2 \oplus 4 \\ \bullet & \bullet & 0 & 3 \oplus 5 \oplus 7 \\ \bullet & 1 & \times & 2 \oplus 4 \\ 0 & \bullet & 1 & 3 \oplus 5 \oplus 7 \\ \bullet & - & - & 6 \\ 1 & \times & 1 & 3 \oplus 5 \oplus 7 \end{array}\right].$$

*La solución devuelta por el algoritmo es*

$$(A^H, G^H) = \left(\left[\begin{array}{ccc} 0 & \bullet & \bullet \\ - & \bullet & 1 \\ 1 & 1 & - \\ 1 & 0 & \bullet \\ \bullet & \bullet & \bullet \\ - & \bullet & - \\ \bullet & \bullet & 0 \end{array}\right], \{\{5\},\{7,2\},\{1,4,3\},\{6\}\}\right),$$

*donde* $A^H$ *es la MSE* $A$ *con la asignación de indeterminaciones encontradas, y* $G^H$ *son los grupos de estados de* $A$ *(representados por las filas de* $A$*). Nótese que* $A^H$ *es la matriz* $A_3^H$ *con el orden original de filas y columnas, y que* $G^H$ *es el resultado de aplicar la permutación* $\pi_f$ *a los elementos de* $G_3^H$*.*

*En la solución encontrada, el número de grupos de la MSE (o de estados de la FSMIM) es 4. Sin embargo, teniendo en cuenta el número de indeterminaciones de las filas de* $A$*, por la Proposición 3.3 se tiene que la cota inferior del número de grupos que se pueden obtener es* $\lceil \frac{2^2+2^1+2^0+2^1+2^3+2^1+2^2}{2^3} \rceil = \lceil \frac{23}{8} \rceil = 3$ *(los términos del numerador aparecen en el mismo orden que las filas de* $A^H$*).*

La bondad de la solución obtenida por el algoritmo 1 depende de la ordenación de filas y columnas utilizada. Por lo tanto, una heurística simple para intentar encontrar mejores soluciones consiste en probar diferentes ordenaciones y seleccionar la mejor solución obtenida. Esta es la idea en la que se basa el algoritmo voraz básico (algoritmo 2), que calcula cuatro soluciones utilizando el algoritmo 1 con cuatro ordenaciones diferentes (que ordenan las filas y columnas en orden creciente o decreciente del número de indeterminaciones) y devuelve la solución de menor coste medido como el valor de la función *TAM*. Estas cuatro ordenaciones permiten mejorar la solución encontrada, pero no garantizan la obtención del óptimo.

**Algoritmo 2** Algoritmo de agrupación voraz básico.

**Entrada:**
$A = (a_{i,j}) \in MSE^{n \times r}$
$TAM : MSE^{n \times r} \times \{1, \ldots, n\} \to \mathbb{R}_+$ ▷ *Función tamaño de FSMIM.*
**Salida:** Una tupla $(A^H, G^H)$ donde $G^H$ es una partición de $\{1, \ldots, n\}$
1: Sea $\pi_c^{asc} : \{1, \ldots, r\} \to \{1, \ldots, r\}$ una ordenación de las columnas de $A$ en orden ascendente del número de indeterminaciones.
2: Sea $\pi_c^{des} : \{1, \ldots, r\} \to \{1, \ldots, r\}$ la ordenación inversa de $\pi_c^{asc}$.
3: Sea $\pi_f^{asc} : \{1, \ldots, n\} \to \{1, \ldots, n\}$ una ordenación de las filas de $A$ en orden ascendente del número de indeterminaciones.
4: Sea $\pi_f^{des} : \{1, \ldots, n\} \to \{1, \ldots, n\}$ la ordenación inversa de $\pi_f^{asc}$.
5: $S \leftarrow \emptyset$
6: **para todo** $(\pi_f, \pi_c) \in \{\pi_f^{asc}, \pi_f^{des}\} \times \{\pi_c^{asc}, \pi_c^{des}\}$ **hacer**
7: $(A^H, G^H) \leftarrow \text{AVPF}(A, \pi_f, \pi_c)$ ▷ *algoritmo 1*
8: S $\leftarrow S \cup \{(A^H, G^H)\}$
9: **fin para**
10: **devolver** $\underset{(A^H, G^H) \in S}{\arg\min} \; TAM(A^H, |G^H|)$

## 3.3. Algoritmo de agrupación lexicográfico

Se denominará *número de coincidencias de indeterminaciones* entre dos grupos al número de columnas que están indeterminadas en ambos grupos. En términos generales, buscar una ordenación de filas de la MSE que aumente el número de coincidencias de indeterminaciones entre filas consecutivas permite mejorar los resultados del algoritmo 1. Mientras mayor sea el número de coincidencias de indeterminaciones entre los pares de grupos que selecciona el algoritmo en cada paso, mayor será el número de columnas indeterminadas en los nuevos grupos obtenidos. Por otra parte, aquellas columnas que no estén indeterminadas en los dos grupos de cada par, darán lugar a indeterminaciones anuladas. Obviamente, un mayor número de columnas indeterminadas en los nuevos grupos que se generan en cada paso favorece la agrupación posterior.

Los criterios de ordenación empleados en el algoritmo 2 presentan dos problemas: por una parte, en el caso de que varias filas tengan el mismo número de indeterminaciones, estas quedarán ordenadas de forma arbitraria; y, por otra, no garantiza necesariamente que el número de coincidencias de indeterminaciones entre dos grupos consecutivos sea elevado.

**Ejemplo 3.5.** *Los problemas mencionados pueden observarse en este sencillo ejemplo. Las filas y columnas de la MSE $A \in MSE^{6\times5}$ están ordenadas en orden decreciente del número de indeterminaciones:*

$$A = \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} - & - & - & \bullet & \bullet \\ - & \bullet & \bullet & - & - \\ - & \bullet & - & \bullet & \bullet \\ \bullet & - & \bullet & - & \bullet \\ - & - & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}.$$

*Las filas 1 y 2 presentan una única coincidencia de indeterminaciones (en la columna 1) a pesar de que ambas filas contienen tres indeterminaciones. Respecto a la ordenación de filas con el mismo número de indeterminaciones, por una parte, las filas 1 y 2, y por otra, las filas 3, 4 y 5 han sido ordenadas de forma arbitraria. Si el orden de estos grupos de filas fuera*

$$A' = \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} - & \bullet & \bullet & - & - \\ - & - & - & \bullet & \bullet \\ - & - & \bullet & \bullet & \bullet \\ - & \bullet & - & \bullet & \bullet \\ \bullet & - & \bullet & - & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{array}\right] \end{array} \begin{array}{c} 2 \\ 1 \\ 5 \\ 3 \\ 4 \\ 6 \end{array},$$

*se conseguiría aumentar el número de coincidencias entre las filas etiquetadas como 1 y 5, que quedarían consecutivas. De todas formas, en este ejemplo dicha ordenación no consigue mejorar el resultado final, ya que el algoritmo comienza agrupando las dos primeras filas.*

*Si se aplica el algoritmo 1 a la matriz A, tras procesar la primera columna, se obtienen los grupos $G_1^H = \{\{1,2\},\{3,5\},\{4\},\{6\}\}$ y la MSEE*

$$A_1^H = \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc|c} 0 & \times & \times & \bullet & \bullet & 1 \oplus 2 \\ 1 & \bullet & \bullet & \times & \times & 1 \oplus 2 \\ 0 & \bullet & \times & \bullet & \bullet & 3 \oplus 5 \\ \bullet & - & \bullet & - & \bullet & 4 \\ 1 & \times & \bullet & \bullet & \bullet & 3 \oplus 5 \\ \bullet & \bullet & \bullet & \bullet & \bullet & 6 \end{array}\right] \end{array}.$$

*Debido a que sólo había una coincidencia de indeterminaciones en las filas 1 y 2, el grupo* $\{1, 2\}$ *no tiene ninguna columna indeterminada, por lo que no puede seguir agrupándose con el resto de grupos. De igual manera, el grupo* $\{3, 5\}$ *tampoco tiene columnas indeterminadas. Como consecuencia, el algoritmo no puede realizar más agrupaciones y termina con el resultado* $(A_1^H, G_1^H)$.

Con el fin de evitar los problemas anteriores y mejorar los resultados del algoritmo 1, propusimos un nuevo criterio de ordenación para el algoritmo algoritmo 2, dando lugar al algoritmo 3. Este algoritmo se incluyó en la herramienta FSMIM-Gen 1.2, utilizada para realizar el estudio experimental en [García-Vargas y Senhadji-Navarro, 2015].

**Algoritmo 3** Algoritmo de agrupación voraz con ordenación lexicográfica.

**Entrada:**
$A = (a_{i,j}) \in MSE^{n \times r}$
$TAM : MSE^{n \times r} \times \{1, \ldots, n\} \to \mathbb{R}_+$ ▷ *Función tamaño de FSMIM.*
**Salida:** Una tupla $(A, G)$ donde $G$ es una partición de $\{1, \ldots, n\}$
1: Sea $\pi_c^{asc} : \{1, \ldots, r\} \to \{1, \ldots, r\}$ una ordenación de las columnas de $A$ en orden ascendente del número de indeterminaciones.
2: Sea $\pi_c^{des} : \{1, \ldots, r\} \to \{1, \ldots, r\}$ la ordenación inversa de $\pi_c^{asc}$.
3: $S \leftarrow \emptyset$
4: **para todo** $\pi_c \in \{\pi_c^{asc}, \pi_c^{des}\}$ **hacer**
5: Sea $\pi_f^{asc} : \{1, \ldots, n\} \to \{1, \ldots, n\}$ una ordenación de las filas de $A$ en orden lexicográfico ascendente de las tuplas $T_i = (I_{\pi_c(1)}(A, \{i\}), \ldots, I_{\pi_c(r)}(A, \{i\}))$, donde $T_i$ es la tupla asociada a la fila $i$.
6: Sea $\pi_f^{des} : \{1, \ldots, n\} \to \{1, \ldots, n\}$ la ordenación inversa de $\pi_f^{asc}$.
7: **para todo** $\pi_f \in \{\pi_f^{asc}, \pi_f^{des}\}$ **hacer**
8: $(A^H, G^H) \leftarrow \text{AVPF}(A, \pi_f, \pi_c)$ ▷ *algoritmo 1*
9: S $\leftarrow S \cup \{(A^H, G^H)\}$
10: **fin para**
11: **fin para**
12: **devolver** $\underset{(A^H, G^H) \in S}{\arg\min} \; TAM(A^H, |G^H|)$

El algoritmo 3 se diferencia del algoritmo 2 tan solo en la ordenación de filas utilizada. Dada una MSE $A = (a_{i,j}) \in MSE^{n \times r}$ y una ordenación de columnas $\pi_c$, el algoritmo 3 calcula para cada fila $i$ de $A$ la tupla $T_i =$

$(I_{\pi_c(1)}(A,\{i\}),\ldots,I_{\pi_c(r)}(A,\{i\})) \in \mathbb{B}^r$, cuyo elemento de índice $j$ vale 1 si $a_{i,\pi_c(j)}$ es una ESI, o 0 si es una entrada efectiva.

Las filas son ordenadas en orden lexicográfico[2] creciente o decreciente de las tuplas $T_i$. Esta ordenación tiende a concentrar en filas consecutivas las indeterminaciones que se encuentran en las columnas con más prioridad (aquellas que son procesadas primero por el algoritmo 1 debido a la ordenación de columnas $\pi_c$). En términos generales, la ordenación lexicográfica consigue que el número de coincidencias de indeterminaciones entre filas consecutivas sea mayor que el obtenido por la ordenación del algoritmo 2.

**Ejemplo 3.6.** *Si se aplica la ordenación lexicográfica del algoritmo 3 a las filas de la MSE A del Ejemplo 3.5, se obtiene la MSE*

$$A^L = \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc} - & - & - & \bullet & \bullet \\ - & - & \bullet & \bullet & \bullet \\ - & \bullet & - & \bullet & \bullet \\ - & \bullet & \bullet & - & - \\ \bullet & - & \bullet & - & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{array}\right] \end{array} \begin{array}{c} 1 \\ 5 \\ 3 \\ 2 \\ 4 \\ 6 \end{array}.$$

*Se puede observar cómo el número de coincidencias de indeterminaciones de las dos primeras filas es mayor en $A^L$ que en $A$.*

*Tras procesar la primera columna de $A^L$, el algoritmo 1 obtiene los grupos $G_1^H = \{\{1,5\},\{3,2\},\{4\},\{6\}\}$ y la MSEE*

$$A_1^H = \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{ccccc|c} 0 & - & \times & \bullet & \bullet & 1 \oplus 5 \\ 1 & - & \bullet & \bullet & \bullet & 1 \oplus 5 \\ 0 & \bullet & \times & \bullet & \bullet & 3 \oplus 2 \\ 1 & \bullet & \bullet & \times & \times & 3 \oplus 2 \\ \bullet & - & \bullet & - & \bullet & 4 \\ \bullet & \bullet & \bullet & \bullet & \bullet & 6 \end{array}\right] \end{array}.$$

*Debido a que el número de coincidencias de indeterminaciones de las filas etiquetadas 1 y 5 era dos, ahora el grupo $\{1,5\}$ tiene la columna 2 indeterminada, lo que permite al algoritmo 1 aprovechar la dicha columna*

[2] Sean $x, y \in \mathbb{N}^r$ dos tuplas. Se dice que $x$ es lexicográficamente menor que $y$ si existe un índice $j \in \{1,\ldots,r\}$ tal que $x_i = y_i$ para $i = 1,\ldots,j-1$ y $x_j < y_j$.

*para formar un nuevo grupo. Como resultado final se obtienen los grupos* $G_2^H = \{\{1,5,4\},\{3,2\},\{6\}\}$ *y la MSEE*

$$A_2^H = \left[\begin{array}{ccccc|c} 1 & 2 & 3 & 4 & 5 & \\ 0 & 0 & \times & \bullet & \bullet & 1 \oplus 5 \oplus 4 \\ 1 & 0 & \bullet & \bullet & \bullet & 1 \oplus 5 \oplus 4 \\ 0 & \bullet & \times & \bullet & \bullet & 3 \oplus 2 \\ 1 & \bullet & \bullet & \times & \times & 3 \oplus 2 \\ \bullet & 1 & \bullet & - & \bullet & 1 \oplus 5 \oplus 4 \\ \bullet & \bullet & \bullet & \bullet & \bullet & 6 \end{array}\right].$$

*Por lo tanto, con la nueva ordenación del algoritmo 3 se consigue reducir el número de grupos finales respecto a la ordenación del algoritmo 2.*

*En este caso tampoco se alcanza la cota inferior del número de grupos, que es igual a* $\lceil \frac{2^2+2^3+2^3+2^2+2^3+2^5}{2^5} \rceil = \lceil \frac{64}{32} \rceil = 2$ *(los términos del numerador aparecen en el mismo orden que las filas de* $A_2^H$*).*

## 3.4. Algoritmo de agrupación lexicográfico con control del tamaño de la FSMIM

Los Algoritmos 2 y 3 establecen una ordenación de las filas y columnas de la MSE previa al procedimiento de agrupación (realizado por el algoritmo 1). Como se ha estudiado, esta ordenación previa depende de características como el número de indeterminaciones de las filas y columnas, o de la posición de dichas indeterminaciones. Sin embargo, durante el procedimiento de agrupación, cada vez que se forma un grupo nuevo se pueden anular algunas indeterminaciones, modificándose las características que han determinado la ordenación inicial. Como consecuencia, el orden de las columnas o de los grupos (de filas) puede dejar de coincidir con el criterio de ordenación utilizado inicialmente.

Por otra parte, el algoritmo 1 devuelve el mínimo número de grupos que puede alcanzar. Sin embargo, tal como se estudió en la sección 2.4.2.1, la solución óptima para la arquitectura FSMIM-T (es decir, la ROM de tamaño mínimo) no se obtiene necesariamente con el mínimo número de grupos. Esto significa que reducir el número de grupos por debajo del valor correspondiente a la solución óptima puede aumentar el tamaño de la ROM. Además, en ambas arquitecturas FSMIM, el mínimo número de grupos puede no ser el valor más conveniente cuando se implementa la ROM con bloques de memoria empotrados. En estos casos, debido a la discretización de la función

*TAM*, la solución óptima no se obtiene para un único valor del número de grupos, sino para un rango de valores. En términos generales, puesto que la agrupación puede repercutir negativamente en la complejidad del banco de selectores de la FSMIM, el mayor valor de dicho rango es el que permite obtener mejores soluciones al problema de optimización de FSMIM.

Con el fin de solucionar los problemas mencionados, se propone en esta tesis el algoritmo 4 como una mejora del algoritmo 1. Este ha sido uno de los algoritmos implementados en la nueva herramienta de generación de FSMIM (FSMIM-Gen 2.0), utilizada en el estudio experimental que se presenta en esta memoria. A continuación se indican las diferencias principales con el algoritmo 1:

- Antes de procesar cada columna, el algoritmo ordena las columnas y los grupos.
    - Las líneas 5 y 18 seleccionan las columnas que están indeterminadas en más de un grupo. La línea 7 ordena las columnas seleccionadas en orden creciente o decreciente (según el parámetro $\text{crec}_c$) del número de grupos en los que cada columna está indeterminada (nótese que las indeterminaciones anuladas no son tenidas en cuenta para la ordenación). La primera columna en el orden establecido ($c_1$) será la columna que se procesará en el bucle interno (líneas 10 a 17).
    - La línea 8 selecciona los grupos que tienen la columna $c_1$ indeterminada. Estos serán los grupos procesados en el bucle interno. La línea 9 ordena lexicográficamente los grupos seleccionados de forma similar al algoritmo 3 pero incluyendo en las tuplas solamente información de las columnas seleccionadas y ordenadas previamente. Serán ordenados en orden creciente o decreciente en función del valor del parámetro $\text{crec}_f$.
- Al igual que el algoritmo 1, el algoritmo 4 procesa la MSE completa para realizar todas las agrupaciones posibles. Sin embargo, se devuelve la mejor de todas las soluciones encontradas: aquella que minimiza la función *TAM* usando el mayor número de grupos. Para conseguirlo, después de realizar cada agrupación se actualiza la mejor solución encontrada ($A^H, G^H$) sólo si el coste de la nueva solución obtenida es menor (líneas 14 y 15).

Aunque el criterio de ordenación de columnas y grupos viene determinado por el propio algoritmo, los parámetros $\text{crec}_f$ y $\text{crec}_c$ determinan el orden

**Algoritmo 4** Procedimiento de agrupación lexicográfico con control de tamaño de la FSMIM (ALCT).

**Entrada:**
$A = (a_{i,j}) \in MSE^{n \times r}$
$TAM : MSE^{n \times r} \times \{1, \ldots, n\} \to \mathbb{R}_+$ ▷ *Función tamaño de FSMIM.*
$\text{crec}_f \in \{verdadero, falso\}$
$\text{crec}_c \in \{verdadero, falso\}$
**Salida:** Una tupla $(A^H, G^H)$ donde $G^H$ es una partición de $\{1, \ldots, n\}$

**procedimiento** ALCT$(A, t, \text{crec}_f, \text{crec}_c)$
  $G' \leftarrow \{\{1\}, \ldots, \{n\}\}$ ▷ *Inicialmente cada fila forma un grupo.*
  $A' \leftarrow A$
  $(A^H, G^H) \leftarrow (A', G')$ ▷ *Inicialmente la mejor solución es* $(A', G')$.
  $C = \{c : 1 \leq c \leq r,\ \sum_{g \in G} I_c(A', g) > 1\}$
  **mientras** $C \neq \emptyset$ **hacer**
    Sean $c_1, \ldots, c_k$ los elementos de $C$ ordenados en orden creciente de $\sum_{g \in G'} I_{c_i}(A', g)$ si $\text{crec}_c = verdadero$, o en orden decreciente si $\text{crec}_c = falso$.
    $G^L \leftarrow \{g \in G' : I_{c_1}(A', g) = 1\}$
    Sean $g_1, \ldots, g_{|G^L|}$ los elementos de $G^L$ en orden lexicográfico creciente de las tuplas $T_{g_i} = (I_{c_1}(A', g_i), \ldots, I_{c_k}(A', g_i))$ si $\text{crec}_f = verdadero$, o en orden decreciente si $\text{crec}_f = falso$.
    **para** $i \leftarrow 1$ **hasta** $\lfloor \frac{|G^L|}{2} \rfloor$ **hacer**
      $a'_{s,c_1} \leftarrow 0$ para todo $s \in g_{2i-1}$
      $a'_{s,c_1} \leftarrow 1$ para todo $s \in g_{2i}$
      $G' \leftarrow (G' \setminus \{g_{2i-1}, g_{2i}\}) \cup \{g_{2i-1} \cup g_{2i}\}$
      **si** $TAM(A', |G'|) < TAM(A^H, |G^H|)$ **entonces**
        $(A^H, G^H) \leftarrow (A', G')$ ▷ *Encontrada una solución mejor.*
      **fin si**
    **fin para**
    $C = \{c : 1 \leq c \leq r,\ \sum_{g \in G'} I_c(A', g) > 1\}$
  **fin mientras**
  **devolver** $(A^H, G^H)$
**fin procedimiento**

(creciente o decreciente) en el que serán procesadas las filas (o grupos de filas) y las columnas de la MSE. Esto permite aplicar la misma heurística de los Algoritmos 2 y 3, que consiste en probar diferentes ordenaciones y seleccionar la mejor solución obtenida.

El criterio de selección utilizado por los Algoritmos 2 y 3 se basa exclusivamente en la función $TAM$. Esto es debido a que fueron diseñados para ser usados en la fase final de agrupación de estados de la estrategia de optimización implementada por la herramienta FSMIM-Gen 1.2[3]. Sin embargo, en las nuevas estrategias de optimización implementadas en FSMIM-Gen 2.0, tras las fases de agrupación de estados siempre se realiza una minimización de la selección de entradas. Por lo tanto, en el caso de que varias soluciones alcancen el menor valor de $TAM$, es conveniente seleccionar la más favorable para la fase de minimización de la selección de entradas posterior.

El algoritmo 5 calcula cuatro soluciones mediante el algoritmo 4 y las ordena en función del tamaño de la ROM y del número de filas de la MSE con indeterminaciones (línea 6). Esta ordenación permite seleccionar las soluciones con tamaño mínimo; de estas, las que presentan el número mayor de filas con al menos una indeterminación; de estas, las que presentan el número mayor de filas con al menos dos indeterminaciones, etc.

**Algoritmo 5** Algoritmo de agrupación lexicográfica con control de tamaño de la FSMIM.

**Entrada:**
$A = (a_{i,j}) \in MSE^{n \times r}$
$TAM : MSE^{n \times r} \times \{1, \ldots, n\} \to \mathbb{R}_+$
**Salida:** Una tupla $(A^H, G^H)$ donde $G^H$ es una partición de $\{1, \ldots, n\}$

1: $S \leftarrow \emptyset$
2: **para todo** $(\text{crec}_f, \text{crec}_c) \in \{verdadero, falso\} \times \{verdadero, falso\}$ **hacer**
3: $(A^H, G^H) \leftarrow \text{ALCT}(A, TAM, \text{crec}_f, \text{crec}_c)$ ▷ *algoritmo 4*
4: $\text{S} \leftarrow S \cup \{(A^H, G^H)\}$
5: **fin para**
6: **devolver** $(A^H, G^H) \in S$ con menor $(TAM(A^H, G^H), -Q_1, \ldots, -Q_r)$ en orden lexicográfico, donde $Q_k$ es el número de filas de $A$ con al menos $k$ indeterminaciones.

[3]Cada algoritmo ha sido implementado en una versión diferente de la herramienta FSMIM-Gen 1.2; sin embargo, todas implementan la misma estrategia de optimización de dos fases.

## 3.5. Algoritmo de agrupación máxima

Los algoritmos estudiados han sido diseñados para ser utilizados en una fase de agrupación de estados precedida por una fase de minimización de selección de entradas. Como consecuencia, para no alterar la solución encontrada en la fase previa, se impone a los algoritmos la restricción de que la posición de las entradas seleccionadas en la MSE no puede ser modificada. El número de grupos que se puede obtener depende obviamente de la posición de las indeterminaciones en la MSE, ya que las indeterminaciones de dos estados deben coincidir en una misma columna para que sea posible su agrupación. Por lo tanto, ningún algoritmo que mantenga esta restricción puede garantizar la obtención del mínimo número de grupos.

El algoritmo que se presenta en esta sección (algoritmo 6) elimina la restricción para poder encontrar el mínimo número de grupos, lo que supone que no puede ser utilizado en una fase de agrupación de estados posterior a una fase de minimización de la selección de entradas. Por este motivo, ha sido incluido en la nueva herramienta FSMIM-Gen 2.0 para implementar una nueva estrategia de optimización con una fase inicial de agrupación de estados.

El algoritmo 6 permuta las entradas de la MSE para que las indeterminaciones se sitúen en los últimos elementos de cada fila, de esta forma consigue maximizar el número de coincidencias de indeterminaciones entre filas. Posteriormente, obtiene una solución que minimiza el número de grupos utilizando el algoritmo 1 sin ordenar las filas y columnas.

**Algoritmo 6** Algoritmo de agrupación máxima.

**Entrada:** $A = (a_{i,j}) \in MSE^{n \times r}$
**Salida:** Una tupla $(A^H, G^H)$ donde $G^H$ es una partición de $\{1, \ldots, n\}$

1: Aplicar a cada fila $i$ de $A$ una permutación de entradas tal que para todo $a_{ij}, a_{ik} \in A$ se cumple que si $a_{ij}$ es una ESI y $a_{ik}$ no es una ESI entonces $j > k$.
2: Sea $\pi_f : \{1, \ldots, n\} \to \{1, \ldots, n\}$ tal que $\pi_f(i) = i$. ▷ *Permutación identidad.*
3: Sea $\pi_c : \{1, \ldots, r\} \to \{1, \ldots, r\}$ tal que $\pi_c(i) = i$. ▷ *Permutación identidad.*
4: **devolver** AVPF$(A, \pi_f, \pi_c)$ ▷ *algoritmo 1*

**Lema 3.7.** *Sea $A \in MSE^{n \times r}$ la MSE que resulta tras aplicar la permutación de entradas de la línea 1 del algoritmo 6. Sea $G_j^H$ el conjunto de grupos*

*obtenido por el algoritmo 1 tras procesar la columna $j$ de $A$ (líneas de la 8 a la 12 del algoritmo 1), y $G_0^H = \{\{1\}, \ldots, \{n\}\}$, el conjunto de grupos inicial. Para todo $1 \leq j \leq r$, se cumple que*

$$|G_j^H| = \left\lceil \frac{|G_{j-1}^H| + c_j}{2} \right\rceil, \tag{3.5}$$

*donde $c_j$ es el número de entradas efectivas de la columna $j$ de $A$.*

***Demostración***. Sea $A_{j-1}^H = (a_{ij})$ la MSE obtenida por el algoritmo 1 tras procesar la columna $j-1$. Sea $i \in \{1, \ldots, n\}$ tal que $a_{ij}$ es una entrada efectiva. Por la permutación de entradas realizada en $A$ (línea 1 del algoritmo 6), para todo $k < j$, se cumple que $a_{ik}$ es una entrada efectiva, lo que implica que la fila $i$ no ha podido ser agrupada en el procesamiento de la columna $k$. Por lo tanto, si $a_{ij}$ es una entrada efectiva, entonces el grupo que contiene la fila $i$ es $\{i\}$.

Sea $F_{j-1} = \{\{i\} : a_{ij} \text{ es una entrada efectiva, } 1 \leq i \leq n\} \subseteq G_{j-1}^H$. De la implicación anterior se tiene que, para todo $g \in G_{j-1}^H \setminus F_{j-1}$, se cumple que $a_{ij}$ es una ESI para todo $k \in g$; por lo tanto, la columna $j$ está indeterminada en $g$, es decir, $I_j(A_{j-1}^H, g) = 1$. Entonces, el conjunto de grupos seleccionados por el algoritmo 1 en la línea 6 para procesar la columna $j$ es $G = G_{j-1}^H \setminus F_{j-1}$. Puesto que $G_{j-1}^H$ es una partición y que $|F_{j-1}| = c_j$, se tiene que $|G| = |G_{j-1}^H| - c_j$. Se puede concluir que el número de grupos obtenido por el algoritmo 1 tras procesar la columna $j$ es igual a $|G_j^H| = \left\lceil \frac{|G|}{2} \right\rceil + |F_{j-1}| = \left\lceil \frac{|G_{j-1}^H| - c_j}{2} \right\rceil + c_j = \left\lceil \frac{|G_{j-1}^H| + c_j}{2} \right\rceil$. □

**Ejemplo 3.8.** *Para ilustrar el algoritmo 6 se utilizará la MSE $A \in MSE^{7\times 3}$ del Ejemplo 3.4. Tras aplicar la permutación de entradas a las filas de $A$ (línea 1), se obtiene la MSE*

$$A = \begin{array}{c} \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{bmatrix} \bullet & \bullet & - \\ \bullet & - & - \\ - & - & - \\ \bullet & - & - \\ \bullet & \bullet & \bullet \\ \bullet & - & - \\ \bullet & \bullet & - \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix},$$

*(nótese que no se ha realizado ninguna permutación de columnas ni de filas). Con esta distribución de indeterminaciones, el algoritmo 1 realiza las agrupaciones que se muestran a continuación.*

*Antes de procesar la primera columna, la MSE y el conjunto de grupos iniciales son* $A_0^H = A$ *y* $G_0^H = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\}\}$*; por lo tanto,* $|G_0^H| = 7$*. Sean* $A_j^H$ *y* $G_j^H$*, con* $1 \leq j \leq 3$*, la MSE y el conjunto de grupos, respectivamente, tras procesar la columna* $j$ *de* $A$*. Sea* $C = (c_1, c_2, c_3) = (6, 3, 1)$*, donde* $c_j$ *es el número de entradas efectivas en la columna* $j$ *de* $A$ *(este valor es constante y no depende del algoritmo). El número de grupos que se puede obtener con la columna 1 es* $\left|G_1^H\right| = \left\lceil \frac{|G_0^H|+c_1}{2} \right\rceil = \left\lceil \frac{7+6}{2} \right\rceil = 7$*. En este caso no se ha podido realizar ninguna agrupación en la columna 1, por lo tanto,* $(A_1^H, G_1^H) = (A_0^H, G_0^H)$*.*

*El número de grupos que se pueden conseguir con el procesamiento de la columna 2 es* $\left|G_2^H\right| = \left\lceil \frac{|G_1^H|+c_2}{2} \right\rceil = \left\lceil \frac{7+3}{2} \right\rceil = 5$*. Como resultado, el algoritmo obtiene el conjunto de grupos* $G_2^H = \{\{1\},\{2,3\},\{4,6\},\{5\},\{7\}\}$ *y la MSEE*

$$A_2^H = \left[\begin{array}{ccc|c} \bullet & \bullet & - & 1 \\ \bullet & 0 & - & 2 \oplus 3 \\ - & 1 & - & 2 \oplus 3 \\ \bullet & 0 & - & 4 \oplus 6 \\ \bullet & \bullet & \bullet & 5 \\ \bullet & 1 & - & 4 \oplus 6 \\ \bullet & \bullet & - & 7 \end{array}\right].$$

*Finalmente, el número de grupos que se obtiene tras procesar la columna 3 es* $\left|G_3^H\right| = \left\lceil \frac{|G_2^H|+c_3}{2} \right\rceil = \left\lceil \frac{5+1}{2} \right\rceil = 3$*, siendo* $G_3^H = \{\{1,2,3\},\{4,6,7\},\{5\}$ *y*

$$A_3^H = \left[\begin{array}{ccc|c} \bullet & \bullet & 0 & 1 \oplus 2 \oplus 3 \\ \bullet & 0 & 1 & 1 \oplus 2 \oplus 3 \\ - & 1 & 1 & 1 \oplus 2 \oplus 3 \\ \bullet & 0 & 0 & 4 \oplus 6 \oplus 7 \\ \bullet & \bullet & \bullet & 5 \\ \bullet & 1 & 0 & 4 \oplus 6 \oplus 7 \\ \bullet & \bullet & 1 & 4 \oplus 6 \oplus 7 \end{array}\right].$$

*Por lo tanto, el resultado del algoritmo 6 es la solución*

$$(A^H, G^H) = \left( \begin{bmatrix} \bullet & \bullet & 0 \\ \bullet & 0 & 1 \\ - & 1 & 1 \\ \bullet & 0 & 0 \\ \bullet & \bullet & \bullet \\ \bullet & 1 & 0 \\ \bullet & \bullet & 1 \end{bmatrix}, \{\{1,2,3\},\{4,6,7\},\{5\}\} \right).$$

*Mientras que el número de grupos de la solución encontrada por el algoritmo 2 para esta MSE era mayor que la cota inferior establecida por la Proposición 3.3 (véase el final del Ejemplo 3.4, página 76), el algoritmo 6 sí logra alcanzar dicha cota.*

**Proposición 3.9.** *El algoritmo 6 siempre alcanza la cota inferior del número de grupos establecida por la Proposición 3.3. Por lo tanto, la ecuación 3.3 determina el número de grupos mínimo que se puede obtener en la fase de agrupación de estados.*

***Demostración.*** Sea $A = (a_{ij}) \in MSE^{n \times r}$ la MSE que resulta tras aplicar la permutación de entradas de la línea 1 del algoritmo 6. Debe demostrarse que el número de grupos obtenido por el algoritmo 1 tras procesar las $r$ columnas de $A$ es igual al valor establecido por la ecuación 3.3; es decir, debe demostrarse que

$$|G_r| = \left\lceil \frac{\sum_{i=1}^{n} 2^{w_i}}{2^r} \right\rceil, \tag{3.6}$$

donde $G_k$ es el conjunto de grupos obtenido por el algoritmo 1 al procesar la columna $k$ de $A$, y $w_i$ es el número de entradas efectivas de la fila $i$ de $A$.

En primer lugar, resulta conveniente la definición de la matriz $F = (f_{i,j}) \in \mathbb{B}^{n \times r}$, donde

$$f_{i,j} = \begin{cases} 1 & \text{si } a_{ij} \text{ es una entrada efectiva} \\ 0 & \text{en otro caso} \end{cases}.$$

A partir de $F$ se puede calcular el valor de $w_i$ y $c_j$ como

$$w_i = \sum_{j=1}^{r} f_{i,j}, \qquad\qquad c_j = \sum_{i=1}^{n} f_{i,j}.$$

Sea $w_i^{(k)} = \sum_{j=1}^{k} f_{i,j}$. Puesto que $w_i = w_i^{(r)}$, la ecuación 3.6 quedará demostrada si

$$|G_k| = \left\lceil \frac{\sum_{i=1}^{n} 2^{w_i^{(k)}}}{2^k} \right\rceil \text{ para } k = 1, \ldots, r, \tag{3.7}$$

que será demostrado por inducción.

Para $k = 1$,

$$\left\lceil \frac{\sum_{i=1}^{n} 2^{w_i^{(1)}}}{2} \right\rceil = \left\lceil \frac{\sum_{i=1}^{n} 2^{f_{i,1}}}{2} \right\rceil.$$

Puesto que $c_1 = \sum_{i=1}^{n} f_{i,1}$, se tiene que $\sum_{i=1}^{n} 2^{f_{i,1}} = (n - c_1)2^0 + c_1 2^1 = n + c_1 = |G_0| + c_1$ (inicialmente cada grupo de $G_0$ está formado por un índice de fila de $A$). Por lo tanto,

$$\left\lceil \frac{\sum_{i=1}^{n} 2^{w_i^{(1)}}}{2} \right\rceil = \left\lceil \frac{|G_0| + c_1}{2} \right\rceil = |G_1|,$$

donde la última igualdad se cumple por el Lema 3.7.

Supóngase ahora que la ecuación 3.7 se cumple para $|G_k|$. Entonces, por el Lema 3.7 se tiene que

$$\begin{aligned}
|G_{k+1}| &= \left\lceil \frac{|G_k| + c_{k+1}}{2} \right\rceil \\
&= \left\lceil \frac{\left\lceil \frac{\sum_{i=1}^{n} 2^{w_i^{(k)}}}{2^k} \right\rceil + c_{k+1}}{2} \right\rceil \\
&= \left\lceil \frac{\left\lceil \frac{2^k c_{k+1} + \sum_{i=1}^{n} 2^{w_i^{(k)}}}{2^k} \right\rceil}{2} \right\rceil \\
&= \left\lceil \frac{2^k c_{k+1} + \sum_{i=1}^{n} 2^{w_i^{(k)}}}{2^{k+1}} \right\rceil \\
&= \left\lceil \frac{2^k \sum_{i=1}^{n} f_{i,k+1} + \sum_{i=1}^{n} 2^{w_i^{(k)}}}{2^{k+1}} \right\rceil \\
&= \left\lceil \frac{\sum_{i=1}^{n} (2^k f_{i,k+1} + 2^{w_i^{(k)}})}{2^{k+1}} \right\rceil
\end{aligned}$$

Para calcular el valor de cada término $2^k f_{i,k+1} + 2^{w_i^{(k)}}$ se tienen dos posibilidades:

- Si $f_{i,k+1} = 0$, entonces $w_i^{(k)} = w_i^{(k+1)}$. Por lo tanto, $2^k f_{i,k+1} + 2^{w_i^{(k)}} = 2^{w_i^{(k)}} = 2^{w_i^{(k+1)}}$.

- Si $f_{i,k+1} = 1$, entonces, por la permutación de entradas de $A$ se cumple que $f_{i,l} = 1$ para todo $l < k + 1$. Esto implica que $w_i^{(k)} = k$ y que $w_i^{(k+1)} = k+1$. Por lo tanto, $2^k f_{i,k+1} + 2^{w_i^{(k)}} = 2^k + 2^k = 2^{k+1} = 2^{w_i^{(k+1)}}$.

En ambos casos se cumple que $2^k f_{i,k+1} + 2^{w_i^{(k)}} = 2^{w_i^{(k+1)}}$; por tanto, se verifica que

$$|G_{k+1}| = \left\lceil \frac{\sum_{i=1}^{n} 2^{w_i^{(k+1)}}}{2^{k+1}} \right\rceil,$$

lo que demuestra la proposición. □

A diferencia del algoritmo 4, que utiliza la función *TAM* para encontrar el número de grupos que optimiza el tamaño de la ROM, el algoritmo 6 siempre obtiene el mínimo número de grupos, a pesar de que dicho valor puede no ser el óptimo respecto al tamaño de la ROM de las FSMIM-T. Esto se debe a que el algoritmo se aplica en una estrategia de optimización que comienza con una fase de agrupación de estados, es decir, antes de minimizar el coste de selección de entradas. En este contexto, el cálculo realizado por la función *TAM* no es adecuado para estimar el tamaño de una FSMIM-T por dos motivos. Por una parte, el ancho de la ROM de la arquitectura FSMIM-T depende del coste de selección de entradas (véase la ecuación 2.26). Por lo tanto, el tamaño de la ROM calculado por *TAM* es mayor que el que tendrá al completarse la estrategia de optimización (es decir, tras la fase posterior de minimización de la selección de entradas). Por otra parte, debido al redondeo por exceso de la ecuación 2.29, el efecto de la la agrupación de estados sobre el ancho de la ROM también depende del coste de selección de entradas. Es decir, el incremento en el ancho producido por la asignación de indeterminaciones dependerá del coste de selección de entradas final, obtenido tras la fase posterior de minimización de la selección de entradas. Como consecuencia, en el caso de la optimización de FSMIM-T, el uso de la función *TAM* puede dar lugar a soluciones demasiado conservadoras respecto al número de grupos (es decir, en las que el número de grupos es muy alto).

En la arquitectura FSMIM-S, puesto que el tamaño de ROM mínimo siempre se obtiene con el número mínimo de estados, el uso de la función *TAM* no es necesario cuando el tamaño de la ROM se mide en bits. Sin embargo, su uso sí es conveniente cuando el tamaño se mide en número de bloques, tal como se estudió en la sección 3.4. En estos casos, es suficiente con

modificar el algoritmo 6 para que la agrupación se realice con el algoritmo 4 en vez del algoritmo 1. Como resultado se obtiene el algoritmo 7.

**Algoritmo 7** Algoritmo de agrupación máxima con control de tamaño.

**Entrada:**
$A = (a_{i,j}) \in MSE^{n \times r}$
$TAM : MSE^{n \times r} \times \{1, \ldots, n\} \to \mathbb{R}_+$
**Salida:** Una tupla $(A^H, G^H)$ donde $G^H$ es una partición de $\{1, \ldots, n\}$
1: Aplicar a cada fila $i$ de $A$ una permutación de entradas tal que para todo $a_{ij}, a_{ik} \in A$ se cumple que si $a_{ij}$ es una ESI y $a_{ik}$ no es una ESI entonces $j > k$.
2: **devolver** ALCT($A$, $TAM, verdadero, verdadero$) ▷ *algoritmo 4*

Suponiendo que $TAM(A, G) = |G|$, donde $A \in MSE^{n \times r}$ y $G$ es una partición de $\{1, \ldots, n\}$, la Proposición 3.9 es válida también para el algoritmo 7, ya que la ordenación de filas y columnas que realiza el algoritmo 4 no afecta al número de grupos obtenido. Es fácil comprobar que la ordenación de columnas en orden creciente del número de indeterminaciones ($\text{crec}_c = verdadero$) mantiene el orden establecido por la permutación de entradas realizada en la línea 1 del algoritmo 7. Por otra parte, el orden en el que se procesan las filas no influye en el número de grupos obtenido (obsérvese que el orden de las filas no es relevante en la demostración de la Proposición 3.9).

Capítulo 4

# Minimización de la selección de entradas

Los algoritmos para la minimización de la selección de entradas que se estudian en este capítulo pueden dividirse en dos categorías: algoritmos basados en el Problema de la Suma de Subconjuntos Compatibles y los algoritmos basados en el Problema del $k$-Emparejamiento Parcial Maximal Mínimo.

A la primera categoría pertenece el algoritmo implementado en la herramienta FSMIM-Gen 1.2 y empleado para el estudio experimental en [García-Vargas et al., 2005, 2007; García-Vargas y Senhadji-Navarro, 2015]. Tanto el algoritmo como el problema de optimización en el que se basa fueron propuestos en [García-Vargas, 2006].

El resto de algoritmos que se proponen en esta tesis, y que han sido implementados en la herramienta FSMIM-Gen 2.0, pertenecen a la segunda categoría. Todos ellos están basado en un nuevo problema de optimización que propusimos en [García-Vargas y Senhadji-Navarro, 2012] como parte del trabajo desarrollado para esta tesis, que ha permitido demostrar que el problema de la minimización de la selección de entradas es *NP*-completo.

Al igual que ocurría con los algoritmos de agrupación de estados estudiados, los algoritmos presentados en este capítulo operan con una MSE en la que cada fila representa un estado simple de la FSMIM o, en el caso del algoritmo que se estudiará en la última sección, un metaestado[1].

[1]La única diferencia entre considerar cada fila de la MSE un metaestado o un estado simple radica en la existencia o no de estados compuestos.

## 4.1. Suma de subconjuntos compatibles

### 4.1.1. El Problema de la Mochila

El Problema de la Mochila [Mathews, 1896; Dantzig, 2007] es un problema de optimización combinatoria que puede ser descrito informalmente como sigue: Dado un conjunto de objetos, cada uno de ellos con un peso y un beneficio asociado, el problema consiste en encontrar la combinación de objetos tal que la suma de sus beneficios sea máxima sin sobrepasar un peso máximo. El nombre del problema procede de una interpretación clásica en la que los objetos deben ser empaquetas en una mochila cuya capacidad está limitada por el peso de los objetos.

Una descripción formal del problema de la mochila es la siguiente: sea $N = \{1, \ldots, n\}$ un conjunto de $n$ objetos con beneficio $p_1, \ldots, p_n$ y peso $w_1, \ldots, w_n$ (donde el beneficio y el peso son valores enteros), y sea $W$ la capacidad máxima; el objetivo es buscar un subconjunto de $N$ tal que el beneficio total de los objetos seleccionados sea máximo y que el peso total no exceda la capacidad $W$. Alternativamente, puede ser formulado como la solución al siguiente problema de *Programación Lineal Entera* (PLE):

$$\begin{aligned} \text{maximizar} \ & \sum_{j=1}^{n} p_j x_j \\ \text{sujeto a} \ & \sum_{j=1}^{n} w_j x_j \leq W, \\ & x_j \in \{0, 1\}, \ j = 1, \ldots, n, \end{aligned} \tag{4.1}$$

donde el valor de $x_j$ indica si el objeto $j$ se selecciona ($x_j = 1$) o no ($x_j = 0$). Por lo tanto, la solución al problema es el conjunto $S = \{j \in N : x_j = 1\}$.

El Problema de la Mochila es *NP*-completo [Kellerer et al., 2004a]. Sin embargo, puede ser resuelto en tiempo seudopolinomial utilizando algoritmos basados en *programación dinámica*, que es una técnica general desarrollada por el matemático Richard Bellman [Bellman, 1954] para resolver problemas complejos que pueden ser divididos en subproblemas. Esta técnica se utiliza en problemas que cumplen el *principio de optimalidad de Bellman*: "dada una secuencia óptima de decisiones, toda subsecuencia de ella es, a su vez, óptima".

Dada una solución óptima del problema de la mochila, es obvio que si eliminamos el objeto $k$ de la solución, el conjunto restante debe ser una solución óptima del subproblema definido por la capacidad $W - w_k$ y el conjunto de objetos $N \setminus \{k\}$. Además, para cualquier otra solución óptima

a dicho subproblema, al añadir $k$ se obtiene una nueva solución óptima al problema completo.

Dado un subproblema determinado por el subconjunto de objetos $\{1, \ldots, j\}$ y una mochila de capacidad $d \leq W$, se define $\mathrm{PM}_j(d)$ como el problema de

$$\begin{aligned} \text{maximizar } & \sum_{l=1}^{n} p_l x_l \\ \text{sujeto a } & \sum_{l=1}^{n} w_l x_l \leq d, \\ & x_l \in \{0,1\},\ l = 1, \ldots, n, \end{aligned}$$

con la solución óptima $z_j(d)$ y con el conjunto óptimo de objetos seleccionados $X_j(d)$.

A partir de los valores de $z_{j-1}(d)$, para todas las capacidades $d = 0, \ldots, W$, podemos calcular la solución óptima de $\mathrm{PM}_j(d)$ mediante la fórmula recursiva

$$z_j(d) = \begin{cases} z_{j-1}(d) & \text{si } d < w_j \\ \text{máx}\{z_{j-1}(d), z_{j-1}(d - w_j) + p_j\} & \text{si } d \geq w_j \end{cases},$$

que se conoce como *recursión de Bellman* [Bellman, 1954].

El caso $d < w_j$ significa que el objeto $j$ no puede formar parte de la solución $z_j(d)$ (ya que su peso excede la capacidad $d$); por lo tanto, la solución óptima será igual a $z_{j-1}(d)$.

En caso contrario, tenemos dos posibles opciones: que el objeto se incluya en la solución o no. Si el objeto no se incluye, la solución $z_j(d)$ será igual a $z_{j-1}(d)$. Si el objeto sí se incluye, entonces el beneficio de la solución se incrementa en $p_j$. Sin embargo, al incluir al objeto $j$ también se incrementa el peso de la solución en $w_j$; por lo tanto, dicho objeto sólo puede incluirse en una solución cuyo peso no sobrepase la capacidad $d - w_j$; es decir, cuya solución óptima sea $z_{j-1}(d - w_j)$. Tomando el valor máximo de ambas opciones, obtenemos la solución óptima $z_j(d)$.

El conjunto de objetos que pertenecen a la solución óptima, $X_j(d)$, se calcula mediante la recursión de Bellman con la siguiente fórmula:

$$X_j(d) = \begin{cases} X_{j-1}(d - w_j) \cup \{j\} & \text{si } z_{j-1}(d - w_j) + p_j > z_{j-1}(d)\,, \\ X_{j-1}(d) & \text{si } z_{j-1}(d - w_j) + p_j \leq z_{j-1}(d) \end{cases}$$

Para obtener la solución del problema de la mochila completo, $z_n(W)$, se debe partir de los valores $z_0(d) = 0$ para $d = 0, \ldots, W$ y calcular los valores

$z_j(0), \ldots, z_j(W)$ para todos los objetos desde $j = 1$ hasta $j = n$ mediante la recursión de Bellman. Como resultado, se obtiene el algoritmo 8, que es el algoritmo básico basado en programación dinámica[2].

El Problema de la Mochila se puede utilizar para modelar una gran variedad de problemas de optimización [Kellerer et al., 2004b]. Entre ellos, el Problema de la Suma de Subconjuntos puede ser considerado un caso especial del Problema de la Mochila que surge cuando el beneficio de cada objeto es igual a su peso.

Dado un conjunto de objetos $N = \{1, \ldots, n\}$ con pesos $w_1, \ldots, w_n$ (valores enteros) y una capacidad $W$; el objetivo del Problema de la Suma de Subconjuntos es buscar un subconjunto de $N$ tal que el peso total de los objetos sea máximo sin exceder la capacidad $W$.

Dicho problema de optimización es equivalente al siguiente modelo de

**Algoritmo 8** Algoritmo basado en programación dinámica para el Problema de la Mochila

**Entrada:**
Un entero $W$
Enteros $w_1, \ldots, w_n$ ▷ *Peso de los objetos.*
Enteros $p_1, \ldots, p_n$ ▷ *Beneficio de los objetos.*
**Salida:** $S \subseteq N$

```
1: z_0(d) ← 0 para d = 1, ..., W
2: para j ← 1 hasta n hacer
3:     z_j(d) ← z_{j-1}(d) para d = 1, ..., w_j − 1
4:     para d ← w_j hasta W hacer
5:         si z_{j-1}(d − w_j) + w_j > z_{j-1}(d) entonces
6:             z_j(d) ← z_{j-1}(d − w_j) + p_j
7:             X_j(d) ← X_{j-1}(d − w_j) ∪ {j}
8:         si no
9:             z_j(d) ← z_{j-1}(d)
10:            X_j(d) ← X_{j-1}(d)
11:        fin si
12:    fin para
13: fin para
14: devolver X_j(d)
```

[2]Aunque existen otros algoritmos más eficiente basados en programación dinámica [Kellerer et al., 2004b], este es el que muestra más claramente la recursión de Bellman.

PLE:

$$\begin{aligned} \text{maximizar } & \sum_{j=1}^{n} w_j x_j \\ \text{sujeto a } & \sum_{j=1}^{n} w_j x_j \leq W, \\ & x_j \in \{0,1\},\ j = 1,\ldots,n. \end{aligned} \tag{4.2}$$

Como puede observarse, el beneficio $p_j$ desaparece de la descripción del problema y de la ecuación 4.2, quedando tan solo el peso $w_j$, el conjunto de objetos y la capacidad $W$.

### 4.1.2. Suma de subconjuntos compatibles

Dado un conjunto de objetos y una relación de compatibilidad[3] entre objetos, el Problema de la Suma de Subconjuntos Compatibles es una generalización del Problema de la Suma de Subconjuntos que restringe los posibles subconjuntos de objetos que se pueden formar, permitiendo sólo subconjuntos de objetos compatibles. Este problema puede ser descrito como el siguiente problema de optimización:

| Problema de la Suma de Subconjuntos Compatibles | |
|---|---|
| *Instancia:* | Un conjunto de $n$ objetos $N = \{1,\ldots,n\}$, $n$ enteros $w_1,\ldots,w_n$, un entero $W$ y una relación de compatibilidad $\rho \subseteq N \times N$. |
| *Tarea:* | Buscar un subconjunto $S \subseteq N$ que maximice $\sum_{i\in S} w_i$, tal que $\langle i,j\rangle \in \rho$ para todo $i,j \in S$ y que $\sum_{i\in S} w_i \leq W$. |

Posteriormente a su publicación en [García-Vargas, 2006], se han propuesto en la literatura generalizaciones similares del Problema de la Mochila[4]. Concretamente, el Problema de la Suma de Subconjuntos Compatibles puede ser considerado un caso especial del Problema de la Mochila con Grafo de Conflicto, que es una generalización del Problema de la Mochila propuesta en [Pferschy y Schauer, 2009].

Dado un conjunto de enteros $N = \{1,\ldots,n\}$ con pesos $w_1,\ldots,w_n$, un entero $W$ y una relación de compatibilidad $\rho \subseteq N \times N$, el siguiente modelo

[3]Por lo tanto, es una relación reflexiva y simétrica.

[4]Aunque en trabajos independientes que no están basados en el Problema de la Suma de Subconjuntos Compatibles.

PLE es una extensión de la ecuación 4.2 que permite modelar el Problema de la Suma de Subconjuntos Compatibles:

$$\begin{aligned} \text{maximizar } & \sum_{j=1}^{n} w_j x_j \\ \text{sujeto a } & \sum_{j=1}^{n} w_j x_j \leq W, \\ & x_i + x_j \leq 1 \qquad \forall \langle i,j \rangle \notin \rho,\ i = 1,\ldots,n,\ j = 1,\ldots,n, \\ & x_i \in \mathbb{B} \qquad i = 1,\ldots,n, \end{aligned} \tag{4.3}$$

donde la solución al problema es el subconjunto $S = \{i \in N : x_i = 1\}$. La ecuación 4.3 asegura que sólo los objetos compatibles pueden ser seleccionados para formar parte de la solución.

Existen diversas posibilidades para modelar la relación de compatibilidad $\rho$. Una de ellas es mediante la matriz $C = (c_{i,j}) \in \mathbb{B}^{n \times n}$, que será denominada *matriz de compatibilidad*[5], en la que cada elemento se define como

$$c_{i,j} = \begin{cases} 1 & \text{si } \langle i,j \rangle \in \rho \\ 0 & \text{si } \langle i,j \rangle \notin \rho \end{cases}.$$

Puesto que la relación de compatibilidad es reflexiva y simétrica, la matriz de compatibilidad es una matriz simétrica ($c_{i,j} = c_{j,i}$) con los elementos de la diagonal principal $c_{i,i} = 1$.

Dada una solución óptima del Problema de la Suma de Subconjuntos Compatibles, si eliminamos el objeto $k$ de la solución, el conjunto restante será una solución óptima del subproblema definido por la capacidad $W - w_k$ y el conjunto de objetos $N \setminus \{k\}$. Sin embargo, la afirmación recíproca no es cierta; es decir, una solución óptima de dicho subproblema no produce necesariamente una solución óptima del problema completo cuando se incluye el objeto $k$ (por ejemplo, si el objeto $k$ es incompatible con alguno de los objetos seleccionados para el subproblema). Por lo tanto, debido a la relación de compatibilidad, durante la recursión de Bellman, la elección del objeto $j$ para formar parte de la solución $z_j(d)$ restringe el conjunto de objetos que se pueden incluir en las soluciones $z_k(d)$ para $k > j$, ya que sólo podrán incluirse objetos compatibles con el objeto $j$. Como consecuencia, el conjunto de objetos que forma la solución óptima puede ser descartado prematuramente si dicho conjunto contiene algún objeto incompatible con

[5]Otra alternativa es utilizar un grafo; de hecho, la matriz de compatibilidad puede ser considerada la matriz de adyacencia de un grafo no dirigido.

$j$. Esto significa que la recursión de Bellman no garantiza la obtención de soluciones óptimas al problema.

Sin embargo, la técnica de programación dinámica es empleada en el algoritmo voraz propuesto (algoritmo 9) para seleccionar en cada paso al mejor candidato. Se trata de una modificación del algoritmo 8, que consiste en comprobar si el objeto $j$ es compatible con todos los objetos que forman la solución $z_{j-1}(d-w_j)$ antes de seleccionarlo como parte de la solución $z_j(d)$. Esta prueba se realiza mediante la matriz de compatibilidad, calculando la expresión $\sum_{k\in X_{j-1}(d-w_j)}(1-C_{jk})$ (línea 6). Si el valor de la expresión es cero, entonces el objeto es compatible y se añade a la solución; en caso contrario, el objeto es descartado (línea 7). Adicionalmente, en la línea 8 se ha sustituido el beneficio $p_j$ por el peso $w_j$ en el cálculo del valor de $z_j(d)$ (véase la línea 6 del algoritmo 8).

**Algoritmo 9** Algoritmo voraz para el Problema de la Suma de Subconjuntos Compatibles

**Entrada:**
$C = (c_{i,j}) \in \mathbb{B}^{n\times n}$ ▷ *Matriz de compatibilidad.*
Un entero $W$ ▷ *Capacidad máxima.*
Enteros $w_1, \ldots, w_n$ ▷ *$n$ pesos.*
**Salida:** $S \subseteq N$

1: **procedimiento** SSC($C$,$W$,$w_1, \ldots, w_n$)
2: $z_0(d) \leftarrow 0$ para $d = 1, \ldots, W$
3: **para** $j \leftarrow 1$ **hasta** $n$ **hacer**
4: $z_j(d) \leftarrow z_{j-1}(d)$ para $d = 1, \ldots, w_j - 1$
5: **para** $d \leftarrow w_j$ **hasta** $W$ **hacer**
6: compat $\leftarrow \sum_{k\in X_{j-1}(d-w_j)}(1 - c_{j,k})$
7: **si** $z_{j-1}(d - w_j) + w_j > z_{j-1}(d)$ y compat $= 0$ **entonces**
8: $z_j(d) \leftarrow z_{j-1}(d - w_j) + w_j$
9: $X_j(d) \leftarrow X_{j-1}(d - w_j) \cup \{j\}$
10: **si no**
11: $z_j(d) \leftarrow z_{j-1}(d)$
12: $X_j(d) \leftarrow X_{j-1}(d)$
13: **fin si**
14: **fin para**
15: **fin para**
16: **devolver** $X_j(d)$
17: **fin procedimiento**

### 4.1.3. Aplicación del Problema de la Suma de subconjuntos compatibles a la optimización de FSMIM

El Problema de la Suma de Subconjuntos Compatibles ha sido utilizado como parte del algoritmo de minimización de selección de entradas implementado en la herramienta FSMIM-Gen 1.2, empleada para el estudio experimental en [García-Vargas et al., 2005; García-Vargas, 2006; García-Vargas et al., 2007; García-Vargas y Senhadji-Navarro, 2015].

Dicho algoritmo fue diseñado para una estrategia de optimización de FSMIM de dos fases, que comienza con la fase de minimización de la selección de entradas. Por este motivo, sólo puede ser aplicado a las MSE que corresponden a una FSMIM con estados simples y sin constantes $\{0, 1\}$ seleccionadas (es decir, una FSMIM obtenida mediante el procedimiento de conversión trivial). En lo que resta de sección, supondremos que todas las MSE se ajustan a estas características.

Dada una MSE $A$, el algoritmo propuesto se fundamenta en dos ideas básicas: (1) intentar colocar todos los elementos de $A$ que correspondan a la misma entrada en una misma columna de $A$; y (2) recorrer las columnas de $A$ en orden, intentando rellenarlas con el mayor número de elementos que no se hayan colocado en una columna anterior (dando prioridad al punto (1)).

Sea la MSE $A = (a_{i,j}) \in MSE^{n \times k}$ y $X = \{x_1, \ldots, x_m\}$ el conjunto de entradas de $A$ ($a_{ij} \in X \cup \{-\}$). El procesamiento de cada columna se realiza aplicando el Problema de la Suma de Subconjuntos Compatibles, estableciendo las siguientes equivalencias:

- Cada elemento de $x_i \in X$ corresponde a un objeto $i \in N$.
- El peso del objeto $i$ es igual al número de filas de $A$ en las que aparece la entrada $x_i$. Puesto que cada entrada sólo puede aparecer una vez en cada fila como máximo, el peso es igual al número de veces que aparece cada entrada en $A$.
- La capacidad máxima es igual al número de filas de $A$ ($W = n$).
- Se dirá que dos entradas son *compatibles* si ninguna fila de $A$ contiene a ambas entradas o si son la misma entrada. La matriz de compatibilidad $C = (c_{i,j}) \in \mathbb{B}^{m \times m}$ se define como

$$c_{i,j} = \begin{cases} 1 & \text{si } x_i \text{ y } x_j \text{ son compatibles} \\ 0 & \text{en otro caso} \end{cases}.$$

Esta definición de compatibilidad establece una relación reflexiva y simétrica; sin embargo, tal como muestra el siguiente ejemplo, no es una relación transitiva.

**Ejemplo 4.1.** *En la MSE $A \in MSE^{2\times 2}$ que se muestra a continuación, la entrada $x_1$ es compatible con $x_2$ y $x_3$; sin embargo, $x_2$ no es compatible con $x_3$:*

$$A = \begin{bmatrix} x_1 & - \\ x_2 & x_3 \end{bmatrix}.$$

- Cada columna de $A$ corresponde a una instancia del Problema de la Suma de Subconjuntos Compatibles. Por lo tanto, se tienen $k$ instancias que deben ser resueltas secuencialmente.

El algoritmo 10 muestra el procedimiento de asignación de entradas basado en las equivalencias anteriores. Partiendo del conjunto de entradas $X$, el algoritmo aplica el Problema de la Suma de Subconjuntos Compatibles $k$ veces (una por cada columna de $A$). En cada iteración, las entradas seleccionadas son eliminadas de $X$ para que no sean procesadas en la siguiente iteración (línea 10). Las líneas 6 y 7, simplemente establecen los valores de los pesos y la submatriz de compatibilidad asociadas al subconjunto de $X$, que forman parte de la instancia del Problema de la Suma de Subconjuntos Compatibles (donde, $C^J$ representa a la submatriz de $C$ formada por las filas de índice $i \in J$ y las columnas de índice $i \in J$).

Sea $S = \{S_1, \ldots, S_k\}$ la solución del algoritmo 10. Los elementos de $S_j$ determinan qué entradas se asignarán a la columna $j$ de la MSE. Obviamente, la asignación de una entrada $x \in S_j$ a la columna $j$ se realiza sólo a los elementos $a_{i,j}$ de las filas que contienen a $x$.

**Ejemplo 4.2.** *Considérese la siguiente MSE $A \in MSE^{6\times 3}$, que determina la matriz de compatibilidad $C \in \mathbb{B}^{5\times 5}$:*

$$A = \begin{bmatrix} x_1 & - & - \\ x_2 & x_3 & x_4 \\ x_3 & x_1 & - \\ x_1 & x_2 & x_3 \\ x_1 & x_3 & x_5 \\ x_2 & x_4 & x_5 \end{bmatrix}, \qquad C = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{matrix}.$$

*A partir de $A$, se obtiene el conjunto de entradas inicial $X = \{x_1, \ldots, x_5\}$, el valor de los pesos $(w_1, w_2, w_3, w_4, w_5) = (4, 3, 4, 2, 2)$ y la capacidad máxima $W = 6$.*

**Algoritmo 10** Algoritmo voraz de asignación de entradas

**Entrada:**
$A = (a_{i,j}) \in MSE^{W \times k}$ ▷ *$W$ es el número de filas de $A$.*
**Salida:** $S = \{S_1, \ldots, S_k\}$ ▷ *Conjuntos disjuntos de entradas seleccionadas.*

1: $S_i \leftarrow \emptyset$ para $i = 1, \ldots, k$
2: $X \leftarrow \{a_{i,j} : a_{i,j}$ no es una ESI, $1 \leq i \leq W,\ 1 \leq j \leq k\}$
3: $w_i \leftarrow$ número de veces que $x_i$ se encuentra en $A$ para $i = 1, \ldots, |X|$
4: $C \leftarrow$ matriz de compatibilidad de $A$
5: **para** $j \leftarrow 1$ **hasta** $k$ **hacer**
6: Sean $w'_1, \ldots, w'_{|X|}$ los elementos de $\{w_i : x_i \in X\}$
7: Sea $C'$ la submatriz $C^{\{i:x_i \in X\}}$
8: $S' \leftarrow \mathrm{SSC}(C', W, w'_1, \ldots, w'_{|X|})$ ▷ *algoritmo 9*
9: $S_j \leftarrow \{x_i : i \in S'\}$
10: $X \leftarrow X \setminus S_j$
11: **fin para**
12: **devolver** $S$

*La solución del* Problema de la Suma de Subconjuntos Compatibles *obtenida en la primera iteración es $S_1 = \{x_1, x_4\}$, con un peso total igual a 6. Por lo tanto, el conjunto de entradas que serán procesadas en la siguiente iteración es $X = \{x_2, x_3, x_5\}$, en el que no hay ningún par de entradas compatibles. La solución final, tras completar todas las iteraciones es $S_1 = \{x_1, x_4\}$, $S_2 = \{x_3\}$ y $S_3 = \{x_2\}$, que corresponde a la MSE*

$$A = \begin{bmatrix} x_1 & - & - \\ x_4 & x_3 & x_2 \\ x_1 & x_3 & - \\ x_1 & x_3 & x_2 \\ x_1 & x_3 & - \\ x_4 & - & x_2 \end{bmatrix}.$$

*Sin embargo, como puede observarse, la MSE obtenida no contiene la entrada $x_5$; por lo tanto, la solución no es correcta. Puesto que la entrada $x_5$ no es compatible con las entradas $x_2$ y $x_3$, no es posible encontrar una solución al* Problema de la Suma de Subconjuntos Compatibles *que la incluya en $S_2$ o $S_3$.*

La solución propuesta para el problema mostrado en el ejemplo anterior (e implementada en la herramienta FSMIM-Gen 1.2), consiste en seleccionar una entrada $x_j$ y dividirla en dos entradas diferentes: una entrada $x'_j$, que

representa a la entrada $x_j$ de una de las filas de la MSE, y una entrada $x''_j$, que representa a la entrada $x_j$ del resto de filas de la MSE. Por lo tanto, el peso de las entradas $x'_j$ y $x''_j$ es $w'_j = 1$ y $w''_j = w_j - 1$, respectivamente. Tras realizar la división de entradas, se vuelve a aplicar el el algoritmo 10 con el nuevo conjunto de entradas $X = \{x_1, \ldots, x'_j, x''_j, \ldots, x_m\}$. Este procedimiento se repite hasta que la solución obtenida por el algoritmo 10 contiene a todas las entradas del conjunto.

Se denominará *número de compatibilidades entre entradas* al número de pares de entradas que son compatibles. El criterio para determinar la entrada y la fila adecuadas se basa en la idea de que la probabilidad de que el algoritmo 10 encuentre una solución aumenta con el número de compatibilidades entre entradas. La estrategia implementada busca la división de entradas que maximiza el número de unos en la matriz de compatibilidad, lo que equivale a maximizar el número de compatibilidades entre entradas.

**Ejemplo 4.2** (continuación)**.** *A continuación se muestran dos posibles divisiones de entradas. La primera de ellas divide la entrada $x_1$ en la fila 5 de A, dando como resultado el conjunto de entradas $X_1 = \{x'_1, x''_1, x_2, x_3, x_4, x_5\}$ con pesos $(w'_1, w''_1, w_2, w_3, w_4, w_5) = (3, 1, 3, 4, 2, 2)$, la MSE $A_1$ y la matriz de compatibilidad $C_1$:*

$$A_1 = \begin{bmatrix} x'_1 & - & - \\ x_2 & x_3 & x_4 \\ x_3 & x'_1 & - \\ x'_1 & x_2 & x_3 \\ x''_1 & x_3 & x_5 \\ x_2 & x_4 & x_5 \end{bmatrix}, \quad C_1 = \begin{array}{c} \begin{array}{cccccc} x'_1 & x''_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} 1 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 1 & 1 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{array}{c} x'_1 \\ x''_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array}.$$

*El número de compatibilidades de entradas obtenido es 5 (véanse los elementos en negrita de la matriz $C_1$ con valor 1).*

*La segunda división realizada divide la entrada $x_5$ en la fila seis de A, dando como resultado el conjunto de entradas $X_2 = \{x_1, x_2, x_3, x_4, x'_5, x''_5\}$ con pesos $(w_1, w_2, w_3, w_4, w'_5, w''_5) = (4, 3, 4, 2, 1, 1)$, la MSE $A_2$ y la matriz*

*de compatibilidad $C_2$:*

$$A_2 = \begin{bmatrix} x_1 & - & - \\ x_2 & x_3 & x_4 \\ x_3 & x_1 & - \\ x_1 & x_2 & x_3 \\ x_1 & x_3 & x'_5 \\ x_2 & x_4 & x''_5 \end{bmatrix}, \quad C_2 = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x'_5 & x''_5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} \\ 0 & 1 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 1 & 0 & 0 & \mathbf{1} \\ 1 & 0 & 0 & 1 & \mathbf{1} & 0 \\ 0 & 1 & 0 & 1 & 1 & \mathbf{1} \\ 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x'_5 \\ x''_5 \end{matrix}.$$

*En este caso, el número de compatibilidades de entradas aumenta hasta seis. El resto de divisiones posibles da como resultado un número de compatibilidades de entradas menor; por lo tanto, esta es la división seleccionada.*

*Con esta división de entradas, el algoritmo 10 obtiene la solución $S_1 = \{x_1, x_4\}$, $S_2 = \{x_3, x''_5\}$ y $S_3 = \{x_2, x'_5\}$, que corresponde a la MSE*

$$\begin{bmatrix} x_1 & - & - \\ x_4 & x_3 & x_2 \\ x_1 & x_3 & - \\ x_1 & x_3 & x_2 \\ x_1 & x_3 & x_5 \\ x_4 & x_5 & x_2 \end{bmatrix}.$$

*Puesto que todas las entradas de $X$ han sido asignadas a la MSE, el procedimiento de minimización de la selección de entradas ha sido completado.*

La necesidad de realizar división de entradas es un indicio de que la solución óptima del problema de la minimización de la selección de entradas no se obtiene necesariamente minimizando el coste de selección de entradas de cada columna de la MSE de forma independiente. En la siguiente sección se propone un nuevo problema de optimización que modela la minimización de la selección de entradas.

## 4.2. *k*-Emparejamiento Parcial Maximal Mínimo

El Problema del $k$-Emparejamiento Parcial Maximal Mínimo surge como una forma de modelar el problema de la minimización del coste de la selección de entradas [García-Vargas y Senhadji-Navarro, 2012].

A continuación se muestra una descripción informal del problema. Dado un grafo bipartito, se forman $k$ conjuntos mediante el siguiente procedimien-

to: para cada *vértice izquierdo*[6], se selecciona el máximo número posible de *vértices derechos* adyacentes que sea menor o igual que $k$ y se añade cada uno de los vértices a un conjunto diferente. El objetivo es minimizar la suma de la cardinalidad de los $k$ conjuntos.

En el contexto de la minimización de la selección de entradas, los vértices izquierdos representan las filas de la MSE; los vértices adyacentes, las entradas de cada fila; y cada uno de los $k$ conjuntos, el conjunto de entradas seleccionadas para cada una de las $k$ columnas de la MSE. En estos términos, el objetivo minimiza el coste de selección de entradas de la MSE.

Sea $G = (U \cup V, E)$ un grafo bipartito, donde $E \subseteq U \times V$ es el conjunto de aristas del grafo. Dado un vértice $s \in U \cup V$, el conjunto de aristas de $E$ incidentes en $s$ es denotado por $E(s)$, y el grado de $s$, por $\text{grad}_G(s)$. Se define el conjunto de aristas de $E$ incidentes en cualquier vértice del conjunto $S \subseteq U \cup V$ como $E(S) = \bigcup_{s \in S} E(s)$. Se define el *grado mínimo y máximo de los vértices de* $U$ como $\delta_U(G) = \text{mín}\{\text{grad}_G(u) : u \in U\}$ y $\Delta_U(G) = \text{máx}\{\text{grad}_G(u) : u \in U\}$, respectivamente. Dado un subconjunto de aristas $E' \subseteq E$, $V(E')$ y $U(E')$ denotan el conjunto de vértices de $V$ y $U$, respectivamente, en los que incide alguna arista de $E'$.

Un *emparejamiento* en $G$ es un conjunto de aristas $M \subseteq E$ tal que ningún par de aristas es incidente en un mismo vértice de $U \cup V$. En la literatura se han propuesto diferentes relajaciones del concepto de emparejamiento en el contexto de los grafos bipartitos [Harvey et al., 2006; García-Vargas y Senhadji-Navarro, 2012]. La definición del Problema del $k$-Emparejamiento Parcial Maximal Mínimo se basa en esta idea.

**Definición 4.3.** *Se define un* **emparejamiento parcial** *en un grafo bipartito* $G = (U \cup V, E)$ *como un conjunto de aristas* $E' \subseteq E$ *tal que ningún par de aristas es incidente en un mismo vértice de* $U$*. Se dice que un emparejamiento parcial es* **perfecto** *si* $U(E') = U$*.*

Nótese que la restricción se impone exclusivamente sobre los vértices de $U$; es decir, que en un emparejamiento parcial varias aristas de $E'$ pueden ser incidentes en un mismo vértice de $V$. Un emparejamiento parcial perfecto contiene el mayor número de aristas posibles.

**Definición 4.4.** *Se define un* ***k*-emparejamiento parcial** *en un grafo bipartito* $G = (U \cup V, E)$ *como una colección* $P = \{P_1, \ldots, P_k\}$ *de* $k$ *emparejamientos parciales disjuntos; es decir, tales que* $P_i \cap P_j = \emptyset$ *para todo* $i \neq j$*.*

[6]Es habitual que en un grafo bipartito, representado como dos conjuntos disjuntos de vértices situados uno a la izquierda del otro, se denominen informalmente los vértices de cada conjunto como vértices izquierdos y vértices derechos.

*Se define un* ***k*-emparejamiento parcial maximal** *en $G$ como un k--emparejamiento parcial que contiene el máximo número de aristas; es decir, un k-emparejamiento parcial $P = \{P_1, \ldots, P_k\}$ tal que $|\bigcup_{i=1}^{k} P_i|$ es máximo.*

**Lema 4.5.** *Sea $P$ un k-emparejamiento parcial maximal en $G = (U \cup V, E)$. Para todo $u \in U$ se cumple que $|E(u) \cap \bigcup_{i=1}^{k} P_i| = \text{mín}\{k, grad_G(u)\}$.*

***Demostración.*** Obviamente, por la definición de $k$-emparejamiento parcial, $|E(u) \cap \bigcup_{i=1}^{k} P_i| \leq \text{mín}\{k, \text{grad}_G(u)\}$. Razonando por reducción al absurdo, supóngase que existe un $u \in U$ tal que $|E(u) \cap \bigcup_{i=1}^{k} P_i| = n < \text{mín}\{k, \text{grad}_G(u)\}$, entonces existen $k - n$ emparejamientos parciales $\{P^{(1)}, \ldots, P^{(k-n)}\} \subseteq P$ tales que $|E(u) \cap P^{(j)}| = 0$ $(1 \leq j \leq k-n)$ y existen $\text{grad}_G(u) - n$ aristas incidentes en $u$ que pueden ser asignadas a cualquier $P^{(j)}$. Esto implica que $P$ no es maximal, lo que supone una contradicción; por lo tanto, queda demostrado el lema. □

**Corolario 4.6.** *Sea $P$ un k-emparejamiento parcial maximal en $G = (U \cup V, E)$. Entonces, se cumple que todos los emparejamientos parciales $P_i \in P$ son perfectos sii $k \leq \delta_U(G)$.*

***Demostración.*** Si $k \leq \delta_U(G)$ entonces para todo $u \in U$ se cumple que $|E(u) \cap \bigcup_{i=1}^{k} P_i| = k$. Esto implica que $|E(u) \cap P_i| = 1$ para todo $P_i \in P$; por lo tanto, $U(P_i) = U$ ($P_i$ es un emparejamiento parcial perfecto).

Por otra parte, si todo $P_i \in P$ es un emparejamiento parcial perfecto, entonces $U(P_i) = U$ para todo $i = 1, \ldots, k$. Puesto que los emparejamientos parciales $P_i$ son disjuntos, cada vértice de $U$ tiene al menos una arista diferente en cada $P_i$, es decir, tiene $k$ aristas como mínimo. Entonces, $\delta_U(G) \geq k$. □

El Problema del $k$-Emparejamiento Parcial Maximal Mínimo puede ser descrito como el siguiente problema de optimización:

| Problema del $k$-Emparejamiento Parcial Maximal Mínimo | |
|---|---|
| *Instancia:* | Un grafo bipartito $G = (U \cup V, E)$ y un entero positivo $k \leq \Delta_U(G)$. |
| *Tarea:* | Buscar un $k$-emparejamiento parcial maximal $P = \{P_1, \ldots, P_k\}$ en $G$ tal que $\sum_{i=1}^{k} \lvert V(P_i)\rvert$ sea mínimo. |

**Proposición 4.7.** *El* Problema del $k$-Emparejamiento Parcial Maximal Mínimo *es NP-completo.*

***Demostración***. Se demostrará que el problema de decisión, relacionado con el problema de optimización, definido a continuación es *NP*-completo:

| $k$-Emparejamiento Parcial Maximal Mínimo | |
|---|---|
| *Instancia:* | Un grafo bipartito $G = (U \cup V, E)$, un entero positivo $k \leq \Delta_U(G)$ y un entero positivo $B$. |
| *Cuestión:* | ¿Existe un $k$-emparejamiento parcial maximal $P = \{P_1, \ldots, P_k\}$ en $G$ tal que $\sum_{i=1}^{k} \lvert V(P_i)\rvert \leq B$?. |

Es fácil ver que este problema de decisión pertenece a *NP*, ya que se puede comprobar en tiempo polinomial si $P$ es un $k$-emparejamiento parcial maximal tal que $\sum_{i=1}^{k} |V(P_i)| \leq B$. Un algoritmo sólo necesita verificar el Lema 4.5 para todos los vértices $u \in U$ y comprobar que cada $P_i$ contiene como máximo una arista diferente de $E(u)$.

La demostración de que el $k$-Emparejamiento Parcial Maximal Mínimo es *NP*-completo se realizará por restricción al Hitting Set, que se define como el siguiente problema de decisión:

| Hitting Set | |
|---|---|
| *Instancia:* | Una colección $C = \{S_1, \ldots, S_n\}$ de subconjuntos no vacíos de un conjunto finito $S$ y un entero positivo $B \leq \lvert S\rvert$. |
| *Cuestión:* | ¿Existe un subconjunto $H \subseteq S$ con $\lvert H\rvert \leq B$ tal que $H$ contiene al menos un elemento de cada subconjunto de $C$? |

Se denomina *hitting set* a cada solución factible; es decir, a cada subconjunto $H \subseteq S$ que contiene al menos un elemento de cada subconjunto de $C$. En [Karp, 1972] se demostró que el Hitting Set es *NP*-completo. Por otra parte, el Hitting Set es equivalente al Set Cover [Ausiello et al., 1980].

Sea $I = ((S, C), B)$ una instancia del Hitting Set, donde $S = \{s_1, \ldots, s_m\}$ y $C = \{S_1, \ldots, S_n\}$. La instancia $I$ puede transformarse en una instancia del $k$-Emparejamiento Parcial Maximal Mínimo $I' = ((G, k), B)$ como sigue. Se construye un grafo bipartito $G = (U \cup V, E)$, donde $U = \{u_1, \ldots, u_n\}$ y $V = \{v_1, \ldots, v_n\}$, creando una arista $(u_i, v_j) \in E$ para cada $s_j \in S_i$. Es fácil comprobar que $G$ puede ser construido en tiempo polinomial.

La instancia $I'$ será restringida a $k = 1$ en lo que resta de demostración. Todos los subconjuntos $S_i \in C$ son conjuntos no vacíos; por lo tanto,

$\delta_U(G) \geq 1$. Esto implica que cualquier solución factible de $I'$ es un 1-emparejamiento parcial maximal $P = \{P_1\}$ donde $P_1$ es un emparejamiento parcial perfecto.

Se puede afirmar que $(S, C)$ tiene un *hitting set* $H \subseteq S$ tal que $|H| \leq B$ sii $G$ tiene un 1-emparejamiento parcial maximal $P = \{P_1\}$ tal que $|V(P_1)| \leq B$. En primer lugar, supóngase que $H \subseteq S$ es un *hitting set* con $|H| \leq B$. Sea $V' = \{v_i \in V : s_i \in H\}$; puesto que $H$ contiene al menos un elemento de cada $S_i \in C$, cada vértice de $U$ es adyacente a un vértice de $V'$ como mínimo. Por lo tanto, seleccionando para cada vértice en $U$ una única arista incidente en cualquier vértice en $V'$, se puede crear de manera trivial un 1--emparejamiento parcial maximal $P = \{P_1\}$. Se puede comprobar fácilmente que $|V(P_1)| \leq |V'| = |H| \leq B$.

En segundo lugar, supóngase que $P = \{P_1\}$ es un 1-emparejamiento parcial maximal en $G$ tal que $|V(P_1)| \leq B$. Sea $H = \{s_i \in S : v_i \in V(P_1)\}$; puesto que $P_1$ es un emparejamiento parcial perfecto en $G$, para todo $u_i \in U$ existe una arista $(u_i, v_j) \in E$ tal que $v_j \in V(P_1)$. Esto implica que para todo $S_i \in C$ existe un $s_j \in H$ tal que $s_j \in S_i$. Por lo tanto, $H$ es un *hitting set* tal que $|H| = |V(P_1)| \leq B$. □

**Corolario 4.8.** *El* Problema del $k$-Emparejamiento Parcial Maximal Mínimo *es una generalización del* Hitting Set *y del* Set Cover.

### 4.2.1. Modelo de programación lineal entera 0/1

Sea $P = \{P_1, \ldots, P_k\}$ un $k$-emparejamiento parcial en un grafo bipartito $G = (U \cup V, E)$. Se definen las variables binarias $x_{e,i}, y_{v,i} \in \mathbb{B}$ como

$$x_{e,i} = \begin{cases} 1 \text{ si } e \in P_i \\ 0 \text{ en otro caso} \end{cases} \quad \forall e \in E,\ i = 1, \ldots, k,$$

$$y_{v,i} = \begin{cases} 1 \text{ si } v \in V(P_i) \\ 0 \text{ en otro caso} \end{cases} \quad \forall v \in V,\ i = 1, \ldots, k.$$

El Problema del $k$-Emparejamiento Parcial Maximal Mínimo se puede modelar como el siguiente problema de PLE:

$$\text{minimizar} \sum_{i=1}^{k} \sum_{v \in V} y_{v,i} \tag{4.4}$$

$$\text{sujeto a} \sum_{e \in E(u)} x_{e,i} \leq 1, \qquad \forall u \in U,\ i = 1, \ldots, k, \tag{4.5}$$

$$\sum_{i=1}^{k} x_{e,i} \leq 1, \qquad \forall e \in E, \tag{4.6}$$

$$\sum_{i=1}^{k} \sum_{e \in E(u)} x_{e,i} = \text{mín}\{k, \text{grad}_G(u)\}, \quad \forall u \in U, \tag{4.7}$$

$$x_{e,i} \leq y_{v,i}, \qquad \forall e \equiv (u,v) \in E, \; i = 1, \ldots, k, \tag{4.8}$$

$$y_{v,i} \leq \sum_{e \in E(v)} x_{e,i}, \qquad \forall v \in V, \; i = 1, \ldots, k, \tag{4.9}$$

$$x_{e,i} \in \mathbb{B}, \qquad \forall e \in E, \; i = 1, \ldots, k,$$

$$y_{v,i} \in \mathbb{B}, \qquad \forall v \in V, \; i = 1, \ldots, k.$$

Sea $P = \{P_1, \ldots, P_k\}$ la solución obtenida a para el problema PLE, donde $P_i = \{e \in E : x_{e,i} = 1\}$. En la ecuación 4.4, la cardinalidad de $V(P_i)$ se calcula como $\sum_{v \in V} y_{v,i}$. La ecuación 4.5 garantiza que cada $P_i$ contiene como mínimo una arista incidente en cada vértice $u \in U$; es decir, que $P_i$ es un emparejamiento parcial. La ecuación 4.6 asegura que una misma arista no puede pertenecer a diferentes $P_i$; es decir, que los $P_i$ son disjuntos y, por tanto, $P$ es un $k$-emparejamiento parcial. Además, $P$ es maximal, ya que el Lema 4.5 se verifica para todos los vértices $u \in U$ debido a la ecuación 4.7. Por último, las ecuaciones 4.8 y 4.9 garantizan que los valores de las variables $x_{e,i}$ y $y_{v,i}$ son coherentes. Por una parte, la ecuación 4.8 asegura que una arista $e \equiv (u,v) \in E$ puede pertenecer al emparejamiento parcial $P_i$ sólo si el vértice $v$ pertenece a $V(P_i)$; es decir, $x_{e,i}$ puede ser igual a 1 sólo si $y_{v,i} = 1$. Por otra parte, la ecuación 4.9 garantiza que un vértice $v$ puede pertenecer a $V(P_i)$ sólo si existe una arista $e \equiv (u,v)$ perteneciente a $P_i$.

### 4.2.2. Algoritmo voraz

Una estrategia intuitiva para obtener soluciones a partir de una instancia del Problema del $k$-Emparejamiento Parcial Maximal Mínimo es construir el $k$-emparejamiento parcial seleccionando en cada paso el mayor conjunto de aristas incidentes en un mismo vértice $v \in V$ que pueda ser añadido a algún emparejamiento parcial y añadiéndolo a dicho emparejamiento parcial. El procedimiento termina cuando el $k$-emparejamiento parcial obtenido es maximal.

El algoritmo 11 muestra el algoritmo voraz que implementa la estrategia propuesta. Puesto que el algoritmo modifica el grafo bipartito $G = (U \cup$

$V, E$), los conjuntos $U$, $V$ y $E$ de cada nuevo grafo $G$ obtenido en cada paso son denotados por $U_G$, $V_G$ y $E_G$ respectivamente.

**Definición 4.9.** *Dado un grafo bipartito $G = (U \cup V, E)$, un emparejamiento parcial $E' \subseteq E$ y un vértice $v \in V$, se define el conjunto de aristas $\alpha_G(v, E')$ como $\alpha_G(v, E') = \{(u, v) \in E(u) : u \notin U(E')\}$; es decir, como el mayor conjunto de aristas incidentes en $v$ tal que $E' \cup \alpha_G(v, E')$ continua siendo un emparejamiento parcial.*

El algoritmo comienza con $k$ emparejamientos parciales vacíos $P = \{P_1, \ldots, P_k\}$. En cada iteración, busca el conjunto de aristas $\alpha_G(v, P_i)$ con mayor cardinalidad, añade las aristas del conjunto a $P_i$ y las elimina de $G$. El algoritmo termina cuando $|E_G|$ es igual que $\kappa$, que es el número de aristas que quedarán en $G$ cuando el $k$-emparejamiento parcial $P$ sea maximal.

## 4.3. Aplicación del Problema del *k*-Emparejamiento Parcial Maximal Mínimo a la optimización de FSMIM

Tal como se mostrará en esta sección, el subproblema de la minimización de la selección de entradas puede ser modelado con el Problema del $k$--Emparejamiento Parcial Maximal Mínimo.

Sea $A = (a_{ij}) \in MSE^{n \times k}$ una MSE de una FSMIM sin constantes $\{0, 1\}$ seleccionadas. En adelante, se supondrá además que la MSE no contiene entradas efectivas repetidas en una misma fila (esta restricción se cumple para cualquier FSMIM generada a partir de una FSM mediante el procedimiento de conversión trivial y para las obtenidas aplicando transformaciones

**Algoritmo 11** Algoritmo voraz para el Problema del $k$-Emparejamiento Parcial Maximal Mínimo

**Entrada:** Grafo bipartito $G = (U \cup V, E)$, entero positivo $k$
**Salida:** $k$-emparejamiento parcial maximal $P = \{P_1, \ldots, P_k\}$

1: $P_i \leftarrow \emptyset$ para todo $P_i \in P$
2: $\kappa \leftarrow |E_G| - \sum_{u \in U_G} \min\{k, \text{grad}_G(u)\}$
3: **mientras** $|E_G| > \kappa$ **hacer**
4: $\quad (v, P_i) \leftarrow \arg\max_{(v, P_i) \in V \times P} |\alpha_G(v, P_i)|$
5: $\quad P_i \leftarrow P_i \cup \alpha(v, P_i)$
6: $\quad G \leftarrow G - \alpha_G(v, P_i)$
7: **fin mientras**

elementales a dicha FSMIM). A partir de la MSE $A$ se puede construir un grafo bipartito $G = (U \cup V, E)$ donde

$$\begin{aligned} U &= \{1, \ldots, n\}, \\ V &= \{a_{ij} : 1 \leq i \leq n,\ 1 \leq j \leq k,\ a_{ij} \text{ no es una ESI}\}, \\ E &= \{(i, a_{ij}) : 1 \leq i \leq n,\ 1 \leq j \leq k,\ a_{ij} \in V\}. \end{aligned}$$

Cada elemento de $U$ es un índice de fila de $A$ (que, a su vez, representa a un estado de la FSMIM), $V$ contiene las entradas efectivas de la FSMIM, y las aristas de $E$ relacionan cada fila de $A$ con las entradas seleccionadas contenidas en dicha fila (es decir, relacionan a cada estado de la FSMIM con sus entradas efectivas). El grafo $G$ obtenido será denominado *grafo equivalente a la MSE $A$*. Para cada elemento de $A$ que no es una ESI existe una arista diferente en $E$ y viceversa (es decir, $|E|$ es igual al número de elementos de $A$ que no son una ESI).

**Definición 4.10.** *Sea $G = (U \cup V, E)$ el grafo equivalente a una MSE $A \in MSE^{n \times k}$. Sea $P = \{P_1, \ldots, P_k\}$ un $k$-emparejamiento parcial maximal en $G$. Se denominará* **MSE definida por** *$P$ a la MSE $A' = (a'_{i,j}) \in MSE^{n \times k}$ definida como*

$$a'_{i,j} = \begin{cases} v & \text{si } (i, v) \in P_j \\ ESI & \text{en otro caso} \end{cases}.$$

**Proposición 4.11.** *Sea $G = (U \cup V, E)$ el grafo equivalente a una MSE $A \in MSE^{n \times k}$ de una FSMIM sin estados compuestos[7] y sin constantes $\{0, 1\}$ seleccionadas. Si $P = \{P_1, \ldots, P_k\}$ es una solución al* Problema del $k$-Emparejamiento Parcial Maximal Mínimo *para la instancia $(G, k)$, entonces la MSE definida por $P$ corresponde a una permutación de entradas de $A$ que minimiza el coste de selección de entradas.*

***Demostración.*** Puesto que las permutaciones que intercambian elementos indeterminados entre sí no modifican la MSE, estas no serán tenidas en cuenta. Se denominará una *configuración de entradas* de $A$ al conjunto de permutaciones de entradas de $A$ que dan lugar a una misma MSE.

Resulta fácil comprobar que cada $k$-emparejamiento parcial maximal en $G$, $P = \{P_1, \ldots, P_k\}$, corresponde a una configuración de entradas de $A$ diferente. Sea $A'$ la MSE definida por $P$. Puesto que $k$ es el número de columnas de $A$, $k \geq \text{grad}_G(u)$ para todo $u \in U$. Entonces, por el Lema 4.5

[7] Esta restricción no es realmente necesaria, pero se impone para garantizar que cualquier permutación de entradas en $A$ da como resultado una MSE equivalente.

se tiene que $|E(u) \cap \bigcup_{i=1}^{k} P_i| = \text{grad}_G(u)$ para todo $u \in U$. Esto implica que se usan todas las aristas de $E$ y, por lo tanto, en $A'$ se encuentran todos los elementos de $A$ no indeterminados. Obviamente, para cada vértice $u \in U$, la asignación de las aristas incidentes en $u$ a los emparejamientos parciales $P_j \in P$ determinan la posición que tendrá cada entrada efectiva de la fila $u$ de $A$ dentro de la fila $u$ de $A'$.

Igualmente, se puede comprobar fácilmente que cada configuración de entradas de $A$ corresponde a un $k$-emparejamiento parcial maximal en $G$ diferente. Cada columna de $A$ define un emparejamiento parcial en $G$, ya que cada elemento de la columna corresponde a una arista de $E$ incidente en un elemento diferente de $U$ (es decir, una fila diferente de $A$). La matriz $A$ completa define un $k$-emparejamiento parcial (nótese que $A$ tiene $k$ columnas), ya que, puesto que cada elemento de $A$ corresponde a una arista diferente, los $k$ emparejamientos parciales son necesariamente disjuntos. Además, el $k$-emparejamiento parcial es maximal porque se usan todas las aristas de $E$.

Por lo tanto, por una parte, existe una correspondencia biunívoca entre cada configuración de entradas de $A$ y cada $k$-emparejamiento parcial maximal en $G$. Por otra parte, el número de entradas efectivas diferentes de la columna $j$ de $A'$, excluyendo las ESI, es $|V(P_j)|$; es decir, $CSE_j(A') = |V(P_j)|$ (véase la Definición 2.20). Esto implica que el coste de selección de entradas $CSE(A') = \sum_{i=1}^{k} |V(P_j)|$ es mínimo, lo que demuestra la proposición. □

## 4.4. Variantes multiobjetivo del Problema del *k*-Emparejamiento Parcial Maximal Mínimo

Aunque el Problema del $k$-Emparejamiento Parcial Maximal Mínimo permite obtener soluciones óptimas para el subproblema de la minimización de la selección de entradas, en esta sección se presentan dos variantes propuestas para mejorar los resultados del problema general de optimización de FSMIM.

Tal como se mostró en la sección 2.4.2.1, en una FSMIM-T el tamaño de la ROM, además de depender del número de estados, también depende del coste de selección de entradas de cada canal de selección por la ecuación 2.29. La minimización del coste de selección de entradas de la MSE permite reducir el valor de la ecuación 2.29; sin embargo, no garantiza que se alcance el valor mínimo. La variante multiobjetivo que se presenta en el apartado siguiente permite obtener el valor mínimo de dicha ecuación.

Sea $P = \{P_1, \ldots, P_k\}$ un $k$-emparejamiento parcial en un grafo bipartito $G = (U \cup V, E)$. Utilizando la terminología propia del problema de opti-

mización de FSMIM, se denominará *Coste de Selección de Entradas* (CSE) de $P$ a la expresión $\sum_{i=1}^{k} |V(P_i)|$, y *número de bits de selección* de $P$, a la expresión $\sum_{i=1}^{k} \lceil \log_2 |V(P_i)| \rceil$ (equivalente a la ecuación 2.29)

### 4.4.1. *k*-Emparejamiento Parcial Maximal con número de bits de selección y CSE mínimos

En esta sección se presenta una variante del Problema del $k$-Emparejamiento Parcial Maximal Mínimo que busca la solución con número de Bits de selección y Coste de selección de entradas mínimos, por lo que será denominado Problema del $k$-Emparejamiento Parcial Maximal Mínimo BC. Dicho problema puede ser descrito como el siguiente problema de optimización multiobjetivo:

| Problema del $k$-Emparejamiento Parcial Maximal Mínimo BC | |
|---|---|
| *Instancia:* | Un grafo bipartito $G = (U \cup V, E)$ y un entero positivo $k \leq \Delta_U(G)$. |
| *Tarea:* | Buscar un $k$-emparejamiento parcial maximal $P = \{P_1, \ldots, P_k\}$ en $G$ tal que la tupla $\left(\sum_{i=1}^{k} \lceil \log_2 \lvert V(P_i) \rvert \rceil,\ \sum_{i=1}^{k} \lvert V(P_i) \rvert\right)$ sea mínima en orden lexicográfico. |

El objetivo del problema es encontrar la solución con menor CSE de entre todas las que presentan el mínimo número de bits de selección. Por lo tanto, cuando el problema se aplica a la minimización de la selección de entradas de una FSMIM-T, la solución obtenida minimiza el número de bits de selección de la FSMIM-T, aumentando la influencia de la fase de minimización en la reducción del tamaño de la ROM. Sin embargo, como contrapartida, a pesar de que la minimización del número de bits de selección favorece la reducción del CSE, no se alcanza necesariamente el CSE mínimo, tal como muestra el siguiente ejemplo.

**Ejemplo 4.12.** *A continuación se muestran dos permutaciones de entradas*

*diferentes de una MSE* $A \in MSE^{7\times 3}$*:*

$$A_1 = \begin{bmatrix} x_1 & x_4 & x_7 \\ x_1 & x_2 & x_5 \\ x_1 & x_3 & x_6 \\ x_8 & - & x_7 \\ x_8 & - & x_5 \\ x_8 & x_4 & - \\ x_8 & x_2 & - \end{bmatrix}, \qquad A_2 = \begin{bmatrix} x_1 & x_4 & x_7 \\ x_1 & x_2 & x_5 \\ x_1 & x_3 & x_6 \\ - & x_8 & x_7 \\ - & x_8 & x_5 \\ - & x_4 & x_8 \\ - & x_2 & x_8 \end{bmatrix}.$$

*La permutación correspondiente a* $A_1$ *minimiza el CSE, consiguiendo un CSE igual a* $2+3+3=8$ *y requiriendo* $\lceil \log_2 2 \rceil + \lceil \log_2 3 \rceil + \lceil \log_2 3 \rceil = 1+2+2=5$ *bits de selección. Por otra parte, la permutación correspondiente a* $A_2$ *consigue minimizar el número de bits de selección a expensas de un incremento del CSE, obteniendo un CSE igual* $1+4+4=9$ *y requiriendo* $\lceil \log_2 1 \rceil + \lceil \log_2 4 \rceil + \lceil \log_2 4 \rceil = 0+2+2=4$ *bits de selección.*

**Modelo de programación lineal entera 0/1**

Sea $P = \{P_1, \ldots, P_k\}$ un $k$-emparejamiento parcial en un grafo bipartito $G = (U \cup V, E)$. Se definen las variables binarias $x_{e,i}, y_{v,i} \in \mathbb{B}$ como

$$x_{e,i} = \begin{cases} 1 \text{ si } e \in P_i \\ 0 \text{ en otro caso} \end{cases} \quad \forall e \in E,\ i = 1, \ldots, k,$$

$$y_{v,i} = \begin{cases} 1 \text{ si } v \in V(P_i) \\ 0 \text{ en otro caso} \end{cases} \quad \forall v \in V,\ i = 1, \ldots, k;$$

y la variable $d_{j,i} \in \mathbb{B}$, como el valor del dígito de peso $2^j$ asociado a la representación binaria de $2^{\lceil \log_2 |V(P_i)| \rceil} - 1$ (el máximo valor representable con $\lceil \log_2 |V(P_i)| \rceil$ dígitos binarios). Con estas definiciones, el CSE asociado a cada emparejamiento parcial $P_i$ se calcula como $|V(P_i)| = \sum_{v \in V} y_{v,i}$, y el número de bits de selección de $P_i$, como $\lceil \log_2 |V(P_i)| \rceil = \sum_{j=1}^{D} d_{j,i}$, donde $D \geq \lceil \log_2 |V(P_i)| \rceil$.

El Problema del $k$-Emparejamiento Parcial Maximal Mínimo BC se puede modelar como el siguiente problema PLE (que difiere de la formulación PLE del Problema del $k$-Emparejamiento Parcial Maximal Mínimo sólo en las ecuaciones numeradas):

$$\text{minimizar } Q \sum_{i=1}^{k} \sum_{j=1}^{D} d_{j,i} + \sum_{i=1}^{k} \sum_{v \in V} y_{v,i}, \tag{4.10}$$

$$\text{donde } D = \lceil \log_2 \text{mín}\{|U|, |V|\} \rceil , \tag{4.11}$$

$$Q = k\,\text{mín}\{|U|, |V|\}, \tag{4.12}$$

$$\begin{aligned}
\text{sujeto a } & \sum_{e \in E(u)} x_{e,i} \leq 1, \quad \forall u \in U,\ i = 1, \ldots, k, \\
& \sum_{i=1}^{k} x_{e,i} \leq 1, \quad \forall e \in E, \\
& \sum_{i=1}^{k} \sum_{e \in E(u)} x_{e,i} = \text{mín}\{k, \text{grad}_G(u)\}, \quad \forall u \in U, \\
& x_{e,i} \leq y_{v,i}, \quad \forall e \equiv (u, v) \in E,\ i = 1, \ldots, k, \\
& y_{v,i} \leq \sum_{e \in E(v)} x_{e,i}, \quad \forall v \in V,\ i = 1, \ldots, k, \\
& 2^j d_{j,i} \leq \sum_{v \in V} y_{v,i} - 1, \quad j = 1, \ldots, D,\ i = 1, \ldots, k, \qquad (4.13) \\
& \sum_{v \in V} y_{v,i} - 1 \leq \sum_{j=1}^{D} 2^j d_{j,i}, \quad i = 1, \ldots, k, \qquad (4.14) \\
& d_{j,i} \geq d_{j+1,i}, \quad j = 1, \ldots, D-1,\ i = 1, \ldots, k, \qquad (4.15) \\
& x_{e,i} \in \mathbb{B}, \quad \forall e \in E,\ i = 1, \ldots, k, \\
& y_{v,i} \in \mathbb{B}, \quad \forall v \in V,\ i = 1, \ldots, k, \\
& d_{j,i} \in \mathbb{B}, \quad j = 1, \ldots, D,\ i = 1, \ldots, k.
\end{aligned}$$

Sea $P = \{P_1, \ldots, P_k\}$ la solución obtenida para el problema PLE, donde $P_i = \{e \in E : x_{e,i} = 1\}$. Puesto que $|P_i| \leq |U|$, se cumple que $|V(P_i)| \leq \text{mín}\{|U|, |V|\}$. Por lo tanto, $D$ (ecuación 4.11) es una cota del número de bits de selección de cada emparejamiento parcial, y $Q$ (ecuación 4.12), una cota del valor del CSE de $P$. Debido al valor de $Q$, los valores de la ecuación 4.10 ordenan lexicográficamente las diferentes soluciones factibles del problema.

La ecuación 4.13 asegura para cada $P_i$ que las variables $d_{j,i}$ correspondientes a los dígitos binarios con un peso mayor o igual que $|V(P_i)| - 1$ serán 0. La ecuación 4.14 garantiza que la variable $d_{j,i}$ correspondiente al dígito más significativo de la representación binaria de $|V(P_i)| - 1$ tendrá el valor 1. Finalmente, la ecuación 4.15 asegura que todos los dígitos binarios $d_{j,i}$ menos significativos que un dígito con valor 1 tomarán el valor 1.

### 4.4.2. *k*-Emparejamiento Parcial Maximal con CSE, número de bits de selección y cardinalidad ponderada mínimos

Tal como se estudió en el capítulo 3, aumentar el número de coincidencia de indeterminaciones entre las filas de la MSE permite que los algoritmos de agrupación encuentren mejores soluciones. Por lo tanto, en las estrategias de optimización de FSMIM con una fase de minimización de la selección de entradas previa a la fase de agrupación de estados, es conveniente que la solución encontrada por la fase de minimización presente el mayor número posible de coincidencia de indeterminaciones entre las filas de la MSE.

Sea $P = \{P_1, \ldots, P_k\}$ un $k$-emparejamiento parcial maximal en un grafo $G = (U \cup V, E)$ equivalente a la MSE $A \in MSE^{n \times k}$. Sea $A'$ la MSE definida por $P$. El número de indeterminaciones de cada columna de $A'$ es igual a $n - |P_i|$; por tanto, el número de indeterminaciones depende de la cardinalidad del emparejamiento parcial.

Tal como se estudió en la sección 3.5, la permutación de entradas realizada por el algoritmo 6 (que sitúa las indeterminaciones en los últimos elementos de cada fila) permite obtener el mínimo número de grupos. Esta permutación es la que genera una MSE con mayor número de coincidencias de indeterminaciones entre las filas. Resulta fácil comprobar que $A'$ presentará dicha permutación de entradas si y solo si $P$ cumple que $|P_i| \geq |P_j|$ para todo $i \geq j$ $(1 \leq i, j \leq k)$. Sin embargo, no se puede imponer esta restricción a las soluciones del Problema del $k$-Emparejamiento Parcial Maximal Mínimo debido a que, en ese caso, no estará garantizado que se encuentre el CSE mínimo.

**Definición 4.13.** *Se denominará* **cardinalidad ponderada** *de un $k$-emparejamiento parcial $P = \{P_1, \ldots, P_k\}$ al valor $\sum_{i=1}^{k} w_i |P_i|$, donde $w_i \in \mathbb{Z}$ $(1 \leq i \leq k)$ serán denominados pesos.*

En esta sección se presenta una variante al Problema del $k$-Emparejamiento Parcial Maximal Mínimo que intenta encontrar la solución con mayor número de coincidencias de indeterminaciones posible entre las soluciones con CSE mínimo. Dicho problema busca la solución con Coste de selección de entradas, número de Bits de selección y Cardinalidad ponderada mínimos, por lo que será denominado Problema del $k$-Emparejamiento Parcial Maximal Mínimo CBC; y se describe como el siguiente problema de optimización multiobjetivo:

Problema del $k$-Emparejamiento Parcial Maximal Mínimo CBC

*Instancia:* Un grafo bipartito $G = (U \cup V, E)$, un entero positivo $k \leq \Delta_U(G)$ y $k$ enteros no negativos $w_1, \ldots, w_k$.

*Tarea:* Buscar un $k$-emparejamiento parcial maximal $P = \{P_1, \ldots, P_k\}$ en $G$ tal que la tupla

$$\left( \sum_{i=1}^{k} |V(P_i)|, \sum_{i=1}^{k} \lceil \log_2 |V(P_i)| \rceil, \sum_{i=1}^{k} w_i |P_i| \right)$$

sea mínima en orden lexicográfico.

El objetivo del problema es encontrar la solución con menor cardinalidad ponderada de entre las que presentan el mínimo CSE y el mínimo número de bits de selección, en ese orden de prioridad. En promedio, mientras menor sea la cardinalidad ponderada de $P$, menor será la cardinalidad de los $P_i \in P$ de mayor peso y mayor la de los $P_i$ de menor peso. Por tanto, la MSE definida por $P$ presentará un mayor número de indeterminaciones en las columnas correspondientes a los emparejamientos parciales de mayor peso, lo que significa que el número de coincidencias de indeterminaciones aumentará en dichas columnas. Es decir, en el caso de que los pesos formen una sucesión creciente (por ejemplo, $w_i = i$), mientras menor sea la cardinalidad ponderada de una solución factible $P$, mayor será la concentración de indeterminaciones en los últimos elementos de cada fila de la MSE definida por $P$, aproximándonos a la MSE que se obtendría con la permutación de entradas del algoritmo 6.

### Modelo de programación lineal entera 0/1

Sea $P = \{P_1, \ldots, P_k\}$ un $k$-emparejamiento parcial en un grafo bipartito $G = (U \cup V, E)$ y sean $w_1, \ldots, w_k$ números enteros no negativos. El Problema del $k$-Emparejamiento Parcial Maximal Mínimo CBC se puede modelar como el siguiente problema PLE (que difiere de la formulación PLE del Problema del $k$-Emparejamiento Parcial Maximal Mínimo BC sólo en las ecuaciones numeradas):

$$\text{minimizar } Q\left( kD \sum_{i=1}^{k} \sum_{v \in V} y_{v,i} + \sum_{i=1}^{k} \sum_{j=1}^{D} d_{j,i} \right) + \sum_{i=1}^{k} w_i \sum_{e \in E} x_{e,i}, \qquad (4.16)$$

$$\text{donde } D = \lceil \log_2 \text{mín}\{|U|, |V|\} \rceil ,$$

$$Q = |U| \sum_{i=1}^{k} w_i, \tag{4.17}$$

$$\text{sujeto a } \sum_{e \in E(u)} x_{e,i} \leq 1, \quad \forall u \in U,\ i = 1, \ldots, k,$$

$$\sum_{i=1}^{k} x_{e,i} \leq 1, \quad \forall e \in E,$$

$$\sum_{i=1}^{k} \sum_{e \in E(u)} x_{e,i} = \text{mín}\{k, \text{grad}_G(u)\}, \quad \forall u \in U,$$

$$x_{e,i} \leq y_{v,i}, \quad \forall e \equiv (u, v) \in E,\ i = 1, \ldots, k,$$

$$y_{v,i} \leq \sum_{e \in E(v)} x_{e,i}, \quad \forall v \in V,\ i = 1, \ldots, k,$$

$$2^j d_{j,i} \leq \sum_{v \in V} y_{v,i} - 1, \quad j = 1, \ldots, D,\ i = 1, \ldots, k,$$

$$\sum_{v \in V} y_{v,i} - 1 \leq \sum_{j=1}^{D} 2^j d_{j,i}, \quad i = 1, \ldots, k,$$

$$d_{j,i} \geq d_{j+1,i}, \quad j = 1, \ldots, D-1,\ i = 1, \ldots, k,$$

$$x_{e,i} \in \mathbb{B}, \quad \forall e \in E,\ i = 1, \ldots, k,$$

$$y_{v,i} \in \mathbb{B}, \quad \forall v \in V,\ i = 1, \ldots, k,$$

$$d_{j,i} \in \mathbb{B}, \quad j = 1, \ldots, D,\ i = 1, \ldots, k,$$

Sea $P = \{P_1, \ldots, P_k\}$ la solución obtenida para el problema PLE, donde $P_i = \{e \in E : x_{e,i} = 1\}$. La cardinalidad ponderada de $P$ se calcula en la ecuación 4.16 como $\sum_{i=1}^{k} w_i \sum_{e \in E} x_{e,i}$. En la ecuación 4.17, $Q$ es una cota superior de la cardinalidad ponderada de $P$. Por otra parte, $kD$ es una cota superior del número de bits de selección de $P$. Por lo tanto, los valores de la ecuación 4.16 ordenan lexicográficamente las diferentes soluciones factibles del problema.

## 4.5. *k*-Emparejamiento Parcial Maximal Mínimo con ligaduras

Tal como han sido definidos, los problemas de optimización estudiados en este capítulo no se pueden utilizar en fases de minimización de selección de entradas posteriores a una fase de agrupación de estados. Esto es debido a que la MSE que resulta de la fase agrupación de estados corresponde a una

FSMIM con estados compuestos y en la que se seleccionan las constantes $\{0, 1\}$. Por una parte, en estas MSE una misma fila puede contener varios elementos con valor 0 o 1; por lo tanto, el grafo equivalente debe permitir aristas repetidas, es decir, debe ser un multigrafo. Por otra parte, no se pueden permutar libremente las filas correspondiente a un mismo estado compuesto, ya que algunas permutaciones de estas filas dan lugar a una MSE no equivalente.

Tal como se mostró en el capítulo 3, los algoritmos de agrupación de estados crean los grupos mediante operaciones de asignación exclusiva de indeterminaciones. Como consecuencia, se obtienen grupos en los que, para cada par de filas, existe una columna de la MSE en la que ambas filas tienen asignadas constantes diferentes. Esta propiedad es la que permite agrupar los estados; por lo tanto, para garantizar que la MSE obtenida es equivalente a la original, sólo se pueden aplicar permutaciones de entradas que no alteren esta relación entre las filas de cada grupo.

Se denominarán *constantes ligadas* a todas las constantes de un grupo que se encuentran en una misma columna. Dada una MSE generada por una fase de agrupación de estados, un mecanismo para garantizar que se obtiene una MSE equivalente tras la minimización de la selección de entradas es permitir sólo permutaciones de entradas que mantengan las constantes ligadas en una misma columna.

A continuación se propone una extensión del modelo PLE del Problema del $k$-Emparejamiento Parcial Maximal Mínimo y del algoritmo 11 que permite este problema y sus variantes puedan ser utilizados para la minimización de la selección de entradas de una MSE obtenida tras la agrupación de estados.

Sea $G = (U \cup V, E, \gamma)$ un multigrafo bipartito de aristas etiquetadas, donde $U$ y $V$ son los conjuntos de vértices, $E$ es el conjunto de aristas y $\gamma : E \to U \times V$ es la función que mapea las aristas en los pares de vértices. Una arista $e \in E$ es incidente en los vértices $u \in U$ y $v \in V$ si $\gamma(e) = (u, v)$.

Todas las definiciones de la sección 4.2 son válidas para $G$. Por lo tanto, $G$ puede formar parte una instancia del Problema del $k$-Emparejamiento Parcial Maximal Mínimo.

Sea $(A, R)$ una solución de un algoritmo de agrupación de estados, donde $A = (a_{ij}) \in MSE^{n \times k}$ y $R = \{R_1, \ldots, R_{|R|}\}$ es el conjunto de grupos (es decir, una partición de $\{1, \ldots, n\}$). A partir de $A$ se puede construir un multigrafo bipartito $G = (U \cup V, E, \gamma)$, que será denominado grafo equivalente

a $A$, donde

$$\begin{aligned} U &= \{1, \ldots, n\}, \\ V &= \{a_{i,j} : 1 \leq i \leq n,\ 1 \leq j \leq k,\ a_{ij} \text{ no es una ESI}\}, \\ E &= \{(i,j) : a_{i,j} \text{ no es una ESI}\}, \\ \gamma((i,j)) &= (i, a_{i,j}). \end{aligned}$$

A partir de $A$, $R$ y $G$ se define el conjunto de *grupos de aristas ligadas* $L = \{L_{l,j} : 1 \leq l \leq |R|,\ 1 \leq j \leq k\}$, donde $L_{l,j} = \{(i,j) \in E : i \in R_l,\ a_{i,j} \in \{0,1\}\}$ es el conjunto de aristas correspondientes a las constantes ligadas del grupo $l$ que se encuentran en la columna $j$. Obviamente, se cumple que $|L| = k|R|$.

Sea $P = \{P_1, \ldots, P_k\}$ un $k$-emparejamiento parcial en un grafo bipartito $G = (U \cup V, E)$ y sea $A'$ la MSE definida por $P$. Si cada grupo de aristas ligadas se asigna a un mismo emparejamiento parcial (es decir, si para todo $P_i \in P$ y todo $L_{l,j} \in L$ se cumple que $L_{l,j} \subseteq P_i \ \vee \ P_i \cap L_{l,j} = \emptyset$), entonces las constantes ligadas de $A$ se encontrarán en una misma columna de $A'$.

El modelo PLE del Problema del $k$-Emparejamiento Parcial Maximal Mínimo ha sido extendido (con la ecuación numerada) para obtener soluciones en las que cada grupo de aristas ligadas pertenece a un mismo emparejamiento parcial:

$$\begin{aligned} \text{minimizar } & \sum_{i=1}^{k} \sum_{v \in V} y_{v,i} \\ \text{sujeto a } & \sum_{e \in E(u)} x_{e,i} \leq 1, \quad \forall u \in U,\ i = 1, \ldots, k, \\ & \sum_{i=1}^{k} x_{e,i} \leq 1, \quad \forall e \in E, \\ & \sum_{i=1}^{k} \sum_{e \in E(u)} x_{e,i} = \text{mín}\{k, \text{grad}_G(u)\}, \quad \forall u \in U, \\ & x_{e,i} \leq y_{v,i}, \quad \forall e \in E,\ (u,v) = \gamma(e),\ i = 1, \ldots, k, \\ & y_{v,i} \leq \sum_{e \in E(v)} x_{e,i}, \quad \forall v \in V,\ i = 1, \ldots, k, \\ & x_{e,i} = x_{e',i}, \quad \forall e, e' \in L_{l,j},\ \forall L_{l,j} \in L,\ i = 1, \ldots, k, \qquad (4.18) \\ & x_{e,i} \in \mathbb{B}, \quad \forall e \in E,\ i = 1, \ldots, k, \\ & y_{v,i} \in \mathbb{B}, \quad \forall v \in V,\ i = 1, \ldots, k, \end{aligned}$$

Sea $P = \{P_1, \ldots, P_k\}$ la solución obtenida, donde $P_i = \{e \in E : x_{e,i} = 1\}$. La ecuación 4.18 garantiza que las aristas de un mismo grupo $L_{l,j}$ pertenecerán a un mismo $P_i$.

El resto de modelos PLE pueden ser extendidos de manera similar; sin embargo, no se ha implementado para el Problema del $k$-Emparejamiento Parcial Maximal Mínimo CBC debido a que este ha sido propuesto para ser utilizado en una fase inicial de minimización de la selección de entradas.

El algoritmo 12 muestra la modificación propuesta del algoritmo 11 para garantizar que los grupos de aristas ligadas pertenezcan a un mismo emparejamiento parcial. Obsérvese que el conjunto $L$ de grupos de aristas ligadas, que el algoritmo recibe como entrada, está definido como $L = \{L_{l,j} : 1 \leq l \leq r,\ 1 \leq j \leq k\}$, donde $r = \frac{|L|}{k}$ (no es necesario especificar como entrada al algoritmo el valor de $r$ debido a que este se puede obtener a partir de $k$ y de la cardinalidad de $L$).

Por otra parte, a diferencia del algoritmo 11, el algoritmo 12 inicializa cada emparejamiento parcial $P_i$ con las aristas correspondientes a las constantes ligadas de la columna $i$ de la MSE (línea 1). Desde un punto de vista cualitativo, esta inicialización establece la diferencia fundamental entre las soluciones obtenidas por el modelo PLE y las obtenidas por el algoritmo. Se dirá que un grupo de aristas está asignado a un emparejamiento parcial $P_i$ si todas las aristas del grupo pertenecen a $P_i$. En las soluciones del modelo PLE, los diferentes grupos de aristas $L_{lj}$ $(1 \leq l \leq r)$ correspondientes a un mismo valor de $j$ pueden estar asignados a diferentes emparejamientos parciales. Sin embargo, en las soluciones del algoritmo, todos estos grupos están asignados al mismo emparejamiento parcial $(P_j)$.

Puesto que $G$ es un multigrafo (a diferencia del algoritmo 11, en el que $G$ se define como un grafo), debe redefinirse $\alpha_G(v, E')$ como $\alpha_G(v, E') = \{e \in E(u) : (u, v) = \gamma(e),\ u \notin U(E')\}$.

**Algoritmo 12** Algoritmo voraz para el PROBLEMA DEL $k$-EMPAREJAMIENTO PARCIAL MAXIMAL MÍNIMO con ligaduras

**Entrada:**
Multigrafo bipartito $G = (U \cup V, E, \gamma)$
Entero positivo $k$
Grupos de aristas ligadas $L = \{L_{l,j} : 1 \leq l \leq r,\ 1 \leq j \leq k\}$, con $r = \frac{|L|}{k}$.
**Salida:** $k$-emparejamiento parcial maximal $P = \{P_1, \ldots, P_k\}$

1: $P_i \leftarrow \bigcup_{l=1}^{r} L_{l,i}$ para todo $P_i \in P$
2: $\kappa \leftarrow |E_G| - \sum_{u \in U_G} \text{mín}\{k, \text{grad}_G(u)\}$
3: **mientras** $|E_G| > \kappa$ **hacer**
4: $\quad (v, P_i) \leftarrow \underset{(v,P_i) \in V \times P}{\arg\max} |\alpha_G(v, P_i)|$
5: $\quad P_i \leftarrow P_i \cup \alpha(v, P_i)$
6: $\quad G \leftarrow G - \alpha_G(v, P_i)$
7: **fin mientras**

Capítulo 5

# Resultados experimentales

En términos generales, el presente capítulo evalúa la influencia de las aportaciones de esta tesis doctoral en las prestaciones de las implementaciones de máquinas de estados. Uno de los objetivos de este estudio es cuantificar las mejoras obtenidas por las diferentes aportaciones en la generación e implementación de FSMIM, evaluando en la medida de lo posible cada aportación de manera independiente. Por otra parte, se realizará un estudio comparativo entre las prestaciones de las implementaciones de FSM y las prestaciones de las FSMIM obtenidas con las nuevas estrategias de optimización. Como referencia para esta comparativa se emplearán las arquitecturas convencionales de FSM (véase las secciones 1.2.1 y 1.2.5) y las arquitecturas usadas por la herramienta de síntesis empleada en los experimentos.

En la sección 5.1 se proporcionan algunos detalles relacionados con la generación automática de FSMIM, mientras que en la sección 5.2 se describen las pruebas realizadas.

En la sección 5.3 se compara la arquitectura FSMIM-S con la arquitectura FSMIM-T (que es la arquitectura de referencia en esta comparativa). Este estudio pretende evaluar la nueva arquitectura de FSMIM propuesta (es decir, la arquitectura FSMIM-S) al margen de las nuevas estrategias de optimización. Con este fin, las FSMIM comparadas en el estudio han sido generadas utilizando exclusivamente la estrategia de optimización de referencia, compuesta por los algoritmos propuestos en [García-Vargas, 2006], que fue el punto de partida para el trabajo de investigación presentado en esta tesis doctoral. Por otra parte, en esta misma sección se estudia la influencia del ajuste de profundidad (que se describe en la sección 5.1.2) sobre las implementaciones de FSMIM.

En las secciones 5.4 y 5.5 se estudia la influencia de las nuevas estrategias de optimización en las arquitecturas FSMIM-T y FSMIM-S, respectivamente.

Finalmente, en la sección 5.6 se realiza un estudio comparativo entre las implementaciones de FSM y las implementaciones de las FSMIM generadas con las nuevas estrategias.

Conviene aclarar en este punto que en el resto del capítulo se utilizará el término *grupo* para hacer referencia a los estados de una FSMIM[1]. Por el contrario, el término *estado* se reservará para designar a los estados de la FSM.

## 5.1. Generación automática de FSMIM

### 5.1.1. Estrategias de optimización de FSMIM

Las FSMIM empleadas en los experimentos han sido generadas con las herramientas FSMIM-Gen 1.2 (véase la sección H.1), que implementa la estrategia de optimización de referencia, y FSMIM-Gen 2.0 (véase la sección H.2), que implementa las nuevas estrategias propuestas en esta tesis.

A partir de la descripción de una FSM en formato *KISS2* (véase la sección B.1), ambas herramientas permiten generar la descripción de la FSMIM correspondiente en formato *KISS2IM* (véase el apéndice F)

#### 5.1.1.1. Estrategia de optimización de referencia

Las FSMIM generadas con la herramienta FSMIM-Gen 1.2 serán utilizadas como referencia para cuantificar las mejoras obtenidas con las nuevas estrategias propuestas en esta tesis. Esta herramienta ha sido utilizada para el estudio experimental en [García-Vargas y Senhadji-Navarro, 2015]. Se trata de una versión actualizada de la herramienta que se usó en el estudio experimental realizado en [García-Vargas et al., 2005, 2007], que implementaba los algoritmos propuestos en [García-Vargas, 2006].

La estrategia de optimización de FSMIM empleada en FSMIM-Gen 1.2, que será denominada *estrategia M-ref*, consta de dos fases: una fase inicial de minimización de la selección de entradas, que implementa el algoritmo basado en el Problema de la Suma de Subconjuntos Compatibles

[1]Se ha decidido utilizar este término por dos motivos: en primer lugar, indica claramente que se ha aplicado la agrupación de estados a todas las FSMIM analizadas y, en segundo lugar, evita la ambigüedad del término *estado*, que puede hacer referencia tanto a los estados de la FSM como a los estados de la FSMIM equivalente.

(véase la sección 4.1.3); y una fase final de agrupación de estados, que implementa el algoritmo de agrupación lexicográfica (véase el algoritmo 3 en la sección 3.3). Los resultados de la estrategia M-ref serán utilizados como referencia para cuantificar las mejoras obtenidas por los nuevos algoritmos propuestos en esta tesis, implementados en la herramienta FSMIM-Gen 2.0[2].

#### 5.1.1.2. Estrategias de optimización propuestas

En este apartado se describen las nuevas estrategias que se proponen para la optimización de FSMIM, disponibles en la herramienta FSMIM-Gen 2.0. En la descripción, *minimización* hace referencia a la minimización de la selección de entradas; y *agrupación*, a la agrupación de estados.

**Estrategia M-PLE**

La estrategia consta de tres fases: minimización, agrupación y minimización (en ese orden). Las dos fases de minimización se basan en los modelos PLE del Problema del $k$-Emparejamiento Parcial Maximal Mínimo y sus variantes.

1. La primera fase (de minimización) emplea el modelo PLE del Problema del $k$-Emparejamiento Parcial Maximal Mínimo CBC (véase la sección 4.4.2) para realizar la minimización. Puesto que comienza con una fase de minimización, esta estrategia favorece la obtención de implementaciones de FSMIM con bancos de selectores de entradas más simples.

2. La segunda fase (de agrupación) implementa el algoritmo lexicográfico con control de tamaño (véase el algoritmo 5 en la sección 3.4). Debido a que los subproblemas de minimización y la agrupación tienen objetivos en conflicto (véase la sección 2.4), las soluciones obtenidas por la primera fase repercuten negativamente en la agrupación; es decir, en términos generales no será posible obtener una FSMIM con una ROM de tamaño mínimo.

3. De manera recíproca, la fase de agrupación afecta negativamente en la solución obtenida por la primera fase, provocando un aumento de la complejidad del banco de selectores de entradas y, en el caso de la arquitectura FSMIM-T, del ancho de la ROM. Para compensar dicho

[2]Esta herramienta, a diferencia de FSMIM-Gen 1.2, no está disponible aún para su descarga.

aumento, esta estrategia termina con una fase de minimización. En el caso de la arquitectura FSMIM-S, se utiliza el modelo PLE del Problema del $k$-Emparejamiento Parcial Maximal Mínimo (véase la sección 4.2); y en el caso de la arquitectura FSMIM-T, el modelo PLE del Problema del $k$-Emparejamiento Parcial Maximal Mínimo BC (véase la sección 4.4.1). En ambos casos, los modelos han sido extendidos con ligaduras (véase la sección 4.5) para que se mantengan los grupos encontrados en la segunda fase.

**Estrategia M-voraz**

El número y tipo de fases de esta estrategia es igual que en la estrategia M-PLE. La diferencia radica en que la primera fase utiliza el algoritmo voraz estudiado para buscar soluciones al Problema del $k$-Emparejamiento Parcial Maximal Mínimo (algoritmo 11, sección 4.2); y la última fase, la versión de dicho algoritmo con ligaduras (algoritmo 12, sección 4.5).

**Estrategia A-PLE**

La estrategia consta de dos o cuatro fases, dependiendo de la arquitectura en la que se vaya a implementar la FSMIM generada. En el caso de la arquitectura FSMIM-S, se aplica una fase de agrupación seguida de una fase de minimización; sin embargo, en el caso de la arquitectura FSMIM-T, las fases que se aplican son agrupación, minimización, agrupación y minimización (en ese orden). Las dos fases de minimización se basan en los modelos PLE del Problema del $k$-Emparejamiento Parcial Maximal Mínimo. Al comenzar por una fase de agrupación, esta estrategia intenta favorecer la reducción del tamaño de la ROM frente a la simplificación del banco de selectores de entrada.

Puesto que hay ligeras diferencias en cuanto a los algoritmos utilizados, se describirán por separado para cada arquitectura:

- **Estrategia A-PLE para la arquitectura FSMIM-T:**

  1. La primera fase (de agrupación) implementa el algoritmo de agrupación máxima (algoritmo 6, sección 3.5). En esta fase se obtiene el mínimo número de estados. Tal como se explicó en la sección 2.4.2.1, reducir el número de estados hasta su valor mínimo puede incrementar el tamaño de la ROM respecto al tamaño mínimo, ya que este depende también de los bits de selección. A pesar de ello, este algoritmo no comprueba el tamaño de la ROM en cada paso debido a que no puede estimar el ancho que tendrá

la ROM tras la fase de minimización posterior. Por tanto, tras la siguiente fase de minimización debe realizarse una nueva fase de agrupación.

2. La segunda fase (de minimización) se basa en el modelo PLE del Problema del $k$-Emparejamiento Parcial Maximal Mínimo BC (sección 4.4.1) extendido con ligaduras (sección 4.5). Las soluciones de esta fase se verán perjudicadas por la primera fase; es decir, en términos generales, el banco de multiplexores obtenido será más complejo que el de la estrategia M-PLE.
3. La tercera fase (de agrupación) implementa el algoritmo lexicográfico con control de tamaño (algoritmo 5, sección 3.4). Debido a que se han minimizado los bits de selección en la fase anterior, se ha reducido el ancho de la ROM; por tanto, se puede realizar el control de tamaño durante la agrupación. Previamente deben eliminarse de la MSE las constantes establecidas por la primera fase (lo que equivale a deshacer los grupos establecidos en dicha fase). El algoritmo de agrupación que se aplica en esta fase no garantiza que se alcance el mínimo número de grupos; por tanto, los resultados podrían empeorar respecto a la primera fase. Sin embargo, puesto que existe una solución con el mínimo número de grupos para la distribución de las indeterminaciones de la MSE, el algoritmo tiene más posibilidades de alcanzar dicho mínimo.
4. Al igual que en la estrategia M-PLE, se termina con una fase de minimización, que en esta estrategia es idéntica a la segunda fase. Tras completar la fase, se comparan las soluciones obtenidas por todas las fases y se selecciona como solución final la que corresponde a una FSMIM-T con la ROM de menor tamaño.

- **Estrategia A-PLE para la arquitectura FSMIM-S:**

1. La primera fase (de agrupación) implementa el algoritmo de agrupación máxima con control de tamaño (algoritmo 7, sección 3.5). En esta fase se obtiene el número de estados que minimiza el tamaño de la ROM (por lo tanto, es la solución óptima para la agrupación).
2. La segunda fase (de minimización) se basa en el modelo PLE del Problema del $k$-Emparejamiento Parcial Maximal Mínimo (sección 4.2) extendido con ligaduras (sección 4.5). Al igual que ocurre con la arquitectura FSMIM-T, las soluciones de esta fase se verán perjudicadas por la primera fase; es decir, en

términos generales, el banco de selectores de entradas será más complejo que el obtenido en la estrategia M-PLE.

**Estrategia A-voraz**

El número y tipo de fases de esta estrategia coincide con el de la estrategia A-PLE, la única diferencia es que las fases de minimización implementan el algoritmo voraz para el Problema del $k$-Emparejamiento Parcial Maximal Mínimo con ligaduras (algoritmo 12, sección 4.5).

#### 5.1.1.3. Clasificación de las estrategias

Las estrategias de optimización de FSMIM mencionadas pueden clasificarse según el orden de sus fases, en los siguientes tipos:

**De tipo AM:** presentan una fase inicial de agrupación de estados seguida de una fase de minimización de selección de entradas. Pertenecen a este tipo las estrategias A-PLE y A-voraz.

**De tipo MA:** presentan una fase inicial de minimización de selección de entradas seguida de una fase de agrupación de estados. Pertenecen a este tipo las estrategias M-PLE, M-voraz y M-ref.

Según el tipo de algoritmo que se usa en la fase de minimización de selección de entradas, las estrategias se clasifican en los siguientes tipos:

**De tipo PLE:** utilizan modelos PLE para realizar la minimización de selección de entradas. Pertenecen a este tipo las estrategias A-PLE y M-PLE.

**De tipo voraz:** emplean algoritmos voraces para realizar la minimización de selección de entradas. Pertenecen a este tipo las estrategias A-voraz y M-voraz.

### 5.1.2. Ajuste de la profundidad de la FSMIM

La profundidad de los bloques de memoria empotrados de los dispositivos FPGA es configurable; sin embargo, el conjunto de valores permitidos está limitado [Xilinx, 2011c; Altera, 2015]. Esto implica que no es posible implementar memorias con cualquier profundidad usando bloques. Por tanto, en los casos en los que no se puede obtener la profundidad requerida, debe implementarse una ROM con profundidad mayor.

La herramienta FSMIM-Gen 1.2 genera las FSMIM con el menor tamaño que pueden conseguir sus algoritmos. Por otra parte, la herramienta FSMIM-Gen 2.0 se comporta de forma similar cuando el tamaño de la FSMIM se mide en bits (es decir, cuando la función *TAM* utilizada por los algoritmos de agrupación se define como el número de bits que ocupa la ROM de la FSMIM).

Generalmente, cuando estas FSMIM se implementan utilizando bloques de memoria empotrados, la profundidad de la ROM construida con los bloques es mayor que la profundidad de la memoria requerida por la FSMIM (que será denominada *profundidad de la FSMIM*). En estos casos, puede incrementarse la profundidad de la FSMIM para que se ajuste a la de la ROM implementada con bloques de memoria, permitiendo que el número de grupos obtenido por las estrategias de optimización crezca sin aumentar el consumo de bloques. El procedimiento por el que se determina la nueva profundidad de la FSMIM será denominado *ajuste de profundidad*. Dicho procedimiento consiste en determinar el número de grupos que debe tener la FSMIM para que su profundidad se aproxime a la de la ROM implementada con bloques, sin que aumente el número de bloques de memoria necesarios para su implementación. Las FSMIM resultantes serán denominadas *FSMIM con ajuste de profundidad*. Por otra parte, las FSMIM en las que no se ha realizado este ajuste (que tienen el tamaño mínimo alcanzable por los algoritmos de optimización) serán denominadas *FSMIM sin ajuste de profundidad*.

Debido a que permite incrementar el número de grupos, el ajuste de profundidad puede suponer mejoras, tanto en área como en velocidad, en aquellas implementaciones de FSMIM en las que se puede aplicar. Respecto al consumo de bloques, en la arquitectura FSMIM-T el número de bits de selección de entradas puede disminuir al aumentar el número de grupos, reduciéndose el ancho de la ROM y, por tanto, su tamaño (véase la sección 2.4.2.1). Es decir, el ajuste de profundidad favorece la disminución del consumo de bloques de las FSMIM-T. Sin embargo, en la arquitectura FSMIM-S el ajuste de profundidad no afecta al consumo de bloques. Esto es debido a que, por una parte, el ancho de la FSMIM-S no depende del número de grupos (véase la sección 2.4.2.2) y, por otra, el ajuste de profundidad no modifica la profundidad de la ROM implementada por bloques, sino sólo la de la FSMIM. Por tanto, en el caso de la arquitectura FSMIM-S, el ajuste de profundidad no afecta ni al ancho ni a la profundidad de la ROM implementada con bloques.

Respecto al consumo de LUT, el incremento del número de grupos puede reducir el coste de selección de entradas en ambas arquitecturas (véanse las

secciones 2.4.2.1 y 2.4.2.2), lo que favorece la reducción del consumo de LUT y el incremento de la frecuencia de operación máxima.

Para realizar el ajuste de profundidad con la herramienta FSMIM-Gen 2.0, es suficiente con definir la función *TAM* como el número de bloques que ocupa la ROM de la FSMIM[3] (véase la sección G.2.1.1). Con esta función *TAM*, el ajuste de profundidad es realizado automáticamente por el procedimiento de agrupación lexicográfico con control de tamaño (algoritmo 4, sección 3.4).

Sin embargo, no es posible realizar el ajuste de profundidad con las versiones anteriores de la herramienta FSMIM-Gen 1.2. Esto es debido a que el algoritmo de agrupación que implementa (algoritmo 1, sección 3.2) siempre intenta obtener el mínimo número de grupos que puede alcanzar. Para solucionar este problema ha sido necesario establecer un mecanismo alternativo, que incluye una pequeña modificación de dicho algoritmo (véase la sección H.1.1).

## 5.2. Descripción de las pruebas realizadas

Independientemente de la arquitectura utilizada, las prestaciones que se obtienen al implementar las FSMIM en un dispositivo FPGA dependen tanto de las propiedades de la FSMIM como de las características de los recursos disponibles en el dispositivo. Además, debido al ajuste de profundidad, las características de los bloques de memoria empotrados puede influir en las propiedades de la FSMIM generada, tanto en el número de grupos alcanzado por el algoritmo de agrupación como en la complejidad del banco de selectores de entradas (tal como se ha estudiado en la sección 5.1.2).

Por tanto, las pruebas de implementación sobre dispositivos reales no son la mejor opción para evaluar la influencia que tienen las arquitecturas y estrategias de optimización propuestas sobre las propiedades de las FSMIM. Como consecuencia, además de las pruebas de implementación sobre dispositivos FPGA, se ha considerado conveniente la realización de pruebas independientes del dispositivo.

[3]La función *TAM* debe computar el número de bloques ocupados, no la fracción del bloque que está siendo usada.

### 5.2.1. Descripción de las pruebas independientes del dispositivo

El objetivo de las pruebas independientes del dispositivo es cuantificar la influencia de las arquitecturas y estrategias de optimización sobre las propiedades de las FSMIM, sin que la estimación obtenida se vea afectada por las características de los dispositivos FPGA. Los resultados obtenidos permiten medir la capacidad potencial de optimización de las diferentes estrategias.

Los experimentos se han realizado generando las FSMIM sin ajuste de profundidad y calculando los parámetros estudiados a partir de la descripción en formato *KISS2IM* obtenida. Para cada FSMIM se ha medido el tamaño en bits de la ROM de la FSMIM utilizando la ecuación 2.17 y la ecuación 2.23. Esta medida será denominada *tamaño de la FSMIM* o *tamaño genérico*[4] de la FSMIM. El tamaño genérico ofrece una estimación de los valores mínimos alcanzables por las diferentes estrategias de optimización y arquitecturas de FSMIM estudiadas. Además, se ha calculado el coste de selección de entradas (Definición 2.20)[5], que permite estimar la complejidad del banco de selectores de entradas de la FSMIM. Por otra parte, se denominará *profundidad* de la FSMIM a la profundidad (medida en número palabras) de la ROM de la FSMIM.

### 5.2.2. Descripción de las pruebas de implementación

Las pruebas de implementación permiten evaluar la influencia de las aportaciones de esta tesis sobre la implementación de FSM en dispositivos FPGA. El análisis de los resultados experimentales ha permitido, por una parte, estudiar las mejoras obtenidas en las prestaciones de las implementaciones FSMIM, tanto en consumo de recursos como en velocidad. Por otra parte, se ha realizado un estudio comparativo entre las prestaciones de las implementaciones FSMIM generadas por las nuevas estrategias y las prestaciones de otras implementaciones FSM.

A partir de la descripción en formato *KISS2* de las FSM, se han generado todas las descripciones VHDL necesarias para realizar los experimentos. Posteriormente, estas descripciones han sido sintetizadas e implementadas

[4]En general, cuando sea necesario para evitar la ambigüedad, se utilizará el adjetivo *genérico* para diferenciar los resultados obtenidos en las pruebas independientes del dispositivo de los obtenidos en las pruebas de implementación, que son específicos del dispositivo FPGA utilizado.

[5]Debido a que el acrónimo *CSE* hace referencia a una función definida sobre una MSE, se ha decidido utilizar el nombre completo para designar a esta medida en lugar del acrónimo.

en el dispositivo FPGA mediante el entorno integrado de diseño FPGA *ISE WebPACK* versión 14.6 de Xilinx®. El dispositivo FPGA utilizado ha sido XC6SLX75T-4, de la familia Spartan-6 de Xilinx (véase el apéndice K).

A diferencia de las pruebas independientes del dispositivo, en las pruebas de implementación se han empleado las FSMIM con ajuste de profundidad. Cada descripción en formato *KISS2IM*, obtenida con la herramienta FSMIM-Gen 1.2 o FSMIM-Gen 2.0, se ha convertido en una descripción VHDL de la arquitectura FSMIM correspondiente (véase el apéndice J).

Los resultados de implementación obtenidos con las arquitecturas FSMIM-T y FSMIM-S han sido comparados con las siguientes arquitecturas de FSM:

- Arquitectura convencional de FSM basada en memoria (véase la sección 1.2.5). Para su implementación se ha empleado la descripción VHDL que se detalla en la sección I.2.

  Esta arquitectura será denominada *FSM-ROM*.

- Arquitectura de FSM empleada por *ISE WebPACK* basada en celdas lógicas. Esta arquitectura corresponde a la implementación en FPGA de las arquitecturas convencionales de FSM basadas en puertas lógicas (véase sección 1.2.1).

  Para su implementación se ha utilizado una descripción VHDL de comportamiento que permite inferir una FSM a la herramienta de síntesis (véase la sección I.1), lo que garantiza la aplicación de las técnicas de optimización adecuadas para la minimización y la codificación de estados.

  Esta arquitectura será denominada *ISE-LUT*.

- Arquitectura de FSM proporcionada por *ISE WebPACK* basada en memoria. Se trata de una arquitectura basada en memoria diferente a la convencional.

  Para su implementación se ha empleado la misma descripción VHDL que en la arquitectura ISE-LUT, pero configurando *ISE WebPACK* para que implemente la FSM con bloques de memoria empotrados (ya que por defecto la herramienta implementa la FSM utilizando sólo celdas lógicas).

  Esta arquitectura será denominada *ISE-BLOQ*.

Para cada estrategia y arquitectura comparada, se han realizado dos implementaciones: una optimizada en área y otra en velocidad. En la sección K.2 se muestran algunos detalles sobre la configuración del proceso de

síntesis e implementación. Los parámetros analizados han sido el consumo de bloques de memoria, el consumo de LUT y la frecuencia de operación máxima, obtenidos tras la fase de colocación y rutado[6]. En este contexto, se denominará *tamaño de la FSMIM* al tamaño de la ROM medido en número de bloques. Para medir la frecuencia de operación máxima, se han registrado las entradas y salidas de la FSM en las descripciones VHDL de todas las arquitecturas.

### 5.2.3. Baterías de pruebas

Los resultados experimentales se han obtenido a partir de tres baterías de pruebas distintas: una batería de pruebas estándar y dos baterías formadas por FSM generadas automáticamente. De estas, se han seleccionado las FSM que cumplen al menos uno de los siguientes requisitos, necesarios para generar una FSMIM:

1. El número de canales de selección de la FSMIM (es decir, el número máximo de entradas efectivas de la FSM) debe ser menor que el número de entradas de la FSM.

2. Debe existir más de un estado con entradas seleccionadas indeterminadas.

Dichos requisitos no garantizan que la técnica se pueda aplicar con éxito. En el peor caso, algunas estrategias de optimización generan una FSMIM que no consigue reducir el tamaño de ROM. Por ejemplo, en una FSM que cumpla solamente el requisito 2 sólo puede reducirse el tamaño de la FSMIM mediante la fase de agrupación de estados. Sin embargo, en una estrategia de optimización de tipo MA podría ocurrir que la fase de minimización inicial desordenara las indeterminaciones de forma que no se pudiera agrupar ningún estado en la fase de agrupación posterior. En este caso, dicha estrategia no reduciría el tamaño de la ROM (realmente, puesto que no se reduce el número de entradas ni se agrupa ningún estado, en este caso no se generaría una FSMIM).

En un principio, se consideró la posibilidad de excluir de los experimentos los casos en los que la técnica no se puede aplicar con éxito; sin embargo,

[6]La fase de colocación y rutado es la última fase del proceso de implementación de un diseño en FPGA (sin considerar la generación del *bitstream*). Esta fase se encarga de seleccionar los elementos del dispositivo que se usarán para implementar el diseño y de su interconexión.

finalmente se decidió que la comparación con el resto de técnicas de implementación de FSM debía incluir también los casos más desfavorables para las implementaciones de FSMIM.

Con el objeto de describir las características de las baterías de pruebas utilizadas, se ha calculado el número de entradas, de salidas y de estados de cada FSM. Además, para cada una de ellas, se han calculado los siguientes parámetros de las FSMIM:

**Número de canales de selección de la FSMIM**
: Este parámetro, al igual que el resto de parámetros calculados, sólo depende de las características de la FSM y no del proceso de optimización de la FSMIM.

**Profundidad inicial de la FSMIM**
: Es la profundidad de la ROM de la FSMIM antes de agrupar estados. Viene determinada por el número de estados de la FSM y el número de canales de selección de la FSMIM.

**Tamaño inicial de la FSMIM-S**
: Es el tamaño de la FSMIM-S antes de agrupar estados. Viene determinado por el número de salidas de la FSM, el número de bits de codificación de estados de la FSM y la profundidad inicial de la FSMIM. En el caso de las FSMIM-T, el tamaño inicial no se puede calcular a partir de las características de la FSM debido a que el número de bits de selección depende del proceso de optimización.

**Reducción inicial de profundidad de la FSMIM**
: Este parámetro mide la reducción de la profundidad de la ROM de la FSMIM debida exclusivamente a la selección de entradas efectivas. La reducción está calculada respecto a la profundidad de la ROM de la FSM-ROM. El valor 0 indica que la selección de entradas por sí sola no puede reducir el tamaño de la ROM. En estos casos, las reducciones obtenidas sólo pueden ser debidas a la agrupación de estados.

**Ratio de indeterminaciones de la MSE**
: Este parámetro mide el número de elementos indeterminados de la MSE respecto al número total de elementos. Su valor, que siempre es menor que 1, puede ser utilizado como un primer indicador de la efectividad de la fase de agrupación de estados: en términos generales, la efectividad debe crecer con el aumento del ratio de indeterminaciones. El valor 0 indica que la agrupación de estados no es posible debido a que no hay indeterminaciones en la MSE.

La primera batería de pruebas utilizada, denominada *MCNC FSM Benchmarks* [Yang, 1991], consiste en un conjunto de 46 FSM descritas en formato *KISS2* (la sección B.1 contiene una descripción detallada de este formato). Desde su creación en el año 1988, esta batería de pruebas ha sido ampliamente utilizada en trabajos relacionados con la implementación electrónica de máquinas de estado [Avedillo et al., 1994; Chow et al., 1996; El-Maleh et al., 2006; Janarthanan et al., 2007]. En el marco de esta tesis doctoral, la batería de pruebas MCNC ha sido utilizada para el estudio experimental de los trabajos publicados en [Senhadji-Navarro et al., 2004; García-Vargas et al., 2005, 2007; García-Vargas y Senhadji-Navarro, 2015].

De las 46 FSM que componen la batería de pruebas, el 56 % (26 FSM) cumplen los requisitos para generar una FSMIM. El apéndice B contiene la tabla de características de las FSM seleccionadas (además de información adicional sobre la batería de pruebas MCNC). La tabla 5.1 muestra un resumen estadístico. Las tres primeras filas contienen el número de entradas, de estados y de salidas de la FSM. Las filas "Núm. canales", "Prof. inic.", "Tam. inic." y "Red. inic." contienen los valores estadísticos del número de canales de selección, la profundidad inicial (medida en Kpalabras), el tamaño inicial (medido en Kbits) y la reducción inicial de profundidad de la FSMIM, respectivamente. La fila "Ratio ind. MSE" contiene el ratio de indeterminaciones de la MSE.

Con el objeto de ampliar el número de FSM, se han utilizado dos baterías de pruebas adicionales que han sido generadas con diferentes herramientas. Estas baterías de pruebas se han identificado con el nombre BP-$x$ donde $x$ es el número medio de estados de sus FSM. La primera de ellas (denominada BP-24), ha sido generada por la herramienta *FSM Benchmark Generator* [Józwiak et al., 2004] y obtenida gracias a la gentileza del profesor Lech Józ-

**Tabla 5.1:** Resumen estadístico de las características de la batería de pruebas MCNC.

| | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| Núm. Entradas | 9,0 | 5,5 | 3,0 | 6,2 | 7,5 | 8,8 | 27,0 |
| Núm. Estados | 30,3 | 30,6 | 4,0 | 15,2 | 19,5 | 31,5 | 128,0 |
| Núm. Salidas | 11,2 | 11,0 | 1,0 | 6,0 | 7,5 | 18,2 | 56,0 |
| Núm. Canales | 5,3 | 1,9 | 2,0 | 4,0 | 5,0 | 6,8 | 9,0 |
| Ratio ind. MSE | 0,5 | 0,2 | 0,0 | 0,3 | 0,5 | 0,6 | 0,9 |
| Prof. inic. (Kpal.) | 4,0 | 11,7 | 0,0 | 0,3 | 1,1 | 3,0 | 60,5 |
| Tam. inic. (Kbits) | 176,7 | 742,6 | 0,1 | 2,6 | 13,8 | 55,8 | 3811,5 |
| Red. inic. ( %) | 64,7 | 35,3 | 0,0 | 50,0 | 75,0 | 92,2 | 100,0 |

**Tabla 5.2:** Resumen estadístico de las características de la batería de pruebas BP-24.

| | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| Núm. Entradas | 5,5 | 3,3 | 1,0 | 3,0 | 5,0 | 8,0 | 16,0 |
| Núm. Estados | 24,2 | 17,7 | 5,0 | 8,0 | 18,0 | 38,0 | 62,0 |
| Núm. Salidas | 5,8 | 3,4 | 1,0 | 3,0 | 5,0 | 8,0 | 16,0 |
| Núm. Canales | 4,2 | 2,6 | 1,0 | 2,0 | 4,0 | 5,0 | 12,0 |
| Ratio ind. MSE | 0,3 | 0,2 | 0,0 | 0,1 | 0,3 | 0,5 | 0,9 |
| Prof. inic. (Kpal.) | 6,1 | 23,2 | 0,0 | 0,1 | 0,2 | 1,4 | 172,0 |
| Tam. inic. (Kbits) | 77,0 | 283,0 | 0,1 | 0,5 | 1,8 | 14,0 | 1892,0 |
| Red. inic. (%) | 29,2 | 39,1 | 0,0 | 0,0 | 0,0 | 50,0 | 99,9 |

wiak[7]. Esta batería de pruebas consta de 151 FSM generadas aleatoriamente, de las cuales se han seleccionado 108. El estudio experimental publicado en [García-Vargas y Senhadji-Navarro, 2015] se realizó empleando estas FSM junto con las de la batería de pruebas MCNC. La tabla 5.2 muestra un resumen estadístico de las características de las FSM seleccionadas (la tabla completa puede consultarse en el apéndice C).

En las últimas décadas, la escala de integración ha permitido un rápido aumento de la complejidad de los sistemas que se implementan en un único chip. Como consecuencia, las herramientas de síntesis deben ser capaces de manejar diseños que incluyen máquinas de estados de gran tamaño (hasta 500 estados y más de 1000 transiciones) [Duff y Saucier, 1991; Shihming Liu, 1995; Desai et al., 2003]. Sin embargo, como puede observarse en las tablas 5.1 y 5.2, las baterías de pruebas MCNC y BP-24 están compuestas principalmente de FSM pequeñas. El número medio de estados de la batería BP-24 es 24 aprox.; aunque este valor es algo mayor para la batería de pruebas MCNC (30 estados aprox.), el 92 % de sus FSM tienen menos de 50 estados.

Por otra parte, el tamaño inicial de las FSMIM-S es pequeño comparado con el tamaño mínimo de los bloques de memoria del dispositivo FPGA utilizado en los experimentos (9 Kbits). Antes de la fase de agrupación de estados, el 42 % de las FSM de la batería de pruebas MCNC pueden implementarse con la arquitectura FSMIM-S utilizando un sólo bloque de 9 Kbits. Este porcentaje aumenta al 67 % en la batería de pruebas BP-24. Esto significa que no es posible reducir más el número de bloques. Por tanto, la fase de agrupación no puede aportar ningún beneficio adicional en estos casos. Aunque el tamaño inicial medio mostrado en las tablas es elevado,

[7]Profesor del Departamento de Información y Sistemas de Comunicación de la Facultad de Ingeniería Eléctrica de la Universidad de Tecnología de Eindhoven (Países Bajos)

esto se debe a la existencia de casos extremos. Si eliminamos la mayor FSM de la batería de pruebas MCNC (la FSM *scf*), el tamaño inicial medio se reduce a 31 Kbits (equivalentes a 3 bloques y medio de 9 Kbits aprox.). En la batería de pruebas BP-24, si eliminamos los 11 casos que requieren más de 10 bloques, el tamaño inicial medio se reduce a 8,6 Kbits (menos de 1 bloque).

Con el objetivo de incluir máquinas de estados de mayor tamaño en los experimentos, se ha desarrollado una herramienta (denominada *RandomFSMIM*) para generar FSM aleatorias. Además de las características de las FSM, *RandomFSMIM* permite ajustar parámetros de las FSMIM que son independientes del proceso de optimización como el número de canales de selección o el ratio de indeterminaciones de la MSE (véase el apéndice D). Con esta herramienta se ha obtenido la tercera batería de pruebas usada en los experimentos (denominada *BP-184*), que consiste en 108 FSM aleatorias. La tabla de características de las FSM y los parámetros utilizados para generarlas pueden consultarse en el apéndice E. La tabla 5.3 muestra un resumen estadístico de sus características. El número de estados de las FSM varía entre 35 y 497, con 184 estados de media aproximadamente. Por lo tanto, esta batería de pruebas complementa a las dos anteriores al aportar un número significativo de FSM grandes. El conjunto completo de FSM utilizadas está compuesto por un total de 242 FSM (26 de la batería de pruebas MCNC, 108 de BP-24 y 108 de BP-184), de las cuales el 25 % presenta más de 110 estados (tal como se muestra en la tabla 5.4).

### 5.2.4. Indicadores y gráficas utilizadas.

En este apartado se definen los diferentes indicadores utilizados para el análisis comparativo de los resultados experimentales, así como algunas de las gráficas empleadas para representarlos.

**Tabla 5.3:** Resumen estadístico de las características de la batería de pruebas BP-184.

| | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| Núm. Entradas | 11,0 | 2,6 | 7,0 | 9,0 | 11,0 | 13,0 | 15,0 |
| Núm. Estados | 184,2 | 142,5 | 35,0 | 67,5 | 142,5 | 253,0 | 497,0 |
| Núm. Salidas | 4,6 | 1,4 | 2,0 | 4,0 | 4,5 | 6,0 | 7,0 |
| Núm. Canales | 8,0 | 0,8 | 7,0 | 7,0 | 8,0 | 9,0 | 9,0 |
| Ratio ind. MSE | 0,5 | 0,2 | 0,2 | 0,2 | 0,5 | 0,7 | 0,8 |
| Prof. inic. (Kpal.) | 53,8 | 55,7 | 4,9 | 17,5 | 32,3 | 64,6 | 248,5 |
| Tam. inic. (Kbits) | 650,7 | 676,7 | 53,6 | 223,0 | 391,5 | 769,0 | 3230,5 |
| Red. inic. (%) | 62,0 | 44,3 | 0,0 | 0,0 | 87,5 | 98,4 | 98,4 |

**Tabla 5.4:** Resumen estadístico de las características del conjunto completo de FSM.

| | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| Núm. Entradas | 8,3 | 4,2 | 1,0 | 5,0 | 8,0 | 11,0 | 27,0 |
| Núm. Estados | 96,3 | 124,6 | 4,0 | 16,0 | 45,0 | 110,0 | 497,0 |
| Núm. Salidas | 5,8 | 4,7 | 1,0 | 3,2 | 5,0 | 7,0 | 56,0 |
| Núm. Canales | 6,0 | 2,6 | 1,0 | 4,0 | 7,0 | 8,0 | 12,0 |
| Ratio ind. MSE | 0,4 | 0,2 | 0,0 | 0,2 | 0,5 | 0,6 | 0,9 |
| Prof. inic. (Kpal.) | 27,2 | 47,0 | 0,0 | 0,2 | 5,6 | 31,4 | 248,5 |
| Tam. inic. (Kbits) | 126,3 | 275,6 | 0,0 | 1,3 | 37,2 | 146,6 | 3388,0 |
| Red. inic. (%) | 47,7 | 44,2 | 0,0 | 0,0 | 50,0 | 87,5 | 100,0 |

#### 5.2.4.1. Indicadores

**Definición 5.1.** *Dadas dos medidas $p$ y $q$, se define la* **Reducción Relativa al Peor Caso** *(Reducción RPC) de $p$ respecto a $q$ como*

$$\frac{q-p}{\max\{|p|,|q|\}}.$$

*De forma similar, se define el* **Incremento Relativo al Peor Caso** *(Incremento RPC) de $p$ respecto a $q$ como*

$$\frac{p-q}{\min\{|p|,|q|\}}.$$

**Definición 5.2.** *Sean $A$ y $B$ dos entidades que se desean comparar (por ejemplo, dos arquitecturas, dos estrategias de optimización de FSMIM, etc.). Sea $r$ la reducción o el incremento RPC de una medida de $A$ respecto a $B$ tal que los valores de $r > 0$ indican una mejora de $A$ sobre $B$. Dado un conjunto de valores de $r$ obtenido a partir de una batería de pruebas, se define el* **porcentaje de éxito** *de $A$ como el porcentaje de casos de la batería de pruebas en los que $r > 0$; es decir, el porcentaje de casos en los que $A$ mejora a $B$ (en estos casos se dirá que $A$ es* **mejor opción** *que $B$). Se define el* **porcentaje de fracaso** *de $A$ como el porcentaje de casos en los que $r < 0$; es decir, el porcentaje de casos en los que $B$ mejora a $A$. Por lo tanto, el porcentaje de fracaso de $A$ es igual al porcentaje de éxito de $B$. Por otra parte, la suma del porcentaje de éxito $A$ más el porcentaje de fracaso $A$ es menor o igual que 100 %.*

**Definición 5.3.** *Sea $r_0$ una reducción o incremento RPC de una medida de $A$ respecto a $B$ tal que los valores $r_0 > 0$ indican una mejora de $A$ sobre $B$. De manera similar, sea $r_1$ una reducción o incremento RPC de una medida*

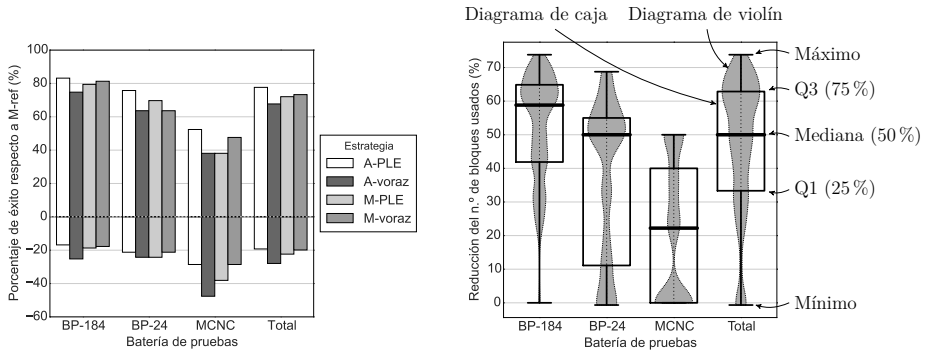
*de A respecto a B tal que los valores* $r_1 > 0$ *indican una mejora de A sobre B. Dado un conjunto de datos compuesto por los valores de* $r_0$ *y* $r_1$ *obtenidos a partir de una batería de pruebas, se define el* **porcentaje de éxito neto** *de A como el porcentaje de casos en los que* $r_0 \geq 0 \ \wedge \ r_1 \geq 0$ *menos el porcentaje de casos en los que* $r_0 = 0 \ \wedge \ r_1 = 0$*. Es decir, el porcentaje de casos en los que A mejora a B en al menos una de las medidas sin empeorar en la otra (en estos casos se dirá que A es* **mejor opción** *que B). Se define el* **porcentaje de fracaso neto** *de A como el porcentaje de casos en los que* $r_0 \leq 0 \ \wedge \ r_1 \leq 0$ *menos el porcentaje de casos en los que* $r_0 = 0 \ \wedge \ r_1 = 0$*. Es decir, el porcentaje de casos en los que B mejora a A en al menos una de las medidas sin empeorar en la otra. Por lo tanto, el porcentaje de fracaso neto de A es igual al porcentaje de éxito neto de B. La suma del porcentaje de éxito neto de A más el porcentaje de fracaso neto A es menor o igual que 100 %.*

Aunque el porcentaje de éxito neto y el porcentaje de fracaso neto se ha definido para dos medidas, dicha definición puede extenderse fácilmente a un número mayor de medidas.

#### 5.2.4.2. Gráficas

Los porcentajes de éxito se han representado con un diagrama de barras en el que cada barra marca dos valores: un valor positivo, que corresponde a un porcentaje de éxito, y un valor negativo, que corresponde a un porcentaje de fracaso. Por ejemplo, en el diagrama de la figura 5.1a, la barra más a la izquierda indica que la estrategia A-PLE obtiene un porcentaje de éxito del 80 % y un porcentaje de fracaso del 20 %, aproximadamente. Este mismo tipo de diagramas se ha utilizado para mostrar los porcentajes de éxito neto.

Por otra parte, para representar la distribución de los valores obtenidos con una medida o con un índice (incremento RPC o reducción RPC), se ha utilizado un diagrama de cajas superpuesto a un diagrama de violín (véase la figura 5.1b). El diagrama de cajas [Beyer, 1981] es un gráfico compuesto por un conjunto de rectángulos (o *cajas*) que proporciona información sobre los cuartiles y los valores máximo y mínimo de la muestra. Cada diagrama de cajas se ha superpuesto a un diagrama de violín [Hintze y Nelson, 1998], formado por curvas de densidad en espejo en las que el ancho es proporcional al número de valores pertenecientes a cada región.




**(a)** **(b)**

**Figura 5.1:** Ejemplos de gráficas utilizadas: (a) diagrama de porcentajes de éxito (b) diagrama para mostrar la distribución de valores.

## 5.3. Comparativa entre las arquitecturas FSMIM-T y FSMIM-S

En esta sección se presenta un estudio comparativo entre implementaciones de las arquitecturas FSMIM-S y FSMIM-T. Las FSMIM utilizadas en los experimentos no han sido generadas aplicando las nuevas estrategias de optimización propuestas en esta tesis sino la estrategia de referencia, implementada en la herramienta FSMIM-Gen 1.2. De esta forma, se consigue evaluar únicamente la influencia de la nueva arquitectura en las implementaciones de FSMIM, evitando que el resto de aportaciones de esta tesis afecte a los resultados.

El resto de la sección se divide en tres apartados. El primero de ellos compara las arquitecturas FSMIM-S y FSMIM-T a partir de los resultados de las pruebas independientes del dispositivo. El segundo apartado presenta una comparativa entre implementaciones en FPGA de FSMIM con y sin ajuste de profundidad. Los resultados de dicha comparación permitirán justificar el uso del ajuste de profundidad en el resto de las pruebas de implementación realizadas en el marco de esta tesis. Finalmente, en el tercer apartado se comparan las arquitecturas FSMIM-S y FSMIM-T utilizando los resultados de las pruebas de implementación realizadas con las FSMIM con ajuste de profundidad.

### 5.3.1. Pruebas independientes del dispositivo

El propósito de este estudio es cuantificar la reducción RPC del tamaño genérico de la ROM que se puede conseguir al utilizar la arquitectura FSMIM-S respecto a la arquitectura FSMIM-T. Esta medida será denominada *reducción genérica* del tamaño de la ROM. En las implementaciones de FSMIM en dispositivos FPGA, el grado de aproximación a la reducción genérica depende del tamaño de la ROM de la FSMIM y de las características de los bloques de memoria del dispositivo (es decir de las diferentes geometrías disponibles en los bloques).

Las FSMIM se han generado sin ajuste de profundidad usando la herramienta FSMIM-Gen 1.2. Posteriormente, para cada FSMIM se ha calculado el tamaño de la ROM requerido por las arquitecturas FSMIM-S y FSMIM-T. La tabla 5.5 muestra un resumen estadístico de los valores obtenidos. Atendiendo al tamaño medio de las FSMIM resultantes, puede establecerse la siguiente ordenación de las baterías de pruebas de mayor a menor tamaño: BP-184, BP-24 y MCNC. Este orden se mantiene para las dos arquitecturas.

Para facilitar la comparación, se ha calculado la reducción RPC del tamaño de la ROM de las implementaciones FSMIM-S con respecto a FSMIM-T (la tabla 5.6 y la figura 5.2 muestran un resumen estadístico). La reducción media para la muestra completa es del 43 %, llegando a alcanzarse reducciones entre el 59 % y el 73 % en el 25 % de los casos. La reducción es del 0 % sólo para 7 casos (pertenecientes a la batería de pruebas BP-24). En estos casos, la estrategia de optimización utilizada no se ha podido aplicar con éxito a pesar de que las FSM cumplen los requisitos mínimos para generar una FSMIM.

Respecto a la contribución de cada batería de pruebas a los resultados totales, claramente, los mejores resultados se obtienen para la batería de pruebas BP-184, con una reducción media del 59 %. Todos los casos excepto uno presentan una reducción superior al 43 %. Por otra parte, tanto la batería de pruebas MCNC como BP-24 presentan una reducción media cercana al 30 %. Además, como se puede apreciar, los valores de los cuartiles de ambas baterías de pruebas son similares.

### 5.3.2. Estudio de la influencia del ajuste de la profundidad sobre las implementaciones de FSMIM

Con objeto de realizar un estudio preliminar de la arquitectura FSMIM-S, se implementaron en el dispositivo FPGA XC6SLX75-3 (Spartan-6) las FSMIM generadas a partir de las baterías de pruebas MCNC y BP-24 en

**Tabla 5.5:** Resumen estadístico del tamaño de la ROM (Kbits) obtenida por las arquitecturas FSMIM-S y FSMIM-T.

| Bat. de pruebas | Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-S | 133,7 | 166,6 | 10,5 | 39,0 | 73,3 | 162,8 | 1079,0 |
| | FSMIM-T | 375,4 | 507,8 | 17,5 | 85,8 | 186,0 | 400,0 | 3320,0 |
| BP-24 | FSMIM-S | 64,3 | 251,7 | 0,0 | 0,3 | 1,2 | 10,8 | 1848,0 |
| | FSMIM-T | 90,0 | 332,3 | 0,1 | 0,5 | 1,9 | 17,5 | 2184,0 |
| MCNC | FSMIM-S | 8,9 | 13,3 | 0,1 | 1,2 | 5,0 | 11,0 | 63,0 |
| | FSMIM-T | 13,1 | 17,9 | 0,1 | 1,7 | 7,2 | 16,2 | 78,0 |
| Total | FSMIM-S | 89,3 | 205,8 | 0,0 | 1,2 | 18,3 | 78,0 | 1848,0 |
| | FSMIM-T | 209,1 | 431,8 | 0,1 | 1,9 | 35,5 | 198,1 | 3320,0 |

las pruebas independientes del dispositivo descritas en la sección 5.3.1. Los resultados obtenidos se utilizaron, por una parte, para comparar las dos arquitecturas de FSMIM entre sí y, por otra parte, para comparar las FSMIM con las implementaciones de FSM tradicionales. El estudio ha sido publicado en [García-Vargas y Senhadji-Navarro, 2015]. A diferencia del resto de pruebas de implementación presentadas en este capítulo, que utilizan FSMIM generadas con ajuste de profundidad, las FSMIM utilizadas en dicho estudio se generaron sin ajuste de profundidad. Las pruebas realizadas con estas FSMIM serán denominadas *pruebas sin ajuste de profundidad*. Cuando sea necesario para evitar la ambigüedad, el resto de pruebas de implementación serán denominadas *pruebas con ajuste de profundidad*.

Las pruebas sin ajuste de profundidad han sido utilizadas en este apartado principalmente para evaluar la influencia del ajuste de la profundidad sobre las implementaciones de FSMIM. Con este fin, los resultados de las pruebas sin ajuste de profundidad presentadas en [García-Vargas y Senhadji-Navarro, 2015] han sido comparados con los resultados obtenidos en las pruebas con ajuste de profundidad. El estudio se ha realizado solamente con las baterías MCNC y BP-24 debido a que estas fueron las baterías usadas en las pruebas sin ajuste de profundidad. Para el análisis de los resultados se han seleccionado los casos que cumplen las siguientes condiciones:

1. **La implementación de la FSM-ROM debe ocupar más de un bloque de 9 Kbits.** Los bloques de memoria de 9 Kbits son los más pequeños de las Spartan-6; por lo tanto, esta condición es necesaria para que las implementaciones FSMIM puedan mejorar las prestaciones de las implementaciones FSM-ROM.

**Tabla 5.6:** Resumen estadístico de la reducción RPC del tamaño de la ROM de FSMIM-S respecto a FSMIM-T (%).

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 59,3 | 8,5 | 35,0 | 52,0 | 60,3 | 66,7 | 73,8 |
| BP-24 | 30,1 | 16,0 | 0,0 | 20,0 | 30,0 | 38,9 | 71,0 |
| MCNC | 31,5 | 8,7 | 16,7 | 22,3 | 31,4 | 37,3 | 47,1 |
| Total | 43,3 | 19,0 | 0,0 | 29,2 | 45,6 | 59,2 | 73,8 |

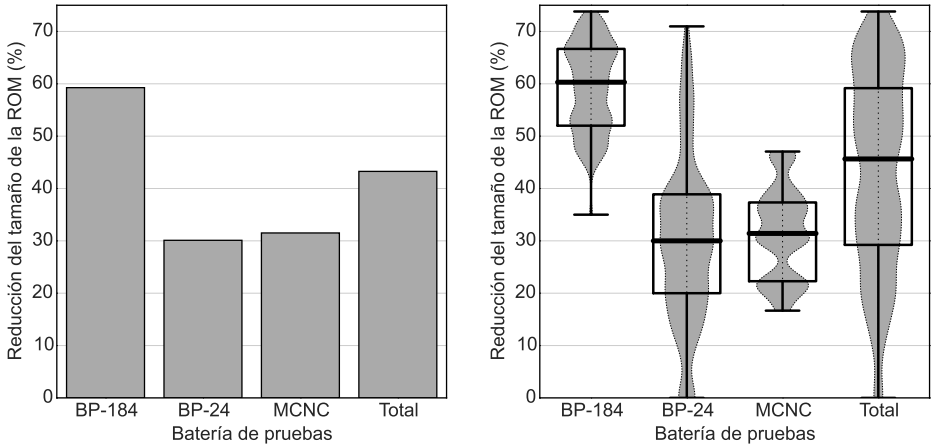



**Figura 5.2:** Reducción RPC del tamaño de la ROM de FSMIM-S respecto a FSMIM-T: (a) reducciones medias, (b) distribución de los valores obtenidos.

2. **La generación de la FSMIM con ajuste profundidad debe tener éxito.** Para que sea posible obtener una FSMIM deben cumplirse los siguientes requisitos: (a) que el número de canales de selección de la FSMIM sea menor que el número de entradas de la FSM, o (b) que la fase de agrupación de estados consiga agrupar dos estados como mínimo. En aquellas FSM en las que el requisito (a) no se cumple, es la fase de agrupación de estados la que determina si se puede obtener una FSMIM. En estos casos, a pesar de que la generación de una FSMIM sin ajuste de profundidad sea posible, no está garantizada con el ajuste de profundidad debido a que el número de estados que se agrupan puede disminuir para que la profundidad de la ROM sea múltiplo de la de los bloques.

Entre las dos baterías de pruebas utilizadas suman 56 casos que satisfacen las dos condiciones anteriores. La tabla 5.7 muestra un resumen estadístico de los resultados de implementación obtenidos en las pruebas sin ajuste de profundidad usando las arquitecturas FSMIM-T y FSMIM-S; la tabla 5.8 muestra los resultados de las pruebas con ajuste de profundidad. Los datos de ambas tablas corresponden a los experimentos realizados con *ISE WebPACK* configurado para optimizar el área. La fila "Bloq." contiene el número de bloques de 18 Kbits usados (los bloques residuales de 9 Kbits cuentan como 0,5), la fila "Frec." contiene la frecuencia de operación máxima en MHz, y la fila "LUT" contiene el número de LUT ocupadas.

Para cada tipo de prueba se ha utilizado un dispositivo Spartan-6 con diferente *speed grade*. Sin embargo, las características de las LUT y de los bloques de memoria empotrados de ambos dispositivos, así como el número de recursos disponibles, son iguales. Por lo tanto, los resultados pueden ser utilizados para realizar un estudio comparativo de consumo de LUT y bloques.

**Tabla 5.7:** Resumen estadístico de los resultados de implementación obtenidos en las pruebas sin ajuste de profundidad (dispositivo XC6SLX75-3).

| Bat. de pruebas | Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| Bloq. | FSMIM-S | 4,6 | 14,4 | 0,5 | 0,5 | 0,5 | 1,6 | 79,5 |
| | FSMIM-T | 7,0 | 19,6 | 0,5 | 0,5 | 1,0 | 2,6 | 97,0 |
| Frec. | FSMIM-S | 204,0 | 50,1 | 82,0 | 172,8 | 210,0 | 245,5 | 275,0 |
| | FSMIM-T | 233,2 | 50,3 | 84,0 | 227,2 | 246,0 | 267,2 | 289,0 |
| LUT | FSMIM-S | 22,6 | 22,0 | 4,0 | 6,0 | 15,5 | 29,0 | 86,0 |
| | FSMIM-T | 7,6 | 9,0 | 2,0 | 4,0 | 4,5 | 7,0 | 49,0 |

**Tabla 5.8:** Resumen estadístico de los resultados de implementación obtenidos en las pruebas con ajuste de profundidad (dispositivo XC6SLX75T-4).

| Bat. de pruebas | Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| Bloq. | FSMIM-S | 4,6 | 14,3 | 0,5 | 0,5 | 0,5 | 1,6 | 75,5 |
| | FSMIM-T | 6,8 | 19,5 | 0,5 | 0,5 | 1,0 | 2,5 | 100,0 |
| Frec. | FSMIM-S | 207,4 | 53,0 | 79,2 | 185,3 | 212,6 | 253,8 | 278,9 |
| | FSMIM-T | 243,1 | 54,5 | 83,1 | 231,3 | 260,2 | 275,0 | 302,9 |
| LUT | FSMIM-S | 22,1 | 25,4 | 3,0 | 5,8 | 12,5 | 30,2 | 136,0 |
| | FSMIM-T | 6,6 | 9,1 | 1,0 | 2,8 | 4,0 | 6,2 | 53,0 |

### Estudio del consumo de bloques

La tabla 5.9 muestra la reducción RPC del consumo de bloques obtenida por las FSMIM con ajuste de profundidad respecto a las FSMIM sin ajuste de profundidad. Como puede observarse, la reducción media en el consumo de bloques obtenida para las FSMIM-S es prácticamente cero (0.02 %). Tal como se explicó en la sección 5.1.2, el consumo de bloques en la FSMIM-S debería ser independiente del ajuste de profundidad; por lo tanto, la reducción debería ser cero en todos los casos. Sin embargo, en los experimentos realizados con las FSMIM-S, hay dos casos en los que el resultado se desvía del valor teórico. Se trata de los valores mínimo y máximo que aparecen en la tabla 5.9. El valor mínimo corresponde a una reducción del 4 % obtenida sin el ajuste de profundidad, mientras que el valor máximo corresponde a una reducción del 5 % obtenida con el ajuste de profundidad. Esta aparente incongruencia en los resultados experimentales se debe a la minimización llevada a cabo por *ISE WebPACK* durante la fase de síntesis de bajo nivel. La herramienta elimina los bloques de memoria empotrados que contienen un único valor (es decir, sólo ceros o sólo unos) y los sustituye por un valor constante.

Además, es capaz de detectar conjuntos de bloques con el mismo contenido y reducir cada conjunto a un único bloque siempre se sea posible (eliminando así los bloques redundantes).

Con objeto de comprobar la corrección de los datos, se ha calculado, para cada FSMIM y para cada arquitectura, el número de bloques que ocupa la descripción HDL de la ROM antes de la minimización realizada durante la síntesis (se ha denominado a este valor *número de bloques estimado* de la ROM). En el caso de las FSMIM-S, el número de bloques estimado es el mismo en los dos tipos de pruebas realizadas (por lo tanto la reducción antes de la síntesis de bajo nivel es del 0 % en todos los casos).

A diferencia de las FSMIM-S, el número de bloques usados por las FSMIM-T sí puede verse afectado por el ajuste de la profundidad (véase la sección 5.1.2). Tal como puede observarse en la tabla 5.9, la reducción

**Tabla 5.9:** Resumen estadístico de la reducción RPC del número de bloques usados obtenida por las FSMIM con ajuste de profundidad respecto a las FSMIM sin ajuste de profundidad ( %).

| Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| FSMIM-S | 0,0 | 0,9 | −4,0 | 0,0 | 0,0 | 0,0 | 5,0 |
| FSMIM-T | 4,2 | 11,3 | −3,0 | 0,0 | 0,0 | 0,0 | 50,0 |

media obtenida con el ajuste de profundidad en la arquitectura FSMIM-T es del 4,2 % (esta media sube ligeramente al 4,6 % si se realiza el cálculo con el número de bloques estimado). El porcentaje de éxito es del 21 % (12 casos), con una reducción media del 20 %. Por otra parte, el porcentaje de éxito obtenido sin el ajuste de profundidad es tan sólo del 4 % (2 casos), con una reducción media del 3 %. En estos dos casos, la mejora no se debe al ajuste de la profundidad, sino a la minimización realizada por la síntesis de bajo nivel.

**Estudio de consumo de LUT**

Respecto al consumo de LUT, de nuevo, el ajuste de la profundidad afecta de forma diferente a las dos arquitecturas de FSMIM (véase la sección 5.1.2). En la arquitectura FSMIM-T, el aumento de grupos que se produce al ajustar la profundidad favorece la simplificación del banco de selectores de entrada debido a que los valores constantes seleccionados por algunos de los estados desaparecen y dan lugar a indeterminaciones. Sin embargo, en la arquitectura FSMIM-S, además de este efecto positivo en el banco de selectores de entrada, aparece otro de signo contrario. El aumento del número de grupos puede incrementar el número de bits de codificación de grupos y, por lo tanto, el número de salidas del codificador de grupos, aunque la complejidad de las funciones lógicas asociadas a cada salida puede disminuir (por ejemplo, en el caso extremo de que el número de grupos sea igual al número de estados, es decir, de que cada grupo esté formado por un único estado, el codificador de grupos desaparece).

Resulta difícil estimar la influencia que tendrá el ajuste de profundidad en la implementación final de las FSMIM-S. El consumo de LUT depende de parámetros como la codificación asignada a los estados de la FSM y a los grupos de la FSMIM. Estos parámetros tienen un peso importante en el resultado del proceso de síntesis. Sin embargo, puesto que la influencia de la codificación de estados y grupos no ha sido objeto de estudio en esta tesis, se ha utilizado una codificación arbitraria en las pruebas de implementación bajo la premisa de que, en términos generales, dicha codificación afectará de forma similar a todas las pruebas realizadas.

Los resultados experimentales muestran una reducción media del 9,5 % con el ajuste de profundidad (véase la tabla 5.10). A pesar de que este valor es modesto, el número de casos de éxito obtenido con el ajuste de profundidad es mucho mayor que el obtenido sin el ajuste de profundidad, tal como muestra la figura 5.3b. El porcentaje de éxito conseguido con ajuste de profundidad es del 60 % mientras que sin ajuste de profundidad es del

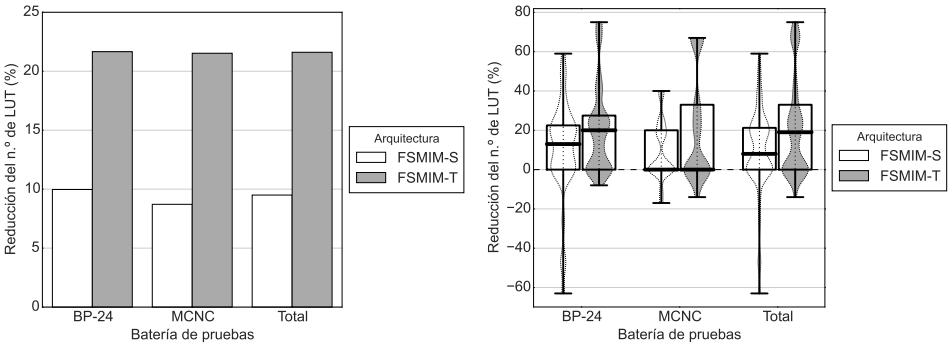
14 %. Como puede observarse en la figura 5.3, las dos baterías de pruebas utilizadas obtienen unas reducciones medias similares (figura 5.3a). Además, aunque el intervalo de valores es mayor en la batería BP-24, las distribuciones de ambas baterías presentan ciertas similitudes (véase el valor de los cuartiles Q1 y Q2 en la figura 5.3b).

El estudio de la influencia del ajuste de la profundidad en la arquitectura FSMIM-T es más simple que en la arquitectura FSMIM-S debido, principalmente, a que el consumo de LUT no depende de la codificación de estados y grupos: en la arquitectura FSMIM-T el banco de multiplexores no está controlado por los bits de codificación de estado y no existe el codificador de grupo. Incrementar el número de grupos puede reducir la complejidad del banco de multiplexores, pero nunca la aumentará; por lo tanto, desde un punto de vista teórico, no puede empeorar el consumo de LUT. Sin embargo, algunos resultados experimentales parecen contradecir esta afirmación (véase la reducción negativa que aparece en la tabla 5.10 y en la figura 5.3b).

En dos casos (correspondientes a las FSM *s510* y *fsm_rd_074*), el au-

**Tabla 5.10:** Resumen estadístico de la reducción RPC del número de LUT obtenido por las FSMIM con ajuste profundidad respecto a las FSMIM sin ajuste de profundidad ( %).

| Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| FSMIM-S | 9,5 | 22,3 | −63,0 | 0,0 | 8,0 | 21,2 | 59,0 |
| FSMIM-T | 21,6 | 25,2 | −14,0 | 0,0 | 19,0 | 33,0 | 75,0 |




**(a)** **(b)**

**Figura 5.3:** Resumen estadístico de la reducción RPC del número de LUT obtenido con ajuste de profundidad respecto al obtenido sin ajuste de profundidad: (a) reducciones medias, (b) distribución de los valores obtenidos.

mento del número de grupos conseguido por la profundidad ajustada a bloques respecto a la profundidad mínima, a pesar de que da lugar a multiplexores más simples, provoca un incremento en el consumo de LUT. Estos resultados son debidos a los algoritmos de síntesis usados por *ISE WebPACK*. Por ejemplo, en la FSM *s510* (de MCNC), con el ajuste de profundidad se consigue reducir el número de entradas de uno de los multiplexores de 15 a 14 entradas. Sin embargo, inesperadamente, la herramienta usa una LUT más para implementar el multiplexor con 14 entradas.

Para el resto de casos, los resultados confirman el comportamiento esperado respecto al consumo de LUT (véase la tabla 5.10). La reducción media obtenida con el ajuste de profundidad es del 21,6 %, con un porcentaje de éxito superior al 50 % y una reducción mayor o igual al 19 % para la mitad de los casos. Al igual que ocurría con la arquitectura FSMIM-S, los resultados obtenidos en las dos baterías de pruebas presentan una reducción media similar (figura 5.3a) y las distribuciones de sus valores no muestran grandes diferencias (figura 5.3b).

Las mejoras obtenidas en el consumo de recursos al implementar las FSMIM con ajuste de profundidad, especialmente en el caso de las FSMIM-T, determinaron la decisión de incorporarlo en los estudios presentados en esta tesis.

### 5.3.3. Pruebas de implementación

Las pruebas independientes del dispositivo han demostrado que la arquitectura FSMIM-S consigue importantes reducciones en el tamaño de la ROM comparada con la arquitectura FSMIM-T. Sin embargo, en implementaciones en FPGA, esta reducción se obtiene a expensas de un incremento en el número de recursos lógicos que se requieren para implementar las FSMIM-S. Este incremento se debe al aumento de la complejidad del banco de selectores de entradas y al empleo del codificador de grupos. Por otra parte, la reducción obtenida puede verse afectada por las características de los bloques de memoria empotrados usados para implementar la ROM.

Las pruebas de implementación tienen un doble objetivo en el estudio de la arquitectura FSMIM-S: (a) cuantificar el grado de aproximación a la reducción genérica del tamaño de la ROM que se obtiene al utilizar los bloques de memoria empotrados de los dispositivos FPGA, y (b) evaluar la influencia del codificador de grupos y del aumento de complejidad de los selectores de entradas sobre el consumo de LUT y la velocidad de las implementaciones de FSMIM.

El estudio comparativo de las arquitecturas FSMIM-S y FSMIM-T se

ha realizado utilizando el conjunto completo de FSM. Cuando se genera una FSMIM con ajuste de profundidad, dicho ajuste depende del tamaño de la ROM; por lo tanto, también depende de la arquitectura de FSMIM que se va a utilizar. En las pruebas realizadas, a partir de cada FSM, se ha generado una FSMIM con ajuste de profundidad diferente para cada arquitectura. Posteriormente, cada FSMIM se ha implementado dos veces en el dispositivo FPGA, configurando *ISE WebPACK* con un objetivo de optimización diferente en cada implementación (optimización en área o en velocidad). Para el análisis de los resultados se han seleccionado las FSM que cumplen las siguientes condiciones:

1. **La implementación de la FSM-ROM debe ocupar más de un bloque de 9 Kbits** (véase la sección 5.3.2, página 142).

2. **La generación de la FSMIM con ajuste de profundidad debe tener éxito** (véase la sección 5.3.2, página 142).

3. **La implementación de la FSMIM-T debe ocupar más de un bloque de 9 Kbits.** Esta condición es necesaria para que la implementación FSMIM-S pueda reducir el consumo de bloques de la implementación FSMIM-T.

De todas las FSM que cumplen las condiciones anteriores, una de ellas (*fsm_e9i12o4s497*, perteneciente a BP-184) ha sido excluida del estudio debido a que la implementación FSMIM-T requiere más de 172 bloques de memoria (el máximo disponible en el dispositivo). La implementación FSMIM-S, por el contrario, ocupa menos de la mitad (68,5 bloques). El número de casos final analizados es 141 (107 casos de BP-184, 21 de BP-24 y 13 de MCNC).

El estudio se ha dividido en dos partes: la primera está dedicada al análisis de los resultados de las implementaciones optimizadas en área, y la segunda, a las optimizadas en velocidad. En ambas partes, se analizarán tanto el área ocupada (en términos de número de LUT y de bloques de memoria) como la frecuencia de operación máxima. La arquitectura FSMIM-T se ha tomado como referencia para todos los cálculos comparativos realizados.

#### 5.3.3.1. Análisis de las implementaciones optimizadas en área

**Estudio del consumo de bloques**

El siguiente estudio pretende evaluar la influencia del uso de bloques de memoria empotrados sobre la reducción del tamaño de la ROM obtenida con

la arquitectura FSMIM-S respecto a la arquitectura FSMIM-T. Se intentará estudiar, además, en qué grado se pueden usar los resultados de las pruebas independientes del dispositivo como estimación previa de la reducción que se obtendrá en dispositivos FPGA.

El efecto más directo del uso de bloques de memoria sobre los resultados de las pruebas es el aumento de la granularidad en la medida del tamaño de la ROM. Mientras que el tamaño de la ROM se mide en bits para el cálculo de la reducción genérica; en las pruebas de implementación realizadas, la medida tiene una resolución de 0,5 bloques de 18 Kbits (donde el valor 0,5 corresponde a un bloque de 9 Kbits). Como consecuencia, la reducción relativa que se puede obtener en las pruebas de implementación sufre una discretización que depende del número de bloques de la FSMIM-T.

Dada una FSMIM-T de $N$ bloques de 18 Kbits, tenemos $2N$ posibles valores para la reducción relativa, que vienen dados por la expresión $\frac{i}{N}$ para $i = 0, 1, \ldots, 2N - 1$. Por ejemplo, una FSMIM-T que ocupe un bloque solo podrá ser reducida el 0 % o el 50 %. La consecuencia más inmediata de esta discretización ha sido la exclusión de este estudio de los casos en los que la FSMIM-T ocupa 0,5 bloques. Se trata de 22 de las 164 FSM que cumplen las condiciones 1 y 2 para la selección de la muestra (página 149). Por otra parte, debido a que la discretización disminuye con el número de bloques, la diferencia entre la reducción genérica y la reducción obtenida en las pruebas de implementación decrece con el incremento del número de bloques de la FSMIM-T. Es decir, la aproximación de la reducción genérica a la reducción obtenida en las pruebas de implementación mejora con al aumentar el número de bloques usados.

La tabla 5.11 muestra un resumen estadístico del número de bloques usados por las implementaciones FSMIM-S y FSMIM-T. Como puede observarse, la ocupación de bloques para la FSMIM-S varía entre 0,5 bloques (es decir, un bloque de 9 Kbits) y 75,5 bloques; mientras que el número máximo de bloques para la FSMIM-T es 143 (sin tener en cuenta el caso que ha sido excluido por ocupar más de 172 bloques).

La tabla 5.12 y en la figura 5.4 muestran un resumen estadístico de la reducción RPC del número de bloques usados por implementaciones FSMIM-S respecto a implementaciones FSMIM-T. La reducción media es del 47,5 %, llegando a alcanzarse reducciones entre el 50 % y el 73,8 % para la mitad de los casos estudiados. El porcentaje de éxito ha sido del 92 %, con una reducción media del 51,5 %.

La FSMIM-S consigue reducir el tamaño de la FSMIM-T en todos los casos excepto en 11. En 9 de estos 11 casos, la FSMIM-T ocupa un único bloque de 18 Kbits y tiene una profundidad menor o igual a 4 Kpalabras, lo

**Tabla 5.11:** Resumen estadístico del número de bloques ocupados por las arquitecturas FSMIM-S y FSMIM-T en las pruebas de implementación con optimización en área.

| Bat. de pruebas | Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-S | 9,6 | 9,1 | 1,0 | 3,0 | 6,5 | 13,0 | 49,5 |
| | FSMIM-T | 24,9 | 27,7 | 1,0 | 6,2 | 15,0 | 34,5 | 143,0 |
| BP-24 | FSMIM-S | 11,0 | 22,2 | 0,5 | 0,5 | 2,0 | 4,5 | 75,5 |
| | FSMIM-T | 16,6 | 29,7 | 1,0 | 1,0 | 3,0 | 10,0 | 100,0 |
| MCNC | FSMIM-S | 1,3 | 0,9 | 0,5 | 1,0 | 1,0 | 1,0 | 3,5 |
| | FSMIM-T | 1,7 | 1,2 | 1,0 | 1,0 | 1,0 | 1,5 | 4,5 |
| Total | FSMIM-S | 9,0 | 11,8 | 0,5 | 2,0 | 5,5 | 12,0 | 75,5 |
| | FSMIM-T | 21,5 | 27,5 | 1,0 | 3,5 | 9,5 | 29,0 | 143,0 |

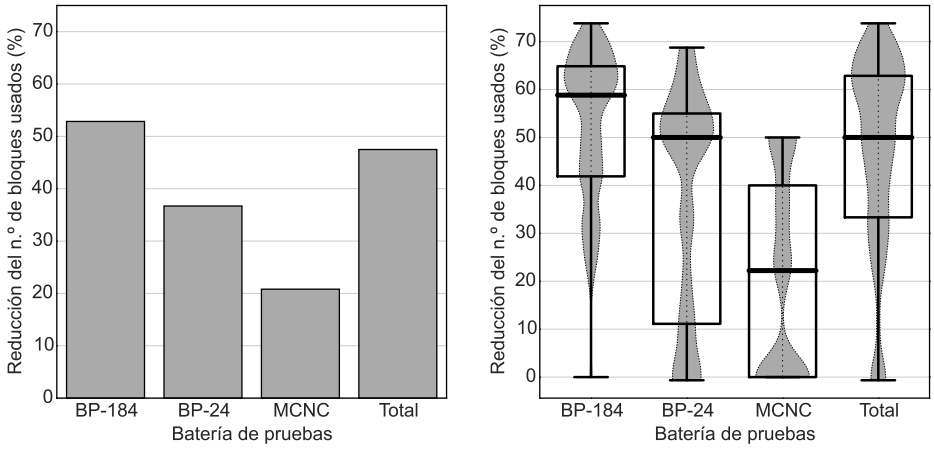
que significa que el ancho de los bloques de 9 Kbits que forman la ROM es mayor o igual que 2 (por ejemplo, una ROM de 2K×7 estará formada por dos bloques de 9 Kbits con configuración 2K×4). Este resultado muestra que otro factor que influye en las reducciones obtenidas en las pruebas de implementación es el conjunto de las posibles geometrías que pueden configurarse en cada bloque. Debido a que el ancho de los bloques aumenta cuando la profundidad decrece, la influencia es mayor en las FSMIM-T pequeñas, con una profundidad menor que la profundidad máxima de un bloque de 9 Kbits (8 Kpalabras). Puesto que la arquitectura FSMIM-S reduce el ancho de palabra de la ROM respecto a la arquitectura FSMIM-T, mientras mayor sea el ancho de los bloques, menor será la efectividad de la técnica (es decir, para eliminar un bloque será necesario reducir el ancho de palabra en un mayor número de bits).

Según la ecuación 2.25, dada una FSMIM cualquiera, el tamaño de la FSMIM-S siempre es menor o igual que el tamaño de la FSMIM-T. Sin embargo, para la FSM *fsm_rd_059* (perteneciente a la batería de pruebas BP-24), la FSMIM-S tiene un tamaño mayor que el de la FSMIM-T. Esta aparente contradicción de los resultados se debe, como en ocasiones anteriores, al proceso de síntesis de bajo nivel que realiza *ISE WebPACK*. Para esta FSM, el número de bloques estimado de la ROM de la FSMIM-S (116 bloques) es menor que el de la FSMIM-T (136 bloques); sin embargo, el algoritmo de síntesis utiliza un bloque de 9 Kbits menos para implementar la FSMIM-T (la implementación utiliza 75 bloques para la FSMIM-T y 75,5 para la FSMIM-S).

Con el fin de comparar los resultados de las pruebas de implementación con los obtenidos en las pruebas independientes, se ha calculado el valor

**Tabla 5.12:** Resumen estadístico de la reducción RPC del número de bloques usados por implementaciones FSMIM-S respecto a FSMIM-T en las pruebas con optimización en área (%).

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 52,9 | 15,4 | 0,0 | 41,9 | 58,8 | 64,9 | 73,8 |
| BP-24 | 36,7 | 24,2 | −0,7 | 11,1 | 50,0 | 55,0 | 68,8 |
| MCNC | 20,8 | 21,8 | 0,0 | 0,0 | 22,2 | 40,0 | 50,0 |
| Total | 47,5 | 20,2 | −0,7 | 33,3 | 50,0 | 62,9 | 73,8 |




**(a)** **(b)**

**Figura 5.4:** Resumen estadístico de la reducción RPC del número de bloques usados por implementaciones FSMIM-S respecto a FSMIM-T en las pruebas con optimización en área: (a) reducciones medias, (b) distribución de los valores obtenidos.

$R_{\text{imp}} - R_{\text{gen}}$, que será denominado *error genérico*, donde $R_{\text{imp}}$ es la reducción RPC del número de bloques usados por las implementaciones FSMIM-S respecto a FSMIM-T, y $R_{\text{gen}}$ es la reducción genérica (es decir, la reducción RPC del tamaño genérico de las FSMIM-S respecto a FSMIM-T).

La figura 5.5 muestra la relación entre el número de bloques de la FSMIM-T y el error genérico. La línea de regresión local muestra claramente cómo el error se reduce con el aumento del número de bloques. Puede observarse, además, que el error es positivo para algunos casos en los que el número de bloques es pequeño; es decir, que la reducción obtenida es mayor que la reducción genérica. El error genérico medio es −6,4; es decir, la

reducción media obtenida para el conjunto completo de FSM está sólo 6,4 puntos por debajo de la reducción genérica media.

Si se analizan los resultados de cada batería de pruebas por separado, puede observarse que la batería de pruebas MCNC es la que presenta mayores errores genéricos de la reducción. Su reducción media, igual al 20,8 %, es casi 11 puntos inferior a la reducción genérica media. Por otra parte, mientras que el porcentaje de éxito en las pruebas independientes del dispositivo era del 100 %, este porcentaje se reduce al 54 % en las pruebas de implementación. Estos resultados se explican por el reducido tamaño de las FSMIM-T obtenidas con esta batería de pruebas: la mayor FSMIM-T ocupa 4,5 bloques, siendo el número de bloques menor que 2 en el 75 % de los casos (véase la tabla 5.11)

Al igual que ocurría en las pruebas independientes del dispositivo, el mejor resultado se obtiene para la batería de pruebas BP-184, con una reducción media del 52,9 % y reducciones superiores al 58 % para la mitad de los casos. Tan sólo hay un caso en el que la FSMIM-S no consigue reducir el tamaño de la ROM de la FSMIM-T. La reducción media está 6,3 puntos por debajo de la reducción genérica media. Como cabía esperar, es la batería de pruebas que presenta menores errores genéricos de la reducción. Esto es debido a que las mayores FSMIM-T son las generadas con esta batería de pruebas.

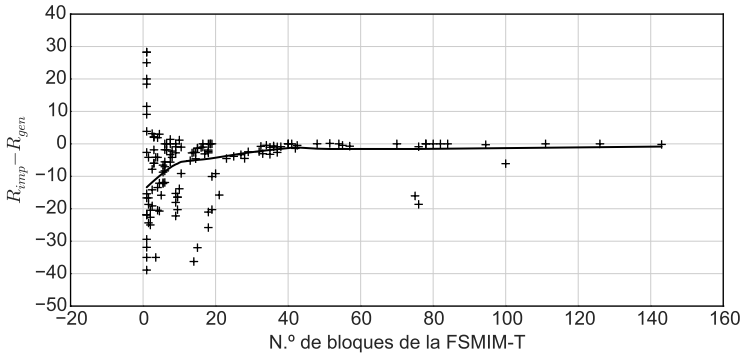



**Figura 5.5:** Relación entre el número de bloques de la FSMIM-T y el error genérico de la reducción del tamaño de la FSMIM-S respecto a la FSMIM-T: $R_{\text{imp}} - R_{\text{gen}}$, donde $R_{\text{imp}}$ es la reducción obtenida en las pruebas de implementación con optimización en área y $R_{\text{gen}}$ es la reducción genérica.

### Estudio del consumo de LUT

Este apartado tiene dos objetivos principales: (a) evaluar el incremento en el consumo de LUT de las implementaciones FSMIM-S respecto a las FSMIM-T, y (b) estudiar la eficiencia en el uso de bloques, medida como el número de LUT que consigue ahorrar cada arquitectura por cada bloque usado respecto a las implementaciones convencionales de FSM (es decir, implementaciones ISE-LUT). Será necesario, por lo tanto, analizar tanto el número de bloques y de LUT utilizados por las FSMIM como el número de LUT necesarios en las implementaciones ISE-LUT. La tabla 5.13 muestra un resumen estadístico del número de LUT usados por cada arquitectura. Como cabía esperar, la batería de pruebas MCNC es la que obtiene el menor número medio de LUT; y BP-184, la que obtiene el mayor. Aunque esta relación se mantiene en ambas arquitecturas, se puede observar cómo el número de LUT crece más rápidamente en la arquitectura FSMIM-S.

Para relacionar el consumo de LUT en ambas arquitecturas se ha calculado el incremento RPC del número de LUT de FSMIM-S respecto a FSMIM-T (véase la tabla 5.14). El incremento medio es 1 565 %; es decir, para conseguir una reducción media del número de bloques respecto a la FSMIM-T del 47 % aproximadamente, las FSMIM-S necesitan utilizar casi 16 veces más LUT que las FSMIM-T. Posiblemente, buscando una codificación de grupos y de estados adecuada, se podría reducir sensiblemente el número de LUT necesarias para implementar el codificador de grupos y el selector de entradas de la FSMIM-S. Este es un estudio interesante que no se ha afrontado en esta tesis y que formará parte de los trabajos futuros.

En cualquier caso, aunque el incremento de LUT puede parecer muy elevado en comparación con el ahorro de bloques, hay que considerar, por

**Tabla 5.13:** Resumen estadístico del número de LUT usadas por las arquitecturas FSMIM-S y FSMIM-T en las pruebas de implementación con optimización en área.

| Bat. de pruebas | Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-S | 265,7 | 245,0 | 16,0 | 74,0 | 184,0 | 392,0 | 1111,0 |
| | FSMIM-T | 16,2 | 14,6 | 2,0 | 6,0 | 12,0 | 19,0 | 70,0 |
| BP-24 | FSMIM-S | 37,0 | 33,1 | 4,0 | 11,0 | 32,0 | 50,0 | 136,0 |
| | FSMIM-T | 11,1 | 13,5 | 1,0 | 3,0 | 5,0 | 15,0 | 53,0 |
| MCNC | FSMIM-S | 18,5 | 17,7 | 3,0 | 11,0 | 15,0 | 18,0 | 75,0 |
| | FSMIM-T | 5,8 | 2,7 | 1,0 | 4,0 | 5,0 | 7,0 | 12,0 |
| Total | FSMIM-S | 208,8 | 236,4 | 3,0 | 34,0 | 102,0 | 299,0 | 1111,0 |
| | FSMIM-T | 14,5 | 14,1 | 1,0 | 5,0 | 10,0 | 17,0 | 70,0 |

**Tabla 5.14:** Resumen estadístico del incremento RPC del número de LUT usadas por la arquitectura FSMIM-S respecto a FSMIM-T en las pruebas de implementación con optimización en área ( %).

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 1952,9 | 2086,2 | 171,4 | 541,3 | 1312,5 | 2497,3 | 12850,0 |
| BP-24 | 427,8 | 363,1 | 25,0 | 150,0 | 325,0 | 600,0 | 1200,0 |
| MCNC | 210,1 | 152,1 | 77,8 | 85,7 | 175,0 | 220,0 | 525,0 |
| Total | 1565,1 | 1948,2 | 25,0 | 346,2 | 894,1 | 1909,1 | 12850,0 |

una parte, que las LUT son un recurso muy abundante en comparación con el número de bloques empotrados disponibles en los dispositivos FPGA. Por ejemplo, las FSMIM-T usan de media 21,5 bloques de los 172 disponibles en el dispositivo utilizado (el 12 %) mientras que las FSMIM-S usan de media 208,8 LUT de las 46 678 disponibles (el 0,4 %).

Por otra parte, el objetivo inicial de las arquitecturas FSMIM es la implementación de FSM mediante el uso eficiente de los recursos disponibles (tanto LUT como bloques de memoria empotrados). La arquitectura de referencia para las FSM es la empleada en las implementaciones convencionales sobre celdas lógicas. Por lo tanto, una buena medida de la eficiencia en el uso de bloques es el cálculo del número de LUT que se ahorran respecto a la implementación convencional por cada bloque usado para implementar la FSMIM.

**Definición 5.4.** *Sea $L_{ISE\text{-}LUT}$ es el número de LUT consumidas por la implementación convencional de una FSM dada (generada por la herramienta ISE WebPACK). Sea $L_{mem}$ y $B_{mem}$, el número de LUT y el número de bloques, respectivamente, de una implementación basada en memoria de dicha FSM. Se define el* **Número de LUT Ahorradas por Bloque** *(NLAB) de la implementación basada en memoria como*

$$NLAB = \frac{L_{ISE\text{-}LUT} - L_{mem}}{B_{mem}}. \tag{5.1}$$

La tabla 5.15 muestra un resumen estadístico del NLAB obtenido por las arquitecturas FSMIM-S y FSMIM-T en las pruebas de implementación. Puede observarse que las baterías de pruebas que presentan mayores NLAB son las que consumían un mayor número de LUT. El NLAB medio es 169,6 LUT/bloque para las implementaciones FSMIM-T y 360,4 LUT/bloque para las implementaciones FSMIM-S. La proporción de LUT por bloque en el dispositivo utilizado, aproximadamente 271 LUT/bloque, puede ser útil como

punto de referencia para valorar la magnitud de estos valores. Por ejemplo, con el NLAB medio de las implementaciones FSMIM-S de la batería de pruebas BP-184 (440,8 LUT/bloque), el número de LUT ahorradas por el 62 % de los bloques es equivalente al número total de LUT del dispositivo FPGA utilizado.

La tabla 5.16 muestra un resumen estadístico del incremento RPC del NLAB de las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T (la figura 5.6 representa los valores de la tabla). Las implementaciones FSMIM-S presentan un incremento medio del 89,1 %, con incrementos entre el 74,9 % y el 275,9 % para la mitad de los casos.

El mayor incremento es el de la batería de pruebas BP-184, con un incremento medio del 105,9 %. Presenta incrementos superiores al 100 % para la mitad de los casos y su porcentaje de éxito es del 95 %. Le sigue la batería de pruebas BP-24 con un incremento medio del 62,6 % y con incrementos entre el 66,2 % y el 185 % para la mitad de los casos. Su porcentaje de éxito es del 80 %.

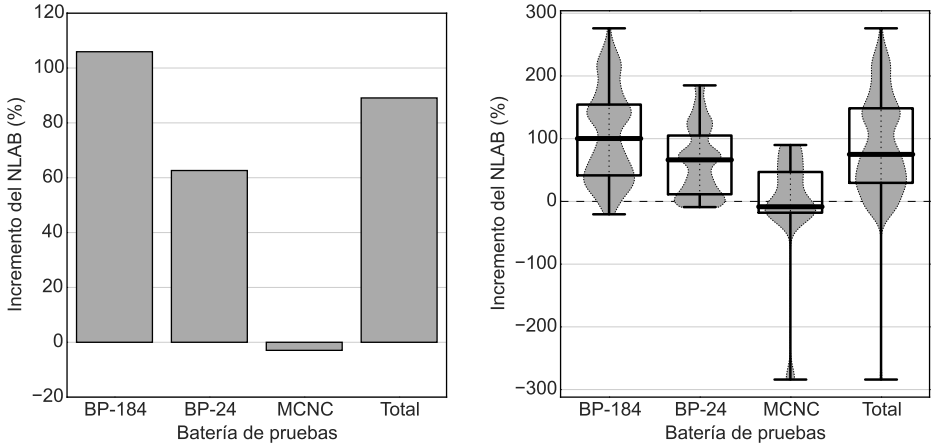
El peor resultado es para la batería de pruebas MCNC, que presenta un incremento medio negativo del $-2,9$ %. Es decir, en promedio, son las implementaciones FSMIM-T las que consiguen un incremento del 2,9 % en el NLAB respecto a las implementaciones FSMIM-S. Sin embargo, el porcentaje de éxito de las implementaciones FSMIM-S es del 46 % con un incremento medio del 55,6 %. Tal como se aprecia en la figura 5.6b, existe un valor atípico extremo en la batería de pruebas MCNC (corresponde a la FSM *scf*). Si eliminamos el mejor y el peor caso de MCNC (es decir, los dos extremos), el incremento medio resultante es del 14,2 %.

**Tabla 5.15:** Resumen estadístico del NLAB obtenido por las arquitecturas FSMIM-S y FSMIM-T en las pruebas de implementación con optimización en área.

| Bat. de pruebas | Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-S | 440,8 | 374,6 | 48,7 | 130,7 | 301,7 | 666,8 | 1356,1 |
| | FSMIM-T | 197,3 | 141,7 | 37,3 | 82,1 | 140,7 | 293,3 | 544,7 |
| BP-24 | FSMIM-S | 142,6 | 144,4 | 20,0 | 57,0 | 90,0 | 164,9 | 622,0 |
| | FSMIM-T | 101,5 | 109,2 | 11,0 | 26,2 | 69,3 | 122,9 | 427,3 |
| MCNC | FSMIM-S | 69,6 | 56,9 | 4,6 | 34,7 | 61,0 | 82,0 | 226,0 |
| | FSMIM-T | 57,9 | 34,6 | 17,0 | 24,0 | 51,0 | 72,0 | 119,0 |
| Total | FSMIM-S | 360,4 | 359,2 | 4,6 | 90,0 | 189,8 | 563,3 | 1356,1 |
| | FSMIM-T | 169,6 | 139,3 | 11,0 | 69,0 | 115,3 | 258,6 | 544,7 |

**Tabla 5.16:** Resumen estadístico del incremento RPC del NLAB de las implementaciones FSMIM-S respecto a FSMIM-T en las pruebas de implementación con optimización en área (%).

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 105,9 | 74,6 | −20,5 | 41,5 | 100,2 | 154,4 | 275,9 |
| BP-24 | 62,6 | 57,3 | −9,1 | 11,4 | 66,2 | 105,0 | 185,0 |
| MCNC | −2,9 | 93,3 | −284,0 | −18,0 | −8,5 | 46,9 | 89,9 |
| Total | 89,1 | 80,9 | −284,0 | 29,6 | 74,9 | 148,4 | 275,9 |




**(a)** **(b)**

**Figura 5.6:** Incremento RPC del NLAB de las implementaciones FSMIM-S respecto a FSMIM-T en las pruebas de implementación con optimización en área: (a) reducciones medias, (b) distribución de los valores obtenidos.

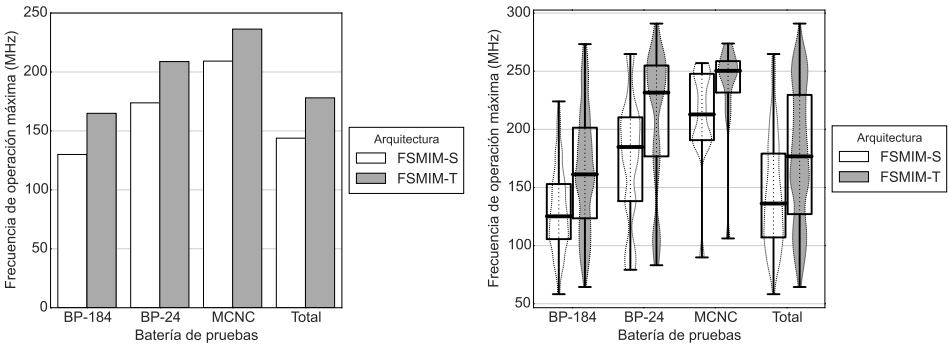
### Estudio de la frecuencia de operación máxima

En este apartado se comparan las frecuencias de operación máximas conseguidas con cada arquitectura de FSMIM cuando se realiza una optimización en área. La tabla 5.17 muestra un resumen estadístico de las frecuencias obtenidas (en MHz), y la figura 5.7, la representación correspondiente. Como cabía esperar, en promedio, las FSM más rápidas son las más pequeñas: las frecuencias mayores se obtienen con la batería de pruebas MCNC, y las frecuencias menores, con la batería de pruebas BP-184.

Por otra parte, el codificador de grupos y el incremento de complejidad del selector de entradas provocan una reducción de la velocidad de las im-

**Tabla 5.17:** Resumen estadístico de la frecuencia de operación máxima (MHz) obtenida por las implementaciones FSMIM-S y FSMIM-T en las pruebas de implementación con optimización en área.

| Bat. de pruebas | Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-S | 130,0 | 35,8 | 58,2 | 105,6 | 125,2 | 153,0 | 224,0 |
| | FSMIM-T | 164,9 | 51,6 | 64,4 | 123,5 | 161,3 | 201,4 | 273,3 |
| BP-24 | FSMIM-S | 173,8 | 54,8 | 79,2 | 138,3 | 184,8 | 210,3 | 264,8 |
| | FSMIM-T | 208,8 | 63,0 | 83,1 | 176,7 | 231,6 | 254,9 | 291,0 |
| MCNC | FSMIM-S | 209,2 | 44,5 | 89,9 | 190,7 | 212,8 | 247,8 | 256,9 |
| | FSMIM-T | 236,4 | 43,1 | 106,2 | 231,6 | 250,3 | 258,7 | 273,8 |
| Total | FSMIM-S | 143,8 | 47,4 | 58,2 | 107,1 | 136,2 | 179,1 | 264,8 |
| | FSMIM-T | 178,1 | 57,7 | 64,4 | 127,1 | 176,7 | 229,6 | 291,0 |




**(a)** **(b)**

**Figura 5.7:** Frecuencia de operación máxima obtenida por las implementaciones FSMIM-S y FSMIM-T en las pruebas de implementación con optimización en área: (a) reducciones medias, (b) distribución de los valores obtenidos.

plementaciones FSMIM-S respecto a las implementaciones FSMIM-T. La tabla 5.18 muestra un resumen estadístico de la reducción RPC de la frecuencia de operación máxima de las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T (la figura 5.8 representa los valores de la tabla). La reducción media es del 17,5 %. El 75 % de los casos presenta una reducción de la frecuencia inferior al 26 %. Comparada con las reducciones de bloques conseguida por las implementaciones FSMIM-S (del orden del 47 %), las reducciones en frecuencia no son muy significativas.

Si se analizan las reducciones obtenidas en cada batería de pruebas, se

observa que el incremento del tamaño medio de las FSM no tiene una influencia significativa en la reducción media de frecuencia: la reducción media obtenida por la batería de pruebas BP-184 está sólo 6.8 puntos por encima de la obtenida con MCNC (véase la figura 5.8a). Sin embargo, el efecto sobre el intervalo de valores obtenidos (es decir, sobre las reducciones máximas y mínimas obtenidas) es mayor (véase la figura 5.8b). De hecho, el porcentaje de casos en los que las implementaciones FSMIM-T son más rápidas que las implementaciones FSMIM-S disminuye ligeramente con el incremento del tamaño medio de las FSM (obsérvese cómo aumentan los valores negativos en la figura 5.8b). Mientras que la frecuencia de las FSMIM-S es menor que la de las FSMIM-T en el 100 % de los casos de MCNC, este porcentaje disminuye ligeramente al 90,5 % en la batería de pruebas BP-24 y al 88,8 % en BP-184. Una codificación óptima de estados y grupos podría aumentar aún más estos porcentajes, especialmente en las FSM grandes, en las que la influencia de la codificación puede ser mayor.

#### 5.3.3.2. Análisis de las implementaciones optimizadas en velocidad

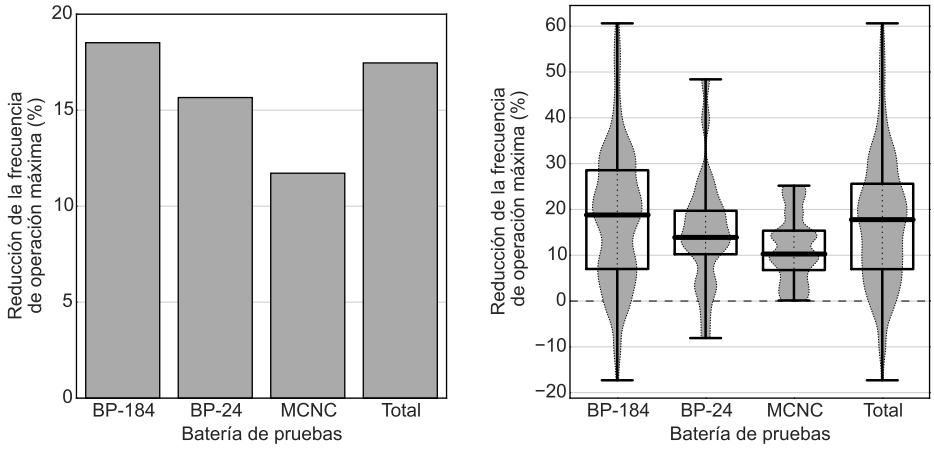
El objetivo de optimización utilizado en las pruebas de implementación no ha afectado al número de bloques usados por las implementaciones FSMIM (es decir, se han obtenido los mismos resultados en las pruebas con optimización en velocidad y en las pruebas con optimización en área). Por lo tanto, en esta sección sólo se estudiará la frecuencia de operación máxima y el consumo de LUT. Para ambos parámetros, se analizarán los resultados obtenidos en las pruebas con optimización en velocidad y se compararán con los obtenidos en las pruebas con optimización en área.

**Estudio de la frecuencia de operación máxima**

El análisis de los resultados de las implementaciones optimizadas en velocidad comenzará por el estudio de la frecuencia de operación máxima, ya que este es el parámetro que se pretende optimizar. La tabla 5.19 muestra un resumen estadístico de las frecuencias de operación máximas obtenidas por las implementaciones FSMIM-S y FSMIM-T, y la figura 5.10, una representación de dicho resumen. El incremento RPC de la frecuencia respecto a los resultados de implementación con optimización en área ha sido pequeño; de media, este incremento ha sido aproximadamente del 2,8 % para las implementaciones FSMIM-T (correspondiente a un aumento medio de 4 MHz) y del 9,4 % para las implementaciones FSMIM-S (correspondiente a

**Tabla 5.18:** Reducción RPC de la frecuencia de operación máxima de las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T en las pruebas de implementación con optimización en área ( %).

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 18,5 | 15,3 | −17,3 | 7,0 | 18,8 | 28,6 | 60,6 |
| BP-24 | 15,7 | 13,0 | −8,1 | 10,2 | 13,9 | 19,7 | 48,4 |
| MCNC | 11,7 | 7,6 | 0,1 | 6,8 | 10,3 | 15,4 | 25,2 |
| Total | 17,5 | 14,5 | −17,3 | 7,0 | 17,8 | 25,6 | 60,6 |




**(a)** **(b)**

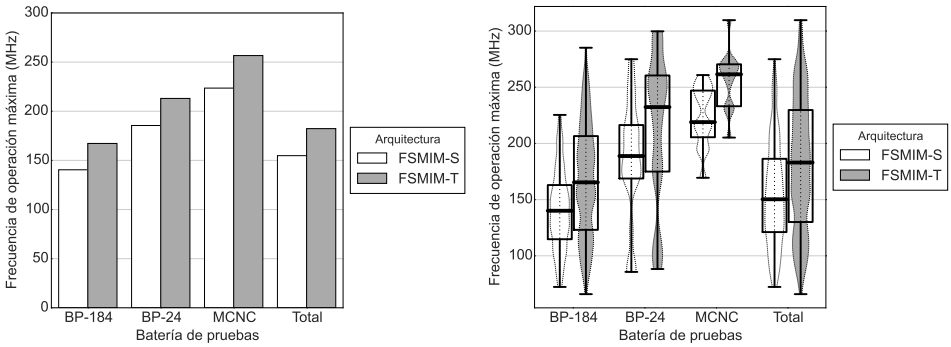
**Figura 5.8:** Reducción RPC de la frecuencia de operación máxima de las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T en las pruebas de implementación con optimización en área: (a) reducciones medias, (b) distribución de los valores obtenidos.

un aumento medio de 11 MHz). Por lo tanto, aunque las implementaciones FSMIM-T siguen siendo las que tienen mejores prestaciones en la velocidad, la influencia de la optimización en velocidad ha sido mayor en las implementaciones FSMIM-S. Este comportamiento puede indicar que todavía queda margen suficiente para reducir la complejidad del banco de selectores de entrada y del codificador de grupos de la FSMIM-S utilizando técnicas como la búsqueda de una codificación óptima de estados y de grupos.

El efecto desigual que ha tenido la optimización en velocidad sobre las dos arquitecturas ha provocado una disminución en la reducción RPC de la

**Tabla 5.19:** Resumen estadístico de la frecuencia de operación máxima (MHz) obtenida por las implementaciones FSMIM-S y FSMIM-T en las pruebas de implementación con optimización en velocidad.

| Bat. de pruebas | Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-S | 140,4 | 35,7 | 72,3 | 114,8 | 140,1 | 163,0 | 225,4 |
| | FSMIM-T | 167,2 | 50,3 | 65,9 | 123,1 | 165,4 | 206,5 | 285,1 |
| BP-24 | FSMIM-S | 185,5 | 53,7 | 85,7 | 168,8 | 188,8 | 216,4 | 275,0 |
| | FSMIM-T | 213,1 | 67,2 | 88,2 | 174,9 | 232,3 | 260,5 | 299,8 |
| MCNC | FSMIM-S | 223,6 | 27,8 | 169,4 | 205,5 | 219,0 | 247,0 | 260,8 |
| | FSMIM-T | 256,6 | 26,4 | 205,2 | 233,1 | 261,5 | 270,4 | 309,7 |
| Total | FSMIM-S | 154,8 | 46,7 | 72,3 | 121,1 | 150,3 | 186,3 | 275,0 |
| | FSMIM-T | 182,3 | 58,7 | 65,9 | 130,1 | 183,0 | 229,7 | 309,7 |




**(a)** **(b)**

**Figura 5.9:** Frecuencia de operación máxima (MHz) obtenida por las implementaciones FSMIM-S y FSMIM-T en las pruebas de implementación con optimización en velocidad: (a) valores medios, (b) distribución de los valores obtenidos.

frecuencia de operación máxima de las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T (véase el resumen estadístico de este índice en la tabla 5.20 y en la figura 5.10). La reducción máxima de la frecuencia ha disminuido 13 puntos respecto a los resultados de las implementaciones con optimización en área (ha pasado del 60,6 % al 47,4 %). La reducción media de la frecuencia es del 13,4 %, y en el 75 % de los casos es menor del 19,6 %. De las tres baterías de pruebas, la única que ha aumentado la reducción media ha sido MCNC.

**Tabla 5.20:** Resumen estadístico de la reducción RPC de la frecuencia de operación máxima de las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T en las pruebas de implementación con optimización en velocidad (%).

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 13,8 | 12,7 | −18,8 | 6,3 | 13,7 | 21,5 | 47,4 |
| BP-24 | 11,3 | 10,5 | −5,9 | 3,4 | 12,9 | 18,7 | 37,1 |
| MCNC | 13,0 | 5,4 | 2,7 | 8,7 | 15,1 | 16,4 | 21,0 |
| Total | 13,4 | 11,9 | −18,8 | 5,8 | 13,7 | 19,6 | 47,4 |

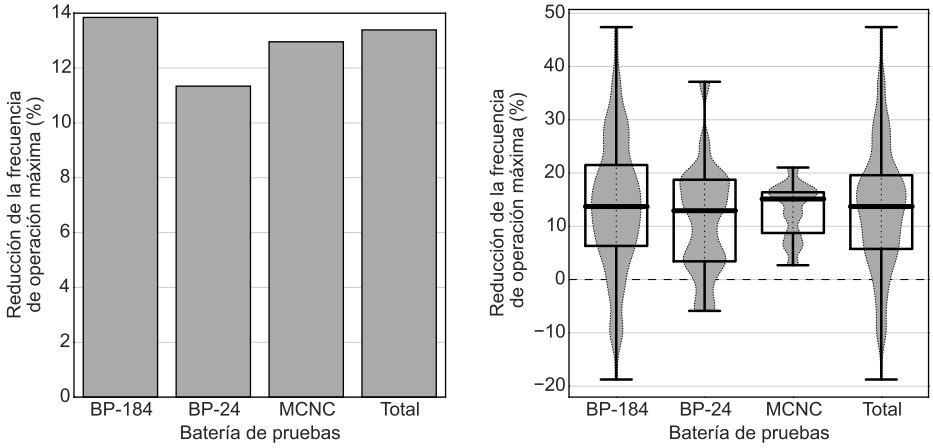



**Figura 5.10:** Reducción RPC de la frecuencia de operación máxima de las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T en las pruebas de implementación con optimización en velocidad: (a) reducciones medias, (b) distribución de los valores obtenidos.

### Estudio del consumo de LUT

En este apartado se estudia cómo afecta la optimización en velocidad al consumo de LUT en las arquitecturas FSMIM. La tabla 5.21 muestra un resumen estadístico del número de LUT usadas por las arquitecturas FSMIM-S y FSMIM-T. Como cabía esperar, en términos generales, la optimización en velocidad ha provocado un incremento en el consumo de LUT. Este impacto negativo sobre el área ocupada ha sido menor en las implementaciones FSMIM-T debido a que el banco de selectores de entrada de esta arquitectura es un componente muy simple, que ofrece poco margen de mejora a los

algoritmos de optimización (tanto en área como en velocidad). Esto explica que las diferencias entre los resultados de implementaciones optimizadas en área y en velocidad sean pequeñas para esta arquitectura.

A pesar de que, en términos absolutos, el aumento del consumo de LUT provocado por la optimización en velocidad ha sido mayor para las implementaciones FSMIM-S, el incremento ha sido menor en términos relativos. La tabla 5.22 muestra un resumen estadístico del incremento RPC del número de LUT usadas por las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T. Tal como puede observarse, el incremento medio de LUT usadas es del 1551,7 % (13 puntos por debajo del incremento obtenido en las pruebas con optimización en área).

Para el cálculo del NLAB se ha tomado como referencia el número de LUT usadas por las implementaciones ISE-LUT optimizadas en velocidad. La tabla 5.23 muestra un resumen estadístico del NLAB obtenido por las implementaciones FSMIM-S y FSMIM-T. En términos generales, el NLAB ha crecido para las dos arquitecturas respecto a las pruebas con optimización en área. Esto se debe a que el incremento del consumo de LUT causado por la optimización en velocidad ha sido mayor para las implementaciones ISE-LUT que para las implementaciones FSMIM. El ahorro medio de LUT ha sido de 198,3 LUT/bloque para las implementaciones FSMIM-T (un 17 % mayor que en las pruebas con optimización en área) y de 431,8 LUT/bloque para las implementaciones FSMIM-S (un 20 % mayor).

Para comparar el ahorro de LUT conseguido por las dos arquitecturas FSMIM, se ha calculado el incremento RPC del NLAB de las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T (la tabla 5.24 y la figura 5.11 muestran un resumen estadístico de los valores obtenidos). El

**Tabla 5.21:** Resumen estadístico del número de LUT usadas por las implementaciones FSMIM-S y FSMIM-T en las pruebas de con optimización en velocidad.

| Bat. de pruebas | Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-S | 314,5 | 312,5 | 17,0 | 77,5 | 200,0 | 463,0 | 1416,0 |
| | FSMIM-T | 18,2 | 17,8 | 2,0 | 6,0 | 13,0 | 19,5 | 76,0 |
| BP-24 | FSMIM-S | 41,6 | 43,4 | 4,0 | 13,0 | 31,0 | 53,0 | 181,0 |
| | FSMIM-T | 12,0 | 14,2 | 1,0 | 3,0 | 7,0 | 15,0 | 53,0 |
| MCNC | FSMIM-S | 20,2 | 20,2 | 3,0 | 12,0 | 17,0 | 20,0 | 85,0 |
| | FSMIM-T | 8,5 | 5,2 | 5,0 | 5,0 | 7,0 | 9,0 | 24,0 |
| Total | FSMIM-S | 246,7 | 298,0 | 3,0 | 34,0 | 107,0 | 381,0 | 1416,0 |
| | FSMIM-T | 16,4 | 16,8 | 1,0 | 6,0 | 11,0 | 19,0 | 76,0 |

**Tabla 5.22:** Resumen estadístico del incremento RPC del número de LUT usadas por las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T en las pruebas con optimización en velocidad ( %).

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 1948,3 | 1814,5 | 171,4 | 593,2 | 1333,3 | 2428,1 | 9380,0 |
| BP-24 | 419,6 | 389,0 | −100,0 | 126,4 | 342,9 | 650,0 | 1400,0 |
| MCNC | 116,0 | 75,0 | −66,7 | 100,0 | 133,3 | 142,9 | 254,2 |
| Total | 1551,7 | 1737,5 | −100,0 | 350,0 | 896,4 | 2063,6 | 9380,0 |

**Tabla 5.23:** Resumen estadístico del NLAB obtenido por las implementaciones FSMIM-S y FSMIM-T en las pruebas con optimización en velocidad.

| Bat. de pruebas | Arquitec. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-S | 529,0 | 482,2 | 52,0 | 159,5 | 311,9 | 827,3 | 1843,2 |
| | FSMIM-T | 230,4 | 168,1 | 43,8 | 98,0 | 162,4 | 335,8 | 822,5 |
| BP-24 | FSMIM-S | 172,9 | 178,3 | 28,0 | 54,8 | 118,0 | 210,9 | 754,5 |
| | FSMIM-T | 121,5 | 134,0 | 10,0 | 32,5 | 79,0 | 139,7 | 518,3 |
| MCNC | FSMIM-S | 87,5 | 73,4 | 19,0 | 38,0 | 74,0 | 96,0 | 298,0 |
| | FSMIM-T | 69,9 | 42,0 | 17,0 | 29,8 | 64,7 | 85,0 | 156,0 |
| Total | FSMIM-S | 431,8 | 456,2 | 19,0 | 113,2 | 225,0 | 622,4 | 1843,2 |
| | FSMIM-T | 198,3 | 165,0 | 10,0 | 82,2 | 139,0 | 274,2 | 822,5 |

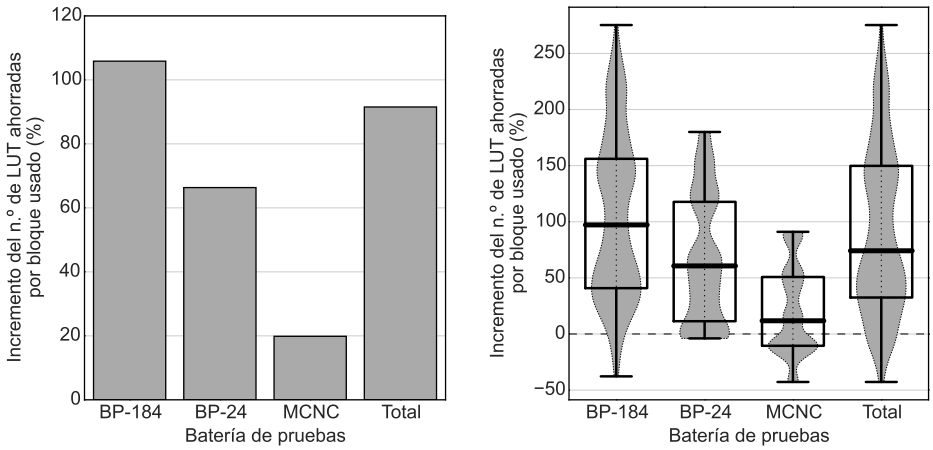
incremento medio del NLAB ha sido del 91,5 % (tan solo 2,4 puntos por encima del obtenido en las pruebas con optimización en área). Las diferencias más importantes se han dado en la batería de pruebas MCNC, que presenta valores mínimos claramente superiores a los de las pruebas con optimización en área (véanse las figuras 5.11 y 5.6).

### 5.3.4. Conclusiones

Aunque el objetivo principal de esta sección ha sido la comparación de las arquitecturas FSMIM-T y FSMIM-S, se ha realizado un estudio previo de la influencia del ajuste de profundidad sobre el consumo de recursos de las

**Tabla 5.24:** Resumen estadístico del incremento RPC del NLAB de las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T en las pruebas con optimización en velocidad ( %).

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 105,8 | 74,3 | −37,8 | 40,8 | 97,2 | 156,1 | 275,3 |
| BP-24 | 66,3 | 59,4 | −3,9 | 11,4 | 60,7 | 117,7 | 180,0 |
| MCNC | 19,9 | 41,6 | −42,8 | −10,5 | 11,8 | 50,8 | 91,0 |
| Total | 91,5 | 74,5 | −42,8 | 32,5 | 74,1 | 149,8 | 275,3 |




**(a)** **(b)**

**Figura 5.11:** Incremento RPC del NLAB de las implementaciones FSMIM-S respecto a las implementaciones FSMIM-T en las pruebas con optimización en velocidad: (a) reducciones medias, (b) distribución de los valores obtenidos.

implementaciones FSMIM. Dicho estudio demuestra que el ajuste de profundidad permite reducir tanto el consumo de bloques de memoria de las implementaciones FSMIM-T como el consumo de LUT de las implementaciones de ambas arquitecturas.

A pesar de que el porcentaje de casos en los que la técnica reduce el número de bloques de las implementaciones FSMIM-T no es muy elevado (21 % de los casos estudiados), puede alcanzar reducciones significativas (la reducción media ha sido del 20 % para los casos de éxito). La técnica es más efectiva respecto al consumo de LUT, ya que consigue mejorar los resultados en más de la mitad de los casos, siendo la reducción media del 22 % aproximadamente.

Por otra parte, el ajuste de profundidad tiene menor influencia en la arquitectura FSMIM-S, ya que sólo afecta al consumo de LUT. Aunque la reducción media ha sido pequeña en esta arquitectura (del 10 % aproximadamente), se ha obtenido un porcentaje de éxito del 60 %.

Estos resultados han justificado el uso del ajuste de profundidad en las FSMIM utilizadas en las pruebas de implementación realizadas en esta tesis. Además, este mecanismo de ajuste se ha incorporado a la herramienta FSMIM-Gen 2.0 como una mejora respecto a la herramienta FSMIM-Gen 1.2.

Respecto al estudio comparativo de ambas arquitecturas FSMIM, este ha demostrado la viabilidad de la arquitectura FSMIM-S para reducir el tamaño de la ROM, obteniendo mejoras importantes en el consumo de bloques de memoria. La reducción media conseguida ha sido del 48 % aproximadamente, con un porcentaje de éxito del 92 %, llegando a alcanzarse reducciones entre el 50 % y el 74 % en la mitad de los casos estudiados.

Sin embargo, esta reducción en el consumo de bloques se consigue a expensas de un incremento en el consumo de LUT, que repercute negativamente en la velocidad de operación. En promedio, la arquitectura FSMIM-S requiere 16 veces más LUT que la arquitectura FSMIM-T, reduciendo su velocidad un 18 % en las implementaciones optimizadas en área y un 13 % en las optimizadas en velocidad. En cualquier caso, aunque presenta un mayor consumo de LUT en términos absolutos, la arquitectura FSMIM-S es más eficiente en el uso de los recursos disponibles, ya que el número medio de LUT ahorradas por cada bloque usado es un 89 % mayor que en la arquitectura FSMIM-T.

Estos resultados muestran que la arquitectura FSMIM-S no sustituye a la arquitectura FSMIM-T, sino que la complementa ampliando el abanico de opciones de diseño disponibles para la implementación de máquinas de estados. La elección de la arquitectura FSMIM adecuada dependerá de los

objetivos de diseño y de los recursos disponibles. La arquitectura FSMIM-S será la mejor opción si el número de bloques disponibles es reducido. Por el contrario, la arquitectura FSMIM-T será la alternativa en diseños en los que la velocidad sea crítica. De cualquier manera, la inclusión de la arquitectura FSMIM-S en el repertorio de arquitecturas disponibles para la implementación de máquinas de estados permite un ajuste más fino de los principales parámetros de diseño: consumo de LUT, consumo bloques de memoria y frecuencia de operación máxima.

## 5.4. Estudio de la influencia de las estrategias de optimización sobre la arquitectura FSMIM-T

El objetivo de este estudio es evaluar las mejoras obtenidas con las nuevas estrategias de optimización (sección 5.1.1.2) cuando se aplican a la arquitectura FSMIM-T respecto a la estrategia de referencia. Por una parte, con las pruebas independientes del dispositivo se analizarán las mejoras en las propiedades de las FSMIM-T, lo que permitirá evaluar la eficiencia de los algoritmos propuestos para las distintas fases de optimización. Por otra parte, con las pruebas de implementación se cuantificará la influencia de las nuevas estrategias en las prestaciones de las implementaciones de FSMIM-T en dispositivos FPGA.

### 5.4.1. Pruebas independientes del dispositivo

El objeto principal de esta sección es el estudio de la influencia de las diferentes estrategias de optimización de FSMIM sobre el tamaño genérico de las FSMIM-T. Además, se estudiará la efectividad de las estas estrategias en la reducción de la complejidad del banco de selectores de entradas de la FSMIM-T (es decir, del banco de multiplexores). Por último, se analizará experimentalmente el coste temporal del proceso de generación de FSMIM, así como la bondad de las soluciones encontradas mediante los modelos basados en PLE[8].

El estudio se ha realizado con las 242 FSM que componen las tres baterías de pruebas. Para cada FSM se han generado cinco FSMIM sin ajuste de profundidad utilizando la estrategia de referencia (M-ref) y las cuatro nuevas estrategias propuestas en esta tesis: A-PLE, A-voraz, M-PLE y M-voraz (véase la sección 5.1). A partir de cada FSMIM, se han calculado las características de las FSMIM-T que se analizarán en esta sección.

Puesto que la complejidad de un multiplexor está directamente relacionada con el número de entradas, se utilizará el coste de selección de entradas[9] (Definición 2.20) para estimar la complejidad del banco de multiplexores de entradas de la FSMIM-T. Se ha elegido esta medida por dos motivos. En primer lugar, el número de recursos necesarios para implementar un multiplexor depende de características que son específicas del dispositivo FPGA,

[8]Las herramientas de optimización que se emplean para resolver modelos PLE pueden calcular una cota superior del error existente entre el coste de la solución óptima y el de la mejor solución encontrada.

[9]Debido a que *CSE* hace referencia a una función definida sobre una MSE, se ha decidido utilizar el nombre completo para designar a esta medida en lugar de la sigla.

tales como el número de entradas de las LUT o la existencia de recursos lógicos adicionales (por ejemplo, los multiplexores empotrados disponibles en los dispositivos FPGA de Xilinx). Debido principalmente a la segunda de las características mencionadas, el estudio del consumo de recursos a partir del número de entradas de cada multiplexor no es factible en estas pruebas. Por tanto, no se estudiará la complejidad del banco de multiplexores en términos de recursos consumidos sino del número de entradas.

En segundo lugar, puesto que el coste de selección de entradas forma parte del objetivo de optimización de las fases de minimización de la selección de entradas y de agrupación de estados, su estudio permitirá evaluar la efectividad de las diferentes estrategias en el proceso de optimización de la FSMIM.

**Estudio del tamaño de la FSMIM-T**

Para cada FSMIM generada en las pruebas independientes del dispositivo se ha calculado el tamaño en Kbits de la FSMIM-T correspondiente (es decir, el tamaño de la ROM requerida en la implementación de FSMIM-T). La tabla 5.25 muestra un resumen estadístico de los valores obtenidos para el conjunto completo de FSM.

En promedio, las estrategias de tipo AM (es decir, A-PLE y A-voraz) obtienen mejores resultados que las estrategias de tipo MA (es decir, M-ref, M-PLE y M-voraz). Esto se debe a que estas estrategias consiguen agrupar un mayor número de estados, reduciendo así la profundidad de la ROM en mayor grado que el resto de estrategias. Además, como cabía esperar, la minimización de selección de entradas basada en PLE obtiene mejores resultados que la basada en algoritmos voraces tanto para las estrategias de tipo AM como para las de tipo MA. Por otra parte, las nuevas estrategias obtienen mejores resultados que la estrategia de referencia; aunque la mejora no es igual para todas las baterías de pruebas. La figura 5.12 muestra el tamaño medio obtenido para cada batería de pruebas. Puede observarse cómo las diferencias entre el tamaño obtenido por la estrategia de referencia y el resto de estrategias es claramente mayor en la batería de pruebas BP-184.

La tabla 5.26 muestra un resumen estadístico del tamaño de las FSMIM-T generadas con la estrategia M-ref. Los datos de la tabla indican que el tamaño de las FSMIM-T generadas con las baterías de pruebas MCNC y BP-24 es pequeño comparado con el tamaño de los bloques de memoria del dispositivo FPGA utilizado en las pruebas (el 75 % de las FSMIM-T tiene un tamaño menor que 18 Kbits). Entre las dos baterías de pruebas hay sólo 48 FSMIM-T con un tamaño mayor de 9 Kbits (el 36 % de los casos) y sólo

10 FSMIM-T con una profundidad mayor que 8 Kpalabras. Esto significa que el margen de mejora para las nuevas estrategias será pequeño cuando estas FSMIM-T se implementen en el dispositivo FPGA.

Con el fin de cuantificar la mejora obtenida al utilizar las nuevas estrategias de optimización de FSMIM, se ha calculado la reducción RPC del tamaño de las FSMIM-T obtenidas por las nuevas estrategias respecto a la estrategia de referencia. La tabla 5.27 muestra un resumen estadístico de los valores obtenidos y la figura 5.13, una representación de dicho resumen.

Respecto al conjunto completo de FSM, la reducción media se encuentra

**Tabla 5.25:** Resumen estadístico del tamaño de las FSMIM-T (Kbits) obtenidas con el conjunto completo de FSM.

| Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| A-PLE | 158,9 | 375,4 | 0,1 | 1,7 | 21,4 | 108,0 | 2980,0 |
| A-voraz | 164,3 | 376,0 | 0,1 | 1,7 | 22,2 | 126,4 | 2980,0 |
| M-PLE | 170,8 | 374,9 | 0,1 | 1,6 | 28,1 | 147,2 | 3154,0 |
| M-voraz | 183,2 | 390,7 | 0,1 | 1,6 | 30,6 | 166,5 | 3198,0 |
| M-ref | 209,1 | 431,8 | 0,1 | 1,9 | 35,5 | 198,1 | 3320,0 |

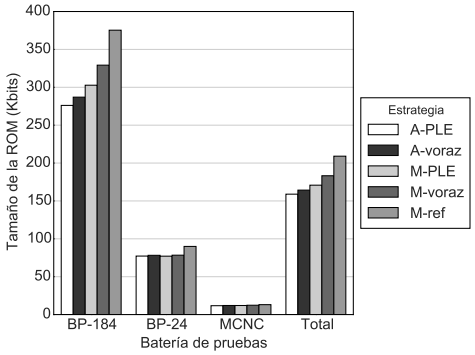



**Figura 5.12:** Tamaño medio de las FSMIM-T (Kbits).

**Tabla 5.26:** Resumen estadístico del tamaño de las FSMIM-T (Kbits) generadas con la estrategia M-ref.

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 375,4 | 507,8 | 17,5 | 85,8 | 186,0 | 400,0 | 3320,0 |
| BP-24 | 90,0 | 332,3 | 0,1 | 0,5 | 1,9 | 17,5 | 2184,0 |
| MCNC | 13,1 | 17,9 | 0,1 | 1,7 | 7,2 | 16,2 | 78,0 |

entre el 10,2 % (obtenida por la estrategia M-voraz) y el 22,5 % (obtenida por la estrategia A-PLE). La estrategia A-PLE obtiene reducciones entre el 39 % y el 75,6 % para el 25 % de los casos, siendo las reducciones de la estrategia A-voraz del mismo orden.

Los mejores resultados se obtienen para la batería de pruebas BP-184 (véase la figura 5.13), en la que se consiguen reducciones medias entre el 14,8 % de la estrategia M-voraz y el 40,2 % de la estrategia A-PLE. En la mitad de los casos, la estrategia A-PLE consigue reducciones entre el 42,6 % y el 75,6 % (las reducciones de la estrategia A-voraz son del mismo orden). Por otra parte, la estrategia M-PLE alcanza reducciones entre el 31,5 % y el 45,9 % en el 25 % de los casos.

Los peores resultados se obtienen para la batería de pruebas MCNC (véase la figura 5.13), en la que la que sólo la estrategia A-PLE consigue reducir el tamaño de la FSMIM-T en más de la mitad de los casos. Dicha estrategia consigue reducciones superiores al 10 % sólo en el 25 % de los casos, mientras que para el resto de estrategias el porcentaje de casos es menor.

La figura 5.13c muestra el porcentaje de éxito de las nuevas estrategias respecto a la estrategia de referencia (los valores negativos corresponden a porcentajes de fracaso). De nuevo, los mejores resultados se dan en la batería de pruebas BP-184 (con porcentajes de éxito superiores al 96 % para todas las estrategias) y los peores, en la batería de pruebas MCNC (con porcentajes de éxito entre el 26 % y el 53 %). Respecto a las diferentes estrategias, los mejores resultados se dan de nuevo en la estrategia A-PLE (con un porcentaje de éxito del 77,3 % de los casos del conjunto completo de FSM) y los peores, en la estrategia M-voraz (con un porcentaje de éxito del 70,3 %). Por el contrario, el porcentaje de fracaso obtenido con el conjunto completo de FSM es inferior al 4,2 % para todas las estrategias. Es en la batería de pruebas MCNC donde se obtiene el peor porcentaje de fracaso con un valor del 11,5 % (que corresponde a 3 casos).

Los diferentes parámetros estadísticos analizados parecen indicar que las mejoras obtenidas por las nuevas estrategias aumentan con el tamaño de las FSM a las que se les aplica la técnica. Para corroborar este comportamiento, se ha estudiado la influencia que tiene el tamaño de la FSM-ROM en las reducciones del tamaño de la FSMIM-T obtenidas por las nuevas estrategias respecto a M-ref (véase la figura 5.14). Las rectas de regresión mostradas en la figura indican claramente que, en todas las estrategias, la reducción crece con el tamaño de la FSM-ROM. Puede observarse, además, cómo la pendiente de la recta es mayor en las estrategias de tipo AM que en las de

**Tabla 5.27:** Resumen estadístico de la reducción RPC del tamaño de la FSMIM-T obtenida por las nuevas estrategias respecto a M-ref ( %).

| Bat. de pruebas | Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | A-PLE | 40,2 | 19,2 | −10,4 | 25,3 | 42,6 | 54,4 | 75,6 |
| | A-voraz | 35,0 | 19,3 | −5,6 | 20,3 | 35,1 | 51,5 | 73,9 |
| | M-PLE | 22,5 | 10,9 | −0,9 | 14,4 | 22,7 | 31,5 | 45,9 |
| | M-voraz | 14,8 | 8,8 | −18,6 | 8,9 | 15,4 | 20,0 | 38,6 |
| BP-24 | A-PLE | 8,5 | 9,2 | −16,7 | 0,0 | 8,2 | 14,4 | 33,3 |
| | A-voraz | 7,4 | 9,0 | −16,7 | 0,0 | 6,5 | 14,3 | 32,2 |
| | M-PLE | 8,8 | 9,4 | 0,0 | 0,0 | 7,5 | 14,8 | 33,3 |
| | M-voraz | 7,4 | 8,9 | −4,2 | 0,0 | 3,6 | 13,5 | 32,2 |
| MCNC | A-PLE | 7,1 | 10,1 | −3,7 | 0,0 | 4,5 | 11,6 | 38,9 |
| | A-voraz | 5,6 | 9,7 | −5,0 | 0,0 | 0,0 | 8,4 | 36,1 |
| | M-PLE | 5,4 | 7,6 | 0,0 | 0,0 | 0,0 | 9,7 | 27,9 |
| | M-voraz | 3,2 | 6,7 | 0,0 | 0,0 | 0,0 | 2,0 | 27,9 |
| Total | A-PLE | 22,5 | 21,6 | −16,7 | 2,7 | 15,5 | 39,0 | 75,6 |
| | A-voraz | 19,5 | 20,1 | −16,7 | 1,3 | 14,3 | 33,6 | 73,9 |
| | M-PLE | 14,6 | 12,3 | −0,9 | 1,6 | 13,5 | 24,1 | 45,9 |
| | M-voraz | 10,2 | 9,6 | −18,6 | 0,0 | 9,5 | 16,9 | 38,6 |

tipo MA y, para cada tipo, en las estrategias con minimización de selección de entradas basada en PLE.

### Estudio del coste de selección de entradas

La figura 5.15 muestra el coste de selección de entradas de las FSMIM-T generadas por las distintas estrategias estudiadas. Como puede observarse, de las nuevas estrategias, las de tipo AM son las que presentan un coste medio más alto. Esto se debe principalmente a que la fase inicial de agrupación de estados restringe el número de permutaciones de entradas posibles que se pueden realizar en la MSE, reduciéndose el número de soluciones respecto al problema de minimización de la selección de entradas original (véase la sección 2.4.2). Como consecuencia, el coste mínimo de selección de entradas de las estrategias de tipo AM es mayor o igual que el coste mínimo de las estrategias de tipo MA. Adicionalmente, puesto que las estrategias de tipo AM consiguen agrupar un mayor número de estados, las MSE resultantes contienen mayor número de constantes y menor número de entradas seleccionadas indeterminadas que las obtenidas con las estrategias de tipo MA. En términos generales, se puede concluir, por lo tanto, que las estrategias de tipo AM consiguen mayores reducciones del tamaño de la ROM que el

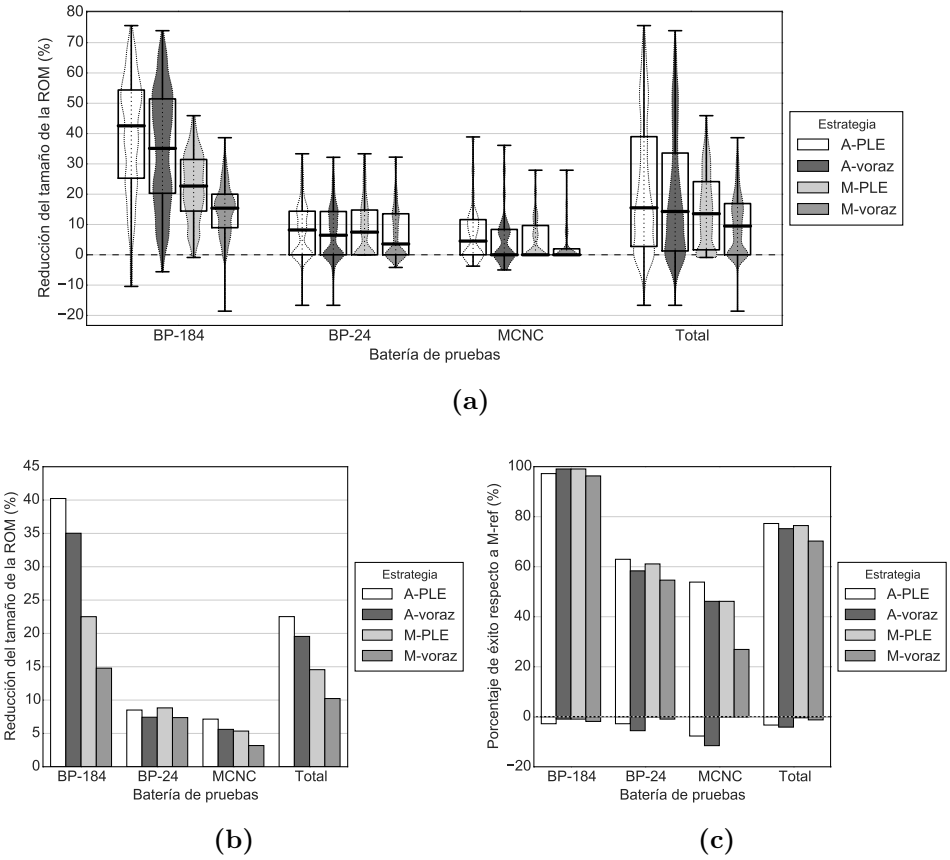



**Figura 5.13:** Reducción RPC del tamaño de la FSMIM-T obtenida por las nuevas estrategias respecto a M-ref: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

resto de estrategias (véase la figura 5.13) a costa de incrementar el coste de selección de entradas.

Por el contrario, las nuevas estrategias de tipo MA son las que tienen, en promedio, menor coste de selección de entradas, que consiguen a expensas de un incremento del tamaño de la ROM respecto a las estrategias de tipo AM. Estos resultados indican que el coste de selección de entradas y el tamaño de la ROM son objetivos de optimización en conflicto. Sin embargo, comparadas con la estrategia M-ref, las nuevas estrategias de tipo MA mejoran ambos parámetros, ya que presentan, en promedio, menores tamaños de ROM y menores costes de selección de entradas.

Para comparar convenientemente las nuevas estrategias con la estrategia

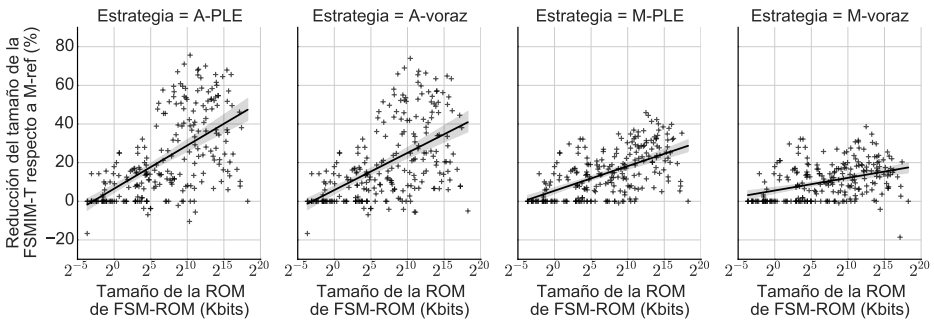


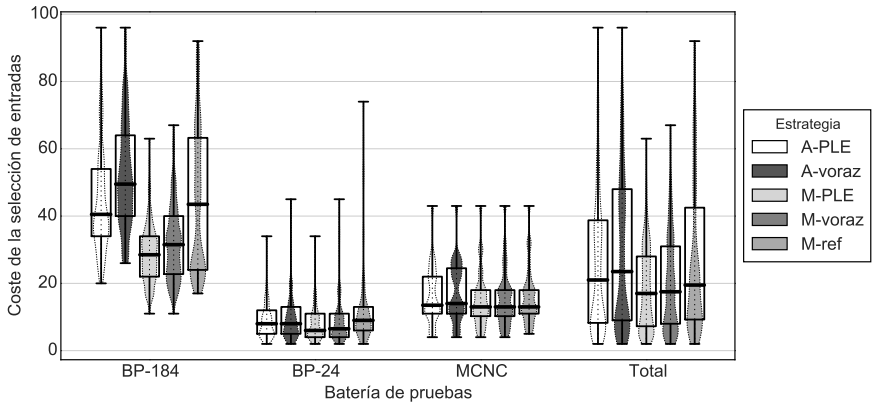

**Figura 5.14:** Reducción RPC del tamaño de la FSMIM-T obtenida por las nuevas estrategias respecto a M-ref frente al tamaño de la ROM de la implementación FSM-ROM.

M-ref, se ha calculado la reducción RPC del coste de selección de entradas conseguida por las nuevas estrategias respecto a M-ref. La figura 5.16 muestra un resumen estadístico de los valores obtenidos.

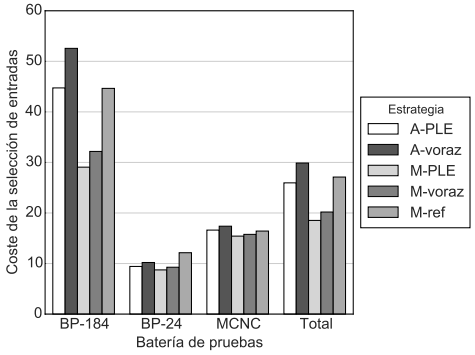
Los mejores resultados se obtienen para las nuevas estrategias de tipo MA, que presentan reducciones medias del 21 % (estrategia M-PLE) y del 18 % (estrategia M-voraz) para el conjunto completo de FSM (véase la figura 5.16b), y un porcentaje de éxito cercano al 60 % (véase la figura 5.16c). En el 25 % de los casos, la estrategia M-PLE presenta reducciones entre el 43 % y 67 % (véase la figura 5.16a). Por otra parte, el porcentaje de fracaso es inferior al 7 % para ambas estrategias.

Los peores resultados para las nuevas estrategias de tipo MA se presentan en la batería de pruebas MCNC, con una reducción media del 6 % (estrategia M-PLE) y del 4 % (estrategia M-voraz), y con un porcentaje de éxito del 31 % (estrategia M-PLE) y del 19 % (estrategia M-voraz). Los mejores resultados para estas estrategias se presentan en la batería de pruebas BP-184 con una reducción media del 27 % (estrategia M-PLE) y del 21 % (estrategia M-voraz), y con un porcentaje de éxito del 81 % (estrategia M-PLE) y del 78 % (estrategia M-voraz).

Respecto a las estrategias de tipo AM, como cabía esperar, estas obtienen peores resultados que las de tipo MA, presentando reducciones medias negativas en dos de las tres baterías de pruebas. Sin embargo, para la estrategia A-PLE las reducciones medias nunca son inferiores al −6 %. Cabe destacar que, en la batería de pruebas BP-24, estas estrategias consiguen reducciones medias del 13 % (estrategia A-PLE) y del 10 % (estrategia A-voraz), con porcentajes de éxito cercanos al 50 %; mejorando tanto el coste de selección de entradas como el tamaño de la ROM respecto a M-ref.




**(a)**




**(b)**

**Figura 5.15:** Coste de selección de entradas de las FSMIM-T generadas en las pruebas independientes del dispositivo: (a) distribución de los valores obtenidos, (b) valor medio.

Además de influir en la complejidad del banco de selectores de entradas, el coste de selección repercute en el tamaño de la ROM de la FSMIM-T (ya que los bits de selección de entradas aumentan el ancho de la ROM). Por otra parte, como se ha mencionado anteriormente, la reducción del tamaño de la FSMIM-T también influye en el coste de selección de entradas. Por lo tanto, con objeto de obtener una visión completa de la influencia de las nuevas estrategias en las características de las FSMIM-T, es necesario realizar un estudio que relacione ambos parámetros.

Se ha calculado el porcentaje de éxito neto de la reducción RPC del tamaño de la ROM y de la reducción RPC del coste de selección de entradas de las nuevas estrategias respecto a M-ref. La figura 5.17 muestra los valores

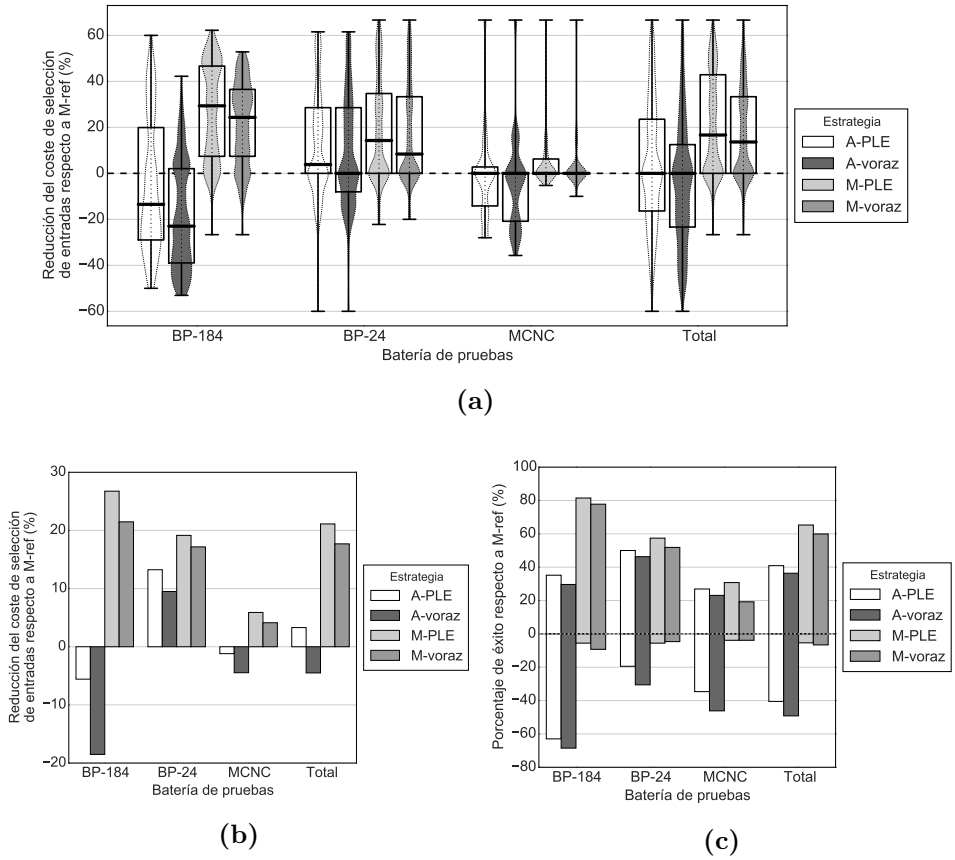



**Figura 5.16:** Reducción RPC del coste de selección de entradas de las FSMIM-T obtenida por las nuevas estrategias respecto a M-ref en las pruebas independientes del dispositivo: (a) distribución de los valores obtenidos, (b) valor medio, (c) porcentaje de éxito.

obtenidos. El porcentaje de éxito de las estrategias de tipo MA es superior al 65 % para el conjunto completo de FSM, alcanzándose porcentajes superiores al 87 % en la batería de pruebas BP-184. Por el contrario, el porcentaje de fracaso neto es inferior al 1 % en todas las baterías de pruebas.

Respecto a las estrategias de tipo AM, los porcentajes de éxito son similares a los obtenidos exclusivamente en la reducción del coste de selección de entradas. Sin embargo, el porcentaje de fracaso neto se ha reducido en más de 26 puntos.

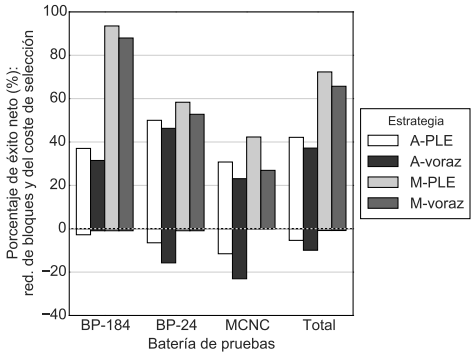



**Figura 5.17:** Porcentaje de éxito neto de la reducción RPC del tamaño de la ROM y de la reducción RPC del coste de selección de entradas de las FSMIM-T obtenido por las nuevas estrategias respecto a M-ref en las pruebas independientes del dispositivo.

### Estudio del coste temporal de la generación de FSMIM

En este apartado se realizará un estudio comparativo del tiempo de generación de las FSMIM utilizadas en las pruebas independientes del dispositivo. El objetivo de este estudio no es analizar la complejidad de los algoritmos utilizados, sino obtener una estimación orientativa del coste temporal. Existen varias razones por las que el tiempo obtenido en los experimentos no puede ser considerado un indicador completamente fiable del coste temporal de los algoritmos.

En primer lugar, las nuevas estrategias han sido implementadas utilizando un lenguaje de programación diferente al usado para implementar la estrategia de referencia (aunque, en ambos casos, se trata de lenguajes interpretados). En segundo lugar, todas las aplicaciones han sido desarrolladas con el único objetivo de disponer de herramientas para explorar las diferentes ideas que han surgido a lo largo del este trabajo de investigación (tanto las que se han presentado en esta tesis como muchas otras que han quedado finalmente desechadas o postergadas para los trabajos futuros). Por lo tanto, se trata de prototipos desarrollados sin aplicar técnicas para optimizar las prestaciones (ni en velocidad, ni en consumo de memoria).

Todos los experimentos se han realizado en un computador con el sistema operativo Linux de 64 bits ejecutándose en un procesador Intel® Core™ i7-860 con una frecuencia de reloj de 2.80 GHz, 4 núcleos y memoria caché de 8 MB. El sistema disponía de 8 GB de memoria RAM. Para la generación de FSMIM con la estrategia de referencia, se ha utilizado la aplicación FSMIM-Gen 1.2, que ha sido desarrollada usando Octave 3.8.1 [Eaton et al.,

2009]. Para generar las FSMIM con las nuevas estrategias, se ha utilizado la aplicación FSMIM-Gen 2.0, desarrollada con SAGE versión 6.6.1 [Sage, 2014], que utiliza como lenguaje de programación una extensión de Python 2.7.6 [Python, 2014]. En el caso de las estrategias de tipo PLE, el motor de optimización utilizado por SAGE ha sido Gurobi 5.6 [Gurobi, 2015]. El tiempo límite establecido para el proceso de optimización realizado por Gurobi ha sido de 900 segundos en cada una de las fases de minimización de selección de entradas.

La figura 5.18 muestra los tiempos de generación de FSMIM de las estrategias estudiadas. Estos tiempos incluyen tanto el tiempo de CPU de usuario como el del sistema. Como cabía esperar, las estrategias de tipo voraz son las más rápidas, con un tiempo medio menor de 2 segundos para el conjunto completo de FSM. El tiempo medio consumido por la estrategia M-ref es de 95 segundos. Por lo tanto, las estrategias de tipo voraz consiguen encontrar mejores soluciones que la estrategia M-ref en un tiempo menor.

Como cabía esperar, las estrategias de tipo PLE son las que requieren mayor tiempo de generación de FSMIM, con tiempos medios de 883 seg. (estrategia A-PLE) y 630 seg. (estrategia M-PLE). Como contrapartida, estas técnicas son las que generan las FSMIM con mejores prestaciones. Puesto que estas estrategias utilizan los mismos algoritmos de agrupación de estados que las estrategias de tipo voraz, este incremento de tiempo se debe a las fases de minimización de selección de entradas basadas en PLE, que realizan una búsqueda exhaustiva del espacio de soluciones. Los resultados indican, además, que los modelos PLE utilizados por las estrategias A-PLE tienen mayor coste computacional que los utilizados por las estrategias M-PLE.

El mayor consumo de tiempo se produce con la batería de pruebas BP-184. En promedio, las estrategias de tipo voraz consumen menos de 3 segundos en esta batería de pruebas, mientras que la estrategia M-ref requiere 211 segundos. Las estrategias de tipo PLE, presentan tiempos medios de generación por encima de los 1200 segundos. Esto se debe a que, en la mayoría de los casos, se ha agotado el tiempo límite en alguna de las fases de minimización de selección de entradas antes de encontrar la solución óptima.

Con objeto de estimar la posible desviación entre las soluciones encontradas y el valor óptimo, se ha calculado la distancia relativa entre la mejor solución y la mejor cota del valor óptimo encontrados por las estrategias de tipo PLE. La figura 5.19 muestra un resumen estadístico de los valores obtenidos para la última fase de minimización de selección de entradas.

Como puede observarse, la distancia relativa es prácticamente inexistente en las baterías de pruebas MCNC y BP-24: la distancia es cero (es decir, se ha encontrado la solución óptima) en todos los casos excepto siete en la

estrategia A-PLE y cuatro en la estrategia M-PLE. Sin embargo, en la batería de pruebas BP-184, el 73 % y el 88 % de los casos presentan una distancia relativa mayor que cero para las estrategias A-PLE y M-PLE, respectivamente, alcanzando distancias medias del 25 % (para la estrategia A-PLE) y del 19 % (para la estrategia M-PLE). Estos valores indican que, aumentando el tiempo límite, podrían mejorarse significativamente los resultados obtenidos en un número elevado de casos.

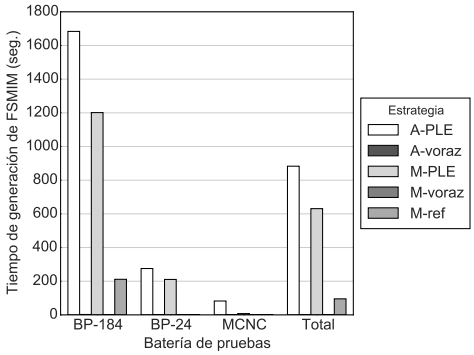



**Figura 5.18:** Tiempo medio de generación de FSMIM (seg.) de las diferentes estrategias en las pruebas independientes del dispositivo.

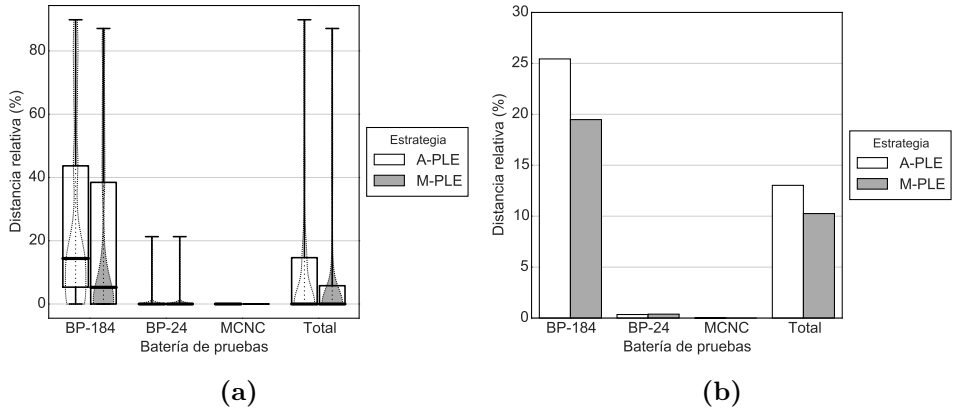



**Figura 5.19:** Distancia relativa entre la mejor cota del valor óptimo y la mejor solución encontrada en la última fase de minimización de selección de entradas para las estrategias de tipo PLE: (a) Distribución de los valores obtenidos y (b) valor medio.

### 5.4.2. Pruebas de implementación

Las pruebas independientes del dispositivo han mostrado que, comparadas con la estrategia M-ref, las nuevas estrategias consiguen reducir el tamaño de la FSMIM-T en un porcentaje de casos muy elevado, siendo estas reducciones especialmente significativas en las estrategias de tipo AM. Por otra parte, las nuevas estrategias de tipo MA consiguen reducir el coste de selección de entradas en prácticamente todos los casos en los que reducen el tamaño de la FSMIM-T, a pesar de que el coste de selección y el tamaño de la FSMIM-T son parámetros en conflicto.

Las pruebas de implementación que se presentan en esta sección persiguen, principalmente, los siguientes objetivos: (a) estudiar el efecto que tiene el uso de los bloques de memoria empotrados del dispositivo FPGA en las reducciones del tamaño de las FSMIM-T conseguidas por las nuevas estrategias, y (b) evaluar la influencia que tiene la reducción del coste de selección de entradas conseguida por las nuevas estrategias sobre el consumo de LUT y sobre la frecuencia de operación máxima de las implementaciones de FSMIM-T.

El estudio comparativo de las estrategias se ha realizado utilizando el conjunto completo de FSM. Para cada FSM se han generado cinco FSMIM-T con ajuste de profundidad, una por cada estrategia. Las FSMIM-T generadas se han implementado dos veces en el dispositivo FPGA, configurando *ISE WebPACK* con un objetivo de optimización diferente en cada implementación (optimización en área y en velocidad). Para el análisis de resultados se han seleccionado las FSM que cumplen las siguientes condiciones:

1. La implementación de la FSM-ROM ocupa más de un bloque de 9 Kbits (véase la sección 5.3.2, página 142).

2. Todas las estrategias consiguen generar una FSMIM-T al aplicar el ajuste de profundidad (véase la sección 5.3.2, página 142).

La FSM *fsm_e9i12o4s497*, perteneciente a la batería de pruebas BP-184, ha sido excluida del estudio debido a que todas las FSMIM-T generadas a partir de ella requieren más de 172 bloques de memoria (el máximo disponible en el dispositivo FPGA utilizado). En total, se han analizados los resultados de 161 FSM (107 de la batería de pruebas BP-184, 33 de BP-24 y 21 de MCNC).

El estudio se ha dividido en dos partes: la primera está dedicada al análisis de los resultados de las implementaciones optimizadas en área, y la segunda, a las optimizadas en velocidad. En ambas partes, se analizarán

tanto el área ocupada (en términos de número de LUT y de bloques de memoria) como la frecuencia de operación máxima. La estrategia M-ref se ha tomado como referencia para todos los cálculos comparativos realizados.

#### 5.4.2.1. Análisis de las implementaciones optimizadas en área

**Estudio del consumo de bloques**

La tabla 5.28 muestra un resumen estadístico del número de bloques ocupados por las FSMIM-T generadas con el conjunto completo de FSM. Como puede observarse, desde un punto de vista cualitativo, los resultados son en promedio similares a los obtenidos en las pruebas independientes del dispositivo. Las FSMIM-T de las estrategias de tipo AM consumen menos bloques que las de tipo MA. Se mantiene la relación entre las estrategias de tipo PLE y las de tipo voraz; es decir, tanto para las dos estrategias de tipo AM como para las dos de tipo MA, la que usa minimización de selección de entradas basadas en PLE consigue mejores resultados que la que usa minimización basada en algoritmos voraces. Finalmente, las nuevas estrategias obtienen mejores resultados que la estrategia M-ref.

El análisis de los datos de cada batería de pruebas por separado muestra un comportamiento similar con las pruebas independientes del dispositivo, como pone de manifiesto el parecido entre la figura 5.20 (que representa el número medio de bloques obtenido para cada batería de pruebas) y la figura 5.12 (que muestra el tamaño medio de las FSMIM-T obtenido en las pruebas independientes del dispositivo).

Un análisis preliminar del tamaño de las FSMIM-T generadas con la estrategia M-ref, permitirá una mejor interpretación de las reducciones de tamaño alcanzadas por las nuevas estrategias. La tabla 5.29 muestra un resumen estadístico del número de bloques ocupados por las FSMIM-T generadas con la estrategia M-ref para cada batería de pruebas. Tal como se indicó en la sección 5.3.3.1, el efecto más importante del uso de bloques sobre los resultados de las pruebas de implementación es la discretización de las posibles reducciones en el tamaño de las FSMIM-T. La consecuencia más inmediata es que, independientemente de la reducción genérica obtenida, no es posible reducir el tamaño de las FSMIM-T que ocupan 0,5 bloques. En esta categoría se encuentran el 38 % de los casos de la batería de pruebas MCNC y el 42 % de BP-24.

Por otra parte, las reducciones posibles para las FSMIM-T que ocupan 1 bloque son 0 % y 50 %. Esto significa que las reducciones genéricas deben ser mayores o iguales al 50 % para garantizar una reducción en el número de

**Tabla 5.28:** Resumen estadístico del número de bloques usados por las implementaciones FSMIM-T obtenidas con el conjunto completo de FSM en las pruebas de implementación con optimización en área. Las estrategias están ordenadas de menor a mayor número medio de bloques.

| Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| A-PLE | 13,1 | 22,4 | 0,5 | 1,0 | 3,5 | 13,5 | 132,0 |
| A-voraz | 13,8 | 22,6 | 0,5 | 1,0 | 3,5 | 15,0 | 132,0 |
| M-PLE | 14,2 | 21,4 | 0,5 | 1,5 | 5,5 | 17,5 | 123,0 |
| M-voraz | 15,7 | 23,1 | 0,5 | 1,5 | 6,0 | 19,0 | 124,5 |
| M-ref | 18,5 | 26,3 | 0,5 | 1,5 | 7,5 | 20,0 | 143,0 |

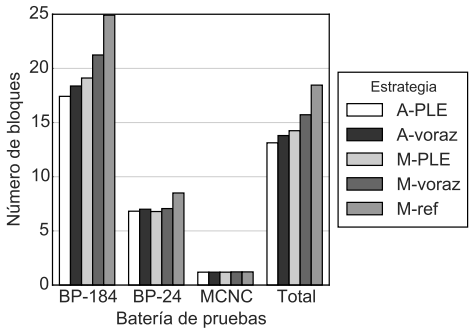



**Figura 5.20:** Número medio de bloques usados por las implementaciones FSMIM-T en las pruebas de implementación con optimización en área.

bloques en estos casos. El número de FSMIM-T que ocupan 1 bloque en las baterías de pruebas MCNC y BP-24 es aproximadamente del 43 % y del 22 %, respectivamente. Sin embargo, todas las reducciones genéricas obtenidas en las pruebas independientes del dispositivo fueron menores al 50 % en las baterías de pruebas MCNC y BP-24. Esto implica que no está garantizada la reducción de bloques en estos casos: algunas de estas reducciones genéricas darán lugar a un 0 % de reducción de bloques y otras, a un 50 %. El motivo es que, en las pruebas independientes del dispositivo, las FSMIM-T se generan sin ajuste de profundidad; por lo tanto, uno de los dos bloques de 9 Kbits usados para implementar la ROM en estos casos puede no estar ocupado completamente. Esto permite que reducciones genéricas inferiores al 50 % puedan ser suficientes para eliminar uno de los bloques.

El análisis realizado muestra que las nuevas estrategias disponen de un estrecho margen de mejora en las baterías de pruebas MCNC y BP-24. En la batería de pruebas MCNC, sólo el 19 % de las FSMIM-T ocupan más

de un bloque. Se trata de cuatro casos que ocupan entre 1,5 y 4,5 bloques. Por lo tanto, en el caso de MCNC, se podría decir que el margen de mejora es prácticamente inexistente. En la batería de pruebas BP-24, el porcentaje de casos con más de un bloque sube hasta el 36 %; seis de estas FSMIM-T ocupan entre 8 y 100 bloques.

Respecto a las reducciones del número de bloques, la tabla 5.30 y la figura 5.21 muestran el resumen estadístico de la reducción RPC del consumo de bloques obtenida por las nuevas estrategias respecto a M-ref. Las reducciones medias obtenidas con el conjunto completo de FSM han sido mayores que las reducciones genéricas, especialmente en las estrategias de tipo AM, que han obtenido reducciones medias cinco puntos superiores a las reducciones genéricas (27,8 % para estrategia A-PLE y 24,2 % para estrategia A-voraz). Las nuevas estrategias de tipo MA han obtenido reducciones del 17,1 % (estrategia M-PLE) y del 11,8 % (estrategia M-voraz). En parte, este incremento se debe a casos en los que la discretización de la reducción genérica ha beneficiado a las reducciones obtenidas. Un buen ejemplo de este efecto se puede observar en los casos de BP-24 en los que las reducciones genéricas menores del 50 % han dado lugar a reducciones de bloques del 50 % (compárense los valores máximos de la estrategia BP-24 de la tabla 5.30 con los que aparecen en la tabla tabla 5.27). En la batería de pruebas BP-184, se puede apreciar también un ligero aumento de las magnitudes de las reducciones que se encuentran por encima del tercer cuartil.

Comparados con las reducciones obtenidas en pruebas independientes del dispositivo, los resultados parecen indicar que el uso de bloques ha afectado más al porcentaje de éxito en la reducción del tamaño de la FSMIM-T que a los valores medios. La batería de pruebas que se ha visto más afectada por el uso de bloques ha sido MCNC. La estrategia M-voraz no ha conseguido reducir el número de bloques de M-ref, y el resto de estrategias lo ha conseguido tan sólo en un caso (consiguiendo una reducción del 20 % de los bloques). Se trata de una de las FSMIM-T que usaban más de 1 bloque.

En la batería de pruebas BP-24, las nuevas estrategias ha reducido su

**Tabla 5.29:** Resumen estadístico del número de bloques usados por las implementaciones de las FSMIM-T generadas con la estrategia M-ref en las pruebas de implementación con optimización en área.

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 24,9 | 27,7 | 1,0 | 6,2 | 15,0 | 34,5 | 143,0 |
| BP-24 | 8,5 | 22,0 | 0,5 | 0,5 | 1,0 | 3,5 | 100,0 |
| MCNC | 1,2 | 1,1 | 0,5 | 0,5 | 1,0 | 1,0 | 4,5 |

porcentaje de éxito a la mitad aproximadamente, obteniendo valores del 27 % para todas las estrategias (obsérvese que estos porcentajes son cercanos al porcentaje de las FSMIM-T generadas con M-ref que ocupaban más de 1 bloque en esta batería de pruebas). Sin embargo, a pesar de esta disminución del porcentaje de éxito, las reducciones medias obtenidas para los casos de éxito tienen una magnitud similar. No se ha producido ningún caso en el que las estrategias hayan aumentado el consumo de bloques.

Como cabía esperar, la batería de pruebas BP-184 es la menos afectada, alcanzando reducciones medias y porcentajes de éxito de orden similar a los obtenidos en las pruebas independientes del dispositivo. Se han conseguido reducciones entre el 16 % y el 23 % en las estrategias de tipo MA, y entre el 34 % y el 39 % en las estrategias de tipo AM. Los porcentajes de éxito han sido superiores al 76 % en las estrategias de tipo MA y al 92 % en las estrategias de tipo AM. Por el contrario, los porcentajes de fracaso han sido muy pequeños: entre el 0 % y el 7 %, según la estrategia.

Las similitudes con la reducción genérica mencionadas también se aprecian en las distribuciones de los valores obtenidos por las nuevas estrategias para esta batería de pruebas (véase la figura 5.21a y la figura 5.13a).

**Tabla 5.30:** Resumen estadístico de la reducción RPC del número de bloques usados por las implementaciones de las FSMIM-T generadas por las nuevas estrategias respecto a M-ref obtenida en las pruebas de implementación con optimización en área ( %).

| Bat. de pruebas | Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | A-PLE | 39,2 | 24,4 | −20,8 | 20,6 | 44,4 | 59,7 | 80,6 |
| | A-voraz | 34,3 | 24,6 | −20,8 | 11,8 | 33,3 | 53,0 | 80,6 |
| | M-PLE | 23,0 | 17,7 | 0,0 | 8,0 | 24,1 | 33,3 | 63,9 |
| | M-voraz | 15,8 | 16,0 | −38,9 | 2,6 | 15,0 | 23,0 | 57,6 |
| BP-24 | A-PLE | 8,0 | 15,3 | 0,0 | 0,0 | 0,0 | 1,3 | 50,0 |
| | A-voraz | 6,5 | 13,8 | 0,0 | 0,0 | 0,0 | 2,6 | 50,0 |
| | M-PLE | 8,2 | 15,5 | 0,0 | 0,0 | 0,0 | 1,3 | 50,0 |
| | M-voraz | 6,4 | 13,9 | 0,0 | 0,0 | 0,0 | 2,6 | 50,0 |
| MCNC | A-PLE | 1,0 | 4,4 | 0,0 | 0,0 | 0,0 | 0,0 | 20,0 |
| | A-voraz | 1,0 | 4,4 | 0,0 | 0,0 | 0,0 | 0,0 | 20,0 |
| | M-PLE | 1,0 | 4,4 | 0,0 | 0,0 | 0,0 | 0,0 | 20,0 |
| | M-voraz | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| Total | A-PLE | 27,8 | 26,6 | −20,8 | 0,0 | 23,1 | 50,0 | 80,6 |
| | A-voraz | 24,2 | 25,4 | −20,8 | 0,0 | 16,7 | 48,6 | 80,6 |
| | M-PLE | 17,1 | 18,2 | 0,0 | 0,0 | 11,1 | 28,1 | 63,9 |
| | M-voraz | 11,8 | 15,6 | −38,9 | 0,0 | 8,3 | 19,0 | 57,6 |

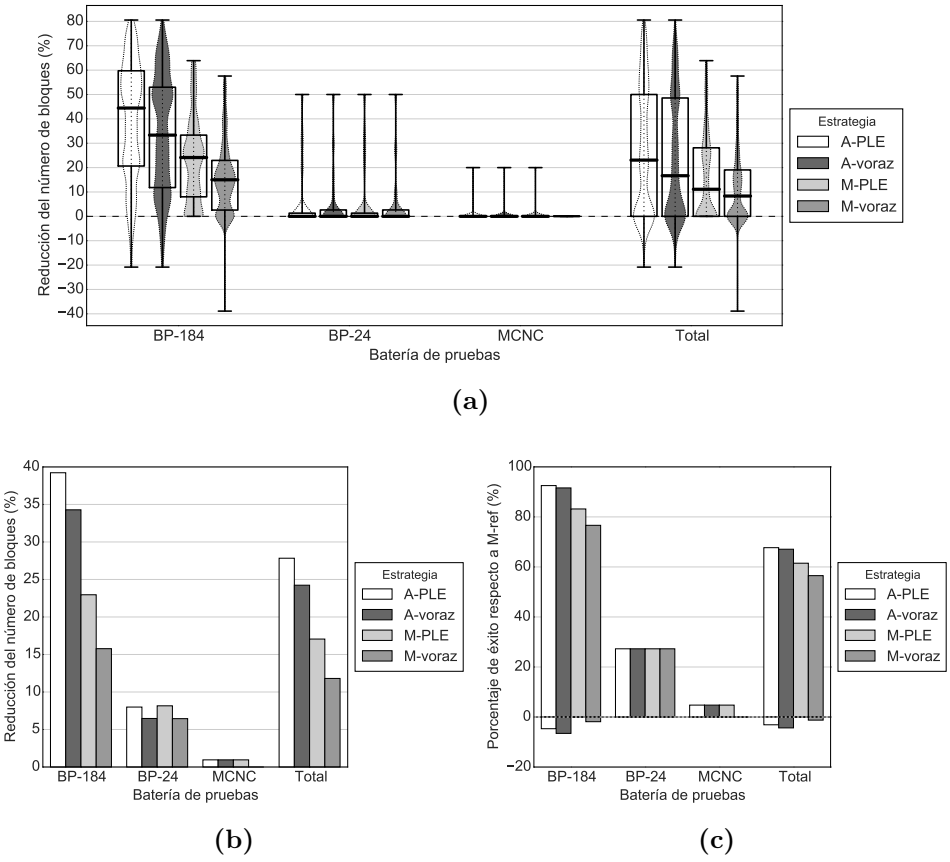



**Figura 5.21:** Reducción RPC del número de bloques usados por las implementaciones de las FSMIM-T generadas por las nuevas estrategias respecto a M-ref obtenida en las pruebas de implementación con optimización en área: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

### Estudio del consumo de LUT

En la arquitectura utilizada en esta tesis para las implementaciones de ROM, cuando la profundidad de la ROM es mayor que la profundidad máxima de los bloques de memoria del dispositivo, se requiere un banco de multiplexores para interconectar los bloques [Senhadji-Navarro et al., 2012], denominado banco de multiplexores de la ROM (véase la sección 1.2.4 y el apéndice G). La complejidad del banco de multiplexores de la ROM depende la geometría de la ROM: el número de entradas del banco viene determinado por la proporción entre la profundidad de la ROM y la profundidad máxima de los bloques, y el número de salidas, por el ancho de la ROM. Puesto que los

multiplexores se implementan con LUT, el consumo de LUT de una implementación FSMIM depende tanto de la complejidad del banco de selectores de entrada como de la geometría de la ROM.

El coste de selección de entradas forma parte del objetivo de optimización de los algoritmos utilizados en las fases de minimización de selección de entradas. En el caso de la arquitectura FSMIM-T, esta optimización tiene un doble objetivo: reducir la complejidad del banco de selectores de entrada y reducir el ancho de la ROM. Por lo tanto, en los casos en los que la profundidad de la ROM es mayor que la profundidad máxima de los bloques de memoria, la reducción del coste de selección de entradas tiene una doble influencia en la reducción del consumo de LUT, ya que, además de reducir el número de LUT necesarias para implementar el banco de selectores de entradas, permite reducir el número de LUT requeridas para implementar el banco de multiplexores de la ROM.

Sin embargo, la fase de agrupación de estados tiene dos efectos de signo contrario en el consumo de LUT. Por una parte, favorece el incremento del coste de selección de entradas, lo que contrarresta la efectividad de la fase de minimización de selección de entradas, tal como se ha mencionado anteriormente. Por otra parte, la reducción de la profundidad de una FSMIM permite disminuir la complejidad del banco de multiplexores de la ROM; de hecho, este desaparece si la agrupación de estados consigue reducir la profundidad por debajo de la profundidad máxima de los bloques de memoria.

Los resultados de las pruebas de implementación permiten estudiar la bondad del coste de selección como estimación del consumo de LUT. La figura 5.22 muestra el consumo de LUT de las implementaciones de las FSMIM-T frente al coste de selección de entradas. Para facilitar la interpretación de los datos, los puntos correspondientes a los casos en los que la ROM tiene una profundidad mayor que 16 Kpalabras (la profundidad máxima de los bloques de memoria del dispositivo FPGA utilizado) se han marcado con un triángulo.

La nube de puntos correspondientes a los casos con profundidad menor que 16 Kpalabras muestra una clara relación lineal entre el coste y el número de LUT. La dispersión de los puntos respecto a dicha relación lineal es especialmente pequeña en las estrategias de tipo MA y algo mayor en las de tipo AM. Estos datos parecen mostrar que el coste de selección de entradas es una buena estimación del número de LUT cuando no se requieren bancos de multiplexores de la ROM, aunque todavía queda margen de mejora.

Respecto a los casos con profundidad mayor que 16 Kpalabras, aunque se aprecia cierta tendencia en el incremento del consumo de LUT con el aumento del coste de selección de entradas, la nube de puntos muestra un

comportamiento más disperso. El uso del coste de selección de entradas como estimación del número de LUT es claramente mejorable en estos casos.

Durante la generación de las FSMIM-T se puede conocer o estimar el número de entradas que tendrán los multiplexores del banco de selectores de entradas y los del banco de multiplexores de la ROM. Por otra parte, conociendo la arquitectura del dispositivo FPGA en el que se va a realizar la implementación, es posible estimar el número de LUT requerido por cada multiplexor en función de su número de entradas. El resultado de las fases de minimización de selección de entradas puede ser mejorado si se incorpora al objetivo de optimización una función de coste que estime el número de LUT de los multiplexores a partir del número de entradas. Este estudio formará parte de los trabajos futuros.

Respecto al consumo de LUT de las implementaciones FSMIM-T, la tabla 5.31 muestra un resumen estadístico de los resultados obtenidos por las diferentes estrategias con el conjunto completo de FSM. En promedio, los mejores resultados se obtienen con las nuevas estrategias de tipo MA, que requieren el menor número de LUT. Las estrategias de tipo AM obtienen mejores resultados que la estrategia M-ref a pesar de que, en las pruebas independientes del dispositivo, la estrategia A-voraz presentaba un coste medio de selección de entradas mayor que la estrategia M-ref, y la estrategia A-PLE, un coste similar. Esta mejora en los resultados de las estrategias de tipo AM respecto a las pruebas independientes del dispositivo se debe a la profundidad de las FSMIM-T: en promedio, las estrategias de tipo AM son las que generan las FSMIM-T con menor profundidad, lo que favorece la reducción del número de LUT necesarias para implementar el banco de multiplexores de la ROM.

Por esta misma razón, tal como muestra la figura 5.23, el número medio de LUT de la batería de pruebas BP-24 es mayor que el de MCNC a pesar de que presentaba un coste medio de selección de entradas menor: con la estrategia M-ref, por ejemplo, mientras que el 10 % de las FSMIM-T generadas con la batería de pruebas BP-24 tiene una profundidad mayor que 16 Kpalabras, no se da ningún caso en MCNC.

La tabla 5.32 muestra un resumen estadístico del número de LUT usadas por las implementaciones de las FSMIM-T generadas con la estrategia M-ref. Tal como puede observarse, en términos generales, el número de LUT ocupado por las FSMIM-T es pequeño: el 75 % de los casos no supera las 16 LUT. Esta cantidad es muy poco significativa comparada con el número de LUT que ocupan las implementaciones ISE-LUT. De hecho, en el 92 % de los casos, el número de LUT de la FSMIM-T representa menos del 10 % del número de LUT de la ISE-LUT.

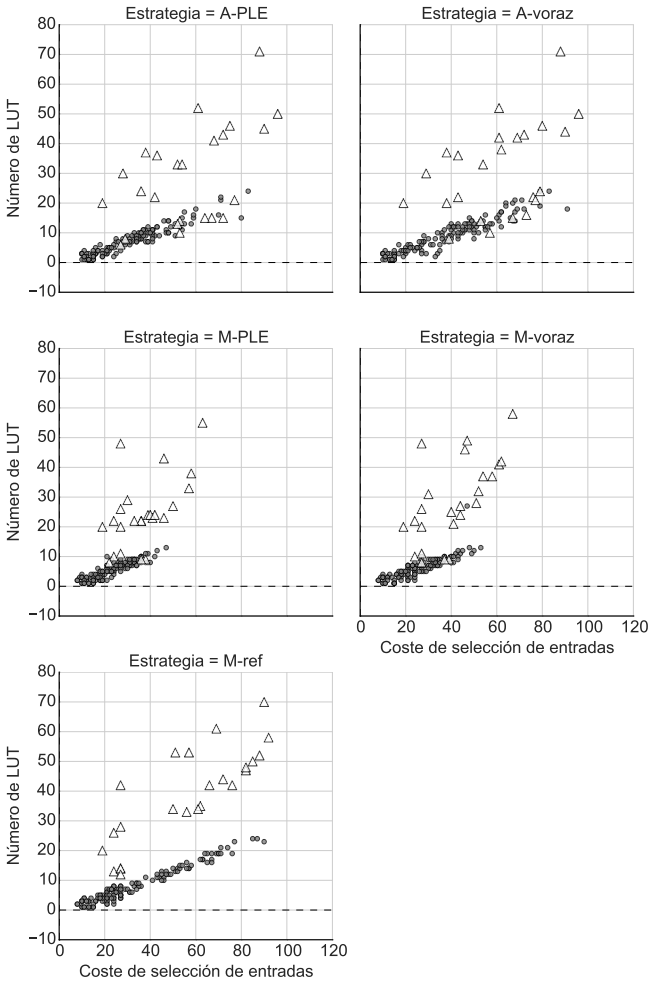



**Figura 5.22:** Número de LUT usadas por las implementaciones FSMIM-T en las pruebas de implementación con optimización en área frente al coste de selección de entradas. Los puntos marcados con un triángulo corresponden a los casos en los que la ROM de la FSMIM-T tiene una profundidad mayor de 16 Kpalabras.

La figura 5.24 representa un resumen estadístico de la reducción RPC del número de LUT usadas por las implementaciones de las FSMIM-T generadas por las nuevas estrategias respecto a M-ref. Respecto al conjunto completo de FSM, los mejores resultados se obtienen para las estrategias de tipo MA, en especial para la estrategia M-PLE, que presenta la mejor reducción media (20 %), el mayor porcentaje de éxito (63 %) y el menor porcentaje de fracaso

**Tabla 5.31:** Resumen estadístico del número de LUT usadas por las implementaciones FSMIM-T obtenidas con el conjunto completo de FSM en las pruebas de implementación con optimización en área. Las estrategias están ordenadas de menor a mayor número medio de LUT.

| Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| M-PLE | 8,6 | 8,5 | 1,0 | 4,0 | 7,0 | 9,0 | 55,0 |
| M-voraz | 9,5 | 10,1 | 1,0 | 4,0 | 7,0 | 9,0 | 58,0 |
| A-PLE | 10,8 | 10,8 | 1,0 | 4,0 | 8,0 | 12,0 | 71,0 |
| A-voraz | 12,4 | 11,2 | 1,0 | 5,0 | 10,0 | 14,0 | 71,0 |
| M-ref | 12,8 | 13,6 | 1,0 | 4,0 | 8,0 | 16,0 | 70,0 |

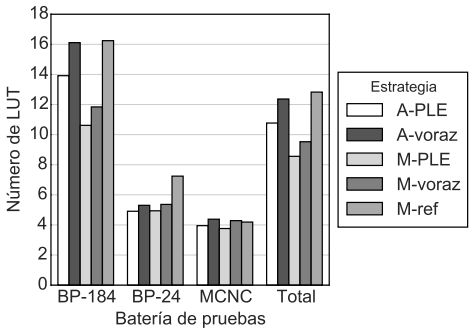



**Figura 5.23:** Número medio de LUT usadas por las implementaciones FSMIM-T en las pruebas de implementación con optimización en área.

**Tabla 5.32:** Resumen estadístico del número de LUT usadas por las implementaciones de las FSMIM-T generadas con la estrategia M-ref en las pruebas de implementación con optimización en área.

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 16,2 | 14,6 | 2,0 | 6,0 | 12,0 | 19,0 | 70,0 |
| BP-24 | 7,2 | 9,9 | 1,0 | 3,0 | 4,0 | 7,0 | 53,0 |
| MCNC | 4,2 | 3,0 | 1,0 | 2,0 | 4,0 | 6,0 | 12,0 |

(12 %). El comportamiento de las diferentes estrategias es muy desigual en las distintas baterías de pruebas. La que se comporta de forma más uniforme es la estrategia M-PLE, que siempre es la mejor estrategia en promedio. Por el contrario, las estrategias de tipo voraz presentan reducciones negativas en algunas de las baterías de pruebas.

Respecto al estudio del número de LUT ahorradas, la tabla 5.33 muestra un resumen estadístico del NLAB obtenido por las implementaciones de

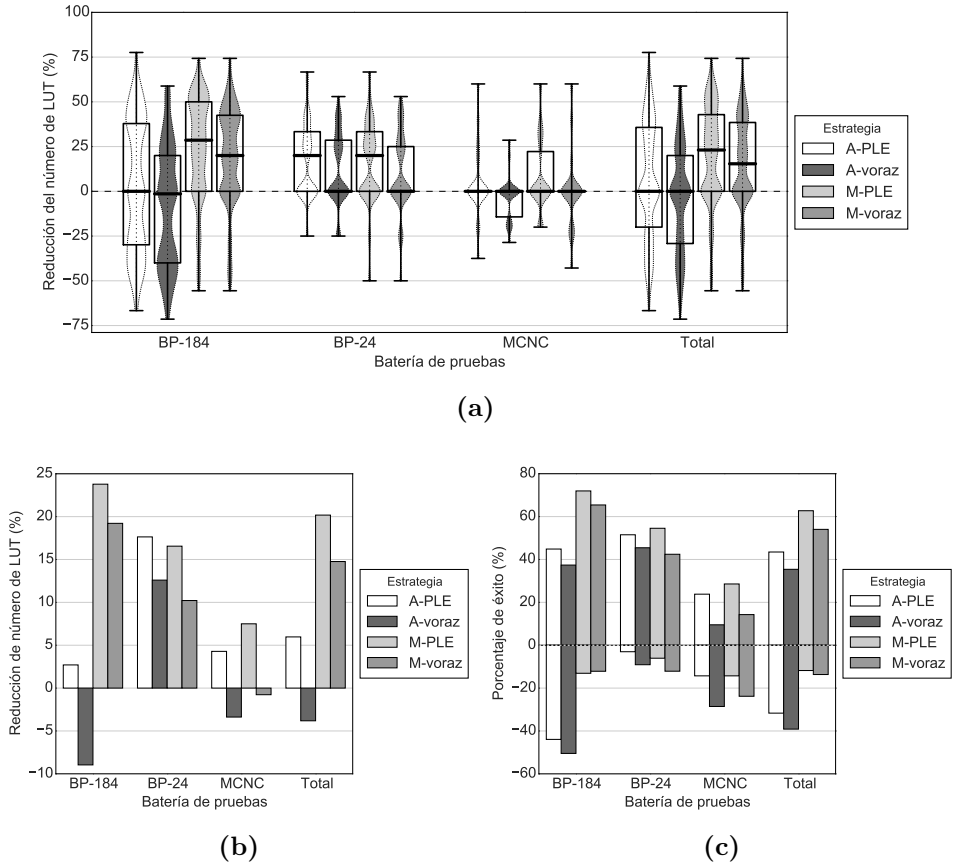



**Figura 5.24:** Reducción RPC del número de LUT usadas por las implementaciones de las FSMIM-T generadas por las nuevas estrategias respecto a M-ref obtenida en las pruebas de implementación con optimización en área: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

las FSMIM-T generadas con el conjunto completo de FSM. En promedio, ordenadas de mayor a menor NLAB, las primeras estrategias son las de tipo AM, que obtienen los mejores resultados debido a que son las que generan las FSMIM-T que requieren el menor número de bloques. Les siguen las nuevas estrategias de tipo MA, quedando en último lugar la estrategia M-ref. Dentro de cada uno de los tipos anteriores, las estrategias basadas en PLE obtienen mejores resultados que las basadas en algoritmos voraces.

Los resultados obtenidos con las diferentes baterías de pruebas se muestran en la tabla 5.34 (que contiene un resumen estadístico del NLAB obtenido por la estrategia M-ref) y en la figura 5.25 (que representa los valores

medios obtenidos por las diferentes estrategias). Como puede observarse, los mayores valores medios y las mayores diferencias entre estrategias se obtienen para la batería BP-184, con valores medios que oscilan entre las 197 LUT/bloque de M-ref y las 307 LUT/bloque de A-PLE. Los valores medios y diferencias más pequeñas se dan en la batería de pruebas MCNC, con valores medios cercanos a las 56,3 LUT/bloque de M-ref.

Tal como se indicó al principio de este apartado, el número de LUT que ocupan las implementaciones FSMIM-T es poco significativo comparado con

**Tabla 5.33:** Resumen estadístico del NLAB obtenido por las implementaciones de las FSMIM-T generadas con el conjunto completo de FSM en las pruebas de implementación con optimización en área.

| Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| A-PLE | 235,0 | 162,9 | 12,0 | 101,2 | 198,4 | 347,5 | 689,8 |
| A-voraz | 218,0 | 152,1 | 12,0 | 101,2 | 176,3 | 326,2 | 611,2 |
| M-PLE | 199,8 | 160,0 | 12,0 | 77,6 | 133,9 | 298,4 | 603,5 |
| M-voraz | 183,4 | 147,3 | 12,0 | 70,2 | 123,8 | 266,7 | 603,5 |
| M-ref | 160,8 | 136,5 | 10,0 | 59,0 | 109,4 | 241,2 | 544,7 |

**Tabla 5.34:** Resumen estadístico del NLAB obtenido por las implementaciones de las FSMIM-T generadas por la estrategia M-ref en las pruebas de implementación con optimización en área.

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 197,3 | 141,7 | 37,3 | 82,1 | 140,7 | 293,3 | 544,7 |
| BP-24 | 112,3 | 111,0 | 10,0 | 26,2 | 72,0 | 146,0 | 427,3 |
| MCNC | 56,3 | 30,8 | 17,0 | 30,0 | 51,0 | 72,0 | 119,0 |

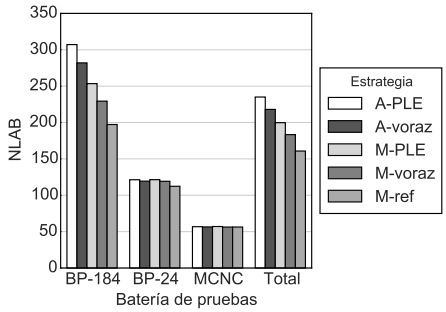



**Figura 5.25:** NLAB medio obtenido por las implementaciones FSMIM-T en las pruebas de implementación con optimización en área.

el número de LUT que ocupan las implementaciones ISE-LUT. Esto significa que el número de bloques tendrá más peso que el número de LUT en el incremento del NLAB obtenido por las nuevas estrategias respecto a M-ref. La tabla 5.35 muestra el incremento RPC del NLAB de las implementaciones FSMIM-T generadas por las nuevas estrategias respecto a M-ref, y la figura 5.26, una representación de dicho resumen. Si se comparan los incrementos medios (véase la figura 5.26b) con la reducción media del número de bloques (véase la figura 5.21b), se observa un comportamiento similar desde el punto de vista cualitativo.

Las estrategias de tipo AM son las que obtienen los mejores resultados. El incremento medio obtenido por la estrategia A-PLE es del 68,8 %, con un porcentaje de éxito del 76 % (en el 25 % de los casos se alcanzan incrementos entre el 105,7 % y el 410,8 %). Por su parte, los resultados de la estrategia A-voraz son del mismo orden aunque ligeramente inferiores. Respecto a las baterías de pruebas, los peores resultados se obtienen con MCNC, en la que la estrategia A-PLE presenta un porcentaje de éxito del 24 % (con un incremento medio del 9 % para los casos de éxito) y un porcentaje de fracaso del 14 % (sin embargo, el incremento medio en los casos de fracaso es tan solo del −3 %). El porcentaje de éxito asciende al 58 % en la batería de pruebas BP-24 (con un incremento medio del 26 % para los casos de éxito) y al 92 % en BP-184 (con un incremento medio del 108 % para los casos de éxito). En ambas baterías de pruebas, el porcentaje de fracaso es inferior al 7 % y el peor incremento medio obtenido para los casos de fracaso es del −8 %. Los valores obtenidos por A-voraz para estas dos últimas baterías de pruebas son ligeramente inferiores.

Las estrategias de tipo MA obtienen peores resultados que las de tipo AM. El incremento medio obtenido por la estrategia M-PLE es del 29,6 %, con un porcentaje de éxito del 73 % y con incrementos entre el 40,6 % y el 178,4 % en la cuarta parte de los casos. Respecto a las baterías de pruebas, los peores resultados se obtienen para MCNC, en la que M-PLE alcanza un porcentaje de éxito del 29 % (con un incremento medio del 8 % para los casos de éxito) y un porcentaje de fracaso del 14 % (pero con un incremento medio en los casos de fracaso de tan solo −1 %). En la batería de pruebas BP-24, el porcentaje de éxito es del 61 % (alcanzando un incremento medio del 25 % en estos casos). El porcentaje de éxito mayor se presenta en la batería de pruebas BP-184, donde el 86 % de los casos consigue mejorar el NLAB (con un incremento medio del 46 % para los casos de éxito). El porcentaje de fracaso no supera el 6 % en estas dos últimas baterías de pruebas. Aunque los resultados son inferiores en el caso de la estrategia M-voraz (especialmente en la batería de pruebas MCNC) esta estrategia consigue un incremento

medio del 18,3 % en el conjunto completo de FSM y un porcentaje de éxito del 64 %.

### Estudio de la frecuencia de operación máxima

Los valores de la frecuencia de operación máxima obtenidos en las pruebas de implementación con optimización en área han sido similares a los obtenidos en las pruebas de optimización en velocidad (que se analizarán en la sección 5.4.2.2). En promedio, las diferencias relativas entre las frecuencias obtenidas por todas las estrategias con ambos tipos de objetivos de optimización ha sido del 4,5 %. En el 93 % de los casos, las diferencias relativas han sido inferiores al 10 %.

Debido a que la frecuencia de operación máxima es un parámetro de menor importancia cuando se realizan implementaciones optimizadas en área y a que el objetivo de optimización ha tenido una escasa influencia en los resultados, se ha decidido realizar en este apartado un estudio somero de las frecuencias de operación máximas obtenidas. Será en la sección dedicada a los resultados con optimización en velocidad donde se realizará un estudio más detallado.

**Tabla 5.35:** Resumen estadístico del incremento RPC del NLAB obtenido por las implementaciones de las FSMIM-T generadas por las nuevas estrategias respecto a M-ref en las pruebas de implementación con optimización en área ( %).

| Bat. de pruebas | Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | A-PLE | 99,5 | 91,0 | −26,3 | 28,2 | 82,9 | 147,7 | 410,8 |
| | A-voraz | 80,5 | 80,7 | −26,3 | 16,8 | 54,8 | 112,9 | 410,4 |
| | M-PLE | 39,7 | 41,3 | −0,1 | 8,9 | 32,7 | 52,6 | 178,4 |
| | M-voraz | 24,1 | 30,2 | −63,2 | 2,4 | 18,6 | 31,5 | 136,1 |
| BP-24 | A-PLE | 14,9 | 27,5 | −1,8 | 0,0 | 1,3 | 20,0 | 101,3 |
| | A-voraz | 12,0 | 25,8 | −3,4 | 0,0 | 0,2 | 9,5 | 101,3 |
| | M-PLE | 15,2 | 27,7 | −1,8 | 0,0 | 1,3 | 20,0 | 101,3 |
| | M-voraz | 11,9 | 25,9 | −3,4 | 0,0 | 0,0 | 9,5 | 101,3 |
| MCNC | A-PLE | 1,7 | 6,7 | −6,2 | 0,0 | 0,0 | 0,0 | 29,2 |
| | A-voraz | 1,0 | 5,7 | −4,1 | −0,9 | 0,0 | 0,0 | 25,0 |
| | M-PLE | 2,1 | 6,5 | −1,3 | 0,0 | 0,0 | 2,3 | 29,2 |
| | M-voraz | −0,1 | 1,7 | −4,1 | 0,0 | 0,0 | 0,0 | 4,5 |
| Total | A-PLE | 68,8 | 86,2 | −26,3 | 1,0 | 32,6 | 105,7 | 410,8 |
| | A-voraz | 55,6 | 75,0 | −26,3 | 0,0 | 22,4 | 94,2 | 410,4 |
| | M-PLE | 29,6 | 38,6 | −1,8 | 0,0 | 14,8 | 40,6 | 178,4 |
| | M-voraz | 18,3 | 28,5 | −63,2 | 0,0 | 10,1 | 24,1 | 136,1 |

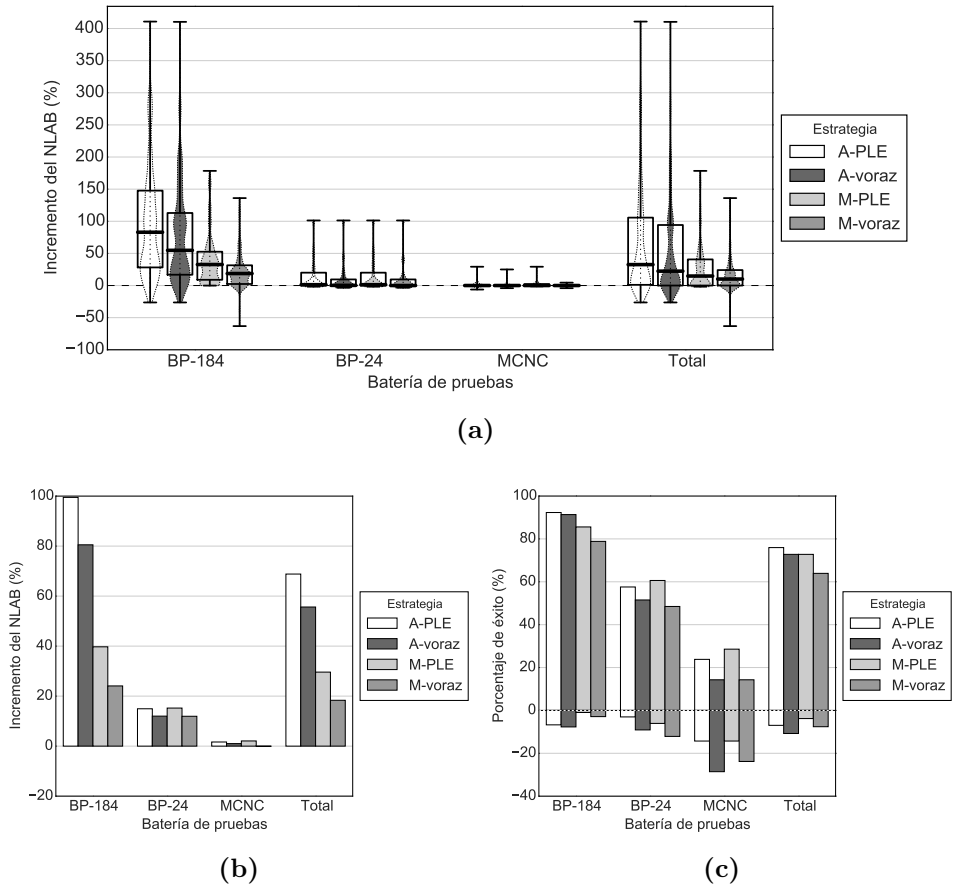



**Figura 5.26:** Incremento RPC del NLAB obtenido por las implementaciones de las FSMIM-T generadas por las nuevas estrategias respeto a M-ref en las pruebas de implementación con optimización en área: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

Se ha calculado el incremento RPC de la frecuencia de operación máxima conseguido por las nuevas estrategias respecto a M-ref. La figura 5.27 muestra un resumen estadístico de los valores obtenidos. Aunque los incrementos medios son modestos (figura 5.27b), en algunos casos se consiguen incrementos significativos. Los mejores resultados se obtienen con las estrategias de tipo PLE, alcanzándose incrementos medios del 10 % (estrategia A-PLE) y del 8 % (estrategia M-PLE) aproximadamente.

En la batería de pruebas BP-184, el incremento medio obtenido por la estrategia A-PLE es del 14 %, consiguiendo incrementos entre el 20 % y el

57 % en el 34 % de los casos. La estrategia M-PLE alcanza un incremento medio del 11 % en esta batería de pruebas, con incrementos entre el 20 % y el 41 % en el 19 % de los casos. Por otra parte, en la batería de pruebas BP-24, la estrategia A-PLE consigue incrementos entre el 10 % y el 27 % en el 21 % de los casos; y la estrategia M-PLE, incrementos entre el 10 % y el 24 % en el 15 % de los casos.

El porcentaje de éxito en el incremento de frecuencia es superior al 67 % para todas las estrategias en el conjunto completo de FSM mientras que el porcentaje de fracaso es inferior al 28 %. Además, las estrategias de tipo PLE alcanzan incrementos medios para los casos de éxito superiores al 13 %; mientras que el peor incremento medio que obtienen para los casos de fracaso es del −4 %.

Con el fin de estudiar como afectan a la frecuencia de operación máxima las mejoras conseguidas por las nuevas estrategias respecto al NLAB y al consumo de bloques, se ha calculado, por una parte, el porcentaje de éxito neto del incremento RPC de la frecuencia de operación máxima y de la reducción RPC del número de bloques usados (figura 5.28a); y, por otra parte, el porcentaje de éxito neto del incremento RPC de la frecuencia de operación máxima y del incremento RPC del NLAB (figura 5.28b). Como se puede observar, ambos porcentajes de éxito neto presentan valores similares.

Respecto al porcentaje de éxito neto en frecuencia y uso de bloques, los valores obtenidos con el conjunto completo de FSM se encuentran entre el 68 % (de la estrategia A-voraz) y el 78 % (de la estrategia A-PLE). El porcentaje de fracaso neto es inferior o igual al 17 % en todas las estrategias.

Respecto al porcentaje de éxito neto en frecuencia y NLAB, los valores obtenidos con el conjunto completo de FSM se encuentran entre el 64 % (de la estrategia A-voraz) y el 75 % (de la estrategia A-PLE). El porcentaje de fracaso neto es inferior al 16 % en todas las estrategias.

#### 5.4.2.2. Análisis de las implementaciones optimizadas en velocidad

En esta sección se estudia la frecuencia de operación máxima y el consumo de LUT de las FSMIM-T implementadas con optimización en velocidad. El consumo de bloques no será estudiado debido a que los resultados han sido idénticos a los de las pruebas con optimización en área.

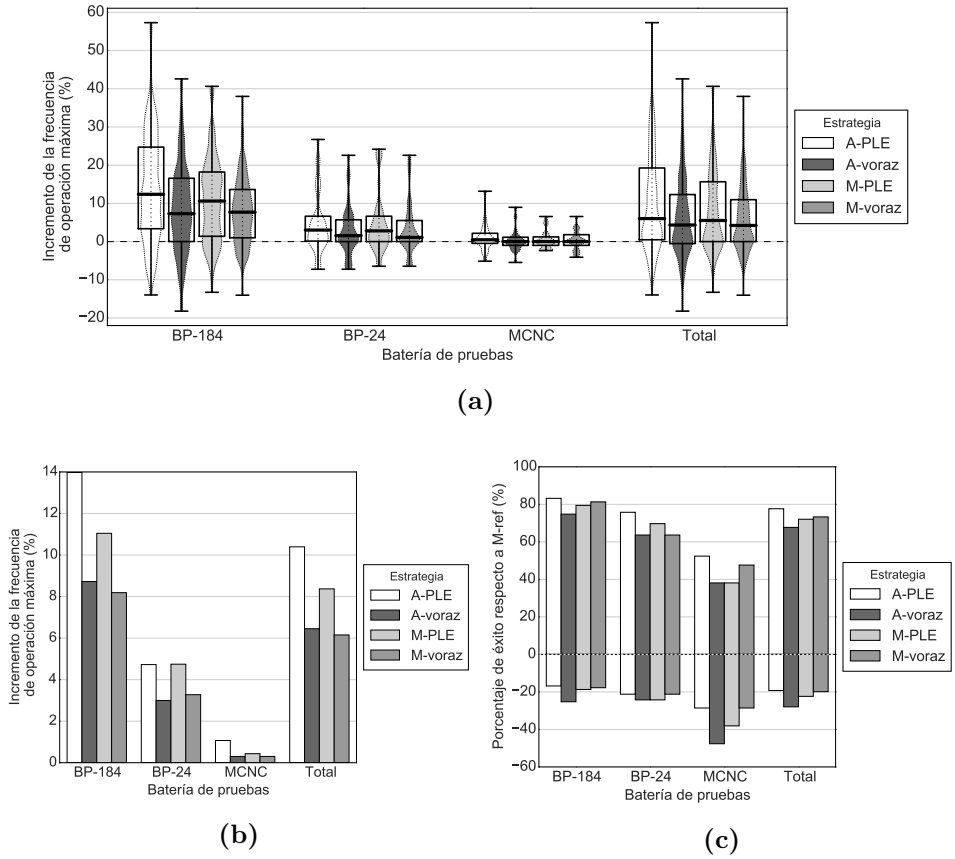
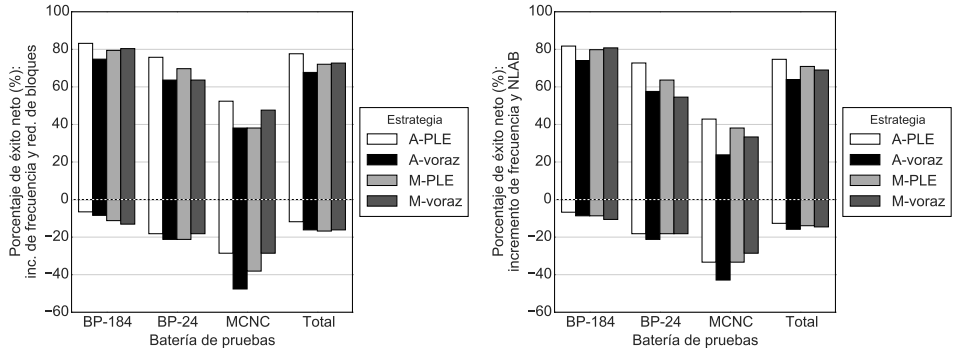



**Figura 5.27:** Incremento RPC de velocidad obtenido por las implementaciones de las FSMIM-T generadas por las nuevas estrategias respecto a M-ref en las pruebas de implementación con optimización en área: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

### Estudio de la frecuencia de operación máxima

La tabla 5.36 muestra un resumen estadístico de las frecuencias de operación máximas obtenidas con el conjunto completo de FSM. Como puede observarse, para todas las estrategias, los valores obtenidos se encuentran aproximadamente entre los 65 MHz y los 320 MHz. Las diferencias entre los valores medios son poco significativas. En promedio, las estrategias de tipo PLE son las que obtienen mejores resultados, seguidas por las de tipo voraz. Además, para cada uno de los tipos anteriores, la estrategia de tipo AM correspondiente obtiene mejores resultados que la de tipo MA. Los peores resultados se obtienen para la estrategia M-ref.




**(a)** **(b)**

**Figura 5.28:** Porcentaje de éxito neto en la mejora de velocidad y de otros parámetros de las implementaciones de las FSMIM-T obtenido por las nuevas estrategias respecto a M-ref en las pruebas de implementación con optimización en área: (a) incremento RPC de velocidad y reducción RPC del número de bloques, y (b) incremento RPC de velocidad e incremento RPC del NLAB.

El análisis de los valores obtenidos en cada batería de pruebas indica que las mayores frecuencias se alcanzan en la batería de pruebas MCNC y las menores en BP-184 (véase la figura 5.29, que muestra los valores medios obtenidos para cada batería de pruebas, y la tabla 5.37, que muestra los valores obtenidos con la estrategia M-ref en las distintas baterías de pruebas).

Aunque las diferencias entre las frecuencias medias es pequeña, las nuevas estrategias consiguen mejoras de velocidad en un porcentaje importante de los casos, alcanzándose incrementos significativos para algunas FSMIM-T. La tabla 5.38 y la figura 5.30 muestran un resumen estadístico del incremento RPC de velocidad obtenido por las implementaciones de las FSMIM-T generadas con las nuevas estrategias respecto a M-ref.

En el conjunto completo de FSM, las nuevas estrategias obtienen incrementos medios mayores o iguales al 5 %, con un porcentaje de éxito entre el 70 % y 78 %, y con un porcentaje de fracaso que no supera el 22 %. Los incrementos medios obtenidos para los casos de éxito son mayores que el 8 %, mientras que el peor resultado para los casos de fracaso corresponde a un incremento medio del −4,2 %.

Las estrategias que obtienen los mejores resultados son las de tipo PLE, con incrementos medios del 10,5 % (estrategia A-PLE) y del 8,2 % (estrategia M-PLE). La estrategia A-PLE consigue incrementos superiores al 10 % en el 45 % de los casos y superiores al 16 % en el 25 % de los casos, alcanzando un valor máximo del 55,2 %. Por su parte, la estrategia M-PLE obtiene

**Tabla 5.36:** Resumen estadístico de la frecuencia de operación máxima (MHz) obtenida por las implementaciones FSMIM-T con el conjunto completo de FSM en las pruebas de implementación con optimización en velocidad. Las estrategias están ordenadas de mayor a menor frecuencia.

| Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| A-PLE | 214,7 | 63,3 | 74,5 | 170,0 | 220,5 | 260,5 | 318,8 |
| M-PLE | 210,5 | 63,4 | 71,2 | 160,8 | 209,0 | 261,9 | 318,8 |
| A-voraz | 208,9 | 64,2 | 74,5 | 158,3 | 216,4 | 256,0 | 318,8 |
| M-voraz | 205,8 | 65,3 | 71,2 | 157,9 | 210,3 | 255,4 | 318,8 |
| M-ref | 197,8 | 67,2 | 65,9 | 149,6 | 193,2 | 255,0 | 320,1 |

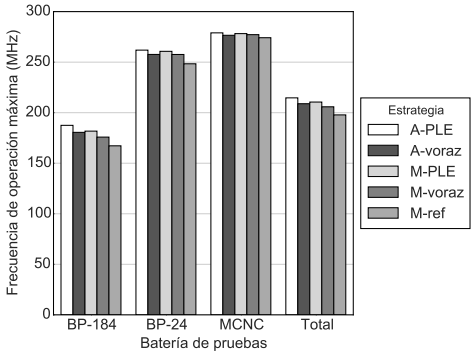



**Figura 5.29:** Valores medios de la frecuencia de operación máxima obtenida por las implementaciones FSMIM-T en las pruebas de implementación con optimización en velocidad.

**Tabla 5.37:** Resumen estadístico de la frecuencia de operación máxima (MHz) obtenida por las implementaciones FSMIM-T generadas con la estrategia M-ref en las pruebas de implementación con optimización en velocidad.

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 167,2 | 50,3 | 65,9 | 123,1 | 165,4 | 206,5 | 285,1 |
| BP-24 | 248,4 | 63,1 | 88,2 | 223,6 | 262,7 | 297,8 | 319,1 |
| MCNC | 274,2 | 32,4 | 205,2 | 261,5 | 272,2 | 302,9 | 320,1 |

incrementos superiores al 10 % en el 42 % de los casos y superiores al 15 % en el 25 % de los casos, alcanzando un valor máximo del 37,8 %. Aunque las estrategias de tipo voraz obtienen peores resultados que las de tipo PLE, la estrategia A-voraz consigue incrementos superiores al 10 % en el 31 % de los casos; y la estrategia M-voraz, en el 23 % de los casos.

A continuación se analizarán los datos de las distintas baterías de prue-

bas por separado. En todas ellas, las estrategias de tipo PLE son las que consiguen mejores resultados. A pesar de que era la batería de pruebas que presentaba las frecuencias más pequeñas, BP-184 es en la que se han producido los incrementos medios, los porcentajes de éxito y los valores máximos más altos. Los porcentajes de éxito se encuentran entre el 76 % y el 86 %, con incrementos medios entre el 9 % y el 16 % para los casos de éxito; mientras que los porcentajes de fracaso oscilan entre el 14 % y el 23 %, con incrementos medios entre el −5 % y el −4 %. Las estrategias A-PLE y M-PLE han obtenido incrementos medios del 13,3 % y 10 %, respectivamente. Ambas estrategias consiguen incrementos superiores al 10 % en la mitad de los casos.

La batería de pruebas BP-24 presenta porcentajes de éxito que se encuentran entre el 58 % y el 70 %, con incrementos medios entre el 9 % y el 11 % para los casos de éxito; mientras que los porcentajes de fracaso varían del 21 % al 30 %, con incrementos medios del −3 %. Las estrategias de tipo voraz logran incrementos mayores del 10 % en más del 24 % de los casos; y las estrategias de tipo PLE, en más del 36 % de los casos.

Los peores resultados se obtienen en la batería de pruebas MCNC, en la que el incremento medio no supera el 2 %. Sin embargo, todas las estrategias logran un porcentaje de éxito del 48 % excepto la estrategia A-voraz, en la que este porcentaje baja al 43 %, y se consiguen incrementos medios para los casos de éxito que varían del 3 % al 5 %. Aunque los porcentajes de fracaso son más significativos en esta batería de pruebas, son inferiores a los porcentajes de éxito. Los porcentajes de fracaso oscilan entre el 19 % y el 33 %, con incrementos medios para los casos de fracaso entre el −3 % y el −2 %.

Comparada con la optimización en área, se puede observar que la optimización en velocidad ha conseguido reducir los porcentajes de fracaso y aumentar los incrementos de velocidad en las baterías de pruebas BP-24 y MCNC (aunque la mejora ha sido más modesta en esta última). Sin embargo, los incrementos de velocidad se han reducido levemente en la batería BP-184. Por lo tanto, en términos generales, la optimización de velocidad de las implementaciones de FSMIM-T no beneficia claramente a ninguna de las estrategias estudiadas en esta tesis.

Tal como se ha mostrado en el análisis anterior, ordenadas de mayor a menor incremento de velocidad respecto a M-ref, las mejores estrategias son A-PLE, M-PLE, A-voraz y M-voraz. Es decir, primero las de tipo PLE seguidas de las de tipo voraz, y, para cada uno de los tipos anteriores, primero la estrategia de tipo AM correspondiente. Sin embargo, esta ordenación no muestra una correlación evidente con las reducciones del consumo de bloques

**Tabla 5.38:** Resumen estadístico del incremento RPC de velocidad obtenido por las implementaciones de las FSMIM-T generadas por las nuevas estrategias respecto a M-ref en las pruebas de implementación con optimización en velocidad ( %).

| Bat. de pruebas | Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | A-PLE | 13,3 | 12,7 | −14,8 | 5,3 | 12,6 | 21,4 | 55,2 |
| | A-voraz | 8,9 | 10,9 | −14,8 | 2,3 | 6,5 | 16,3 | 46,2 |
| | M-PLE | 10,0 | 10,5 | −9,6 | 1,5 | 10,0 | 16,3 | 37,8 |
| | M-voraz | 5,9 | 8,2 | −15,2 | 0,0 | 4,6 | 10,7 | 36,5 |
| BP-24 | A-PLE | 6,7 | 8,8 | −7,1 | −0,4 | 4,9 | 15,1 | 26,9 |
| | A-voraz | 4,8 | 8,2 | −5,7 | −0,2 | 1,3 | 11,1 | 25,2 |
| | M-PLE | 6,3 | 8,6 | −5,1 | −0,7 | 2,9 | 15,4 | 22,8 |
| | M-voraz | 4,8 | 7,3 | −4,4 | 0,0 | 1,8 | 9,2 | 21,0 |
| MCNC | A-PLE | 2,0 | 3,9 | −3,2 | 0,0 | 0,0 | 4,1 | 12,6 |
| | A-voraz | 1,0 | 4,1 | −5,5 | −0,5 | 0,0 | 2,8 | 12,9 |
| | M-PLE | 1,7 | 4,1 | −3,2 | 0,0 | 0,0 | 2,1 | 13,5 |
| | M-voraz | 1,2 | 3,0 | −3,2 | 0,0 | 0,0 | 2,1 | 7,9 |
| Total | A-PLE | 10,5 | 11,9 | −14,8 | 1,3 | 8,4 | 16,7 | 55,2 |
| | A-voraz | 7,0 | 10,1 | −14,8 | 0,0 | 5,2 | 13,7 | 46,2 |
| | M-PLE | 8,2 | 9,9 | −9,6 | 0,0 | 4,9 | 15,4 | 37,8 |
| | M-voraz | 5,0 | 7,6 | −15,2 | 0,0 | 3,1 | 9,4 | 36,5 |

de memoria, que eran mayores para las estrategias de tipo AM (encabezadas por A-PLE), ni con las reducciones del coste de selección de entradas, que eran mayores para las estrategias de tipo MA (con M-PLE en primer lugar). Este hecho parece indicar que la influencia del tamaño de la ROM sobre la velocidad de las implementaciones FSMIM-T es del mismo orden que la influencia del coste de selección de entradas.

Con objeto de mostrar la influencia de estos parámetros en la velocidad de las implementaciones FSMIM-T, se ha representado tanto el incremento RPC de la frecuencia de operación máxima como la reducción RPC del consumo de bloques frente a la reducción RPC del coste de selección de entradas, obtenidos por las nuevas estrategias respecto a M-ref (véase la figura 5.31). Como se puede observar, especialmente en los diagramas correspondientes a las estrategias de tipo AM, el coste de selección de entradas divide cada nube de puntos en dos regiones con comportamientos diferentes: una de ellas formada por los puntos con reducciones negativas (denominada *región con incremento del coste*) y la otra, por los puntos con reducciones positivas (denominada *región con reducción del coste*).

En la región con reducción del coste el comportamiento es el esperado.

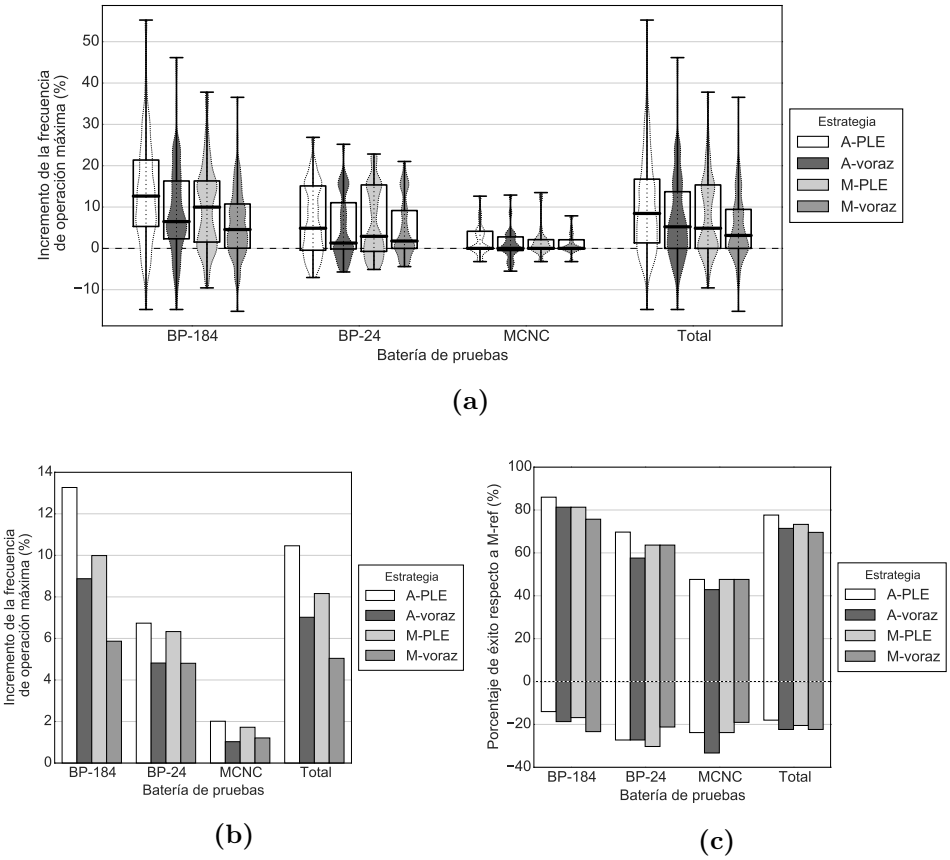



**Figura 5.30:** Incremento RPC de velocidad obtenido por las implementaciones de las FSMIM-T generadas por las nuevas estrategias respecto a M-ref en las pruebas de implementación con optimización en velocidad: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

La reducción del coste favorece tanto al incremento de velocidad, debido a que se reduce la complejidad del banco de multiplexores (figura 5.31a), como a la reducción del consumo de bloques, debido a que se reduce el ancho de la FSMIM-T (figura 5.31b).

Sin embargo, la región con incremento del coste parece mostrar el comportamiento opuesto: en vez de aumentar, tanto el incremento de velocidad como la reducción de bloques parecen disminuir con la reducción del coste de selección. Esta región corresponde principalmente a los casos en los que las nuevas estrategias han conseguido agrupar más estados que M-ref, es decir, las nuevas estrategias han reducido la profundidad de la ROM en ma-

yor medida que M-ref. Por una parte, en general, una mayor agrupación de estados consigue reducir el consumo de bloques a expensas de un incremento en el coste de selección de entradas. Aunque este efecto es mayor en las estrategias de tipo AM (tal como se explicó en la sección 5.4.1, página 172), también se produce en las estrategias de tipo MA. La agrupación de estados disminuye el número de entradas seleccionadas indeterminadas de la MSE, lo que provoca un aumento del coste de selección de entradas. Por otra parte, la reducción de bloques tiene un efecto positivo en la velocidad, debido a la simplificación del banco de multiplexores de la ROM y, en menor parte, a que se reducen los retrasos debidos a las líneas de rutado. Por lo tanto, en esta región, la agrupación de estados favorece la reducción de bloques y el incremento de velocidad obtenido por las nuevas estrategias respecto a M-ref al mismo tiempo que perjudica a la reducción del coste de selección de entradas.

La comparación de las rectas de regresión locales de la figura 5.31a con las de la figura 5.31b permite observar que el comportamiento cualitativo de ambas figuras es similar. Esto implica que la reducción de bloques tiene mayor influencia en el incremento de velocidad de las FSMIM-T que la reducción del coste de selección de entradas (de no ser así, en la figura 5.31a, la pendiente de la recta en la región con incremento del coste no sería descendente sino ascendente). Por otra parte, se puede observar en las nubes de puntos de las estrategias AM que la pendiente del incremento de velocidad es mayor en la región con reducción del coste que en la región con incremento del coste (figura 5.31a), al contrario de lo que ocurre con la pendiente de la reducción del número de bloques (figura 5.31b). Este comportamiento confirma que el incremento del coste influye negativamente en la velocidad.

El análisis anterior ha mostrado que, junto con el número de bloques usados, el coste de selección de entradas y, por lo tanto, la complejidad del banco de selectores de entrada, tienen una influencia significativa en la velocidad de las FSMIM-T. El camino crítico en el banco de selectores de entrada viene impuesto por el multiplexor con mayor número de entradas. Sin embargo, el objetivo de las fases de minimización de selección de entradas propuestas en esta tesis no es reducir el camino crítico del banco de selectores de entrada, sino el número de recursos necesarios para su implementación. La incorporación de nuevos objetivos de optimización que minimicen el número de entradas máximo de los multiplexores puede aumentar la velocidad de las FSMIM-T generadas. Esto daría lugar a una nueva clase de estrategias para la generación de FSMIM-T optimizadas en velocidad. El estudio de este nuevo objetivo de optimización formará parte de los trabajos futuros.

Para finalizar, se estudiará el porcentaje de casos en los que las nue-

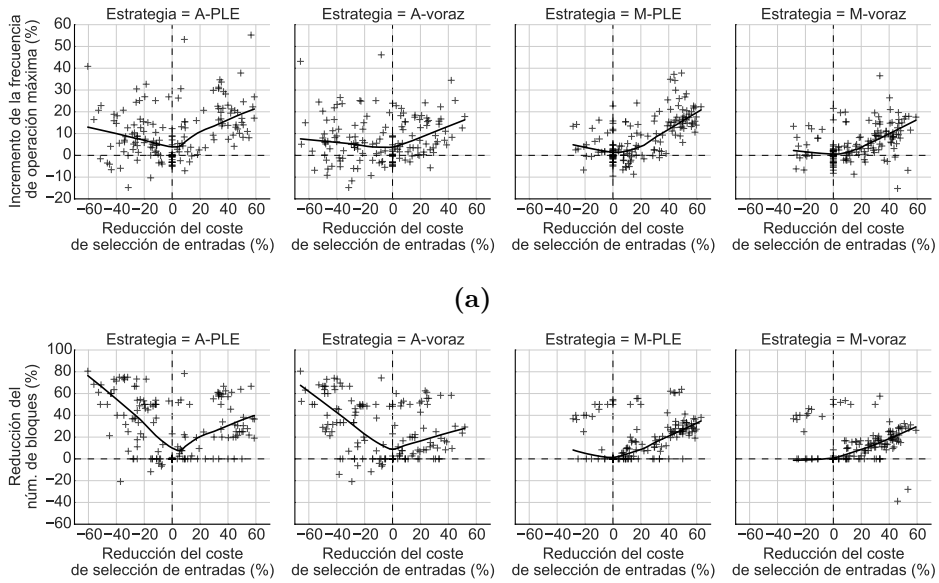



**Figura 5.31:** Influencia del coste de selección de entradas y del consumo de bloques de memoria en la frecuencia de operación máxima obtenida en las pruebas de implementación de FSMIM-T con optimización en velocidad. Las gráficas muestran la reducción RPC del coste de selección de entradas de las FSMIM-T generadas por las nuevas estrategias respecto a M-ref frente a: (a) el incremento RPC de velocidad de las FSMIM-T generadas por las nuevas estrategias respecto a M-ref, y (b) la reducción RPC del número de bloques usados por las FSMIM-T generadas por las nuevas estrategias respecto a M-ref. Cada gráfica muestra la recta de regresión local de la nube de puntos correspondiente.

vas estrategias son mejor alternativa que M-ref respecto a la frecuencia de operación máxima y el número de bloques usados, es decir, respecto a los dos parámetros más importantes para las implementaciones de FSMIM-T optimizadas en velocidad. La figura 5.32 muestra el porcentaje de éxito neto del incremento RPC de velocidad y de la reducción RPC del consumo de bloques obtenido por las nuevas estrategias respecto a M-ref. Para el conjunto completo de FSM, el porcentaje de éxito neto de las nuevas estrategias varía entre el 70 % y el 78 %, mientras que los porcentajes de fracaso neto no superan el 14 %. La batería de pruebas en la que se obtienen los mejores resultados es BP-184, con porcentajes de éxito neto entre el 76 % y el 86 %, y con porcentajes de fracaso neto que no superan el 12 %. Finalmente, aunque los peores resultados se presentan en la batería de pruebas MCNC, el

porcentaje de éxito neto duplica al porcentaje de fracaso neto en todas las estrategias excepto A-voraz.

### Estudio del consumo de LUT

Comparadas con las pruebas de implementación con optimización en área, el cambio de objetivo de optimización ha tenido un efecto desigual en los resultados de las implementaciones. En el 61 % de las pruebas realizadas con optimización en velocidad se han obtenido el mismo consumo de LUT que en las pruebas con optimización en área (este dato incluye a todas las estrategias estudiadas y al conjunto completo de FSM). Sin embargo, en el 39 % restante, la distancia relativa media entre los resultados de ambos tipos de pruebas ha sido del 24 %.

Puesto que el consumo de LUT es un parámetro secundario en las implementaciones con optimización en velocidad, la comparativa entre las nuevas estrategias y M-ref se realizará usando solamente los indicadores que se han considerado más importantes, comenzando por la reducción del consumo de LUT. La figura 5.33 muestra un resumen estadístico de la reducción RPC del número de LUT obtenida por las nuevas estrategias respecto a M-ref. Comparados con los resultados de las implementaciones optimizadas en área mostrados en la figura 5.24, se puede observar un ligero deterioro de los valores medios obtenidos en las pruebas con optimización en velocidad. Sin embargo, en promedio, las estrategias de tipo MA siguen presentando buenos resultados. Especialmente la estrategia M-PLE, que consigue la mejor

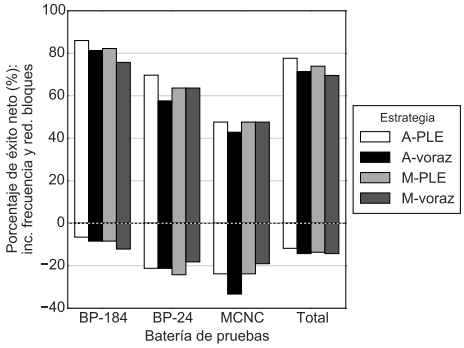



**Figura 5.32:** Porcentaje de éxito neto del incremento RPC de velocidad y de la reducción RPC del consumo de bloques obtenido por las nuevas estrategias respecto a M-ref en las pruebas de implementación de FSMIM-T con optimización en velocidad.

reducción media (19 %), el mayor porcentaje de éxito (61 %) y el menor porcentaje de fracaso (14 %).

Igual que ocurría en las pruebas con optimización en área, el comportamiento de las diferentes estrategias es muy desigual en las distintas baterías de pruebas. A excepción de la estrategia M-PLE, que siempre obtiene reducciones medias positivas, el resto de las estrategias presentan reducciones negativas en alguna batería de pruebas. En el caso de las estrategias de tipo MA, los porcentajes de éxito son superiores a los porcentajes de fracaso (excepto para la estrategia M-voraz en la batería de pruebas MCNC); además, las reducciones medias para los casos de éxito tienen mayor magnitud que las de los casos de fracaso.

La figura 5.34 muestra un resumen estadístico del incremento RPC del NLAB de las implementaciones FSMIM-T. La influencia de la optimización en velocidad sobre los incrementos medios ha sido muy pequeña comparada con los resultados de la optimización en área: las diferencias entre los incrementos medios obtenidos en ambas pruebas oscilan entre los $\pm 3$ puntos. Debido a que esta diferencia es pequeña, no se repetirá el análisis realizado en la sección 5.3.3.1.

Sin embargo, la influencia ha sido algo mayor en los porcentaje de éxito. Aunque la estrategia M-PLE mantiene el mismo porcentaje de fracaso y consigue un pequeño aumento en el porcentaje de éxito, los resultados del resto de estrategias han empeorado. A pesar de ello, los resultados obtenidos por las nuevas estrategias siguen siendo buenos. Al igual que ocurría con la optimización en área, las estrategias de tipo AM son las que obtienen mejores resultados. Para el conjunto completo de FSM, todas las estrategias consiguen porcentajes de éxito entre el 62 % de la estrategia M-voraz (con un incremento medio del 30 % para los casos de éxito) y el 75 % de la estrategia A-PLE (con un incremento medio del 93 % para los casos de éxito). Además, el porcentaje de fracaso es inferior al 15 % para todas las estrategias, con incrementos medios para los casos de fracaso entre el $-9$ % y el $-4$ %.

Respecto a las baterías de pruebas, los peores resultados se obtienen para MCNC. La estrategia M-PLE es la que presenta el mejor porcentaje de éxito en esta batería de pruebas (43 %) y el porcentaje de fracaso más pequeño (5 %). Las dos estrategias de tipo AM presentan un porcentaje de éxito del 24 % y un porcentaje de fracaso del 29 %; sin embargo, el incremento medio para los casos de fracaso es tan solo del $-3$ %, mientras que para los casos de éxito es del 11 %, aproximadamente. Por otra parte, en la batería de pruebas BP-24, los porcentajes de éxito varían entre el 39 % y el 58 %, con incrementos medios para los casos de éxito entre el 25 % y el 28 %; mientras que los porcentajes de fracaso son inferiores al 22 %, con incrementos medios

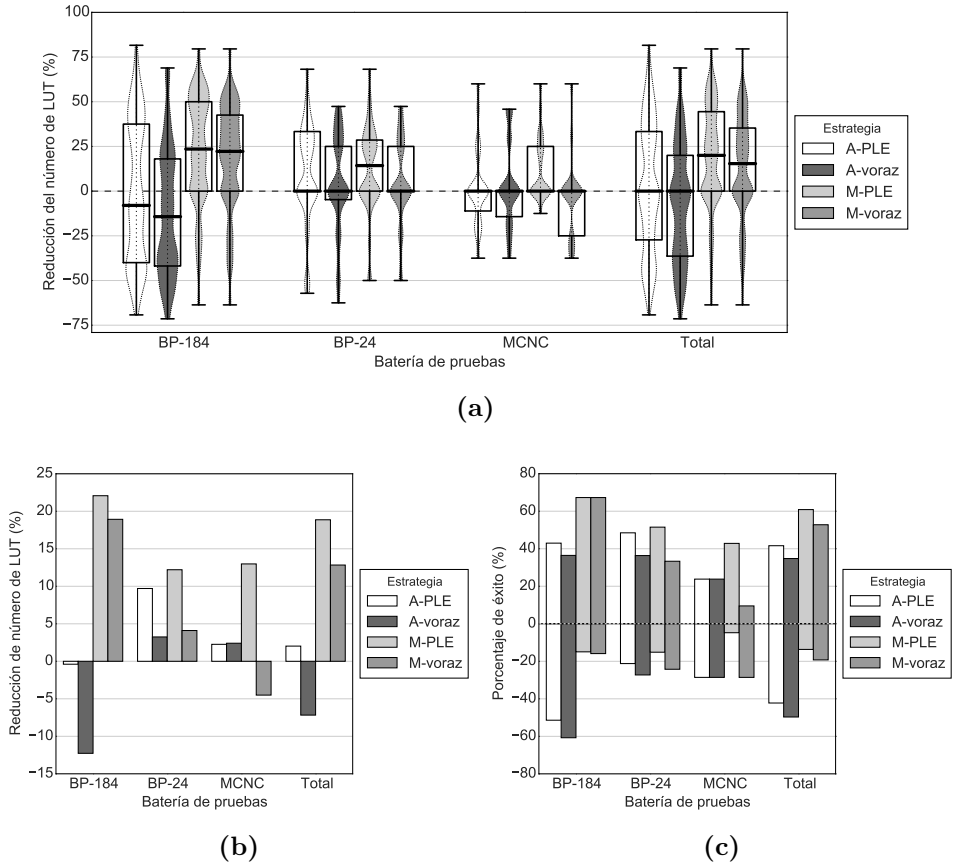



**Figura 5.33:** Reducción RPC del número de LUT usadas por las implementaciones de las FSMIM-T generadas por las nuevas estrategias respecto a M-ref obtenida en las pruebas de implementación con optimización en velocidad: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

para los casos de fracaso entre el $-5\,\%$ y el $-3\,\%$. Para finalizar, los mejores resultados se obtienen en la batería de pruebas BP-184, en la que se consiguen porcentajes de éxito entre el $80\,\%$ y el $92\,\%$, y porcentajes de fracaso que no superan el $9\,\%$. Las estrategias de tipo AM consiguen incrementos medios por encima del $90\,\%$ para los casos de éxito y tan solo del $-9\,\%$ para los casos de fracaso.

Para finalizar, se estudiará el porcentaje de casos en los que las nuevas estrategias son mejor alternativa que M-ref respecto a la velocidad y al NLAB. Puesto que el NLAB permite relacionar el consumo de bloques con el de LUT, este estudio involucra a los tres parámetros analizados en las

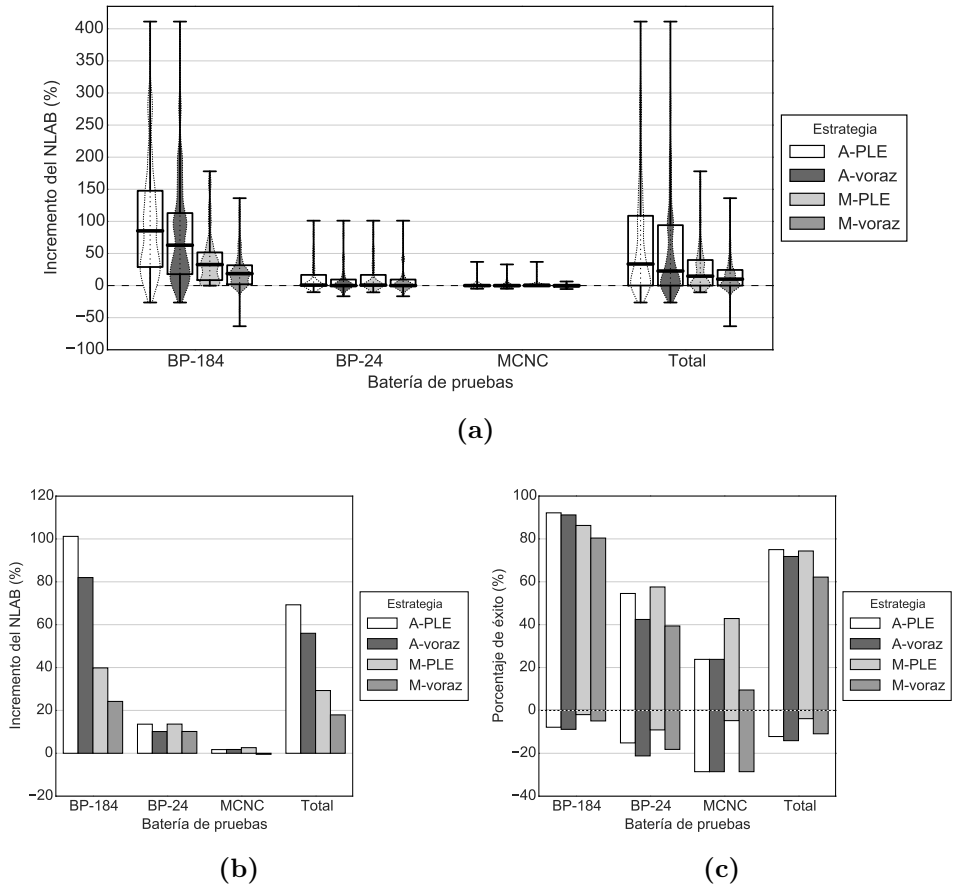



**Figura 5.34:** Incremento RPC del NLAB obtenido por las implementaciones de las FSMIM-T generadas por las nuevas estrategias respeto a M-ref en las pruebas de implementación con optimización en velocidad: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

pruebas de implementación: el consumo de bloques, el consumo de LUT y la velocidad. La figura 5.35 muestra el porcentaje de éxito neto del incremento RPC de velocidad y del incremento RPC del NLAB obtenido por las nuevas estrategias respecto a M-ref. Para el conjunto completo de FSM, el porcentaje de éxito neto de las nuevas estrategias varía entre el 63 % y el 72 %, mientras que los porcentajes de fracaso neto son inferiores al 14 %. La batería de pruebas en la que se obtienen los mejores resultados es BP-184, con porcentajes de éxito neto entre el 75 % y el 86 %, y con porcentajes de fracaso neto que no alcanzan el 11 %. En la batería de pruebas BP-24, los

porcentajes de éxito neto casi triplican a los porcentajes de fracaso neto. Los peores resultados se presentan en la batería de pruebas MCNC, en la que los porcentajes de éxito y de fracaso son similares para todas las estrategias excepto M-PLE; en esta última el porcentaje de éxito neto es tres veces el porcentaje de fracaso neto.

### 5.4.3. Conclusiones

Respecto a las nuevas estrategias de optimización de FSMIM-T presentadas en esta tesis, los resultados del estudio han mostrado que, tanto para las dos estrategias de tipo AM (A-PLE y A-voraz) como para las dos de tipo MA (M-PLE y M-voraz), la estrategia de tipo PLE correspondiente obtiene mejores resultados que la de tipo voraz. Sin embargo, atendiendo al coste temporal de los algoritmos que utilizan, las estrategias más rápidas son las de tipo voraz (seguidas por la estrategia de referencia), siendo las estrategias de tipo PLE las que presentan un mayor coste temporal. Por lo tanto, cada estrategia de tipo voraz puede ser considerada como una alternativa de menor coste computacional a la estrategia de tipo PLE correspondiente, que consume menor tiempo de CPU a expensas de generar una FSMIM-T con menores prestaciones. Esta gama de estrategias permitirá implementar en una futura herramienta diferentes niveles de esfuerzo de optimización, similares a los que se incluyen en la mayoría de las herramientas de síntesis comerciales [Xilinx, 2013].

En cualquier caso, los resultados han mostrado, por una parte, que tanto las estrategias de tipo PLE como las de tipo voraz consiguen mejores

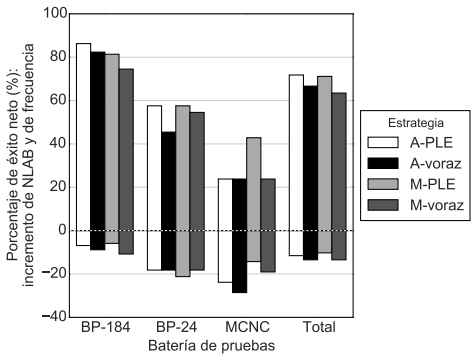



**Figura 5.35:** Porcentaje de éxito neto del incremento RPC de velocidad y del incremento RPC del NLAB obtenido por las nuevas estrategias respecto a M-ref en las pruebas de implementación de FSMIM-T con optimización en velocidad.

resultados que la estrategia de referencia. Por otra parte, desde un punto de vista cualitativo, el análisis comparativo de los resultados obtenidos por las estrategias de tipo voraz es similar al de los resultados obtenidos por las estrategias de tipo PLE, siendo las diferencias esencialmente cuantitativas. Por este motivo, este apartado se centrarán exclusivamente en las estrategias de tipo PLE.

El estudio realizado ha demostrado que las nuevas estrategias permiten obtener mejoras significativas tanto en el consumo de recursos como en la velocidad de las FSMIM-T. Las pruebas independientes del dispositivo han permitido comprobar que las FSMIM-T generadas por las nuevas estrategias ocupan menos cantidad de memoria ROM que las generadas por M-ref. En más del 70 % de los casos estudiados, el tamaño mínimo de las FSMIM-T generadas por las nuevas estrategias ha sido estrictamente menor que el de las generadas M-ref, siendo despreciable el porcentaje de fracaso de las nuevas estrategias (inferior al 5 %). Dependiendo de las características de los bloques de memoria (es decir, del tamaño y geometrías disponibles), estos resultados se aproximarán en mayor o menor medida a los obtenidos en las implementaciones reales sobre dispositivos FPGA. En las pruebas realizadas con el dispositivo FPGA de la familia Spartan-6, los porcentajes de éxito se han reducido ligeramente hasta valores entre el 60 % y el 70 %. Sin embargo, los porcentajes de fracaso siguen siendo despreciables y, además, se alcanzan reducciones medias del consumo de bloques respecto a M-ref próximas al 17 % (M-PLE) y al 28 % (A-PLE). Cabe destacar que en la cuarta parte de los casos, la estrategia A-PLE, que es la que consume el menor número de bloques, ha conseguido reducciones entre el 50 % el 81 %. Estos datos indican que la estrategia M-ref no puede competir con las nuevas estrategias en el consumo de bloques.

En un porcentaje considerable de los casos, estas reducciones en el consumo de bloques vienen acompañadas también de una reducción del consumo de LUT. La estrategia M-PLE consigue mejorar el consumo de LUT de M-ref en más del 60 % de los casos, presentando un pequeño porcentaje de fracaso del 12 %. Esta estrategia consigue reducciones medias por encima del 25 % en prácticamente la mitad de los casos y es claramente mejor opción que la estrategia M-ref en consumo de recursos (tanto bloques como LUT). Por otra parte, la estrategia A-PLE da prioridad al consumo de bloques frente al consumo de LUT, logrando las mayores tasas de reducción del número de bloques a expensas de un aumento en el consumo de LUT. A pesar de ello, la estrategia A-PLE usa menos LUT que la estrategia M-ref en el 40 % de los casos estudiados, siendo mejor opción que M-ref tanto en consumo de bloques como de LUT en prácticamente todos estos casos.

Debido a que mejoran tanto el consumo de bloques de memoria como el de LUT, las nuevas estrategias hacen un uso más eficiente de los bloques que M-ref, ahorrando mayor número de LUT por cada bloque usado. Ambas estrategias logran mejorar el NLAB obtenido por M-ref en un amplio número de casos (superior al 73 %), alcanzando incrementos medios próximos al 30 % (estrategia M-PLE) y al 69 % (estrategia A-PLE). En el 25 % de los casos, el NLAB obtenido por la estrategia A-PLE es más del doble que el de M-ref, llegando incluso a valores que son cuatro veces más grandes que los de M-ref.

Respecto a la velocidad de operación, aunque las mejoras son menos significativas en promedio (con incrementos de velocidad del 8 % y 11 % aproximadamente), los porcentajes de éxito son superiores al 70 %. A pesar de que el área y la velocidad son parámetros en conflicto, estas mejoras de velocidad se suman a las del resto de parámetros. Como consecuencia, en más del 70 % de los casos las nuevas estrategias son mejor alternativa en velocidad y consumo de bloques que M-ref, y en más del 60 % son mejor alternativa en velocidad y ahorro de LUT por bloque usado, mientras que M-ref es mejor alternativa en menos del 14 % para ambas comparativas.

Del estudio anterior se puede concluir que las nuevas estrategias sustituyen eficazmente a la estrategia M-ref, mejorando sus prestaciones. Por otra parte, el abanico de estrategias propuestas no compiten entre sí, sino que se complementan, ofreciendo al diseñador varias alternativas para alcanzar los objetivos de diseño respecto a las restricciones de área. La estrategia M-PLE permite generar las FSMIM-T con menor consumo de LUT a expensas de un mayor uso de bloques de memoria. Por el contrario, la estrategia A-PLE permite generar las FSMIM-T con el menor consumo de bloques de memoria a costa de aumentar el consumo de LUT, siendo también esta estrategia la que hace un uso más eficiente de los bloques para ahorrar LUT. Respecto a la velocidad de operación, no hay diferencias significativas entre ellas.

## 5.5. Estudio de la influencia de las estrategias de optimización sobre la arquitectura FSMIM-S

El objetivo principal de este estudio es evaluar las mejoras obtenidas por las nuevas estrategias de optimización cuando se aplican a la arquitectura FSMIM-S respecto a la estrategia de referencia. Por otra parte, se realizará un estudio comparativo con los resultados obtenidos con la arquitectura FSMIM-T.

Tal como se estudió en la sección 2.4, las diferencias entre las arquitecturas FSMIM-S y FSMIM-T afectan de forma desigual al problema de optimización de FSMIM y, por tanto, a la efectividad de las diferentes estrategias de optimización. Una revisión de dichas diferencias facilitará, por una parte, la interpretación de los resultados experimentales obtenidos en las pruebas realizadas con la arquitectura FSMIM-S y, por otra, la comparación con los que se obtuvieron en las pruebas realizadas con la arquitectura FSMIM-T.

En primer lugar, la influencia de las estrategias en el tamaño de la ROM es diferente en ambas arquitecturas. La ROM de la arquitectura FSMIM-S no contiene los bits de selección de entradas ni los bits de codificación de grupo; por lo tanto, su ancho no depende de las características de las FSMIM. Como consecuencia, en esta arquitectura las estrategias de optimización sólo pueden reducir la profundidad de la ROM. Esto significa que la única fase que puede reducir el tamaño de la FSMIM-S es la fase de agrupación de estados.

En las estrategias de tipo AM, la minimización de selección de entradas no influye en el tamaño de las FSMIM-S, ya que este viene determinado por la fase inicial de agrupación de estados. Como consecuencia, en las pruebas independientes del dispositivo (sin ajuste de profundidad), las FSMIM-S generadas por las estrategias de tipo AM tienen el tamaño mínimo debido a que la fase inicial de agrupación de estados siempre consigue obtener el número de grupos mínimo y, por lo tanto, la profundidad mínima. En las pruebas de implementación (con ajuste de profundidad), las FSMIM-S generadas por estas estrategias ocupan el número de bloques mínimo (sin tener en cuenta la posible reducción posterior del número de bloques debida al proceso de síntesis de bajo nivel). Por el contrario, en la arquitectura FSMIM-T no se alcanza necesariamente la profundidad mínima debido a que el ancho de la ROM puede crecer con la agrupación de estados (por el aumento de los bits de selección de entradas).

Por otra parte, en las estrategias de tipo MA, las permutaciones de

entradas de la MSE que se realiza en la fase de minimización de la selección de entradas puede limitar la efectividad de la fase posterior de agrupación de estados. En estas estrategias, la minimización de selección de entradas influye indirectamente en el tamaño de las FSMIM-S, impidiendo en muchos casos que se alcance el tamaño mínimo.

Respecto a la complejidad del banco de selectores, la influencia de las estrategias es menor en la arquitectura FSMIM-S que en la arquitectura FSMIM-T. Mientras que la complejidad del banco de selectores de entrada de la FSMIM-T depende del coste de selección de entradas, la complejidad del banco de selectores de la FSMIM-S depende tanto del coste de selección de entradas como de la codificación de los estados. Los algoritmos de minimización de selección de entradas han sido diseñados para la arquitectura FSMIM-T con el doble objetivo de minimizar el coste de selección de entradas y el ancho de la ROM. Tal como ha demostrado el estudio experimental realizado en la sección 5.4, el impacto de estos algoritmos en la reducción del banco de multiplexores de entrada de las FSMIM-T es significativo. Sin embargo, aunque la minimización del coste de selección también puede favorecer la reducción de complejidad del banco de selectores de entrada de la FSMIM-S, su influencia se verá mermada por el impacto de la codificación de estados. Tal como se mencionó en la sección 5.3.3.1, formarán parte de los trabajos futuros tanto el estudio de la influencia de la codificación de estados en el banco de selectores de entradas como el desarrollo de algoritmos de codificación que minimicen su complejidad.

### 5.5.1. Pruebas independientes del dispositivo

El objetivo principal de las pruebas independientes del dispositivo es el estudio de la influencia de las diferentes estrategias de optimización de FSMIM sobre el tamaño genérico de las FSMIM-S. En esta arquitectura, el análisis del tamaño genérico permitirá realizar también un estudio de la efectividad de los diferentes algoritmos de agrupación de estados. Finalmente, se evaluará la influencia de la arquitectura FSMIM-S en el proceso de optimización del coste de selección de entradas.

La influencia de las estrategias sobre el codificador de grupos no se estudiará en estas pruebas. La complejidad del codificador de grupos depende en gran medida de la codificación de grupos y estados, que no son características propias de las FSMIM. De hecho, esta codificación se realiza como parte del proceso de síntesis en las implementaciones FPGA, que es posterior a la generación de la FSMIM-S. Aunque la complejidad del codificador de grupos también depende del número de grupos obtenido por las fases de agrupación

de estados, no existe una relación clara entre el número de grupos y la complejidad que sea independiente de la codificación. Como consecuencia, por una parte, la simplificación del codificador de grupos no es el objetivo de ninguna de las fases de la generación de FSMIM y, por otra, no se dispone de una medida genérica que permita estimar su complejidad a partir de las características de las FSMIM-S.

El estudio se ha realizado utilizando las 242 FSM de las tres baterías de pruebas. A partir de cada FSM se han generado cinco FSMIM sin ajuste de profundidad utilizando la estrategia de referencia y las nuevas estrategias propuestas en esta tesis. Las características de las FSMIM-S analizadas en esta sección han sido calculadas a partir de dichas FSMIM.

### Estudio del tamaño de la FSMIM-S

Como se ha mencionado anteriormente, el tamaño de la ROM de las FSMIM-S viene determinado exclusivamente por la fase de agrupación de estados. Esto implica que el tamaño puede ser utilizado como una medida del grado de agrupación alcanzado por las estrategias, ya que depende linealmente del número de grupos (véase la ecuación 2.23). Por lo tanto, el estudio realizado en esta sección ofrece una doble lectura: la interpretación habitual, es decir, un estudio de la cantidad de memoria necesaria para las implementaciones de FSMIM-S; y una interpretación alternativa como estudio de la eficiencia de los diferentes algoritmos de agrupación estudiados en esta tesis. La tabla 5.39 muestra un resumen estadístico del tamaño en Kbits de las FSMIM-S generadas por las diferentes estrategias con el conjunto completo de FSM (es decir, el tamaño de la ROM requerida para implementar las FSMIM en la arquitectura FSMIM-S).

Al igual que ocurría con las FSMIM-T, en promedio, las FSMIM-S generadas a partir de las nuevas estrategias requieren una ROM de menor tamaño que las generadas con M-ref, siendo las estrategias de tipo AM las que obtienen los mejores resultados. Sin embargo, como cabía esperar, las diferencias entre los resultados obtenidos por las distintas estrategias son menores que en las pruebas realizadas con las FSMIM-T. Tal como se ha explicado en la introducción de la sección, mientras que las estrategias pueden reducir tanto el ancho como la profundidad de la ROM de las FSMIM-T, sólo es posible reducir la profundidad en la arquitectura FSMIM-S. Por lo tanto, el impacto de las estrategias es menor.

Por otra parte, como puede observarse, los valores obtenidos por las dos estrategias de tipo AM son idénticos. En ambas estrategias, la fase inicial de agrupación de estados consigue agrupar el máximo número de estados

posible, lo que les permite generar las FSMIM-S de tamaño mínimo. Por el contrario, en las estrategias de tipo MA, el número de estados que se consigue agrupar es diferente para cada estrategia debido a que utilizan algoritmos distintos para la minimización de selección de entradas, que afectan de forma diferente a la fase de agrupación de estados posterior. La estrategia M-PLE obtiene mejores resultados que la estrategia M-voraz, lo que indica que la MSE generada por los algoritmos de minimización de selección de entradas basados en PLE es más adecuada para la agrupación de estados que la generada por los algoritmos voraces.

La comparación de los resultados medios obtenidos por las diferentes estrategias pone de manifiesto, por una parte, que las mayores tasas de agrupación se consiguen con la fase de agrupación de estado inicial y, por otra parte, que el uso combinado de los nuevos algoritmos de minimización de selección de entradas y de agrupación de estados consiguen agrupar un mayor número de estados que los usados por la estrategia M-ref.

Respecto a los resultados obtenidos en cada batería de pruebas, los datos parecen indicar que la efectividad de las nuevas estrategias aumenta con el tamaño de las FSMIM-S. La figura 5.36 muestra el tamaño medio de las FSMIM-S (en Kbits) para cada batería de pruebas. Puede observarse como las diferencias entre el tamaño medio de las FSMIM-S generadas por las diferentes estrategias son claramente mayores en la batería de pruebas BP-184. De hecho, las diferencias medias son prácticamente despreciables en el resto de baterías de pruebas.

El análisis del tamaño de las FSMIM-S generadas por la estrategia M-ref permite una primera valoración del margen de mejora que tendrán las nuevas estrategias en las implementaciones sobre dispositivos FPGA (véase la tabla 5.40). El tamaño de las FSMIM-S obtenidas con las baterías de pruebas MCNC y BP-24 es pequeño comparado con el tamaño de los bloques de memoria del dispositivo FPGA utilizado en las pruebas de implementación. El 82 % de las FSMIM-S tienen un tamaño menor de 18 Kbits. Respecto a los bloques de 9 Kbits (recuérdese que un bloque de 18 Kbits puede dividirse en dos de 9 Kbits), el 72 % de las FSMIM-S tiene un tamaño menor y sólo en 10 casos se supera la profundidad máxima de estos bloques (8 Kpalabras). Estos datos indican que las FSM de las baterías MCNC y BP-24 ofrecen a las nuevas estrategias un margen muy pequeño para mejorar las implementaciones en FPGA de las FSMIM-S generadas con la estrategia M-ref. Este margen de mejora es incluso menor que el que proporciona la arquitectura FSMIM-T.

Se ha calculado la reducción RPC del tamaño de las FSMIM-S obtenidas por las nuevas estrategias respecto a M-ref con el fin de cuantificar las

**Tabla 5.39:** Resumen estadístico del tamaño de las FSMIM-S (Kbits) obtenidas con el conjunto completo de FSM.

| Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| A-voraz | 72,66 | 193,71 | 0,05 | 1,18 | 13,00 | 51,75 | 1804,00 |
| A-PLE | 72,66 | 193,71 | 0,05 | 1,18 | 13,00 | 51,75 | 1804,00 |
| M-PLE | 84,82 | 202,98 | 0,05 | 1,18 | 16,78 | 75,75 | 1848,00 |
| M-voraz | 86,87 | 204,42 | 0,05 | 1,18 | 18,00 | 79,88 | 1848,00 |
| M-ref | 89,29 | 205,75 | 0,05 | 1,20 | 18,30 | 78,00 | 1848,00 |

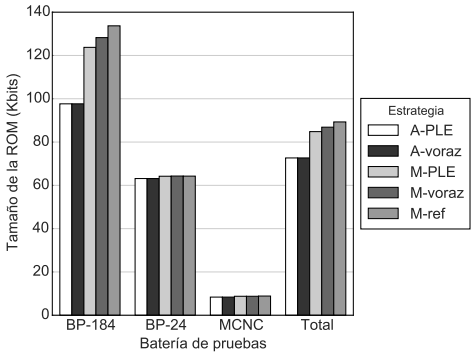



**Figura 5.36:** Tamaño medio de las FSMIM-S (Kbits).

**Tabla 5.40:** Resumen estadístico del tamaño de las FSMIM-S (Kbits) generadas con la estrategia M-ref.

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 133,69 | 166,61 | 10,50 | 39,00 | 73,31 | 162,75 | 1079,00 |
| BP-24 | 64,26 | 251,71 | 0,05 | 0,31 | 1,20 | 10,83 | 1848,00 |
| MCNC | 8,86 | 13,33 | 0,08 | 1,19 | 5,03 | 10,98 | 63,00 |

mejoras conseguidas. La tabla 5.41 muestra un resumen estadístico de los valores calculados y la figura 5.37, una representación de dicho resumen.

Respecto al conjunto completo de FSM, aunque todas las estrategias propuestas consiguen reducciones medias positivas, las mejores estrategias son, con un amplio margen de diferencia, las de tipo AM, que alcanzan una reducción media del 18,8 % mientras que las reducciones obtenidas por las de tipo MA no superan el 5,4 %. Las estrategias de tipo AM presentan un porcentaje de éxito del 62 %, con una reducción media del 30 % para los casos de éxito, y alcanzan reducciones entre el 34 % y el 76,5 % en el 25 %

de los casos. Cabe destacar que el porcentaje de fracaso de estas estrategias siempre es del 0 % debido a que las FSMIM-S que generan alcanzan el tamaño mínimo. Por otra parte, aunque las estrategias de tipo MA obtienen reducciones negativas en algunos casos, el porcentaje de fracaso no supera el 9 % (con reducciones medias para los casos de fracaso entre −7 % y −6 %), mientras que el porcentaje de éxito se encuentra entre el 34 % de la estrategia M-voraz y el 43 % de la estrategia M-PLE.

Respecto a las baterías de pruebas, los peores resultados se obtienen para MCNC y BP-24, con reducciones medias para las estrategias de tipo AM que no superan el 5,8 %. Sin embargo, el porcentaje de éxito para estas estrategias se encuentra entre el 32 % de BP-24 y el 35 % de MCNC. Los mejores resultados se obtienen para la batería de pruebas BP-184, en la que las estrategias de tipo AM obtienen una reducción media del 37,8 %, alcanzando reducciones entre el 55,6 % y el 76,5 % en el 25 % de los casos.

Si se comparan estas reducciones con las obtenidas en la arquitectura FSMIM-T (véase la figura 5.13), se puede observar que la influencia de las estrategias de tipo MA es sensiblemente menor sobre la arquitectura FSMIM-S. Las estrategias de tipo AM también presentan un comportamiento similar en las baterías de pruebas MCNC y BP-24; sin embargo, en la

**Tabla 5.41:** Resumen estadístico de la reducción RPC del tamaño de la FSMIM-S obtenida por las nuevas estrategias respecto a M-ref ( %).

| Bat. de pruebas | Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | A-PLE | 37,8 | 20,2 | 5,8 | 19,5 | 38,7 | 55,6 | 76,5 |
| | A-voraz | 37,8 | 20,2 | 5,8 | 19,5 | 38,7 | 55,6 | 76,5 |
| | M-PLE | 10,6 | 10,3 | −14,3 | 3,7 | 8,3 | 15,9 | 40,0 |
| | M-voraz | 6,3 | 10,3 | −23,5 | 0,0 | 3,9 | 14,3 | 28,6 |
| BP-24 | A-PLE | 3,0 | 6,5 | 0,0 | 0,0 | 0,0 | 2,4 | 25,0 |
| | A-voraz | 3,0 | 6,5 | 0,0 | 0,0 | 0,0 | 2,4 | 25,0 |
| | M-PLE | 1,3 | 4,8 | 0,0 | 0,0 | 0,0 | 0,0 | 25,0 |
| | M-voraz | 0,9 | 4,5 | −9,1 | 0,0 | 0,0 | 0,0 | 25,0 |
| MCNC | A-PLE | 5,8 | 11,1 | 0,0 | 0,0 | 0,0 | 10,8 | 50,0 |
| | A-voraz | 5,8 | 11,1 | 0,0 | 0,0 | 0,0 | 10,8 | 50,0 |
| | M-PLE | 1,0 | 4,9 | 0,0 | 0,0 | 0,0 | 0,0 | 25,0 |
| | M-voraz | 1,0 | 4,9 | 0,0 | 0,0 | 0,0 | 0,0 | 25,0 |
| Total | A-PLE | 18,8 | 22,4 | 0,0 | 0,0 | 9,5 | 34,0 | 76,5 |
| | A-voraz | 18,8 | 22,4 | 0,0 | 0,0 | 9,5 | 34,0 | 76,5 |
| | M-PLE | 5,4 | 9,0 | −14,3 | 0,0 | 0,0 | 8,5 | 40,0 |
| | M-voraz | 3,3 | 8,1 | −23,5 | 0,0 | 0,0 | 3,9 | 28,6 |

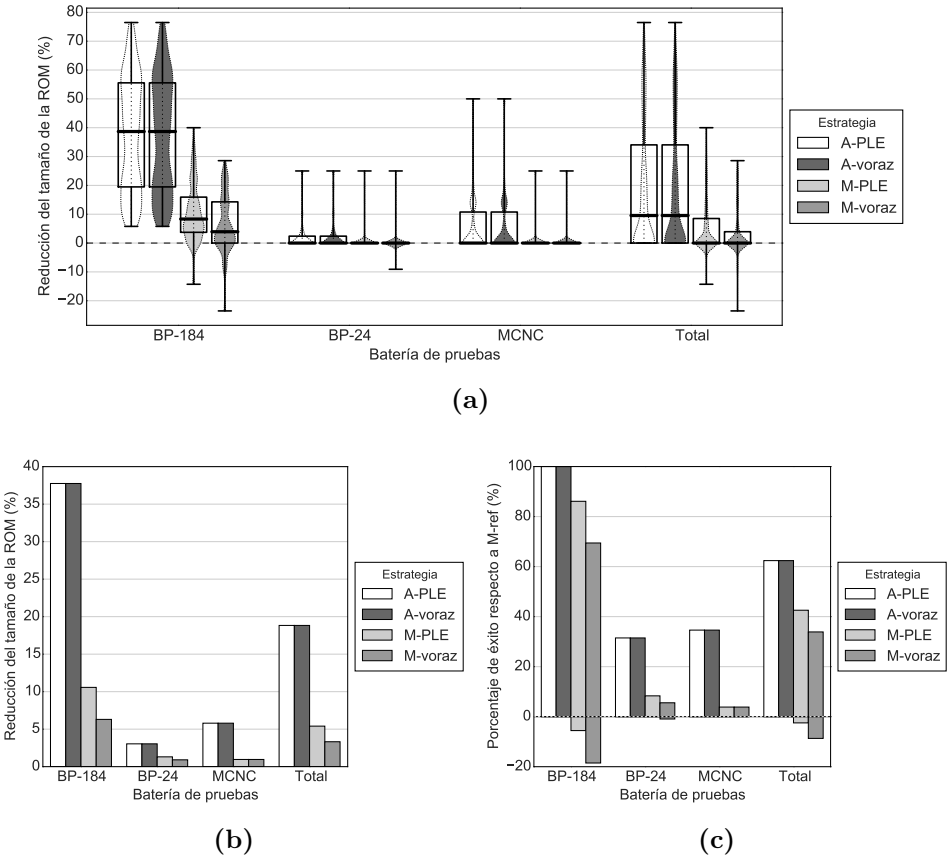



**Figura 5.37:** Reducción RPC del tamaño de la FSMIM-S obtenida por las nuevas estrategias respecto a M-ref: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

batería de pruebas BP-184, el impacto en ambas arquitecturas es del mismo orden. Además, estas estrategias presentan la ventaja de que las FSMIM-S que generan tienen el tamaño mínimo.

Se ha estudiado la influencia que tiene el tamaño de la FSM-ROM en las reducciones de obtenidas por las nuevas estrategias respecto a M-ref (véase la figura 5.38). Las rectas de regresión mostradas en la figura indican que la reducción crece con el tamaño de la FSM-ROM. En las estrategias de tipo MA, la pendiente de la recta de regresión es pequeña. Sin embargo, la pendiente es pronunciada en las estrategias de tipo AM, similar a las obtenidas con la arquitectura FSMIM-T (véase la figura 5.14). Como se puede apreciar, las nubes de puntos pueden dividirse en dos regiones: la

región formada por las FSM con un tamaño de FSM-ROM menor de 32 Kbits aproximadamente (*región izquierda*) y la formada por las que tienen un tamaño mayor (*región derecha*). En las nubes de puntos de las estrategias de tipo AM, los puntos con reducción cero corresponden a los casos en los que la estrategia M-ref ha conseguido agrupar el máximo número de estados posible, es decir, en los que ha generado una FSMIM-S con tamaño mínimo. Como puede observarse, la región izquierda está formada principalmente por estos puntos, mientras que la región derecha contiene a la mayoría de los casos con reducciones mayores que cero.

### Estudio del coste de selección de entradas

La figura 5.39 muestra un resumen estadístico del coste de selección de entradas de las FSMIM-S generadas por las diferentes estrategias. Si se compara con los resultados obtenidos con la arquitectura FSMIM-T (véase la figura 5.15), se observan escasas diferencias. En promedio, como cabía esperar, las estrategias que presentan un coste medio más alto son las que conseguían mayores reducciones en el tamaño de la ROM, es decir, las de tipo AM. Por el contrario, las nuevas estrategias de tipo MA son las que alcanzan el menor coste medio, aunque, como consecuencia, generan FSMIM-S de mayor tamaño que las estrategias de tipo AM. Comparadas con la estrategia M-ref, las estrategias de tipo MA mejoran tanto el tamaño de la ROM como el coste de selección de entradas, y las de tipo AM mejoran el tamaño de la ROM a expensas de agravar el coste de selección de entradas.

Sin embargo, aunque la mayoría de los valores obtenidos por ambas arquitecturas son idénticos, existe un porcentaje de casos en los que los resultados no coinciden. Estas diferencias se hacen más evidentes cuando se

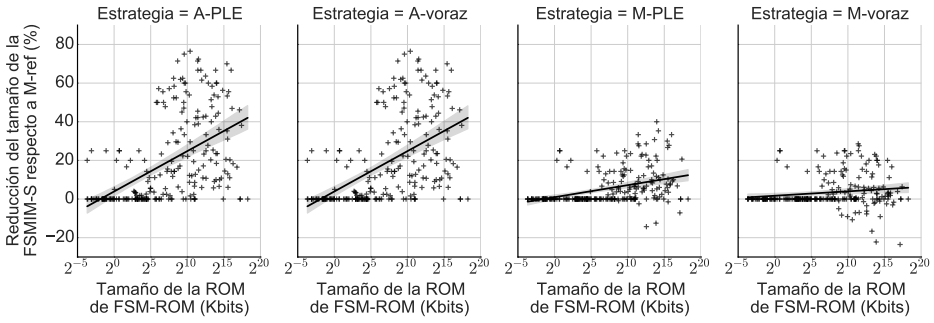



**Figura 5.38:** Reducción RPC del tamaño de la FSMIM-S obtenida por las nuevas estrategias respecto a M-ref frente al tamaño de la ROM de la implementación FSM-ROM.

comparan las reducciones en el coste de selección de entradas respecto a M-ref. Sin embargo, antes de comenzar el estudio de las reducciones, se analizarán los casos en los que se han obtenido valores distintos para cada arquitectura.

El efecto más importante de la arquitectura sobre el coste de selección de entradas está relacionado con la fase de agrupación de estados y se debe a que el coste de selección de entradas tiende a incrementarse con la disminución del número de grupos (véase la sección 5.4.1, página 172). Tal como se ha explicado anteriormente (en la introducción de la sección 5.5), en la arquitectura FSMIM-S las fases de agrupación de estados de las nuevas estrategias obtienen el mínimo número de grupos alcanzable por sus algoritmos de agrupación. Sin embargo, esto no ocurre necesariamente en la arquitectura FSMIM-T (véase el algoritmo de agrupación lexicográfico con control de tamaño, sección 3.4). Como consecuencia, en términos generales, las nuevas estrategias consiguen agrupar un mayor número de estados en la arquitectura FSMIM-S que en la arquitectura FSMIM-T, lo que repercute negativamente en el coste de selección de entradas de la arquitectura FSMIM-S.

Este efecto aparece con mayor frecuencia en las nuevas estrategias de tipo MA (aproximadamente en el 35 % de los casos) que en las de tipo AM (aproximadamente en el 21 % de los casos). Sin embargo, en las estrategias de tipo AM el impacto es mayor (con un incremento medio del 13 % aproximadamente) que en las de tipo MA (con un incremento medio del 18 % aproximadamente).

En la estrategia M-ref, el efecto de la arquitectura es menos significativo que en las nuevas estrategias; tan sólo en el 8 % de los casos se produce un incremento del coste de selección. El algoritmo de agrupación de estados de esta estrategia aplica diferentes heurísticas y selecciona la que permite obtener la FSMIM de menor tamaño. La diferencia en el tamaño de la ROM de ambas arquitecturas provoca que en algunos casos se seleccione una heurística distinta para cada una de ellas.

El hecho de que el efecto negativo de la arquitectura FSMIM-S sea más significativo en las nuevas estrategias que en la estrategia M-ref es la causa de que las reducciones del coste de selección de entradas de las nuevas estrategias sean menores en esta arquitectura que en la arquitectura FSMIM-T. La figura 5.40 muestra un resumen estadístico de la reducción RPC del coste de selección de entradas obtenida por las nuevas estrategias respecto a M-ref.

Para el conjunto completo de FSM, los mejores resultados se obtienen con las nuevas estrategias de tipo MA, que presentan reducciones medias del

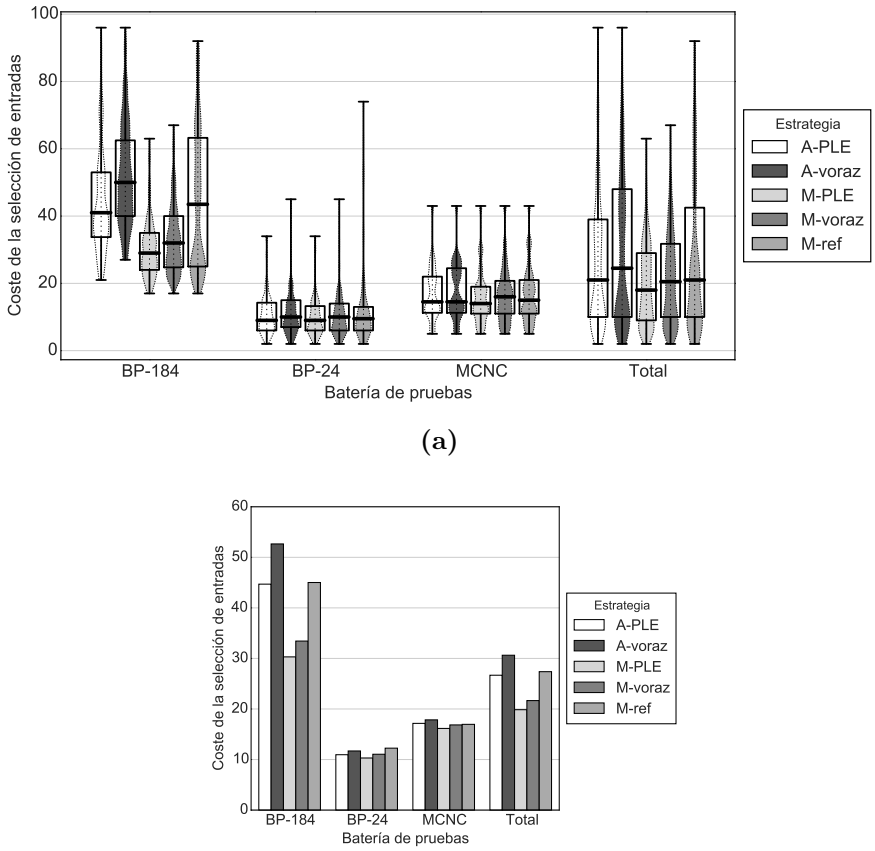



**Figura 5.39:** Coste de selección de entradas de las FSMIM-S generadas en las pruebas independientes del dispositivo: (a) distribución de los valores obtenidos, (b) valor medio.

14 % (estrategia M-PLE) y del 9 % (estrategia M-voraz). Los porcentajes de éxito se encuentran entre el 35 % de la estrategia M-voraz y el 49 % de la estrategia M-PLE, con una reducción media del 29 % para los casos de éxito en ambas estrategias. Por otra parte, los porcentajes de fracaso son inferiores al 9 %.

Es en las baterías de pruebas MCNC y BP-24 donde estas estrategias obtienen los peores resultados, con reducciones medias inferiores al 6 % y porcentajes de éxito cercanos al 30 % en la estrategia M-PLE o al 14 % en la estrategia M-voraz. Sin embargo, en la batería de pruebas BP-184 se obtienen reducciones medias entre el 18 % y el 24 %, porcentajes de éxito superiores al 59 % y porcentajes de fracaso inferiores al 7 %. Se puede ob-

servar en la figura 5.40a cómo la mitad de los valores son superiores al 20 % en ambas estrategias.

Respecto a las estrategias de tipo AM, en promedio, obtienen peores resultados que la estrategia M-ref en todas las baterías de pruebas. La estrategia A-voraz obtiene una reducción media del −10 % con el conjunto completo de FSM y un porcentaje de fracaso del 56 % con una reducción media del −25 % para los casos de fracaso. Sin embargo, la reducción media en la estrategia A-PLE es sólo del −2 %, aunque el porcentaje de fracaso es del 45 % con una reducción media del −22 % para los casos de fracaso. A pesar de ello, ambas estrategias presentan porcentajes de éxito superiores al 21 % en todas las baterías de pruebas y reducciones medias entre el 16 % y 27 % para los casos de éxito del conjunto completo de FSM.

Con objeto de comparar las estrategias respecto a las mejoras obtenidas tanto en el tamaño de la FSMIM-S como en el coste de selección de entradas, se ha calculado el porcentaje de éxito neto de la reducción RPC del tamaño de la ROM y de la reducción RPC del coste de selección de entradas obtenido por las nuevas estrategias (véase la figura 5.41). En el conjunto completo de FSM, el porcentaje de casos en los que las nuevas estrategias son mejor alternativa que M-ref varía entre el 26 % de la estrategia A-voraz y el 57 % de M-PLE. Sin embargo, el porcentaje de casos en los que la estrategia M-ref es la mejor alternativa es pequeño: varía entre 2,5 % de las estrategia M-PLE y el 10 % de A-voraz.

### 5.5.2. Pruebas de implementación

Las pruebas independientes del dispositivo han mostrado que las estrategias de tipo AM consiguen reducir el tamaño de las FSMIM-S respecto a M-ref en un porcentaje de casos elevado, obteniendo reducciones muy significativas cuando se aplican a FSM grandes. Respecto a las nuevas estrategias de tipo MA, aunque las mejoras obtenidas para las FSM pequeñas no son relevantes, estas son más significativas para FSM grandes. Además, dichas estrategias consiguen reducir tanto el coste de selección de entradas como el tamaño de la ROM en un porcentaje de casos importante.

Las pruebas de implementación presentadas en esta sección persiguen los mismos objetivos que las realizadas con la arquitectura FSMIM-T: (a) estudiar el efecto que tiene el uso de los bloques de memoria empotrados del dispositivo FPGA en las reducciones de tamaño de la FSMIM-S conseguidas por las nuevas estrategias, y (b) evaluar la influencia que tiene la reducción del coste de selección de entradas obtenida por las nuevas estrategias so-

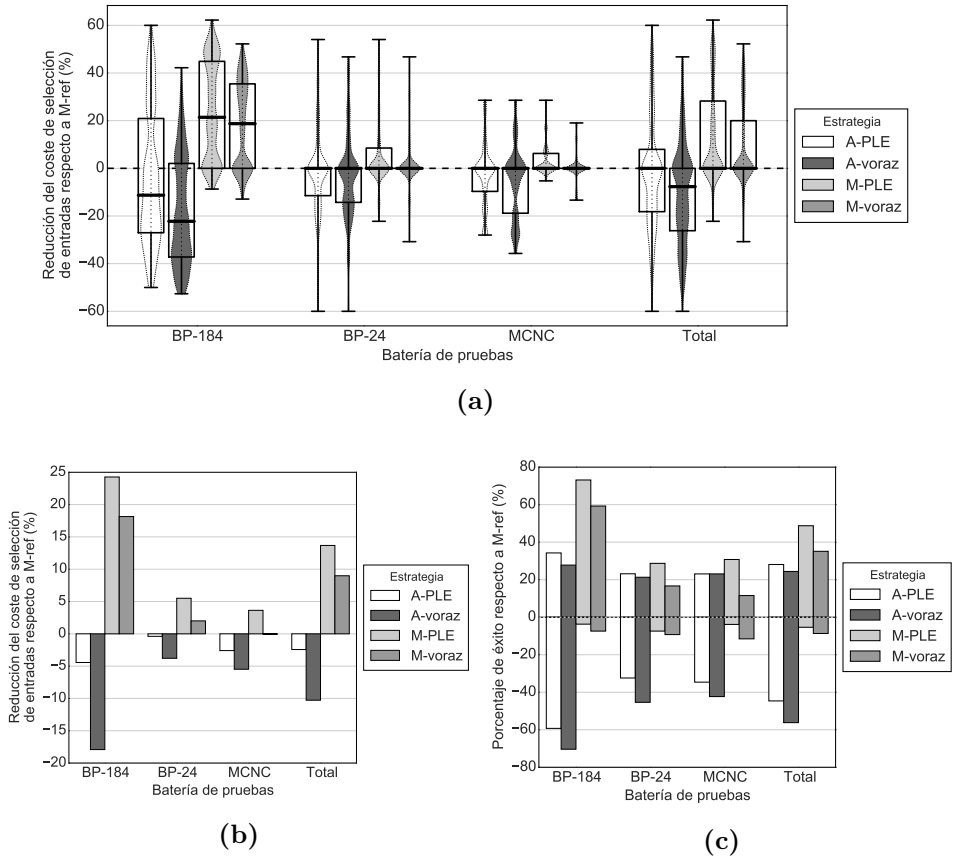



**Figura 5.40:** Reducción RPC del coste de selección de entradas de las FSMIM-S obtenida por las nuevas estrategias respecto a M-ref en las pruebas independientes del dispositivo: (a) distribución de los valores obtenidos, (b) valor medio, (c) porcentaje de éxito.

bre el consumo de LUT y sobre la frecuencia de operación máxima de las implementaciones de FSMIM-S.

El estudio comparativo de las estrategias se ha realizado a partir del conjunto completo de FSM. Para cada FSM se han generado cinco FSMIM-S con ajuste de profundidad utilizando las cinco estrategias. Posteriormente, se han realizado dos implementaciones con cada FSMIM-S usando *ISE WebPACK*: una optimizada en área y otra, en velocidad. Para el análisis de resultados se han seleccionado las FSM que cumplen las siguientes condiciones:

1. La implementación de la FSM-ROM debe ocupar más de un bloque de 9 Kbits (véase la sección 5.3.2, página 142).

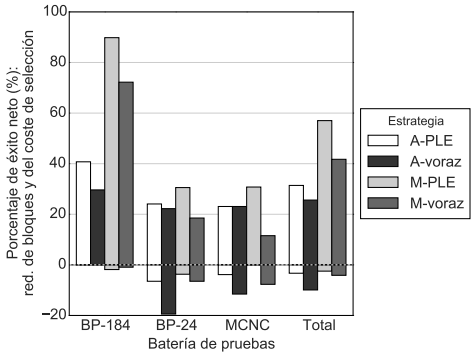



**Figura 5.41:** Porcentaje de éxito neto de la reducción RPC del tamaño de la ROM y de la reducción RPC del coste de selección de entradas de las FSMIM-S obtenido por las nuevas estrategias respecto a M-ref en las pruebas independientes del dispositivo.

2. Todas las estrategias deben generar con éxito una FSMIM-S ajuste de profundidad (véase la sección 5.3.2, página 142).

En total, se han analizado los resultados de 164 FSM (21 de la batería de pruebas MCNC, 35 de BP-24 y 108 de BP-184). El estudio se ha dividido en dos partes: la primera dedicada al análisis de los resultados de las implementaciones optimizadas en área, y la segunda dedicada a las optimizadas en velocidad. En ambas partes se analizarán tanto el área ocupada (en términos de LUT y de bloques de memoria empotrados) como la frecuencia de operación máxima.

#### 5.5.2.1. Análisis de las implementaciones optimizadas en área

**Estudio del consumo de bloques**

El resumen estadístico del número de bloques ocupados por las implementaciones FSMIM-S generadas con el conjunto completo de FSM se muestra en la tabla 5.42. Como puede observarse, el número medio de bloques es pequeño en todas las estrategias. El mejor valor medio, alcanzado por las estrategias de tipo AM, es casi un 22 % menor que el conseguido por M-ref. La figura 5.42 representa el número medio de bloques obtenido para cada batería de pruebas. Si se compara con la figura 5.36 (que representa el tamaño medio obtenido en las pruebas independientes del dispositivo), se observa que, desde un punto de vista cualitativo, los resultados son similares en promedio. En las baterías de pruebas MCNC y BP-24, los valores medios obtenidos por las diferentes estrategias son prácticamente iguales, siendo en

la batería de pruebas BP-184 donde se hacen evidentes las diferencias entre las estrategias.

Por otra parte, las diferencias entre los resultados de las estrategias basadas en PLE y las basadas en algoritmos voraces han sido muy pequeñas. Tal como se indicó anteriormente, el tamaño de las FSMIM-S generadas por las dos estrategias de tipo AM siempre coincide debido a las características de la arquitectura. Sin embargo, aunque esta propiedad no se cumple para las nuevas estrategias de tipo MA, estas han obtenido los mismos resultados en todos los casos excepto en 18, pertenecientes a la batería de pruebas BP-184.

El efecto más importante del uso de bloques de memoria sobre los resultados ha sido que el margen de mejora de las nuevas estrategias es casi inexistente en las baterías de pruebas MCNC y BP-24. Este efecto ha sido incluso mayor que en la arquitectura FSMIM-T. La tabla 5.43 muestra un resumen estadístico del número de bloques ocupados por las FSMIM-S ge-

**Tabla 5.42:** Resumen estadístico del número de bloques usados por las implementaciones FSMIM-S obtenidas con el conjunto completo de FSM en las pruebas de implementación con optimización en área. Las estrategias están ordenadas de menor a mayor número medio de bloques.

| Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| A-voraz | 6,5 | 11,7 | 0,5 | 1,0 | 2,0 | 6,5 | 78,0 |
| A-PLE | 6,5 | 11,7 | 0,5 | 1,0 | 2,0 | 6,5 | 78,0 |
| M-PLE | 7,7 | 12,1 | 0,5 | 1,0 | 3,0 | 11,0 | 78,5 |
| M-voraz | 8,0 | 12,2 | 0,5 | 1,0 | 3,5 | 11,0 | 78,5 |
| M-ref | 8,3 | 12,3 | 0,5 | 1,0 | 3,5 | 12,0 | 75,5 |

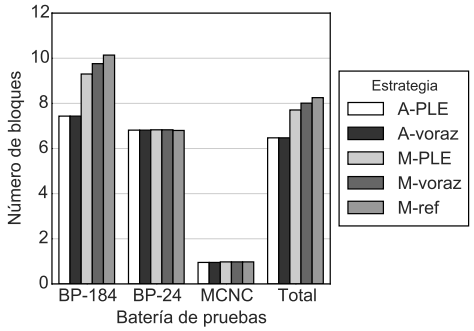



**Figura 5.42:** Número medio de bloques usados por las implementaciones FSMIM-S en las pruebas de implementación con optimización en área.

neradas con la estrategia M-ref para cada batería de pruebas. La mediana es 0,5 bloques para las baterías de pruebas MCNC y BP-24, lo que implica que hay un alto porcentaje de casos en los que no se pueden mejorar los resultados de M-ref debido al uso de bloques (exactamente, en el 52 % de las FSM de MCNC y en el 57 % de BP-24). De hecho, en la batería de pruebas MCNC, tan sólo el 14 % de los casos (3 casos) ocupan más de 1 bloque, siendo el máximo 3,5 bloques. En la batería de pruebas BP-24, este porcentaje asciende al 34 % (12 casos), de los que la mitad ocupan menos de 5 bloques.

El estudio de las reducciones obtenidas por las nuevas estrategias pone claramente de manifiesto el escaso margen de mejora existente en estas baterías de pruebas. La tabla 5.44 muestra un resumen estadístico de la reducción RPC del número de bloques obtenida por las nuevas estrategias respecto a M-ref, y la figura 5.43, una representación de dicho resumen. El consumo de bloques de las FSMIM-S generadas con las baterías de pruebas MCNC y BP-24 es prácticamente igual para todas las estrategias, es decir, la reducción es del 0 % en todos los casos excepto en dos casos en los que mejoran algunas de las nuevas estrategias y un caso en el que empeoran todas (incluso las estrategias de tipo AM). Este último caso corresponde a la FSM *fsm_rd_059*, en la que la síntesis de bajo nivel consigue reducir el número de bloques estimado de la ROM de la FSMIM-S, tal como se mencionó en la sección 5.3.3.1. A pesar de que el tamaño de la FSMIM-S generada por las estrategias de tipo AM (equivalente a 113 bloques) es menor que el de la generada por M-ref (equivalente a 116 bloques), el proceso de síntesis consigue eliminar un mayor número de bloques en la FSMIM-S generada por M-ref (ocupando 75,5 bloques) que en la generada por las estrategias de tipo AM (ocupando 78 bloques).

Por el contrario, en la batería de pruebas BP-184, el impacto del uso de bloques ha sido poco significativo debido al mayor tamaño de las FSM de esta batería de pruebas. Las reducciones medias obtenidas por las estrategias de tipo MA han sido similares a las de las pruebas independientes del dispositivo; sin embargo, el porcentaje de éxito ha disminuido. La estrategia

**Tabla 5.43:** Resumen estadístico del número de bloques usados por las implementaciones de las FSMIM-S generadas con la estrategia M-ref en las pruebas de implementación con optimización en área.

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 10,1 | 10,7 | 1,0 | 3,0 | 6,5 | 13,0 | 68,5 |
| BP-24 | 6,8 | 17,8 | 0,5 | 0,5 | 0,5 | 2,5 | 75,5 |
| MCNC | 1,0 | 0,8 | 0,5 | 0,5 | 0,5 | 1,0 | 3,5 |

M-PLE presenta una reducción media del 10 % y un porcentaje de éxito del 31 % (con una reducción media del 34 % para los casos de éxito). La estrategia M-voraz consigue un porcentaje de éxito del 20 % y un pequeño porcentaje de fracaso del 3 %. Por otra parte, en el caso de las estrategias de tipo AM, tanto las reducciones como los porcentajes de éxito han sido similares a los de las pruebas independientes del dispositivo: una reducción media del 35 % y un porcentaje de éxito del 76 % (con una reducción media para los casos de éxito del 47 %).

### Estudio del consumo de LUT

Como se ha mencionado en anteriores ocasiones, la relación entre el consumo de LUT y las características de las FSMIM es más compleja en la arquitectura FSMIM-S que en la arquitectura FSMIM-T. En primer lugar, en la arquitectura FSMIM-S existen más elementos que se implementan con LUT, ya que, además del banco de selectores de entradas y del banco de multiplexores de la ROM, esta arquitectura contiene el codificador de grupos. En segundo lugar, tanto la complejidad del banco de selectores de entrada como la del codificador de grupos dependen en gran medida de la

**Tabla 5.44:** Resumen estadístico de la reducción RPC del número de bloques usados por las implementaciones de las FSMIM-S generadas por las nuevas estrategias respecto a M-ref obtenida en las pruebas de implementación con optimización en área ( %).

| Bat. de pruebas | Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | A-PLE | 35,4 | 26,8 | 0,0 | 6,6 | 45,5 | 50,0 | 88,5 |
| | A-voraz | 35,4 | 26,8 | 0,0 | 6,6 | 45,5 | 50,0 | 88,5 |
| | M-PLE | 10,4 | 18,8 | 0,0 | 0,0 | 0,0 | 8,8 | 57,1 |
| | M-voraz | 6,0 | 17,9 | −50,0 | 0,0 | 0,0 | 0,0 | 57,1 |
| BP-24 | A-PLE | −0,0 | 0,7 | −3,2 | 0,0 | 0,0 | 0,0 | 2,9 |
| | A-voraz | −0,0 | 0,7 | −3,2 | 0,0 | 0,0 | 0,0 | 2,9 |
| | M-PLE | −0,0 | 0,8 | −3,8 | 0,0 | 0,0 | 0,0 | 2,9 |
| | M-voraz | −0,0 | 0,8 | −3,8 | 0,0 | 0,0 | 0,0 | 2,9 |
| MCNC | A-PLE | 2,4 | 10,9 | 0,0 | 0,0 | 0,0 | 0,0 | 50,0 |
| | A-voraz | 2,4 | 10,9 | 0,0 | 0,0 | 0,0 | 0,0 | 50,0 |
| | M-PLE | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| | M-voraz | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| Total | A-PLE | 23,6 | 27,5 | −3,2 | 0,0 | 6,1 | 50,0 | 88,5 |
| | A-voraz | 23,6 | 27,5 | −3,2 | 0,0 | 6,1 | 50,0 | 88,5 |
| | M-PLE | 6,8 | 16,0 | −3,8 | 0,0 | 0,0 | 0,0 | 57,1 |
| | M-voraz | 3,9 | 14,8 | −50,0 | 0,0 | 0,0 | 0,0 | 57,1 |

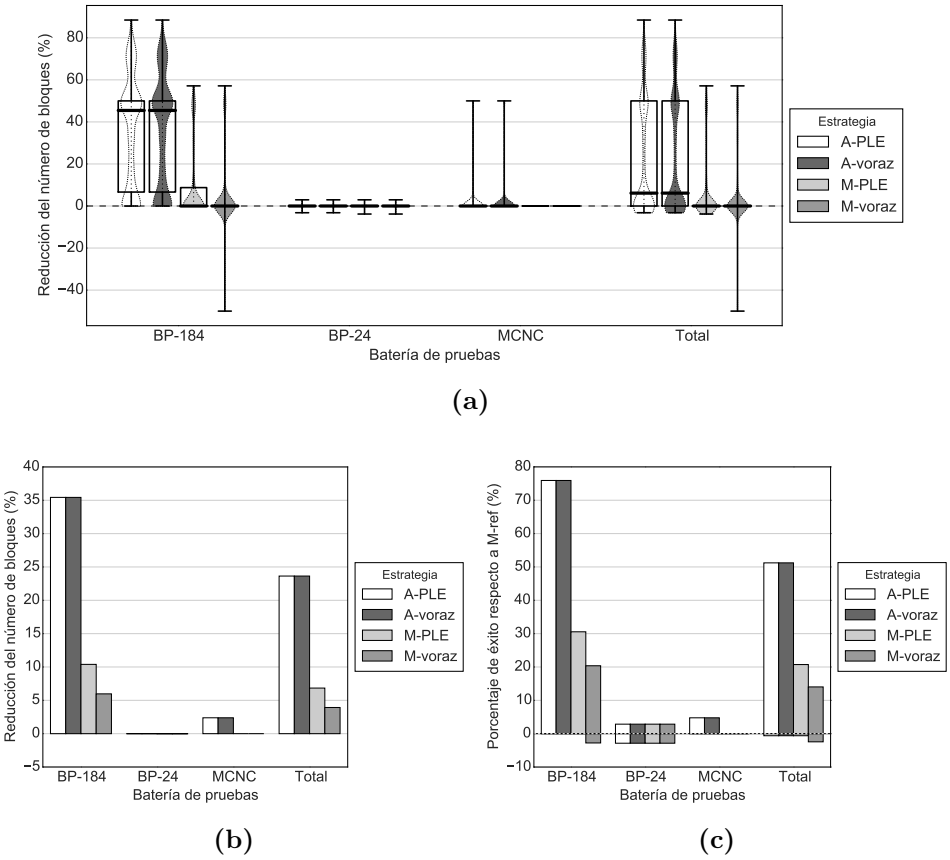



**Figura 5.43:** Reducción RPC del número de bloques usados por las implementaciones de las FSMIM-S generadas por las nuevas estrategias respecto a M-ref obtenida en las pruebas de implementación con optimización en área: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

codificación de estados y grupos, que se establece en un proceso posterior e independiente de la generación de la FSMIM-S.

El número de entradas del codificador de grupos viene determinado por el número de bits de codificación de estados, y el número de salidas, por el número de bits de codificación de grupos. Por lo tanto, la característica de la FSMIM que afecta de forma más inmediata a la complejidad del codificador de grupos es el número de grupos (ya que el número de estados no se considera una característica de las FSMIM sino de las FSM). Debido a que la reducción del número de grupos permite disminuir el número de bits de codificación de grupos, la fase de agrupación de estados favorece la

simplificación del codificador de grupos. Sin embargo, la complejidad de la función lógica asociada a cada salida del codificador depende tanto de la relación entre estados y grupos como de la codificación de estados y de grupos utilizada. Aunque la relación entre estados y grupos se establece en la fase de agrupación de estados, los algoritmos utilizados no incluyen ningún criterio para reducir la complejidad del codificador. Esto se debe principalmente a que los algoritmos se diseñaron inicialmente para la arquitectura FSMIM-T, que no tiene codificador de grupos. Puesto que la codificación de estados y grupos se realiza después de la generación de la FSMIM, las únicas características de los grupos que pueden ser consideradas por los algoritmos son el número de estados de cada grupo y el número de entradas asociadas a cada estado. El estudio de la influencia de estas características en la posterior simplificación del codificador de grupos formará parte de los trabajos futuros.

Con objeto de cuantificar la influencia del codificador de grupos en el consumo final de LUT, se ha calculado el número máximo de LUT que puede llegar a ocupar cada codificador en función de las características de la FSMIM-S. Dicha cota es muy conservadora debido a que se ha supuesto que las funciones lógicas no están simplificadas y ocupan el número máximo de LUT que corresponde al número de variables de la función (que es igual al número de bits de codificación de estados). En promedio, según el valor obtenido por esta cota superior, el consumo de LUT máximo del codificador de grupos representa, según la estrategia, entre el 20 % y el 23 % del número total de LUT usadas por las implementaciones de las FSMIM-S. Estos valores nos indican que, aunque la influencia del codificador de grupos no es despreciable, el consumo de LUT de las implementaciones viene determinado principalmente por el banco de selectores de entradas.

Debido a que el banco de selectores de entrada tiene un peso mayor en el consumo de LUT y a que el coste de selección de entradas es el único parámetro que se optimiza para reducir dicho consumo, se ha estudiado la influencia del coste de selección de entradas en el número de LUT ocupadas por las FSMIM-S. La figura 5.44 representa el número de LUT usadas frente al coste de selección de entradas de las FSMIM-S. Para distinguir los casos en los que se requiere un banco de multiplexores de la ROM, es decir, aquellos en los que la profundidad de la ROM es mayor que la profundidad máxima de los bloques de memoria del dispositivo FPGA utilizado (16 Kpalabras), se han marcado con un triángulo los puntos correspondientes. Para compensar la diferencia de escala entre el crecimiento del coste de selección y el del número de LUT, se ha utilizado una escala logarítmica para representar el número de LUT.

Las nubes de puntos de todas las estrategias indican claramente que existe una relación entre el crecimiento del consumo de LUT y el del coste de selección de entradas. Sin embargo, para un mismo coste de selección de entradas, la dispersión de los puntos correspondientes es muy grande, aumentando también dicha dispersión con el incremento del coste de selección. Estos valores demuestran la existencia de factores que tienen un efecto en la complejidad del banco de selectores de entradas mucho mayor que el del coste de selección de entradas. Como consecuencia, la influencia de la minimización del coste de selección de entradas en la reducción del consumo de LUT es menos significativa en esta arquitectura.

Respecto a los casos con profundidad mayor que 16 Kpalabras, los puntos correspondientes no muestran un comportamiento diferente al resto de casos, lo que indica que el banco de multiplexores de la ROM tiene poca influencia en el consumo de LUT. Esto se debe a que el número de LUT necesarias para implementar dicho banco de multiplexores no es significativo comparado con el número total requerido por las FSMIM-S. Este hecho puede comprobarse comparando el consumo de LUT en estos casos con el de la arquitectura FSMIM-T, que se muestra en la figura 5.22 (debe tenerse en cuenta que el banco de multiplexores de la ROM de las FSMIM-S requiere menos LUT que el de las FSMIM-T debido a que la ROM de las FSMIM-S es menor).

El consumo de LUT de las implementaciones de FSMIM-S puede consultarse en la tabla 5.45, que muestra un resumen estadístico de los resultados obtenidos por las diferentes estrategias con el conjunto completo de FSM. En promedio, las nuevas estrategias de tipo MA son las que consumen menos LUT; sin embargo, en esta arquitectura es la estrategia M-voraz la que obtiene los mejores resultados a pesar de que presenta un coste de selección mayor que el de M-PLE. Por otra parte, las estrategias de tipo AM son las que consumen un mayor número de LUT. A diferencia de lo que ocurría en las FSMIM-T, la simplificación del banco de multiplexores de la ROM debida a la reducción de la profundidad que consiguen estas estrategias tiene un impacto pequeño en la reducción del consumo de LUT.

La figura 5.45 muestra el número medio de LUT obtenido por las diferentes estrategias para cada batería de pruebas y la tabla 5.46, un resumen estadístico de los valores obtenidos por la estrategia M-ref. Tal como puede observarse, la influencia del tamaño de las FSM en el consumo de LUT es muy importante. Por ejemplo, para la estrategia M-ref, el consumo medio de LUT en la batería BP-184 es aproximadamente 20 veces mayor que el de MCNC y 10 veces mayor que el de BP-24. Estas tasas son incluso mayores en el caso de las estrategias de tipo AM.

Respecto a las reducciones en el consumo de LUT conseguidas por las

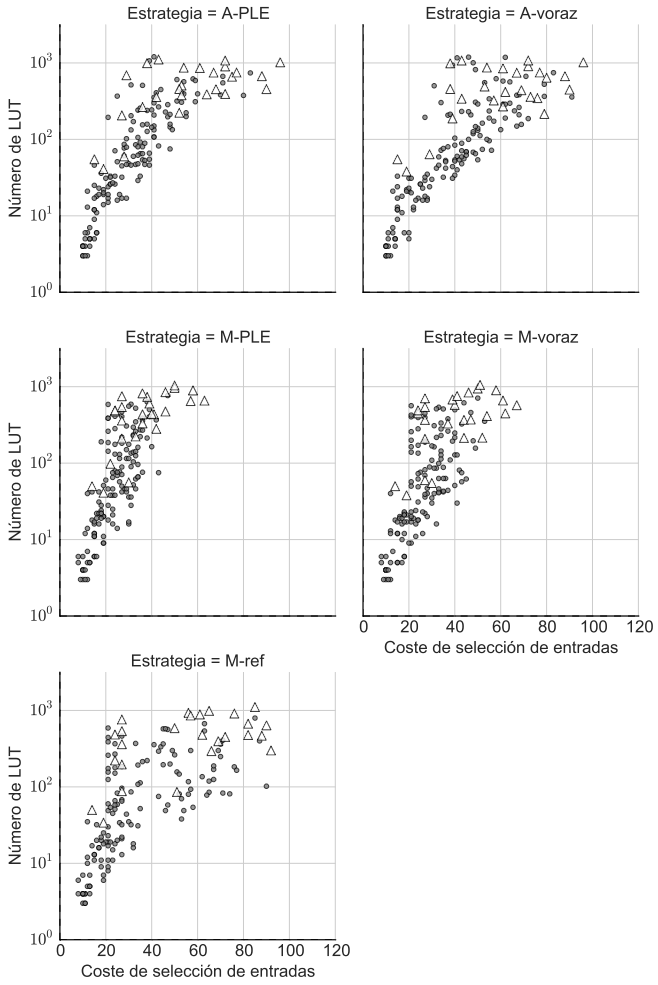



**Figura 5.44:** Número de LUT usadas por las implementaciones FSMIM-S en las pruebas de implementación con optimización en área frente al coste de selección de entradas. Los puntos marcados con un triángulo corresponden a los casos en los que la ROM de la FSMIM-S tiene una profundidad mayor de 16 Kpalabras.

nuevas estrategias respecto a M-ref, la figura 5.46 representa un resumen estadístico de las reducciones RPC obtenidas. En promedio, el valor de las reducciones que presentan las diferentes estrategias es poco significativo, variando entre el ±8 % según la estrategia. Los peores resultados los obtienen las estrategias de tipo AM, que presentan reducciones negativas y porcentajes de fracaso superiores al 51 %; por otra parte, los porcentajes de éxito

**Tabla 5.45:** Resumen estadístico del número de LUT usadas por las implementaciones FSMIM-S obtenidas con el conjunto completo de FSM en las pruebas de implementación con optimización en área. Las estrategias están ordenadas de menor a mayor número medio de LUT.

| Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| M-voraz | 172,1 | 222,8 | 3,0 | 19,0 | 62,0 | 250,5 | 1056,0 |
| M-PLE | 180,4 | 229,4 | 3,0 | 20,0 | 67,5 | 270,5 | 1036,0 |
| M-ref | 186,9 | 237,9 | 3,0 | 19,0 | 80,5 | 293,2 | 1111,0 |
| A-PLE | 213,2 | 272,8 | 3,0 | 23,8 | 80,0 | 312,5 | 1204,0 |
| A-voraz | 220,1 | 294,4 | 3,0 | 23,0 | 72,0 | 298,5 | 1191,0 |

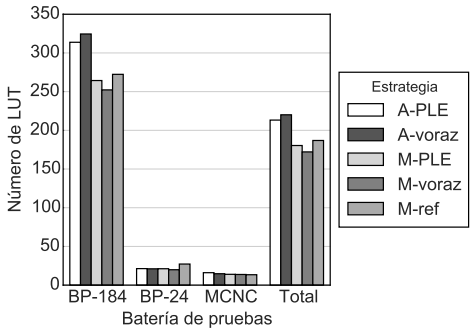



**Figura 5.45:** Número medio de LUT usadas por las implementaciones FSMIM-S en las pruebas de implementación con optimización en área.

**Tabla 5.46:** Resumen estadístico del número de LUT usadas por las implementaciones de las FSMIM-S generadas con la estrategia M-ref en las pruebas de implementación con optimización en área.

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 272,4 | 253,6 | 16,0 | 76,0 | 186,0 | 396,8 | 1111,0 |
| BP-24 | 27,2 | 28,9 | 4,0 | 6,5 | 17,0 | 36,5 | 136,0 |
| MCNC | 13,4 | 15,3 | 3,0 | 5,0 | 11,0 | 16,0 | 75,0 |

son superiores al 32 %. Los mejores resultados los obtienen las estrategias de tipo MA, que presentan reducciones medias mayores que cero y porcentajes de éxito superiores al 49 % (aunque los porcentajes de fracaso se encuentran entre el 24 % y el 35 %). Además, al contrario de lo que cabía esperar, para cada tipo de estrategia, las que están basadas en algoritmos voraces obtienen mejores resultados que la estrategia correspondiente basada en PLE.

La batería de pruebas en la que se obtienen peores resultados es MCNC,

en la que todas las estrategias aumentan el consumo de LUT. Sin embargo, en la batería de pruebas BP-24 (que es en la que se obtienen los mejores resultados), todas las estrategias consiguen reducciones superiores al 10 %, con porcentajes de éxito superiores al 24 % (llegando al 29 % en la estrategia M-voraz). En esta batería de pruebas, el 25 % de los casos presenta reducciones entre el 22 % y el 66 %, aproximadamente.

A continuación se analizará el número de LUT ahorradas con la arquitectura FSMIM-S. La tabla 5.47 muestra un resumen estadístico del NLAB obtenido por las implementaciones de las FSMIM-S generadas con el conjunto completo de FSM. En promedio, ordenadas de mayor a menor NLAB,

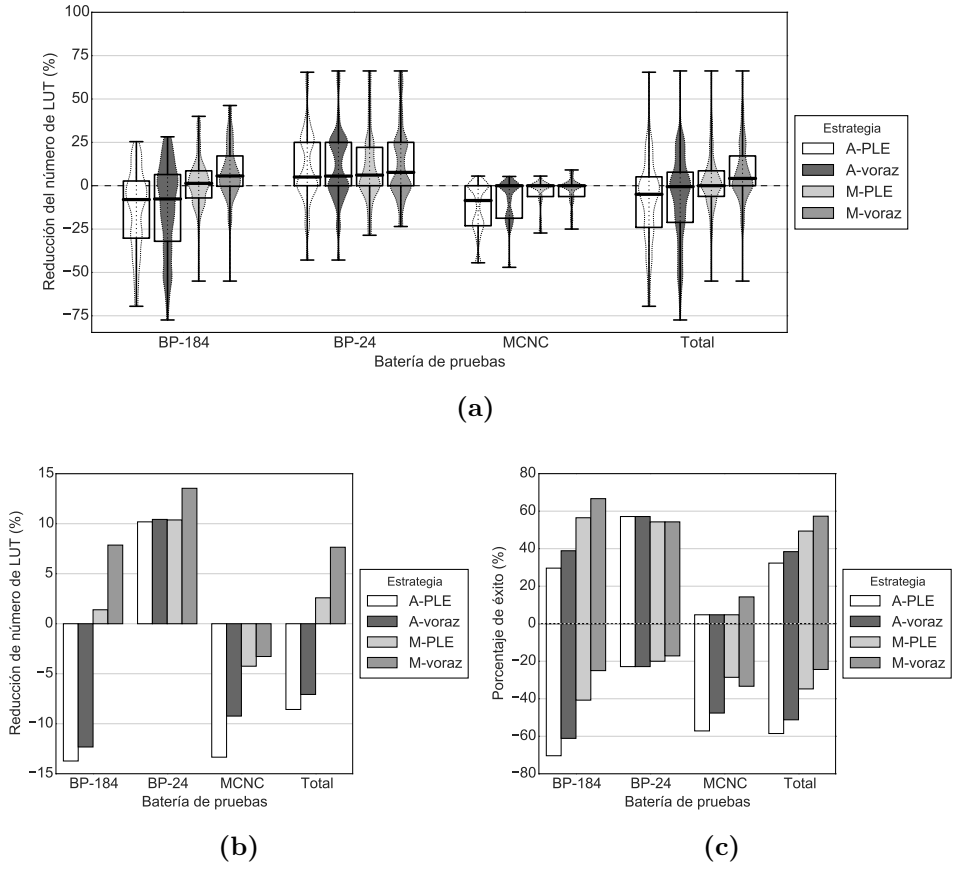



**Figura 5.46:** Reducción RPC del número de LUT usadas por las implementaciones de las FSMIM-S generadas por las nuevas estrategias respecto a M-ref obtenida en las pruebas de implementación con optimización en área: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

las primeras estrategias son las de tipo AM. En estas estrategias, el aumento del consumo de LUT se ve compensado por la reducción de bloques, lo que les permite obtener los mejores resultados. Les siguen las nuevas estrategias de tipo MA, quedando en último lugar, la estrategia M-ref. Para cada uno de los tipos anteriores, las estrategias basadas en PLE consiguen mejores resultados que las basadas en algoritmos voraces.

Respecto al NLAB de las diferentes baterías de pruebas, la tabla 5.48 muestra un resumen estadístico de los valores obtenidos por la estrategia M-ref. Por su parte, la figura 5.47 representa los valores medios conseguidos por las diferentes estrategias. Los resultados indican que el NLAB aumenta con el tamaño de las FSM. Las baterías de pruebas MCNC y BP-24 presentan valores medios próximos a las 61 y las 126 LUT/bloque, respectivamente, para todas las estrategias. Los mayores valores medios se obtienen con la batería de pruebas BP-184, con un NLAB medio que varía entre las 441 LUT/bloque de M-ref y a las 617 LUT/bloque de A-PLE. Esta es, por lo tanto, la batería de pruebas con mayores diferencias entre los valores obtenidos por las diferentes estrategias, tal como mostrará el cálculo de los incrementos obtenidos por las nuevas estrategias respecto a M-ref.

La tabla 5.49 muestra un resumen estadístico del incremento RPC del NLAB conseguido por las nuevas estrategias respecto a M-ref y la figura 5.48, una representación de dicho resumen. Claramente, los mejores resultados los obtienen las estrategias de tipo AM. Ambas presentan un incremento medio del 52 % aproximadamente para el conjunto completo de FSM y un porcentaje de éxito del 71 % (con un incremento medio próximo al 79 % para los casos de éxito); por el contrario, el porcentaje de fracaso no llega al 20 %. Las estrategias de tipo MA obtienen incrementos medios poco significativos (entre el 8 % de M-voraz y el 13 % de M-PLE). Sin embargo, los porcentajes de éxito son superiores al 60 % en ambas estrategias (con incrementos medios entre el 15 % y el 22 % para los casos de éxito).

Respecto a las baterías de pruebas, los incrementos medios obtenidos por las nuevas estrategias en MCNC y BP-24 son despreciables (con valores entre el 3 % y el −5 %). Sin embargo, el comportamiento de los porcentajes de éxito es desigual en ambas baterías de pruebas. En la batería de pruebas MCNC, los porcentajes de éxito son muy pequeños (entre el 5 % y el 14 %) y los porcentajes de fracaso son considerables (entre el 29 % y el 52 %). Por el contrario, en la batería de pruebas BP-24 el comportamiento se invierte: los porcentajes de éxito son grandes (cercanos al 60 %) mientras que los porcentajes de fracaso son pequeños (cercanos al 20 %). Los valores para la batería de pruebas BP-184 son los más significativos, especialmente para las estrategias de tipo AM, que presentan un incremento medio cercano al

**Tabla 5.47:** Resumen estadístico del NLAB obtenido por las implementaciones de las FSMIM-S generadas con el conjunto completo de FSM en las pruebas de implementación con optimización en área.

| Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| A-PLE | 437,1 | 421,7 | 2,6 | 111,5 | 299,7 | 639,3 | 1598,6 |
| A-voraz | 434,5 | 422,6 | −54,0 | 106,0 | 296,3 | 633,0 | 1598,3 |
| M-PLE | 348,6 | 369,6 | 4,6 | 91,7 | 166,9 | 514,2 | 1599,0 |
| M-voraz | 333,6 | 352,2 | 4,6 | 88,8 | 167,1 | 496,1 | 1356,8 |
| M-ref | 322,0 | 348,4 | 4,6 | 75,2 | 157,6 | 488,9 | 1356,1 |

**Tabla 5.48:** Resumen estadístico del NLAB obtenido por las implementaciones de las FSMIM-S generadas por la estrategia M-ref en las pruebas de implementación con optimización en área.

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 440,8 | 374,6 | 48,7 | 130,7 | 301,7 | 666,8 | 1356,1 |
| BP-24 | 125,7 | 127,5 | 10,0 | 35,0 | 90,0 | 167,4 | 622,0 |
| MCNC | 60,8 | 47,5 | 4,6 | 30,0 | 47,0 | 80,0 | 226,0 |

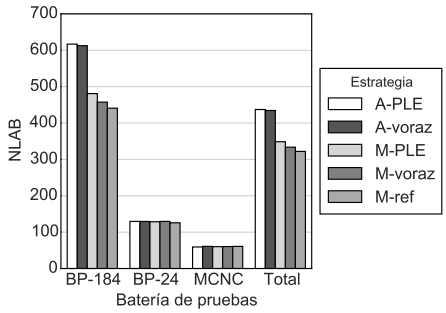



**Figura 5.47:** NLAB medio obtenido por las implementaciones FSMIM-S en las pruebas de implementación con optimización en área.

80 % y porcentajes de éxito superiores al 87 % (en el 25 % de los casos, la estrategia A-PLE presenta incrementos entre el 101 % y el 695 %, siendo del mismo orden los de la estrategia A-voraz).

### Estudio de la frecuencia de operación máxima

La influencia del tipo de optimización (en área o en velocidad) en la frecuencia de operación máxima ha sido mayor en las implementaciones FSMIM-S que en las implementaciones FSMIM-T. En promedio, las diferencias relati-

**Tabla 5.49:** Resumen estadístico del incremento RPC del NLAB obtenido por las implementaciones de las FSMIM-S generadas por las nuevas estrategias respecto a M-ref en las pruebas de implementación con optimización en área ( %).

| Bat. de pruebas | Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | A-PLE | 78,7 | 126,0 | −695,7 | 7,2 | 79,9 | 101,1 | 695,5 |
| | A-voraz | 79,6 | 107,3 | −325,9 | 6,9 | 67,8 | 101,5 | 674,6 |
| | M-PLE | 19,3 | 35,9 | −8,6 | 0,0 | 0,8 | 13,5 | 129,2 |
| | M-voraz | 11,8 | 33,8 | −96,9 | 0,0 | 0,7 | 5,1 | 124,0 |
| BP-24 | A-PLE | 3,4 | 6,7 | −6,4 | 0,0 | 1,1 | 6,2 | 28,7 |
| | A-voraz | 3,2 | 6,7 | −6,4 | 0,0 | 1,0 | 6,3 | 29,6 |
| | M-PLE | 2,5 | 6,5 | −8,7 | 0,0 | 0,7 | 4,1 | 29,6 |
| | M-voraz | 3,0 | 6,5 | −6,4 | 0,0 | 1,5 | 5,2 | 29,6 |
| MCNC | A-PLE | −5,3 | 26,1 | −77,8 | −6,2 | −2,0 | 0,0 | 70,2 |
| | A-voraz | 2,4 | 16,0 | −7,9 | −3,0 | 0,0 | 0,0 | 66,0 |
| | M-PLE | −1,3 | 2,9 | −10,9 | −0,9 | 0,0 | 0,0 | 1,2 |
| | M-voraz | −1,0 | 2,2 | −7,1 | −1,7 | 0,0 | 0,0 | 1,8 |
| Total | A-PLE | 51,2 | 108,7 | −695,7 | 0,0 | 11,1 | 94,7 | 695,5 |
| | A-voraz | 52,7 | 94,1 | −325,9 | 0,0 | 11,2 | 94,8 | 674,6 |
| | M-PLE | 12,9 | 30,4 | −10,9 | 0,0 | 0,3 | 5,9 | 129,2 |
| | M-voraz | 8,2 | 27,8 | −96,9 | 0,0 | 0,5 | 4,1 | 124,0 |

vas entre las frecuencias obtenidas con los dos tipos de optimización ha sido del 8 %. Tan sólo el 20 % de los casos ha presentado diferencias relativas superiores al 10 %.

Se ha calculado el incremento RPC de la frecuencia de operación máxima conseguido por las nuevas estrategias respecto a M-ref. La figura 5.49 muestra un resumen estadístico de los valores obtenidos. En promedio, los incrementos medios obtenidos para el conjunto completo de FSM son pequeños. Los peores resultados se obtienen para la estrategia A-voraz, con un incremento medio próximo al −10 %; sin embargo, para el resto de estrategias los incrementos medios se encuentran entre el −5 % de A-PLE y el 2 % de M-voraz.

Los valores extremos de mayor magnitud se presentan para los incrementos negativos, lo que significa que la estrategia M-ref es la que alcanza las mayores cotas de incrementos de frecuencia, especialmente en las estrategias de tipo AM. Sin embargo, el hecho de que el incremento medio para las estrategias de tipo MA sea mayor que cero indica que estas presentan una mayor proporción de incrementos positivos. En la estrategia M-PLE, aunque el porcentaje de éxito es tan sólo 5 puntos mayor que el porcenta-

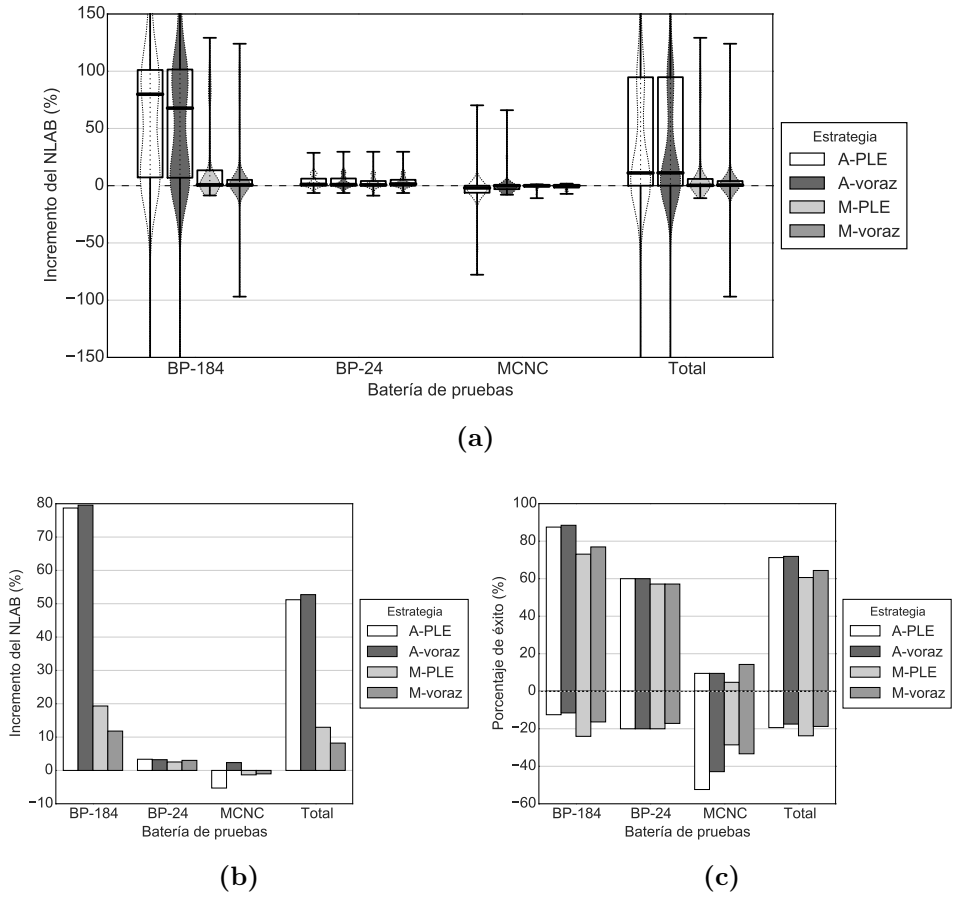



**Figura 5.48:** Incremento RPC del NLAB obtenido por las implementaciones de las FSMIM-S generadas por las nuevas estrategias respeto a M-ref en las pruebas de implementación con optimización en área: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito. Para facilitar la visualización de los datos, en (a) se han representado sólo los incrementos en el intervalo $(-150\,\%, 150\,\%)$.

je de fracaso, se obtienen incrementos superiores al 10 % en el 10 % de los casos, mientras que los incrementos inferiores al −10 % se presentan sólo en el 4 % de los casos. En promedio, los mejores resultados se obtienen con la estrategia M-voraz, que presenta un porcentaje de éxito del 62 % (23 puntos superior al porcentaje de fracaso). Además, esta estrategia alcanza incrementos superiores al 10 % en el 12 % de los casos, mientras que los incrementos inferiores al −10 % tan sólo se presentan en el 3 % de los casos.

Al igual que ocurría con el consumo de LUT, las estrategias de tipo AM

presentan peores resultados, obteniendo incrementos medios negativos. Sin embargo, los porcentajes de éxito son del mismo orden que los porcentajes de fracaso (en el caso de las estrategia A-PLE, el porcentaje de éxito es tan sólo 6 puntos inferior al porcentaje de fracaso, mientras que en A-voraz son iguales).

Respecto al análisis de las diferentes baterías de pruebas, BP-184 es en la que se obtienen los peores resultados, especialmente en las estrategias de tipo AM. En esta batería de pruebas, tan sólo la estrategia M-voraz presenta porcentajes de éxito superiores a los porcentajes de fracaso e incrementos medios mayores que cero. Los mejores resultados se obtienen en la batería de pruebas BP-24, en la que todas las estrategias obtienen incrementos medios próximos al 5 %. Además, tanto en esta batería de pruebas como en MCNC los porcentajes de éxito son superiores a los porcentajes de fracaso para todas las estrategias.

Con el fin de estudiar como afectan a la frecuencia de operación máxima las mejoras conseguidas por las nuevas estrategias respecto al NLAB y al consumo de bloques, se ha calculado, por una parte, el porcentaje de éxito neto del incremento RPC de la frecuencia de operación máxima y de la reducción RPC del número de bloques usados (figura 5.50a); y, por otra parte, el porcentaje de éxito neto del incremento RPC de la frecuencia de operación máxima y del incremento RPC del NLAB (figura 5.50b).

Para el conjunto complejo de FSM, las estrategias de tipo AM son mejor alternativa que M-ref respecto a la frecuencia y uso de bloques en el 47 % y el 50 % de los casos, mientras que el número de casos en los que M-ref es mejor alternativa no supera el 23 %. Por otra parte, las estrategias de tipo MA son mejor alternativa en el 52 % y el 61 % de los casos, mientras que M-ref es mejor alternativa en menos del 40 % de los casos. Por otra parte, respecto al porcentaje de éxito neto en frecuencia y NLAB, el porcentaje de casos en los que las nuevas estrategias son mejor alternativa que M-ref se encuentra entre el 41 % de la estrategia A-PLE y el 53 % de M-voraz. Sin embargo, el porcentaje de casos en los que M-ref es mejor alternativa es inferior al 24 % para todas las estrategias.

#### 5.5.2.2. Análisis de las implementaciones optimizadas en velocidad

El consumo de bloques de las implementaciones optimizadas en velocidad ha sido idéntico al de las implementaciones optimizadas en área. Por lo tanto, en esta sección sólo se estudiará la frecuencia de operación máxima y el consumo de LUT.

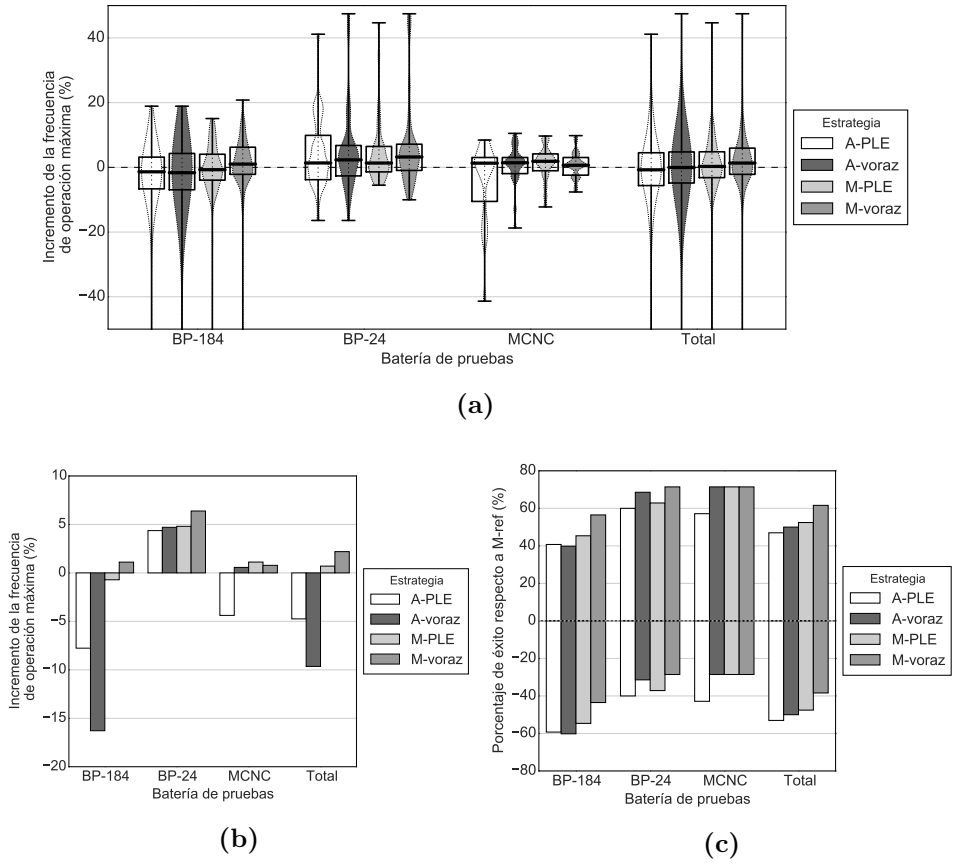
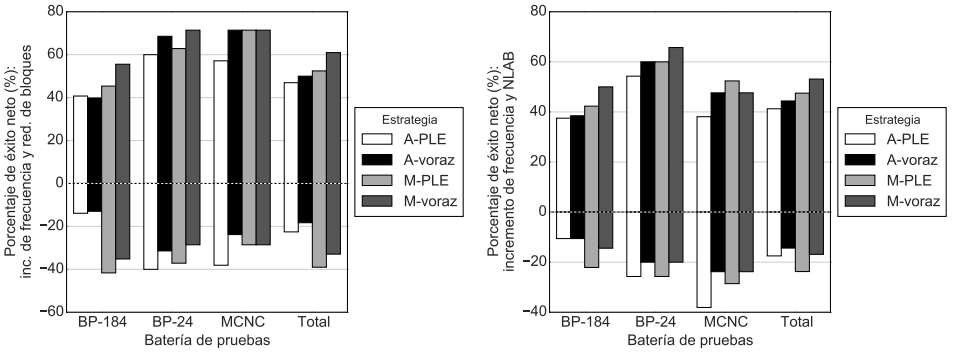



**Figura 5.49:** Incremento RPC de velocidad obtenido por las implementaciones de las FSMIM-S generadas por las nuevas estrategias respecto a M-ref en las pruebas de implementación con optimización en área: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito. Para facilitar la visualización de los datos, en (a) se han representado sólo los incrementos en el intervalo $(-50\,\%, 50\,\%)$. De izquierda a derecha, los valores mínimos excluidos para BP-184 son $-199\,\%$, $-297\,\%$, $-96\,\%$ y $-96\,\%$ (los valores para el conjunto completo de FSM son idénticos).

### Estudio de la frecuencia de operación máxima

El resumen estadístico de las frecuencias de operación máximas obtenidas con el conjunto completo de FSM se muestra en la tabla 5.50. Por su parte, la tabla 5.51 muestra el resumen estadístico de las frecuencias obtenidas por M-ref en cada batería de pruebas. Las frecuencias alcanzadas por las diferentes estrategias se encuentran entre los 53 y los 292 MHz. Las diferencias entre las frecuencias medias son despreciables tanto para el conjunto com-




**(a)** **(b)**

**Figura 5.50:** Porcentaje de éxito neto en la mejora de velocidad y de otros parámetros de las implementaciones de las FSMIM-S obtenido por las nuevas estrategias respecto a M-ref en las pruebas de implementación con optimización en área: (a) incremento RPC de velocidad y reducción RPC del número de bloques, y (b) incremento RPC de velocidad e incremento RPC del NLAB.

pleto de FSM (no superan los 5 MHz) como para cada batería de pruebas por separado (véase la figura 5.51, que muestra los valores medios obtenidos con cada batería de pruebas). Las frecuencias más bajas se obtienen en la batería de pruebas BP-184 y las más altas en MCNC, presentando frecuencias medias un 70 % mayores que las de BP-184. En la batería de pruebas BP-24 se obtienen frecuencias ligeramente inferiores a las de MCNC.

Comparado con los resultados de las implementaciones optimizadas en área, la optimización en velocidad ha reducido la distancia entre las frecuencias obtenidas por las diferentes estrategias. La tabla 5.52 y la figura 5.52 muestran un resumen estadístico del incremento RPC de velocidad obtenido por las implementaciones de las FSMIM-S generadas con las nuevas estrategias respecto a M-ref. La diferencia entre los incrementos máximos y mínimos se ha reducido respecto a los obtenidos con la optimización en área. Igualmente, los incrementos medios son ahora despreciables en todos los casos: en el conjunto completo de FSM varían entre el $\pm 2\,\%$ y en ninguna batería de pruebas supera el $\pm 4\,\%$.

La magnitud de los incrementos negativos indica que la estrategia M-ref sigue siendo la que alcanza mayores incrementos. Sin embargo, el número de casos con incrementos superiores al 10 % es mayor para las estrategias de tipo MA que para M-ref. La estrategia M-PLE consigue incrementos superiores al 10 % en el 7 % de los casos, mientras que son inferiores al $-10\,\%$ sólo en el 4 % de los casos. Por su parte, la estrategia M-voraz alcanza incremen-

**Tabla 5.50:** Resumen estadístico de la frecuencia de operación máxima (MHz) obtenida por las implementaciones FSMIM-S con el conjunto completo de FSM en las pruebas de implementación con optimización en velocidad. Las estrategias están ordenadas de mayor a menor frecuencia.

| Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| M-voraz | 169,6 | 56,5 | 60,9 | 124,7 | 165,8 | 209,7 | 292,4 |
| M-PLE | 167,9 | 55,4 | 61,1 | 126,5 | 161,1 | 207,7 | 289,3 |
| M-ref | 167,1 | 55,9 | 58,4 | 124,1 | 159,4 | 204,2 | 280,6 |
| A-voraz | 166,0 | 54,2 | 54,7 | 126,6 | 158,6 | 197,3 | 292,4 |
| A-PLE | 164,5 | 54,4 | 52,7 | 123,9 | 155,7 | 202,2 | 292,4 |

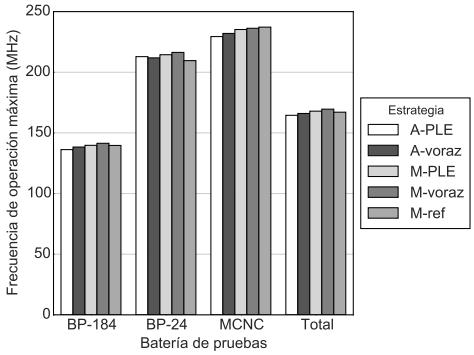



**Figura 5.51:** Valores medios de la frecuencia de operación máxima obtenida por las implementaciones FSMIM-S en las pruebas de implementación con optimización en velocidad.

**Tabla 5.51:** Resumen estadístico de la frecuencia de operación máxima (MHz) obtenida por las implementaciones FSMIM-S generadas con la estrategia M-ref en las pruebas de implementación con optimización en velocidad.

| Bat. de pruebas | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| BP-184 | 139,7 | 36,4 | 58,4 | 113,4 | 139,7 | 162,1 | 225,4 |
| BP-24 | 209,6 | 55,2 | 85,7 | 180,4 | 210,4 | 256,1 | 280,6 |
| MCNC | 237,2 | 28,6 | 169,4 | 215,5 | 247,0 | 259,0 | 269,2 |

tos superiores al 10 % en el 8 % de los casos, mientras que los incrementos inferiores al $-10$ % se presentan tan sólo en el 2 % de los casos. Además, excepto para la estrategia A-PLE, los porcentajes de éxito de las nuevas estrategias son mayores que los porcentajes de fracaso. La mayor diferencia se obtiene en la estrategia M-voraz, que consigue un porcentaje de éxito del 63 % mientras que el porcentaje de fracaso es del 36 %. Este mismo com-

portamiento se observa en la batería de pruebas BP-184. Por otra parte, mientras que en la batería de pruebas MCNC los porcentajes de éxito son inferiores a los porcentajes de fracaso, la relación se invierte en BP-24.

Los datos anteriores muestran que, aunque las diferencias son muy poco significativas, las mejores estrategias ordenadas de mayor a menor incremento de velocidad y porcentaje de éxito son M-voraz, M-PLE, A-voraz y A-PLE. Por lo tanto, parece que las estrategias basadas en algoritmos voraces obtienen mejores resultados que las estrategias correspondientes basadas en PLE a pesar de que presentan mayor coste de selección de entradas.

Para finalizar, se estudiará el porcentaje de casos en los que las nuevas estrategias son mejor alternativa que M-ref respecto a la frecuencia de operación máxima y el número de bloques usados, es decir, respecto a los dos parámetros principales en las implementaciones de FSMIM-S optimizadas en velocidad. La figura 5.53 muestra el porcentaje de éxito neto del incremento RPC de velocidad y de la reducción RPC del consumo de bloques obtenido por las nuevas estrategias respecto a M-ref.

En el conjunto completo de FSM, el porcentaje de éxito neto de las nuevas estrategias es mayor que el porcentaje de fracaso neto. Las mayores

**Tabla 5.52:** Resumen estadístico del incremento RPC de velocidad obtenido por las implementaciones de las FSMIM-S generadas por las nuevas estrategias respecto a M-ref en las pruebas de implementación con optimización en velocidad ( %).

| Bat. de pruebas | Estrategia | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | A-PLE | −2,4 | 13,4 | −66,9 | −7,4 | −0,8 | 4,6 | 27,6 |
| | A-voraz | −0,8 | 12,7 | −57,5 | −5,6 | 0,5 | 6,4 | 27,6 |
| | M-PLE | 0,2 | 9,1 | −61,7 | −2,1 | 0,6 | 4,0 | 24,9 |
| | M-voraz | 1,0 | 8,8 | −55,9 | −1,2 | 1,3 | 4,1 | 23,1 |
| BP-24 | A-PLE | 1,9 | 9,2 | −23,6 | −1,9 | 1,9 | 5,6 | 24,6 |
| | A-voraz | 1,4 | 11,4 | −35,4 | −1,7 | 2,9 | 5,3 | 24,6 |
| | M-PLE | 2,8 | 6,7 | −6,8 | −1,2 | 1,9 | 4,3 | 20,6 |
| | M-voraz | 3,7 | 8,0 | −12,4 | −0,5 | 1,9 | 6,0 | 24,6 |
| MCNC | A-PLE | −3,7 | 7,3 | −20,0 | −6,5 | −4,0 | −0,9 | 8,9 |
| | A-voraz | −2,4 | 6,0 | −11,7 | −6,5 | −3,1 | 0,6 | 8,1 |
| | M-PLE | −0,8 | 5,0 | −9,9 | −4,0 | −0,9 | 0,6 | 9,6 |
| | M-voraz | −0,4 | 4,4 | −7,4 | −3,5 | −0,8 | 0,9 | 7,9 |
| Total | A-PLE | −1,6 | 12,0 | −66,9 | −6,3 | −0,8 | 4,4 | 27,6 |
| | A-voraz | −0,6 | 11,8 | −57,5 | −5,4 | 0,6 | 5,4 | 27,6 |
| | M-PLE | 0,6 | 8,3 | −61,7 | −2,1 | 0,6 | 3,9 | 24,9 |
| | M-voraz | 1,4 | 8,2 | −55,9 | −1,2 | 1,0 | 4,2 | 24,6 |

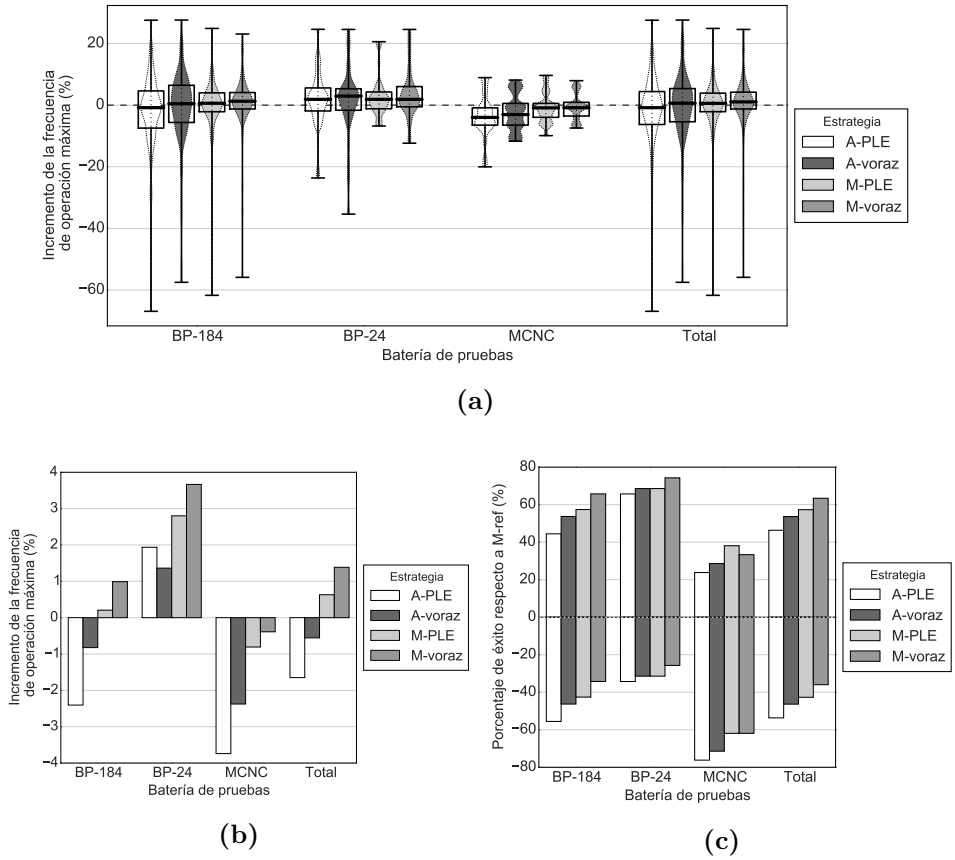



**Figura 5.52:** Incremento RPC de velocidad obtenido por las implementaciones de las FSMIM-S generadas por las nuevas estrategias respecto a M-ref en las pruebas de implementación con optimización en velocidad: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

diferencias se dan en las estrategias basadas en algoritmos voraces, en las que los porcentajes de éxito son el doble que los porcentajes de fracaso. Este mismo comportamiento se presenta en las baterías de pruebas BP-24 y BP-184; sin embargo, en la batería de pruebas MCNC, la relación entre los porcentajes de éxito y fracaso se invierte para todas las estrategias.

### Estudio del consumo de LUT

La comparativa entre las nuevas estrategias y M-ref se realizará utilizando solamente los indicadores que se han considerado más importantes. El estudio comenzará con el análisis de la reducción del consumo de LUT. La

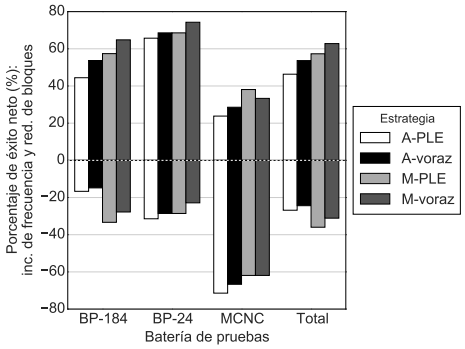



**Figura 5.53:** Porcentaje de éxito neto del incremento RPC de velocidad y de la reducción RPC del consumo de bloques obtenido por las nuevas estrategias respecto a M-ref en las pruebas de implementación de FSMIM-S con optimización en velocidad.

figura 5.54 muestra un resumen estadístico de la reducción RPC del número de LUT obtenida por las nuevas estrategias respecto a M-ref.

Comparados con los resultados de las implementaciones optimizadas en área, se observan escasas diferencias (véase la figura 5.24), a pesar de que la distancia relativa media entre los resultados obtenidos con ambos objetivos de optimización es del 13 %. El único cambio que se aprecia es una ligera mejora de los resultados obtenidos por las estrategias de tipo MA en la batería de pruebas MCNC, tanto en los incrementos medios como en los porcentajes de éxito.

En el conjunto completo de FSM, los peores resultados los obtienen las estrategias de tipo AM, con reducciones medias negativas y porcentajes de éxito inferiores a los porcentajes de fracaso (aunque en el caso de A-voraz, sólo hay 10 puntos de diferencia). Los mejores resultados se presentan en las estrategias de tipo MA, con reducciones medias positivas, porcentajes de éxito entre el 49 % y el 60 %, y porcentajes de fracaso inferiores al 35 %. La estrategia M-voraz presenta en todas las baterías de pruebas reducciones medias mayores que cero y porcentajes de éxito superiores a los porcentajes de fracaso (en las baterías de pruebas BP-184 y BP-24 el porcentaje de éxito es casi el triple que el porcentaje de fracaso).

Al igual que ocurría con las reducciones en el consumo de LUT, el impacto de la optimización en velocidad sobre el NLAB también es poco significativo comparado con las implementaciones optimizadas en área. La figura 5.55 muestra un resumen estadístico del incremento RPC del NLAB de las implementaciones FSMIM-S. Los mejores resultados los obtienen las estrategias de tipo AM, con incrementos medios próximos al 60 % para el

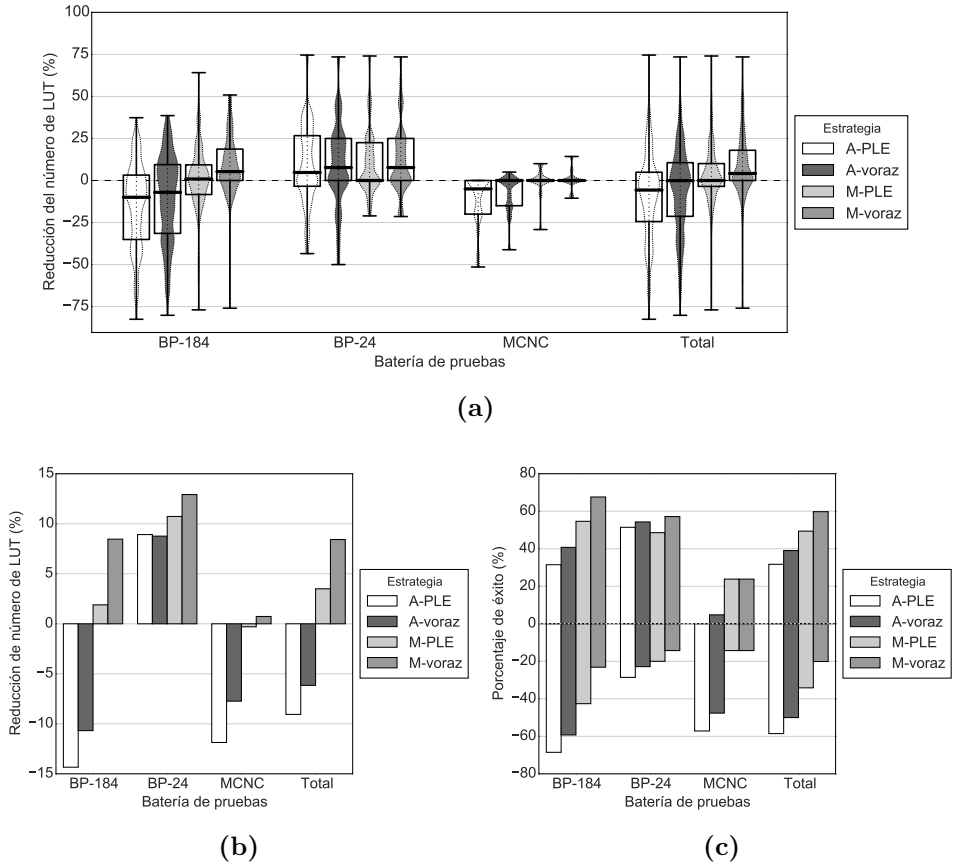



**Figura 5.54:** Reducción RPC del número de LUT usadas por las implementaciones de las FSMIM-S generadas por las nuevas estrategias respecto a M-ref obtenida en las pruebas de implementación con optimización en velocidad: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

conjunto completo de FSM y superiores al 89 % para la batería de pruebas BP-184. Los incrementos medios son despreciables en el resto de baterías de pruebas. Además, en todas las baterías de pruebas excepto MCNC, el porcentaje de éxito de estas estrategias es superior al porcentaje de fracaso.

Las estrategias de tipo MA obtienen incrementos medios moderados en el conjunto completo de FSM. Sin embargo, los incrementos son superiores al 11 % en la batería de pruebas BP-184 y, además, obtienen porcentajes de éxito superiores a los porcentajes de fracaso en todas las baterías de pruebas.

Para finalizar el estudio, se analizará el porcentaje de casos en los que las nuevas estrategias son mejor alternativa que M-ref respecto a la velocidad y

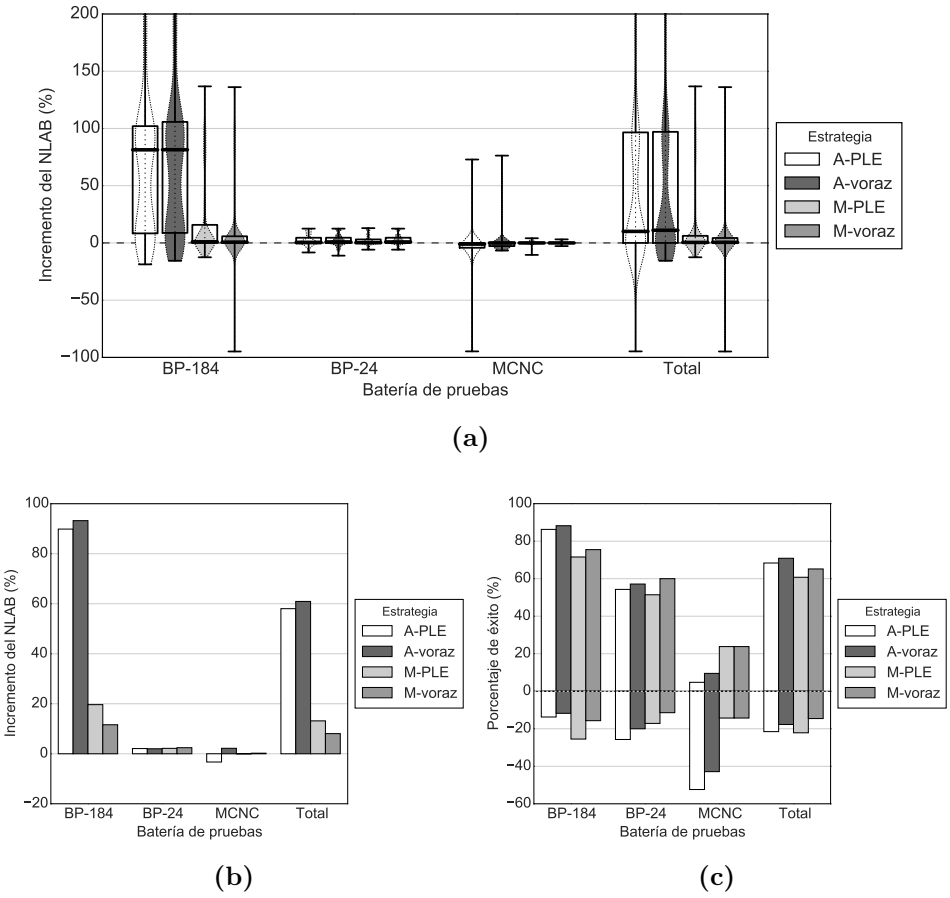



**Figura 5.55:** Incremento RPC del NLAB obtenido por las implementaciones de las FSMIM-S generadas por las nuevas estrategias respeto a M-ref en las pruebas de implementación con optimización en velocidad: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito. Para facilitar la visualización de los datos, en (a) se han representado sólo los incrementos en el intervalo $(-100\,\%, 200\,\%)$. De izquierda a derecha, los valores máximos excluidos para BP-184 son 677 % y 686 % (los valores para el conjunto completo de FSM son los mismos).

el NLAB. La figura 5.56 muestra el porcentaje de éxito neto del incremento RPC de velocidad y del incremento RPC del NLAB obtenido por las nuevas estrategias respecto a M-ref. El porcentaje de éxito neto de las nuevas estrategias varía entre el 42 % y el 56 % para el conjunto completo de FSM, mientras que los porcentajes de fracaso neto no superan el 25 %. Los peores resultados se presentan en la batería de pruebas MCNC, en la que los porcentajes de fracaso neto son mayores que los porcentajes de éxito (aunque

en la estrategia M-PLE están casi igualados). Sin embargo, en el resto de baterías de pruebas, son los porcentajes de éxito neto los que superan a los porcentajes de fracaso.

### 5.5.3. Conclusiones

El objetivo fundamental de la arquitectura FSMIM-S es obtener implementaciones FSMIM que requieran un menor consumo de memoria que las implementaciones FSMIM-T. Para conseguirlo, el banco de selectores de entradas de la FSMIM-S realiza parte de las funciones lógicas que las FSMIM-T implementan en la ROM, lo que supone un incremento del consumo de LUT. Sin embargo, tal como se ha mostrado en la sección 5.3.4, a pesar de que usa más LUT que la arquitectura FSMIM-T, la arquitectura FSMIM-S hace un uso más eficiente de los recursos disponibles en la FPGA, ya que consigue ahorrar un mayor número de LUT por cada bloque usado. Por lo tanto, los dos objetivos más importantes de las estrategias de optimización de FSMIM-S son, primero, la minimización del consumo de bloques, y segundo, la maximización del NLAB.

Tal como ha mostrado el estudio experimental, por una parte, las únicas estrategias de optimización de FSMIM-S que permiten alcanzar completamente el primer objetivo son las de tipo AM, ya que las FSMIM generadas por estas estrategias siempre tienen el tamaño mínimo, lo que las convierte en la mejor opción desde el punto de vista del consumo de memoria. Esta propiedad se ha visto reflejada en los resultados de las pruebas independientes del dispositivo, en las que han obtenido un porcentaje de fracaso del 0 %

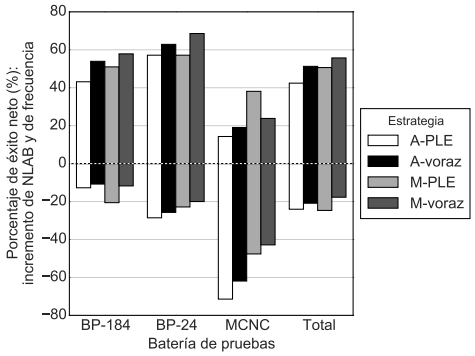



**Figura 5.56:** Porcentaje de éxito neto del incremento RPC de velocidad y del incremento RPC del NLAB obtenido por las nuevas estrategias respecto a M-ref en las pruebas de implementación de FSMIM-S con optimización en velocidad.

en la reducción del tamaño de la ROM respecto a M-ref y el mayor porcentaje de éxito (el 62 %). En las pruebas de implementación todas las estrategias han conseguido obtener una FSMIM-S de tamaño mínimo en la mitad de los casos. Como consecuencia, el porcentaje de éxito de las estrategias de tipo AM ha bajado al 50 % aproximadamente; sin embargo, la reducción media para los casos de éxito ha sido significativa (del 47 %).

Por otra parte, las estrategias de tipo AM consiguen la mayor reducción de bloques a expensas de sufrir también el mayor incremento de LUT. Sin embargo, a pesar de usar más LUT que el resto de estrategias, son las que consiguen el mayor NLAB medio (en este sentido, entre las estrategias de tipo AM y el resto de estrategias se observa el mismo comportamiento que entre las arquitecturas FSMIM-S y FSMIM-T). Estas estrategias consiguen un incremento medio del NLAB respecto a M-ref cercano al 51 %, con incrementos superiores al 94 % en el 25 % de los casos. Por lo tanto, en los casos en los que el resto de estrategias no consigue generar una FSMIM-S de tamaño mínimo, las estrategias de tipo AM también son la mejor opción respecto al uso eficiente de los bloques de memoria.

En cuanto a las estrategias de tipo MA, la principal ventaja que presentan frente a las estrategias de tipo AM es un menor consumo de LUT. A pesar de que la reducción del número de LUT no está entre los objetivos mencionados anteriormente, esta característica puede ser útil en dos escenarios. Por una parte, las estrategias de tipo MA permiten aumentar el número de alternativas disponibles para alcanzar los objetivos de diseño respecto a las restricciones de área. Por otra parte, en aquellos casos en los que las estrategias de tipo MA consiguen generar una FSMIM-S de tamaño mínimo, estas son mejor alternativa que las estrategias de tipo AM tanto en el número de recursos consumidos (consumen menos LUT y el mismo número de bloques) como en el uso eficiente de los bloques de memoria (obtienen mayor NLAB en estos casos). En promedio, la estrategia M-voraz es la que obtiene los mejores resultados en el consumo de LUT, siendo mejor opción que M-ref en un porcentaje considerable de los casos.

## 5.6. Estudio comparativo entre implementaciones FSMIM y FSM

En esta sección se realiza un estudio comparativo entre las prestaciones obtenidas con las FSMIM generadas por las nuevas estrategias y las obtenidas con las implementaciones convencionales de FSM. En el estudio se incluye además la arquitectura basada en memoria proporcionada por *ISE WebPACK*. En concreto, los resultados obtenidos en las pruebas de implementación realizadas con las arquitecturas FSMIM-T y FSMIM-S (analizadas en las secciones 5.4.2 y 5.5.2, respectivamente) serán comparados con los de las implementaciones ISE-LUT, ISE-BLOQ y FSM-ROM (véase la sección 5.2.2), optimizadas tanto en área como en velocidad.

Del conjunto completo de FSM, se han seleccionado inicialmente los resultados de las que cumplen las siguientes condiciones:

1. La implementación de la FSM-ROM debe ocupar más de un bloque de 9 Kbits (véase la sección 5.3.2, página 142).

2. Todas las estrategias deben generar con éxito una FSMIM con ajuste de profundidad (véase la sección 5.3.2, página 142).

Se deben excluir de esta selección inicial las FSM para las que no sea posible obtener alguna de las implementaciones comparadas por requerir más recursos que los disponibles en el dispositivo FPGA utilizado. Para las 162 FSM que cumplen las condiciones anteriores, la arquitectura FSMIM-S ha podido implementarse en el dispositivo. Sin embargo, debido a que requieren más de 172 bloques de memoria (el máximo disponible en el dispositivo FPGA), la implementación ISE-BLOQ no es posible en 105 casos; la implementación FSM-ROM, en 60 casos y la implementación FSMIM-T, en un caso. Por otra parte, la implementación ISE-LUT no ha sido posible en cinco casos debido al elevado consumo de LUT. Puesto que la implementación ISE-LUT no utiliza bloques de memoria, no será tenida en cuenta en el estudio del consumo de bloques. Por lo tanto, el número final de FSM incluidas dependerá del parámetro analizado (frecuencia de operación máxima, consumo de bloques o consumo de LUT).

Para evitar una disminución excesiva del número de casos analizados, se ha optado por no incluir ISE-BLOQ en el estudio detallado de los resultados, aunque sí se evaluarán sus prestaciones en este capítulo. Por el contrario, la arquitectura FSM-ROM sí permanecerá en el estudio detallado debido a que es la arquitectura de referencia para las implementaciones de FSM en bloques de memoria.

Con objeto de simplificar la comparación con las implementaciones FSMIM, se ha decidido reducir el número de estrategias de optimización de FSMIM utilizadas en el estudio. Para ambas arquitecturas, se han seleccionado las estrategias de tipo AM y de tipo MA con mejores prestaciones en el consumo de bloques. En el caso de la arquitectura FSMIM-S, puesto que las dos estrategias de tipo AM obtienen idénticos resultados en el uso de bloques, se ha elegido la que consigue mayores reducciones en el consumo de LUT respecto a la implementación ISE-LUT. Las estrategias seleccionadas han sido las basadas en PLE; por lo tanto, se incluirán en el estudio los resultados de implementación de las FSMIM generadas exclusivamente con las estrategias A-PLE y M-PLE.

Las implementaciones FSMIM-S obtenidas con las estrategias A-PLE y M-PLE serán denominadas FSMIM-SA y FSMIM-SM, respectivamente. De igual manera, las implementaciones FSMIM-T obtenidas con las estrategias A-PLE y M-PLE serán denominadas FSMIM-TA y FSMIM-TM, respectivamente.

El resto de la sección se divide en dos partes. En la primera se compara tanto el consumo de LUT y de bloques de memoria como la frecuencia de operación máxima de las implementaciones optimizadas en área. En la segunda parte, dedicada a las implementaciones optimizadas en velocidad, se compara solamente el consumo de LUT y la frecuencia de operación máxima debido a que el consumo de bloques no se ha visto afectado por el cambio del objetivo de optimización.

### 5.6.1. Análisis de las implementaciones optimizadas en área

#### 5.6.1.1. Estudio del consumo de bloques de memoria

Como se ha mencionado en la introducción de la sección, las implementaciones ISE-LUT no serán tenidas en cuenta en este estudio debido a que no utilizan bloques de memoria. Por lo tanto, el conjunto de FSM para el estudio detallado está formado por los casos en los que las implementaciones FSMIM y FSM-ROM han tenido éxito (que suman 102 FSM). La arquitectura FSM-ROM será utilizada como referencia debido a que, desde el punto de vista teórico, es la que determina el tamaño máximo que debe tener la ROM de las implementaciones de FSM basadas en memoria. Es decir, desde el punto de vista del consumo de bloques, las implementaciones de FSM que ocupen más bloques que FSM-ROM deben descartarse en favor de esta. La tabla 5.53 muestra un resumen estadístico del número de bloques usados y la figura 5.57, los valores medios.

El número de bloques requerido por la implementación FSM-ROM para el conjunto completo de FSM varía entre 1 y 171 bloques, abarcando prácticamente el intervalo completo de valores posibles para el dispositivo utilizado (que dispone de 172 bloques). En promedio, esta arquitectura utiliza 40 bloques aproximadamente (el 23 % de los bloques disponibles), mientras que la implementación FSMIM que requiere más bloques utiliza tan solo 9,4 (el 5 % de los bloques disponibles). Si se mide el tamaño de las FSM como el número de bloques ocupados por la implementación FSM-ROM, las baterías de pruebas ordenadas de menor a mayor tamaño medio de sus FSM son MCNC (con un tamaño medio de 10,5 bloques), BP-24 (que duplica el tamaño medio de MCNC) y BP-184 (que triplica el tamaño medio de BP-24). El análisis de los resultados obtenidos en cada batería de pruebas indica que las diferencias entre el consumo de bloques de las implementaciones aumenta con el tamaño de las FSM.

Respecto a la implementación ISE-BLOQ, esta es la que presenta el mayor consumo de bloques, fracasando en 45 FSM del conjunto completo (el 44 % de los casos) por requerir más de 172 bloques. Excluyendo estas 45 FSM, la implementación ISE-BLOQ requiere 36,5 bloques de media, mientras que la implementación FSM-ROM usa 13 bloques y las implementaciones FSMIM, menos de 4 bloques.

Para evaluar las mejoras obtenidas con las implementaciones FSMIM, se ha calculado la reducción RPC del número de bloques usados respecto a FSM-ROM. La tabla 5.54 muestra un resumen estadístico de los valores obtenidos y la figura 5.58, una representación de dicho resumen. Como puede observarse, para el conjunto completo de FSM, todas las implementaciones FSMIM alcanzan reducciones medias importantes que varían entre el 67,3 % de FSMIM-TM y el 78 % de FSMIM-SA. Además, todas consiguen reducciones por encima del 90 % en el 25 % de los casos.

Las mayores diferencias entre las reducciones medias obtenidas por las implementaciones FSMIM se producen en la batería de pruebas BP-184, que presenta valores entre el 67,7 % de FSMIM-TM y el 84,9 % de FSMIM-SA. Por el contrario, las diferencias entre las reducciones medias son menores en las baterías de pruebas MCNC (con reducciones entre el 76,3 % y el 78 %) y BP-24 (con valores entre el 61,5 % y el 65,7 %).

En contra de lo que cabía esperar, la implementación ISE-BLOQ usa más bloques que la implementación FSM-ROM en prácticamente todos los casos. Tan solo consigue mejorar el consumo de bloques en un caso, consumiendo en el resto de casos aproximadamente el 50 % más bloques que FSM-ROM. Estos datos parecen indicar que, desde un punto de vista cualitativo, la arquitectura utilizada por la herramienta *ISE WebPACK* para

**Tabla 5.53:** Resumen estadístico del número de bloques usados por las implementaciones FSM-ROM y FSMIM optimizadas en área. Para cada batería de pruebas, las implementaciones están ordenadas de menor a mayor número medio de bloques.

| Bat. de pruebas | Implement. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-SA | 5,4 | 7,6 | 0,5 | 1,0 | 3,0 | 6,5 | 44,0 |
| | FSMIM-SM | 7,2 | 8,5 | 1,0 | 1,5 | 5,5 | 11,0 | 49,5 |
| | FSMIM-TA | 11,4 | 19,0 | 1,0 | 1,8 | 5,5 | 12,8 | 112,0 |
| | FSMIM-TM | 13,5 | 20,1 | 1,0 | 3,0 | 8,5 | 14,5 | 123,0 |
| | FSM-ROM | 59,8 | 48,6 | 6,0 | 21,8 | 42,0 | 88,0 | 171,0 |
| BP-24 | FSMIM-SA | 5,0 | 13,9 | 0,5 | 0,5 | 0,5 | 1,8 | 66,0 |
| | FSMIM-SM | 5,0 | 13,9 | 0,5 | 0,5 | 0,5 | 1,8 | 66,0 |
| | FSMIM-TM | 6,8 | 18,4 | 0,5 | 0,5 | 0,5 | 2,2 | 75,0 |
| | FSMIM-TA | 6,8 | 18,4 | 0,5 | 0,5 | 0,5 | 2,2 | 75,0 |
| | FSM-ROM | 20,8 | 36,1 | 1,0 | 1,1 | 3,8 | 8,9 | 110,5 |
| MCNC | FSMIM-SA | 0,7 | 0,2 | 0,5 | 0,5 | 0,5 | 1,0 | 1,0 |
| | FSMIM-SM | 0,7 | 0,3 | 0,5 | 0,5 | 0,5 | 1,0 | 1,0 |
| | FSMIM-TA | 0,8 | 0,3 | 0,5 | 0,5 | 1,0 | 1,0 | 1,5 |
| | FSMIM-TM | 0,8 | 0,3 | 0,5 | 0,5 | 1,0 | 1,0 | 1,5 |
| | FSM-ROM | 10,5 | 13,1 | 1,0 | 1,5 | 3,5 | 15,0 | 48,0 |
| Total | FSMIM-SA | 4,5 | 9,5 | 0,5 | 0,5 | 1,0 | 3,5 | 66,0 |
| | FSMIM-SM | 5,4 | 10,0 | 0,5 | 0,5 | 1,5 | 6,4 | 66,0 |
| | FSMIM-TA | 8,3 | 17,5 | 0,5 | 1,0 | 1,5 | 6,4 | 112,0 |
| | FSMIM-TM | 9,4 | 18,3 | 0,5 | 1,0 | 2,5 | 9,5 | 123,0 |
| | FSM-ROM | 40,1 | 46,2 | 1,0 | 5,5 | 21,8 | 55,9 | 171,0 |

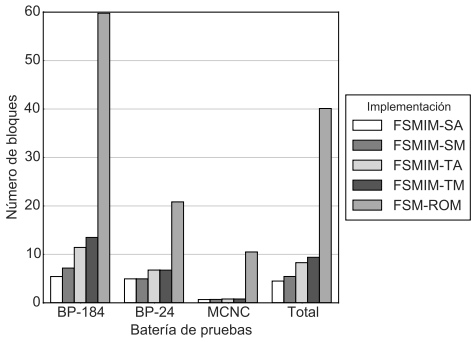


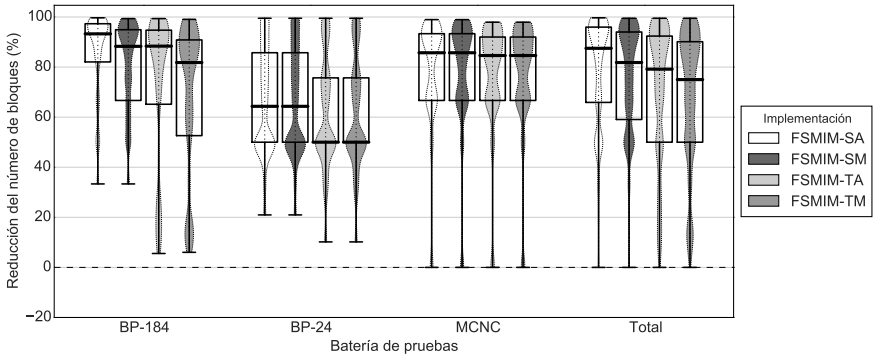

**Figura 5.57:** Número medio de bloques usados por las implementaciones FSM-ROM y FSMIM optimizadas en área.

las implementaciones ISE-BLOQ utiliza los bloques de memoria de forma diferente a las implementaciones convencionales de FSM en ROM. No se ha realizado un estudio teórico de dicha arquitectura en esta tesis debido a dos motivos: por una parte, no presenta ningún beneficio práctico frente a la arquitectura convencional basada en memoria y, por otra, Xilinx no proporciona detalles de la arquitectura.

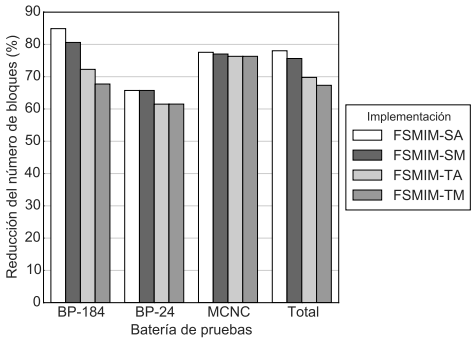
En promedio para el conjunto completo de FSM, las implementaciones FSMIM ordenadas de menor a mayor capacidad de reducción del número de bloques son FSMIM-TM, FSMIM-TA, FSMIM-SM y FSMIM-SA. Sin embargo, las elevadas reducciones en el número de bloques que consiguen respecto a FSM-ROM impiden que se aprecien en los resultados las posibles mejoras obtenidas al usar una de estas implementaciones frente al resto. Para cuantificar dichas mejoras, se ha calculado la reducción RPC del número de bloques que consigue cada implementación respecto a FSMIM-TM. La figura 5.59 muestra los valores medios. Para el conjunto completo de FSM se pueden obtener mejoras entre el 13 % al 38 % según la implementación elegida (debe tenerse en cuenta que la reducción de partida alcanzada por FSMIM-TM es del 67 % respecto a FSM-ROM). Como puede observarse,

**Tabla 5.54:** Resumen estadístico de la reducción RPC del número de bloques usados por las implementaciones FSMIM optimizadas en área respecto a FSM-ROM ( %).

| Bat. de pruebas | Implement. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-SA | 84,9 | 18,1 | 33,3 | 82,1 | 93,3 | 97,3 | 99,7 |
| | FSMIM-SM | 80,6 | 18,2 | 33,3 | 66,7 | 88,3 | 94,9 | 99,4 |
| | FSMIM-TA | 72,3 | 31,0 | 5,6 | 65,2 | 88,4 | 94,7 | 99,4 |
| | FSMIM-TM | 67,7 | 30,4 | 6,0 | 52,6 | 81,8 | 90,8 | 99,1 |
| BP-24 | FSMIM-SA | 65,7 | 21,8 | 21,0 | 50,0 | 64,3 | 85,7 | 99,5 |
| | FSMIM-SM | 65,7 | 21,8 | 21,0 | 50,0 | 64,3 | 85,7 | 99,5 |
| | FSMIM-TA | 61,5 | 22,9 | 10,2 | 50,0 | 50,0 | 75,7 | 99,5 |
| | FSMIM-TM | 61,5 | 22,9 | 10,2 | 50,0 | 50,0 | 75,7 | 99,5 |
| MCNC | FSMIM-SA | 77,6 | 24,6 | 0,0 | 66,7 | 85,7 | 93,3 | 99,0 |
| | FSMIM-SM | 77,0 | 24,4 | 0,0 | 66,7 | 85,7 | 93,3 | 99,0 |
| | FSMIM-TA | 76,3 | 24,0 | 0,0 | 66,7 | 84,6 | 92,0 | 97,9 |
| | FSMIM-TM | 76,3 | 24,0 | 0,0 | 66,7 | 84,6 | 92,0 | 97,9 |
| Total | FSMIM-SA | 78,0 | 21,9 | 0,0 | 65,9 | 87,5 | 96,0 | 99,7 |
| | FSMIM-SM | 75,6 | 21,3 | 0,0 | 59,1 | 81,8 | 94,0 | 99,5 |
| | FSMIM-TA | 69,8 | 28,1 | 0,0 | 50,0 | 79,2 | 92,4 | 99,5 |
| | FSMIM-TM | 67,3 | 27,6 | 0,0 | 50,0 | 75,0 | 90,1 | 99,5 |




**(a)**




**(b)**

**Figura 5.58:** Reducción RPC del número de bloques usados por las implementaciones FSMIM optimizadas en área respecto a FSM-ROM: (a) Distribución de los valores obtenidos y (b) valor medio.

las mejoras son mayores para la batería de pruebas BP-184, alcanzando reducciones entre el 25 % y el 60 % aproximadamente.

### Reducción del tamaño genérico de la ROM respecto a la implementación FSM-ROM

La selección inicial realizada para el estudio comparativo ha favorecido la obtención de reducciones elevadas por parte de las implementaciones FSMIM. Por una parte, esto es debido a que se han seleccionado las FSM para las que las estrategias han conseguido generar una FSMIM con ajuste de profundidad, lo que significa que, indirectamente, se han desechado los casos en los que no se ha conseguido reducir el número de bloques estimado de la

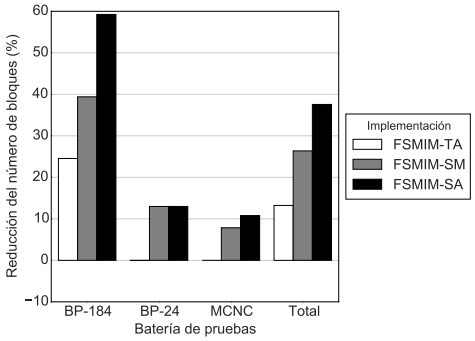



**Figura 5.59:** Reducción RPC media del número de bloques usados por las implementaciones FSMIM optimizadas en área respecto a FSMIM-TM.

ROM respecto a FSM-ROM. Tan sólo en uno de los casos, la reducción en el número de bloques estimado de la ROM conseguida por las FSMIM no se ha visto reflejada en los resultados de implementación. Por otra parte, se han excluido las FSM en las que la implementación FSM-ROM ocupa un sólo bloque de 9 Kbits.

Con objeto de obtener datos menos sesgados sobre la capacidad de mejora de las implementaciones FSMIM, se ha decidido realizar un breve estudio de la reducción del tamaño genérico de la ROM utilizando el conjunto completo de FSM sin excluir ningún caso. Se ha calculado la reducción RPC del tamaño de la ROM obtenida por las FSMIM respecto a FSM-ROM utilizando los resultados obtenidos en las pruebas independientes del dispositivo. Para facilitar la comparación con los resultados de las pruebas de implementación, las estrategias de optimización de FSMIM se han nombrado utilizado los mismos nombres de las implementaciones correspondientes; sin embargo, debe tenerse en cuenta que son resultados genéricos obtenidos con las FSMIM sin ajuste profundidad. La figura 5.60 muestra un resumen estadístico de los valores obtenidos.

Las tres baterías de pruebas utilizadas en esta tesis contienen un total de 305 FSM. De estas, el 79 % (242 FSM) cumplían los requisitos mínimos para poder generar una FSMIM y fueron seleccionadas para realizar los experimentos (son las que forman el conjunto completo de FSM).

El porcentaje de éxito es muy alto para el conjunto completo de FSM, alcanzando valores por encima del 85 % y llegando al 100 % en las estrategias FSMIM-SA. El peor resultado se obtiene en la batería de pruebas BP-24, en la que la arquitectura FSMIM-T obtiene porcentajes de éxito próximos al 69 %. Sin embargo, las arquitecturas FSMIM-S reducen la ROM en más

del 93 % de los casos. Por otra parte, el porcentaje de éxito de todas las estrategias es superior al 96 % para el resto de baterías de pruebas.

Respecto a las reducciones obtenidas, comparadas con los resultados de implementación, aunque se produce una bajada significativa en la batería BP-24 (con reducciones medias entre el 35 % de FSMIM-TA y el 45 % de FSMIM-SA), en el resto de baterías de pruebas mejora, obteniéndose reducciones medias entre el 80 % y el 94 % con todas las estrategias.

#### 5.6.1.2. Estudio del consumo de LUT

Para el estudio detallado del consumo de LUT se considerarán todas las implementaciones excepto ISE-BLOQ (debido al elevado número de casos que

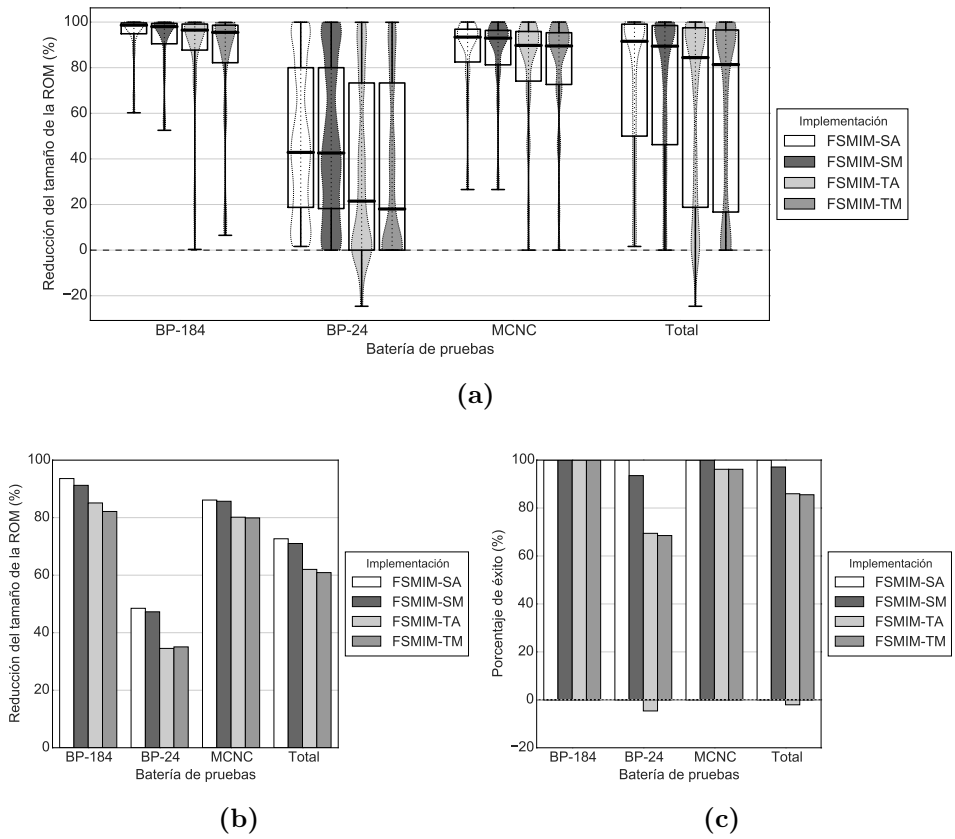



**Figura 5.60:** Reducción RPC del tamaño de la ROM obtenida por las arquitecturas FSMIM respecto a FSM-ROM en pruebas independientes del dispositivo: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

deben excluirse si se incluye esta implementación), siendo la implementación ISE-LUT la referencia principal para las comparativas realizadas. El conjunto completo de FSM usado en este estudio está compuesto por las 100 FSM en las que dichas implementaciones han tenido éxito. La tabla 5.55 muestra un resumen estadístico del número de LUT usadas para cada batería de pruebas.

El consumo de LUT de la implementación ISE-LUT varía entre las 9 y las 39 412 LUT (el 84 % del número total disponible en el dispositivo). En promedio, la batería de pruebas MCNC es la que contiene las FSM más pequeñas, que ocupan 55 LUT de media; las FSM de la batería de pruebas BP-24 ocupan 6 veces más; y las de BP-184, 8 veces más que las de esta última.

Cabe destacar que, en promedio, el consumo de LUT de las implementaciones FSM-ROM es mayor que el de las implementaciones basadas en la arquitectura FSMIM-T. En las implementaciones FSM-ROM, este consumo se debe al banco de multiplexores de la ROM, que se requiere sólo cuando la profundidad de la ROM es mayor de 16 Kpalabras (la profundidad máxima de los bloques de 18 Kbits). Esto ocurre en el 51 % de los casos del conjunto completo de FSM (que son el 83 % de las FSM de la batería de pruebas BP-184, el 20 % de las de BP-24 y sólo una FSM de MCNC). El 49 % restante no requiere ningún banco de multiplexores de la ROM y, por lo tanto, no consume LUT. Sin embargo, el número medio de LUT de la implementación FSM-ROM casi triplica al de la implementación FSMIM-TM. Esto significa que la reducción de la profundidad de la ROM, que se consigue gracias al banco de multiplexores de entradas de la FSMIM-T, tiene un impacto importante tanto en la reducción de bloques de memoria como en la reducción de LUT respecto a las implementaciones FSM-ROM.

Respecto a la implementación ISE-BLOQ, a pesar de que era la que presentaba el mayor consumo de bloques (incluso mayor que el de la implementación FSM-ROM), también es la implementación de FSM basada en memoria que requiere el mayor número medio de LUT. Para poder compararla con el resto de implementaciones, se han realizado los cálculos excluyendo las 43 FSM para las que esta implementación no es posible. Mientras que el resto de implementaciones de FSM basadas en memoria consumen menos de 24 LUT, la implementación ISE-BLOQ usa 373 LUT. Cabe destacar que este valor es próximo al consumo medio de LUT de la implementación ISE-LUT (436 LUT), lo que indica que la implementación ISE-BLOQ hace un uso poco eficiente de los bloques de memoria (incluso desde el punto de vista del ahorro de LUT).

Para evaluar el ahorro de LUT obtenido por las implementaciones de

**Tabla 5.55:** Resumen estadístico del número de LUT usadas por las implementaciones optimizadas en área. Para cada batería de pruebas, las implementaciones están ordenadas de menor a mayor número medio de LUT.

| Bat. de pruebas | Implement. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-TM | 6,4 | 3,1 | 2,0 | 5,0 | 7,0 | 8,0 | 22,0 |
| | FSMIM-TA | 8,4 | 4,7 | 2,0 | 7,0 | 8,0 | 10,0 | 37,0 |
| | FSM-ROM | 23,0 | 30,9 | 0,0 | 7,0 | 13,0 | 27,0 | 181,0 |
| | FSMIM-SM | 142,0 | 144,5 | 16,0 | 38,0 | 75,0 | 186,0 | 586,0 |
| | FSMIM-SA | 211,8 | 263,1 | 24,0 | 51,0 | 102,0 | 241,0 | 1204,0 |
| | ISE-LUT | 3472,5 | 6238,2 | 103,0 | 298,0 | 1013,0 | 3511,0 | 39412,0 |
| BP-24 | FSMIM-TA | 4,6 | 5,9 | 1,0 | 3,0 | 3,0 | 4,0 | 30,0 |
| | FSMIM-TM | 4,6 | 5,7 | 1,0 | 3,0 | 3,0 | 4,0 | 29,0 |
| | FSM-ROM | 7,6 | 16,1 | 0,0 | 0,0 | 0,0 | 0,0 | 48,0 |
| | FSMIM-SM | 18,2 | 15,2 | 3,0 | 6,0 | 15,0 | 22,0 | 56,0 |
| | FSMIM-SA | 18,2 | 15,1 | 3,0 | 5,2 | 17,0 | 22,8 | 60,0 |
| | ISE-LUT | 388,8 | 799,1 | 9,0 | 27,8 | 94,0 | 290,8 | 4100,0 |
| MCNC | FSM-ROM | 0,8 | 3,4 | 0,0 | 0,0 | 0,0 | 0,0 | 14,0 |
| | FSMIM-TM | 2,9 | 1,9 | 1,0 | 1,0 | 2,0 | 5,0 | 7,0 |
| | FSMIM-TA | 3,1 | 2,2 | 1,0 | 1,0 | 2,0 | 5,0 | 8,0 |
| | FSMIM-SM | 9,9 | 6,6 | 3,0 | 5,0 | 9,0 | 11,0 | 22,0 |
| | FSMIM-SA | 11,4 | 7,8 | 3,0 | 5,0 | 9,0 | 16,0 | 26,0 |
| | ISE-LUT | 55,3 | 35,8 | 15,0 | 26,0 | 50,0 | 77,0 | 126,0 |
| Total | FSMIM-TM | 5,3 | 4,1 | 1,0 | 3,0 | 5,0 | 7,0 | 29,0 |
| | FSMIM-TA | 6,3 | 5,2 | 1,0 | 3,0 | 5,5 | 8,0 | 37,0 |
| | FSM-ROM | 14,6 | 25,8 | 0,0 | 0,0 | 6,0 | 18,5 | 181,0 |
| | FSMIM-SM | 82,4 | 122,9 | 3,0 | 12,0 | 33,5 | 81,5 | 586,0 |
| | FSMIM-SA | 119,7 | 214,7 | 3,0 | 14,8 | 38,5 | 128,5 | 1204,0 |
| | ISE-LUT | 1966,4 | 4819,0 | 9,0 | 76,0 | 249,5 | 1267,2 | 39412,0 |

FSM basadas en memoria, se ha calculado la reducción RPC del número de LUT respecto a ISE-LUT. La tabla 5.56 muestra un resumen estadístico de los valores obtenidos para las implementaciones incluidas en el estudio detallado y la figura 5.61, una representación de dicho resumen.

El porcentaje de éxito es del 100 % para todas las implementaciones y baterías de pruebas. Las reducciones medias obtenidas para el conjunto completo de FSM son muy altas, oscilando entre el 82,3 % de la implementación FSMIM-SA y el 96,2 % de la implementación FSMIM-TM. Además, en el 75 % de los casos se obtienen reducciones superiores al 78 % con las implementaciones basadas en la arquitectura FSMIM-S y superiores al 95 % con el resto de implementaciones analizadas. Cabe destacar que, aunque FSM-ROM es la implementación que presenta la mayor reducción media en las baterías de pruebas MCNC y BP-24, las reducciones medias obtenidas por las implementaciones basadas en la arquitectura FSMIM-T son mayores que las de FSM-ROM en la batería de pruebas BP-184 y en el conjunto completo de FSM.

En promedio, la mejor opción respecto al consumo de LUT para FSM pequeñas es la arquitectura FSM-ROM. Sin embargo, el consumo de LUT de las implementaciones FSM-ROM aumenta más rápidamente con el tamaño de las FSM que el de las implementaciones FSMIM. Esta relación se muestra en la figura 5.62, que representa la reducción en el consumo de LUT que obtienen las implementaciones FSMIM respecto a FSM-ROM frente al tamaño de las FSM (medido como la profundidad de la ROM de la arquitectura FSM-ROM). Como puede observarse, la reducción es del $-100\,\%$ cuando la profundidad de la ROM de FSM-ROM es menor o igual a 16 Kpalabras (es decir, es la implementación FSM-ROM la que consigue una reducción del $100\,\%$ respecto a las implementaciones FSMIM). Esto se debe, como se ha explicado anteriormente, a que en estos casos la ROM se implementa usando exclusivamente bloques de memoria.

Sin embargo, las líneas de regresión local de las nubes de puntos muestran como la reducción de las implementaciones FSMIM crece con la profundidad de la ROM para valores superiores a 16 Kpalabras. De hecho, las implementaciones FSMIM-T consumen menos LUT que FSM-ROM en prácticamente todos los casos con profundidad mayor que 16 Kpalabras, alcanzando reducciones superiores al $50\,\%$ en la mayoría de ellos. En estos casos, que suponen el $46\,\%$ de las FSM del conjunto completo, la arquitectura FSMIM-T es mejor opción que FSM-ROM tanto en consumo de bloques como en consumo de LUT. Respecto a las implementaciones FSMIM-S, los resultados muestran que pueden llegar a consumir menos LUT que FSM-ROM a partir de las 125 Kpalabras de profundidad (de hecho, obtienen reducciones superiores al $25\,\%$ en los tres casos de mayor profundidad).

Si se incluyen en el cálculo las 57 FSM para las que la implementación FSM-ROM no es posible, las implementaciones basadas en FSMIM obtienen reducciones medias similares (con diferencias medias que no superan el $1\,\%$).

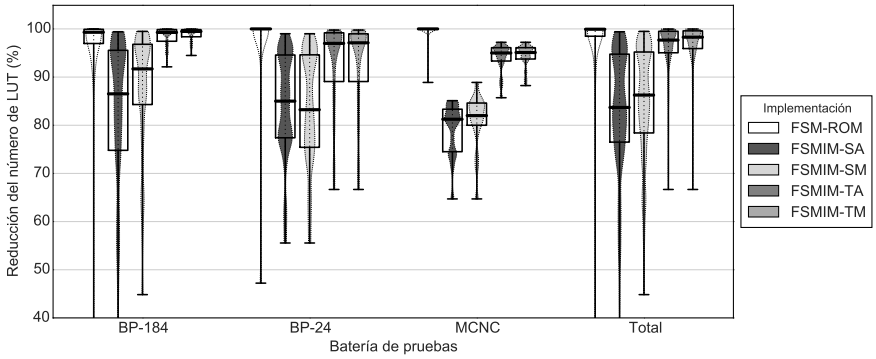
Por otra parte, la implementación ISE-BLOQ obtiene una reducción media del $-6,3\,\%$ para las FSM en las que esta implementación es posible (es decir, para el conjunto completo de FSM menos 43 casos). En el $53\,\%$ de los casos, el número de LUT usadas es mayor que el de la propia implementación ISE-LUT. Esto significa que el consumo de recursos de ISE-BLOQ es completamente ineficiente en estos casos, ya que a este incremento en el consumo de LUT respecto a la implementación de FSM convencional hay que sumar el uso de bloques de memoria. La única batería de pruebas en la que consigue una reducción media mayor que cero es BP-184, lo que parece indicar que esta arquitectura está diseñada para FSM grandes. Sin embargo, la reducción media en este caso (del $17\,\%$) es despreciable comparada

con las del resto de implementaciones basadas en memoria (que presentan reducciones medias mayores del 85 %).

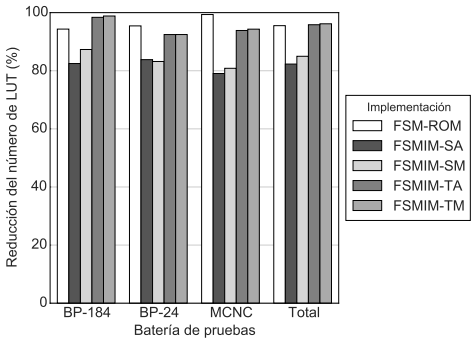
En promedio para el conjunto completo de FSM, las implementaciones basadas en memoria ordenadas de mayor a menor consumo de LUT son FSMIM-SA, FSMIM-SM, FSMIM-TA, FSMIM-TM y FSM-ROM. Con el fin de cuantificar las mejoras obtenidas al usar una de estas implementaciones frente al resto, se ha calculado la reducción RPC del número de LUT que consigue cada implementación respecto a FSMIM-SA (que es la que presenta el mayor consumo). La figura 5.63 muestra los valores medios obtenidos. Para el conjunto completo de FSM, se pueden obtener mejoras próximas al 80 % con la implementación FSM-ROM y las dos implementaciones basadas en la arquitectura FSMIM-T. Sin embargo, la mejora obtenida con la implementación FSMIM-SM (generada con la estrategia M-PLE) es sólo del 15 %, lo que indica que la influencia de la estrategia utilizada sobre las implementaciones FSMIM es poco significativa comparada con el impacto de la arquitectura. Las reducciones medias obtenidas por la implementación FSM-ROM en las baterías de pruebas BP-24 y MCNC superan en 15 y 23

**Tabla 5.56:** Resumen estadístico de la reducción RPC del número de LUT usadas por las implementaciones optimizadas en área respecto a ISE-LUT ( %).

| Bat. de pruebas | Implement. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSM-ROM | 94,3 | 15,0 | 29,6 | 97,0 | 99,3 | 99,9 | 100,0 |
| | FSMIM-SA | 82,5 | 18,3 | 2,0 | 74,8 | 86,5 | 95,6 | 99,4 |
| | FSMIM-SM | 87,3 | 12,8 | 44,8 | 84,3 | 91,7 | 96,8 | 99,5 |
| | FSMIM-TA | 98,4 | 2,0 | 92,1 | 97,4 | 99,3 | 99,9 | $\approx 100{,}0$ |
| | FSMIM-TM | 98,8 | 1,4 | 94,5 | 98,4 | 99,5 | 99,8 | $\approx 100{,}0$ |
| BP-24 | FSM-ROM | 95,4 | 13,7 | 47,2 | 100,0 | 100,0 | 100,0 | 100,0 |
| | FSMIM-SA | 83,8 | 11,4 | 55,6 | 77,4 | 85,0 | 94,6 | 99,0 |
| | FSMIM-SM | 83,2 | 12,2 | 55,6 | 75,4 | 83,2 | 94,6 | 99,0 |
| | FSMIM-TA | 92,5 | 9,6 | 66,7 | 89,1 | 97,0 | 99,2 | 99,8 |
| | FSMIM-TM | 92,5 | 9,6 | 66,7 | 89,1 | 97,1 | 99,0 | 99,8 |
| MCNC | FSM-ROM | 99,3 | 2,7 | 88,9 | 100,0 | 100,0 | 100,0 | 100,0 |
| | FSMIM-SA | 79,1 | 5,7 | 64,7 | 74,5 | 81,2 | 83,3 | 85,1 |
| | FSMIM-SM | 80,8 | 6,0 | 64,7 | 80,0 | 82,0 | 84,6 | 88,9 |
| | FSMIM-TA | 93,9 | 3,2 | 85,7 | 93,3 | 95,0 | 96,1 | 97,2 |
| | FSMIM-TM | 94,3 | 2,5 | 88,2 | 93,8 | 95,1 | 96,1 | 97,2 |
| Total | FSM-ROM | 95,5 | 13,3 | 29,6 | 98,5 | 99,9 | 100,0 | 100,0 |
| | FSMIM-SA | 82,3 | 14,9 | 2,0 | 76,5 | 83,7 | 94,8 | 99,4 |
| | FSMIM-SM | 85,0 | 11,9 | 44,8 | 78,4 | 86,3 | 95,2 | 99,5 |
| | FSMIM-TA | 95,8 | 6,2 | 66,7 | 95,0 | 97,7 | 99,5 | $\approx 100{,}0$ |
| | FSMIM-TM | 96,2 | 6,1 | 66,7 | 95,9 | 98,3 | 99,6 | $\approx 100{,}0$ |




**(a)**




**(b)**

**Figura 5.61:** Reducción RPC del número de LUT usadas por las implementaciones optimizadas en área respecto a ISE-LUT: (a) Distribución de los valores obtenidos y (b) valor medio. Para facilitar la visualización de los datos, en (a) se han representado sólo las reducciones en el intervalo (40 %, 100 %). De izquierda a derecha, los valores mínimos excluidos para BP-184 son 29 % y 2 % (los valores para el conjunto completo de FSM son los mismos).

puntos a las obtenidas por la implementación FSMIM-TM, respectivamente; sin embargo, en la batería de pruebas BP-184, la reducción media conseguida por FSMIM-TM supera en 16 puntos a la de FSM-ROM.

### Estudio del NLAB

Con objeto de estudiar la eficiencia en el uso de bloques de las implementaciones basadas en memoria, se ha calculado el NLAB obtenido por cada una de ellas. Debido a que, en promedio, la implementación ISE-BLOQ consume más LUT que la implementación ISE-LUT, esta ha sido excluida del cálcu-

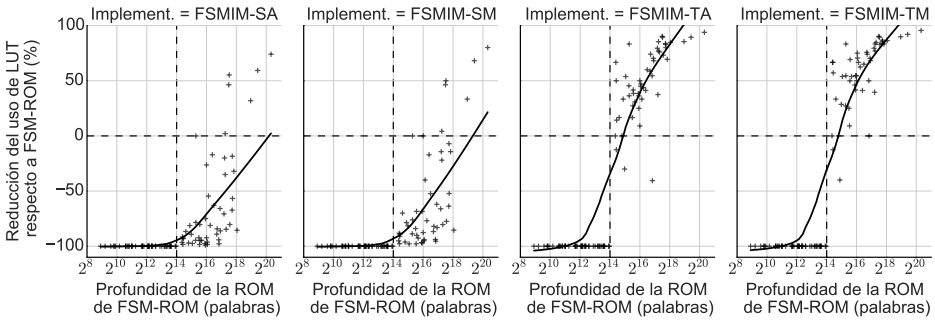



**Figura 5.62:** Reducción RPC del número de LUT usadas por las implementaciones FSMIM optimizadas en área respecto a FSM-ROM frente a la profundidad de la ROM de FSM-ROM. Se ha marcado el valor de la profundidad máxima de los bloques de memoria de 18 Kbits ($2^{14}$ palabras).

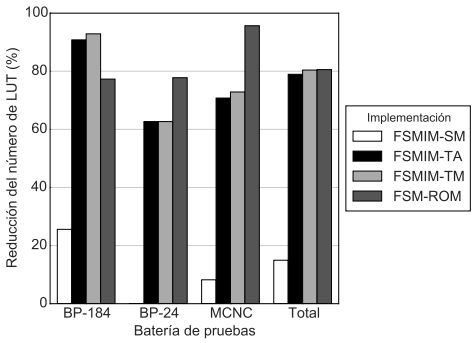



**Figura 5.63:** Reducción RPC media del número de LUT usadas por las implementaciones de FSM basadas en memoria respecto a FSMIM-SA. Los resultados han sido obtenidos con las implementaciones optimizadas en área.

lo del NLAB. La tabla 5.57 muestra un resumen estadístico de los valores obtenidos y la figura 5.65, una representación de dicho resumen.

Como puede observarse, en promedio, las implementaciones FSMIM son significativamente más eficientes que la implementación FSM-ROM, que presenta un NLAB medio inferior al 40 % del obtenido por la menos eficiente de las implementaciones FSMIM. Por otra parte, las implementaciones basadas en la arquitectura FSMIM-S son las más eficientes en el uso de bloques a pesar de que consumen más LUT que el resto de implementaciones.

La comparación de los datos obtenidos en cada batería de pruebas muestra que el valor del NLAB aumenta significativamente con el tamaño de las FSM, al igual que las diferencias entre los valores conseguidos por las imple-

**Tabla 5.57:** Resumen estadístico del NLAB de las implementaciones optimizadas en área incluidas en el estudio detallado. Las implementaciones están ordenadas de menor a mayor NLAB.

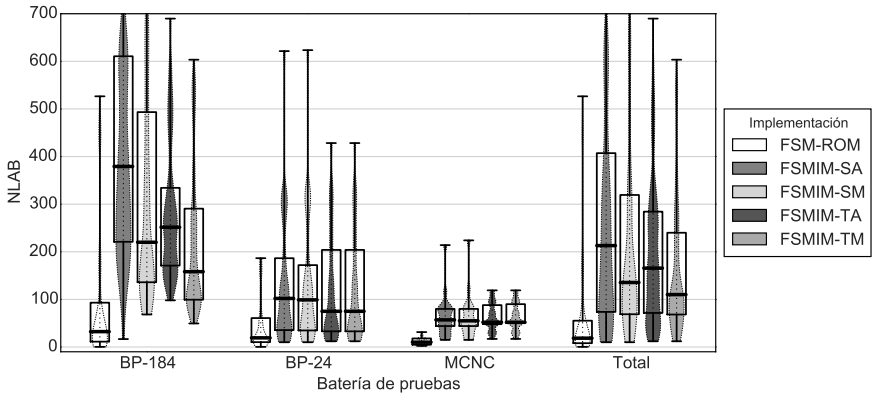
| Implementación | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| FSM-ROM | 67,4 | 116,1 | 0,4 | 8,0 | 18,6 | 55,3 | 526,5 |
| FSMIM-TM | 168,6 | 146,1 | 12,0 | 68,2 | 110,1 | 240,0 | 603,5 |
| FSMIM-TA | 201,1 | 155,4 | 12,0 | 71,5 | 165,7 | 284,3 | 689,8 |
| FSMIM-SM | 241,7 | 266,4 | 10,0 | 68,8 | 135,5 | 319,4 | 1308,5 |
| FSMIM-SA | 304,6 | 312,9 | 10,0 | 73,8 | 213,0 | 407,2 | 1397,0 |

mentaciones. De hecho, si se incluyen en el cálculo del NLAB las 57 FSM en las que la implementación FSM-ROM no es posible, se obtienen incrementos en el NLAB medio próximos al 18 % para las implementaciones FSMIM-T y al 44 % para las implementaciones FSMIM-S. La tabla 5.58 muestra un resumen estadístico de los valores obtenidos. La implementación FSMIM-TM presenta el NLAB medio más pequeño, con un valor de 198 LUT/bloque (con este NLAB, los 172 bloques del dispositivo FPGA usado en los experimentos permitirían ahorrar el 73 % del total de LUT). Por su parte, la implementación FSMIM-SA es la que consigue el mayor valor medio, igual a 437 LUT/bloque (con este NLAB, el 62 % de los bloques podrían ahorrar una cantidad equivalente al 100 % de las LUT del dispositivo FPGA).

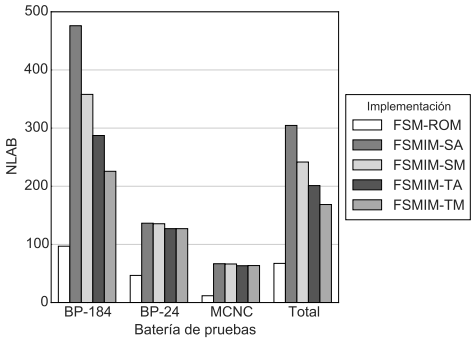
A pesar de que la implementación FSM-ROM es la que consume el menor número de LUT en más de la mitad de los casos, la comparación del NLAB pone de manifiesto que las implementaciones FSMIM son mucho más eficientes en el uso de bloques. La tabla 5.59 muestra un resumen estadístico del incremento RPC del NLAB conseguido por las implementaciones FSMIM respecto a FSM-ROM y la figura 5.66, los valores medios y el porcentaje de éxito. En el mejor de los casos, el NLAB obtenido por las implementaciones FSMIM puede llegar a ser varios ordenes de magnitud mayores que el de la implementación FSM-ROM. En promedio, para el conjunto completo

**Tabla 5.58:** Resumen estadístico del NLAB de las implementaciones optimizadas en área para la muestra que incluye las 57 FSM en las que la implementación FSM-ROM no es posible. Las implementaciones están ordenadas de menor a mayor NLAB.

| Implementación | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| FSMIM-TM | 198,2 | 159,2 | 12,0 | 77,5 | 133,0 | 291,2 | 603,5 |
| FSMIM-TA | 233,3 | 162,0 | 12,0 | 98,7 | 195,1 | 339,8 | 689,8 |
| FSMIM-SM | 348,1 | 367,7 | 4,6 | 92,0 | 168,2 | 509,2 | 1599,0 |
| FSMIM-SA | 436,7 | 418,2 | 2,6 | 114,0 | 301,0 | 635,9 | 1598,6 |




**(a)**



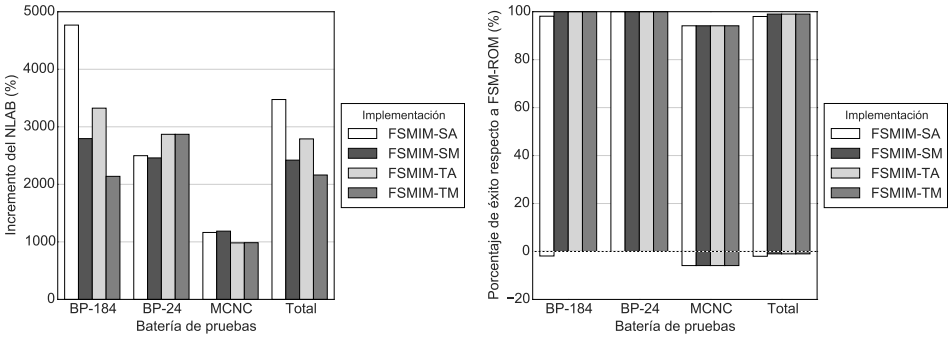

**(b)**

**Figura 5.65:** Resumen estadístico del NLAB de las implementaciones optimizadas en área: (a) Distribución de los valores obtenidos y (b) valor medio. Para facilitar la visualización de los datos, en (a) se han representado sólo los valores en el intervalo $(0, 700)$. De izquierda a derecha, los valores máximos excluidos para BP-184 son 1 308,5 y 1 397 (los valores para el conjunto completo de FSM son los mismos).

de FSM, el NLAB de las implementaciones FSMIM es entre 22 y 35 veces mayor que el de la implementación FSM-ROM. Incluso en las baterías de pruebas MCNC y BP-24, que contienen las FSM más pequeñas, se consiguen incrementos elevados, que se encuentran entre el 982 % y el 2 869 %. La implementación FSM-ROM consigue un NLAB mayor que las implementaciones FSMIM tan sólo en un caso de la batería de pruebas MCNC y en otro de BP-184 (aunque en este último, sólo consigue superar a la implementación FSMIM-SA).

**Tabla 5.59:** Resumen estadístico del incremento RPC del NLAB obtenido por las implementaciones optimizadas en área respecto a FSM-ROM ( %).

| Bat. de pruebas | Implement. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSMIM-SA | 4767,9 | 12112,0 | −30,7 | 399,5 | 1110,0 | 3485,2 | 67598,1 |
| | FSMIM-SM | 2794,6 | 6949,4 | 45,3 | 251,3 | 687,7 | 1851,3 | 36569,8 |
| | FSMIM-TA | 3324,1 | 8021,6 | 5,9 | 225,0 | 766,7 | 1892,3 | 43339,6 |
| | FSMIM-TM | 2138,9 | 5436,2 | 10,0 | 131,8 | 491,5 | 1114,3 | 29423,9 |
| BP-24 | FSMIM-SA | 2498,1 | 7362,0 | 11,1 | 60,4 | 146,3 | 482,0 | 29716,2 |
| | FSMIM-SM | 2459,1 | 7241,6 | 11,1 | 60,0 | 139,9 | 482,9 | 29716,2 |
| | FSMIM-TA | 2869,5 | 9352,7 | 11,7 | 61,4 | 99,1 | 316,9 | 39300,0 |
| | FSMIM-TM | 2869,8 | 9352,6 | 11,7 | 62,4 | 99,0 | 316,5 | 39300,0 |
| MCNC | FSMIM-SA | 1163,6 | 2137,5 | −20,0 | 153,8 | 415,8 | 1118,8 | 9071,4 |
| | FSMIM-SM | 1186,4 | 2246,2 | −20,0 | 153,8 | 361,6 | 1193,8 | 9500,0 |
| | FSMIM-TA | 982,3 | 1290,7 | −5,9 | 188,9 | 517,6 | 1101,3 | 5000,0 |
| | FSMIM-TM | 985,8 | 1291,7 | −5,9 | 188,9 | 517,6 | 1101,3 | 5000,0 |
| Total | FSMIM-SA | 3474,2 | 9786,3 | −30,7 | 141,3 | 625,3 | 1833,0 | 67598,1 |
| | FSMIM-SM | 2420,6 | 6471,5 | −20,0 | 136,7 | 391,0 | 1401,3 | 36569,8 |
| | FSMIM-TA | 2789,6 | 7772,1 | −5,9 | 92,9 | 375,1 | 1256,8 | 43339,6 |
| | FSMIM-TM | 2162,1 | 6465,7 | −5,9 | 97,0 | 299,0 | 944,6 | 39300,0 |




**(a)** **(b)**

**Figura 5.66:** Incremento RPC del NLAB obtenido por las implementaciones optimizadas en área respecto a FSM-ROM: (a) valor medio y (b) porcentaje de éxito.

#### 5.6.1.3. Estudio de la frecuencia de operación máxima

Al igual que en el estudio del consumo de LUT, el estudio detallado de la frecuencia de operación máxima se realizará con todas las implementaciones excepto ISE-BLOQ, siendo ISE-LUT la referencia principal para la mayoría de análisis comparativo. El conjunto completo de FSM utilizado en el estudio está compuesto por las 100 FSM en las que dichas implementaciones han sido posible. La tabla 5.60 muestra un resumen estadístico de la frecuencia

de operación máxima obtenida en cada batería de pruebas y la figura 5.67, una representación de los valores obtenidos.

La gama de frecuencias alcanzadas por la implementación ISE-LUT abarca desde los 5 MHz (obtenido en la batería de pruebas BP-184) hasta los 300 MHz (obtenido en la batería de pruebas BP-24). Cabe destacar que, en promedio, la velocidad de las implementaciones basadas en memoria es mayor que la de la implementación ISE-LUT, presentando incrementos de la frecuencia media que varían entre el 31 % y el 75 %. Por otra parte, las implementaciones FSMIM-T son las que obtienen las mayores frecuencias medias, seguidas por la implementación FSM-ROM.

Como cabía esperar, las frecuencias medias más altas se producen en la batería de pruebas MCNC, que contiene las FSM más pequeñas, mientras que las frecuencias menores se obtienen en la batería de pruebas BP-184. El diagrama de frecuencias medias (figura 5.67b) muestra que la caída de frecuencia que se produce al aumentar el tamaño medio de las FSM es mayor en la implementación ISE-LUT que en las implementaciones FSMIM. De igual manera, también es mayor la caída de frecuencia que sufre la implementación FSM-ROM en la batería de pruebas BP-184, debido principalmente a que el banco de multiplexores de la ROM se encuentra en el camino crítico.

Los datos anteriores indican de las implementaciones FSMIM son las que sufren una menor degradación de la velocidad con el aumento del tamaño de las FSM. Esta característica puede observarse más claramente en los incrementos de velocidad alcanzados por las implementaciones basadas en memoria. La tabla 5.61 muestra un resumen estadístico del incremento RPC de la frecuencia de operación máxima respecto a ISE-LUT y la figura 5.68, una representación de dicho resumen.

Los incrementos obtenidos por todas las implementaciones en el conjunto completo de FSM son muy significativos. Las implementaciones FSMIM-T son las que consiguen los mejores resultados, presentando incrementos medios del 246,5 % y 252,4 %, un porcentaje de éxito del 94 % y un pequeño porcentaje de fracaso del 6 %. El resto de implementaciones basadas en memoria alcanzan incrementos entre el 138,7 % de FSMIM-SA y el 171,3 % de FSM-ROM, porcentajes de éxito entre el 72 % de FSMIM-SA y el 81 % de FSM-ROM, y porcentajes de fracaso que no superan el 28 %. Cabe destacar que los incrementos medios obtenidos por ISE-LUT respecto a las implementaciones basadas en memoria para los casos en los que estas fracasan no superan el 53 % (siendo inferiores al 11 % para las implementaciones FSMIM).

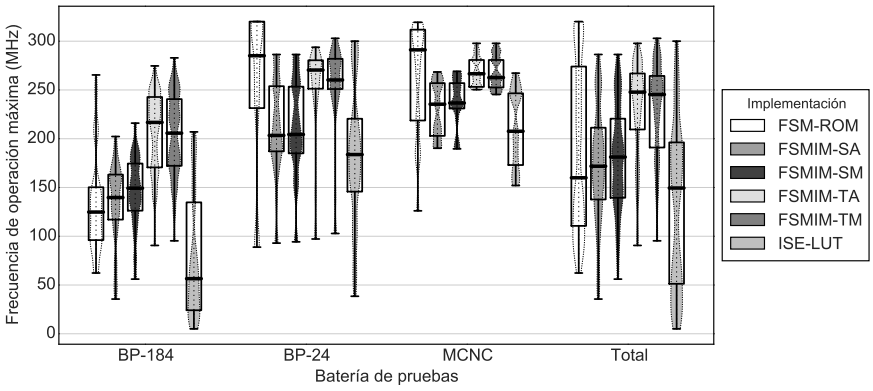
El análisis de los incrementos obtenidos en cada batería de pruebas muestra dos aspectos en la distribución de los resultados. En primer lugar, se

**Tabla 5.60:** Resumen estadístico de la frecuencia de operación máxima (MHz) obtenida por las implementaciones optimizadas en área. Para cada batería de pruebas, las implementaciones están ordenadas de menor a mayor frecuencia media.

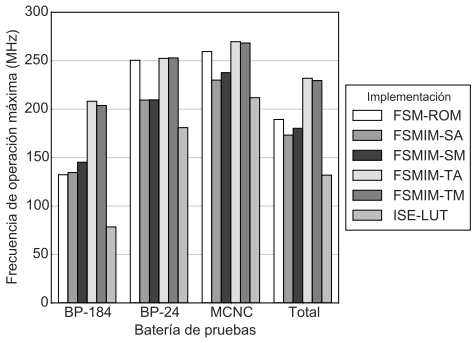
| Bat. de pruebas | Implement. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | ISE-LUT | 78,5 | 63,7 | 5,0 | 24,1 | 56,5 | 134,8 | 207,0 |
| | FSM-ROM | 132,3 | 50,2 | 62,3 | 96,0 | 124,8 | 150,2 | 265,4 |
| | FSMIM-SA | 134,5 | 41,1 | 35,6 | 117,1 | 139,7 | 163,2 | 202,3 |
| | FSMIM-SM | 145,2 | 37,3 | 56,0 | 126,2 | 149,3 | 174,6 | 216,0 |
| | FSMIM-TM | 203,8 | 42,9 | 95,3 | 172,2 | 205,7 | 240,6 | 282,9 |
| | FSMIM-TA | 208,1 | 41,0 | 90,6 | 170,6 | 216,7 | 242,8 | 274,6 |
| BP-24 | ISE-LUT | 180,9 | 71,2 | 38,5 | 145,8 | 183,8 | 220,5 | 300,0 |
| | FSMIM-SA | 209,4 | 50,6 | 93,0 | 187,0 | 203,4 | 253,9 | 286,4 |
| | FSMIM-SM | 209,6 | 51,2 | 94,3 | 185,0 | 204,4 | 253,4 | 286,4 |
| | FSM-ROM | 250,4 | 84,1 | 88,8 | 231,4 | 285,1 | 320,1 | 320,1 |
| | FSMIM-TA | 252,4 | 49,8 | 97,2 | 251,3 | 270,4 | 280,4 | 293,8 |
| | FSMIM-TM | 252,9 | 50,6 | 102,8 | 250,9 | 260,2 | 281,9 | 302,9 |
| MCNC | ISE-LUT | 211,8 | 41,1 | 152,0 | 173,0 | 207,6 | 246,5 | 267,2 |
| | FSMIM-SA | 230,0 | 28,7 | 190,3 | 202,9 | 235,4 | 257,0 | 268,5 |
| | FSMIM-SM | 237,7 | 25,6 | 189,6 | 231,0 | 236,6 | 257,0 | 269,1 |
| | FSM-ROM | 259,4 | 60,3 | 126,1 | 218,7 | 291,1 | 311,8 | 319,4 |
| | FSMIM-TM | 268,3 | 17,5 | 245,3 | 252,5 | 262,7 | 280,7 | 297,8 |
| | FSMIM-TA | 269,6 | 15,9 | 250,3 | 253,2 | 266,6 | 280,7 | 297,8 |
| Total | ISE-LUT | 131,9 | 85,1 | 5,0 | 51,2 | 149,4 | 196,2 | 300,0 |
| | FSMIM-SA | 173,2 | 59,3 | 35,6 | 137,7 | 171,8 | 211,3 | 286,4 |
| | FSMIM-SM | 180,2 | 55,6 | 56,0 | 139,5 | 181,2 | 220,6 | 286,4 |
| | FSM-ROM | 189,3 | 87,7 | 62,3 | 110,6 | 160,0 | 274,0 | 320,1 |
| | FSMIM-TM | 229,5 | 50,4 | 95,3 | 191,0 | 245,3 | 264,4 | 302,9 |
| | FSMIM-TA | 231,9 | 48,2 | 90,6 | 209,5 | 247,8 | 266,9 | 297,8 |

observa claramente la influencia del tamaño de las FSM en los incrementos medios. En la batería de pruebas MCNC, que es la que contiene la mayor proporción de FSM pequeñas (en número de LUT de la implementación ISE-LUT y en número de bloques de FSM-ROM), los incrementos medios no superan el 31 %; en la batería BP-24 oscilan entre el 29 % y el 57 %; y en la batería BP-184, que es la que contiene las FSM más grandes, son superiores al 242 %. Cabe destacar que, en esta última batería de pruebas, las implementaciones FSMIM-T consiguen un porcentaje de éxito del 100 %. Estos datos indican que las mejoras en velocidad obtenidas por las implementaciones basadas en memoria crecen con el tamaño de las FSM.

Por otra parte, las diferencias entre los incrementos medios obtenidos por las implementaciones FSMIM-T y el resto de implementaciones es claramente mayor en la batería de pruebas BP-184. Por lo tanto, el tamaño




**(a)**




**(b)**

**Figura 5.67:** Frecuencias de operación máximas (MHz) obtenidas por las implementaciones optimizadas en área: (a) Distribución de los valores y (b) valor medio.

de la FSM afecta de forma diferente a las distintas implementaciones de FSM basadas en memoria. Para cuantificar tanto la influencia del tamaño de estas implementaciones como las mejoras que se obtienen al usar una de ellas frente al resto, se ha calculado el incremento RPC de velocidad respecto a FSMIM-SA (que es la que presenta el menor incremento frecuencia medio respecto a ISE-LUT). La figura 5.69 muestra los incrementos medios obtenidos.

La mejora obtenida por la implementación FSMIM-SM (es decir, manteniendo la arquitectura FSMIM-S y cambiando de estrategia) es tan solo del 8 % para el conjunto completo de FSM. Sin embargo las mejoras obtenidas al utilizar la arquitectura FSMIM-T es del 46 % aproximadamente. Esto indica que, al igual que ocurría con el consumo de LUT, el impacto de las

estrategias de optimización de FSMIM en la velocidad es poco significativo comparado con el de la arquitectura FSMIM utilizada. Por otra parte, los incrementos medios obtenidos en la batería de pruebas BP-184 superan en más de dos veces a los obtenidos en el resto de baterías de pruebas.

Por el contrario, la mejora conseguida por la implementación FSM-ROM no parece aumentar con el tamaño de la FSM, ya que en la batería de pruebas BP-184 es despreciable a pesar de que en MCNC y BP-24 conseguía incrementos de más del 10 % respecto a FSMIM-SA. Con objeto de estudiar la relación entre el tamaño de la FSM y la velocidad de las implementaciones basadas en memoria, se ha representado el incremento RPC de velocidad respecto a FSM-ROM que obtienen las implementaciones FSMIM frente al tamaño de la FSM, medido como el número de bloques de la ROM de la implementación FSM-ROM (véase la figura 5.70).

Para FSM pequeñas, la implementación FSM-ROM es más rápida que las implementaciones FSMIM. Sin embargo, los primeros casos en los que las implementaciones FSMIM-T son más rápidas que FSM-ROM aparecen para las FSM con una ROM de entre 4 y 8 bloques de memoria. A partir de ese punto, las líneas de regresión local muestran que el incremento de velocidad aumenta con el tamaño de la FSM. Las implementaciones FSMIM-T son más rápidas en el 66 % de los casos, alcanzando incrementos de frecuencia superiores al 50 % en la mayoría de ellos. En estos casos, las FSMIM-T son mejor opción que FSM-ROM tanto en velocidad como en consumo de bloques; además, si a estos dos parámetros se une el consumo de LUT, las FSMIM-T son mejor opción en el 47 % de los casos. Por su parte, el número de casos en los que las implementaciones FSMIM-S son más rápidas que FSM-ROM varía entre el 41 % de FSMIM-SA y el 47 % de FSMIM-SM. Los primeros casos en los que estas implementaciones mejoran la velocidad de FSM-ROM se producen en las FSM con una ROM de entre 8 y 32 bloques de memoria.

Se ha estudiado también la relación entre el tamaño de la FSM y las mejoras en velocidad obtenidas por las implementaciones FSMIM respecto a ISE-LUT. Este análisis comparativo se ha realizado incluyendo las 57 FSM que fueron excluidas del estudio detallado debido a que la implementación FSM-ROM no era posible. Puesto que entre estas FSM se encuentran las de mayor tamaño, los incrementos medios de velocidad respecto a ISE-LUT obtenidos con la nueva muestra han aumentado entre el 40 % y el 60 % (véase la tabla 5.62, que contiene un resumen estadístico). La relación entre el incremento RPC de velocidad obtenido por las implementaciones FSMIM respecto a ISE-LUT y el número de estados de la FSM se representa en la figura 5.71. Las líneas de regresión local muestran que el incremento de

velocidad crece con el número de estados de la FSM, siendo el crecimiento más pronunciado en las implementaciones FSMIM-T.

Finalmente, respecto a la implementación ISE-BLOQ, el análisis comparativo se ha realizado excluyendo de los cálculos las 43 FSM en las que esta ha fallado. La implementación ISE-BLOQ es la que presenta los peores resultados de velocidad, debido posiblemente a su mayor consumo de LUT y de bloques de memoria. Ha obtenido una frecuencia media inferior a la del resto de implementaciones tanto en el conjunto completo de FSM como en las diferentes baterías de pruebas. De hecho, obtiene un incremento RPC negativo respecto a ISE-LUT del $-93,9\,\%$, consiguiendo mejoras de velocidad en tan sólo 4 casos. Por lo tanto, en las implementaciones optimizadas en área, esta arquitectura no presenta ninguna ventaja respecto a las demás, siendo la peor opción de diseño disponible.

**Tabla 5.61:** Resumen estadístico del incremento RPC de velocidad obtenido por las implementaciones optimizadas en área respecto a ISE-LUT ( %).

| Bat. de pruebas | Implement. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSM-ROM | 289,1 | 479,4 | −219,8 | 16,6 | 135,6 | 371,6 | 2173,4 |
| | FSMIM-SA | 242,2 | 339,3 | −30,9 | 22,3 | 119,1 | 307,1 | 1593,1 |
| | FSMIM-SM | 295,2 | 444,9 | −26,7 | 35,2 | 133,2 | 326,5 | 2286,8 |
| | FSMIM-TA | 434,2 | 490,6 | 17,2 | 77,0 | 295,0 | 507,6 | 2481,6 |
| | FSMIM-TM | 423,2 | 485,4 | 17,1 | 86,9 | 261,9 | 507,3 | 2382,0 |
| BP-24 | FSM-ROM | 47,5 | 64,5 | −121,5 | 10,2 | 52,4 | 83,1 | 156,1 |
| | FSMIM-SA | 28,5 | 41,2 | −16,5 | −2,7 | 18,4 | 47,5 | 141,7 |
| | FSMIM-SM | 28,6 | 42,0 | −19,7 | −2,7 | 20,1 | 47,5 | 146,7 |
| | FSMIM-TA | 56,3 | 48,5 | −15,2 | 25,4 | 51,0 | 80,6 | 152,6 |
| | FSMIM-TM | 56,7 | 48,9 | −15,2 | 27,3 | 51,0 | 79,5 | 167,3 |
| MCNC | FSM-ROM | 22,3 | 20,7 | −20,6 | 13,2 | 16,3 | 29,8 | 69,5 |
| | FSMIM-SA | 10,7 | 17,8 | −16,4 | −1,0 | 10,1 | 26,0 | 43,8 |
| | FSMIM-SM | 15,2 | 21,0 | −16,4 | −1,0 | 11,7 | 27,2 | 55,1 |
| | FSMIM-TA | 31,8 | 26,3 | −3,6 | 8,6 | 30,8 | 46,3 | 84,2 |
| | FSMIM-TM | 30,9 | 24,8 | −3,6 | 8,6 | 30,8 | 43,4 | 81,4 |
| Total | FSM-ROM | 171,3 | 371,4 | −219,8 | 12,3 | 53,8 | 148,1 | 2173,4 |
| | FSMIM-SA | 138,7 | 270,6 | −30,9 | −1,3 | 32,6 | 133,9 | 1593,1 |
| | FSMIM-SM | 167,6 | 350,9 | −26,7 | 3,8 | 36,7 | 141,7 | 2286,8 |
| | FSMIM-TA | 252,4 | 406,1 | −15,2 | 33,5 | 74,1 | 303,8 | 2481,6 |
| | FSMIM-TM | 246,5 | 400,2 | −15,2 | 30,4 | 77,1 | 270,6 | 2382,0 |

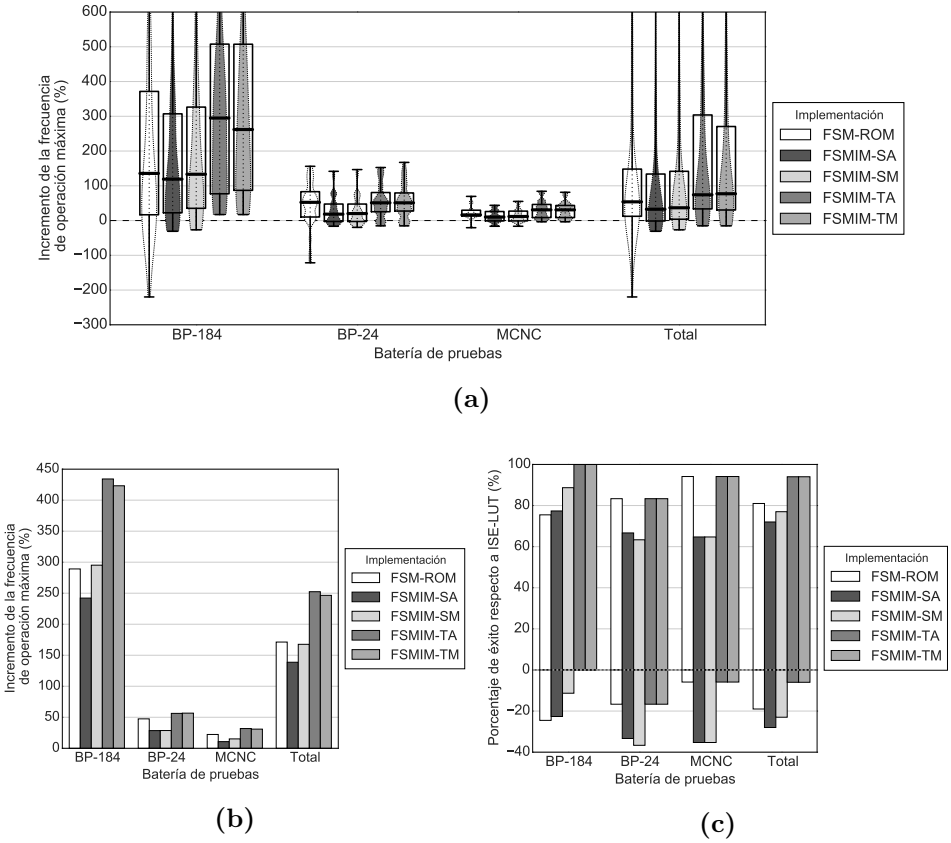



**Figura 5.68:** Incremento RPC de velocidad obtenido por las implementaciones optimizadas en área respecto a ISE-LUT: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito. Para facilitar la visualización de los datos, en (a) se han representado sólo los valores en el intervalo $(-300, 600)$. De izquierda a derecha, los valores máximos excluidos para BP-184 son aproximadamente 2 173 %, 1 593 %, 2 286 %, 2 481 % y 2 382 % (los valores para el conjunto completo de FSM son los mismos).

### 5.6.2. Análisis de las implementaciones optimizadas en velocidad

El consumo de bloques de memoria de las implementaciones optimizadas en velocidad ha sido idéntico al de las implementaciones optimizadas en área; por lo tanto, en esta sección sólo se estudiará la frecuencia de operación máxima y el consumo de LUT.

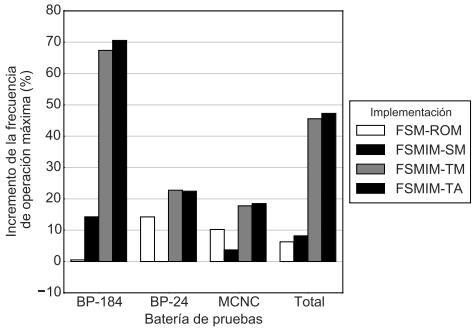



**Figura 5.69:** Incremento RPC medio de velocidad obtenido por las implementaciones de FSM basadas en memoria respecto a FSMIM-SA. Los resultados han sido obtenidos con las implementaciones optimizadas en área.

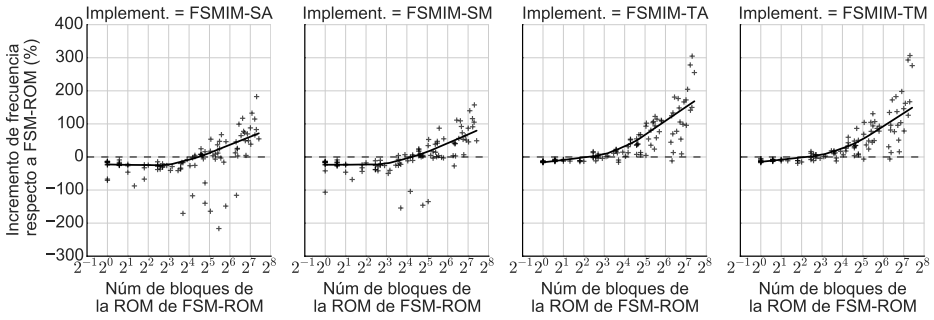



**Figura 5.70:** Incremento RPC de velocidad obtenido por las implementaciones FSMIM optimizadas en área respecto a FSM-ROM frente al tamaño de la ROM de FSM-ROM (número de bloques).

**Tabla 5.62:** Resumen estadístico del incremento RPC de la frecuencia de operación máxima ( %) obtenida por las implementaciones optimizadas en área para la muestra que incluye las 57 FSM en las que la implementación FSM-ROM no es posible (en total, 157 FSM). Las implementaciones están ordenadas de menor a mayor frecuencia media.

| Implementación | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| FSMIM-SA | 235,4 | 398,8 | −111,5 | 4,5 | 74,2 | 293,1 | 2284,9 |
| FSMIM-SM | 249,3 | 417,8 | −45,6 | 8,2 | 76,7 | 311,0 | 2311,5 |
| FSMIM-TM | 349,0 | 474,8 | −15,2 | 42,3 | 138,3 | 468,2 | 2504,6 |
| FSMIM-TA | 354,2 | 467,3 | −19,1 | 47,0 | 142,1 | 452,4 | 2481,6 |

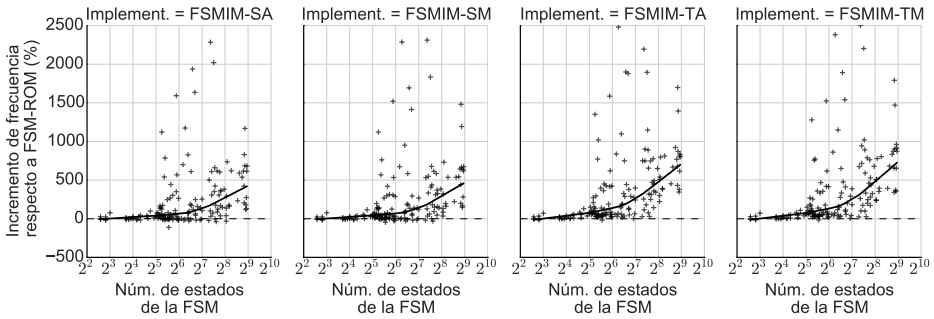



**Figura 5.71:** Incremento RPC de velocidad obtenido por las implementaciones FSMIM optimizadas en área respecto a ISE-LUT frente al número de estados de la FSM. Los resultados han sido calculados incluyendo las 57 FSM en las que la implementación FSM-ROM no es posible (en total se han usado 157 FSM).

#### 5.6.2.1. Estudio de la frecuencia de operación máxima

El estudio detallado de la frecuencia de operación máxima obtenida se realizará con todas las implementaciones excepto ISE-BLOQ. Por lo tanto, se analizarán los resultados de las FSM del conjunto completo para las que dichas implementaciones han sido posibles en el dispositivo FPGA utilizado, que suman un total de 97 FSM. Debido a que el consumo de recursos aumenta cuando las implementaciones se optimizan en velocidad, el número de casos analizados es ligeramente menor que para las implementaciones optimizadas en área.

La optimización en velocidad ha favorecido principalmente a las implementaciones ISE-LUT, que han experimentado un incremento de frecuencia medio del 50 % respecto a los obtenidos con la optimización en área. El incremento ha sido significativamente menor en las implementaciones FSMIM-S, con incrementos medios entre el 6 % y el 8 %. Por último, las menos favorecidas han sido las implementaciones FSMIM-T y FSM-ROM, que presentan incrementos medios entre el 3 % y el 4 %. La tabla 5.63 muestra un resumen estadístico de la frecuencia de operación máxima obtenida en cada batería de pruebas y la figura 5.72, una representación de dichos valores.

Los valores extremos del intervalo de frecuencias de operación máximas alcanzadas por la implementación ISE-LUT han crecido significativamente, abarcando dicho intervalo desde los 43 MHz hasta los 479 MHz (frente a los valores extremos de 5 y 300 MHz obtenidos con la optimización en área). Por otra parte, en promedio, la implementación ISE-LUT es la más rápida en las baterías de pruebas MCNC y BP-24. Sin embargo, en la batería BP-184,

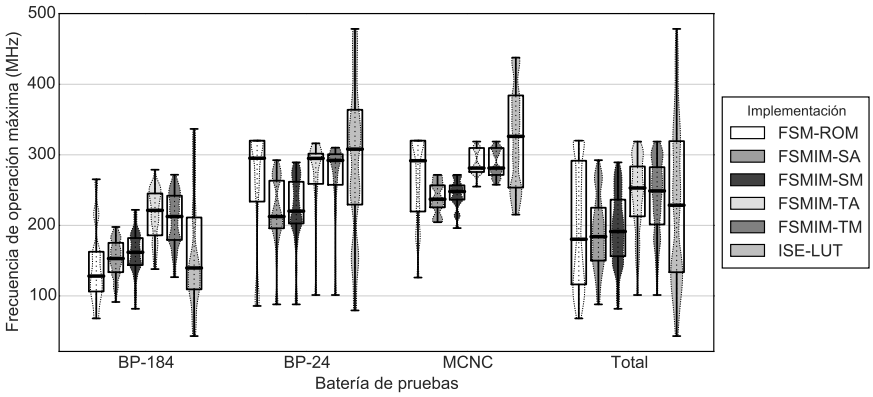
sigue siendo claramente más lenta que las implementaciones FSMIM-T y presenta frecuencias similares a las de las implementaciones FSMIM-S.

Como en anteriores ocasiones, para facilitar la comparación entre las implementaciones, se ha calculado el incremento RPC de la frecuencia de operación máxima respecto a ISE-LUT. La tabla 5.64 muestra un resumen estadístico de los valores obtenidos y la figura 5.73, una representación gráfica.

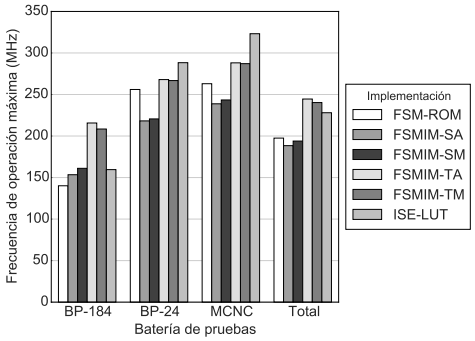
En promedio, para el conjunto completo de FSM, las únicas implementaciones que mejoran la velocidad de ISE-LUT son las implementaciones FSMIM-T. Estas presentan incrementos medios del 25 % (FSMIM-TM) y del 28 % (FSMIM-TA), y porcentajes de éxito superiores al 61 %. En el 25 % de los casos, la implementación FSMIM-TA consigue incrementos entre el 59 % y el 259 %. Por su parte, la implementación FSMIM-TM alcanza

**Tabla 5.63:** Resumen estadístico de la frecuencia de operación máxima (MHz) obtenida por las implementaciones optimizadas en velocidad. Para cada batería de pruebas, las implementaciones están ordenadas de menor a mayor frecuencia media.

| Bat. de pruebas | Implement. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSM-ROM | 140,1 | 51,4 | 68,1 | 106,3 | 128,1 | 162,6 | 265,3 |
| | FSMIM-SA | 153,4 | 26,2 | 91,4 | 133,6 | 153,0 | 175,3 | 197,7 |
| | ISE-LUT | 159,5 | 78,2 | 43,1 | 109,3 | 139,6 | 211,2 | 336,7 |
| | FSMIM-SM | 161,1 | 30,4 | 81,7 | 143,7 | 161,8 | 182,0 | 222,0 |
| | FSMIM-TM | 208,4 | 37,5 | 126,6 | 179,3 | 212,5 | 242,0 | 271,7 |
| | FSMIM-TA | 215,7 | 36,6 | 138,0 | 185,8 | 221,3 | 245,2 | 278,9 |
| BP-24 | FSMIM-SA | 218,2 | 50,7 | 88,0 | 195,8 | 212,5 | 263,3 | 292,4 |
| | FSMIM-SM | 220,6 | 50,2 | 88,0 | 202,7 | 220,2 | 261,8 | 289,3 |
| | FSM-ROM | 256,0 | 85,0 | 85,8 | 233,7 | 295,2 | 320,1 | 320,1 |
| | FSMIM-TM | 266,8 | 56,6 | 101,3 | 257,5 | 292,1 | 300,8 | 310,2 |
| | FSMIM-TA | 268,0 | 57,8 | 101,3 | 258,8 | 295,0 | 301,7 | 316,3 |
| | ISE-LUT | 288,3 | 109,0 | 79,4 | 229,4 | 308,0 | 363,8 | 478,5 |
| MCNC | FSMIM-SA | 238,8 | 22,7 | 204,5 | 225,5 | 237,1 | 256,7 | 271,6 |
| | FSMIM-SM | 243,4 | 21,3 | 196,1 | 236,5 | 248,0 | 256,7 | 271,6 |
| | FSM-ROM | 263,0 | 63,1 | 126,0 | 219,6 | 291,6 | 320,1 | 320,1 |
| | FSMIM-TM | 287,1 | 19,9 | 257,6 | 271,4 | 281,2 | 309,7 | 318,8 |
| | FSMIM-TA | 288,1 | 18,2 | 255,0 | 275,6 | 281,2 | 309,7 | 318,8 |
| | ISE-LUT | 323,2 | 74,6 | 215,2 | 253,6 | 326,2 | 384,2 | 437,8 |
| Total | FSMIM-SA | 188,4 | 50,8 | 88,0 | 150,0 | 184,0 | 225,0 | 292,4 |
| | FSMIM-SM | 193,9 | 50,3 | 81,7 | 156,2 | 191,3 | 236,5 | 289,3 |
| | FSM-ROM | 197,5 | 88,0 | 68,1 | 116,3 | 180,3 | 291,6 | 320,1 |
| | ISE-LUT | 228,0 | 113,2 | 43,1 | 133,6 | 228,6 | 319,4 | 478,5 |
| | FSMIM-TM | 240,3 | 53,8 | 101,3 | 201,3 | 248,8 | 282,5 | 318,8 |
| | FSMIM-TA | 244,5 | 51,8 | 101,3 | 212,8 | 253,0 | 283,4 | 318,8 |




**(a)**




**(b)**

**Figura 5.72:** Frecuencias de operación máximas (MHz) obtenidas por las implementaciones optimizadas en velocidad: (a) Distribución de los valores y (b) valor medio.

valores similares, aunque ligeramente inferiores. A pesar de que las implementaciones FSMIM-S obtienen incrementos medios negativos, estos no son muy significativos (del $-8,9\,\%$ en el peor caso), consiguiendo porcentajes de éxito del $34\,\%$ y $37\,\%$. El peor resultado lo obtiene la implementación FSM-ROM, con un incremento medio del $-17,5\,\%$, aunque el porcentaje de éxito es del $39\,\%$.

En términos generales, la implementación ISE-LUT es la mejor opción en velocidad para las FSM pequeñas. Sin embargo, los incrementos medios y porcentajes de éxito de cada batería de pruebas muestran que la frecuencia de las implementaciones basadas en memoria se ve menos afectada por el aumento de tamaño de las FSM. De hecho, en la batería de pruebas BP-184,

las implementaciones FSMIM-T son la mejor opción (con incrementos medios superiores al 53 % y un porcentaje de éxito del 82 %) y, por otra parte, las implementaciones FSM-ROM y FSMIM-S son una alternativa viable (presentando un porcentaje de éxito del 50 % aproximadamente).

Con objeto de comparar las distintas implementaciones basadas en memoria, se ha calculado el incremento RPC de velocidad respecto a FSM-ROM (que es la que presenta el menor incremento de velocidad medio respecto a ISE-LUT). En promedio para el conjunto completo de FSM, la elección de las implementaciones FSMIM-T permite obtener mejoras significativas respecto al resto de implementaciones basadas en memoria (superiores al 45 % respecto FSM-ROM). Esta implementación obtiene incrementos superiores al 67 % en la batería de pruebas BP-184 y próximos al 16 % en el resto de baterías de pruebas. Por otra parte, aunque la implementación FSM-ROM es mejor alternativa que las implementaciones FSMIM-S en las baterías de pruebas MCNC y BP-24, las implementaciones FSMIM-S son mejor alternativa en la batería de pruebas BP-184 (con incrementos medios próximos al 20 %). Desde el punto de vista cualitativo, la relación entre el tamaño de la ROM de la implementación FSM-ROM y los incrementos de velocidad obtenidos por las implementaciones FSMIM es similar a la observada en las implementaciones optimizadas en área (véase la sección 5.6.1.3, figura 5.70). De hecho, el análisis realizado en dicha sección es válido también para los resultados obtenidos con la optimización en velocidad (con ligeras diferencias de $\pm 2$ puntos en los porcentajes de éxito).

Por otra parte, para estudiar la relación entre el tamaño de las FSM y las mejoras de velocidad obtenidas por las implementaciones FSMIM respecto a ISE-LUT, se han incluido en el análisis las 53 FSM para las que la implementación FSM-ROM ha fallado. Debido a que entre estas FSM se encuentran las de mayor tamaño, los incrementos medios de velocidad han aumentado. Como consecuencia, los incrementos medios de las implementaciones FSMIM-S en esta nueva muestra son positivos (véase la tabla 5.65). La figura 5.75 representa el incremento RPC de velocidad obtenido por las implementaciones FSMIM frente al número de estados de la FSM. Tal como muestran las líneas de regresión local, aunque el incremento es negativo para las FSM más pequeñas, este crece de forma casi lineal con el número de estados de la FSM, presentando mayor pendiente en las implementaciones FSMIM-T. Pueden distinguirse claramente dos zonas en las nubes de puntos, delimitadas por el valor correspondiente a los 32 estados. La implementación ISE-LUT es más rápida que las implementaciones FSMIM en prácticamente todas las FSM con menos de 32 estados. Sin embargo, las implementaciones FSMIM-T son más rápidas en la mayoría de las FSM con más de 32 es-

**Tabla 5.64:** Resumen estadístico del incremento RPC de velocidad obtenido por las implementaciones optimizadas en velocidad respecto a ISE-LUT ( %).

| Bat. de pruebas | Implement. | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|---|
| BP-184 | FSM-ROM | −13,2 | 98,5 | −328,2 | −68,5 | −4,7 | 26,4 | 216,8 |
| | FSMIM-SA | 11,3 | 65,3 | −102,4 | −27,7 | 0,8 | 42,5 | 183,0 |
| | FSMIM-SM | 17,8 | 64,9 | −93,4 | −26,0 | 7,3 | 43,1 | 233,3 |
| | FSMIM-TA | 59,3 | 61,6 | −29,3 | 18,2 | 54,2 | 87,2 | 259,3 |
| | FSMIM-TM | 53,8 | 60,5 | −32,5 | 15,0 | 45,2 | 87,1 | 237,2 |
| BP-24 | FSM-ROM | −20,2 | 74,1 | −246,1 | −39,5 | −11,2 | 22,7 | 89,2 |
| | FSMIM-SA | −28,1 | 30,4 | −76,2 | −51,6 | −29,0 | −3,6 | 28,4 |
| | FSMIM-SM | −26,6 | 31,0 | −76,2 | −50,7 | −29,7 | −2,0 | 36,0 |
| | FSMIM-TA | −2,1 | 33,9 | −57,8 | −24,4 | −7,9 | 24,8 | 69,9 |
| | FSMIM-TM | −2,5 | 33,7 | −57,8 | −24,0 | −9,8 | 21,5 | 75,6 |
| MCNC | FSM-ROM | −25,1 | 20,4 | −70,8 | −36,0 | −27,6 | −19,0 | 22,7 |
| | FSMIM-SA | −34,5 | 23,9 | −71,8 | −49,6 | −38,5 | −12,6 | 4,8 |
| | FSMIM-SM | −32,2 | 25,3 | −71,8 | −49,6 | −35,5 | −8,5 | 10,0 |
| | FSMIM-TA | −10,6 | 22,7 | −41,4 | −23,9 | −16,3 | 10,4 | 29,1 |
| | FSMIM-TM | −11,0 | 21,7 | −41,4 | −23,9 | −15,2 | 8,3 | 28,9 |
| Total | FSM-ROM | −17,5 | 81,9 | −328,2 | −42,1 | −15,8 | 22,2 | 216,8 |
| | FSMIM-SA | −8,9 | 54,7 | −102,4 | −42,2 | −15,5 | 10,1 | 183,0 |
| | FSMIM-SM | −4,7 | 55,6 | −93,4 | −43,6 | −8,9 | 17,1 | 233,3 |
| | FSMIM-TA | 28,1 | 58,6 | −57,8 | −15,1 | 18,5 | 59,1 | 259,3 |
| | FSMIM-TM | 25,0 | 56,5 | −57,8 | −14,2 | 13,3 | 50,8 | 237,2 |

tados y las implementaciones FSMIM-S, en un porcentaje elevado de ellas. Por lo tanto, para FSM grandes, las implementaciones FSMIM-S son una alternativa viable y las implementaciones FSMIM-T, la mejor opción.

Por último, se realizará un breve análisis de los resultados de la implementación ISE-BLOQ, excluyendo las 40 FSM para las que esta ha fallado. La optimización en velocidad no ha favorecido a la implementación

**Tabla 5.65:** Resumen estadístico del incremento RPC de la frecuencia de operación máxima ( %) obtenida por las implementaciones optimizadas en velocidad para la muestra que incluye las 53 FSM en las que la implementación FSM-ROM no es posible (en total, 150 FSM). Las implementaciones están ordenadas de menor a mayor frecuencia media.

| Implementación | Media | Desv. | Mín. | Q1 | Q2 | Q3 | Máx. |
|---|---|---|---|---|---|---|---|
| FSMIM-SA | 7,4 | 71,0 | −144,1 | −38,3 | −7,3 | 30,8 | 231,7 |
| FSMIM-SM | 9,3 | 67,6 | −106,1 | −35,0 | −3,0 | 35,4 | 233,3 |
| FSMIM-TM | 42,5 | 69,3 | −64,1 | −10,2 | 26,2 | 78,7 | 237,2 |
| FSMIM-TA | 46,1 | 70,5 | −67,3 | −10,1 | 33,6 | 77,4 | 259,3 |

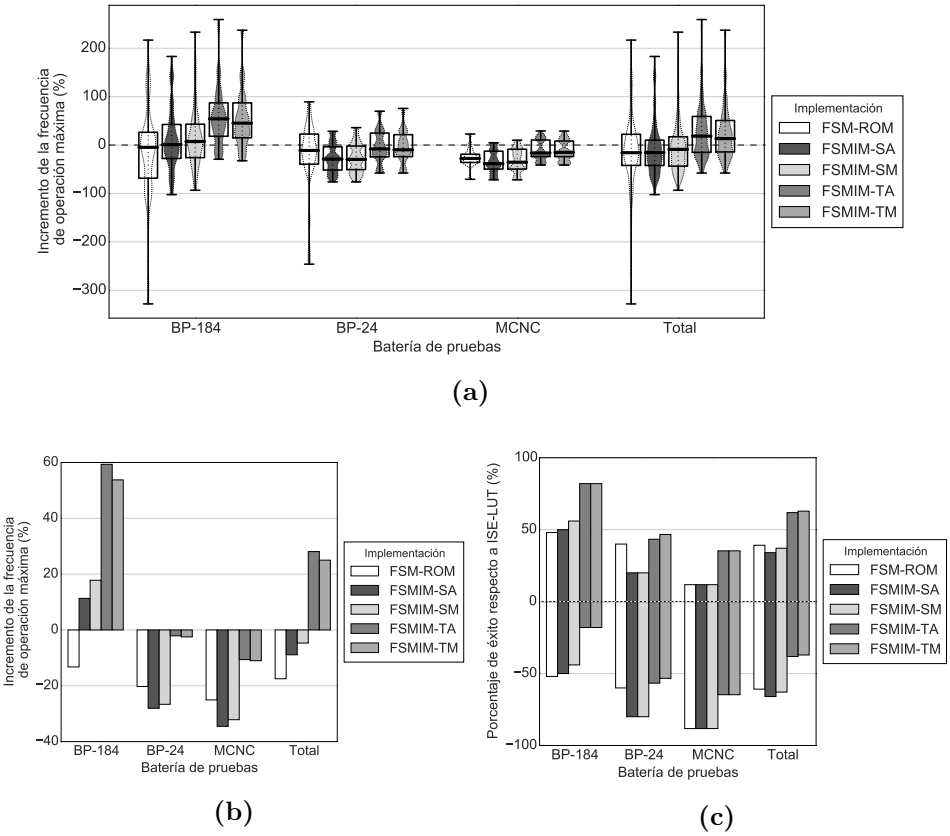



**Figura 5.73:** Incremento RPC de velocidad obtenido por las implementaciones optimizadas en velocidad respecto a ISE-LUT: (a) Distribución de los valores obtenidos, (b) valor medio y (c) porcentaje de éxito.

ISE-BLOQ frente al resto de implementaciones estudiadas, por lo que, de nuevo, es la que presenta los peores resultados de velocidad. Obtiene un incremento negativo de frecuencia respecto a la implementación ISE-LUT del $-84\,\%$, consiguiendo mejoras de velocidad en un sólo caso.

#### 5.6.2.2. Estudio del consumo de LUT

Para el estudio del consumo de LUT se han seleccionado las mismas FSM que para el estudio de la frecuencia de operación máxima. En términos generales, las implementaciones optimizadas en velocidad consumen más recursos lógicos que las optimizadas en área. En promedio, el mayor incremento relativo del número de LUT consumidas lo ha experimentado la implementación

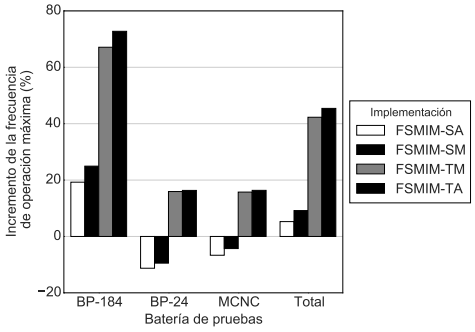



**Figura 5.74:** Incremento RPC medio de velocidad obtenido por las implementaciones de FSM basadas en memoria respecto a FSM-ROM. Los resultados han sido obtenidos con las implementaciones optimizadas en velocidad.

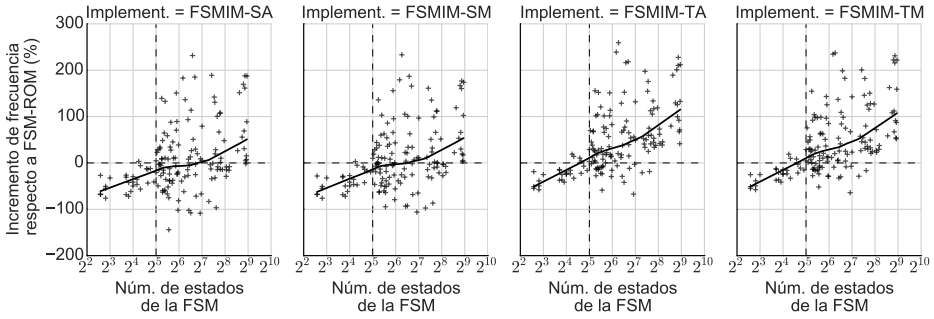



**Figura 5.75:** Incremento RPC de velocidad obtenido por las implementaciones FSMIM optimizadas en velocidad respecto a ISE-LUT frente al número de estados de la FSM. Los resultados han sido calculados incluyendo las 53 FSM en las que la implementación FSM-ROM no es posible (en total se han usado 150 FSM).

FSM-ROM (con un incremento medio del 25 %). Por otra parte, las implementaciones ISE-LUT y FSMIM-T han sufrido incrementos medios próximos al 14 %, y las implementaciones FSMIM-S, incrementos que no superan el 3 %.

Sin embargo, debido a las grandes diferencias en el consumo de LUT que existen entre las implementaciones basadas en memoria y la implementación ISE-LUT, estos valores relativos corresponden a cantidades despreciables en algunos casos. En términos absolutos, la implementación ISE-LUT ha sido la que ha aumentado el consumo de LUT en mayor medida con un incremento medio de 154 LUT, mientras que el incremento no supera las 6 LUT de

media en las implementaciones FSMIM-S y es inferior a las 2 LUT en el resto de implementaciones.

En cualquier caso, las diferencias entre los resultados obtenidos con las implementaciones optimizadas en velocidad y los obtenidos con las optimizadas en área no han sido suficientes para cambiar significativamente la reducción RPC del consumo de LUT respecto a ISE-LUT (las diferencias absolutas entre las reducciones medias no superan los dos puntos porcentuales). Por lo tanto, el análisis realizado en la sección 5.6.1.2 sobre el consumo de LUT sigue siendo válido para esta sección.

El consumo de bloques no ha cambiado respecto a las implementaciones optimizadas en área. Sin embargo, el número de LUT ahorradas sí ha crecido. Esto es debido a que, en términos absolutos, el incremento del número de LUT de la implementación ISE-LUT ha sido mayor que el de las implementaciones basadas en memoria. Como consecuencia, se ha producido un aumento del NLAB. En promedio, para el conjunto completo de FSM, el NLAB ha crecido entre un 13 % y un 16 %, lo que ha supuesto un incremento medio de 25 y 32 LUT/bloque para las implementaciones FSMIM-T, de 41 y 57 LUT/bloque para las implementaciones FSMIM-S, y de 7 LUT/bloque para FSM-ROM.

Al igual que ha ocurrido con las reducciones en el consumo de LUT, estos valores no han cambiado significativamente el incremento RPC del NLAB obtenido por las implementaciones FSMIM respecto a la implementación FSM-ROM. Comparados con los incrementos obtenidos con la optimización en área, el NLAB medio ha crecido menos del 6 % en las implementaciones FSMIM-S y menos del 3 % en las implementaciones FSMIM-T. Por lo tanto, el análisis realizado en la sección 5.6.1.2 sobre el NLAB obtenido por las implementaciones basadas en memoria sigue siendo válido para esta sección.

### 5.6.3. Conclusiones

Al margen del análisis de las prestaciones de las arquitecturas FSMIM, la conclusión más clara que se puede obtener de los resultados experimentales es que las implementaciones ISE-BLOQ no han mostrado ningún beneficio respecto al resto de implementaciones estudiadas, por lo tanto, no serán mencionadas en este apartado.

Del conjunto de FSM consideradas inicialmente para realizar los experimentos, las técnicas de implementación de FSMIM se han podido aplicar en el 79 % de ellas, que fueron las seleccionadas para las pruebas independientes del dispositivo. En estas pruebas, las arquitecturas FSMIM han demostrado una gran capacidad para reducir el tamaño de la ROM de la arquitectura

FSM-ROM. Especialmente importantes han sido los resultados obtenidos por las nuevas estrategias de optimización con la arquitectura FSMIM-S, que han conseguido reducciones medias superiores al 70 % en el conjunto completo de FSM y porcentajes de éxito por encima del 96 % en todas las baterías de pruebas (llegando al 100 % para FSMIM-SA).

Respecto al consumo de LUT, en promedio, el uso de la arquitectura FSMIM-S permite ahorrar más del 82 % de las LUT usadas por las implementaciones ISE-LUT, mientras que las arquitecturas FSMIM-T y FSM-ROM permiten ahorrar más del 95 %. La arquitectura FSM-ROM no requiere ninguna LUT cuando la profundidad de la ROM es menor que la profundidad máxima de los bloques de memoria (16 Kpalabras en los dispositivos de la familia Spartan-6). Sin embargo, las pruebas de implementación han mostrado que, cuando la profundidad es mayor, el banco de multiplexores de la ROM requerido por las implementaciones FSM-ROM consume más LUT que las implementaciones FSMIM-T. Como consecuencia, en el 46 % de los casos estudiados, las implementaciones FSMIM-T han sido mejor opción que las implementaciones FSM-ROM en consumo de LUT y de bloques.

A pesar de presentar el mayor consumo de LUT, las implementaciones FSMIM-S son las más eficientes en el uso de los bloques, ya que ahorran más LUT por cada bloque usado que el resto de implementaciones basadas en memoria. En cualquier caso, todas las implementaciones FSMIM consiguen un NLAB mayor que la implementación FSM-ROM incluso en la mayoría de los casos en los que esta no consume ninguna LUT. En promedio, alcanzan valores del NLAB entre 22 y 35 veces mayores que los obtenidos por FSM-ROM. Por otra parte, los valores medios del NLAB de las implementaciones FSMIM-S (348 y 437 LUT/bloque) son mayores que la proporción máxima entre el número de LUT y bloques existente en los dispositivos de la familia Spartan-6 (344 LUT/bloque en el dispositivo XC6SLX150T). Esto indica que, en una amplia mayoría de los casos, las implementaciones FSMIM-S hacen un uso más eficiente de los recursos disponibles en los dispositivos FPGA que la implementación ISE-LUT.

A partir del análisis anterior, se puede concluir que las arquitecturas FSMIM mejoran las prestaciones de la arquitectura FSM-ROM en la mayoría de los escenarios, presentando una alternativa para cada objetivo de diseño. Si el número de bloques de memoria libres es limitado, las implementaciones FSMIM son mejor alternativa que FSM-ROM, siendo la implementación FSMIM-SA la mejor opción. Por otra parte, si la reducción del consumo de LUT es uno de los objetivos principales, la implementación FSM-ROM es la mejor opción sólo para FSM pequeñas (aquellas en las que la implementación FSM-ROM requiere una ROM con menor profundidad que

los bloques de memoria); sin embargo, en el resto de casos las implementaciones FSMIM-T son mejor alternativa, principalmente la implementación FSMIM-TM. En términos generales, las implementaciones FSMIM son las que hacen un uso más eficiente de los recursos, ahorrando más LUT por cada bloque usado, siendo FSMIM-SA la más eficiente de todas. La posibilidad de aprovechar eficientemente los diferentes tipos de recursos disponibles en las FPGA permite implementaciones en dispositivos más pequeños (respecto al número de recursos), lo que favorece la reducción de costes. En este sentido, las dos arquitecturas FSMIM junto con las diferentes estrategias de optimización ofrecen un amplio conjunto de alternativas para alcanzar los objetivos de diseño.

Respecto a la velocidad de operación, el objetivo de optimización utilizado en el proceso de síntesis e implementación ha tenido un efecto significativo en los resultados. En primer lugar, los resultados de las implementaciones optimizadas en área han mostrado que las implementaciones de FSM basadas en memoria son más rápidas que ISE-LUT en la mayoría de los casos, siendo las implementaciones FSMIM-T las que presentan, en promedio, las mayores frecuencias de operación máximas.

En cualquier caso, las implementaciones FSMIM sufren una menor degradación de la velocidad con el tamaño de la FSM que las implementaciones ISE-LUT y FSM-ROM, lo que les permite mejorar sus resultados en un amplio porcentaje de los casos. Las implementaciones FSMIM-T, por una parte, consiguen superar la velocidad de las implementaciones ISE-LUT en el 94 % de los casos, obteniendo un incremento de velocidad medio cercano al 250 %; por otra parte, mejoran la velocidad de las implementaciones FSM-ROM en el 66 % de los casos, alcanzando incrementos de velocidad superiores al 50 % en la mayoría de ellos. En el 47 % de los casos, las implementaciones FSMIM-T son mejor opción que FSM-ROM considerando todos los parámetros analizados: velocidad de operación, consumo de LUT y consumo de bloques. Es decir, en el 47 % de los casos, las implementaciones FSMIM-T mejoran alguno de los parámetros respecto a FSM-ROM sin empeorar el resto de parámetros. Por su parte, las implementaciones FSMIM-S, son más rápidas que ISE-LUT en el 72 % de los casos, con incrementos medios superiores al 138 %; además, consiguen mejorar la velocidad de FSM-ROM en más del 40 % de los casos.

La comparación de las prestaciones de las implementaciones de FSM estudiadas permite concluir que, en los diseños en los que la velocidad no es crítica, las implementaciones FSMIM presentan el mejor equilibrio entre consumo de recursos y velocidad de operación.

Respecto a los resultados de las implementaciones optimizadas en ve-

locidad, la implementación ISE-LUT ha sido la que ha experimentado el mayor incremento de velocidad respecto a los resultados de las implementaciones optimizadas en área. Como consecuencia, esta implementación alcanza una velocidad media mayor que las implementaciones FSM-ROM y FSMIM-S. Sin embargo, las implementaciones FSMIM-T siguen siendo las que consiguen, en promedio, las mayores frecuencias de operación máximas, consiguiendo incrementos medios superiores al 25 % respecto a ISE-LUT y porcentajes de éxito superiores al 61 %. Por su parte, las implementaciones FSMIM-S presentan porcentajes de éxito del 34 % como mínimo.

Desde el punto de vista cualitativo, la optimización en velocidad no ha cambiado la relación entre las frecuencias obtenidas por las implementaciones FSM-ROM y FSMIM, ni la relación entre el tamaño de las FSM y la degradación de la velocidad. Las implementaciones FSMIM siguen siendo las que se ven menos afectadas por el aumento del tamaño de las FSM. Mientras que la implementación ISE-LUT es más rápida que las implementaciones FSMIM en prácticamente todas las FSM con menos de 32 estados, las FSMIM-T son más rápidas en la mayoría de las FSM con más de 32 estados, y las implementaciones FSMIM-S lo son en un porcentaje importante de ellas. Por lo tanto, aunque la mejor opción para FSM pequeñas es la implementación ISE-LUT; para FSM grandes, las implementaciones FSMIM-S son una alternativa viable y las implementaciones FSMIM-T son la mejor opción.

Capítulo 6

# Estado de las técnicas para la implementación basada en memoria de FSM

En este apéndice se realiza una breve revisión de las diferentes técnicas propuestas en la literatura en el contexto de la implementación de FSM con recursos de memoria; contexto en el que se ha desarrollado el trabajo de investigación presentado en esta tesis doctoral. El objetivo de estas técnicas es el uso eficiente tanto de los bloques de memoria empotrados como de las celdas lógicas disponibles en los dispositivos FPGA.

En los casos en los que ha sido posible, se han comparado los resultados obtenidos con las técnicas analizadas y los obtenidos con las arquitecturas FSMIM.

## 6.1. Técnicas basadas en descomposición funcional de la función de transición

Sea $X = \{x_1, \ldots, x_q\}$ un conjunto de variables booleanas y $t$ una función tal que $t(x_1, \ldots, x_q) \in \mathbb{B}^z$. Sean $U$ y $V$ dos subconjuntos de $X$ no necesariamente disjuntos tales que $U \cup V = X$, donde $U$ se denomina conjunto de *variables libres* y $V$, conjunto de *variables ligadas*. Supóngase que se renombran las variables de $X$ de forma que $U = \{x_1, \ldots, x_r\}$ y $V = \{x_{q-s+1}, \ldots, x_q\}$, siendo $r = |U|$ y $s = |V|$; entonces, para toda $q$-tupla $x = (x_1, \ldots, x_q)$, se define $x^U = (x_1, \ldots, x_r)$ y $x^V = (x_{q-s+1}, \ldots, x_q)$.

Dadas dos funciones $g : \mathbb{B}^s \rightarrow \mathbb{B}^w$ y $f : \mathbb{B}^{r+w} \rightarrow \mathbb{B}^z$, se dice que el par de funciones $(f, g)$ representa una descomposición en serie de $t$ respecto a $(U, V)$ si $t(x) = f(x^U, g(x^V))$ para todo $x \in \mathbb{B}^q$ [Rawski et al., 2005] (véase la figura 6.2a). En el caso de que $r + w < q$, el número de entradas de la función $f$ será menor que el número de entradas de la función original $t$.

La técnica propuesta en [Rawski et al., 2011] consiste en aplicar la descomposición funcional a la función de transición de la FSM, implementando $f$ en una memoria ROM y $g$ en un circuito combinacional (denominado *modificador de direcciones*), tal como muestra la figura 6.2b. Esta técnica consigue reducir la profundidad de la ROM en los casos en los que $r+w < q+p$.

Para el estudio experimental realizado en [Rawski et al., 2011], los autores de la técnica seleccionaron cinco de las 46 FSM que componen la batería de pruebas MCNC. En el estudio no se indica el criterio de selección utilizado; por tanto, no se conoce el número de casos en los que la técnica puede ser aplicada con éxito. Como consecuencia, no hay información suficiente para estimar el grado de representatividad de los resultados seleccionados.

El dispositivo FPGA empleado fue EPF10K10LC84-3, de la familia de

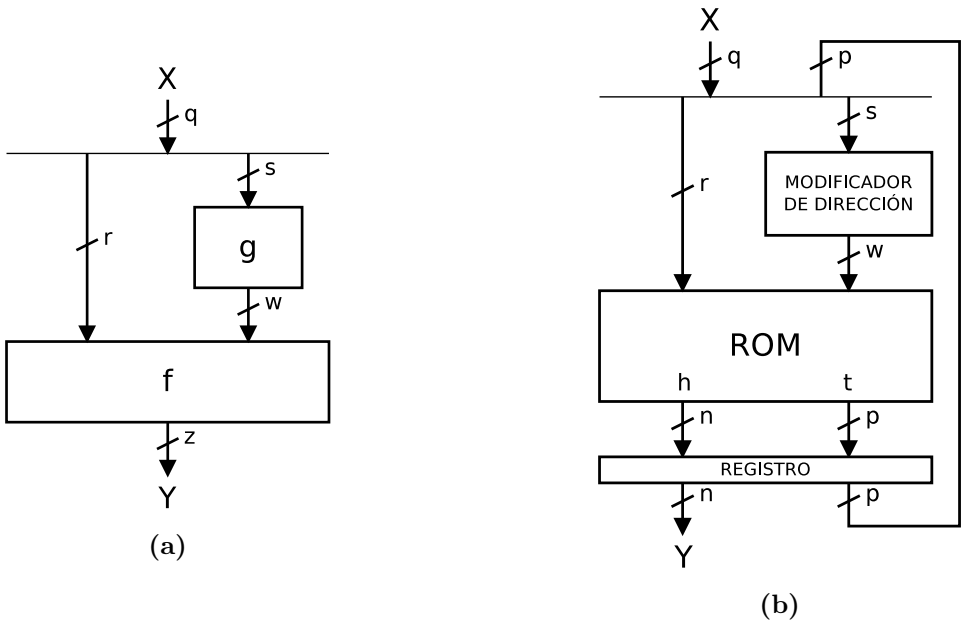



**Figura 6.2:** Descomposición funcional: (a) representación esquemática de una descomposición en serie y (b) implementación de una FSM mediante la técnica basada en descomposición funcional.

dispositivos *FLEX 10K* de Altera. Este dispositivo dispone de LUT de cuatro entradas y de bloques de memoria empotrados de 2 Kbits, que permiten las siguientes configuraciones: 2K × 1, 1K × 2, 512 × 4 y 256 × 8.

Debido a que se han empleado dispositivos FPGA y herramientas de síntesis de fabricantes diferentes, los resultados obtenidos con la técnica basada en descomposición funcional no han podido comparase con los obtenidos en el estudio experimental presentado en esta tesis doctoral. Como consecuencia, ha sido necesaria la realización de nuevos experimentos diseñados expresamente para esta comparativa. Se ha considerado que el escaso número de casos que se pueden comparar no justifican el esfuerzo de adaptar los experimentos a una herramienta de síntesis nueva. Se ha optado por utilizar un dispositivo de la familia Spartan-3 de Xilinx para estimar el consumo de recursos de las arquitecturas FSMIM en el dispositivo FLEX 10K de Altera. El procedimiento para realizar la estimación ha sido el siguiente:

- Se genera una descripción VHDL estructural de la ROM de las FSMIM empleando las geometrías de bloque del dispositivo FLEX 10K (véase la herramienta ROMGEN, apéndice G). En esta descripción estructural, cada bloque de 2 Kbits se describe como una ROM independiente que se implementa con un sólo bloque del dispositivo Spartan-3 (los dispositivos Spartan-3 disponen de bloques de 18 Kbits). Esto permite evaluar el consumo de bloques de las arquitecturas FSMIM en el dispositivo de Altera.

- A pesar de que las LUT de ambos dispositivos son de 4 entradas, la arquitectura interna de cada dispositivo es muy diferente. En los dispositivos Spartan-3 los recursos lógicos se organizan en *Bloques Lógicos Configurables* (CLB, sigla del inglés Configurable Logic Block). Cada CLB se divide en cuatro *slices* y cada *slice* contiene dos LUT de 4 entradas y dos multiplexores empotrados (además de otros recursos que no son relevantes para este estudio). Los multiplexores empotrados se denominan F5MUX y FiMUX, donde FiMUX es F6MUX, F7MUX o F8MUX, según el tipo de *slice* en el que se encuentre. Se trata de multiplexores de dos entradas que permiten la implementación de multiplexores de 4, 16, o 32 entradas, o de cualquier función de entre cinco y ocho entradas, empleando un sólo nivel de LUT [Xilinx, 2005] (véase la figura 6.3).

  El dispositivo FLEX 10K no dispone de multiplexores empotrados. Por tanto, para implementar en este dispositivo funciones de cinco a ocho entradas siguiendo el esquema mostrado en la figura 6.3, se requieren

LUT adicionales que sustituyan a los multiplexores (una LUT de 4 entradas puede implementar un multiplexor 2:1).

Una vez realizada la implementación de las descripciones VHDL de las FSMIM, cada multiplexor empotrado se ha computado como una LUT. Esto permite estimar el consumo de LUT de las FSMIM en el dispositivo FLEX 10K a partir de los resultados de implementación obtenidos con el dispositivo Spartan-3.

Sin embargo, debe tenerse en cuenta durante el análisis de los resultados que el cómputo realizado puede dar lugar a una sobrestimación del número de LUT consumidas, ya que no se ha considerado que el dispositivo FLEX 10K permite formar cadenas de LUT, utilizando puertas lógicas (*OR* o *AND*) para conectar las salidas de LUT adyacentes.

Una de las cinco FSM seleccionadas en [Rawski et al., 2011] ha sido descartada debido a que no cumplía los requisitos para generar una FSMIM. A partir de las cuatro restantes se han generado las FSMIM-S y las FSMIM-T correspondientes utilizando la estrategia de optimización A-PLE con ajuste

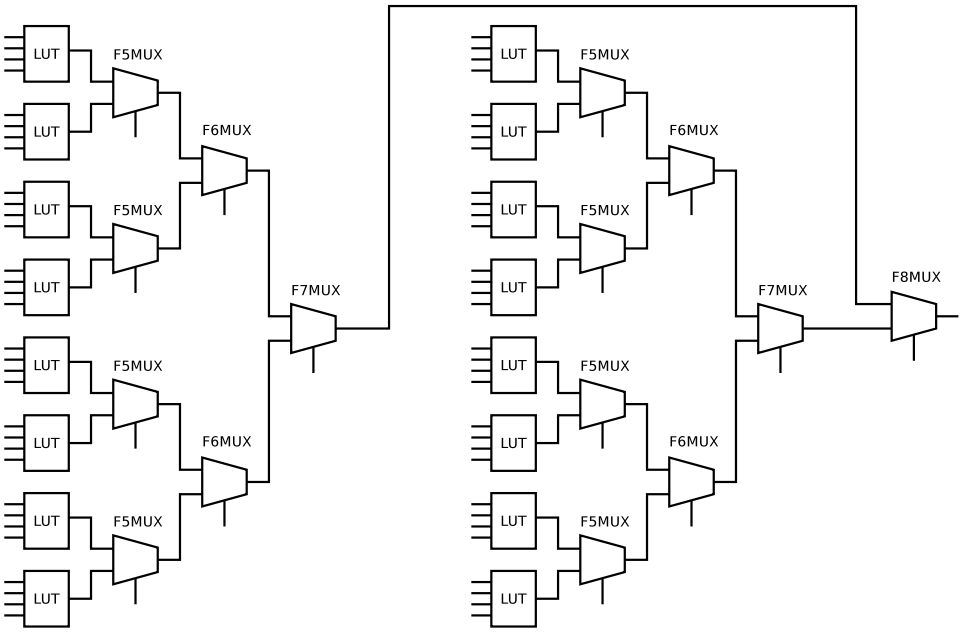



**Figura 6.3:** Esquema para la implementación de un multiplexor 32:1 o de cualquier función de ocho entradas empleando los multiplexores empotrados del dispositivo Spartan-3. El dígito contenido en el nombre de cada multiplexor hace referencia al número de entradas de la función que se puede implementar con dicho multiplexor.

de profundidad. La tabla 6.1 muestra el consumo de bloques y de LUT de las diferentes implementaciones de FSM comparadas. Para algunas FSM, la técnica de descomposición funcional permite obtener más de una solución. Las columnas etiquetadas "DF1" y "DF2" contienen las dos soluciones publicadas en [Rawski et al., 2011] para dichas FSM.

Como puede observarse, en promedio, las técnicas se pueden ordenar claramente en orden creciente del consumo de bloques y decreciente del consumo de LUT como sigue: DF1, DF2, FSMIM-S y FSMIM-T. Suponiendo que la muestra utilizada fuera representativa, podría afirmarse que las técnicas basadas en descomposición funcional y las basadas en las FSMIM se complementan perfectamente, ofreciendo al diseñador diferentes alternativas para ajustar el consumo de ambos tipos de recursos.

Se ha calculado el NLAB de las diferentes técnicas tomando como referencia los datos publicados en [Rawski et al., 2011] referentes al consumo de LUT de las implementaciones convencionales de FSM basada en celdas lógicas. La tabla 6.2 muestra los valores obtenidos. En tres de los cuatro casos estudiados, las técnicas basadas en descomposición funcional consiguen ahorrar un mayor número de LUT por cada bloque usado, mientras que ambas arquitecturas FSMIM obtienen mejores resultados para la FSM *s386*.

Respecto a la velocidad, en [Rawski et al., 2011] no se ofrecía ninguna información que permitiera estimar la frecuencia de operación máxima o el incremento (o reducción) de velocidad respecto a las implementaciones convencionales de FSM. Sin embargo, el mayor consumo de LUT de las técnicas basadas en descomposición funcional es un indicio de que la velocidad de las implementaciones FSMIM podría ser mayor en términos generales que la velocidad de las implementaciones basadas en descomposición funcional.

**Tabla 6.1:** Consumo de recursos de las implementaciones de FSM generadas con la técnica basada descomposición funcional (dispositivo FLEX10K) y de las implementaciones de FSMIM (consumo estimado a partir de un dispositivo Spartan-3).

| FSM | DF1 | | DF2 | | FSMIM-S | | FSMIM-T | |
|---|---|---|---|---|---|---|---|---|
| | Bloques | LUT | Bloques | LUT | Bloques | LUT | Bloques | LUT |
| cse | 1 | 21 | 2 | 9 | 3 | 11 | 4 | 3 |
| mark1 | 1 | 16 | 2 | 9 | 3 | 6 | 3 | 1 |
| s1 | 1 | 107 | 2 | 69 | 3 | 57 | 6 | 12 |
| s386 | 1 | 35 | 1 | 35 | 2 | 13 | 2 | 4 |
| Media | 1 | 44,8 | 1,8 | 30,5 | 2,8 | 21,8 | 3,8 | 5 |

**Tabla 6.2:** NLAB obtenido por las implementaciones de FSM generadas con la técnica basada en descomposición funcional y por las implementaciones de FSMIM.

| FSM | DF1 | DF2 | FSMIM-S | FSMIM-T |
|---|---|---|---|---|
| cse | 71 | 41,5 | 27 | 22,2 |
| mark1 | 22 | 14,5 | 10,7 | 12,3 |
| s1 | 57 | 47,5 | 35,7 | 25,3 |
| s386 | 19 | 19 | 20,5 | 25 |
| Media | 42,2 | 30,6 | 23,5 | 21,2 |

## 6.2. Técnicas basadas en descomposición estructural de FSM

Básicamente, la descomposición estructural de una FSM consiste en dividir la arquitectura que implementa la función de transición y de salida en varios niveles. La figura 6.4 muestra un ejemplo de descomposición estructural de una FSM [Barkalov y Palagin, 1997]. El nivel superior es un circuito combinacional que codifica alguna característica de la FSM. En el ejemplo de la figura 6.4, el bloque $C$ codifica los diferentes conjuntos de *microoperaciones* de la FSM[1]. Comparada con la implementación convencional con puertas lógicas de las funciones de transición y de salida de la FSM, el beneficio potencial de este circuito codificador es que implementa menos funciones booleanas, por lo que requiere menos recursos lógicos (LUT en el caso de las FPGA). En el ejemplo de la figura 6.4, esto ocurre si el tamaño en bits del código asignado a las microoperaciones ($z$) es menor que el número de bits de salida ($n$). Los siguientes niveles son los encargados de realizar la decodificación. Por ejemplo, en la figura 6.4, el bloque $D$ genera las salidas asociadas a cada código de microoperación. El circuito que implementa la decodificación tiene una estructura más regular, lo que significa que puede ser implementado con bloques de memoria empotrados. En resumen, en términos genéricos, la descomposición estructural puede reducir el consumo de LUT de la implementación de una FSM por dos vías: por una parte, el bloque codificador consume menos LUT que la función de transición y, por otra, el bloque decodificador puede implementarse utilizando bloques de memoria en lugar de LUT.

[1]En este contexto, se denomina microoperación a cada señal de salida que se activa en la transición.

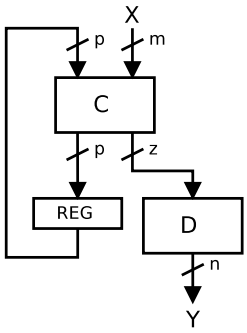



**Figura 6.4:** Ejemplo de descomposición estructural de una FSM.

### 6.2.1. Implementación multi-nivel

Bajo el nombre de *implementación multi-nivel* de FSM, en [Bukowiec, 2008] se proponen diferentes técnicas inspiradas en la descomposición estructural, que aplican una codificación múltiple [Bukowiec, 2005] de algunos parámetros de la FSM, tales como las *microinstrucciones*[2] o los estados. El conjunto de valores del parámetro elegido se separa en varios subconjuntos, no necesariamente disjuntos, y cada valor se codifica de forma independiente en cada subconjunto. Esto significa que un mismo código puede estar repetido en los diferentes subconjuntos.

Por ejemplo, para cada estado $e$ de la FSM se puede definir un conjunto de microinstrucciones diferente, formado por las microinstrucciones que se activan en alguna de las transiciones de $e$. A cada microinstrucción del conjunto se le asigna un código diferente, de forma que las microinstrucciones se pueden identificar por el par formado por el código del estado $e$ y el código de la microinstrucción asociado al conjunto correspondiente. La codificación múltiple da lugar a arquitecturas diferentes a la mostrada en la figura 6.4.

La figura 6.5 muestra una de las arquitecturas propuestas en [Bukowiec, 2008]. El circuito combinacional $C$ realiza una codificación múltiple de los estados y las microinstrucciones. Al igual que en el ejemplo de la figura 6.4, se asigna un código binario a las diferentes microinstrucciones de la FSM. Sea $E$ el conjunto de estados de la FSM y $Z = \{\mu_1, \ldots, \mu_{|Z|}\}$ el conjunto de códigos de microinstrucción. El número de bits de codificación necesarios es $z = \lceil \log_2 |Z| \rceil$. Sin embargo, en esta arquitectura, para cada código microinstrucción $\mu_i \in Z$ se define un subconjunto de estados $E_{\mu_i} \subseteq E$ tal que un estado $e \in E$ pertenece a $E_{\mu_i}$ si existe una transición en la que se

[2]En este contexto, una microinstrucción es el conjunto de microoperaciones que se activan en una transición.

activa la microinstrucción $\mu_i$ y en la que $e$ es el estado siguiente. Los estados de cada conjunto $E_{\mu_i}$ se codifican empleando una codificación binaria independiente de la del resto de subconjuntos (estos códigos serán denominados *discriminantes*). De esta forma, un estado $e \in E_{\mu_i}$ se puede identificar por el par $(c_{\mu_i}(e), \mu_i)$, donde $c_{\mu_i}(e)$ es el discriminante asignado al estado $e$ en el subconjunto $E_{\mu_i}$. Obviamente, en el caso de que un estado pertenezca a diferentes subconjuntos, le corresponderán diferentes pares (aunque, puesto que no hay relación entre los discriminantes usados para los diferentes subconjuntos, puede ocurrir que un estado tenga asociado el mismo discriminante en algunos de los subconjuntos a los que pertenece).

En la figura 6.5, el circuito $C$ codifica el estado siguiente utilizando el par $(c_{\mu_i}(e), \mu_i)$; por lo tanto, el número de bits de codificación necesarios depende el subconjunto con mayor cardinalidad; es decir $v = \text{máx}_{\mu \in Z} \lceil \log_2 |E_\mu| \rceil$. El consumo de LUT de la implementación de este circuito será menor que el de las funciones de transición y salida de la FSM si $z + v < n + p$, donde $p = \lceil \log_2 |E| \rceil$ es el número de bits de codificación de estados de la FSM y $n$ es el número de bits de salida. Por otra parte, los circuitos $D_1$ y $D_2$ realizan la decodificación del estado siguiente y de las salidas de la FSM. Ambos pueden ser implementados con memoria ROM.

El estudio experimental presentado en [Bukowiec, 2008] se realizó con prácticamente todas las FSM de la batería de pruebas MCNC, aplicándose todas las técnicas basadas en la implementación multi-nivel a cada una de ellas. El dispositivo FPGA empleado fue XCV50-BG256-6 de la familia Virtex de Xilinx. Debido a las diferencias entre este dispositivo y el empleado

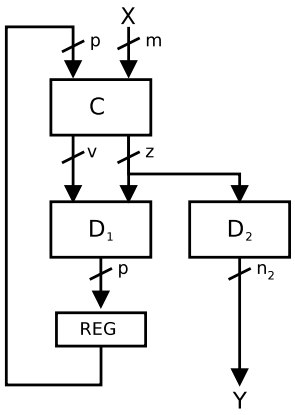



**Figura 6.5:** Descomposición estructural de una FSM mediante codificación múltiple de los estados y las microinstrucciones.

en el estudio experimental presentado en esta tesis, ha sido necesaria la realización de nuevos experimentos para poder comparar estas técnicas con las FSMIM.

Los dispositivos de la familia Virtex están obsoletos y no se incluyen en la versión 14.6 de la herramienta de síntesis *ISE WebPACK*[3], por lo que el consumo de recursos de las arquitecturas FSMIM ha sido estimado mediante un procedimiento similar al empleado en la sección 6.1, con las siguientes particularidades:

- La descripción VHDL estructural de la ROM se ha generado con las geometrías de bloque de los dispositivos de la familia Virtex: $4K \times 1$, $2K \times 2$, $1K \times 4$, $512 \times 8$ y $256 \times 16$.

- Al igual que los dispositivos Spartan-3, los dispositivos Virtex disponen de LUT de cuatro entradas y de multiplexores empotrados equivalentes a los multiplexores F5MUX y F6MUX. Sin embargo, no disponen de los multiplexores F7MUX y F8MUX. Por este motivo, la estimación del consumo de LUT en los dispositivos Virtex se ha calculado sumando el número de LUT requeridas en el dispositivo Spartan-3 más el número de multiplexores F7MUX y F8MUX.

De las FSM utilizadas en el estudio experimental realizado en [Bukowiec, 2008], se han seleccionado aquellas que cumplen los requisitos para generar una FSMIM (23 casos en total). Las FSMIM se han generado con ajuste de profundidad mediante la estrategia de optimización A-PLE. El consumo de bloques y de LUT obtenido por las técnicas comparadas se muestra en la tabla 6.3. La columna "Multi-nivel" contiene para cada FSM el mejor resultado obtenido por las diferentes técnicas de implementación multi-nivel, que corresponde al de menor consumo de LUT y de bloques (es ese orden de prioridad)[4].

Comparando los valores medios obtenidos, se observa que las implementaciones FSMIM consumen menos LUT que las implementaciones multi--nivel a expensas de un aumento del consumo de bloques. Sin embargo, un análisis más detallado de los resultados de la tabla muestra que existen casos en los que las implementaciones FSMIM mejoran tanto el consumo de LUT como de bloques. Se ha calculado el porcentaje de éxito neto de la reducción RPC del consumo de LUT y de la reducción RPC del consumo de bloques obtenido por las implementaciones FSMIM respecto a las implementaciones multi-nivel. Teniendo en cuenta este cálculo, la arquitectura FSMIM-S, es

[3]Esta ha sido la versión empleada en el estudio experimental realizado en esta tesis.
[4]Este es el criterio empleado en [Bukowiec, 2008].

mejor alternativa que la implementación multi-nivel en el 70 % de los casos, mientras que el número de casos en los que la implementación multi-nivel es mejor alternativa no supera el 9 %. Por su parte, la arquitectura FSMIM-T es mejor alternativa que la implementación multi-nivel en el 52 % de los casos, mientras que la implementación multi-nivel no es mejor alterativa en ninguno de los casos estudiados.

Para facilitar la comparación entre los resultados, se han calculado las reducciones (o incrementos) del número de bloques, número de LUT y NLAB obtenidos por las implementaciones FSMIM respecto a las implementaciones multi-nivel. La tabla 6.4 muestra los valores medios. En promedio la implementación multi-nivel consigue reducir el número de bloques respecto a las implementaciones FSMIM[5]. Sin embargo, el porcentaje de reducción es pequeño en comparación con la reducción del número de LUT conseguida por las implementaciones FSMIM. Como resultado, las implementaciones FSMIM obtienen importantes incrementos en el NLAB respecto a las implementaciones multi-nivel.

### 6.2.2. Descomposición estructural con partición de las microoperaciones

La arquitectura propuesta en [Barkalov et al., 2012] se muestra en la figura 6.6. Al igual que en la arquitectura mostrada en la figura 6.5, por una parte, el circuito combinacional $C$ realiza una codificación múltiple de los estados y microoperaciones, y, por otra, los circuitos $D_1$ y $D_2$ decodifican el estado siguiente y las salidas de la FSM. Sin embargo, en esta arquitectura, para aprovechar las diferentes geometrías disponibles en los bloques de memoria empotrados de los dispositivos FPGA, y reducir así el fraccionamiento interno de dichos bloques, se realiza una partición del conjunto de microoperaciones en dos subconjuntos (denominados $Y^1$ e $Y^2$ en la figura 6.6), implementándose la decodificación de uno de los subconjuntos en $D_1$ y la del otro en $D_2$. Esto permite ajustar el ancho de la memoria necesaria para implementar $D_1$ o $D_2$.

En el estudio presentado en [Barkalov et al., 2012] se indica que la técnica fue probada con todas las FSM de la batería de pruebas MCNC. Sin embargo, solo se muestran los resultados de diez FSM. Puesto que no se aclara el criterio de selección empleado, se desconoce el grado de representatividad de estos resultados. Los experimentos se realizaron con el dispositivo

[5]Obsérvese que la reducción RPC mostrada en la tabla es negativa en el caso del número de bloques.

**Tabla 6.3:** Consumo de recursos de las implementaciones de FSM generadas con técnicas basadas en implementación multi-nivel (dispositivo Virtex) y de las implementaciones de FSMIM (consumo estimado a partir de un dispositivo Spartan-3).

| FSM | Multi-Level | | FSMIM-S | | FSMIM-T | |
|---|---|---|---|---|---|---|
| | Bloques | LUT | Bloques | LUT | Bloques | LUT |
| bbsse | 1 | 31 | 1 | 8 | 1 | 3 |
| cse | 1 | 41 | 2 | 10 | 2 | 3 |
| ex1 | 3 | 87 | 3 | 39 | 4 | 6 |
| ex4 | 1 | 6 | 1 | 5 | 1 | 2 |
| ex6 | 1 | 17 | 1 | 4 | 1 | 2 |
| keyb | 2 | 61 | 2 | 25 | 3 | 2 |
| mark1 | 2 | 10 | 2 | 5 | 2 | 1 |
| mc | 1 | 2 | 1 | 2 | 1 | 1 |
| opus | 1 | 13 | 1 | 9 | 1 | 1 |
| planet | 7 | 65 | 2 | 55 | 2 | 7 |
| pma | 1 | 29 | 2 | 35 | 3 | 6 |
| s1488 | 6 | 102 | 4 | 53 | 4 | 5 |
| s1494 | 6 | 87 | 4 | 53 | 4 | 5 |
| s208 | 1 | 16 | 1 | 9 | 2 | 3 |
| s386 | 1 | 27 | 1 | 9 | 1 | 3 |
| s420 | 2 | 15 | 1 | 11 | 2 | 3 |
| s510 | 1 | 27 | 1 | 25 | 2 | 11 |
| s820 | 3 | 97 | 6 | 51 | 9 | 9 |
| s832 | 3 | 85 | 12 | 49 | 16 | 9 |
| sand | 2 | 77 | 2 | 49 | 3 | 8 |
| scf | 6 | 86 | 16 | 139 | 19 | 17 |
| sse | 1 | 31 | 1 | 9 | 1 | 3 |
| styr | 1 | 70 | 4 | 34 | 6 | 5 |
| Media | 2,3 | 47 | 3,1 | 29,9 | 3,9 | 5 |

**Tabla 6.4:** Comparativa de los resultados obtenidos por las implementaciones de FSMIM y las implementaciones multi-nivel ( %). Se muestran los valores medios de los incrementos y reducciones RPC calculados respecto a las implementaciones multi-nivel.

| | FSMIM-S | FSMIM-T |
|---|---|---|
| Red. Bloques | −7,6 | −20,4 |
| Red. LUT | 38,2 | 85,2 |
| Inc. NLAB | 52,6 | 73,0 |

XCV50 de la familia Virtex de Xilinx. Por tanto, para comparar esta técnica con las arquitecturas FSMIM se pueden utilizar los resultados de las implementaciones de FSMIM obtenidos con el dispositivo Spartan-3 (véase la sección 6.2.1).

De las diez FSM seleccionadas en [Barkalov et al., 2012], la mitad no cumplían los requisitos para generar una FSMIM y han sido excluidas de la comparativa. La tabla 6.5 muestra el consumo de bloques y de LUT para las cinco FSM restantes. La columna "DEPM" contiene los resultados de la técnica basada en descomposición estructural con partición de las micro-operaciones. El resto de columnas contienen los resultados obtenidos por las FSMIM con ajuste de profundidad, que han sido generadas con la estrategia de optimización A-PLE.

En promedio, comparada con la arquitectura FSMIM-T, el consumo de bloques de la técnica DEPM es ligeramente menor a expensas de un incremento considerable de LUT. Sin embargo, la arquitectura FSMIM-S mejora los resultados de la técnica DEPM tanto en consumo de bloques como de LUT. Respecto al NLAB, tal como muestra la tabla 6.6, ambas arquitecturas FSMIM consiguen ahorrar más LUT por bloque que la técnica DEPM. Por tanto, suponiendo que el número de casos estudiados sea representativo, se puede concluir que la técnica DEPM solamente es una alternativa a las implementaciones FSMIM en aquellos casos en los que una implementación FSMIM no es posible.

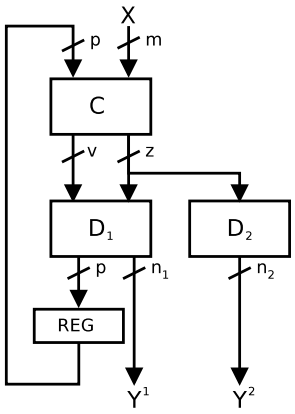



**Figura 6.6:** Descomposición estructural con partición de microoperaciones de una FSM.

**Tabla 6.5:** Consumo de recursos de las implementaciones de FSM generadas con la técnica basada en descomposición estructural con partición de microoperaciones (dispositivo Virtex) y de las implementaciones de FSMIM (consumo estimado a partir de un dispositivo Spartan-3).

| FSM | DEPM | | FSMIM-S | | FSMIM-T | |
|---|---|---|---|---|---|---|
| | Bloques | LUT | Bloques | LUT | Bloques | LUT |
| cse | 1 | 41 | 2 | 10 | 2 | 3 |
| ex1 | 3 | 87 | 3 | 39 | 4 | 6 |
| keyb | 2 | 60 | 2 | 25 | 3 | 2 |
| planet | 4 | 65 | 2 | 55 | 2 | 7 |
| s1494 | 4 | 87 | 4 | 53 | 4 | 5 |
| Media | 2,8 | 68 | 2,6 | 36,4 | 3 | 4,6 |

**Tabla 6.6:** NLAB obtenido por las implementaciones de FSM generadas con la técnica basada en descomposición estructural con partición de microoperaciones y por las implementaciones de FSMIM.

| FSM | DEPM | FSMIM-S | FSMIM-T |
|---|---|---|---|
| cse | 37 | 34 | 37,5 |
| ex1 | 6 | 22 | 24,8 |
| keyb | 15 | 32,5 | 29,3 |
| planet | 45,8 | 96,5 | 120,5 |
| s1494 | 29,5 | 38 | 50 |
| Media | 26,6 | 44,6 | 52,4 |

## 6.3. Técnicas basadas en codificación de transiciones

Las técnicas basadas en codificación de transiciones asignan un código a cada transición de la FSM, que es utilizado para direccionar la memoria en lugar de las entradas de la FSM. Esto permite reducir el tamaño de la memoria en los casos en los que el número de bits de codificación de las transiciones es menor que el número de entradas de la FSM. El código de la transición es generado por un circuito combinacional a partir del estado presente y de las entradas. Dentro de esta clasificación se encuentran las técnicas propuestas en [Cooke et al., 2015] y en [Sklyarov, 2000].

La primera de las técnicas no será tenida en cuenta en este estudio debido a que se emplea en el contexto de las FSM reconfigurables. Los códigos de transición se utilizan para diferenciar las transiciones de un mismo estado; por lo tanto, las transiciones de la FSM se identifican por el código de estado presente y el código de transición. En este caso, la dirección de la memoria

que contiene las transiciones de la FSM se forma concatenando el código del estado presente y el código de transición. La arquitectura propuesta requiere tres memorias asíncronas, implementadas como RAM distribuida (es decir, empleando LUT en lugar de bloques empotrados); por tanto, no es de aplicación en el marco de esta tesis.

La técnica propuesta en [Sklyarov, 2000] busca conjuntos de transiciones iguales (transiciones con el mismo estado siguiente y la misma salida) y asigna un código único a cada conjunto (este código será denominado *código de transición*). Para obtener el código transición se emplea una codificación de estados, denominada *codificación difusa de estados*[6], que asigna a los estados códigos de longitud variable (denominados *códigos difusos*). En los códigos difusos que no alcanzan la longitud máxima, algunos bits están indefinidos, dependiendo el valor de estos bits del estado presente y de las entradas de la FSM. El código de cada transición se obtiene a partir del código difuso del estado presente, asignando valores a los bits indefinidos de dicho código.

Por tanto, puesto que los códigos de transición deben diferenciar, por una parte, a las transiciones de la FSM y, por otra, a los estados, el número de bits de codificación necesarios es

$$z \geq \text{máx}\{\lceil \log_2 |E| \rceil, \lceil \log_2 d \rceil\}, \tag{6.1}$$

donde $E$ es el conjunto de estados y $d$ el número de transiciones diferentes de la FSM. Este valor determina también el tamaño máximo de los códigos difusos de estado. La elección de una codificación difusa de los estados adecuada puede reducir el valor de $z$; sin embargo, se trata de un problema complejo para el que no se ha encontrado una algoritmo que encuentre la solución óptima en tiempo polinomial (el autor propone en [Sklyarov, 2002a] un algoritmo evolutivo).

La figura 6.7 muestra la arquitectura usada por esta técnica, que será denominada *FSM con codificación difusa* (FCD). El codificador de transiciones es un circuito combinacional que genera el código de transición a partir del código difuso del estado presente y de las entradas de la FSM. El tamaño requerido de la ROM es $2^z(z + n)$, donde $n$ es el número de salidas de la FSM.

Desafortunadamente, en la bibliografía consultada no se han encontrado resultados experimentales obtenidos con una batería de pruebas estándar, por lo que no se ha podido comparar experimentalmente esta técnica con las arquitecturas FSMIM. Por lo tanto, a continuación se realizará un análisis cualitativo.

[6]Este termino, empleado por el autor de la técnica, no está relacionado con las *máquinas de estados difusos*.

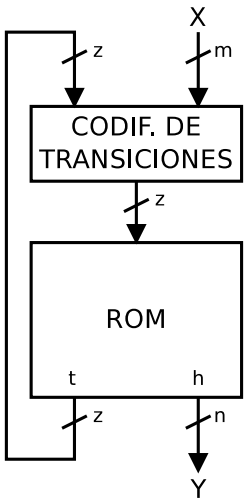



**Figura 6.7:** Arquitectura de una FSM con codificación difusa.

Respecto al tamaño de la ROM, la codificación de transiciones puede requerir en algunos casos una ROM con menor profundidad que la de la FSMIM correspondiente, aunque no se puede estimar la frecuencia de casos en los que esto podría ocurrir. Sin embargo, el ancho de la ROM de las FSMIM-S es menor o igual que el de la ROM de las FCD, ya que $z \geq \lceil \log_2 |E| \rceil$ (véase la ecuación 6.1 y la ecuación 2.21, en la que $p = \lceil \log_2 |E| \rceil$). En el caso de las FSMIM-T, el ancho de la ROM depende de características como el número de entradas de cada selector de entradas, por lo que el ancho requerido por cada arquitectura no es fácilmente comparable. Debido a que, a diferencia de las FSMIM-S, el ancho de las FSMIM-T puede llegar a ser mayor que el de las FCD, el consumo de bloques de las FSMIM-T puede incrementarse respecto a las FCD en un porcentaje de casos mayor que para las FSMIM-S.

Respecto al consumo de LUT, el codificador de transiciones implementa $z$ funciones booleanas, mientras que el banco de selectores de las FSMIM implementa $r$ funciones (una por cada selector de entradas), donde $r$ es el número de entradas efectivas de la FSM. La relación entre los valores de $z$ y $r$ resulta difícil de determinar, no siendo ninguno de los parámetros claramente menor que el otro. Sin embargo, en términos generales, la complejidad de las funciones booleanas implementadas por el codificador de transiciones, medida como el número de variables booleanas de la función, es mayor que las del banco de selectores de entradas. Cada función del codificador de transiciones depende de $z + m$ entradas, donde $m$ es el número total de

entradas de la FSM. Por el contrario, cada selector de entradas del banco de selectores de la FSMIM depende, por una parte, de un subconjunto muy reducido de las entradas de la FSM y, por otra, de los bits de codificación de estado (en el caso de las FSMIM-S) o de los bits de selección de entradas (en el caso de las FSMIM-T). El número de bits de codificación de estados es menor o igual que $z$ (véase la ecuación 6.1), mientras que el número de bits de selección de entradas de cada selector es significativamente menor que el número de bits de codificación de estados y, por tanto, que $z$. En promedio, en las FSM analizadas en el estudio experimental del capítulo 5, el número de bits de selección del canal con mayor número de entradas es menor que el 9 % del número de bits de codificación de estados.

Finalmente, respecto a la velocidad, el menor consumo de LUT de las FSMIM favorece el incremento de la frecuencia máxima de operación, por lo que es de esperar que consigan alcanzar mayor velocidad que las FCD.

Capítulo 7

# Conclusiones y trabajos futuros

## 7.1. Conclusiones

Esta tesis doctoral supone una contribución a la implementación de *Máquinas de Estados Finitos* (FSM, sigla del inglés Finite State Machine), en particular a la implementación de FSM utilizando los bloques de memoria empotrados disponibles en los dispositivos FPGA. El estudio realizado tiene como punto de partida la mejora de las prestaciones de las implementaciones de FSM basadas en el modelo de *Máquina de Estados Finitos con Multiplexión de Entradas* (FSMIM, sigla del inglés Finite State Machine with Input Multiplexing), que permite aprovechar las indeterminaciones en las entradas de una FSM para reducir el tamaño de la memoria requerida. El objetivo principal de este modelo es conseguir reducciones importantes del consumo de bloques mediante el empleo de un número reducido de *Tablas de Búsqueda* (LUT, sigla del inglés Look-Up Table)[1]. Este objetivo establece un problema de optimización que se divide en dos subproblemas, denominados *minimización de la selección de entradas* y *agrupación de estados*. Mientras que el objetivo principal de la minimización de la selección de entradas es reducir el consumo de LUT, el objetivo principal de la agrupación de estados es reducir el consumo de bloques de memoria empotrados.

El modelo de FSMIM incluye, por una parte, una arquitectura denominada *FSMIM con selección de entradas basada en transiciones* (FSMIM-T, sigla del inglés **FSMIM** with **T**ransition-based input selection) y, por otra, un procedimiento para generar las implementaciones FSMIM-T. Este pro-

[1]Las LUT son los elementos configurables básicos para la implementación de lógica combinacional disponibles en los dispositivos FPGA.

cedimiento, que será denominado estrategia de optimización de referencia, se divide en dos fases: una fase inicial de minimización de la selección de entradas seguida de una fase final de agrupación de estados.

En este escenario, las aportaciones principales de esta tesis doctoral son las siguientes:

1. Se ha propuesto una nueva arquitectura de FSMIM, denominada *FSMIM con selección de entradas basada en estados* (FSMIM-S, sigla del inglés **FSMIM** with **S**tate-based input selection). El objetivo de esta arquitectura es permitir implementaciones de FSMIM con menor consumo de bloques que la arquitectura FSMIM-T. El estudio experimental realizado muestra que la reducción del número de bloques se consigue a expensas de un incremento en el consumo de LUT, lo que indica que la arquitectura FSMIM-S no sustituye a la arquitectura FSMIM-T, sino que la complementa ampliando el abanico de opciones de diseño disponibles para la implementación de máquinas de estados y permitiendo un ajuste más fino de los principales parámetros de diseño: consumo de LUT, consumo de bloques de memoria y frecuencia de operación máxima.

2. Se han definido nuevas estrategias de optimización de FSMIM compuestas de dos o más fases. Estas se pueden clasificar en dos categorías: (a) estrategias de tipo MA, que comienzan con una fase de minimización de selección de entradas y dan prioridad a la reducción del consumo de celdas lógicas de las implementaciones de FSMIM; y (b) estrategias de tipo AM que comienzan con una fase de agrupación de estados y dan prioridad a la reducción del consumo de bloques de memoria.

3. Se han propuesto nuevos algoritmos para el subproblema de la agrupación de estados. Para las estrategias de tipo AM se ha propuesto un algoritmo que consigue agrupar el máximo número de estados, y para las de tipo MA, en las que no es posible garantizar la agrupación máxima, se ha propuesto un algoritmo que mejora los resultados de la estrategia de optimización de referencia.

4. Se han realizado las siguientes contribuciones relacionadas con el subproblema de la minimización de la selección de entradas:

   *a*) Se ha propuesto un nuevo problema *NP*-completo, denominado Problema del *k*-Emparejamiento Parcial Maximal Mínimo, que modela el subproblema de la minimización de la selección

de entradas utilizando un grafo bipartito. Se ha demostrado que dicho problema es *NP*-completo mediante *reducción* al problema Hitting Set, lo que ha permitido demostrar que la optimización de FSMIM involucra problemas NP.

*b*) Se han propuesto las siguientes variantes multiobjetivo del Problema del *k*-Emparejamiento Parcial Maximal Mínimo, que permiten mejorar las prestaciones de las FSMIM obtenidas por las estrategias de optimización:

- Problema del *k*-Emparejamiento Parcial Maximal Mínimo BC
- Problema del *k*-Emparejamiento Parcial Maximal Mínimo CBC

Tanto para el Problema del *k*-Emparejamiento Parcial Maximal Mínimo como para sus variantes, se han propuesto extensiones que permiten la búsqueda de soluciones en fases de minimización de la selección de entradas posteriores a una fase de agrupación de estados.

*c*) Se ha formulado un modelo de *Programación Lineal Entera* (PLE) para cada una de los problemas propuestos. Esto permite obtener soluciones al subproblema de minimización de la selección de entradas aprovechando toda la potencia de las herramientas existentes para la optimización de problemas de PLE. Por otra parte, se ha propuesto un algoritmo voraz para el Problema del *k*-Emparejamiento Parcial Maximal Mínimo y para la extensión correspondiente. Estos algoritmos son una alternativa a los modelos de PLE, presentando menor coste computacional a expensas de obtener peores soluciones. Ambas alternativas permitirán implementar en una futura herramienta diferentes niveles de esfuerzo de optimización, que ofrecerán la posibilidad de elegir entre mejores soluciones o menor tiempo de CPU.

5. Se han propuesto mecanismos para ajustar la profundidad de la memoria requerida por las implementaciones de FSMIM a un valor múltiplo de la profundidad de los bloques de memoria empotrados. Los resultados experimentales han mostrado que este mecanismo permite reducir, por una parte, el consumo de LUT de las implementaciones de ambas arquitecturas FSMIM y, por otra, el consumo de bloques de las implementaciones de FSMIM-T.

6. Se ha estudiado la efectividad de las estrategias de optimización propuestas en la mejora de las prestaciones de las implementaciones de FSMIM. Los resultados experimentales han mostrado que las estrategias propuestas permiten obtener mejoras significativas en el consumo de bloques de las implementaciones de FSMIM respecto a la estrategia de referencia. Mientras que en la arquitectura FSMIM-S la efectividad en el consumo de LUT y en la velocidad ha sido menor, en la arquitectura FSMIM-T las mejoras conseguidas en estos parámetros han sido importantes. Se puede concluir que las estrategias propuestas sustituyen eficazmente a la estrategia de referencia y ofrecen al diseñador diferentes alternativas para alcanzar los objetivos de diseño respecto a las restricciones de área y velocidad.

7. Se han comparado experimentalmente las prestaciones de las implementaciones de FSMIM y de las correspondientes implementaciones convencionales de FSM. Los resultados permiten extraer las siguientes conclusiones:

   *a*) La posibilidad de aprovechar eficientemente los diferentes tipos de recursos disponibles en las FPGA permite implementaciones en dispositivos más pequeños (respecto al número de recursos), lo que favorece la reducción de costes. En este sentido, las dos arquitecturas FSMIM junto con las diferentes estrategias de optimización ofrecen un amplio conjunto de alternativas para alcanzar los objetivos de diseño.

   *b*) En términos generales, las implementaciones de FSMIM hacen un uso muy eficiente de los recursos, consiguiendo reducciones importantes tanto en el número de LUT respecto a las implementaciones convencionales de FSM basadas en celdas lógicas como en el número de bloques respecto a las implementaciones convencionales de FSM basadas en memoria. Además, consiguen ahorrar más LUT por cada bloque usado que las implementaciones convencionales basadas en memoria.

   *c*) En los diseños en los que la velocidad no es crítica (los optimizados en área), las implementaciones de FSMIM son las que alcanzan las mayores frecuencias de operación; por tanto, presentan el mejor equilibrio entre el consumo de recursos y velocidad.

   *d*) En los diseños optimizados en velocidad, las implementaciones de FSMIM son la mejor opción para la implementación de FSM grandes.

8. Se han comparado las FSMIM con otras técnicas propuestas en la literatura para la implementación basada en memoria de FSM. A partir de los escasos resultados experimentales aportados por los autores, se ha realizado un estudio comparativo que muestra, por una parte, que las FSMIM mejoran el consumo de bloques y de LUT respecto a algunas de las técnicas comparadas; y, por otra, que se complementan con el resto de técnicas, ampliando el conjunto de soluciones disponibles para el diseñador.

Para finalizar las conclusiones, cabe mencionar que el trabajo de investigación desarrollado en el marco de esta tesis doctoral ha dado lugar a diversas publicaciones en revistas de relevancia científica que pueden consultarse en el apéndice A.

## 7.2. Trabajos futuros

Durante el transcurso del trabajo realizado en esta tesis doctoral han surgido nuevas ideas que finalmente no han sido abordadas y que pueden ser objeto de futuras líneas de investigación. A continuación se detallan las que se han considerado más importantes:

1. *Estudiar la influencia de la codificación de estados en la complejidad del banco de selectores de entradas de la arquitectura FSMIM-S.* Una codificación de estados adecuada puede reducir el número de LUT necesarias para implementar el banco de selectores de entradas de las implementaciones FSMIM-S sin afectar al tamaño de la memoria requerida. Este estudio tiene dos vertientes. Por una parte, el estudio teórico de las características de las funciones lógicas implementadas por el banco de selectores de entradas permitiría proponer algoritmos específicos para la asignación de códigos a los estados. Por otra parte, sería necesario un estudio comparativo de los resultados obtenidos por dichos algoritmos y los obtenidos por otras herramientas de minimización de funciones lógicas.

2. *Estudiar la influencia de la codificación de grupos en la complejidad del codificador de grupos de la arquitectura FSMIM-S.* El número de LUT necesarios para implementar el codificador de grupos depende tanto de la codificación de estados de la FSM como de la codificación de los grupos. Por tanto, una asignación de códigos adecuada a los grupos puede reducir el consumo de LUT de las implementaciones de

FSMIM-S. Por otra parte, el grado de simplificación del codificador de grupos que se obtiene con la codificación puede depender de parámetros como el número de estados de cada grupo y el número de entradas asociadas a cada estado. Un estudio de la influencia de estos parámetros permitiría incorporar en los algoritmos de agrupación de estados diferentes criterios para decidir qué estados deben agruparse.

3. *Definir nuevas variantes del* Problema del *k*-Emparejamiento Parcial Maximal Mínimo *que incluyan una estimación del consumo de LUT del banco de multiplexores de la arquitectura FSMIM-T.* Puesto que el consumo de LUT de un multiplexor depende fuertemente del número de entradas, para cada familia de dispositivos FPGA se puede calcular experimentalmente la función que relaciona el número de entradas con el consumo. Esta función puede incorporarse a los objetivos de los modelos de PLE propuestos en esta tesis doctoral, lo que permitiría mejorar el consumo de LUT de las soluciones obtenidas.

4. *Definir nuevas variantes del* Problema del *k*-Emparejamiento Parcial Maximal Mínimo *que mejoren la velocidad de las implementaciones de FSMIM-T.* El retraso impuesto por el banco de multiplexores de entradas de la arquitectura FSMIM-T depende del multiplexor con mayor número de entradas. Por lo tanto, la definición de nuevos objetivos de optimización que minimicen el número de entradas máximo del banco de multiplexores podría aumentar velocidad de las implementaciones de FSMIM-T. Esto daría lugar a un nuevo tipo de estrategias de optimización de FSMIM-T, adecuadas para los diseños optimizados en velocidad. Adicionalmente, sería conveniente el estudio de la influencia de estas estrategias también en la velocidad de las implementaciones de FSMIM-S.

5. *Buscar otras aplicaciones al* Problema del *k*-Emparejamiento Parcial Maximal Mínimo*.* La aplicación de este problema de optimización en contextos diferentes al de las FSMIM permitiría aprovechar los algoritmos propuestos para encontrar soluciones a otros problemas.

6. *Proponer algoritmos voraces para las variantes del* Problema del *k*--Emparejamiento Parcial Maximal Mínimo*.* Este estudio permitiría mejorar los resultados de las estrategias de optimización basadas en algoritmos voraces.

7. *Estudiar y proponer mejoras a los modelos de PLE presentados.* El objetivo de estas mejoras es reducir el tiempo de búsqueda de soluciones, lo que permitiría, además, mejorar las soluciones obtenidas en los casos en los que el tiempo límite establecido por el usuario se agote antes de encontrar la solución óptima.

8. *Publicar la herramienta FSMIM-Gen 2.0.* Aunque dicha herramienta es completamente funcional desde el punto de vista de la generación de FSMIM, es necesario prepararla para su distribución, ya que se trata de un prototipo desarrollado para explorar las diferentes ideas que han surgido durante este trabajo de investigación y para realizar el estudio experimental presentado en esta tesis. En primer lugar, debe realizarse un análisis de rendimiento y la posterior optimización del código; y en segundo lugar, debe prepararse la herramienta para facilitar su uso por parte de los usuarios.

Apéndice A

# Publicaciones del autor sobre el trabajo de investigación presentado en esta tesis doctoral

**Publicaciones en revistas**

- García-Vargas, I. y Senhadji-Navarro, R. (2015). Finite State Machines with Input Multiplexing: A performance study. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 34, n.º 5, páginas 867–871.

  **Factor de impacto (JCR[1] 2014)[2]: 1,003 (Q2)**

- García-Vargas, I. y Senhadji-Navarro, R. (2012). The Minimum Maximal k-Partial-Matching problem. *Optimization Letters,* diciembre 2013, vol. 7, n.º 8, páginas 1959–1968. Publicado *online* en 2012.

  **Factor de impacto (JCR 2012): 1,654 (Q1)**

- Senhadji-Navarro, R., García-Vargas, I., Jimenez-Moreno, G., y Civit-Ballcels, A. (2004). ROM-based FSM implementation using input multiplexing in FPGA devices. *Electronics Letters*, vol. 40, n.º 20, páginas 1249–1251.

  **Factor de impacto (JCR 2004): 0,968 (Q2)**

[1] Journal Citation Report.

[2] En el momento de la publicación de esta tesis no está disponible todavía el JCR del año 2015.

**Publicaciones en congresos**

- García-Vargas, I., Senhadji-Navarro, R., Jimenez-Moreno, G., Civit-Balcells, A., y Guerra-Gutierrez, P. (2007). ROM-based Finite State Machine Implementation in Low Cost FPGAs. En *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, páginas 2342–2347.

- García-Vargas, I., Senhadji-Navarro, R., Jiménez-Moreno, G., y Civit-Ballcels, A. (2005). Máquinas de Estados Finitos con Multiplexión de Entradas: Implementación sobre bloques RAM en FPGA. En *V Jornadas sobre Computación Reconfigurable y Aplicaciones (JCRA'2005)*, páginas 369–374.

Apéndice B

# Batería de pruebas MCNC

Esta batería de pruebas, creada para realizar síntesis lógica y optimización (*logic synthesis and optimization benchmarks*), es distribuida por el Centro de Microelectrónica de Carolina de Norte e incluye los conjuntos de pruebas ISCAS'85 e ISCAS'89. El primer informe de gran difusión (versión 2.0) data de 1988 [Lisanke, 1988; Yang, 1991] y ha sido ampliado en dos ocasiones. Para los estudios experimentales de esta tesis se ha utilizado la versión 4.0 [McElvain, 1993], que forma parte del conjunto de baterías de pruebas ACM/SIGDA [ACM/SIGDA, 2012].

Está dividida en cuatro categorías fundamentales: Máquinas de estado en formato KISS, lógica secuencial multinivel en formato BLIF extendido (*Berkeley Logic Interchange Format*) o SLIF (*Structure Logic Interchange Format*), lógica combinacional multinivel en BLIF extendido o SLIF, y lógica de dos niveles en formato PLA. Cada FSM dispone de dos archivos que contienen la descripción en formato *KISS2* de la FSM (sección B.1) y los patrones de prueba.

Los patrones de prueba de MCNC han permitido validar las descripciones VHDL de las arquitecturas de FSM implementadas y, como consecuencia, el funcionamiento de las diferentes herramientas desarrolladas para realizar el estudio experimental. Para cada fichero de patrones de prueba, se ha generado una descripción VHDL para simulación que permite comprobar las diferentes arquitecturas de la FSM correspondiente. La simulación de dicha descripción se ha realizado con el simulador HDL integrado en *ISE WebPACK*: ISim [Xilinx, 2012].

De las 46 FSM disponibles en MCNC, se han seleccionado 26 para las pruebas experimentales (véase el criterio de selección utilizado en la sec-

ción 5.2.3). La tabla B.1 muestra las características de todas las FSM seleccionadas. Las columnas "N.º ents.", "N.º ests." y "N.º sals." contienen el número de entradas, de salidas y de estados de la FSM, respectivamente. La columna "Ratio inds. *Matriz de Selección de Entradas* (MSE)" contiene el ratio de indeterminaciones de la MSE. La columna "Tam. inic." contiene el tamaño inicial de la FSMIM-S (medido en Kbits). Finalmente, las columnas "N.º cans. sel.", "Prof. inic." y "Red. inic." contienen el número de canales de selección, la profundidad inicial (medida en Kpalabras) y la reducción inicial de profundidad de la FSMIM, respectivamente. Estos parámetros se describen en la sección 5.2.3.

**Tabla B.1:** Características de las FSM de la batería de pruebas MCNC

| Nombre | N.º ents. | N.º ests. | N.º sals. | N.º cans. sel. | Ratio inds. MSE | Prof. inic. (Kpal.) | Tam. inic. (Kbits) | Red. inic. (%) |
|---|---|---|---|---|---|---|---|---|
| bbara_bbtas | 4 | 128 | 2 | 4 | 0,3 | 2 | 18 | 0 |
| bbsse | 7 | 16 | 7 | 5 | 0,5 | 0,5 | 5,5 | 75 |
| cse | 7 | 16 | 7 | 6 | 0,3 | 1 | 11 | 50 |
| ex1 | 9 | 20 | 19 | 6 | 0,4 | 1,2 | 30 | 87,5 |
| ex4 | 6 | 14 | 9 | 3 | 0,5 | 0,1 | 1,4 | 87,5 |
| ex6 | 5 | 8 | 8 | 3 | 0,1 | 0,1 | 0,7 | 75 |
| keyb | 7 | 19 | 2 | 7 | 0,5 | 2,4 | 16,6 | 0 |
| mark1 | 5 | 15 | 16 | 4 | 0,7 | 0,2 | 4,7 | 50 |
| mc | 3 | 4 | 5 | 2 | 0,2 | 0 | 0,1 | 50 |
| opus | 5 | 10 | 6 | 5 | 0,6 | 0,3 | 3,1 | 0 |
| planet | 7 | 48 | 19 | 5 | 0,8 | 1,5 | 37,5 | 75 |
| pma | 8 | 24 | 8 | 6 | 0,4 | 1,5 | 19,5 | 75 |
| s1 | 8 | 20 | 6 | 8 | 0,6 | 5 | 55 | 0 |
| s1488 | 8 | 48 | 19 | 6 | 0,6 | 3 | 75 | 75 |
| s1494 | 8 | 48 | 19 | 6 | 0,6 | 3 | 75 | 75 |
| s208 | 8 | 18 | 2 | 4 | 0 | 0,3 | 2 | 93,8 |
| s27 | 4 | 6 | 1 | 4 | 0,3 | 0,1 | 0,4 | 0 |
| s386 | 7 | 13 | 7 | 5 | 0,4 | 0,4 | 4,5 | 75 |
| s420 | 8 | 18 | 2 | 4 | 0 | 0,3 | 2 | 93,8 |
| s510 | 19 | 47 | 7 | 2 | 0,7 | 0,2 | 2,4 | 100 |
| s820 | 18 | 25 | 19 | 8 | 0,5 | 6,2 | 150 | 99,9 |
| s832 | 18 | 25 | 19 | 8 | 0,5 | 6,2 | 150 | 99,9 |
| sand | 11 | 32 | 9 | 7 | 0,7 | 4 | 56 | 93,8 |
| scf | 27 | 121 | 56 | 9 | 0,9 | 60,5 | 3811,5 | 100 |
| sse | 7 | 16 | 7 | 5 | 0,5 | 0,5 | 5,5 | 75 |
| styr | 9 | 30 | 10 | 7 | 0,5 | 3,8 | 56,2 | 75 |

## B.1. El formato *KISS2*

El formato *KISS* tiene su origen en el programa de minimización de funciones lógicas de dos niveles *espresso* [Rudell y Sangiovanni-Vincentelli, 1987], usado para los dispositivos PLA[1]. El formato evolucionó para dar soporte a las máquinas de estados finitos (llamado *KISS2*) y se ha extendido su utilización en múltiples herramientas para la asignación y minimización de estados.

El formato *KISS2* contiene la descripción de la FSM en un formato similar a las *Tabla de Transición de Estados* (TTE). Cada línea contiene una transición de la FSM definida por el estado presente y el valor de las entradas que la activan, así como el estado siguiente y el valor de las salidas de la transición. Los valores de las entradas y salidas se representan como un vector de bits en el que cada bit se corresponde con una entrada o salida de la FSM. Las indeterminaciones en los valores de las entradas o salidas se indican con el símbolo '-' que representa a los valores 0 y 1 indistintamente.

Los archivos *KISS2* son archivos de texto con la estructura mostrada en la figura B.1, donde el texto encerrado entre corchetes es opcional y el texto delimitado por los símbolos '`<>`' son campos con el siguiente significado:

**num-entradas** Es el número de entradas de la FSM.

**num-salidas** Es el número de salidas de la FSM.

**num-terminos** Es el número de 4-tuplas que describen las transiciones de la FSM:

`<entrada> <estado-presente> <estado-siguiente> <salida>`

**num-estados** Es el número de estados diferentes que aparecen en las columnas `<estado-presente>` y `<estado-siguiente>`.

**estado-reset** Es el nombre simbólico del estado de reset de la FSM. Debe ser alguno de los nombres que aparecen en la columna `<estado-presente>`.

**entrada** Es una secuencia de valores $\{0, 1, \text{-}\}$ de longitud `num-entradas`.

**estado-presente** Es el nombre simbólico del estado presente de cada transición. El símbolo '`*`' sustituye a cualquier estado de la FSM.

[1]Un dispositivo PLA (sigla del inglés Programmable Logic Array) es un dispositivo programable, anterior a las FPGA y CPLD, utilizado para implementar lógica combinacional.

**estado-siguiente** Es el nombre simbólico del estado siguiente de cada transición.

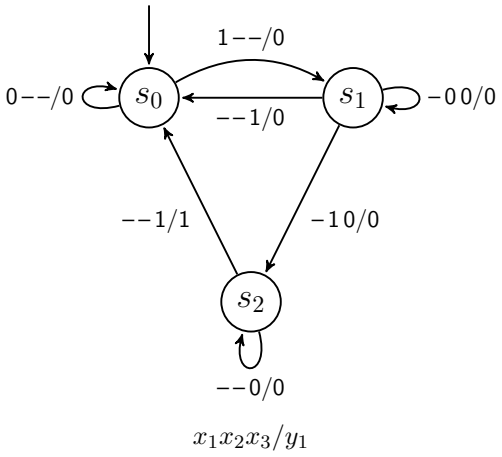
**salida** Es una secuencia de valores $\{0, 1, \text{-}\}$ de longitud `num-salidas`.

El carácter '`#`' marca el inicio de un comentario que se extiende hasta el final de la línea.

La figura B.2 muestra un ejemplo de la descripción en formato *KISS2* (figura B.2a) de una FSM cuyo *Diagrama de Transición de Estados* (DTE) se muestra en la figura B.2b.

```
# Estructura general de un fichero KISS2
.i <num-entradas>
.o <num-salidas>
[.p <num-terminos>]
[.s <num-estados>]
[.r <estado-reset>]
<entrada> <estado-presente> <estado-siguiente> <salida>
⋮
<entrada> <estado-presente> <estado-siguiente> <salida>
[.e]
```

**Figura B.1:** Formato *KISS2*




**(a)**

```
# Ejemplo de fichero KISS2
.i 3
.o 1
.p 7
.s 2
.r s0
0-- s0 s0 0
1-- s0 s1 0
-00 s1 s1 0
--1 s1 s0 0
-10 s1 s2 0
--0 s2 s2 0
--1 s2 s0 1
.e
```

**(b)**

**Figura B.2:** Ejemplo de la descripción en formato *KISS2* de una FSM: (a) DTE, (b) Archivo *KISS2*

Apéndice C

# Batería de pruebas BP-24

La batería de pruebas BP-24 consta de 108 FSM aleatorias generadas con la herramienta *FSM Benchmark Generator* [Józwiak et al., 2004] y cedidas por el profesor Lech Józwiack del Departamento de Información y Sistemas de Comunicación de la Facultad de Ingeniería Eléctrica de la Universidad Tecnología de Eindhoven (Paises Bajos). La tabla C.1 muestra las características de las FSM. Las columnas "N.º ents.", "N.º ests." y "N.º sals." contienen el número de entradas, de salidas y de estados de la FSM, respectivamente. La columna "Ratio inds. MSE" contiene el ratio de indeterminaciones de la MSE. La columna "Tam. inic." contiene el tamaño inicial de la FSMIM-S (medido en Kbits). Finalmente, las columnas "N.º cans. sel.", "Prof. inic." y "Red. inic." contienen el número de canales de selección, la profundidad inicial (medida en Kpalabras) y la reducción inicial de profundidad de la FSMIM, respectivamente. Estos parámetros se describen en la sección 5.2.3.

**Tabla C.1:** Características de las FSM de la batería de pruebas BP-24

| Nombre | N.º ents. | N.º ests. | N.º sals. | N.º cans. sel. | Ratio inds. MSE | Prof. inic. (Kpal.) | Tam. inic. (Kbits) | Red. inic. (%) |
|---|---|---|---|---|---|---|---|---|
| fsm_rd_001 | 2 | 20 | 8 | 2 | 0,6 | 0,1 | 1 | 0 |
| fsm_rd_002 | 4 | 10 | 4 | 2 | 0,3 | 0 | 0,3 | 75 |
| fsm_rd_003 | 3 | 8 | 4 | 2 | 0,4 | 0 | 0,2 | 50 |
| fsm_rd_004 | 3 | 7 | 4 | 2 | 0,6 | 0 | 0,2 | 50 |
| fsm_rd_007 | 6 | 5 | 12 | 4 | 0,2 | 0,1 | 1,2 | 75 |
| fsm_rd_008 | 8 | 7 | 2 | 3 | 0 | 0,1 | 0,3 | 96,9 |
| fsm_rd_009 | 8 | 7 | 2 | 3 | 0 | 0,1 | 0,3 | 96,9 |
| fsm_rd_010 | 8 | 14 | 2 | 4 | 0,4 | 0,2 | 1,3 | 93,8 |
| fsm_rd_011 | 8 | 6 | 4 | 4 | 0,4 | 0,1 | 0,7 | 93,8 |
| fsm_rd_012 | 5 | 7 | 8 | 4 | 0,8 | 0,1 | 1,2 | 50 |
| fsm_rd_013 | 8 | 8 | 8 | 4 | 0,2 | 0,1 | 1,4 | 93,8 |
| fsm_rd_014 | 8 | 6 | 4 | 8 | 0,4 | 1,5 | 10,5 | 0 |
| fsm_rd_016 | 7 | 7 | 12 | 4 | 0,3 | 0,1 | 1,6 | 87,5 |
| fsm_rd_017 | 8 | 13 | 12 | 4 | 0,4 | 0,2 | 3,2 | 93,8 |

Continúa en la página siguiente

**Tabla C.1:** Continúa desde la página anterior

| Nombre | N.º ents. | N.º ests. | N.º sals. | N.º cans. sel. | Ratio inds. MSE | Prof. inic. (Kpal.) | Tam. inic. (Kbits) | Red. inic. (%) |
|---|---|---|---|---|---|---|---|---|
| fsm_rd_018 | 8 | 6 | 2 | 4 | 0,3 | 0,1 | 0,5 | 93,8 |
| fsm_rd_019 | 8 | 6 | 2 | 4 | 0,3 | 0,1 | 0,5 | 93,8 |
| fsm_rd_020 | 8 | 8 | 4 | 8 | 0,4 | 2 | 14 | 0 |
| fsm_rd_021 | 8 | 8 | 4 | 8 | 0,4 | 2 | 14 | 0 |
| fsm_rd_022 | 4 | 13 | 12 | 3 | 0,6 | 0,1 | 1,6 | 50 |
| fsm_rd_023 | 3 | 10 | 4 | 3 | 0,6 | 0,1 | 0,6 | 0 |
| fsm_rd_025 | 3 | 6 | 2 | 3 | 0,1 | 0 | 0,2 | 0 |
| fsm_rd_026 | 3 | 20 | 5 | 3 | 0,3 | 0,2 | 1,6 | 0 |
| fsm_rd_027 | 5 | 22 | 3 | 5 | 0 | 0,7 | 5,5 | 0 |
| fsm_rd_029 | 5 | 26 | 3 | 5 | 0 | 0,8 | 6,5 | 0 |
| fsm_rd_030 | 5 | 28 | 3 | 5 | 0 | 0,9 | 7 | 0 |
| fsm_rd_031 | 5 | 31 | 3 | 5 | 0 | 1 | 7,8 | 0 |
| fsm_rd_032 | 5 | 32 | 3 | 5 | 0,1 | 1 | 8 | 0 |
| fsm_rd_033 | 4 | 32 | 3 | 4 | 0,1 | 0,5 | 4 | 0 |
| fsm_rd_034 | 5 | 30 | 3 | 5 | 0 | 0,9 | 7,5 | 0 |
| fsm_rd_035 | 3 | 12 | 6 | 3 | 0,2 | 0,1 | 0,9 | 0 |
| fsm_rd_037 | 11 | 62 | 5 | 11 | 0 | 124 | 1364 | 0 |
| fsm_rd_038 | 4 | 21 | 6 | 4 | 0,1 | 0,3 | 3,6 | 0 |
| fsm_rd_039 | 4 | 7 | 12 | 3 | 0,6 | 0,1 | 0,8 | 50 |
| fsm_rd_040 | 4 | 7 | 12 | 3 | 0,6 | 0,1 | 0,8 | 50 |
| fsm_rd_041 | 2 | 20 | 6 | 2 | 0,6 | 0,1 | 0,9 | 0 |
| fsm_rd_042 | 8 | 13 | 4 | 4 | 0,4 | 0,2 | 1,6 | 93,8 |
| fsm_rd_043 | 5 | 48 | 8 | 5 | 0,3 | 1,5 | 21 | 0 |
| fsm_rd_044 | 5 | 50 | 8 | 5 | 0,2 | 1,6 | 21,9 | 0 |
| fsm_rd_045 | 5 | 48 | 8 | 5 | 0,2 | 1,5 | 21 | 0 |
| fsm_rd_048 | 4 | 13 | 12 | 3 | 0,6 | 0,1 | 1,6 | 50 |
| fsm_rd_049 | 2 | 5 | 3 | 2 | 0,4 | 0 | 0,1 | 0 |
| fsm_rd_050 | 3 | 16 | 4 | 3 | 0,2 | 0,1 | 1 | 0 |
| fsm_rd_053 | 4 | 13 | 4 | 4 | 0,3 | 0,2 | 1,6 | 0 |
| fsm_rd_054 | 8 | 14 | 4 | 8 | 0,3 | 3,5 | 28 | 0 |
| fsm_rd_055 | 2 | 9 | 8 | 2 | 0,2 | 0 | 0,4 | 0 |
| fsm_rd_056 | 2 | 15 | 9 | 2 | 0,5 | 0,1 | 0,8 | 0 |
| fsm_rd_057 | 2 | 15 | 2 | 2 | 0,5 | 0,1 | 0,4 | 0 |
| fsm_rd_059 | 12 | 43 | 5 | 12 | 0 | 172 | 1892 | 0 |
| fsm_rd_060 | 8 | 57 | 8 | 8 | 0 | 14,2 | 199,5 | 0 |
| fsm_rd_061 | 2 | 33 | 2 | 2 | 0,5 | 0,1 | 1 | 0 |
| fsm_rd_062 | 8 | 41 | 8 | 8 | 0 | 10,2 | 143,5 | 0 |
| fsm_rd_063 | 8 | 37 | 8 | 8 | 0 | 9,2 | 129,5 | 0 |
| fsm_rd_064 | 2 | 14 | 5 | 2 | 0,6 | 0,1 | 0,5 | 0 |
| fsm_rd_065 | 2 | 14 | 5 | 2 | 0,6 | 0,1 | 0,5 | 0 |
| fsm_rd_066 | 2 | 14 | 2 | 2 | 0,6 | 0,1 | 0,3 | 0 |
| fsm_rd_067 | 2 | 25 | 2 | 2 | 0,4 | 0,1 | 0,7 | 0 |
| fsm_rd_068 | 12 | 45 | 5 | 2 | 0,3 | 0,2 | 1,9 | 99,9 |
| fsm_rd_069 | 12 | 46 | 5 | 3 | 0,7 | 0,4 | 4 | 99,8 |
| fsm_rd_070 | 2 | 26 | 2 | 2 | 0,5 | 0,1 | 0,7 | 0 |
| fsm_rd_071 | 4 | 48 | 16 | 3 | 0,7 | 0,4 | 8,2 | 50 |
| fsm_rd_072 | 2 | 7 | 4 | 2 | 0,6 | 0 | 0,2 | 0 |
| fsm_rd_073 | 4 | 6 | 2 | 4 | 0,5 | 0,1 | 0,5 | 0 |
| fsm_rd_074 | 12 | 37 | 8 | 11 | 0,4 | 74 | 1036 | 50 |
| fsm_rd_075 | 12 | 38 | 8 | 9 | 0,3 | 19 | 266 | 87,5 |
| fsm_rd_077 | 8 | 14 | 3 | 8 | 0,2 | 3,5 | 24,5 | 0 |
| fsm_rd_081 | 10 | 39 | 12 | 2 | 0,5 | 0,2 | 2,7 | 99,6 |
| fsm_rd_083 | 11 | 42 | 11 | 11 | 0,3 | 84 | 1428 | 0 |
| fsm_rd_084 | 2 | 7 | 2 | 2 | 0,6 | 0 | 0,1 | 0 |
| fsm_rd_085 | 8 | 39 | 12 | 2 | 0,7 | 0,2 | 2,7 | 98,4 |
| fsm_rd_086 | 2 | 5 | 3 | 1 | 0,4 | 0 | 0,1 | 50 |
| fsm_rd_087 | 9 | 32 | 9 | 9 | 0 | 16 | 224 | 0 |
| fsm_rd_091 | 4 | 60 | 4 | 4 | 0,5 | 0,9 | 9,4 | 0 |
| fsm_rd_092 | 1 | 8 | 2 | 1 | 0,9 | 0 | 0,1 | 0 |
| fsm_rd_094 | 1 | 7 | 3 | 1 | 0,9 | 0 | 0,1 | 0 |
| fsm_rd_095 | 2 | 7 | 3 | 1 | 0,7 | 0 | 0,1 | 50 |
| fsm_rd_097 | 16 | 27 | 4 | 8 | 0,5 | 6,8 | 60,8 | 99,6 |
| fsm_rd_098 | 5 | 36 | 11 | 5 | 0,3 | 1,1 | 19,1 | 0 |
| fsm_rd_099 | 5 | 42 | 11 | 5 | 0,2 | 1,3 | 22,3 | 0 |
| fsm_rd_100 | 15 | 38 | 12 | 8 | 0,7 | 9,5 | 171 | 99,2 |
| fsm_rd_103 | 3 | 8 | 2 | 1 | 0 | 0 | 0,1 | 75 |
| fsm_rd_104 | 2 | 6 | 2 | 1 | 0 | 0 | 0,1 | 50 |
| fsm_rd_105 | 6 | 50 | 8 | 6 | 0,3 | 3,1 | 43,8 | 0 |
| fsm_rd_106 | 15 | 54 | 12 | 7 | 0,6 | 6,8 | 121,5 | 99,6 |
| fsm_rd_107 | 8 | 32 | 6 | 5 | 0,2 | 1 | 11 | 87,5 |

Continúa en la página siguiente

**Tabla C.1:** Continúa desde la página anterior

| Nombre | N.º ents. | N.º ests. | N.º sals. | N.º cans. sel. | Ratio inds. MSE | Prof. inic. (Kpal.) | Tam. inic. (Kbits) | Red. inic. (%) |
|---|---|---|---|---|---|---|---|---|
| fsm_rd_108 | 4 | 60 | 4 | 4 | 0,1 | 0,9 | 9,4 | 0 |
| fsm_rd_109 | 5 | 52 | 3 | 5 | 0,1 | 1,6 | 14,6 | 0 |
| fsm_rd_110 | 6 | 53 | 3 | 4 | 0,3 | 0,8 | 7,5 | 75 |
| fsm_rd_111 | 7 | 34 | 5 | 4 | 0,6 | 0,5 | 5,8 | 87,5 |
| fsm_rd_112 | 7 | 35 | 5 | 4 | 0,4 | 0,5 | 6 | 87,5 |
| fsm_rd_113 | 8 | 46 | 3 | 6 | 0 | 2,9 | 25,9 | 75 |
| fsm_rd_114 | 2 | 8 | 10 | 2 | 0,3 | 0 | 0,4 | 0 |
| fsm_rd_115 | 2 | 8 | 8 | 2 | 0,6 | 0 | 0,3 | 0 |
| fsm_rd_121 | 2 | 9 | 8 | 2 | 0,2 | 0 | 0,4 | 0 |
| fsm_rd_122 | 11 | 26 | 7 | 11 | 0 | 52 | 624 | 0 |
| fsm_rd_124 | 3 | 5 | 6 | 2 | 0 | 0 | 0,2 | 50 |
| fsm_rd_130 | 2 | 6 | 5 | 2 | 0,6 | 0 | 0,2 | 0 |
| fsm_rd_135 | 2 | 5 | 1 | 2 | 0,2 | 0 | 0,1 | 0 |
| fsm_rd_136 | 2 | 6 | 2 | 1 | 0,3 | 0 | 0,1 | 50 |
| fsm_rd_140 | 2 | 8 | 8 | 2 | 0,2 | 0 | 0,3 | 0 |
| fsm_rd_143 | 5 | 8 | 2 | 4 | 0,5 | 0,1 | 0,6 | 50 |
| fsm_rd_144 | 4 | 54 | 4 | 4 | 0 | 0,8 | 8,4 | 0 |
| fsm_rd_145 | 4 | 54 | 10 | 4 | 0 | 0,8 | 13,5 | 0 |
| fsm_rd_146 | 4 | 60 | 4 | 4 | 0 | 0,9 | 9,4 | 0 |
| fsm_rd_147 | 4 | 61 | 6 | 4 | 0,1 | 1 | 11,4 | 0 |
| fsm_rd_148 | 4 | 37 | 7 | 4 | 0,2 | 0,6 | 7,5 | 0 |
| fsm_rd_149 | 7 | 33 | 11 | 6 | 0 | 2,1 | 35,1 | 50 |
| fsm_rd_150 | 7 | 40 | 9 | 7 | 0,4 | 5 | 75 | 0 |
| fsm_rd_152 | 7 | 34 | 7 | 7 | 0,3 | 4,2 | 55,2 | 0 |

Apéndice D

# *RandomFSMIM*: Una herramienta para generar máquinas de estado aleatorias

*RandomFSMIM* es una herramienta que permite crear máquinas de estado aleatorias para la realización de pruebas y evaluación de rendimiento de FSMIM. Además de las características de las FSM, la herramienta permite controlar algunas propiedades de las FSMIM que son independientes del proceso de optimización. Admite las siguientes opciones:

**Opción '-i'**: Permite especificar el número de entradas de la FSM.

**Opción '-e'**: Permite especificar el número de canales de selección de la FSMIM.

**Opción '-o'**: Permite especificar el número de salidas de las FSM.

**Opción '-s'**: Permite especificar el número de estados de la FSM.

**Opción '-rd'**: Permite especificar el ratio de indeterminaciones de la MSE.

**Opción '-rt'**: Permite especificar el ratio de transiciones de la FSM respecto al número máximo de transiciones de la FSMIM. El número máximo de transiciones es igual a $2^{N_c}$, donde $N_c$ es el número de canales de selección.

**Opción '-rte'**: Permite especificar el ratio de transiciones respecto al número total de transiciones efectivas. El número total de transiciones efectivas

es el número de transiciones asociadas a las entradas efectivas y se calcula como $\sum_{i=1}^{N_s} 2^{N_e^i}$, donde $N_s$ es el número de estados y $N_e^i$ es el número de entradas efectivas del estado $i$. Este parámetro sólo afecta a la FSM generada si se omite la opción '-rt'.

**Opción '-pd'**: Permite especificar la probabilidad de aparición de indeterminaciones en la salida (por defecto toma el valor 0.33).

**Opción '-p1'**: Permite especificar la probabilidad de aparición del valor '1' en la salida (por defecto toma el valor 0.33).

**Opción '-pr'**: Indica que deben usarse valores cercanos a las probabilidades establecidas por las opciones '-pd' y '-p1' en el caso de que no sea posible respetarlas. Si no se especifica esta opción y no es posible mantener las probabilidades, se genera un mensaje de error y finaliza la ejecución del programa.

**Opción '-m'**: Indica que debe generarse una descripción de la FSMIM en formato *KISS2IM* (véase el apéndice F). Si se omite, se genera una descripción de la FSM en formato *KISS2*.

Apéndice E

# Batería de pruebas BP-184

La batería de pruebas BP-184 está compuesta por 108 FSM aleatorias que se han generado utilizando la herramienta *RandomFSMIM* con los siguientes parámetros:

1. **Número de canales de selección ($N_c$)**: toma los valores 7, 8 y 9.
2. **Número de entradas de la FSM ($N_e$)**: toma los valores $N_c + 0$, $N_c + 3$ y $N_c + 6$ para cada valor de $N_c$.
3. **Ratio de indeterminaciones de la MSE ($R$)**: toma los valores 0.25, 0.5 y 0.75 para cada valor de la tupla $(N_c, N_e)$.
4. **Número de estados ($N_s$)**: para cada valor de la tupla $(N_c, N_e, R)$, se generan cuatro valores aleatorios para $N_s$ de forma que el número de bits de codificación de estados ($B_s$) de cada valor sea 6, 7, 8 y 9.
5. **Ratio de transiciones efectivas (opción '-rte')**: toma un valor aleatorio en el intervalo $[0.2, 0.3]$.
6. **Número de salidas**: toma un valor aleatorio en el intervalo $[N_{max}, N_{max} + 3]$ donde $N_{max} = 11 - B_s$.
7. **Probabilidad de aparición de '1' en la salida**: toma un valor aleatorio en el intervalo $[0.01, 0.05]$. Las FSM se han generado con la opción '-pr' activa, lo que implica que la herramienta usará el valor más cercano posible a la probabilidad de aparición de '1' especificada.
8. **Probabilidad de aparición de indeterminaciones en la salida**: toma el valor 0.33 (valor por defecto de la herramienta).

La tabla E.1 muestra las características de las FSM obtenidas con los parámetros indicados. Las columnas "N.º ents.", "N.º ests." y "N.º sals." contienen el número de entradas, de salidas y de estados de la FSM, respectivamente. La columna "Ratio inds. MSE" contiene el ratio de indeterminaciones de la MSE. La columna "Tam. inic." contiene el tamaño inicial de la FSMIM-S (medido en Kbits). Finalmente, las columnas "N.º cans. sel.", "Prof. inic." y "Red. inic." contienen el número de canales de selección, la profundidad inicial (medida en Kpalabras) y la reducción inicial de profundidad de la FSMIM, respectivamente. Estos parámetros se describen en la sección 5.2.3.

**Tabla E.1:** Características de las FSM de la batería de pruebas BP-184

| Nombre | N.º ents. | N.º ests. | N.º sals. | N.º cans. sel. | Ratio inds. MSE | Prof. inic. (Kpal.) | Tam. inic. (Kbits) | Red. inic. (%) |
|---|---|---|---|---|---|---|---|---|
| fsm_e7i10o2s265 | 10 | 265 | 2 | 7 | 0,7 | 33,1 | 364,4 | 87,5 |
| fsm_e7i10o2s428 | 10 | 428 | 2 | 7 | 0,5 | 53,5 | 588,5 | 87,5 |
| fsm_e7i10o3s189 | 10 | 189 | 3 | 7 | 0,7 | 23,6 | 259,9 | 87,5 |
| fsm_e7i10o4s122 | 10 | 122 | 4 | 7 | 0,5 | 15,2 | 167,8 | 87,5 |
| fsm_e7i10o4s249 | 10 | 249 | 4 | 7 | 0,5 | 31,1 | 373,5 | 87,5 |
| fsm_e7i10o4s309 | 10 | 309 | 4 | 7 | 0,2 | 38,6 | 502,1 | 87,5 |
| fsm_e7i10o5s105 | 10 | 105 | 5 | 7 | 0,2 | 13,1 | 157,5 | 87,5 |
| fsm_e7i10o5s159 | 10 | 159 | 5 | 7 | 0,2 | 19,9 | 258,4 | 87,5 |
| fsm_e7i10o5s39 | 10 | 39 | 5 | 7 | 0,2 | 4,9 | 53,6 | 87,5 |
| fsm_e7i10o5s76 | 10 | 76 | 5 | 7 | 0,8 | 9,5 | 114 | 87,5 |
| fsm_e7i10o6s47 | 10 | 47 | 6 | 7 | 0,5 | 5,9 | 70,5 | 87,5 |
| fsm_e7i10o6s61 | 10 | 61 | 6 | 7 | 0,7 | 7,6 | 91,5 | 87,5 |
| fsm_e7i13o3s177 | 13 | 177 | 3 | 7 | 0,5 | 22,1 | 243,4 | 98,4 |
| fsm_e7i13o3s486 | 13 | 486 | 3 | 7 | 0,7 | 60,8 | 729 | 98,4 |
| fsm_e7i13o4s196 | 13 | 196 | 4 | 7 | 0,8 | 24,5 | 294 | 98,4 |
| fsm_e7i13o4s218 | 13 | 218 | 4 | 7 | 0,2 | 27,2 | 327 | 98,4 |
| fsm_e7i13o4s454 | 13 | 454 | 4 | 7 | 0,5 | 56,8 | 737,8 | 98,4 |
| fsm_e7i13o4s466 | 13 | 466 | 4 | 7 | 0,2 | 58,2 | 757,2 | 98,4 |
| fsm_e7i13o4s69 | 13 | 69 | 4 | 7 | 0,5 | 8,6 | 94,9 | 98,4 |
| fsm_e7i13o6s125 | 13 | 125 | 6 | 7 | 0,7 | 15,6 | 203,1 | 98,4 |
| fsm_e7i13o6s128 | 13 | 128 | 6 | 7 | 0,2 | 16 | 208 | 98,4 |
| fsm_e7i13o6s62 | 13 | 62 | 6 | 7 | 0,7 | 7,8 | 93 | 98,4 |
| fsm_e7i13o6s63 | 13 | 63 | 6 | 7 | 0,2 | 7,9 | 94,5 | 98,4 |
| fsm_e7i13o7s58 | 13 | 58 | 7 | 7 | 0,5 | 7,2 | 94,2 | 98,4 |
| fsm_e7i7o2s487 | 7 | 487 | 2 | 7 | 0,5 | 60,9 | 669,6 | 0 |
| fsm_e7i7o4s171 | 7 | 171 | 4 | 7 | 0,2 | 21,4 | 256,5 | 0 |
| fsm_e7i7o4s266 | 7 | 266 | 4 | 7 | 0,2 | 33,2 | 432,2 | 0 |
| fsm_e7i7o4s475 | 7 | 475 | 4 | 7 | 0,7 | 59,4 | 771,9 | 0 |
| fsm_e7i7o4s74 | 7 | 74 | 4 | 7 | 0,7 | 9,2 | 101,8 | 0 |
| fsm_e7i7o5s164 | 7 | 164 | 5 | 7 | 0,5 | 20,5 | 266,5 | 0 |
| fsm_e7i7o5s222 | 7 | 222 | 5 | 7 | 0,7 | 27,8 | 360,8 | 0 |
| fsm_e7i7o5s85 | 7 | 85 | 5 | 7 | 0,5 | 10,6 | 127,5 | 0 |
| fsm_e7i7o6s41 | 7 | 41 | 6 | 7 | 0,5 | 5,1 | 61,5 | 0 |
| fsm_e7i7o6s59 | 7 | 59 | 6 | 7 | 0,2 | 7,4 | 88,5 | 0 |
| fsm_e7i7o6s98 | 7 | 98 | 6 | 7 | 0,2 | 12,2 | 159,2 | 0 |
| fsm_e7i7o7s42 | 7 | 42 | 7 | 7 | 0,7 | 5,2 | 68,2 | 0 |
| fsm_e8i11o2s479 | 11 | 479 | 2 | 8 | 0,8 | 119,8 | 1317,2 | 87,5 |
| fsm_e8i11o3s492 | 11 | 492 | 3 | 8 | 0,5 | 123 | 1476 | 87,5 |
| fsm_e8i11o4s105 | 11 | 105 | 4 | 8 | 0,5 | 26,2 | 288,8 | 87,5 |
| fsm_e8i11o4s167 | 11 | 167 | 4 | 8 | 0,8 | 41,8 | 501 | 87,5 |
| fsm_e8i11o4s188 | 11 | 188 | 4 | 8 | 0,2 | 47 | 564 | 87,5 |
| fsm_e8i11o4s193 | 11 | 193 | 4 | 8 | 0,5 | 48,2 | 579 | 87,5 |
| fsm_e8i11o4s472 | 11 | 472 | 4 | 8 | 0,2 | 118 | 1534 | 87,5 |
| fsm_e8i11o5s35 | 11 | 35 | 5 | 8 | 0,8 | 8,8 | 96,2 | 87,5 |
| fsm_e8i11o5s45 | 11 | 45 | 5 | 8 | 0,5 | 11,2 | 123,8 | 87,5 |
| fsm_e8i11o6s107 | 11 | 107 | 6 | 8 | 0,8 | 26,8 | 347,8 | 87,5 |
| fsm_e8i11o6s72 | 11 | 72 | 6 | 8 | 0,2 | 18 | 234 | 87,5 |
| fsm_e8i11o7s60 | 11 | 60 | 7 | 8 | 0,2 | 15 | 195 | 87,5 |

Continúa en la página siguiente

**Tabla E.1:** Continúa desde la página anterior

| Nombre | N.º ents. | N.º ests. | N.º sals. | N.º cans. sel. | Ratio inds. MSE | Prof. inic. (Kpal.) | Tam. inic. (Kbits) | Red. inic. (%) |
|---|---|---|---|---|---|---|---|---|
| fsm_e8i14o2s291 | 14 | 291 | 2 | 8 | 0,8 | 72,8 | 800,2 | 98,4 |
| fsm_e8i14o2s445 | 14 | 445 | 2 | 8 | 0,5 | 111,2 | 1223,8 | 98,4 |
| fsm_e8i14o4s236 | 14 | 236 | 4 | 8 | 0,8 | 59 | 708 | 98,4 |
| fsm_e8i14o4s273 | 14 | 273 | 4 | 8 | 0,2 | 68,2 | 887,2 | 98,4 |
| fsm_e8i14o4s91 | 14 | 91 | 4 | 8 | 0,8 | 22,8 | 250,2 | 98,4 |
| fsm_e8i14o4s92 | 14 | 92 | 4 | 8 | 0,2 | 23 | 253 | 98,4 |
| fsm_e8i14o5s171 | 14 | 171 | 5 | 8 | 0,2 | 42,8 | 555,8 | 98,4 |
| fsm_e8i14o5s213 | 14 | 213 | 5 | 8 | 0,5 | 53,2 | 692,2 | 98,4 |
| fsm_e8i14o5s41 | 14 | 41 | 5 | 8 | 0,2 | 10,2 | 112,8 | 98,4 |
| fsm_e8i14o5s43 | 14 | 43 | 5 | 8 | 0,8 | 10,8 | 118,2 | 98,4 |
| fsm_e8i14o6s52 | 14 | 52 | 6 | 8 | 0,5 | 13 | 156 | 98,4 |
| fsm_e8i14o6s93 | 14 | 93 | 6 | 8 | 0,5 | 23,2 | 302,2 | 98,4 |
| fsm_e8i8o2s268 | 8 | 268 | 2 | 8 | 0,5 | 67 | 737 | 0 |
| fsm_e8i8o2s458 | 8 | 458 | 2 | 8 | 0,2 | 114,5 | 1259,5 | 0 |
| fsm_e8i8o3s185 | 8 | 185 | 3 | 8 | 0,2 | 46,2 | 508,8 | 0 |
| fsm_e8i8o4s266 | 8 | 266 | 4 | 8 | 0,8 | 66,5 | 864,5 | 0 |
| fsm_e8i8o4s84 | 8 | 84 | 4 | 8 | 0,2 | 21 | 231 | 0 |
| fsm_e8i8o5s190 | 8 | 190 | 5 | 8 | 0,5 | 47,5 | 617,5 | 0 |
| fsm_e8i8o5s203 | 8 | 203 | 5 | 8 | 0,8 | 50,8 | 659,8 | 0 |
| fsm_e8i8o6s36 | 8 | 36 | 6 | 8 | 0,5 | 9 | 108 | 0 |
| fsm_e8i8o6s62 | 8 | 62 | 6 | 8 | 0,8 | 15,5 | 186 | 0 |
| fsm_e8i8o6s86 | 8 | 86 | 6 | 8 | 0,8 | 21,5 | 279,5 | 0 |
| fsm_e8i8o6s87 | 8 | 87 | 6 | 8 | 0,5 | 21,8 | 282,8 | 0 |
| fsm_e8i8o7s50 | 8 | 50 | 7 | 8 | 0,2 | 12,5 | 162,5 | 0 |
| fsm_e9i12o3s223 | 12 | 223 | 3 | 9 | 0,5 | 111,5 | 1226,5 | 87,5 |
| fsm_e9i12o4s223 | 12 | 223 | 4 | 9 | 0,7 | 111,5 | 1338 | 87,5 |
| fsm_e9i12o4s288 | 12 | 288 | 4 | 9 | 0,8 | 144 | 1872 | 87,5 |
| fsm_e9i12o4s395 | 12 | 395 | 4 | 9 | 0,5 | 197,5 | 2567,5 | 87,5 |
| fsm_e9i12o4s497 | 12 | 497 | 4 | 9 | 0,2 | 248,5 | 3230,5 | 87,5 |
| fsm_e9i12o4s84 | 12 | 84 | 4 | 9 | 0,8 | 42 | 462 | 87,5 |
| fsm_e9i12o5s103 | 12 | 103 | 5 | 9 | 0,2 | 51,5 | 618 | 87,5 |
| fsm_e9i12o5s182 | 12 | 182 | 5 | 9 | 0,2 | 91 | 1183 | 87,5 |
| fsm_e9i12o6s111 | 12 | 111 | 6 | 9 | 0,5 | 55,5 | 721,5 | 87,5 |
| fsm_e9i12o6s38 | 12 | 38 | 6 | 9 | 0,2 | 19 | 228 | 87,5 |
| fsm_e9i12o6s57 | 12 | 57 | 6 | 9 | 0,7 | 28,5 | 342 | 87,5 |
| fsm_e9i12o7s53 | 12 | 53 | 7 | 9 | 0,5 | 26,5 | 344,5 | 87,5 |
| fsm_e9i15o2s495 | 15 | 495 | 2 | 9 | 0,5 | 247,5 | 2722,5 | 98,4 |
| fsm_e9i15o3s165 | 15 | 165 | 3 | 9 | 0,2 | 82,5 | 907,5 | 98,4 |
| fsm_e9i15o3s202 | 15 | 202 | 3 | 9 | 0,5 | 101 | 1111 | 98,4 |
| fsm_e9i15o3s374 | 15 | 374 | 3 | 9 | 0,7 | 187 | 2244 | 98,4 |
| fsm_e9i15o4s272 | 15 | 272 | 4 | 9 | 0,2 | 136 | 1768 | 98,4 |
| fsm_e9i15o4s83 | 15 | 83 | 4 | 9 | 0,5 | 41,5 | 456,5 | 98,4 |
| fsm_e9i15o5s157 | 15 | 157 | 5 | 9 | 0,7 | 78,5 | 1020,5 | 98,4 |
| fsm_e9i15o6s105 | 15 | 105 | 6 | 9 | 0,7 | 52,5 | 682,5 | 98,4 |
| fsm_e9i15o6s96 | 15 | 96 | 6 | 9 | 0,2 | 48 | 624 | 98,4 |
| fsm_e9i15o7s36 | 15 | 36 | 7 | 9 | 0,5 | 18 | 234 | 98,4 |
| fsm_e9i15o7s40 | 15 | 40 | 7 | 9 | 0,8 | 20 | 260 | 98,4 |
| fsm_e9i15o7s42 | 15 | 42 | 7 | 9 | 0,2 | 21 | 273 | 98,4 |
| fsm_e9i9o2s401 | 9 | 401 | 2 | 9 | 0,2 | 200,5 | 2205,5 | 0 |
| fsm_e9i9o3s193 | 9 | 193 | 3 | 9 | 0,2 | 96,5 | 1061,5 | 0 |
| fsm_e9i9o3s226 | 9 | 226 | 3 | 9 | 0,5 | 113 | 1243 | 0 |
| fsm_e9i9o3s464 | 9 | 464 | 3 | 9 | 0,5 | 232 | 2784 | 0 |
| fsm_e9i9o4s455 | 9 | 455 | 4 | 9 | 0,7 | 227,5 | 2957,5 | 0 |
| fsm_e9i9o4s77 | 9 | 77 | 4 | 9 | 0,2 | 38,5 | 423,5 | 0 |
| fsm_e9i9o5s128 | 9 | 128 | 5 | 9 | 0,8 | 64 | 768 | 0 |
| fsm_e9i9o5s206 | 9 | 206 | 5 | 9 | 0,7 | 103 | 1339 | 0 |
| fsm_e9i9o5s90 | 9 | 90 | 5 | 9 | 0,5 | 45 | 540 | 0 |
| fsm_e9i9o6s49 | 9 | 49 | 6 | 9 | 0,5 | 24,5 | 294 | 0 |
| fsm_e9i9o6s59 | 9 | 59 | 6 | 9 | 0,2 | 29,5 | 354 | 0 |
| fsm_e9i9o7s63 | 9 | 63 | 7 | 9 | 0,7 | 31,5 | 409,5 | 0 |

Apéndice F

# El formato *KISS2IM*

El formato *KISS2IM* (del ingles *KISS2* with Imput Multiplexing) es una extensión del formato *KISS2* para permitir la especificación de FSMIM. Esto ha permitido que tanto FSMIM-Gen 1.2 como las diferentes herramientas que forman parte de FSMIM-Gen 2.0 puedan interaccionar entre ellas.

Al igual que ocurría con el formato *KISS2*, los archivos *KISS2IM* son archivos de texto con la estructura mostrada en la figura F.1, donde el texto encerrado entre corchetes es opcional, y el texto delimitado por los símbolos '`<>`' es un campo cuyo valor debe ser especificado por el usuario. El carácter '`#`' marca el inicio de un comentario que se extiende hasta el final de línea.

Un fichero con formato *KISS2IM* se divide en tres partes: la primera contiene la definición de los parámetros generales de la FSMIM (líneas 2 a 7), la segunda contiene la definición de la MSE (líneas 10 a 13) y, finalmente, la tercera contiene las transiciones de la FSMIM (líneas 16 a 18).

### Definición de parámetros generales

Cada línea de la definición de parámetros generales contiene un único campo, precedido por una palabra clave (`.i`, `.o`, `.p`, `.s`, `.r` o `.ris`). Los campos tienen el siguiente significado:

**num-entradas** Es el número de entradas de la FSMIM.

**num-salidas** Es el número de salidas de la FSMIM.

**num-términos** Es el número de 6-tuplas que describen las transiciones de la FSMIM.

**num-estados** Es el número de estados de la FSMIM. Debe coincidir con el número de estados diferentes que aparecen en los campos `<estado-pres>` y `<estado-sig>` que aparecen en la definición de las transiciones de la FSMIM.

**estado-reset** Es el nombre simbólico del estado de reset de la FSMIM. Debe coincidir con alguno de los nombres de los estados que aparecen en los campos `<estado-pres>` de la definición de las transiciones de la FSMIM. Por defecto, toma el valor del primer campo `<estado-pres>` que aparece en la definición de las transiciones.

**selec-reset** Es un número que identifica a la selección de entradas del metaestado inicial, definido por la tupla (`<estado-reset>`, `<selec-reset>`). Por defecto, toma el valor del primer campo `<sel-pres>` que aparece en la definición de las transiciones de la FSMIM.

### Definición de la MSE

Las líneas de la definición de la MSE comienzan con la palabra clave '`.is`' seguida de un índice. Cada línea corresponde a una selección de entradas que se identifica por el índice. Los índices puede aparecer desordenados, pero no

```
1  # Estructura general de un fichero KISS2IM
2  .i <num-entradas>
3  .o <num-salidas>
4  [.p <num-teérminos>]
5  [.s <num-estados>]
6  [.r <estado-reset>]
7  [.ris <selec-reset>]
8
9  # Matriz de Selección de Entradas
10 .is 1   <id-entrada> <id-entrada> ... <id-entrada>
11 .is 2   <id-entrada> <id-entrada> ... <id-entrada>
12   ⋮
13 .is <N> <id-entrada> <id-entrada> ... <id-entrada>
14
15 # Transiciones
16 <sel-pres> <entrada> <estado-pres> <estado-sig> <sel-sig> <salida>
17   ⋮
18 <sel-pres> <entrada> <estado-pres> <estado-sig> <sel-sig> <salida>
19 [.e]
```

**Figura F.1:** Formato *KISS2IM*

deben estar repetidos y deben encontrarse en el rango $[1, N]$, donde $N$ es el número de selecciones de entradas especificadas ($N$ corresponde al número de líneas de la MSE). Cada campo `<id-entrada>` puede tomar uno de los siguientes valores:

- Los valores `0` o `1`, que representan a las constantes seleccionadas.
- El carácter '`-`', que representa una *Entrada Seleccionada Indeterminada* (ESI).
- La expresión `i<X>`, que representa a la entrada seleccionada de índice `<X>`, que debe ser un valor entre 1 y `<num-entradas>`, ambos incluidos.

El número de campos `<id-entrada>` de cada línea debe ser igual al número de canales de selección de la FSMIM.

### Definición de las transiciones

La definición de las transiciones de la FSMIM consiste en un conjunto de líneas (una para cada transición) con los siguientes campos:

**sel-pres** Es el identificador de la selección de entradas presente de la transición. Su formato es `is<X>`, donde `<X>` es el índice de la selección de entradas correspondiente.

**entrada** Es una secuencia de valores '`0`', '`1`' y '`-`', de longitud igual al número de campos `<id-entrada>` de cada selección de entradas. Cada valor de la secuencia representa al valor de la entrada seleccionada por el campo `<id-entrada>` correspondiente de la selección de entradas `<sel-pres>`.

**estado-pres** Es el nombre simbólico del estado presente de la transición.

**estado-sig** Es el nombre simbólico del estado siguiente de la transición.

**sel-sig** Es el identificador de la selección de entradas siguiente de la transición. Su formato es `is<X>`, donde `<X>` es el índice de la selección de entradas correspondiente.

**salida** Es una secuencia de valores '`0`', '`1`' y '`-`', de longitud `<num-salidas>`.

Obviamente, la tupla (`<estado-pres>`, `<sel-pres>`) representa al metaestado presente de la FSMIM y la tupla (`<estado-sig>`, `<sel-sig>`), al metaestado siguiente.

La figura F.2 muestra un ejemplo de la descripción *KISS2IM* de una FSMIM.

```
.i 7
.o 2
.p 170
.s 8
.r st1

.is 1  i7  0 i6  0 i5  - i4
.is 2  i7 i1 i6 i2 i5 i3 i4
.is 3  i7 i1 i6 i2 i5 i3 i4
.is 4  i7 i1 i6 i2 i5 i3 i4
.is 5  i7  1 i6 i2 i5 i3 i4
.is 6  i7  0 i6 i2 i5 i3 i4
.is 7  i7  1 i6 i2 i5 i3 i4
.is 8  i7  0 i6  - i5 i3 i4
.is 9  i7  1 i6  1 i5 i3 i4
.is 10 i7  0 i6  1 i5 i3 i4
.is 11 i7  1 i6  0 i5  -  -
.is 12 i7  1 i6  1 i5  -  -
.is 13 i7  0 i6  1  -  -  -
.is 14 i7  1 i6  0  -  -  -
.is 15 i7  0 i6  0  -  -  -
.is 16 i7  1  -  1  -  -  -
.is 17 i7  0  -  1  -  -  -
.is 18  -  1  -  0  -  -  -
.is 19  -  0  -  0  -  -  -

is1  00000-0 st1 st2 is2 1-
is1  00001-0 st1 st3 is3 1-
is1  00100-0 st1 st3 is3 1-
is1  10000-0 st1 st3 is3 1-
is1  00001-1 st1 st4 is4 1-
is1  00000-1 st1 st4 is4 1-
⋮
is17 10-1--- st8 st1 is1 -0
is18 -1-0--- st8 st1 is1 -0
is19 -0-0--- st8 st1 is1 -1
.e
```

**Figura F.2:** Ejemplo de descripción *KISS2IM*

Apéndice G

# Generación automática de la descripción VHDL de una ROM

Para poder generar las FSMIM con profundidad ajustada a bloques, FSMIM-Gen 2.0 necesita conocer el número de bloques de memoria empotrados que ocupa la implementación de la ROM de la FSMIM. En el caso de la herramienta FSMIM-Gen 1.2, el procedimiento de ajuste de profundidad necesita calcular la profundidad de la ROM implementada con bloques. Ambos parámetros deben calcularse múltiples veces durante la generación de la FSMIM, antes de su implementación.

Puesto que no es posible determinar a priori el número de bloques ni la profundidad de las implementaciones de ROM realizadas por la herramienta de síntesis[1], se decidió desarrollar una herramienta para generar descripciones estructurales de memorias ROM, utilizando como componentes los bloques de memoria empotrados del dispositivo FPGA.

Esta herramienta, denominada ROMGEN, permite generar dos tipos de descripciones de ROM: una descripción estructural con bloques de memoria empotrados y una descripción de comportamiento.

Las descripciones de comportamiento se han utilizado para los experimentos publicados en [García-Vargas y Senhadji-Navarro, 2015] y analizados en la sección 5.3.2), mientras que las descripciones estructurales se han utilizado en el resto de experimentos realizados en esta tesis.

[1] Al no ser *ISE WebPACK* una herramienta de software libre, la información sobre los algoritmos empleados para generar implementaciones de ROM no está accesible.

## G.1. Descripción de comportamiento de la ROM

La figura G.1 muestra la descripción de comportamiento generada por ROMGEN para una ROM con geometría 19K × 6 (la inicialización del contenido de la ROM aparece incompleta).

A partir de esta descripción, la herramienta de síntesis es capaz de inferir una memoria ROM y de implementarla utilizando bloques de memoria empotrados. La descripción está basada en la plantilla que proporciona la propia herramienta *ISE WebPACK*.

```
8  entity rom19K6 is
9      port(   clk: in std_logic;
10             reset: in std_logic;
11             addr: in std_logic_vector(15-1 downto 0);
12             data: out std_logic_vector(6-1 downto 0));
13 end rom19K6;
14
15 architecture rom19K6_arch of rom19K6 is
16
17     constant nwords_max : integer := 19456-1;
18     constant data_max : integer := 6-1;
19     type rom_type is array (0 to nwords_max)
20         of std_logic_vector( data_max downto 0 );
21     constant ROM: rom_type := (
22     "110001",
23     "011100",
24     .
25     .
26     "001010",
27     "000000" );
28
29 begin
30     process (clk)
31     begin
32         if (clk'event and clk = '1') then
33             if reset = '1' then
34                 data <= "000000";
35             else
36                 data <= ROM( conv_integer(addr) );
37             end if;
38         end if;
39     end process;
40 end rom19K6_arch;
```

**Figura G.1:** Ejemplo de descripción de comportamiento de una ROM con geometría 19K × 6.

## G.2. Descripción estructural de la ROM

### G.2.1. Cálculo de la distribución de bloques

En esta sección se describe el algoritmo utilizado para determinar la geometría y la interconexión de los bloques que forman la descripción de la ROM. Dicho algoritmo calcula además el número de bloques utilizados y la profundidad de la ROM.

Todas las familias de dispositivos FPGA actuales de Xilinx disponen de bloques de memoria. Estos bloques pueden ser configurados con diferentes geometrías y, en algunas familias, pueden dividirse en dos bloques independientes de la mitad de capacidad. Por ejemplo, los bloques de memoria de 18 Kbits de la Spartan-6 pueden dividirse en dos bloques independientes de 9 Kbits.

Las características de interés para generar una descripción utilizando los bloques de memoria son las geometrías que se pueden configurar en cada bloque y el peso del bloque. El peso permite distinguir entre un bloque completo y cualquiera de los bloques que lo forman, y debe ser un valor inversamente proporcional al número de partes en las que se puede dividir el bloque completo.

**Definición G.1.** *Se define una* **geometría de bloque** *(o simplemente, geometría) como la tupla $(p, a, w)$, donde $p \in \mathbb{N}_+$ es la profundidad del bloque en número de palabras (o filas), $a \in \mathbb{N}_+$ es el ancho en número de bits (o columnas) y $w \in \mathbb{R}_+$ es el peso.*

*Dada una geometría $g = (p, a, w)$, la profundidad de $g$ será denotada por prof(g); el ancho, por ancho(g), y el peso, por peso(g).*

Por ejemplo, las características de los bloques de la Spartan-6 se pueden modelar con el siguiente conjunto de geometrías de bloques, donde el peso 1 corresponde a un bloque de 18 Kbits y el peso $\frac{1}{2}$, a uno de 9 Kbits (véase la sección K.1):

$$G = \{(16\text{K}, 1, 1), (8\text{K}, 2, 1), (4\text{K}, 4, 1), (2\text{K}, 9, 1), (1\text{K}, 18, 1), (512, 36, 1),\\ (8\text{K}, 1, \tfrac{1}{2}), (4\text{K}, 2, \tfrac{1}{2}), (2\text{K}, 4, \tfrac{1}{2}), (1\text{K}, 9, \tfrac{1}{2}), (512, 18, \tfrac{1}{2}), (256, 36, \tfrac{1}{2})\}$$

**Definición G.2.** *Sea $G$ un conjunto de geometrías de bloque. Sean $f$ y $c$ la profundidad (en número de palabras o filas) y el ancho (en número de bits o columnas) de una ROM, respectivamente. Se define una* **distribución de bloques** *como un conjunto finito de tuplas $([f_0, f_1], [c_0, c_1], g)$, donde $[f_0, f_1]$ es un rango de filas tal que $1 \leq f_0, f_1 \leq f$, $[c_0, c_1]$ es un rango de columnas tal que $1 \leq c_0, c_1 \leq c$, y $g \in G$ es una geometría de bloque. Cada tupla*

*establece un rango de filas y columnas de la ROM que serán cubiertos por 1 o varios bloques con geometría $g$.*

El algoritmo para calcular la distribución de bloques se divide en dos partes: (a) un procedimiento que selecciona la geometría más adecuada (según el criterio del algoritmo) para cubrir una memoria con $n$ palabras de $m$ bits (algoritmo 13), y (b) el procedimiento que determina la distribución de los bloques propiamente dicha (algoritmo 14).

El procedimiento SELECCIONA (algoritmo 13) está diseñado para el dispositivo Spartan-6 o para cualquier dispositivo con bloques de dos pesos diferentes como máximo. Dada una memoria de dimensión $n_f \times n_c$, este procedimiento sólo selecciona una geometría de bloque con peso mínimo si la profundidad y el ancho de la geometría pueden contener a la memoria. En caso contrario, selecciona una geometría de bloque con peso máximo, siendo necesarios varios bloques si dicha geometría no puede contener a la memoria (es decir, si su profundidad es menor que $n_f$ o su ancho es menor que $n_c$).

El procedimiento DISBLOQ (algoritmo 14), además de la distribución de bloques de la ROM, calcula el número de bloques ocupados (que es la suma ponderada de los bloques utilizados por el peso de cada bloque) y la profundidad. El funcionamiento del algoritmo se ilustra a continuación con un ejemplo.

**Ejemplo G.3.** *Supóngase que se calcula la distribución de bloques para una memoria de* $35\text{K} \times 10$ *(figura G.3a), utilizando el conjunto de geometrías del dispositivo Spartan-6 (G). La llamada al procedimiento* DISBLOQ$(n_f = 35\text{K}, n_c = 10, G)$ *realizaría los cálculos que se indican a continuación.*

*En la primera iteración del bucle, la llamada al procedimiento* SELECCIONA$(35\text{K}, 10, G)$ *devuelve la geometría* $(16\text{K}, 1, 1)$*. Con la geometría obtenida, se calcula el número de filas de bloques* $b_f = \lfloor \frac{35\text{K}}{16\text{K}} \rfloor = 2$ *y el*

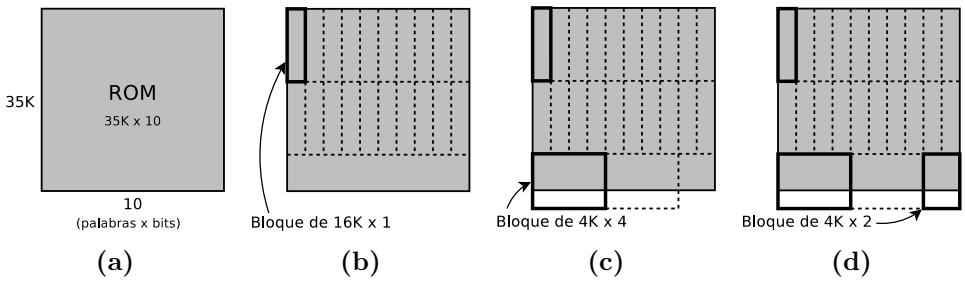



**Figura G.3:** Ejemplo del cálculo de la distribución de bloques para una ROM con geometría $35\text{K} \times 10$.

**Algoritmo 13** Procedimiento para seleccionar la geometría de bloque (procedimiento específico para el dispositivo Spartan-6).

**Entrada:**

$n_f$ ▷ *Número de filas (profundidad).*
$n_c$ ▷ *Número de columnas (ancho).*
$G$ ▷ *Conjunto de geometrías de bloques.*

**Salida:** $g \in G$

1: **procedimiento** SELECCIONA($n_f, n_c, G$)
2: $w_{\text{mín}} \leftarrow \text{mín}_{g \in G}\, \text{peso}(g)$
3: $w_{\text{máx}} \leftarrow \text{máx}_{g \in G}\, \text{peso}(g)$
4: $g \leftarrow (0, 0, 0)$
5: $G' \leftarrow \{g \in G : \text{peso}(g) = w_{\text{mín}}, \text{ancho}(g) \geq n_c, \text{prof}(g) \geq n_f\}$
6: **si** $G' = \emptyset$ **entonces** ▷ *Intentamos con peso $w_{máx}$.*
7: $G' \leftarrow \{g \in G : \text{peso}(g) = w_{\text{máx}}, \text{ancho}(g) \geq n_c, \text{prof}(g) \geq n_f\}$
8: **fin si**
9: **si** $G' \neq \emptyset$ **entonces**
10: $g \leftarrow \arg \text{máx}_{g \in G'}\, \text{prof}(g)$
11: **si no**
12: $G' \leftarrow \{g \in G : \text{peso}(g) = w_{\text{máx}}\}$
13: **si** $n_f > \text{máx}_{g \in G'}\, \text{prof}(g)$ **entonces**
14: $g \leftarrow \arg \text{máx}_{g \in G'}\, \text{prof}(g)$
15: **si no**
16: $g \leftarrow \arg \text{mín}_{g \in G'}\, \text{prof}(g)$,
donde $G' = \{g \in G : \text{peso}(g) = w_{\text{máx}}, \text{prof}(g) \geq n_f\}$
17: **fin si**
18: **fin si**
19: **devolver** $g$
20: **fin procedimiento**

*número de columnas de bloques $b_c = \lfloor \frac{10}{1} \rfloor = 10$ (lineas 8 y 9 del algoritmo 14). Lo que significa que tenemos $2 \times 10$ bloques con geometría $16\text{K} \times 1$ (véase la figura G.3b), que ocupan $n'_f = 2 \cdot 16\text{K} = 32\text{K}$ filas y $n'_c = 10 \cdot 1 = 10$ columnas (líneas 10 y 11).*

*Con estos datos se calcula la distribución de bloques para las columnas residuales (línea 12) con la llamada* DISBLOQ($35\text{K}, 0, G$)*. Puesto que el número de columnas residuales es 0, obtenemos la tupla $(0, 0, \emptyset)$.*

*El número de bloques ocupados es $n_b = 2 \cdot 10 \cdot 1 = 20$ (línea 14) y la distribución obtenida es $D = \{([0, 32\text{K} - 1], [0, 10 - 1], (16\text{K}, 1, 1))\}$ (línea 15).*

*El número de filas que sobran para la siguiente iteración es $n_f = 35\text{K} -$*

**Algoritmo 14** Procedimiento para generar la distribución de bloques de la ROM.

**Entrada:**

$n_f$ ▷ *Número de filas (profundidad).*
$n_c$ ▷ *Número de columnas (ancho).*
$G$ ▷ *Conjunto de geometrías de bloques.*

**Salida:** $(n_b, p_{\text{des}}, D)$, donde $n_b$ es el tamaño en bloques de la ROM, $p_{\text{des}}$ es la profundidad de la descripción por bloques de la ROM y $D$ es la distribución de bloques.

1: **procedimiento** DISBLOQ$(n_f, n_c, G)$
2: $D \leftarrow \emptyset$
3: $n_b \leftarrow 0$ ▷ *Número de bloques.*
4: $d_f \leftarrow 0$ ▷ *Desplazamiento de filas.*
5: $p_{\text{des}} \leftarrow 0$ ▷ *Profundidad de la descripción.*
6: **mientras** $n_f > 0$ **y** $n_c > 0$ **hacer**
7: $g \leftarrow$ SELECCIONA$(n_f, n_c, G)$
8: $b_f \leftarrow \max\{1, \lfloor \frac{n_f}{\text{prof}(g)} \rfloor\}$ ▷ *Número de filas de bloques.*
9: $b_c \leftarrow \max\{1, \lfloor \frac{n_c}{\text{ancho}(g)} \rfloor\}$ ▷ *Número de columnas de bloques.*
10: $n'_f \leftarrow \min\{n_f,\ b_f \cdot \text{prof}(g)\}$ ▷ *Número de filas ocupadas.*
11: $n'_c \leftarrow \min\{n_c,\ b_c \cdot \text{ancho}(g)\}$ ▷ *Número de columnas ocupadas.*
12: $(n'_b, x, D') \leftarrow$ DISBLOQ$(n_f, n_c - n'_c, G)$ ▷ *Columnas residuales.*
13: $D' \leftarrow \{([f_0 + d_f, f_1 + d_f], [c_0 + n'_c, c_1 + n'_c], g)$
$: ([f_0, f_1], [c_0, c_1], g) \in D'\}$
14: $n_b \leftarrow n_b + n'_b + b_f \cdot b_c \cdot \text{peso}(g)$
15: $D \leftarrow D \cup D' \cup \{([d_f, n'_f + d_f - 1], [d_c, n'_c + d_c - 1], g)\}$
16: $n_f \leftarrow n_f - n'_f$
17: $d_f \leftarrow d_f + n'_f$
18: $p_{\text{des}} \leftarrow d_f - n'_f + b_f \cdot \text{prof}(g)$
19: **fin mientras**
20: **devolver** $(n_b, p_{\text{des}}, D)$
21: **fin procedimiento**

32K = 3K *y el desplazamiento de filas es* $d_f = 0 + 32\text{K}$. *Puesto que* $n_f$ *es mayor que cero y, por tanto, habrá otra iteración, el cálculo de* $p_{des}$ *es irrelevante, ya que se volverá a calcular en la siguiente iteración. Puesto que el número de filas que sobran siempre es menor que la profundidad máxima de los bloques, la segunda iteración (si se produce) siempre es la última, ya que bastará una fila más de bloques para ocupar toda la memoria. Esto*

*significa que el bucle de este algoritmo itera dos veces como máximo en todos los casos.*

*En la segunda iteración, la llamada al procedimiento* SELECCIONA *con los parámetros* $(3\text{K}, 10, G)$ *devuelve la geometría de bloque* $(4\text{K}, 4, 1)$. *Con esta geometría se calcula el número de filas de bloques* $b_f = \text{mín}\{1, \lfloor \frac{3\text{K}}{4\text{K}} \rfloor\} = 1$ *y el número de columnas de bloques* $b_c = \lfloor \frac{10}{4} \rfloor = 2$ *(lineas 8 y 9). En esta ocasión tenemos* $1 \times 2$ *bloques con geometría* $4\text{K} \times 4$ *(véase la figura G.3c), que ocupan* $n'_f = \text{mín}\{3\text{K}, 1 \cdot 4\text{K}\} = 3\text{K}$ *filas y* $n'_c = 2 \cdot 4 = 8$ *columnas (líneas 10 y 11).*

*En esta iteración quedan 2 columnas residuales. La llamada* DISBLOQ$(3\text{K}, 2, G)$ *emplea un bloque con geometría* $(4\text{K}, 2, \frac{1}{2})$ *para ocupar el espacio de memoria residual de dimensión* $3\text{K} \times 2$ *(véase la figura G.3c), devolviendo la tupla* $(\frac{1}{2}, 4\text{K}, \{([0, 3\text{K}-1], [0, 2-1], (4\text{K}, 2, \frac{1}{2}))\})$. *Con la distribución obtenida se calcula el nuevo valor de* $D' = \{([32\text{K}, 32\text{K} + 3\text{K} - 1], [8, 8+2-1], (4\text{K}, 2, \frac{1}{2}))\}$ *(línea 13).*

*El número de bloques se incrementa, ocupándose* $n_b = 20 + \frac{1}{2} + 1 \cdot 2 \cdot 1 = 22{,}5$ *bloques. A la distribución* $D$ *se le añaden los conjuntos* $D'$ *y* $\{([32\text{K}, 32\text{K} + 3\text{K} - 1], [0, 8-1], (4\text{K}, 4, \frac{1}{2}))\}$, *dando como resultado la distribución final*

$$\begin{aligned} D = \{&([0, 32\text{K}-1], [0, 9], (16\text{K}, 1, 1)), \\ &([32\text{K}, 35\text{K}-1], [0, 7], (4\text{K}, 4, \tfrac{1}{2})), \\ &([32\text{K}, 35\text{K}-1], [8, 9], (4\text{K}, 2, \tfrac{1}{2}))\}. \end{aligned}$$

*Finalmente, se calcula la profundidad de la descripción* $p_{des} = 35\text{K} - 3\text{K} + 1 \cdot 4\text{K} = 36\text{K}$.

#### G.2.1.1. Cálculo de las funciones tamaño de ROM y tamaño de FSMIM

Sea DISBLOQNB el procedimiento DISBLOQ sin el cálculo de la distribución de bloques ni de la profundidad de la descripción; por lo tanto, sólo devuelve el número de bloques ocupados por la implementación ROM. Dado un conjunto de geometrías $G$, se puede definir de forma trivial la función tamaño de ROM ($R : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_+$) a partir del procedimiento DISBLOQNB como

$$R(p, b) = \text{DISBLOQNB}(p, b, G).$$

Por lo tanto, la función $TAM : MSE^{s \times r} \times \{1, \ldots, n\} \to \mathbb{R}_+$ se define en el caso de una FSMIM-T como

$$TAM(A, k) = R(2^r k,\ n + \lceil \log_2 k \rceil + \sum_{j=1}^{r} \lceil \log_2 CSE_j(A) \rceil),$$

donde $n$ es el número de salidas de la FSMIM (véanse las ecuaciones 2.26, 2.28 y 2.29). En el caso de una FSMIM-S, la función se define como

$$TAM(A,k) = R(2^r k,\ n + \lceil \log_2 s \rceil)$$

(véase la ecuación 2.30).

### G.2.2. Descripción VHDL

Las figuras G.4 a G.7 muestran parte de la descripción estructural generada por ROMGEN para la ROM descrita en la figura G.1. Se ha utilizado para la descripción estructural la siguiente distribución de bloques, que ha sido obtenida con el procedimiento DisBloq:

$$\begin{aligned} D = \{&([0, 16\text{K}-1], [0,5],\ (16\text{K}, 1, 1)\ ),\\ &([16\text{K}, 19\text{K}-1], [0,3],\ (4\text{K}, 4, \tfrac{1}{2})\ ),\\ &([16\text{K}, 19\text{K}-1], [4,5],\ (4\text{K}, 2, \tfrac{1}{2})\ )\}. \end{aligned}$$

La figura G.4 muestra la declaración de la entidad `rom19K6`, de los componentes y de las señales internas. Las líneas 18 a 30 contienen la declaración de los seis componentes correspondientes a los bloques 16K × 1 (todos presentan la misma interfaz). Las líneas 32 a 37 contienen la declaración del componente correspondiente al bloque 4K × 4, y las líneas 39 a 44, la del componente correspondiente al bloque 4K × 2. A continuación se encuentra la declaración del componente correspondiente a los multiplexores 2:1 y de las señales internas.

La figura G.5 muestra la instanciación de los componentes. Los seis bloques de profundidad 16K contienen la información del rango de filas $[0, 16\text{K}-1]$ (líneas 60 a 70), mientras que los dos bloques de profundidad 4K contienen la información del rango de filas $[16\text{K}, 19\text{K}-1]$ (líneas 73 a 75). Puede observarse cómo la única señal de dirección que no está conectada a los bloques es `addr(14)`, ya es la señal que controla el banco de multiplexores, que permite seleccionar el dato entre los dos grupos de bloques. Dicha señal debe ser registrada para que la salida de la ROM sea síncrona (líneas 77 a 82). El *banco de multiplexores de la ROM* está formado por seis multiplexores de dos entradas (líneas 84 a 92). Cada multiplexor determina el valor de un bit diferente de la salida de la ROM (señal `data` de la entidad `rom19K6`). En cada operación de lectura, el banco de multiplexores selecciona la salida de los bloques que contienen la palabra que se va a leer.

Cada bloque de memoria empotrado se describe mediante la descripción de comportamiento de una ROM con la misma geometría que el bloque.

Por ejemplo, la figura G.6 muestra la descripción correspondiente al bloque con geometría 4K $\times$ 4 utilizado por la entidad `rom19K6`. Habitualmente, las herramientas de síntesis de cada fabricante incluyen bibliotecas de componentes VHDL para instanciar los diferentes tipos de recursos disponibles en sus dispositivos FPGA. Sin embargo, la descripción utilizada por ROMGEN permite instanciar los bloques de memoria utilizando código VHDL independiente del dispositivo FPGA y del fabricante (la única información específica que se necesita es la geometría del bloque).

Finalmente, la figura G.7 muestra la descripción de comportamiento genérica del componente multiplexor utilizado por la entidad `rom19K6`.

```
9  entity rom19K6 is
10     port(  clk: in std_logic;
11            reset : in std_logic;
12            addr: in std_logic_vector(15-1 downto 0);
13            data: out std_logic_vector(6-1 downto 0));
14 end rom19K6;
15
16 architecture rom19K6_arch of rom19K6 is
17
18     component rom19K6_subrom1 is
19         port(  clk: in std_logic;
20                reset : in std_logic;
21                addr: in std_logic_vector(14-1 downto 0);
22                data: out std_logic_vector(1-1 downto 0));
23     end component;
24       ⋮
25     component rom19Kx6_subrom6 is
26         port(  clk: in std_logic;
27                reset : in std_logic;
28                addr: in std_logic_vector(14-1 downto 0);
29                data: out std_logic_vector(1-1 downto 0));
30     end component;
31
32     component rom19K6_subrom7 is
33         port(  clk: in std_logic;
34                reset : in std_logic;
35                addr: in std_logic_vector(12-1 downto 0);
36                data: out std_logic_vector(4-1 downto 0));
37     end component;
38
39     component rom19Kx6_subrom8 is
40         port(  clk: in std_logic;
41                reset : in std_logic;
42                addr: in std_logic_vector(12-1 downto 0);
43                data: out std_logic_vector(2-1 downto 0));
44     end component;
45
46     component rom19K6_mux is
47         port(  input : in std_logic_vector;
48                control : in std_logic_vector;
49                output : out std_logic );
50     end component;
51
52     signal mux_input0 : std_logic_vector(1 downto 0);
53     ...
54     signal mux_input5 : std_logic_vector(1 downto 0);
55     signal mux_addr : std_logic_vector(0 downto 0);
56     signal data_aux0 : std_logic_vector(5 downto 0);
57     signal data_aux1 : std_logic_vector(5 downto 0);
58 begin
```

**Figura G.4:** Ejemplo de descripción estructural de una ROM de dimensión 19K × 6: declaración de la entidad `rom19K6`, de los componentes y de las señales internas.

```
58 begin
59     Inst_rom19K6_subrom1: rom19K6_subrom1
60         port map( clk, reset, addr(13 downto 0), data_aux0(0 to 0) );
61     Inst_rom19K6_subrom2: rom19K6_subrom2
62         port map( clk, reset, addr(13 downto 0), data_aux0(1 to 1) );
63     Inst_rom19K6_subrom3: rom19K6_subrom3
64         port map( clk, reset, addr(13 downto 0), data_aux0(2 to 2) );
65     Inst_rom19K6_subrom4: rom19K6_subrom4
66         port map( clk, reset, addr(13 downto 0), data_aux0(3 to 3) );
67     Inst_rom19K6_subrom5: rom19K6_subrom5
68         port map( clk, reset, addr(13 downto 0), data_aux0(4 to 4) );
69     Inst_rom19K6_subrom6: rom19K6_subrom6
70         port map( clk, reset, addr(13 downto 0), data_aux0(5 to 5) );
71
72     Inst_rom19K6_subrom7: rom19K6_subrom7
73         port map( clk, reset, addr(11 downto 0), data_aux1(3 downto 0) );
74     Inst_rom19K6_subrom8: rom19K6_subrom8
75         port map( clk, reset, addr(11 downto 0), data_aux1(5 downto 4) );
76
77     process (clk)
78     begin
79         if  clk'event and clk='1' then
80             mux_addr <= addr(14 downto 14);
81         end if;
82     end process;
83
84     mux_input0 <= data_aux1(0) & data_aux0(0);
85     rom19K6_mux0: rom19K6_mux port map ( input => mux_input0,
86         control => mux_addr(0 downto 0),
87         output => data(0));
88       .
         .
89     mux_input5 <= data_aux1(5) & data_aux0(5);
90     rom19K6_mux5: rom19K6_mux port map ( input => mux_input5,
91         control => mux_addr(0 downto 0),
92         output => data(5));
93
94 end rom19K6_arch;
```

**Figura G.5:** Ejemplo de descripción estructural de una ROM de dimensión 19K × 6: instanciación de los componentes.

```
8  entity rom19K6_subrom7 is
9      port(  clk: in std_logic;
10            reset: in std_logic;
11            addr: in std_logic_vector(12-1 downto 0);
12            data: out std_logic_vector(4-1 downto 0));
13 end rom19K6_subrom7;
14
15 architecture rom19K6_subrom7_arch of rom19K6_subrom7 is
16
17     constant nwords_max : integer := 4096-1;
18     constant data_max : integer := 4-1;
19     type rom_type is array (0 to nwords_max)
20         of std_logic_vector(data_max downto 0);
21     constant ROM: rom_type := (
22     "0110",
23     "1011",
24     :
25     "----",
26     "----" );
27
28 begin
29     process (clk)
30     begin
31         if (clk'event and clk = '1') then
32             if reset = '1' then
33                 data <= "0000";
34             else
35                 data <= ROM( conv_integer(addr) );
36             end if;
37           end if;
38      end process;
39 end rom19K6_subrom7_arch;
```

**Figura G.6:** Ejemplo de descripción de comportamiento del un bloque con geometría $4K \times 4$.

```
13 entity rom19Kx6_mux is
14     port(  input : in std_logic_vector;
15            control : in std_logic_vector;
16            output : out std_logic );
17 end rom19Kx6_mux;
18
19 architecture rom19Kx6_mux_arch of rom19Kx6_mux is
20 begin
21     output <= input( conv_integer( unsigned(control) ) );
22 end rom19Kx6_mux_arch;
```

**Figura G.7:** Ejemplo de descripción de comportamiento genérica del multiplexor.

Apéndice H

# Herramientas para la generación de FSMIM

## H.1. FSMIM-Gen 1.2

FSMIM-Gen 1.2 es una actualización de la aplicación empleada para el estudio experimental realizado en [García-Vargas y Senhadji-Navarro, 2015] (FSMIM-Gen 1.1), que implementa la estrategia de optimización de referencia (sección 5.1.1.1). Esta actualización fue necesaria para poder realizar el ajuste de profundidad en las FSMIM generadas con la estrategia de referencia.

La diferencia principal entre la aplicación original y FSMIM-Gen 1.2 consiste en que esta última permite al usuario especificar una cota inferior del número de grupos de la FSMIM generada. Como se explicará mas adelante, esta modificación permite ajustar la profundidad de una FSMIM realizando múltiples ejecuciones de la herramienta.

FSMIM-Gen 1.2, al igual que todas las versiones anteriores de la herramienta, ha sido desarrollada con el lenguaje de programación Octave 3.8.1 [Eaton et al., 2009]. La aplicación lee la descripción de la FSM de un fichero en formato *KISS2* y escribe la descripción de la FSMIM generada en el fichero de salida. Además del nombre de los ficheros de entrada y de salida, admite las siguientes opciones (sólo se muestran las más importantes):

**Opción** `format`: Permite especificar el formato de la descripción de la FSMIM generada. El valor '`vhdl`' indica que debe generarse una descripción VHDL (sólo para la arquitectura FSMIM-T). El valor

'`kiss2im`' indica que debe generarse una descripción *KISS2IM* de la FSMIM.

**Opción `ngroups`**: Permite especificar el límite inferior del número de grupos que se puede obtener. La herramienta implementa una modificación del procedimiento de agrupación voraz con permutación fija (algoritmo 1, sección 3.2) que termina la agrupación si se alcanza el límite establecido. Un valor 0 indica que no existe ningún límite (en este caso se obtiene el mismo resultado que con el algoritmo de agrupación original).

**Opción `arch`**: Permite indicar si la FSMIM generada será implementada en una arquitectura FSMIM-T (valor '`t`') o FSMIM-S (valor '`s`'). Este parámetro afecta exclusivamente a la función *TAM* que utiliza el algoritmo de agrupación voraz con ordenación lexicográfica para evaluar el coste de la solución encontrada (véase el algoritmo 3, sección 3.3).

### H.1.1. Ajuste de profundidad por bloques mediante FSMIM-Gen 1.2

Debido a que el algoritmo de agrupación de estados implementado por la herramienta siempre intenta obtener el mínimo número de grupos, no se puede realizar el ajuste de profundidad durante el proceso de agrupación de estados.

Por tanto, ha sido necesario un mecanismo alternativo para realizar dicho ajuste que no requiera un cambio sustancial en el algoritmo de agrupación empleado.

El procedimiento implementado para ajustar la profundidad se muestra en el algoritmo 15. En resumen, el procedimiento simplemente consiste en ejecutar FSMIM-Gen 1.2 y comprobar si puede aumentarse el número de grupos en la FSMIM obtenida. En caso afirmativo, se vuelve a ejecutar FSMIM-Gen 1.2 ajustando el parámetro `ngroups` al valor adecuado. El procedimiento se repite mientras sea posible incrementar el número de grupos y mientras no aumente el tamaño (medido en número de bloques de memoria) de la FSMIM generada.

Dada una FSMIM con $r$ canales de selección, cada grupo ocupa $2^r$ direcciones de memoria. Por tanto, el número de grupos que caben en la ROM implementada con bloques se puede calcular fácilmente dividiendo la profundidad de la ROM (obtenida con el algoritmo 14, sección G.2.1) entre $2^r$ (líneas 6 y 14).

**Algoritmo 15** Procedimiento para realizar el ajuste de profundidad con la herramienta FSMIM-Gen 1.2.

**Entrada:** *fsm* ▷ *Nombre del archivo KISS2.*
**Salida:** Archivo *KISS2IM* con la descripción de la FSMIM.

1: Ejecuta FSMIM-Gen 1.2 con el archivo *fsm* para obtener la FSMIM $F = (E, P, t, h, p_0)$.
2: *solucion* ← archivo *KISS2IM* generado por FSMIM-Gen 1.2
3: Sea $r$ el número de canales de selección de $F$
4: $n_{\text{máx}} \leftarrow |P|$ ▷ *Número máximo de grupos.*
5: Sea $prof_{\text{máx}}$ y *tam* la profundidad y el tamaño de la ROM de bloques de $F$, respectivamente, calculados con el procedimiento DisBloq (algoritmo 14).
6: $n_{\text{gr}} \leftarrow \lceil \frac{prof_{\text{máx}}}{2^r} \rceil$ ▷ *Número de grupos que caben en la ROM*
7: *terminar* ← *falso*
8: **mientras** $|E| < n_{\text{gr}}$ **y** $|E| < n_{\text{máx}}$ **y** *terminar* = *falso* **hacer**
9: Ejecuta FSMIM-Gen 1.2 con el archivo *fsm* y la opción `ngroups` = mín$\{n_{\text{gr}}, n_{\text{máx}}\}$ para obtener la FSMIM $F$
10: $tam_{\text{anterior}} \leftarrow tam$
11: Sea $prof_{\text{máx}}$ y *tam* la profundidad y el tamaño de la ROM de bloques de $F$, respectivamente, calculados con el procedimiento DisBloq.
12: **si** $tam < tam_{\text{anterior}}$ **entonces**
13: *solucion* ← archivo *KISS2IM* generado por FSMIM-Gen 1.2
14: $n_{\text{gr}} \leftarrow \lceil \frac{prof_{\text{máx}}}{2^r} \rceil$ ▷ *Número de grupos que caben en la ROM*
15: **si no**
16: *terminar* ← *verdadero*
17: **fin si**
18: **fin mientras**
19: Devolver el archivo *solución.*

## H.2. FSMIM-Gen 2.0

FSMIM-Gen 2.0 se ha desarrollado con el fin de explorar todas las ideas surgidas en el contexto de este trabajo de investigación y de evaluar experimentalmente todos los algoritmos propuestos en esta memoria de tesis doctoral.

Se trata de una biblioteca de objetos y funciones desarrollados con el lenguaje de programación Python 2.7.6 [Python, 2014] y con SAGE [Sage, 2014]. SAGE es un entorno de programación para el cálculo matemático basado en una extensión de Python 2.7.6. El motor de optimización utilizado para la búsqueda de soluciones de los modelos PLE propuestos ha sido Gurobi 5.6 [Gurobi, 2015].

A pesar de que la denominación FSMIM-Gen 2.0 ha sido utilizada para hacer referencia exclusivamente a la aplicación que implementa las estrategias de optimización, esta incluye realmente a las siguientes herramientas:

- Herramienta para la generación de FSMIM. A partir una FSM descrita en el formato *KISS2* (apéndice B.1), la herramienta genera una FSMIM en formato *KISS2IM* (apéndice F) aplicando las nuevas estrategias de optimización propuestas en esta tesis (sección 5.1.1.2).

- Herramienta para la generación de descripciones VHDL de memorias ROM (apéndice G), que ha permitido generar las FSMIM con ajuste de profundidad por bloques (sección 5.1.2).

- Herramienta para la generación de descripciones VHDL de FSMIM a partir de descripciones en formato *KISS2IM*. Esta herramienta permite generar descripciones de las arquitecturas FSMIM-T y FSMIM-S. Ha sido utilizada tanto para las pruebas de implementación realizadas en esta tesis como para el estudio experimental realizado en [García-Vargas y Senhadji-Navarro, 2015].

- Herramienta para la generación de descripciones VHDL de FSM utilizando la arquitectura convencional basada en memoria. La descripción VHDL se crea a partir de una descripción en formato *KISS2*.

- *RandomFSMIM*: herramienta para la generación de máquinas de estado de prueba (apéndice C), utilizada para generar las FSM de la batería de pruebas BP-184.

Apéndice I

# Descripción VHDL de una FSM

Con objeto de implementar las arquitecturas convencionales de FSM y las arquitecturas proporcionadas por *ISE WebPACK*, se ha desarrollado una aplicación que genera dos tipos de descripciones VHDL de una FSM a partir de la especificación en formato *KISS2*: una descripción de comportamiento de la FSM y una descripción estructural de la arquitectura convencional basada en memoria (véase la sección 1.2.5).

## I.1. Descripción VHDL de comportamiento de una FSM

La descripción VHDL de comportamiento utilizada está basada en una plantilla que proporciona la herramienta *ISE WebPACK*. Esta descripción permite inferir una FSM a la herramienta de síntesis, por lo que puede ser utilizada para evaluar las dos arquitecturas de FSM incluidas en *ISE WebPACK*, es decir, la arquitectura basada en celdas lógicas y la arquitectura basada en memoria (denominadas en el capítulo 5 ISE-LUT e ISE-BLOQ, respectivamente).

Para ilustrar el formato de la descripción de comportamiento, se utilizará como ejemplo la FSM *mark1*, perteneciente a la batería de pruebas MCNC, cuya descripción *KISS2* se muestra en la figura I.1. La descripción VHDL generada por la aplicación se muestra en las figuras I.2 a I.6.

La figura I.2 contiene la descripción VHDL correspondiente a la declaración de la entidad y las declaraciones de tipos de la arquitectura. Como puede observarse en las líneas 17 a 32, los estados se declaran como un tipo

```
.i 5
.o 16
.p 22
.s 15
0---- *       state1  -11---1-00------
1---- state1  state3  -11---1-00------
1---- state2  state0  -11---1-00------
1---- state3  state4  101---1-01------
1-111 state4  state13 -11---1-00------
1-110 state4  state10 -11---1-00------
1-10- state4  state9  -11---1-00------
1-011 state4  state8  -11---1-00------
1-010 state4  state7  -11---1-00------
1-001 state4  state6  -11---1-00------
1-000 state4  state5  -11---1-00------
1---- state5  state14 0011--1-00------
1---- state6  state14 00100-0-00000011
1---- state7  state14 001---1100------
1---- state8  state14 010---1-00------
1---- state9  state14 001---1010000101
1---- state10 state11 -11---1-00100000
10--- state11 state13 -11---1-00------
11--- state11 state12 -11---1-00------
1---- state12 state13 -110110-00------
1---- state13 state14 -11---1-00------
1---- state14 state3  -110110-00------
.e
```

**Figura I.1:** Descripción en formato *KISS2* de la FSM *mark1*.

enumerado, dejando a la herramienta de síntesis la elección de la codificación de estados más conveniente.

La figura I.3 muestra el cuerpo de la arquitectura VHDL. Puede observarse que se divide en tres procesos que describen la parte síncrona de la FSM (líneas 4 a 15), la función de salida (líneas 17 a 20) y la función de estado siguiente (líneas 22 a 25). Los dos últimos procesos se mostrarán en detalle más adelante. Respecto a la descripción de la parte síncrona, puede observarse como las entradas y salidas de la FSM están registradas (líneas 7 y 8), esto nos permite medir la frecuencia de operación máxima.

La figura I.4 muestra el contenido del proceso que describe la función de salida. Al tratarse *mark1* de una FSM de Mealy, el proceso depende del estado y del valor de las entradas. En la figura I.5 se muestra el mismo proceso para la FSM de Moore *s1*, perteneciente a la batería de pruebas MCNC. En este caso puede observarse como la salida depende solamente del estado presente.

Finalmente, la figura I.6 muestra el contenido del proceso que describe la función de estado siguiente.

## I.2. Descripción VHDL estructural de la arquitectura convencional basada en memoria de una FSM

La figura I.7 muestra la descripción estructural de la FSM *mark1*, perteneciente a la batería de pruebas MCNC, utilizando la arquitectura convencional basada en memoria (sección 1.2.5).

La figura sólo muestra la entidad principal, que utiliza una ROM como componente (líneas 15 a 20) cuya descripción VHDL se genera mediante la herramienta ROMGEN (véase el apéndice G). Las entradas y salidas de la FSM están registradas para poder calcular la frecuencia de operación máxima (líneas 24 a 30). Finalmente, las líneas 28 y 33 describen la conexión de las señales de la ROM, y la línea 34, la instanciación del componente.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity mark1_clb is
7    port (
8      CLOCK: in std_logic;
9      RESET: in std_logic;
10     input : in std_logic_vector( 4 downto 0 );
11     output : out std_logic_vector( 15 downto 0 )
12     );
13 end mark1_clb;
14
15 architecture mark1_clb_arch of mark1_clb is
16
17       type state_type is (
18          s_state1,
19          s_state2,
20          s_state3,
21          s_state4,
22          s_state5,
23          s_state6,
24          s_state7,
25          s_state8,
26          s_state9,
27          s_state10,
28          s_state11,
29          s_state12,
30          s_state13,
31          s_state14,
32          s_state0
33     );
34     signal state, next_state : state_type;
35
36     signal output_i : std_logic_vector( 15 downto 0 );
37     signal input_i : std_logic_vector( 4 downto 0 );
38 begin
39  :
```

**Figura I.2:** Descripción VHDL de comportamiento de la FSM *mark1*: declaración de la entidad *mark1_clb* y de las señales internas.

```
 1  .
 2  .
 3  begin
 4
 5      SYNC_PROC: process (CLOCK, RESET)
 6      begin
 7          if (clock'event and clock = '1') then
 8              output <= output_i;
 9              input_i <= input;
10              if (reset='1') then
11                  state <= s_state1;
12              else
13                  state <= next_state;
14              end if;
15          end if;
16      end process;
17
18      OUTPUT_DECODE: process (state, input_i)
19      begin
20          .
21          .
22      end process;
23
24      NEXT_STATE_DECODE: process (state, input_i)
25      begin
26          .
27          .
28      end process;
29
30  end mark1_clb_arch;
```

**Figura I.3:** Descripción VHDL de comportamiento de la FSM *mark1*: cuerpo de la arquitectura.

```
 1    .
 2    .
 3    OUTPUT_DECODE: process (state, input_i)
 4    begin
 5      output_i <= (others => '-');
 6
 7      if (state = s_state1 and input_i(4)='1') then
 8          output_i <= "-11---1-00------";
 9      elsif (state = s_state2 and input_i(4)='1') then
10          output_i <= "-11---1-00------";
11      elsif ...
12          .
13          .
14      end if;
15    end process;
16    .
17    .
```

**Figura I.4:** Descripción VHDL de comportamiento de la FSM de Mealy *mark1*: función de salida.

```
1   .
    .
2   OUTPUT_DECODE: process (state)
3   begin
4       output_i <= (others => '-');
5       case (state) is
6           when s_st0 =>
7             output_i <= "000001";
8           when s_st1 =>
9             output_i <= "000011";
10          when s_st2 =>
11           .
             .
12      end case;
13  end process;
14  .
    .
```

**Figura I.5:** Descripción VHDL de comportamiento de la FSM de Moore *s1*: función de salida.

```
1   .
    .
2   NEXT_STATE_DECODE: process (state, input_i)
3   begin
4     next_state <= state;
5
6     case (state) is
7       when s_state1 =>
8         if input_i(4)='1' then
9           next_state <= s_state3;
10        elsif input_i(4)='0' then
11          next_state <= s_state1;
12        end if;
13      when s_state2 =>
14       .
         .
15      when s_state11 =>
16        if input_i(3)='0' and input_i(4)='1' then
17          next_state <= s_state13;
18        elsif input_i(3)='1' and input_i(4)='1' then
19          next_state <= s_state12;
20        elsif input_i(4)='0' then
21          next_state <= s_state1;
22        end if;
23         .
           .
24    end case;
25  end process;
26  .
    .
```

**Figura I.6:** Descripción VHDL de comportamiento de la FSM *mark1*: función de estado siguiente.

```
1  entity mark1 is
2      port (
3          clk: in std_logic;
4          reset: in std_logic;
5          input : in std_logic_vector( 4 downto 0 );
6          output : out std_logic_vector( 15 downto 0 )
7          );
8  end mark1;
9
10 architecture mark1_arch of mark1 is
11     signal in_aux: std_logic_vector( 4 downto 0 );
12     signal addr: std_logic_vector(8 downto 0);
13     signal data: std_logic_vector(19 downto 0);
14
15     component mark1_rom is
16         port( clk: in std_logic;
17               reset: in std_logic;
18               addr: in std_logic_vector(8 downto 0);
19               data: out std_logic_vector(19 downto 0));
20     end component;
21
22 begin
23     --Se registra la E/S de la FSM para evaluar el rendimiento
24     process (clk)
25     begin
26         if rising_edge(clk) then
27             in_aux <= input;
28             output <= data(15 downto 0);
29         end if;
30     end process;
31
32     --ROM
33     addr <= data(19 downto 15) & in_aux(4 downto 0);
34     Inst_mark1_rom: mark1_rom port map(clk, reset, addr, data);
35
36 end mark1_arch;
```

**Figura I.7:** Descripción VHDL estructural de la FSM *mark1* implementada con la arquitectura convencional basada en memoria: entidad superior.

Apéndice J

# Descripción VHDL de una FSMIM

## J.1. Descripción VHDL de una FSMIM-T

La descripción VHDL de una FSMIM-T se compone de tres entidades: la entidad principal, la entidad que modela el multiplexor y la entidad que modela la ROM.

Para describir la entidad principal se utilizará como ejemplo una FSMIM obtenida a partir de la FSM *pma*, perteneciente a la batería de pruebas MCNC. La figura J.1 muestra la descripción de la FSMIM en formato *KISS2IM*, mientras que la figura J.2 y la figura J.3 contienen la descripción VHDL de la implementación correspondiente a la arquitectura FSMIM-T.

Además de la declaración de las señales auxiliares, la figura J.2 contiene la declaración de la entidad superior (`pma`), del componente `pma_mux` (que modela un multiplexor genérico) y del componente `pma_rom` (que modela la ROM). Para describir el multiplexor se utiliza una descripción de comportamiento genérica similar a la mostrada en la figura G.7. Por otra parte, la descripción VHDL de la ROM se genera mediante la herramienta ROMGEN (véase el apéndice G).

La figura J.3 contiene el cuerpo de la arquitectura VHDL. Las señales de entrada y salida de la FSMIM se registran en las líneas 38 a 44 para que la herramienta de síntesis pueda calcular la frecuencia de operación máxima. La ROM se instancia en la línea 48 y la conexión de sus señales se realiza en las líneas 42 y 47. El banco de multiplexores de entradas se describe en las líneas 51 a 78, donde el *Multiplexor 0* selecciona las entradas de la primera columna de la MSE (la que aparece a la izquierda en la descripción en formato *KISS2IM*) y el *Multiplexor 5*, las de la última. Como puede

observarse, a pesar de que existen valores $\{0, 1\}$ en la MSE, no hay señales constantes conectadas a ninguno de los multiplexores. Esto se debe a que la selección de constantes se realiza utilizando el esquema mostrado en la figura 2.9e. Por ejemplo, en el *Multiplexor 0* la señal `data(4)` contiene el valor de la constante (línea 51).

## J.2. Descripción VHDL de una FSMIM-S

La descripción VHDL de la arquitectura FSMIM-S se compone de cuatro entidades: la entidad principal, la entidad que modela el banco de selectores de entradas, la entidad que modela el codificador de grupos y la entidad que modela la ROM.

Para ilustrar la descripción VHDL se utilizará el mismo ejemplo que en la sección anterior. La figura J.4 y la figura J.5 contienen la descripción VHDL de la entidad principal (`pma`). La primera de las figuras contiene la declaración de la entidad `pma`, de las señales internas, del componente `pma_insel` (que modela en banco de selectores de entradas), del componente `pma_grsel` (que modela el codificador de grupos) y del componente `pma_rom` (que modela la ROM). Al igual que en la arquitectura FSMIM-T, la descripción de la ROM se genera con la herramienta ROMGEN (apéndice G).

La figura J.5 muestra el cuerpo de la arquitectura VHDL. Al igual que en la descripción de la arquitectura FSMIM-T, las señales de entrada y salida de la FSMIM se registran para evaluar el rendimiento. La conexión de las señales de la ROM se realiza en las líneas 41 y 46. Como puede observarse, la señal `data(12 downto 8)`, que contiene el código del metaestado presente de la FSMIM, se conecta al banco de selectores de entradas y al codificador de grupos.

La figura J.6 muestra la descripción del banco de selectores de entradas. Las líneas 10 a 12 corresponden a las tres primeras filas de la MSE (véase la descripción *KISS2IM* de la figura J.1), y la línea 14, a la última fila.

Finalmente, la figura J.7 contiene la descripción del codificador de grupos.

```
.i 6
.o 8
.p 73
.s 8
.r g5
.ris 10

.is 10  0  0  i1 i6 -  i5
.is 11  0  0  i1 i8 i7 i4
.is 5   1  -  i1 i8 i7 i5
.is 1   1  i6 i1 i8 i7 i5
.is 2   0  i4 i1 i8 i7 i5
.is 3   i3 i2 i1 i8 i7 i5
.is 4   1  i4 i1 i8 i7 i5
.is 6   0  i6 i1 i8 i7 i5
.is 7   1  i6 i1 i8 i7 i5
.is 8   1  0  i1 -  0  -
.is 9   0  1  i1 i8 i7 i4
.is 12  0  0  i1 -  -  i4
.is 13  1  1  i1 i8 i7 i4
.is 15  0  1  i2 i8 i7 i5
.is 16  0  1  i1 i8 i7 -
.is 17  1  1  i1 -  0  i4
.is 18  0  0  i1 i8 i7 i4
.is 19  0  i2 i1 i8 i3 i5
.is 20  1  1  i1 i8 i4 i5
.is 21  1  1  i1 -  1  -
.is 22  1  0  i1 i8 i4 i5
.is 23  0  1  i1 i8 -  i5
.is 24  1  0  i1 -  1  -
.is 14  1  0  i1 i8 i7 i4

is10 00---1 g5 g3 is9 00000000
is10 0011-0 g5 g2 is5 00000000
is16 01110- g5 g5 is10 00100000
is16 011-1- g5 g4 is20 00100000
  ⋮
is4 1-1-1- g0 g6 is24 11101100
is4 1-11-- g0 g6 is24 11101100
.e
```

**Figura J.1:** Descripción de la FSM *pma* en formato *KISS2IM*.

```
1  entity pma is
2      port (
3          clk: in std_logic;
4          reset: in std_logic;
5          input : in std_logic_vector( 7 downto 0 );
6          output : out std_logic_vector(7 downto 0)
7          );
8  end pma;
9
10 architecture pma_arch of pma is
11     signal in_aux: std_logic_vector(7 downto 0);
12     signal data: std_logic_vector(21 downto 0);
13     signal addr: std_logic_vector(8 downto 0);
14     signal input_mux0 : std_logic_vector(1 downto 0);
15     signal input_mux1 : std_logic_vector(3 downto 0);
16     signal input_mux2 : std_logic_vector(1 downto 0);
17     signal input_mux3 : std_logic_vector(1 downto 0);
18     signal input_mux4 : std_logic_vector(3 downto 0);
19     signal input_mux5 : std_logic_vector(1 downto 0);
20     signal output_mux : std_logic_vector(5 downto 0);
21
22     component pma_mux is
23         port (
24             input : in std_logic_vector;
25             control : in std_logic_vector;
26             output : out std_logic );
27     end component;
28
29     component pma_rom is
30         port( clk: in std_logic;
31               reset: in std_logic;
32               addr: in std_logic_vector(9-1 downto 0);
33               data: out std_logic_vector(22-1 downto 0));
34     end component;
35
36 begin
```

**Figura J.2:** Descripción VHDL de la FSM *pma* implementada con la arquitectura FSMIM-T: declaración de la entidad superior *pma*, de los componentes y de las señales internas.

```
36 begin
37     --Se registra la E/S de la FSMIM para evaluar el rendimiento
38     process (clk)
39     begin
40         if rising_edge(clk) then
41             in_aux <= input;
42             output <= data(21 downto 14);
43         end if;
44     end process;
45
46     --ROM
47     addr <= data(2 downto 0) & output_mux(5 downto 0);
48     Inst_pma_rom: pma_rom port map(clk, reset, addr, data);
49
50     --Multiplexor 0
51     input_mux0 <= data(4) & in_aux(2);
52     muxins_mux0: pma_mux port map (input=>input_mux0,
53         control=>data(3), output=>output_mux(0));
54
55     --Multiplexor 1
56     input_mux1 <= data(7) & in_aux(1) & in_aux(3) & in_aux(5);
57     muxins_mux1: pma_mux port map (input=>input_mux1,
58         control=>data(6 downto 5), output=>output_mux(1));
59
60     --Multiplexor 2
61     input_mux2 <= in_aux(0) & in_aux(1);
62     muxins_mux2: pma_mux port map (input=>input_mux2,
63         control=>data(8), output=>output_mux(2));
64
65     --Multiplexor 3
66     input_mux3 <= in_aux(5) & in_aux(7);
67     muxins_mux3: pma_mux port map (input=>input_mux3,
68         control=>data(9), output=>output_mux(3));
69
70     --Multiplexor 4
71     input_mux4 <= data(12) & in_aux(2) & in_aux(3) & in_aux(6);
72     muxins_mux4: pma_mux port map (input=>input_mux4,
73         control=>data(11 downto 10), output=>output_mux(4));
74
75     --Multiplexor 5
76     input_mux5 <= in_aux(4) & in_aux(3);
77     muxins_mux5: pma_mux port map (input=>input_mux5,
78         control=>data(13), output=>output_mux(5));
79
80 end pma_arch;
```

**Figura J.3:** Descripción VHDL de la FSM *pma* implementada con la arquitectura FSMIM-T: instanciación de componentes y registro de las entradas y salidas de la FSM.

```
1  entity pma is
2      port (
3          clk: in std_logic;
4          reset : in std_logic;
5          input : in std_logic_vector (7 downto 0);
6          output : out std_logic_vector (7 downto 0) );
7  end pma;
8
9  architecture pma_arch of pma is
10     signal in_aux : std_logic_vector (7 downto 0);
11     signal data : std_logic_vector (12 downto 0);
12     signal addr : std_logic_vector (8 downto 0);
13     signal efective_input : std_logic_vector (5 downto 0);
14     signal groups : std_logic_vector (2 downto 0);
15
16     component pma_insel is
17         port (
18             sel : in std_logic_vector(4 downto 0);
19             input : in std_logic_vector(7 downto 0);
20             output : out std_logic_vector(5 downto 0) );
21     end component;
22
23     component pma_grsel is
24         port (
25             sel : in std_logic_vector(4 downto 0);
26             output : out std_logic_vector(2 downto 0) );
27     end component;
28
29     component pma_rom is
30         port( clk: in std_logic;
31               reset: in std_logic;
32               addr: in std_logic_vector(8 downto 0);
33               data: out std_logic_vector(12 downto 0));
34     end component;
35 begin
```

**Figura J.4:** Descripción VHDL de la FSM *pma* implementada con la arquitectura FSMIM-S: declaración de la entidad superior *pma*, de los componentes y de las señales internas.

```
35 begin
36     --Se registra la E/S de la FSMIM para evaluar el rendimiento
37     process (clk)
38     begin
39         if rising_edge(clk) then
40             in_aux <= input;
41             output <= data(7 downto 0);
42         end if;
43     end process;
44
45     --ROM
46     addr <= groups(2 downto 0) & efective_input(5 downto 0);
47     Inst_pma_rom: pma_rom port map(clk, reset, addr, data);
48
49     --Banco de selectores de entradas
50     Inst_pma_insel: pma_insel
51         port map (sel=>data(12 downto 8), input=>in_aux, output=>
       efective_input);
52
53     --Codificador de grupos
54     Inst_pma_grsel: pma_grsel
55         port map (sel=>data(12 downto 8), output=>groups);
56 end pma_arch;
```

**Figura J.5:** Descripción VHDL de la FSM *pma* implementada con la arquitectura FSMIM-S: instanciación de componentes y registro de las entradas y salidas de la FSM.

```
1  entity pma_insel is
2      port (
3          sel : in std_logic_vector(4 downto 0);
4          input : in std_logic_vector(7 downto 0);
5          output : out std_logic_vector(5 downto 0) );
6  end pma_insel;
7
8  architecture pma_insel_arch of pma_insel is
9  begin
10   output <=   '0' & '0' & input(0) & input(5) & '-' & input(4) when sel = "00000"
11     else '0' & '0' & input(0) & input(7) & input(6) & input(3) when sel = "00001"
12     else '1' & '-' & input(0) & input(7) & input(6) & input(4) when sel = "00010"
13      :
14     else '1' & '0' & input(0) & input(7) & input(6) & input(3) when sel = "10111"
15     else "-------";
16 end pma_insel_arch;
```

**Figura J.6:** Descripción VHDL de la FSM *pma* implementada con la arquitectura FSMIM-S: descripción del banco de selectores de entradas.

```
1  entity pma_grsel is
2      port (
3           sel  : in std_logic_vector(4 downto 0);
4           output : out std_logic_vector(2 downto 0) );
5  end pma_grsel;
6
7  architecture pma_grsel_arch of pma_grsel is
8  begin
9      output <= "000" when sel = "00000"
10         else "100" when sel = "00001"
11         else "101" when sel = "00010"
12          :
13         else "100"   when sel = "10111"
14         else "---";
15 end pma_grsel_arch;
```

**Figura J.7:** Descripción VHDL de la FSM *pma* implementada con la arquitectura FSMIM-S: descripción del codificador de grupos.

Apéndice K

# Implementación en FPGA de las descripciones VHDL

## K.1. Dispositivo FPGA

Las pruebas de implementación se han realizado con el dispositivo FPGA XC6SLX75T-4, de la familia de dispositivos Spartan-6 de Xilinx [Xilinx, 2011b]. Se trata del dispositivo que dispone de más recursos de todos los dispositivos de la familia Spartan-6 incluidos en *ISE WebPACK*[1].

La tabla K.1 muestra las características del dispositivo que son relevantes para el estudio experimental.

El *speed grade* es el índice utilizado por Xilinx para indicar la velocidad del dispositivo. Este valor, que está relacionado con los retrasos que impone la lógica del dispositivo, es relativo a los dispositivos de una misma familia.

**Tabla K.1:** Características relevantes del dispositivo XC6SLX75T-4

| *Speed grade* | Núm. *slices* | Núm. LUT | Núm. Bloques de memoria | Núm. *Flip-flops* |
|---|---|---|---|---|
| −4 | 11 662 | 46 648 | 172 | 93 296 |

[1]Debido a que *ISE WebPACK* es distribuido gratuitamente por Xilinx, incluye sólo un conjunto reducido de los dispositivos del fabricante.

En la familia Spartan-6 el *speed grade* varía entre el $-1$ y el $-4$, donde el valor más pequeño corresponde a los dispositivos más rápidos[2].

El recurso principal de los dispositivos de Xilinx para implementar circuitos combinacionales y secuenciales es el *Bloque Lógico Configurable* (CLB, sigla del inglés Configurable Logic Block) [Xilinx, 2011e]. Los CLB se organizan en una matriz bidimensional y se conectan mediante una red de interconexión, creando una arquitectura modelo *isla* [Hauck y DeHon, 2008].

Cada CLB se divide en dos *slices*, que están formados por cuatro LUT y ocho elementos de almacenamiento (*flip-flops*). Las LUT son pequeñas memorias asíncronas de 6 entradas (en el caso de la familia Spartan-6) que se utilizan para construir funciones booleanas. En la familia Spartan-6, el número de *slices* varía entre 600 y 23 038 (por lo tanto, el número de LUT varía entre 2 400 y 92 152).

Una característica importante de los dispositivos FPGA de Xilinx es la incorporación a los CLB de multiplexores empotrados. Se trata de multiplexores de dos entradas que pueden combinarse con las LUT de un CLB para formar cualquier función de siete u ocho entradas con un *slice*. Además, estos multiplexores empotrados permiten la implementación eficiente de multiplexores de hasta 16 entradas, lo que es de especial interés para la arquitectura FSMIM-T. Usando solamente los recursos de un CLB, los dispositivo Spartan-6 pueden implementar los siguientes multiplexores con un sólo nivel de lógica formado por LUT:

- Multiplexores 4:1 usando una LUT.
- Multiplexores 8:1 usando dos LUT y un multiplexor empotrado.
- Multiplexores 16:1 usando cuatro LUT y tres multiplexores empotrados.

Respecto a los bloques de memoria empotrados, la familia Spartan-6 dispone de bloques RAM de 18 Kbits, que pueden configurarse como dos bloques independientes de 9 Kbits o como un sólo bloque de 18 Kbits [Xilinx, 2011c]. La profundidad y el ancho de los bloques puede configurarse, dando lugar a diferentes geometrías de memoria. Cada bloque de 18 Kbits permite las siguientes configuraciones: $16\text{K} \times 1$, $8\text{K} \times 2$, $4\text{K} \times 4$, $2\text{K} \times 9$, $2\text{K} \times 8$, $1\text{K} \times 18$, $1\text{K} \times 16$, $512 \times 36$ y $512 \times 32$. Por su parte, cada bloque de 9 Kbits admite las siguientes configuraciones: $8\text{K} \times 1$, $4\text{K} \times 2$, $2\text{K} \times 4$, $1\text{K} \times 9$, $1\text{K} \times 8$, $512 \times 18$, $512 \times 16$, $256 \times 36$ y $256 \times 32$. Aunque se trata de bloques

[2] La fabricación de los dispositivos Spartan-6 con *speed grade* $-4$ fue interrumpida por Xilinx, siendo sustituido por el *speed grade* $-3$ [Xilinx, 2011a].

de memoria RAM, estos pueden implementar memoria ROM de uno o dos puertos (el uso de dos puertos permitiría, por ejemplo, la implementación de dos FSM iguales con una única memoria ROM). El número de bloques en los dispositivos de la familia Spartan-6 varía entre 12 y 298 bloques.

## K.2. Configuración del proceso de síntesis e implementación

La síntesis e implementación de las arquitecturas FSM en el dispositivo FPGA se ha realizado mediante el entorno integrado de diseño FPGA ISE® WebPACK versión 14.6 de Xilinx. Se trata de una versión gratuita y completamente funcional del entorno de diseño ISE® Design Suite que admite solamente un subconjunto de los dispositivos del fabricante.

La tabla K.2 muestra la configuración de *ISE WebPACK* utilizada para cada una de las arquitecturas estudiadas (la mayoría son las opciones por defecto de la herramienta).

Por otra parte, en la fase de implementación se ha establecido una restricción temporal sobre el periodo de reloj, que ha sido calculada a partir de la estimación de la frecuencia de operación máxima obtenida por el proceso de síntesis.

**Tabla K.2:** Opciones de *ISE WebPACK* empleadas para la síntesis e implementación de las arquitecturas estudiadas.

| **Opción** | **ISE-BLOQ** | **ISE-LUT** | **FSM-ROM** | **FSMIM** |
|---|---|---|---|---|
| Ignore Synthesis Constraint File | NO | | | |
| Output Format | NGC | | | |
| Target Device | xc6slx75t-4-fgg676 | | | |
| Automatic FSM Extraction | YES | YES | NO | NO |
| FSM Encoding Algorithm | Auto | | | |
| Safe Implementation | No | | | |
| FSM Style | Bram | LUT | Bram | Bram |
| RAM Extraction | Yes | | | |
| RAM Style | Block | Auto | Block | Block |
| ROM Extraction | Yes | | | |
| Shift Register Extraction | YES | | | |
| ROM Style | Block | Auto | Block | Block |
| Resource Sharing | YES | | | |
| Asynchronous To Synchronous | NO | | | |
| Shift Register Minimum Size | 2 | | | |
| Use DSP Block | Auto | | | |
| Automatic Register Balancing | No | | | |
| LUT Combining | Auto | | | |
| Reduce Control Sets | Auto | | | |
| Add IO Buffers | YES | | | |
| Global Maximum Fanout | 100000 | | | |
| Add Generic Clock Buffer(BUFG) | 16 | | | |
| Register Duplication | YES | | | |
| Optimize Instantiated Primitives | NO | | | |
| Use Clock Enable | Auto | | | |
| Use Synchronous Set | Auto | | | |
| Use Synchronous Reset | Auto | | | |
| Pack IO Registers into IOBs | Auto | | | |
| Equivalent register Removal | YES | | | |
| Optimization Goal | Area o Speed dependiendo de la prueba. | | | |
| Optimization Effort | 1 (corresponde al esfuerzo normal) | | | |
| Power Reduction | NO | | | |
| Keep Hierarchy | No | | | |
| Netlist Hierarchy | As_Optimized | | | |
| RTL Output | Yes | | | |
| Global Optimization | AllClockNets | | | |
| Read Cores | YES | | | |
| Write Timing Constraints | NO | | | |
| Cross Clock Analysis | NO | | | |
| Case Specifier | Maintain | | | |
| Slice Utilization Ratio | 100 | | | |
| BRAM Utilization Ratio | 100 | | | |
| DSP48 Utilization Ratio | 100 | | | |
| Auto BRAM Packing | NO | | | |
| Slice Utilization Ratio Delta | 5 | | | |

# Referencias


ACM/SIGDA (2012). The benchmark archives at CBL, Collaborative Benchmarking and Experimental Algorithmics Laboratory, North Carolina State University, USA. `http://www.cbl.ncsu.edu/benchmarks`.

Almaini, A., Miller, J., Thomson, P., y Billina, S. (1995). State assignment of finite state machines using a genetic algorithm. *Computers and Digital Techniques, IEE Proceedings* -, vol. 142, n.º 4, páginas 279 –286.

Altera (2010). Introduction to the Quartus II software - Altera (ver. 10.0). `http://www.altera.com`.

Altera (2015). Stratix V Device Handbook; Volume 1:Device Interfaces and Integration. `http://www.altera.com`.

Aly, W. M. (2009). Solving the state assignment problem using stochastic search aided with simulated annealing. vol. , 2, páginas 703–707.

Ashenden, P. J. (2001). *The Designerś Guide to VHDL (Second Edition)*. Morgan Kaufmann Publishers, San Francisco.

Ausiello, G., D'Atri, A., y Protasi, M. (1980). Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, vol. 21, n.º 1, páginas 136–153.

Avedillo, M., Quintana, J., y Huertas, J. (1990). New approach to the state reduction in incompletely specified sequential machines. En *Circuits and Systems, 1990., IEEE International Symposium on*, páginas 440–443 vol.1.

Avedillo, M., Quintana, J., y Huertas, J. (1994). State merging and state splitting via state assignment: a new fsm synthesis algorithm. *Computers and Digital Techniques, IEE Proceedings* -, vol. 141, n.º 4, páginas 229–237.

Baranov, S. I. (1994). *Logic Synthesis for Control Automata*. Kluwer Academic Publishers.

Barkalov, A. y Palagin, A. (1997). Synthesis of Microprogram Control Units. Kiev: IC NAC of Ukraine.

Barkalov, A. y Titarenko, L. (2009). *Logic synthesis for FSM-based control units (Lecture Notes in Electrical Engineering)*. Springer.

Barkalov, A., Titarenko, L., y Barkalov, A.A., J. (2012). Structural decomposition as a tool for the optimization of an fpga-based implementation of a mealy fsm. *Cybernetics and Systems Analysis*, vol. 48, n.º 2, páginas 313–322.

Bellman, R. (1954). The theory of dynamic programming. Technical report, DTIC Document.

Beyer, H. (1981). Tukey, john w.: Exploratory data analysis. addison-wesley publishing company reading, mass. — menlo park, cal., london, amsterdam, don mills, ontario, sydney 1977, xvi, 688 s. *Biometrical Journal*, vol. 23, n.º 4, páginas 413–414.

Borowik, G., Falkowski, B., y Luba, T. (2007). Cost-efficient synthesis for sequential circuits implemented using embedded memory blocks of FPGA's. En *Design and Diagnostics of Electronic Circuits and Systems, 2007. DDECS '07. IEEE*, páginas 1–6.

Borowik, G. y Luba, T. (2009). Decomposing pattern matching circuit. En Moreno-Díaz, R., Pichler, F., y Quesada-Arencibia, A., editors, *Computer Aided Systems Theory - EUROCAST 2009*, páginas 563–570. Springer-Verlag, Berlin, Heidelberg.

Bukowiec, A. (2005). Automata synthesis with application of multiple encoding. En *Proceedings of 2$^{nd}$ Scientific Conference Computer Science-Art or Craft KNWS*, volume 5, páginas 17–22.

Bukowiec, A. (2008). *Synthesis of Finite State Machines for Programmable devices based on multi-level implementation*. PhD thesis, Faculty

of Electrical Engineering, Computer Science and Telecommunications; University of Zielona Góra.

Chiu, G., Singh, D., Manohararajah, V., y Brown, S. (2006). Mapping arbitrary logic functions into synchronous embedded memories for area reduction on FPGAs. En *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, páginas 135 –142.

Chow, S.-H., Ho, Y.-C., Hwang, T., y Liu, C. L. (1996). Low power realization of finite state machines, a decomposition approach. vol. , 1, páginas 315–340.

Cong, J. y Yan, K. (2000). Synthesis for fpgas with embedded memory blocks. En *Proceedings of the 2000 ACM/SIGDA Eighth International Symposium on Field Programmable Gate Arrays*, FPGA '00, páginas 75–82, New York, NY, USA. ACM.

Cooke, P., Hao, L., y Stitt, G. (2015). Finite-state-machine overlay architectures for fast fpga compilation and application portability. *ACM Trans. Embed. Comput. Syst.*, vol. 14, n.º 3, páginas 54:1–54:25.

Dantzig, T. (2007). *Number: The language of science*. Penguin.

Das, J., Lam, A., Wilton, S., Leong, P., y Luk, W. (2011). An analytical model relating fpga architecture to logic density and depth. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, n.º 12, páginas 2229–2242.

Desai, M. P., Narayanan, H., y Patkar, S. B. (2003). The realization of finite state machines by decomposition and the principal lattice of partitions of a submodular function. vol. , 131, páginas 299–310.

Driesse, A., Harrison, S., y Jain, P. (2008). A finite state machine model to represent inverters in photovoltaic system simulations. En *Power Electronics Specialists Conference, 2008. PESC 2008. IEEE*, páginas 2568 –2573.

Duff, C. y Saucier, G. (1991). State assignment based on the reduced dependency theory and recent experimental results. En *Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers., 1991 IEEE International Conference on*, páginas 222–225.

Eaton, J. W., Bateman, D., y Hauberg, S. (2009). *GNU Octave version 3.0.1 manual: a high-level interactive language for numerical computations*. CreateSpace Independent Publishing Platform. `http://www.gnu.org/software/octave/doc/interpreter`.

El-Maleh, A., Sait, S., y Nawaz Khan, F. (2006). Finite state machine state assignment for area and power minimization. En *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, página 4.

Feske, K., Mulka, S., Koegst, M., y Elst, G. (1997). Technology-driven fsm partitioning for synthesis of large sequential circuits targeting lookup-table based FPGAs. En *Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications*, páginas 235–244, London, UK. Springer-Verlag.

Frigerio, L. y Salice, F. (2007). Ram-based fault tolerant state machines for FPGAs. En *Defect and Fault-Tolerance in VLSI Systems, 2007. DFT '07. 22nd IEEE International Symposium on*, páginas 312 –320.

Gao, R., Yu, J., Zhang, M., Tarn, T.-J., y Li, J.-S. (2010). Systems theoretic analysis of the central dogma of molecular biology: Some recent results. *NanoBioscience, IEEE Transactions on*, vol. 9, n.º 1, páginas 59 –70.

García-Vargas, I. (2006). Estudio de las técnicas de optimización de máquinas de estados finitas con multiplexión de entradas y su implementación en dispositivos FPGA. Memoria del Diploma de Estudios Avanzados, Universidad de Sevilla.

García-Vargas, I. y Senhadji-Navarro, R. (2012). The Minimum Maximal k-Partial-Matching problem. *Optimization Letters,* diciembre 2013, vol. 7, n.º 8, páginas 1959–1968. Publicado *online* en 2012.

García-Vargas, I. y Senhadji-Navarro, R. (2015). Finite State Machines with Input Multiplexing: A performance study. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 34, n.º 5, páginas 867–871.

García-Vargas, I., Senhadji-Navarro, R., Jimenez-Moreno, G., Civit-Balcells, A., y Guerra-Gutierrez, P. (2007). ROM-based Finite State Machine Implementation in Low Cost FPGAs. En *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, páginas 2342–2347.

García-Vargas, I., Senhadji-Navarro, R., Jiménez-Moreno, G., y Civit-Ballcels, A. (2005). Máquinas de Estados Finitos con Multiplexión de Entradas: Implementación sobre bloques RAM en FPGA. En *V Jornadas sobre Computación Reconfigurable y Aplicaciones (JCRA'2005)*, páginas 369–374.

Govindarajalu, B. (2010). *Computer Architecture and Organization: Design Principles and Application*. McGraw Hill.

Greiner, A. y Pcheux, F. (1992). Alliance: A complete set of cad tools for teaching vlsi design. En *3rd Eurochip Workshop on VLSI Design Training*, páginas 230–237.

Grotker, T. (2002). *System Design with SystemC*. Kluwer Academic Publishers, Norwell, MA, USA.

Grzes, T. y Solov'ev, V. (2014). Sequential algorithm for low-power encoding internal states of finite state machines. *Journal of Computer and Systems Sciences International*, vol. 53, n.º 1, páginas 92–99.

Gurobi (2015). Gurobi optimizer reference manual. Gurobi Optimization, Inc. `http://www.gurobi.com`.

Harvey, N. J., Ladner, R. E., Lovász, L., y Tamir, T. (2006). Semi-matchings for bipartite graphs and load balancing. *Journal of Algorithms*, vol. 59, n.º 1, páginas 53–78.

Hauck, S. y DeHon, A., editors (2008). *Reconfigurable computing the theory and practice of FPGA-Based computation*. Morgan Kaufmann.

Hintze, J. L. y Nelson, R. D. (1998). Violin plots: A box plot-density trace synergism. *The American Statistician*, vol. 52, n.º 2, páginas 181–184.

Huang, T.-C. (2011). High-yield performance-efficient redundancy analysis for 2d memory. vol. , 54, páginas 1663–1676. 10.1007/s11432-011-4357-x.

Janarthanan, A., Tiwari, A., y Tomko, K. (2007). Power-efficient fsm mapping in FPGAs through semb dormancy control. En *Circuits and Systems, 2007. MWSCAS 2007. 50th Midwest Symposium on*, páginas 502–505.

Jezernik, K., Horvat, R., y Harnik, J. (2012). Finite-state machine motion controller: Servo drives. *Industrial Electronics Magazine, IEEE*, vol. 6, n.º 3, páginas 13–23.

Jhonson, M., Khudanpur, S. P., Ostendorf, M., y Rosenfeld, R., editors (2004). *Mathematical foundations of speech and language processing*. Springer-Verlag New York, Inc.

Józwiak, L., Gawlowski, D., y Slusarczyk, A. (2004). An effective solution of benchmarking problem: Fsm benchmark generator and its application to analysis of state assignment methods. En *Digital System Design, 2004. DSD 2004. Euromicro Symposium on*, páginas 160 – 167.

Kamat, R. K., Shinde, S. A., y Shelake, V. G. (2009). *Unleash the System On Chip using FPGAs and Handel C*. Springer Publishing Company, Incorporated, 1st edition.

Karp, R. (1972). Reducibility among combinatorial problems. En Miller, R., Thatcher, J., y Bohlinger, J., editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, páginas 85–103. Springer US.

Katz, R. H. (1994). *Contemporary logic design*. Benjamin/Cummings (Redwood City, Calif.).

Kellerer, H., Pferschy, U., y Pisinger, D. (2004a). Introduction to np-completeness of knapsack problems. En *Knapsack Problems*, páginas 483–493. Springer Berlin Heidelberg.

Kellerer, H., Pferschy, U., y Pisinger, D. (2004b). Knapsack problems.

Kubátová, H. (2005). Finite state machine implementation in FPGAs. En *Design of Embedded Control Systems*, páginas 175–184. Springer US.

Le Gal, B., Ribon, A., Bossuet, L., y Dallet, D. (2010). Reducing and smoothing power consumption of rom-based controller implementations. En *Proceedings of the 23rd Symposium on Integrated Circuits and System Design*, SBCCI '10, páginas 8–13, New York, NY, USA. ACM.

Lisanke, B. (1988). Logic synthesis and optimization benchmarks; Technical report, MCNC. Research Triangle Park, North Carolina.

Mathews, G. (1896). On the partition of numbers. *Proceedings of the London Mathematical Society*, vol. 1, n.º 1, páginas 486–490.

McElvain, K. (1993). IWLS'93 Benchmark Set: Version 4.0. Distributed as a part of the ILWS'93 benchmark set.

Palnitkar, S. (2003). *Verilog HDL A Guide to Digital Design and Synthesis (2nd Edition)*. Prentice Hall.

Pferschy, U. y Schauer, J. (2009). The knapsack problem with conflict graphs. *J. Graph Algorithms Appl.*, vol. 13, n.º 2, páginas 233–249.

Pfleeger, C. (1973). State reduction in incompletely specified finite-state machines. *Computers, IEEE Transactions on*, vol. C-22, n.º 12, páginas 1099–1102.

Python (2014). Language Python Language Reference, version 2.7.6. Python Software Foundation. `http://www.python.org`.

Rafla, N. I. y Davis, B. L. (2006). A study of finite state machine coding styles for implementation in FPGAs. En *Circuits and Systems, 2006. MWSCAS '06. 49th IEEE International Midwest Symposium on*, volume 1, páginas 337 –341.

Rawski, M., Selvaraj, H., y Luba, T. (2003). An application of functional decomposition in rom-based fsm implementation in fpga devices. En *Digital System Design, 2003. Proceedings. Euromicro Symposium on*, páginas 104–110.

Rawski, M., Selvaraj, H., y Luba, T. (2005). An application of functional decomposition in rom-based fsm implementation in FPGA devices. vol. , 51, páginas 424–434.

Rawski, M., Tomaszewicz, P., Borowik, G., y Łuba, T. (2011). 5 logic synthesis method of digital circuits designed for implementation with embedded memory blocks of fpgas. En Adamski, M., Barkalov, A., y Węgrzyn, M., editors, *Design of Digital Systems and Devices*, volume 79 of *Lecture Notes in Electrical Engineering*, páginas 121–144. Springer Berlin Heidelberg.

Rudell, R. L. y Sangiovanni-Vincentelli, A. L. (1987). Multiple-valued minimization for pla optimization. vol. , 6, páginas 727–750.

Sage (2014). Sage Mathematics Software (Version 6.6.1). The Sage Developers. `http://www.sagemath.org`.

Selvaraj, H., Rawski, M., y Luba, T. (2002). Fsm implementation in embedded memory blocks of programmable logic devices using functional decomposition. En *Information Technology: Coding and Computing, 2002. Proceedings. International Conference on*, páginas 355–360.

Senhadji-Navaro, R. y Garcia-Vargas, I. (2015a). High-speed and area-efficient reconfigurable multiplexer bank for ram-based finite state machine implementations. vol. , página 1550101.

Senhadji-Navaro, R. y Garcia-Vargas, I. (2015b). High-speed and area-efficient reconfigurable multiplexer bank for ram-based finite state machine implementations. *Journal of Circuits, Systems and Computers*, vol. 24, n.º 07, página 1550101.

Senhadji-Navarro, R., García-Vargas, I., y Guisado, J. (2012). Performance evaluation of ram-based implementation of finite state machines in FPGAs. En *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on*, páginas 225–228.

Senhadji-Navarro, R., García-Vargas, I., Jimenez-Moreno, G., y Civit-Balcells, A. (2007). FPGA-based implementation of ram with asymmetric port widths for run-time reconfiguration. En *Electronics, Circuits and Systems, 2007. ICECS 2007. 14th IEEE International Conference on*, páginas 178 –181.

Senhadji-Navarro, R., García-Vargas, I., Jimenez-Moreno, G., y Civit-Ballcels, A. (2004). ROM-based FSM implementation using input multiplexing in FPGA devices. *Electronics Letters*, vol. 40, n.º 20, páginas 1249–1251.

Serhan, H., Henaff, P., Nasr, C., y Ouezdou, F. (2008). State machine-based controller for walk-halt-walk transitions on a biped robot. En *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, páginas 535 –540.

Shihming Liu, Massoud Pedram, A. M. D. (1995). A fast state assignment procedure for large fsms. En *Design Automation, 1995. DAC '95. 32nd Conference on*, páginas 327–332.

Sklyarov, V. (2000). Synthesis and implementation of ram-based finite state machines in FPGAs. En Hartenstein, R. y Gruenbacher, H., editors, *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*, volume 1896 of *Lecture Notes in Computer Science*, páginas 718–727. Springer Berlin / Heidelberg.

Sklyarov, V. (2002a). An evolutionary algorithm for the synthesis of ram-based fsms. En Hendtlass, T. y Ali, M., editors, *Developments in Ap-*

*plied Artificial Intelligence*, volume 2358 of *Lecture Notes in Computer Science*, páginas 108–118. Springer Berlin Heidelberg.

Sklyarov, V. (2002b). Reconfigurable models of finite state machines and their implementation in FPGAs. vol. , 47, páginas 1043–1064.

Sutterlin, T., Huber, S., Dickhaus, H., y Grabe, N. (2009). Modeling multicellular behavior in epidermal tissue homeostasis via finite state machines in multi-agent systems. *Bioinformatics*, vol. 25, n.º 16, páginas 2057–2063.

Tiwari, A. y Tomko, K. (2004). Saving power by mapping finite-state machines into embedded memory blocks in FPGAs. En *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 2, páginas 916–921 Vol.2.

Uma, R. y Dhavachelvan, P. (2012). Finite state machine optimization in FPGAs. En *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, CCSEIT '12, páginas 205–211, New York, NY, USA. ACM.

Villa, T. y Sangiovanni-Vincentelli, A. (1990). Nova: state assignment of finite state machines for optimal two-level logic implementation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 9, n.º 9, páginas 905–924.

Xilinx (2005). Using dedicated multiplexers in spartan-3 generation FPGAs. Application Note: Spartan-3 FPGA Series. `http://www.xilinx.com`.

Xilinx (2007). Ise in-depth tutorial (ver. 9.1). `http://www.xilinx.com`.

Xilinx (2008). Development system reference guide 10.1. `http://www.xilinx.com`.

Xilinx (2011a). Product discontinuation notice for Spartan-6 LXT -4 devices. `http://www.xilinx.com`.

Xilinx (2011b). Spartan-6 family overview. `http://www.xilinx.com`.

Xilinx (2011c). Spartan-6 FPGA block RAM resources. `http://www.xilinx.com`.

Xilinx (2011d). Spartan-6 fpga block ram resources. user guide (ver. 1.5). `http://www.xilinx.com`.

Xilinx (2011e). Spartan-6 FPGA configurable logic block. `http://www.xilinx.com`.

Xilinx (2012). ISim user guide. `http://www.xilinx.com`.

Xilinx (2013). XST User Guide for Virtex-6, Spartan-6, and 7 Series Devices (v 14.5). `http://www.xilinx.com`.

Xilinx (2014). Virtex-6 fpga memory resources. user guide (ver. 1.8). `http://www.xilinx.com`.

Yang, S. (1991). Logic synthesis and optimization benchmarks user guide. version 3.0.