Instituto de Microelectrónica de Sevilla (Universidad de Sevilla y CNM-CSIC)

DISEÑO Y CARACTERIZACIÓN DE CRIPTOCIRCUITOS SEGUROS Y RESISTENTES A ATAQUES FÍSICOS

(DESIGN AND CHARACTERIZATION OF SECURE AND RESILENT CRYPTOCIRCUITS AGAINST SIDE CHANNEL ATTACKS)

Memoria presentada por:
Erica Tena Sánchez
para optar al título de Doctora por la Universidad de Sevilla

Sevilla, 2019

Director de Tesis
Antonio José Acosta Jiménez
Lugar y fecha
Sevilla, 2019

A mis padres,

Agradecimientos

Han sido muchas las personas que me han acompañado durante este viaje, pero en primer lugar, me gustaría comenzar agradeciendo a mi director, Antonio José Acosta Jiménez, el haberme dado la oportunidad de realizar esta Tesis y el ofrecerme todo su apoyo y experiencia durante estos años. Vamos equipo!!

Me gustaría hacer un agradecimiento especial a José Miguel Mora, por toda su ayuda, dedicación desinteresada y apoyo, sobre todo durante mi última etapa de la Tesis. Estoy segura de que el resultado final de la Tesis no hubiera sido el mismo sin su colaboración.

A Javier Castro por guiarme al inicio de este camino y por todas las horas dedicadas para poder poner en marcha esta Tesis.

To Ricardo Chaves for his time, dedication, and for accepting me in Lisbon during my stay.

A Juan Núñez por su colaboración en parte del Capítulo 4 y por su apoyo con cualquier duda que me ha surgido a lo largo de estos años.

A Piedad Brox, por darme la oportunidad de conocer el IMSE y por su ayuda y amistad durante todos estos años.

A todos mis compañeros del IMSE, que son muchos después de estos años, y que me resultaría imposible enumerarlos a todos. Muchas gracias a todos.

A mi familia.

A los siguientes proyectos y entidades por su soporte y financiación:

- **MOBY-DIC**: Model-based synthesis of digital electronic circuits for embedded control (FP7-ICT-2009-4-248858), Comisión Europea.

- **CITIES**: Circuitos Integrados para Transmisión de Información Especialmente Segura (TEC2010-16870), Ministerio de Ciencia e Innovación.
- **CESAR**: Circuitos Microelectrónicos Seguros Frente a Ataques Laterales (TEC2013-45523-R), Ministerio de Economía y competitividad.
- **INTERVALO**: Integración y validación en laboratorio de contramedidas frente a ataques laterales en criptocircuitos microelectrónicos (TEC2016-80549-R), Ministerio de Economía y Competitividad.
- **CRIPTO-BIO**: Diseño microelectrónico para autenticación cripto-biométrica (P08-TIC-03674), Junta de Andalucía.
- **MISAL**: Microelectrónica segura frente a ataques laterales (201450E034), Consejo Superior de Investigaciones Científicas (CSIC).
- **LACRE**: LAboratorio de Criptoanálisis HardwarE (201550E039), Consejo Superior de Investigaciones Científicas (CSIC).
- **V Plan Propio de Investigación** de la Universidad de Sevilla.
- **CRYPTACUS**: Cryptanalysis in ubiquitous computing systems, COST ACTION IC1403, EU Framework Programme Horizon 2020.

Resumen

A diario personas de todo el mundo hacen uso de dispositivos electrónicos en los que almacenan o con los que intercambian información privada. La confidencialidad y privacidad es un derecho frente a posibles intrusos, por lo que la seguridad en las nuevas tecnologías es un factor de transcendental importancia que exige la atención de la comunidad científica.

Los dispositivos electrónicos considerados "seguros", de facto cualquier dispositivo electrónico de uso en telecomunicaciones o que maneje información relevante, hacen uso de la criptografía para garantizar la confidencialidad, autenticación e integridad de los datos procesados. Estos dispositivos criptográficos implementan algoritmos matemáticamente seguros, pero que, debido a su implementación física, pueden revelar información sensible por las fugas de información durante su operación, que pueden ser aprovechadas por un atacante para revelar la clave secreta del dispositivo. Estos ataques, conocidos como ataques de canal lateral, o simplemente ataques laterales, son muy efectivos y explotan información como puede ser el consumo de potencia, emisión electromagnética o tiempos de ejecución, entre otros, para revelar la clave secreta.

La comunidad científica ha centrado su esfuerzo en el diseño de contramedidas para evitar este tipo de ataques. El objetivo principal de esta Tesis es aumentar la seguridad de dispositivos criptográficos hardware frente ataques laterales. Para conseguir este objetivo se han realizado las 3 siguientes macro tareas:

1. Medidas de vulnerabilidad de un dispositivo criptográfico (realización de ataques y métricas de seguridad).
2. Propuestas de contramedidas.
3. Evaluación de su seguridad.

Para la medida de vulnerabilidad de los dispositivos criptográficos, se han implementado tanto ataques basados en el consumo de potencia, como ataques electromagnéticos o el uso de otras métricas como por ejemplo el t-test. Para probar la efectividad de los ataques, se han realizado sobre sistemas de clave privada, utilizándose como demostradores cifradores de bloque (AES y una parte del algoritmo KASUMI) y cifradores de flujo (Trivium). Estas medidas se han realizado tanto en entornos de simulación como de forma experimental, sobre implementaciones ASIC o FPGA. Además, se han evaluado diferentes métricas y test alternativos para poder evaluar la seguridad en diferentes etapas de diseño, así como el poder determinar el nivel de seguridad sin tener que llevar a cabo un ataque completo.

Por otra parte, se han propuesto diferentes metodologías de diseño de contramedidas frente ataques laterales aplicadas a diferentes niveles de abstracción. Las propuestas a nivel de transistor consisten en modificar las estructuras de las celdas lógicas diseñadas para poder obtener un consumo de potencia igual independientemente del dato procesado. A nivel de puerta se proponen diferentes técnicas que varían la temporización del circuito, modificando así los niveles de seguridad alcanzados por los criptocircuitos diseñados. Estas contramedidas, son complementarias y por tanto ambas aplicables en un mismo diseño.

Finalmente, cumplidas las dos tareas anteriores, se ha pasado a una etapa de diseño donde se han integrado en un ASIC los casos de estudio que implementan bloques criptográficos aplicando las contramedidas propuestas a lo largo del desarrollo de la Tesis. La caracterización de los diferentes casos de estudio determinará de forma experimental la ganancia en seguridad obtenida por cada contramedida.

Abstract

Every day, people all over the world use electronic devices to store or exchange private information with each other. Confidentiality and privacy is a right against possible intruders, so security in new technologies is an important factor that requires the attention of the scientific community.

Electronic devices considered "secure", in fact any electronic device for use in telecommunications or handling relevant information, make use of cryptography to ensure the confidentiality, authentication and integrity of the data processed. These cryptographic devices implement mathematically secure algorithms, but due to their physical implementation, they can reveal sensitive information due to data leaks during their normal operation, which can be exploited by an attacker to reveal the device's secret key. These attacks, known as side channel attacks, are very effective and exploit information such as power consumption, electromagnetic radiation or timing, among others, to reveal the secret key.

The scientific community has focused its efforts on the design of countermeasures to prevent this type of attack. The main objective of this Thesis is to increase the security of hardware cryptographic devices against side channel attacks. To achieve this objective, the following 3 tasks have been carried out:

1. Vulnerability measurements of a cryptographic device (execution of attacks and security metrics).

2. Proposals for countermeasures.

3. Security assessment.

To measure the vulnerability of cryptographic devices, attacks based on power consumption, electromagnetic attacks or the use of other metrics such as t-test have been implemented. To test the effectiveness of the attacks, they have been performed on private key systems, using block cipher demonstrators (AES and a part of

the KASUMI algorithm) and stream ciphers (Trivium). These measurements have been carried out both in simulation environments and experimentally on ASIC or FPGA implementations. In addition, different alternative metrics and tests have been evaluated in order to evaluate security at different stages of design, as well as to determine the level of security without having to carry out a complete attack.

On the other hand, different methodologies have been proposed for the design of countermeasures against side channel attacks applied at different levels of abstraction. The proposals at the transistor level consist of modifying the structures of the designed logic cells to obtain an equal power consumption independently of the processed data. At the gate level, different techniques are proposed that vary the timing of the circuit, thus modifying the security levels achieved by the designed cryptocircuits. These countermeasures are complementary and therefore both applicable in the same design.

Finally, once the two previous tasks had been completed, a design stage has been undertaken where the case studies implementing cryptographic blocks have been integrated into an ASIC, applying the countermeasures proposed throughout the development of the Thesis. The characterization of the different case studies will experimentally determine the security gain obtained by each countermeasure.

Introducción, motivación y estructura del documento

El uso de dispositivos electrónicos en los que se almacenan o con los que se intercambia información privada está ampliamente extendido, yendo desde tarjetas bancarias, teléfono móvil, comercio electrónico, televisión de pago, hasta el control de acceso (sistemas de cierre de coches, zonas restringidas, etc.). El mantener información segura en privado frente a posibles intrusos es un derecho, y por ello la seguridad en las nuevas tecnologías se está convirtiendo en un factor de transcendental importancia que está requiriendo con mayor frecuencia la atención de la comunidad científica internacional. La criptografía es el medio para establecer una comunicación e intercambio de información seguros, donde la confidencialidad, autenticación e integridad del mensaje estén garantizados.

Debido al gran aumento de los dispositivos portátiles y con la paulatina implantación de Internet de las Cosas (*Internet of Things*, IoT), las restricciones de los bloques criptográficos son cada vez mayores, exigiendo menor consumo y área para incluirlos en dispositivos con recursos extremadamente limitados. Por este motivo, la implementación hardware de los bloques criptográficos es la tendencia que se está observando en estos últimos años, donde el área y consumo se minimizan obteniendo unas prestaciones excelentes. Es por ello que la implementación de hardware específico para aplicaciones criptográficas es un foco de gran interés.

Si realizamos una búsqueda de la palabra "cryptography" en *Google Académico*, obtenemos la friolera de 812.000 documentos, dando como resultado 323.000 documentos si especificamos la búsqueda a "*hardware cryptography*". La misma búsqueda en la IEEE-Xplore da como resultado 48.817 documentos para la palabra "cryptography", mientras que para "hardware cryptography" obtenemos 6.425[1]. Esto

[1]Datos obtenidos a fecha de Julio 2018.

demuestra el gran interés en el campo de investigación en el que está enfocada esta Tesis, donde no sólo la criptografía en general es de gran interés, sino que la implementación hardware de estos dispositivos se ha convertido en un objetivo de máxima prioridad.

Esta Tesis se ha realizado en el marco de diferentes proyectos de investigación del grupo TIC-180, donde el principal objetivo común es el diseño y evaluación de la seguridad de circuitos criptográficos. La vinculación de la realización de esta Tesis con el Instituto de Microelectrónica de Sevilla y con la experiencia del grupo investigador en el campo de la microelectrónica para la seguridad, hace que los conocimientos que podamos aportar al estudio de implementación de hardware criptográfico sea de tremendo interés.

Por ese motivo, se inició esta Tesis con el principal objetivo de diseñar criptocircuitos microelectrónicos seguros y determinar los niveles de seguridad alcanzados por cada propuesta.

Inicialmente, la seguridad de los sistemas criptográficos venía determinada por la robustez del algoritmo y la longitud de la clave utilizada en la encriptación. Un algoritmo se consideraba seguro si era inviable en términos de coste o tiempo la obtención de la clave secreta tras un ataque por fuerza bruta. Sin embargo, a finales de los años 90 P. C. Kocher demostró que la clave de un dispositivo criptográfico podría ser revelada si se observaban características como el consumo de potencia o los tiempos de ejecución del algoritmo encriptador durante su ejecución. Estos ataques, conocidos como ataques de canal lateral, recibieron mucha atención debido a su gran efectividad y reducido coste en equipación y tiempo. A raíz de este trabajo se empezaron a presentar numerosas contramedidas para evitar este tipo de ataques.

Esta Tesis pretende cubrir el estudio de implementaciones hardware de contramedidas a diferentes niveles de abstracción para dificultar los ataques de canal lateral y evaluar la seguridad alcanzada. Para poder conseguir tal fin, se han desarrollado diferentes tareas, cuyo desarrollo se ha incluido en esta Tesis, dividida en 6 capítulos en los que se persiguen diferentes objetivos. La estructura del documento se ha dividido de la siguiente forma.

En el Capítulo 1 se hace una breve introducción a la criptografía, describiendo los conceptos básicos para la comprensión del resto de la Tesis.

En el Capítulo 2 se hace un estudio de los diferentes ataques laterales y diferentes medidas para evaluar la seguridad de los criptocircuitos. Se han realizado diferentes experimentos donde se muestra la efectividad de los ataques laterales y se proponen nuevas técnicas de ataque donde se explota información obtenida a partir de la implementación física del dispositivo. De esta forma, conseguimos tener las herramientas necesarias para la evaluación de la seguridad de diferentes propuestas, así como determinar cuáles podrían ser las vulnerabilidades para poder aplicar contramedidas efectivas para evitar estos ataques laterales.

En el Capítulo 3 se presentan diferentes técnicas para mejorar la seguridad de los criptocircuitos a nivel de transistor. Para evitar los ataques basados en el consumo de potencia, se han propuesto diferentes metodologías de diseño de celdas digitales donde se consigue obtener el mismo consumo de potencia independientemente del dato procesado. Además, se evalúan diferentes estructuras para poder reducir en la medida de lo posible el consumo de potencia y área sin llegar a perjudicar en exceso los niveles de seguridad alcanzados para poder tener un mejor compromiso entre seguridad y prestaciones.

En el Capítulo 4 se presentan contramedidas a nivel lógico o de puerta, donde se realiza un estudio sobre la influencia de la implementación lógica del circuito en la temporización del mismo, su impacto en la seguridad y el grado de consecución de un objetivo deseable de maximización simultánea de la seguridad y las prestaciones del criptocircuito.

Tras evaluar las diferentes contramedidas propuestas, en el Capítulo 5 se muestra el diseño e integración en un ASIC (*Application Specific Integrated Circuit*) de diferentes demostradores para evaluar de forma experimental el nivel de seguridad obtenido por las propuestas frente a ataques de canal lateral.

Finalmente, en el Capítulo 6 se muestran las conclusiones de esta Tesis y el trabajo futuro.

El conjunto de apéndices (A-E) incluye información relevante para la comprensión de la Tesis.

Índice general

Índice de figuras

Índice de tablas

Algoritmos

Lista de Acrónimos

AES	Advanced Encryption Standard
ASIC	Application Specific Integrated Circuit
BCDL	Balanced Cell-based Differential Logic
CAD	Computer-Aided Design
CEMA	Correlation Electromagnetic Analysis
CMOS	Complementary Metal-Oxide-Semiconductor
CPA	Correlation Power Analysis
CRT	Chinese Remainder Theorem
DDCVSL	Dynamic Differential Cascode Voltage Switch Logic
DDPL	Delay-based Dual-rail Precharge Logic
DEMA	Differential Electromagnetic Analysis
DES	Data Encryption Standard
DPA	Differential Power Analysis
DPDN	Differential Pull Down Network
DPL	Dual-rail Precharge Logic
DPUN	Differential Pull Up Network

DRC	Design Rule Check
DyCML	Dynamic Current Model Logic
ECC	Elliptic Curve Cryptography
EDA	Electronic Design Automation
EM	Electromagnetic
FA	Fault-injection Attack
FPGA	Field Programmable Gate Array
HD	Hamming-Distance
HDL	Hardware Description Language
HW	Hamming-Weigth
iMDPL	improved Masked Dual-rail Precharged Logic
Iot	Internet of Things
IV	Initialization Vector
K	Key
LDPA	Leakage-based Differential Power Analysis
LEF	Library Exchange Format
LSCML	Low-Swing Current Mode Logic
LVS	Layout Vs Schematic
MD	Message Digest
MDPL	Masked Dual-rail Precharged Logic

MTD	Measurements To Disclose
NED	Normalized Energy Deviation
NIST	National Institute of Standards and Technology
NLFSR	Non-Linear Feedback Shift Register
NMOS	Negative-channel Metal-Oxide Semiconductor
NSA	National Security Agency
NSD	Normalized Standard Deviation
OTP	One-Time Pad
PA	Power Analysis
PCB	Printed Circuit Board
PDP	Power-Delay Product
PKC	Public Key Cryptography
PMOS	Positive-channel Metal-Oxide Semiconductor
PTFS	Precision Temperature Forcing System
RSA	Rivest, Shamir and Adleman, algoritmo criptográfico
RTL	Register Transfer Level
SABL	Sense Amplifier Based Logic
SAKURA-G	Side-Channel Attack User Reference Architecture
SCA	Side-Channel Attack
SEMA	Simple Electromagnetic Analysis

SHA	Secure Hash Algorithm
SKC	Secret Key Cryptography
SPA	Simple Power Analysis
SPD	Security-Power-Delay
STTL	Secure Triple Track Logic
TDPL	Three-phased Dual-rail Precharge Logic
TSMC	Taiwan Semiconductor Manufacturing Company Limited
WDDL	Wave Dynamic Differential Logic

1. Introducción al hardware criptográfico

Desde el origen de nuestra civilización se han utilizado escrituras secretas para ocultar informaciones militares, políticas o religiosas a terceros. La criptografía surgió para que aun teniendo el texto accesible, la información fuera ilegible para todo aquel que no fuera el destinatario del mensaje. Esta ha ido evolucionando con el paso de los años, desde elementos más básicos, como pueden ser las tablas de sustitución, dando paso a sistemas más complejos como la máquina Enigma utilizada por los alemanes durante la II Guerra Mundial. Pero el mayor de los saltos en la evolución de la criptografía comienza con la aparición de la informática y sobre todo con Internet y la ciberseguridad.

Hoy en día, la ciberseguridad juega un papel clave en la vida cotidiana. La criptografía puede encontrarse desde el uso de tarjetas bancarias, *smartphones*, transacciones electrónicas, controles de acceso a diferentes tipos de instalaciones, áreas restringidas o incluso vehículos, y se está volviendo más y más imprescindible a medida que la tecnología avanza [3, 4]. A la par que se informatiza nuestra vida y se recurren a elementos portables (objetos inteligentes de uso diario que llevamos siempre encima) para transacciones del día a día, hay más información personal que proteger, no sólo claves de acceso y conversaciones seguras, sino también datos médicos personales y de vital importancia para el individuo. Esta interconexión de terminales mediante la red de Internet que se comunican y transfieren información entre sí se conoce como el Internet de las cosas (IoT, Internet Of Things) [5, 6].

Como se ha descrito en un informe reciente, Gartner estima que los terminales de IoT crecerán a un ritmo de 31.7 % hasta 2020, alcanzando los 20.8 billones de unidades, realizándose un gasto en terminales hardware de este tipo de dispositivos de 3 trillones de USD [5, 6]. Con estos antecedentes y pronósticos, se espera que el

hardware criptográfico demande más eficiencia energética e integración que permita mayor portabilidad, mejores prestaciones hardware, y sobre todo seguridad.

A la vista de este escenario sumamente complejo, se está dedicando mucho tiempo a la investigación de nuevos algoritmos e implementaciones seguras que puedan cumplir con las restricciones impuestas por el mercado proponiendo nuevos algoritmos que mejoren las prestaciones y seguridad de estándares fijados por el NIST (National Institute of Standards and Technology) [7]. El NIST fue fundado en 1901 y ahora forma parte del Departamento de Comercio de los Estados Unidos, siendo su principal misión promover la innovación y competitividad industrial en los Estados Unidos. Dentro de los tópicos del NIST se encuentra la ciberseguridad, y más concretamente la criptografía, donde una de sus principales tareas es la estandarización de algoritmos criptográficos.

En el año 1977 el NIST aprobó como estándar el algoritmo *Data Encryption Standard* (DES) [8] y a principios del año 1997, el NIST puso en marcha una iniciativa para desarrollar un nuevo estándar de encriptación llamado *Advanced Encription Standard* (AES) [9] que sucedería al ya presentado DES y sus diferentes actualizaciones como por ejemplo el TripleDES. El proceso de selección propuesto por el NIST fue un concurso público, donde cualquier persona podía presentar un algoritmo candidato que cumpliera con los requisitos exigidos por la convocatoria. Para la selección del algoritmo finalista, el NIST no realizó la evaluación por sí mismo, sino que coordinó a la comunidad experta en este campo para criptoanalizar y evaluar los costes de implementación de cada uno de los algoritmos presentados. El proceso de evaluación consistía en varias rondas de clasificación, durante el cual se organizaron varias conferencias donde los investigadores presentaban sus conclusiones tras diferentes evaluaciones. En la última conferencia, tras realizar un cuestionario, el algoritmo Rijndael resultó finalista, siendo presentado como el AES en el año 2000.

En los últimos años, debido al gran uso de dispositivos portables donde el uso de las baterías y sus limitados recursos de computación imponen fuertes restricciones en las implementaciones, se están buscando soluciones de bajo coste, referidos como *Lightweight Cryptography* [10,11]. La *Lightweight Cryptography* busca dar soluciones criptográficas a dispositivos con recursos limitados, proponiendo nuevos algoritmos e implementaciones eficientes de muy bajo consumo.

Entre los años 2004 y 2008, ECRYPT (*European Network of Excellence in Cryptology*) lanzó el proyecto eSTREAM [12] con el propósito de promover el diseño de cifradores de flujo eficientes y compactos adecuados para su adopción generalizada. El proyecto lanzó un concurso que, al igual que en el caso del AES, fue público y con varias rondas de clasificación. Tras su finalización en 2008, se propusieron como finalistas los algoritmos Grain v1 [13], MICKEY v2 [14] y Trivium [15].

En Agosto de 2018, el NIST ha iniciado un proceso para solicitar, evaluar y estandarizar algoritmos criptográficos *lightweight* que son adecuados para su uso en entornos restringidos donde el rendimiento de los estándares criptográficos actuales del NIST no es aceptable [16]. El NIST ha decidido crear un portfolio de algoritmos criptográficos *lightweight*, diseñados para uso limitado en aplicaciones y entornos donde las operaciones criptográficas son realizadas por dispositivos restringidos que no pueden usar los estándares NIST existentes. Las propuestas de los posibles algoritmos podrán enviarse hasta el 25 de febrero de 2019, teniendo a continuación 12 meses de evaluación pública donde finalmente se seleccionarán los finalistas para una posible estandarización.

Según Elsevier Scopus, la mayor base de datos de investigación, en caso de hacer la búsqueda de la palabra "criptografía" en su base de datos, retorna una búsqueda de mas de 40 mil documentos. Un gran subconjunto de estos documentos trata de soluciones conceptuales, algorítmicas, de software y de hardware que pueden tenerse en cuenta para *Lightweight Cryptography*. Frente a tan vasta literatura, se aprecia claramente las necesidades actuales en el mundo de la criptografía y qué problemas necesitan solución.

En la sección 1.1 se hará una breve introducción a los conceptos básicos de la criptografía. A continuación, en la sección 1.2, hablaremos sobre la seguridad en los dispositivos criptográficos, presentando después técnicas de protección frente ataques no invasivos en la sección 1.3. Finalmente se muestran las conclusiones en la sección 1.4.

El contenido de este capítulo ha derivado en las publicaciones [CV.2] y [CV.3].

1.1. Conceptos básicos de la criptografía

Los algoritmos criptográficos buscan convertir datos secretos en código ininteligible para personas no autorizadas, protegiendo aquella información secreta del robo o alteración, y habilitando la autentificación.

Para poder comprender las siguientes secciones, vamos a definir los siguientes conceptos. Nos referiremos a *plaintext* (pt) cuando se habla del mensaje plano que se quiere encriptar y *ciphertext* (ct) como el mensaje obtenido tras el cifrado. Fijémonos en la Figura , donde Alice es la persona que quiere transmitir los datos de forma segura y Bob es el receptor del mensaje. Alice posee un *plaintext* y una clave que usa para encriptar la información generando el *ciphertext*. Esta información es transmitida por un medio no seguro hasta llegar a Bob. Bob a su vez posee su clave personal que usa para desencriptar el *ciphertext* y recuperar el mensaje legible (pt) enviado por Alice. El proceso de encriptación y desencriptación está detallado en la Figura 1 de CV. 3.

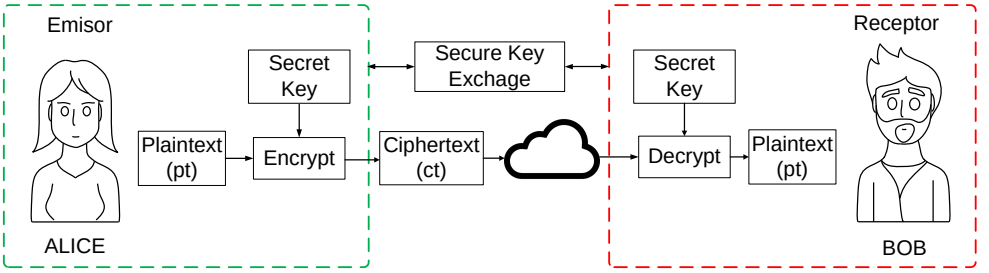

Figura 1.1.: Conceptos básicos de la criptografía.

Para poder alcanzar estos objetivos, la criptografía hace uso de diferentes tipos de algoritmos clasificados en 3 categorías dependiendo del mecanismo de encriptado y el número de claves usadas en el proceso de encriptación (una, dos o ninguna clave), correspondientes a los esquemas de Secret Key Cryptography (SKC), Public Key Criptography (PKC) y Hash de la Figura 2 de CV. 3, respectivamente.

1.1.1. Funciones Hash

Los algoritmos Hash tienen como entrada un *plaintext* de longitud arbitraria y lo transforman en un *ciphertext* con una longitud prefijada sin usar ninguna clave. Estos algoritmos solo funcionan en un único sentido, es decir, una vez transformado el *plaintext* es imposible recuperar el texto original o la longitud del mismo. Su principal uso es el de proporcionar una huella del contenido de un fichero, a menudo utilizado para verificar que el fichero no ha sido alterado por un intruso o virus, permitiendo así comprobar la autenticidad del origen de un mensaje asegurando a su vez la integridad del mismo. Unos de los algoritmos Hash más utilizados son Message Digest (MD) [17] y los SHAs [18].

1.1.2. Criptografía de clave secreta (SKC, Secret Key Cryptography)

También llamada criptografía de clave simétrica, se usa la misma clave para el proceso de encriptado y desencriptado (KeyA = KeyB). Sólo el emisor y receptor del mensaje conocen la clave secreta, por lo que se entiende que en la práctica el canal de comunicación establecido entre ambos es seguro. Los algoritmos SKC se pueden clasificar en dos subgrupos en función de cómo se encripte el *plaintext*: cifradores de flujo si cifran bit a bit o cifradores de bloque si se encripta el *plaintext* por bloques, ver Figura 1 de CV. 2.

- Los cifradores de flujo *o stream ciphers* generan un *keystream* (flujo cifrado) bit a bit a partir de una clave y vector inicial, que posteriormente se utiliza como entrada a una operación XOR junto con el *plaintext*. Estos cifradores implementan algún tipo de realimentación haciendo que el *keystream* varíe continuamente produciendo diferentes *ciphertexts* para el mismo *plaintext* dependiendo de la clave, el vector inicial y el ciclo de encriptación [13, 15, 19]. Existen numerosos *stream ciphers*, como puede ser el One-Time Pad (OTP) [19], donde el *plaintext* y un generador de números aleatorios proporcionan las entradas para la operación XOR bit a bit, generando así el *ciphertex*t. Este cifrador se ha usado extensamente, pero hoy en día ha tenido que ser descartado debido a que la clave utilizada en la encriptación tenía que tener la misma longitud que

el texto a cifrar. Debido a la gran demanda de dispositivos de muy bajo consumo y área reducida, se propuso en 2004 el proyecto eSTREAM [12], donde se pedían propuestas de algoritmos *lightweight* orientados a implementaciones hardware. En este concurso, unos de los algoritmos finalistas y posteriormente más estudiados fueron el cifrador Grain [13] y Trivium [15]. El diseño hardware de Grain está orientado a aplicaciones donde el área, consumo de potencia y memoria son muy limitados. En el caso de Trivium, está diseñado para tener una estructura lo más simplificada posible sin sacrificar la seguridad, velocidad o flexibilidad en la implementación. Otros de los algoritmos *lightweight* orientados a implementaciones hardware presentados en eSTREAM son Mickey, Moustique y F-FCSR-H v2 entre otros [14, 20, 21].

- Por otra parte nos encontramos con los cifradores de bloque o *block ciphers*, los cuales encriptan un bloque completo de datos cada vez. En general, el mismo bloque del *plaintext* se encripta obteniendo el mismo *ciphertext* de salida si se usa la misma clave. Unos de los *block ciphers* más utilizados son el Data Encryption Standard (DES) [8] y el AES [9]. El DES fue diseñado en los años 70 y fue utilizado por el gobierno de los Estados Unidos, consta de una longitud de clave de 56 bits y opera con bloques de datos de 64 bits. En 1997, el NIST inició un proceso público para seleccionar un nuevo cifrador seguro para aplicaciones del gobierno de los Estados Unidos. El cifrador elegido en el concurso fue el AES, que se escogió como el sucesor oficial del 3-DES, variante del DES, en noviembre de 2001. El cifrador AES encripta bloques de datos de 128 bits con claves que pueden ser de 128, 192 o 256 bits. Como en el caso de los cifradores de flujo, debido a la demanda de dispositivos *lightweight,* se están presentando recientemente nuevos cifradores de bloque *lightweight*, como pueden ser el Present [22] (cifrador de bloque *ultra-lightweight* 2.5 veces menor en tamaño que el AES).

La **criptografía de clave secreta** o **criptografía simétrica** utiliza la misma clave para el proceso de encriptado y desencriptado, codificando el *plaintext* bit a bit con los **cifradores de flujo**, y por bloques de datos con los **cifradores de bloque**, planteándose el envío de la clave por un canal no seguro como su principal desventaja.

1.1.3. Criptografía de clave pública (PKC, Public Key Cryptography)

Llamada también criptografía de clave asimétrica, usa un par de claves diferentes para los procesos de encriptado y desencriptado (KeyA $\neq$ KeyB). En este caso la clave KeyA es pública y puede ser utilizada por cualquier persona que desee enviar información privada que solo puede ser desencriptada por el único poseedor de la clave KeyB, asegurando de esta forma la confidencialidad del envío. Por otro lado, cualquier poseedor de la clave pública KeyA podría desencriptar un mensaje cifrado previamente por el único poseedor de la clave KeyB, consiguiendo por tanto la identificación y autentificación del remitente. En 1976 se introduce la PKC [23], que permite junto con el uso de cifradores simétricos, el intercambio de claves secretas incluso en canales de comunicación inseguros o públicos, para salvar la principal desventaja de los cifradores SCK. Los algoritmos PKC que están en uso hoy en día para intercambio de claves o firmas digitales, incluyen el RSA [24] y todos aquellos basados en ECC (*Elliptic Curve Cryptography*) [25, 26] entre otros. El RSA es el algoritmo más implementado, cuyas claves pueden variar entre los 1024 a 4096 bits, previniendo así ataques por fuerza bruta. Unos años más tarde de la presentación del RSA comenzaron los basados en ECC, los cuales requieren claves de menor tamaño para ofrecer los mismos niveles de seguridad.

1.1.4. Implementación Hardware de algoritmos criptográficos

La selección del algoritmo específico a utilizar dentro de estas tres familias, dependerá de la aplicación, el nivel de seguridad que queremos alcanzar, y del coste de implementación y diseño del mismo. Una vez seleccionado el algoritmo, la siguiente decisión importante a tomar es la forma de implementarlo físicamente. Los algoritmos de PKC son más complejos que los de SKC y, para tener el mismo nivel de seguridad necesitan claves con mayor número de bits. En cuanto a velocidad, los cifradores SKC son mucho más rápidos, por lo que son adecuados para cifrar grandes cantidades de datos. Debido a que las propuestas de mejora en seguridad presentadas en esta tesis son aplicables a cualquier tipo de algoritmo, nos centraremos en SKC por su simplicidad y por su extendido uso en aplicaciones de bajo consumo.

Una vez escogido el algoritmo adecuado para nuestra aplicación, el siguiente paso es determinar la implementación optima del mismo. Los algoritmos pueden imple-

mentarse en diferentes niveles de abstracción, desde software/hardware de propósito general, hasta hardware específico. Nos centraremos en algoritmos especialmente diseñados para implementaciones hardware, siendo sus prestaciones mucho más elevadas manteniendo los mismos niveles de seguridad. Dependiendo de las prestaciones requeridas nos decantaremos por una opción u otra, siendo las implementaciones hardware las que mejores prestaciones ofrecen a cambio de aumentar el coste de desarrollo.

En las implementaciones de crypto-hardware empotradas, el algoritmo de encriptación suele incluirse en una FPGA (*Field Programmable Gate Array*) o ASIC (*Application-Specific Integrated Circuit*) como parte de un sistema completo. En muchos casos, para obtener la implementación hardware de un sistema criptográfico, se realiza una síntesis digital de una descripción del algoritmo en un lenguaje hardware (HDL, *Hardware Description Language*). Sin embargo, la implementación resultante puede no ser suficientemente buena en cuanto a prestaciones y/o seguridad, y por ese motivo, el diseñador está obligado a seleccionar cifradores orientados a implementaciones hardware, especialmente en los casos en los que los recursos son limitados (criptografía *lightweight*). Por ejemplo, en [27] se presenta un estudio sobre implementaciones del cifrador Trivium para aplicaciones de muy bajo consumo.

1.2. Seguridad en los cifradores

Tradicionalmente, la seguridad en dispositivos criptográficos estaba exclusivamente vinculada a la fortaleza del algoritmo, que consistía en la capacidad que tenía el cifrador para mantener la información privada protegida. El nivel de seguridad estaba determinado por la formulación matemática del cifrador y la longitud de la clave, a mayor longitud de clave, mayor seguridad. Con la capacidad de cómputo existente hoy en día, los cifradores cuyas claves sean inferiores a 56 bit se consideran no seguros ya que son vulnerables frente ataques de fuerza bruta[1]. En el caso de criptografía *lightweight*, el NIST determina que el nivel de seguridad mínimo expresado

[1] En 1998 el DES Cracker de la EFF (Electronic Frontier Foundation) [28], conocido como "Deep Crack", rompió la clave del DES en menos de 56 horas. Motivo por el cual se consideró no seguro y quedó obsoleto el algoritmo DES (Data Encryption Standard) [8] seleccionado por el NIST como estándar, cuya longitud de clave era de 56 bit.

en numero de bits sería de 112 bits para poder alcanzar un buen compromiso entre prestaciones y seguridad [10], es decir, significa que el atacante tendría que realizar 2^{112} operaciones para romper el cifrador. Como estimación del tiempo de ataque por fuerza bruta, consideremos 1 computador trabajando a 10GHz, que pueda realizar una comprobación de la clave cada ciclo de reloj (lo cual es extremadamente complejo). Permitiría probar 10^{11} claves/s, tendría que probar un total de $2^{112} = 5{,}19 \cdot 10^{33}$ claves, con lo que tardaría $5{,}19 \cdot 10^{22}$s, es decir $1{,}65 \cdot 10^{15}$ años en probar todas las posibles claves. Si la edad del universo es de aproximadamente $13{,}7 \cdot 10^{9}$ años, quiere decir que un atacante necesitaría millones de veces la edad del universo para obtener la clave.

Sin embargo, aunque los algoritmos criptográficos son matemáticamente seguros, sus implementaciones físicas pueden sufrir fugas de información que pueden ser utilizadas por terceros para revelar la información secreta. A mediados de los 90, Kocher demostró que se puede obtener información precisa sobre la clave secreta simplemente observando el circuito durante su modo de funcionamiento normal [29].

Desde entonces, se han publicado numerosas técnicas de ataque a dispositivos criptográficos [2, 30], con el fin de revelar la clave secreta de los dispositivos, con fines alevosos. Estos ataques difieren entre sí en términos de coste, tiempo, equipamiento y la experiencia del atacante. Es por ello por lo que hay varios criterios de clasificación para los ataques a dispositivos criptográficos, principalmente si los ataques son activos o pasivos, y el segundo dependiendo de la interfaz usada en el ataque [2] y [CV.3].

Una primera clasificación puede realizarse en función de si el ataque es activo o pasivo (observar la Figura 1.2):

- **Ataque pasivo:** El dispositivo funciona dentro de sus especificaciones y la clave secreta se revela observando alguna propiedad física del dispositivo, como por ejemplo el tiempo de ejecución o el consumo de potencia.
- **Ataque activo:** El dispositivo criptográfico es manipulado para que funcione de una manera anormal con el objeto de usar este comportamiento erróneo para obtener la clave secreta.

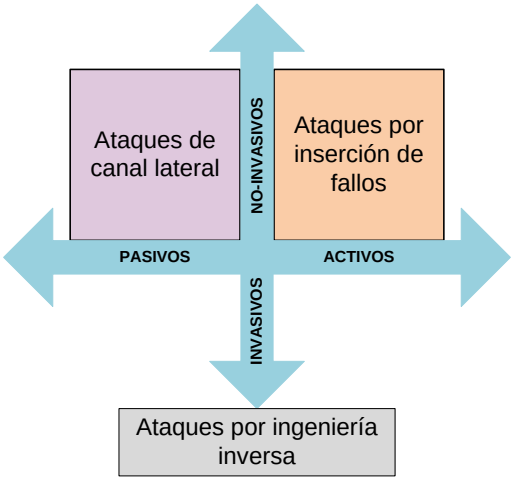

Figura 1.2.: Clasificación de ataques.

Los dispositivos criptográficos presentan diferentes interfaces físicas, algunas de las cuales se podrá acceder de una forma sencilla, mientras otras requieren un equipo especial. Según esto, los ataques se pueden clasificar de tres formas:

- **Ataques invasivos:** Este tipo de ataques modifica de alguna forma el dispositivo criptográfico, sin límite en cuanto a dichas modificaciones, con el fin de revelar la clave secreta. Se accede directamente al dispositivo, haciendo un contacto directo, como por ejemplo, situando una sonda de medida sobre una pista del circuito. Son ataques muy efectivos, pero se necesita una instrumentación costosa y sofisticada, lo que puede suponer una limitación y dejan huella.
- **Ataques semi-invasivos:** En este caso, también se altera el dispositivo pero no se realiza ningún contacto directo con la superficie del chip. Ejemplos de este tipo de ataques son: uso de rayos X, campos electromagnéticos, o luz, para inducir un fallo y obtener información. La dificultad de estos ataques reside en encontrar el lugar exacto donde atacar sobre la superficie del chip.
- **Ataques no invasivos:** En un ataque no invasivo, el dispositivo criptográfico no es alterado y se utilizan únicamente las interfaces accesibles. Por ello, no

se deja evidencia alguna de haber realizado un ataque de este tipo. Una de las ventajas es que la mayoría de ataques no invasivos se pueden realizar con una equipación relativamente poco costosa y son extremadamente potentes, lo que suponen un principal riesgo a tener en cuenta para los diseñadores.

Dentro de los ataques no invasivos nos encontramos con los ataques activos (ataques activos no invasivos), llamados ataques por inserción de fallos (*Fault-injection Attacks*, FA)[2], y con los ataques pasivos (ataques pasivos no invasivos) llamados ataques de canal lateral (*Side-Channel Attacks*, SCA).

1.2.1. Ataques por inserción de fallos

Los FA están clasificados como ataques activos no invasivos. En un FA el atacante inserta un fallo en el dispositivo criptográfico durante la encriptación para hacer que dé un resultado erróneo, es decir, diferente al que daría en un modo de funcionamiento normal. Usando el resultado esperado y el erróneo, los atacantes pueden revelar la clave secreta con la que está operando el dispositivo.

Estos ataques se presentaron en 1997, donde se realizaba un FA al algoritmo RSA utilizando el teorema CRT (*Chinese Remainder Theorem*) [31]. Desde entonces, se han presentado en la literatura numerosos ataques sobre diferentes implementaciones de algoritmos criptográficos, en los cuales consiguen recuperar la clave secreta [32–34]. A continuación se enumeran algunas técnicas de producción del fallo en los FA.

- **Variaciones de tensión de alimentación:** una de las formas más fáciles y baratas de producir el fallo es alimentar el circuito con una tensión diferente a la recomendada o insertar un pico en la tensión de alimentación. De esta forma, la tensión de alimentación variará lo suficiente para causar un funcionamiento incorrecto del sistema, dando a la salida un resultado distinto al esperado [35].
- **Variaciones en la señal de reloj externa:** pueden producir un fallo en la fase de enriptado/desencriptado del criptocircuito. Un ejemplo de este ataque se presenta en [36–38], donde consiguen de forma tanto teórica [36] como experimental [37,38] insertar un fallo en el cifrador de flujo Trivium a traves de un glitch en la señal de reloj. También se presentan varios ataques a los cifradores

[2]No forman parte del contenido de la Tesis.

de bloque AES, DES, Camelia, CAST-128, SEED y MISTY1, donde se atacan insertando fallos en rondas específicas de la encriptación por medio de una señal de reloj con glitches [39].

- **Variaciones de temperatura:** se puede elevar o disminuir la temperatura de los dispositivos para producir un fallo [32].
- **Pulsos electromagnéticos:** se pueden aplicar campos electromagnéticos cerca del dispositivo criptográfico para causar un error [32].

1.2.2. Ataques de canal lateral

En el caso de los SCAs, los atacantes buscan información que pueda medirse durante el modo de funcionamiento normal del dispositivo (consumo de potencia, tiempos de ejecución, radiación electromagnética...), sin dejar rastro alguno de haber sido atacado, recuperando así la clave secreta con la que opera. Los SCAs sobre dispositivos sin proteger requieren un equipamiento mínimo que puede encontrarse en cualquier laboratorio básico y por ello son fáciles de llevar a cabo. Los SCAs más conocidos son los siguientes:

- **Ataques temporales:** se basan en medir los tiempos de ejecución durante la encriptación para poder determinar la clave secreta que se está utilizando [29]. Un ataque de este tipo se presenta en [40] donde se recupera la clave de forma satisfactoria de una tarjeta con una implementación del RSA.
- **Ataques basados en el análisis de consumo de potencia (PA attacks, Power Analysis attacks):** en este tipo de ataques se mide el consumo de potencia del criptocircuito mientras encripta la información [30]. Esta dependencia del consumo con el dato procesado se explota para poder recuperar la clave secreta. Dentro de los PA nos encontramos con los ataques SPA (Simple Power Analysis) si se procesa una única traza de consumo [41], o ataques DPA (Differential Power Analysis) si se procesa más de una traza [2, 42].
- **Ataques electromagnéticos (EM attacks, Electromagnetic attacks):** estos ataques son similares a los PA, pero en vez de monitorizar el consumo de potencia se centran en la medida de la emisión electromagnética [43–48]. En este caso, al igual que en el caso anterior, nos encontraremos frente a ataques

SEMA (Simple Electromagnetic Analysis) si se procesa una única traza, o DEMA (Differential Electromagnetic Analysis) si se procesan multiples trazas electromagnéticas correspondientes a diferentes cifrados para una misma clave.

- **Ataques acústicos:** estos ataques son más novedosos que los de timing, PA o EM, ya que se presentaron en el año 2004. Explotan la relación existente entre el dato procesado y las emisiones acústicas del dispositivo durante la encriptación. Son ataques poco comunes y difíciles de implementar, por lo que son pocos los trabajos presentados al respecto [49, 50].

Hay numerosos SCAs presentados en la literatura para SKC, PKC y funciones Hash. Los que más atención han captado han sido los ataques basados en el análisis de consumo debido a su gran efectividad [2, 51–53].

1.2.2.1. Ataques basados en el análisis de consumo

La idea básica de este tipo de ataques es la de revelar la clave secreta del dispositivo criptográfico analizando su consumo de potencia. Hay dos dependencias con el consumo de potencia que se tienen en cuenta: la dependencia con los datos y la dependencia con las operaciones. En resumen, los ataques basados en el consumo de potencia explotan el hecho de que el consumo de potencia instantáneo de un dispositivo criptográfico depende de los datos y de la operación que esté realizando.

El consumo de potencia de una puerta digital es, normalmente, dependiente del dato que conmuta. Como ejemplo sencillo analizamos el inversor CMOS. Se pueden producir cuatro condiciones a la entrada: que la entrada cambie de '0' a '1', que cambie de '1' a '0', que se mantenga a '0' o que se mantenga a '1'. Tenemos el consumo estático (leakage) cuando el dato a la entrada no cambia, no produciendo un cambio a la salida. El consumo dinámico se produce cuando tenemos un cambio a la salida de '0' a '1', o de '1' a '0' debido a la carga/descarga del condensador a la salida. Dependiendo de la transición entrada/salida del inversor vamos a tener diferentes valores de consumo, como se muestra en la Figura 1.3.

Debido al consumo asimétrico de potencia, los dispositivos criptográficos pueden revelarnos la clave secreta. Monitorizando el consumo de potencia de un dispositivo, se puede determinar qué transición ha tenido lugar, y por tanto, saber el valor

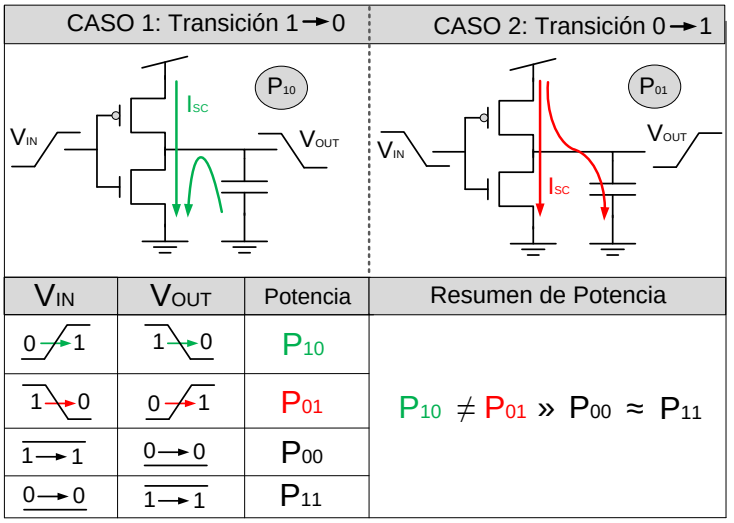

Figura 1.3.: Diferentes consumos de potencia del inversor CMOS dependiendo de las transiciones de entrada/salida y por tanto del dato de entrada.

de entrada al dispositivo. Es esta información la que se usa para el ataque a los dispositivos criptográficos [2].

Dentro de los ataques basados en el análisis de consumo, se pueden diferenciar dos tipos: SPA y DPA [2].

- **El ataque SPA** es una técnica que consiste en interpretar directamente las medidas de consumo obtenidas durante las operaciones de encriptado. Es decir, el atacante intenta obtener la clave directamente de una sola traza de consumo de potencia medida. Este tipo de ataques requieren que el conocimiento del atacante sobre el dispositivo encriptador sea muy alto. Además, al tener que obtener la clave de una única traza, normalmente se necesitan métodos estadísticos más elaborados que en el caso de ataques DPA. Es por ello que este tipo de ataques solo son útiles si se pueden obtener un número muy reducido de trazas.
- **El ataque DPA** es el más popular de los ataques basados en el consumo de potencia. En este caso el atacante no necesita tener un gran conocimiento del

dispositivo encriptador. Incluso operando con señales y en entornos muy ruidosos, este tipo de ataques es capaz de revelar la clave secreta. En comparación con los ataques SPA, los ataques DPA necesitan una gran cantidad de trazas, pero el procesado de datos es más simple que en los ataques SPA.

La seguridad de los dispositivos criptográficos dependen de tres aspectos, el algoritmo matemático implementado, la longitud de la clave y la implementación física del mismo. Por lo tanto, un circuito criptográfico se considerará seguro en la práctica si no hay ningún ataque que pueda revelar la clave en un tiempo razonable y con una capacidad de cómputo realista.

Pero como hemos dicho, debido a la implementación física de los algoritmos puede haber fugas de información que ayuden a revelar la clave. Es por ello por lo que después de presentar los primeros ataques de canal lateral se comenzaron a proponer contramedidas que los protegieran frente a estos ataques, tarea que sigue ininterrumpida hasta la actualidad.

1.3. Protección frente a ataques no invasivos

Si no se emplean contramedidas específicas para bloquear los ataques no invasivos, en la mayoría de los casos, estas implementaciones físicas son vulnerables. La mejor defensa frente a este tipo de ataques reside en el diseño del criptocircuito, pudiéndose aplicar las contramedidas en diferentes niveles de abstracción, teniendo en cuenta los pros y contras de cada una de ellas.

Cuando los diseñadores se enfrentan al diseño de contramedidas frente a ataques no invasivos, necesitan asegurar una buena relación entre prestaciones y seguridad. En aplicaciones criptográficas, la frecuencia máxima de operación del circuito no suele ser un factor limitante. Sin embargo, el área ocupada, consumo de potencia, coste y seguridad son aspectos que hay que tener en cuenta. Los diseñadores, por tanto, buscan cumplir tres aspectos del diseño, a saber: seguridad, coste y prestaciones. Por norma general, es muy complicado cumplir simultáneamente los tres aspectos,

en cambio, dos de ellos se pueden conseguir fácilmente. Por ejemplo, aquellas contramedidas que aumentan la seguridad del diseño sin degradar las prestaciones están siempre ligadas a un aumento del área y por tanto del coste. De la misma forma, un diseño que mantenga niveles de seguridad altos sin incrementar costes se puede conseguir degradando las prestaciones de nuestro diseño.

1.3.1. Protección frente a ataques por inserción de fallos

Existen dos técnicas para proteger un circuito criptográfico frente a FAs [34, 54], [CV.3]:

- Contramedidas hardware: las contramedidas son aplicadas a nivel hardware utilizando mecanismos de prevención, como pueden ser por ejemplo la aplicación de una capa de metal sobre el ASIC para evitar el acceso externo o la iluminación. También hay mecanismos para la detección de luz, variaciones en la alimentación o temporización, o glitches en la señal de reloj [34, 55].
- Otra opción es duplicar el diseño para poder comparar la salida de varios bloques con la misma funcionalidad y chequear la correcta funcionalidad del sistema. En este caso no se evitaría la inserción del fallo, pero se detectaría que ha sido introducido, tomándose por tanto las medidas necesarias como apagar o bloquear el sistema.

1.3.2. Protección frente a ataques de canal lateral

Debido a la dependencia existente entre los datos procesados y las fugas de información del dispositivo durante la encriptación, es posible revelar la clave secreta. Para que un dispositivo criptográfico sea seguro frente a SCAs, las contramedidas aplicadas tienen que eliminar dicha dependencia, o al menos enmascararla lo suficiente para que los ataques sean inviables [2]. Las contramedidas pueden aplicarse a diferentes niveles de abstracción[3]: a nivel de transistor, puerta, arquitectura y algoritmo. Las contramedidas a nivel de algoritmo o arquitectura son muy específicas

[3] El estudio de contramedidas frente a ataques de canal lateral se estudia en detalle en el Capítulo 3.

del algoritmo que se esté implementando, por lo que son difícilmente transferibles y automatizables. Por otro lado, nos encontramos con las contramedidas a nivel de puerta. En este caso no existe dependencia con el algoritmo y pueden aplicarse independientemente del cifrador utilizado. Finalmente tenemos las contramedidas a nivel de transistor, siendo las que mejores resultados ofrecen en cuanto a seguridad se refiere [CV.5]. Además, se pueden aplicar sea cual sea el algoritmo implementado ya que la seguridad reside en la propia construcción de la puerta y no afecta a la funcionalidad de la celda. Dentro de las contramedidas a nivel de transistor nos encontramos con dos tipos: *masking* y *hiding* [2]. Las técnicas de *masking* intentan enmascarar la dependencia del dato procesado con el consumo de potencia, tiempos de ejecución, etc., mientras que las técnicas de *hiding* buscan ocultarla.

1.3.3. Métricas de seguridad

Desde los primeros ataques SCA [29], han sido numerosas las propuestas de nuevos y más elaborados ataques, a la vez que diferentes técnicas para la medida de vulnerabilidad de los circuitos criptográficos [2, 30, 43, 45, 56, 57]. Los diseñadores, tras proponer las contramedidas frente a ataques no invasivos, necesitan determinar la seguridad alcanzada por el criptocircuito para evaluar si el diseño es suficientemente seguro para la aplicación requerida.

La vulnerabilidad o el nivel de seguridad alcanzado por un diseño puede determinarse mediante la realización de ataques o por medio de test o medidas alternativas como puede ser el t-test [57]. Al realizar ataques, es importante distinguir entre el ataque que hace "el malo" sobre el dispositivo final, del que hace "el bueno", para chequear la seguridad de su diseño. En este último caso, los diseñadores realizan un ataque simulando el entorno más propicio para el atacante, determinando si el dispositivo es seguro incluso en el peor de los casos. En el caso de las métricas alternativas, como puede ser el t-test, los diseñadores determinan de forma indirecta el nivel de seguridad alcanzado o la existencia de vulnerabilidades frente a cierto tipo de ataques, pero sin llegar a realizarlos.

La selección entre la realización de los ataques o el uso de métricas alternativas variará en función de la fase de diseño, tipo de algoritmo o aplicación, o de si sólo

queremos determinar si el sistema es vulnerable o por el contrario queremos saber exactamente el punto y condiciones de la fuga de información.

1.4. Conclusiones

La criptografía busca convertir datos secretos en código ininteligible para agentes o entidades no autorizadas, protegiendo aquella información secreta del robo o alteración, y habilitando la autentificación. Para este fin se pueden utilizar diferentes tipos de algoritmos dependiendo de la aplicación y del nivel de seguridad requerido, que pueden ser funciones Hash, cifradores de clave pública/asimétrica o cifradores de clave secreta/simétrica.

Debido a la evolución de la tecnología en las últimas décadas, cada vez se utilizan más dispositivos *wearables*, que demandan seguridad y consumo de potencia muy bajos. Las implementaciones hardware de los cifradores son los diseños más compactos y que mejores prestaciones ofrecen, si son comparados con las soluciones software.

La seguridad de los sistemas criptográficos reside no sólo en la fortaleza del algoritmo criptográfico y la longitud de la clave, sino que también depende de la implementación física. Esto se debe a que existen ataques que explotan la dependencia del dato procesado por el cifrador durante la encriptación con características fisicas del hardware diseñado, para poder revelar la clave secreta. Dentro de estos ataques destacan los ataques de canal lateral conocidos como SCAs, que son ataques de canal lateral pasivos no invasivos.

Los SCAs son los ataques que más riesgo entrañan para los diseñadores de sistemas criptográficos seguros debido a que son ataques que no dejan evidencia ninguna en el dispositivo, además de ser extremadamente efectivos. Desde su proliferación, ha habido numerosas propuestas de contramedidas frente a este tipo de ataques. Las más populares y que mejores niveles de seguridad ofrecen sin degradar en exceso el rendimiento del diseño, son las técnicas de *hiding* aplicadas a nivel de transistor, objeto de análisis en esta Tesis.

2. Vulnerabilidad de circuitos criptográficos frente a ataques de canal lateral

Desde los primeros ataques de potencia [30], han sido numerosas las propuestas de nuevos y más elaborados ataques, a la vez que diferentes técnicas para la medida de vulnerabilidad de los circuitos criptográficos [2,30,43,45,56]. Los cifradores de bloque han recibido considerablemente más atención [8, 9, 22] que los cifradores de flujo [13,15], aunque se han presentado varios trabajos donde se estudia la vulnerabilidad de estos cifradores, ya sea por ataques SCA, o por medio de test alternativos para la medida de seguridad como puede ser el t-test [57].

Dentro de los ataques SCA los más efectivos son los basados en el análisis de consumo [30,56] y en menor medida los basados en emisión electromagnética [43,45], más concretamente los que procesan un número elevado de trazas como son los DPA y DEMA. Dentro de los ataques DPA y DEMA, nos encontramos con diferentes técnicas para la detección de la clave correcta como puede ser la obtención de la diferencia de las trazas de consumo (primera propuesta de "Differential" Power/Electomagnetic Analysis DPA/DEMA [30]) o el cálculo de valores de correlación entre el consumo medido con unos valores hipotéticos de consumo (llamado a veces "Correlation" Power/Electromagnetic Analysis CPA/CEMA) [2]. Debido a que los llamados ataques CPA/CEMA son únicamente una variante de los DPA/DEMA, vamos a designar a todos los ataques como DPA/DEMA tal y como sugieren en [2]. En esta Tesis, debido a su mayor efectividad, nos decantaremos por los ataques DPA que utilizan la correlación para determinar la clave correcta.

Para el estudio de la vulnerabilidad de los dispositivos criptográficos, vamos a comenzar por analizar los ataques a un cifrador de bloque AES, ampliamente utilizado

y atacado en la literatura [2, 51, 58]. Vamos a implementar de forma experimental ataques DPA y DEMA, así como un t-test, para ver la realizabilidad y efectividad de estos ataques. Posteriormente vamos a proponer nuevos ataques sobre un cifrador de flujo Trivium que mejorarán en gran medidas las propuestas presentadas hasta la fecha en la literatura.

Los resultados mostrados en este capítulo se presentaron en [CV.8], [CV.13] y [CV.16].

2.1. Métricas para evaluación de seguridad

Antes de comenzar con el estudio de los SCAs, vamos a determinar qué métricas utilizar para medir el nivel de seguridad de los circuitos criptográficos. En primer lugar, tenemos que tener en cuenta que podemos realizar los ataques en distintos momentos durante la fase de diseño del circuito, por simulación, o a distintos niveles (celdas a nivel de transistor, nivel de bloques), incluso experimental sobre el circuito ya realizado en un ASIC o FPGA. Además, tenemos que determinar si estamos midiendo la seguridad para detectar posibles vulnerabilidades y poder aplicar contramedidas para evitar fugas de información, o simplemente demostrar que un sistema es vulnerable, sin importar los detalles exactos del punto o condiciones de fuga.

En la literatura nos encontramos con numerosas métricas propuestas que se adaptan a cada fase del diseño y nivel de descripción. En primer lugar, en la fase de diseño de celdas seguras, podemos hacer unas primeras estimaciones que nos den una aproximación de la robustez que tiene la celda frente a SCAs, pero decididamente sólo se demuestra dicha resistencia realizando un ataque completo. Entonces nos podemos preguntar ¿por qué realizamos estas estimaciones si podemos realizar un ataque? La respuesta es muy simple, la realización de un ataque DPA no es inmediata, hay que realizar el diseño de al menos una parte del dispositivo criptográfico, el montaje de todo el experimento, simulaciones o medidas en el laboratorio muy costosas, almacenamiento de gran cantidad de datos y procesado de los mismos. Además, en la fase de diseño de la celda, nos podemos encontrar con numerosas propuestas para mejorar la seguridad, por lo que si queremos probar todas las alternativas nos encontraríamos con un número demasiado elevado de posibles combinaciones que

tendríamos que evaluar. Esto supondría que tendríamos que diseñar no solo las celdas correspondientes, sino también los bloques criptográficos, para tantas propuestas como posibles combinaciones de ellas hayamos planteado, y realizar las medidas y procesado posterior convirtiéndose en una tarea inabordable.

En la Figura 2.1 se muestran unos indicadores de métricas aplicables en cada fase de diseño y nivel de abstracción, frente a ataques DPA. A su vez, estas métricas pueden ser aplicables en entornos de simulación y/o experimentales, siendo algunas de ellas no realizables o inadecuadas dependiendo del tipo de implementación que tengamos, por lo que la elección de la métrica de seguridad a utilizar es de suma importancia.

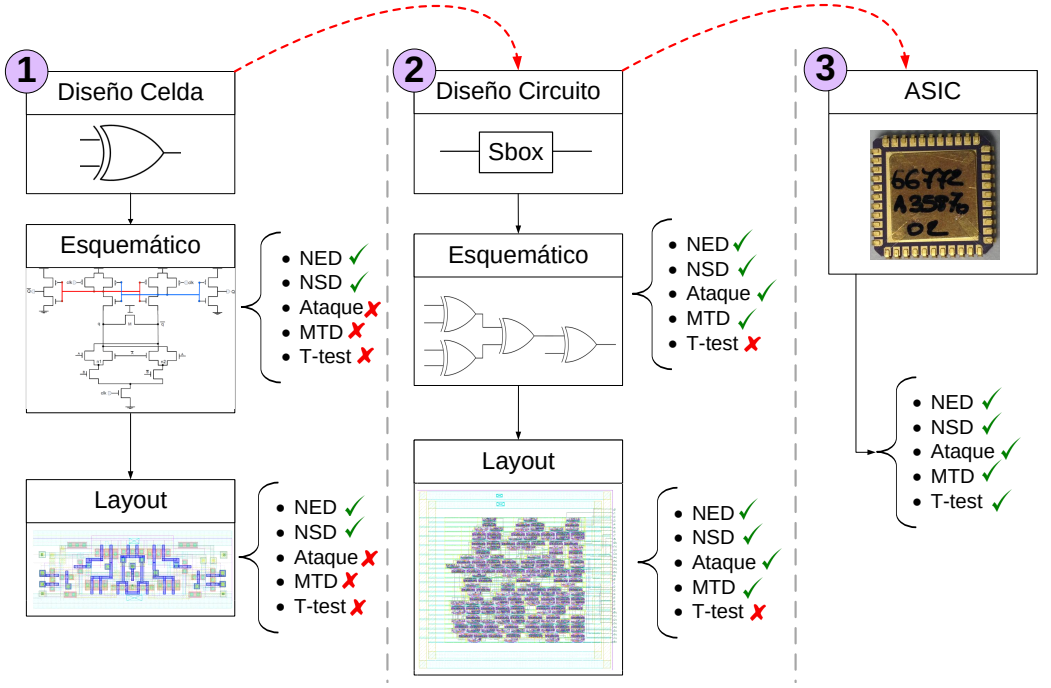

Figura 2.1.: Posibles medidas de seguridad a realizar sobre las celdas DPL según la fase de diseño. ✓ significa aplicable y X no aplicable.

2.1.1. Métricas indirectas de seguridad

Las medidas que vamos a mostrar a continuación nos permiten, mediante una pequeña simulación eléctrica, determinar la robustez de una celda individual, sin la necesidad de realizar el diseño de un sistema completo, y por supuesto nos permitirá comparar las características de seguridad de diferentes alternativas, descartando aquellas menos robustas. De esta forma, podemos determinar en una primera fase de diseño, qué contramedidas propuestas son válidas y pueden pasar a una segunda etapa desechando las que peores métricas muestren.

De las simulaciones eléctricas obtendremos la corriente de polarización, que procesaremos para obtener de cada transición la energía por ciclo (E) calculada a partir de la corriente (I_{AV}) medida:

$$I_{AV} = \frac{1}{T_{CLK}} \int_{-T_{CLK}/2}^{T_{CLK}/2} i_{DD}(t)dt \tag{2.1}$$

$$E = V_{DD} \cdot I_{AV} \cdot T_{CLK} \tag{2.2}$$

Tras calcular la energía por ciclo, podemos obtener:

- Min: Valor mínimo de energía.
- Máx: Valor máximo de energía.
- μ: Valor medio de energía.
- σ: Desviación estándar de valores de energía.

Las medidas más utilizadas en la literatura y que mejor definen el nivel de seguridad de una celda frente a ataques de potencia son las siguientes [59], [CV.5]:

- NED (*Normalized Energy Deviation*):

$$NED = \frac{Max - Min}{Max} \tag{2.3}$$

- NSD (*Normalized Standard Deviation*):

$$NSD = \frac{\sigma}{\mu} \tag{2.4}$$

La interpretación es la siguiente, una celda completamente segura presentaría **NED**=0 y **NSD**=0, indicando que el consumo es el mismo para todas las transiciones y por tanto independiente del dato. En su defecto, la celda es más segura cuando menor sea el valor del **NED** y **NSD**.

Ambas métricas están concebidas para utilizarse en una primera fase, donde se busca descartar las contramedidas que peores resultados de seguridad muestren. Si bien el NED y NSD son aplicables de igual forma en las siguientes fases de diseño, tanto en un entorno de simulación como experimental, no resultan adecuadas y es recomendable el uso de otras métricas.

2.1.2. Métricas directas de seguridad

Una vez seleccionadas las propuestas, se pasa a una segunda fase, donde se implementarán los bloques constitutivos del algoritmo criptográfico que sean más vulnerables a los ataques de canal lateral con las celdas propuestas. En esta fase, ya se pueden realizar ataques DPA, tanto en entornos de simulación como experimentales, que nos mostrarán con más exactitud el nivel de seguridad alcanzado.

De los ataques DPA podemos extraer diferentes informaciones cuantitativas para evaluar la seguridad. Se puede determinar si el ataque ha sido satisfactorio o no comparando la clave obtenida en el ataque con la fijada en el circuito criptográfico. Una vez obtenido un ataque satisfactorio, se puede calcular el MTD (*Measurements To Disclose the key*), para cuantificar el nivel de seguridad [60], ampliamente utilizado en la literatura [61, 62].

El **MTD** nos indica el número mínimo de patrones de entrada que hay que aplicar en el ataque para que siempre se pueda detectar la clave correcta. Por lo que, con mayor MTD mayor nivel de seguridad alcanza el criptocircuito y por tanto la contramedida propuesta.

Por lo tanto, aunque a nivel de circuito se pueden obtener los valores de NED y NSD, vamos a descartar estas medidas ya que no aportan información complementaria y nos proporciona información más exacta el MTD.

2.1.3. Métricas de vulnerabilidad

Existen otras técnicas que sólo buscan determinar si un sistema es vulnerable o no, como puede ser el t-test [57]. Estas técnicas tienen ventajas e inconvenientes. Como principal ventaja tenemos que es mucho más sencilla de implementar y nos ahorramos el laborioso trabajo de desarrollar la estrategia de ataque, determinando si el sistema es seguro o no lo es. Por el contrario, estas técnicas no sirven para comparar dos contramedidas propuestas, ya que sólo dan información de si una implementación es segura, pero no permite determinar el grado de seguridad alcanzado. Además, no son adecuadas para su implementación en entornos de simulación, siendo aplicables por tanto en setups experimentales. Es por esto por lo que en esta Tesis se van a utilizar las métricas definidas en la Tabla 2.1 para poder evaluar las contramedidas propuestas en las diferentes fases de diseño.

En resumen, en cuanto a medida del nivel de seguridad de los diseños criptográficos, vamos a utilizar por tanto el NED, NSD para celdas lógicas y ataques DPA para bloques criptográficos completos. Tras realizar numerosas pruebas, aplicaremos ocasionalmente el t-test en aquellos sistemas resistentes, ya que si bien no sirve para comparar dos implementaciones entre sí, podremos determinar si las implementaciones tienen fuga de información aunque no podamos saber cuál de las propuestas es la más segura.

Tabla 2.1.: Medidas utilizadas en esta Tesis en cada fase de diseño.

	NED	NSD	Ataque	MTD	T-test[1]
Celda	✓	✓	✗	✗	✗
Circuito	✗	✗	✓	✓	✗
ASIC	✗	✗	✓	✓	✗

[1] Se ha explorado parcialmente, algunos resultados aparecen en la Sección 2.3.5.

2.2. Ataques DPA: Bases

El objetivo de un ataque DPA consiste en revelar la clave secreta de los dispositivos criptográficos utilizando una gran cantidad de trazas de consumo de potencia de alimentación medidas de los dispositivos mientras encriptan o desencriptan datos para esa clave secreta.

En la Figura 2.2, se puede ver un esquema muy básico del funcionamiento de un cifrador. A grandes rasgos, en un ataque DPA[1] vamos a tener una clave secreta (*Key*) que queremos averiguar. Para ello, se medirán las trazas de consumo de potencia del criptocircuito, que podrá estar implementado en ASIC, FPGA, o incluso un esquemático por simulación, durante la encriptación de los *plaintexts* que le pasemos como datos de entrada. Con la información de las trazas de consumo y los datos de entrada y/o salida (*ciphertext*) del sistema criptográfico y un modelo de consumo de potencia adecuado se realizarán análisis estadísticos para determinar con qué clave está trabajando el dispositivo. El ataque será satisfactorio en caso de recuperar la clave secreta.

Para realizar un ataque DPA, no es necesario conocer en detalle el dispositivo criptográfico a atacar, sino que basta con saber el algoritmo que implementa. En el caso de los ataques DPA, no son importantes cada uno de los valores de consumo en cada

[1] En esta Tesis, nos referiremos a ataques DPA de primer orden como ataques DPA, quedando fuera del estudio de esta Tesis los ataques de orden superior.

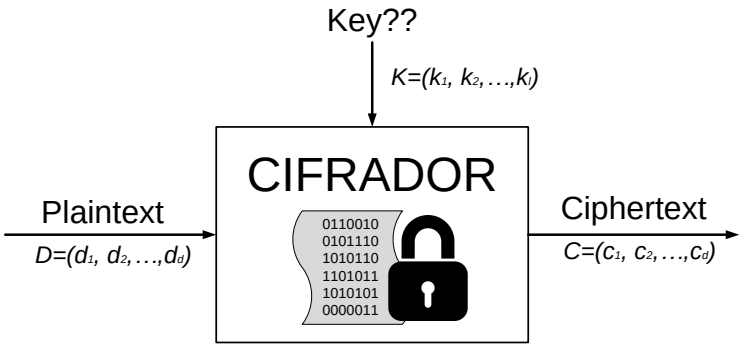

Figura 2.2.: Esquema básico de funcionamiento de un cifrador.

instante durante el periodo de encriptación del que se tiene la traza de consumo. Los ataques DPA intentan asociar una parte de la función de encriptado en un instante determinado de tiempo que depende del dato que se está procesando. Por lo tanto, se puede decir que los ataques DPA se focalizan exclusivamente en la dependencia de los datos con la traza de consumo medida.

Ahora planteamos la estrategia a seguir en la realización de un ataque DPA para obtener la clave secreta que utiliza el dispositivo criptográfico. Como información de partida tenemos el algoritmo que implementa el dispositivo, que es de dominio público y por tanto conocido, el *plaintext* y el *ciphertext*, accesibles por el atacante, y por último el consumo de potencia del dispositivo durante el encriptado. El ataque consta de 5 pasos que se describen a continuación [2].

1. **Selección de una operación ejecutada por el algoritmo durante la encriptación que implique la clave y el texto de entrada o salida.** Dado que el algoritmo es conocido por el atacante, el primer paso de un ataque DPA es escoger una operación que se realice en el algoritmo en el que conozcamos parte de la información que use tanto datos de texto de entrada o salida (suponemos por simplicidad que la dependencia es con la entrada d) como datos de la clave (k), es decir una función *f(d,k),* donde d es una matriz conocida compuesta por datos de entrada, y k es otra matriz con todos los posibles valores de la clave.

2. **Medida de las trazas de consumo**. En este paso se ejecuta la operación del dispositivo criptográfico donde mediremos el consumo de potencia de éste durante su ejecución. Para cada bloque de datos de entrada se va construyendo la matriz $D = (d_1, d_2, ..., d_d)$, donde d_i se corresponde con cada dato conocido en la iésima ronda de ejecución. Durante cada una de esas ejecuciones se mide el consumo de potencia del dispositivo y se almacena la traza de consumo correspondiente a dicha transición. Se construye entonces la matriz T, teniendo para cada dato de entrada d_i su traza de consumo $t_i = (t_{i,1}, t_{i,2}, ..., t_{i,n})$ correspondiente, siendo n la longitud de la traza (número total de puntos muestreados en la medida del consumo). Con esto podemos definir las dimensiones de T, que tendrá d filas y n columnas. En la medida de las trazas y en la construcción de la matriz, hay que tener especial cuidado en tener todas las trazas bien alineadas, ya que la relación de los datos de entrada con los resultados de consumo dependen de la operación que se esté realizando en un instante de tiempo determinado.

3. **Calcular los valores hipotéticos intermedios.** El siguiente paso en el ataque DPA es calcular los valores hipotéticos intermedios para cada una de las posibles claves k. Para ello hay que centrarse en una operación del algoritmo en la que intervengan los valores de entrada y la clave $f(d, k)$. Para cada uno de los datos de entrada d_i se calcularán los valores que tomará el algoritmo en el punto de ataque $f(d_i, k_j)$ para cada una de las posibles claves k_j de la matriz $K = (k_1, k_2, ..., k_l)$, donde l es el número total de posibles claves. De este paso obtenemos la matriz de valores hipotéticos V, que se define como: $V_{i,j} = f(d_i, k_j)$ con $i = 1, 2..., d$ y $j = 1, 2, ..., l$. La matriz V tendrá tantas filas como datos de entrada d tengamos, y tantas columnas como posibles claves.

4. **Asignar los valores hipotéticos intermedios a valores de consumo de potencia.** En este paso hay que transformar los valores de V a la matriz H, que contendrá los valores hipotéticos de consumo de potencia para cada dato y clave. La matriz H tendrá las mismas dimensiones que la matriz V. Para asignar a los datos valores de potencia hay que utilizar unos modelos de potencia cuya elección determinará en gran medida el éxito del ataque [2]. Los 3 modelos de potencia más conocidos son [2]:

a) *Zero-Value Model.* Este modelo asume que el valor de consumo del dispositivo para un valor de V igual a '0' es menor que si toma el valor de '1'. Teniendo esto en cuenta podríamos definir el *Zero-Value Model* como:

$$H_{i,j} = ZV(v_{i,j}) = \begin{cases} 0 & v_{i,j} = 0 \\ 1 & v_{i,j} \neq 0 \end{cases} \tag{2.5}$$

b) *Hamming-Weight Model.* En este modelo el valor del mapeado de potencia consiste en mirar el número de bits a '1' que contiene cada vector de entrada V. Este modelo se suele aplicar siempre y cuando no se pueda aplicar el *Hamming-distance model*, o éste no de buenos resultados.

c) *Hamming-Distance Model.* Este modelo es más complejo que el *Hamming-Weight* Model. Este caso es adecuado para describir el consumo de potencia de buses de datos o registros, en relación al consumo dinámico producido por cambio en los bits de datos. El valor del consumo de potencia que se obtiene al cambiar en un bus el valor de v_0 a valor v_1 es proporcional a la distancia hamming entre v_0 y v_1 . En el caso del ataque tendremos que $HD(v_0, v_1) = HW(v_0 \oplus v_1)$, siendo HW (*Hamming-Weigth*) el número de bits a '1' que contiene la operación $v_0 \oplus v_1$.

5. **Comparar los valores hipotéticos con los valores de consumo de potencia reales obtenidos en las trazas.** Ya tenemos construidas las siguientes matrices de datos: *D, T, K, V* y *H*. Una vez mapeados los valores de V a H (paso 4), cada columna de la matriz H se compara con cada columna de la matriz T. Esto significa que el atacante compara cada uno de los valores hipotéticos de potencia para cada clave posible con todas las trazas de potencia en cada posición de tiempo. Esta comparación de datos puede realizarse mediante la correlación de las matrices H y T. De esta comparación se obtiene una última matriz R, que muestra la correlación existente para cada una de las posibles claves k en cada instante de tiempo, y que se calcula con la siguiente ecuación:

$$r_{i,j} = \frac{\sum_{d=1}^{D}(h_{d,i} - \overline{h_i}) \cdot (t_{d,j} - \overline{t_j})}{\sqrt{\sum_{d=1}^{D}(h_{d,i} - \overline{h_i})^2 \cdot \sum_{d=1}^{D}(t_{d,j} - \overline{t_j})^2}} \tag{2.6}$$

Con la obtención de la matriz R se puede determinar la clave correcta. La clave que más valor de correlación aporte será la clave correcta y la que utiliza por tanto el dispositivo criptográfico. En la Figura 2.3 se muestra un esquema de los pasos a seguir en un ataque DPA.

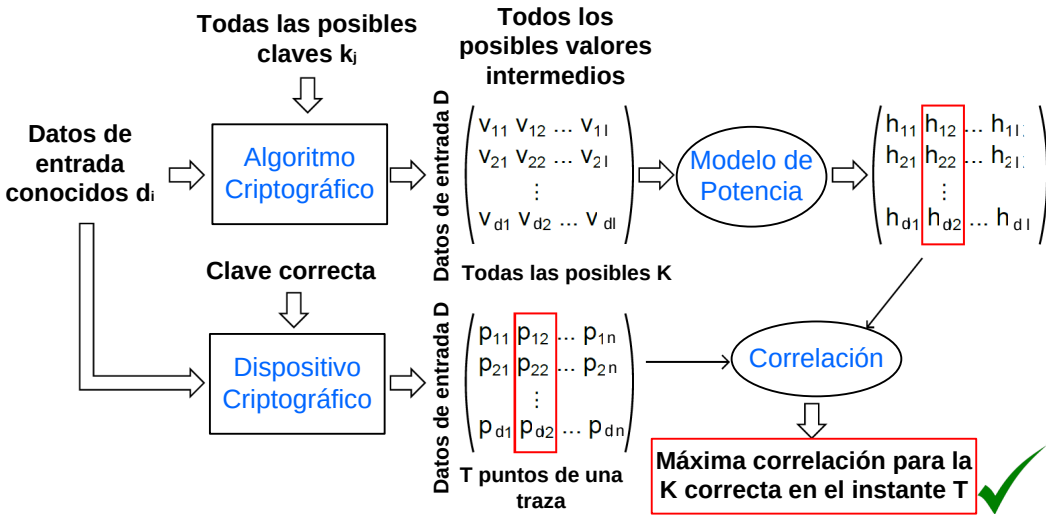

Figura 2.3.: Esquema simplificado del ataque DPA.

2.2.1. Entornos para la medida de trazas de consumo

En los ataques DPA podemos tener diferentes tipos de ataque: ataques reales en los que se ataca un criptocircuito implementado en ASIC o FPGA, o ataques por simulación donde el cifrador es implementado sobre un esquemático con una herramienta de CAD (*Computer-Aided Design*). Aunque los pasos del ataque son los mismos, la forma de medir las trazas y el procesado de los datos hasta obtener la matriz T y D difieren en ambos casos.

En un ataque experimental nos vamos a encontrar en un laboratorio, con un esquema del entorno de ataque semejante al de la Figura 2.4. En primer lugar se necesita un ordenador que controle todo el experimento, coordinando los equipos de medida y

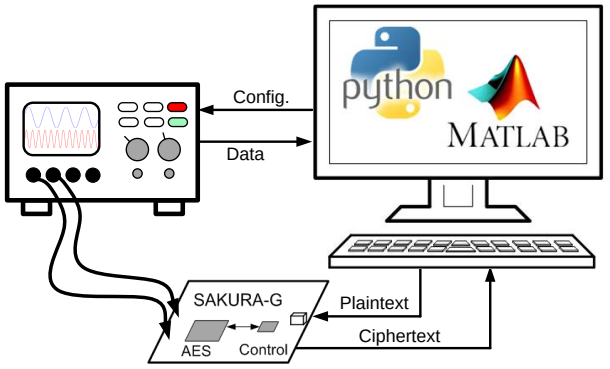

Figura 2.4.: Setup experimental de un ataque DPA.

la operación del cifrador. Las trazas de consumo para cada una de las encriptaciones del *plaintext* se medirán con un osciloscopio y se enviarán al ordenador para almacenarlas en la Matriz T. A su vez, tanto los *plaintext* aplicados como el *ciphertext* se almacenarán para ser utilizados como nuestra matriz de datos conocidos D. En este caso, para poder medir las trazas de consumo se va a necesitar una señal de disparo que nos permita sincronizar la captura y obtener la traza siempre en el mismo ciclo de reloj. Además, hay que tener en cuenta que las trazas necesitan un preprocesado específico para alinearlas correctamente en caso de variaciones de la señal de disparo o adquisición de datos. Cada una de las trazas correspondientes a cada *plaintext* se medirán de forma individual, siendo los pasos genéricos a seguir los mostrados en el organigrama de la Figura 2.5.

Los pasos se repetirán del 1 al 6 para cada traza, y una vez finalizado el proceso, se alinean las trazas de la matriz T.

En caso de encontrarnos en un entorno de simulación, vamos a necesitar únicamente un ordenador que tenga la herramienta para poder implementar nuestro esquemático del criptocircuito y poder realizar una simulación eléctrica del mismo para obtener la traza de consumo. En el esquemático a implementar vamos a tener dos partes: la celda o bloque que simule nuestro cifrador y el bloque que genere los *plaintext* junto con el elemento que almacene el valor de la clave secreta.

Una vez finalizada la adquisición de trazas de consumo y tras almacenar los *plaintexts* y/o *ciphertexts*, un script de Matlab calculará la matriz de correlación que mostrará la clave correcta.

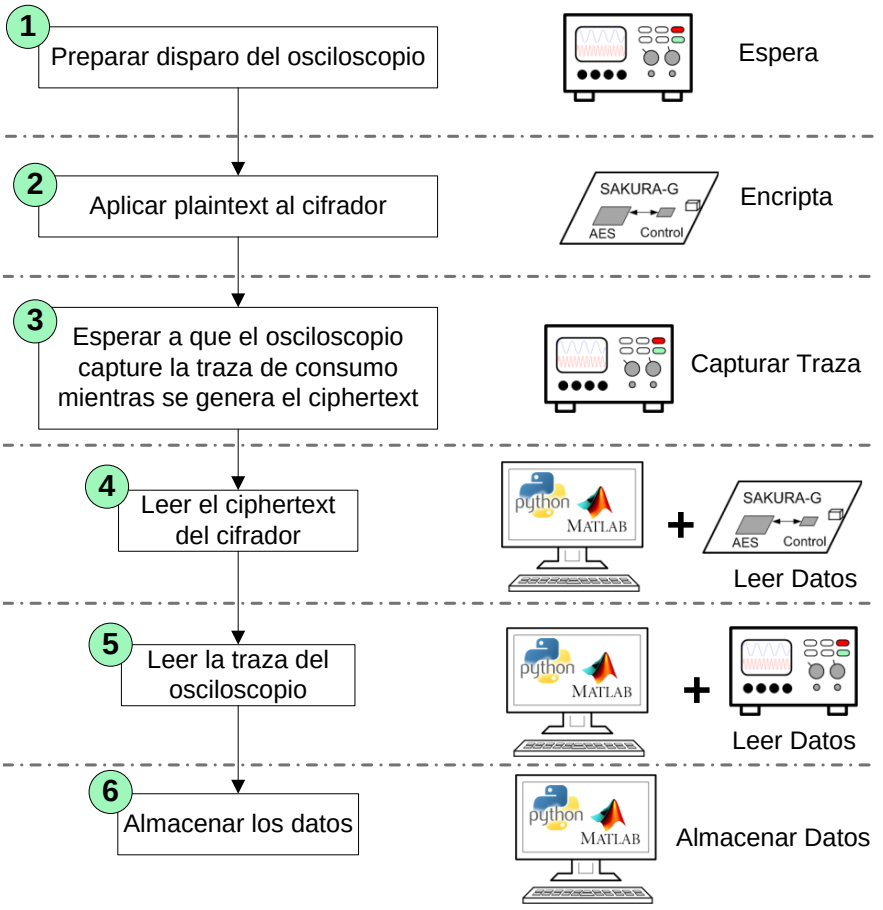

Figura 2.5.: Organigrama de la captura de datos para un ataque DPA experimental.

2.3. Medidas de vulnerabilidad sobre el cifrador de bloque AES

Para comprender mejor las medidas de vulnerabilidad de criptocircuitos, vamos a comenzar por analizar uno de los cifradores más utilizados, y por tanto, más atacados, el AES [9].

El NIST usaba como estándar los algoritmos de encriptación DES y 3-DES, pero empezaron a quedar obsoletos y a mediados de los 90 se propuso la selección de un nuevo estándar. A principios del año 1997 el NIST propuso un concurso abierto donde cualquier persona estaba invitada a proponer un algoritmo que reemplazara el antiguo DES y 3-DES. Durante los años 1997 y 2000 se realizaron varias conferencias donde se iban evaluando y desechando los algoritmos que peores prestaciones ofrecieran. El NIST no se encargaba directamente de la evaluación de los algoritmos presentados, sino que se constituyó un comité de evaluación formado por diferentes estudiosos de la criptocomunidad. El día 2 de Octubre del año 2000 se nombró oficialmente al algoritmo Rijndael como el nuevo estándar AES del NIST [2].

El AES en primera instancia fue utilizado por el gobierno de los Estados Unidos para cifrar datos críticos pero no clasificados. Sin embargo, partir de Junio del 2003, el NSA (*National Security Agency*) determinó que era lo suficientemente seguro para encriptar datos "Top Secret" con una longitud de clave de 128 bits. Por ese motivo, el AES ganó prestigio y se comenzó a adaptar como algoritmo seguro de referencia para bancos, industria y administración de todo el mundo, puesto en el que aún sigue.

La selección del AES como algoritmo de referencia para comenzar con los estudios de vulnerabilidad frente SCAs viene motivado en primer lugar por ser el cifrador en uso más extendido en diferentes ámbitos, además de ser uno de los más estudiados en cuestiones de seguridad [2, 42, 48]. Por otro lado, nos va a permitir corroborar que aún siendo un algoritmo seguro, prueba de ello son las aplicaciones de alto nivel de confidencialidad en las que se usa, su implementación física puede revelarnos información que usada correctamente nos permite averiguar la clave secreta.

2.3.1. Descripción del algoritmo

El algoritmo AES es un cifrador de bloque de clave simétrica. Encripta bloques de datos de 128 bits utilizando una clave, cuya longitud puede variar en 128, 192 o 256 bits. En esta Tesis nos referiremos al AES de 128 bits como AES, que es el que tomaremos como ejemplo en nuestro análisis. Su estructura opera con matrices de datos de 128 bits tanto para la clave como para el *plaintext* como se muestra en la Figura 2.6. Los 128 bits de datos de cada matriz se dividen en 16 bloques de 8 bits, estando compuesta, por ejemplo, la matriz de *plaintext* por d_0, d_1, ..., d_{15}, siendo $d_0 = [bit_0, bit_1, bit_2, bit_3, bit_4, bit_5, bit_6, bit_7]$, $d_1 = [bit_8, bit_9, bit_{10}, bit_{11}, bit_{12}, bit_{13}, bit_{14}, bit_{15}]$ y así sucesivamente.

d_0	d_4	d_8	d_{12}
d_1	d_5	d_9	d_{13}
d_2	d_6	d_{10}	d_{14}
d_3	d_7	d_{11}	d_{15}

k_0	k_4	k_8	k_{12}
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

Figura 2.6.: Bloques de datos del *plaintext* (d) y clave (k).

El algoritmo del AES se puede dividir en dos bloques: el de transformación del bloque de estados (ver pseudocódigo en el Algoritmo 2.1) y el encargado de la expansión de la clave (ver pseudocódigo en el Algoritmo 2.2). El primero de los bloques aplica diferentes transformaciones al estado (S) divididas en 10 rondas. En primer lugar, el estado se carga con los valores del *plaintext*, como se muestra en la Figura 2.7, y seguidamente, se le aplica la función AddRoundKey() junto con el primer valor de la clave. A continuación, cada una de las funciones intermedias del algoritmo se le van aplicando al estado, transformándolo en cada ronda, teniendo al final de las rondas el texto cifrado.

Por otra parte nos encontramos con la expansión de la clave. Es la encargada de generar la claves intermedias utilizadas en cada ronda como se muestra en el pseudocódigo del Algoritmo 2.2. Cada una de las claves de cada ronda se obtiene de la clave expandida anterior. De esta forma, en cada ronda se mezclará el estado con una clave diferente. Una vez finalizada la expansión, se tiene una clave expandida

Algoritmo 2.1 Pseudocódigo del algoritmo AES [2]

```
Entrada: AES_Algorithm (byte in[16], byte out[16], word w[44])
byte state[4,4]
state = in
AddRoundKey(state, w[0,3])
for round = 1 step 1 to 9 do
    SubBytes(state)
    ShiftRows(state)
    MixColumns(state)
    AddRoundKey(state, w[round · 4,(round + 1) · 4 - 1])
end for
SubBytes(state)
ShiftRows(state)
AddRoundKey(state, w[40,43])
out=state
```

Algoritmo 2.2 Pseudocódigo de la expansión de clave del AES [2]

```
Entrada: AES_KeyExpansion (byte key[16])
RC[1,2,...,10] = ('01','02','04','08','10','20','40','80','1B','36')
Rcon[i] = (RC[i],'00','00','00')
for round = 0 step 1 to 3 do
    w[i] = (key[4 · i], key[4 · i + 1], key[4 · i + 2], key[4 · i + 3])
end for
for round = 4 step 1 to 43 do
    temp = w[i - 1]
    if (i mod 4 == 0) then
        temp = SubWord(RotWord(temp)) xor Rcon[i/4]
    end if
    w[i] = w[i-4] xor temp
end for
```

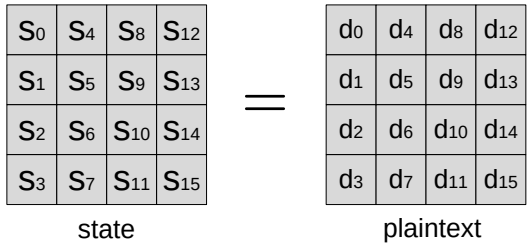

Figura 2.7.: Carga del *plaintext* en el estado.

de 11 veces el tamaño de la clave inicial, es decir, 11 bloques de clave diferentes que se utilizarán para el cifrado.

Función AddRoundKey()

Realiza la operación XOR del bloque de estados con la clave expandida de cada ronda (ver Figura 2.8).

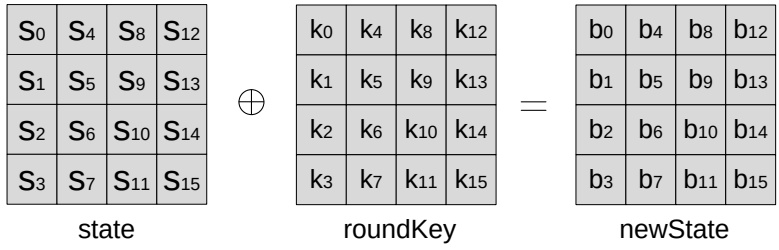

Figura 2.8.: Función AddRoundKey().

Función SubBytes

En esta función se aplica la transformación del bloque de estados con la Sbox8 del AES byte a byte como se muestra en la Figura 2.9. Un bloque Sbox (*Substitution Box*), es un componente básico de los algoritmos de clave simétrica que se utiliza

para romper la relación existente entre el *plaintext* y el *ciphertext*, transformando el dato a la entrada de la Sbox de n bits a un dato de salida de la misma longitud.

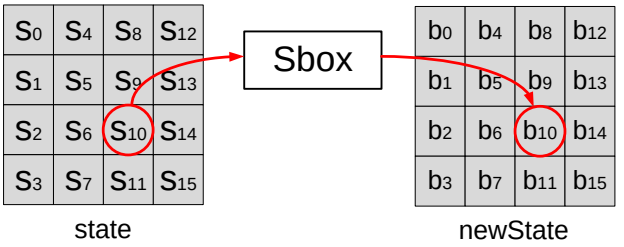

Figura 2.9.: Función SubBytes().

Función ShiftRows

La función ShiftRows(), que se muestra en la Figura 2.10, simplemente rota cada una de las filas de la matriz del estado. La primera fila del estado del AES no varía, mientras que la segunda rota una posición, la tercera fila rota 2 posiciones y la última rota 3 posiciones, siempre realizando la rotación hacia la izquierda. La elección del número de posiciones rotadas por fila está relacionada con los requisitos de difusión del algoritmo.

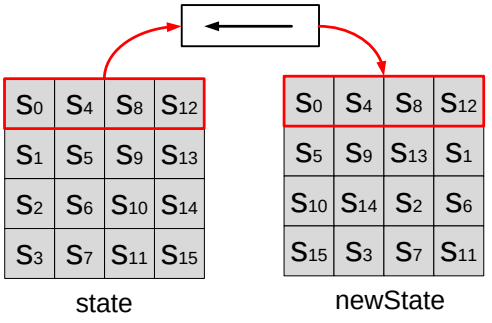

Figura 2.10.: Función ShiftRows().

Función MixColumns

La función MixColumns() multiplica el estado S por una matriz de datos fijos, tal y como se muestra en la Figura 2.11.

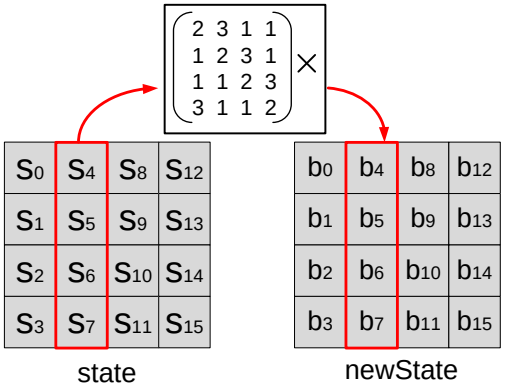

Figura 2.11.: Función MixColumns().

2.3.2. Ejemplo de ataque DPA

El primer paso en la realización del ataque DPA es la selección del punto de ataque. El punto crítico de este algoritmo se encuentra en la operación SubBytes, es decir, en la salida de la Sbox8 del AES [2]. En este punto tenemos que intervienen tanto la clave como el *plaintext* que componen el estado y se transforman después de la operación SubBytes a un nuevo estado.

Como el AES opera en diferentes rondas, tenemos varios instantes de ataque a lo largo de la encriptación, ya que la operación SubBytes se realiza 10 veces. Los puntos escogidos como ataque suelen ser la primera y la última ronda, ya que en la primera tenemos como dato el *plaintext* y en la última ronda el *ciphertext*. En el caso de atacar la primera ronda, se tiene que el estado actual es el resultado de la operación AddRoundKey entre el *plaintext* y la primera clave expandida. Aunque el ataque es viable en esta ronda, se han reportado ataques más efectivos realizándolos en la última ronda, siendo esta la más extendida entre los atacantes.

Para evitar el nivel de seguridad que supone el uso de claves largas (128 bits, 2^{128} posibles claves), se aplica la estrategia de divide y vencerás, atacando byte a byte, recuperando en cada ataque 8 bits (256 posibles claves) hasta recuperar los 128 bits de clave. En el caso de la última ronda tenemos que el *ciphertext* se mezcla con la clave expandida de la última ronda, tal y como se muestra en la Figura 2.12. Podemos apreciar en la ecuación de b0 que la operación ShiftRows no afecta en la primera fila del estado. El estado intermedio de la encriptación (B), se va obteniendo aplicando las funciones del cifrador a la inversa. De este modo, tendremos los valores antes y después de la Sbox, que son los que se utilizarán para calcular la distancia de Hamming y por consiguiente, el consumo hipotético de potencia de la Sbox durante la encriptación en la última ronda de operación.

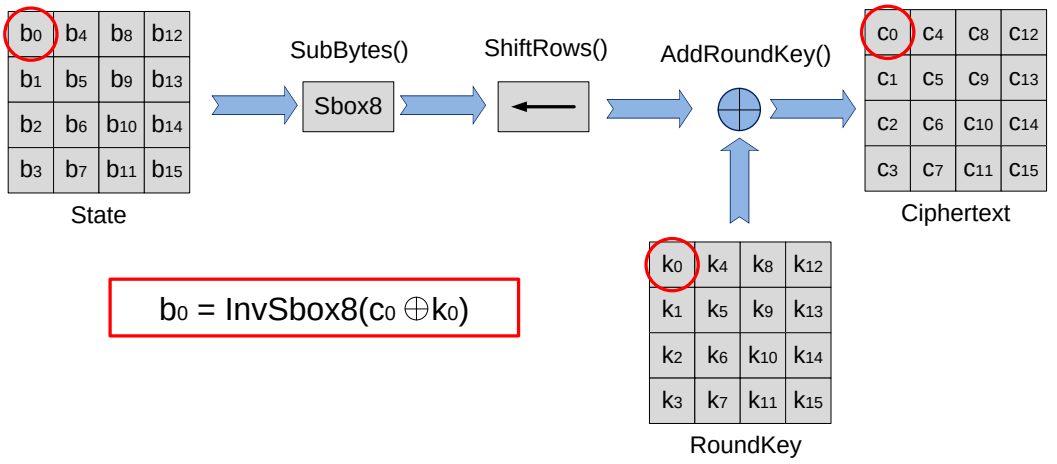

Figura 2.12.: Última ronda de ataque en AES.

Una vez escogido el punto de ataque del cifrador al que se va a atacar, el siguiente paso es medir las trazas de consumo durante la encriptación de un número elevado de *plaintexts* para la misma clave secreta. Para mostrar este ejemplo de ataque al cifrador AES, vamos a realizar un ataque experimental sobre una implementación del AES en FPGA. Los elementos utilizados en el setup de medida son los siguientes: ordenador, FPGA SAKURA-G (*Side-channel AttacK User Reference Architecture*) [63], osciloscopio y fuente de alimentación. Sincronizando todos los equipos,

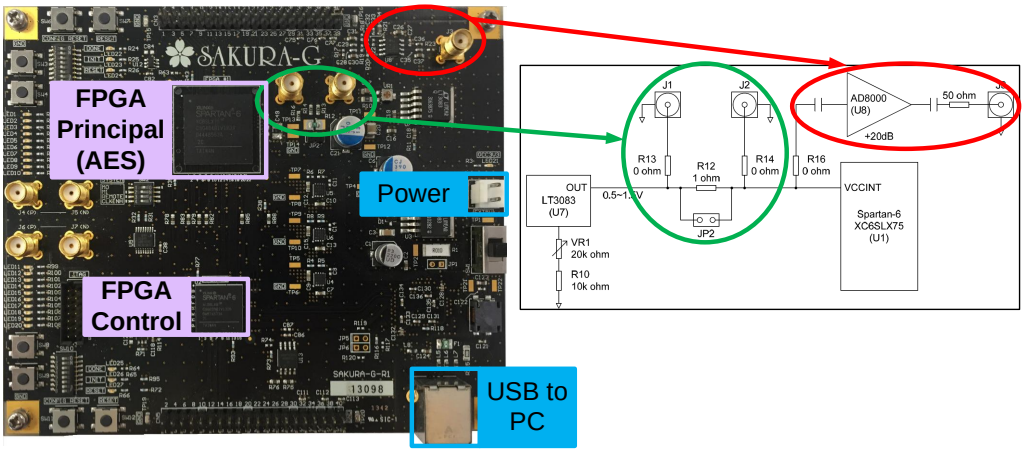

Figura 2.13.: Elementos PCB SAKURA.

podremos medir de forma precisa el consumo de potencia del cifrador durante la encriptación, obteniendo la clave secreta tras procesar los datos almacenados. A continuación explicamos con detalle los pasos y cada uno de los elementos que componen el ataque.

1. **FPGA (SAKURA-G)**

   Para la implementación del AES y comunicación con el ordenador vamos a utilizar la placa SAKURA-G.

   La placa SAKURA está especialmente diseñada para realizar ataques SCA. Dispone de dos FPGAs, una denominada como FPGA de control (una Spartan-6 XC6SLX9 de Xilinx) y la otra FPGA principal (Spartan-6 XC6SLX75 de Xilinx), tal y como se muestra en la Figura 2.13. En la FPGA de control se implementa la parte de la comunicación entre la PCB (*Printed Circuit Board*) y el ordenador (a través de un puerto USB). Además de ser la encargada de recibir los *plaintext* del ordenador y transmitir de vuelta los *ciphertexts* generados por la FPGA principal, se encarga de generar todas las señales de control para el bloque AES. La FPGA de control es, en resumen, la intermediaria entre el ordenador y la FPGA principal que contiene el cifrador.

La FPGA principal es la encargada de implementar el circuito criptográfico, en este caso, el algoritmo AES. Como se trata de un ejemplo de ataque, la implementación del AES utilizada es la proporcionada por los propios diseñadores de la placa SAKURA.

Para poder medir el consumo de la FPGA principal donde está implementado el AES, la SAKURA dispone de dos medios diferentes: podemos medir la diferencia de voltaje en los bornes de la resistencia R12, redondeada en color verde en la Figura 2.13, o podemos utilizar el amplificador indicado en la Figura 2.13 con color rojo. Como con la segunda opción solo necesitaremos un canal del osciloscopio y no precisa realizar ningún procesado adicional, tomaremos esta opción como la más adecuada para nuestro caso de estudio.

2. **Osciloscopio**

   Vamos a disponer de un osciloscopio (modelo Tektronik DPO3034 de 2.5GS/s) para medir las trazas de consumo de nuestro circuito. Como vamos a utilizar la salida del amplificador, necesitaremos dos sondas: una para el punto de medida de consumo de la FPGA principal, y otra para la señal de disparo.

3. **Fuente de alimentación**

   Se va a utilizar una fuente de alimentación externa (modelo TTi PL330DP) para alimentar la PCB y la FPGA SAKURA-G a 5V. La fuente, aunque programable, se va a conectar de forma manual al inicio del experimento debido a que no se necesita realizar ninguna variación en la configuración a lo largo del test.

4. **Ordenador**

   Utilizaremos el ordenador (Intel core i7 de 8GB de memoria RAM con sistema operativo Windows 7 Professional) para diferentes tareas. En primer lugar será el encargado de gestionar el paso de adquisición de trazas con un script. En una segunda parte, una vez almacenados todos los datos tras la encriptación, un segundo script lanzará el ataque sobre estos datos determinando la clave secreta.

   Para la adquisición de las trazas, se utiliza un script en Python. Con este script en primer lugar se realizará la configuración del osciloscopio y la conexión con

la placa SAKURA, como se muestra en la Figura 2.14. A continuación, se le aplicarán *plaintexts* aleatorios (en este experimento 5000) y se medirán las trazas de consumo durante la encriptación. Una vez finalizada la captura de datos, se almacenan los cipehrtexts y las trazas de consumo en dos matrices para su posterior procesado en Matlab. El proceso de medida de las 5000 trazas tarda en torno a 15 minutos.

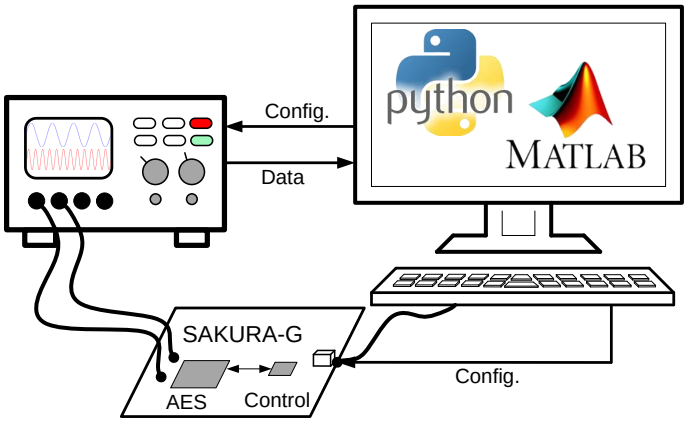

Figura 2.14.: Esquemático de ataque experimental.

Mediante un programa en Matlab, se leerán los datos almacenados de trazas de consumo y *ciphertext* y se procesarán para determinar la clave correcta. En primer lugar se ha seleccionado como punto de ataque la salida de 8 bits de la Sbox8 del AES en su última ronda. A continuación, se han medido las trazas de consumo de potencia durante la encriptación. En este punto, el script de Matlab carga las matrices de datos. Seguidamente determina a partir de los *ciphertext* aplicados y todas las posibles claves (256), todos los valores hipotéticos intermedios (en la Figura 2.12 la matriz B). Aplicando la distancia de Hamming, se calcula la matriz de valores hipotéticos de consumo del AES para cada encriptación y todas las posibles claves. Finalmente, se realizará la función de correlación entre las trazas de consumo medidas con los valores hipotéticos de consumo obtenidos. Aquella clave que muestre mayor valor de correlación destacando sobre las demás será la correspondiente a la

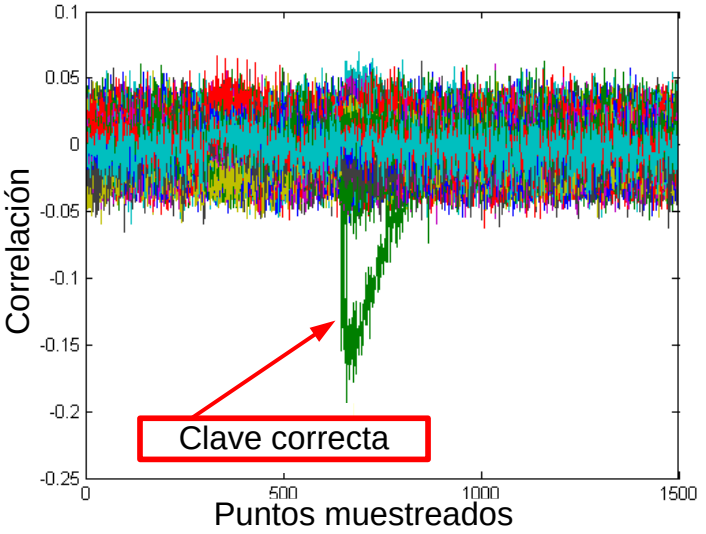

Figura 2.15.: Resultados del ataque al AES para el byte 0.

clave correcta. El ataque en Matlab tarda apenas 2 segundos para cada byte de la clave.

Este proceso en Matlab se repetirá 16 veces para cada byte de la matriz de clave, obteniendo finalmente todos los valores de k_0, k_1, ..., k_{15}. El ataque completo se realiza por tanto en torno a los 16 minutos.

Resultados del ataque DPA

Los resultados del ataque experimental sobre una implementación del AES en FPGA se muestran a continuación. En la Figura 2.15 se muestran los valores de correlación para cada una de las 256 posibles claves de un byte de la matriz de claves. Se puede apreciar que la clave correcta destaca claramente sobre las demás, que se mantienen dentro de un rango de aproximadamente ± 0.07. En la Figura 2.16 se pueden ver los valores de correlación en el punto de máxima correlación obtenido, en función

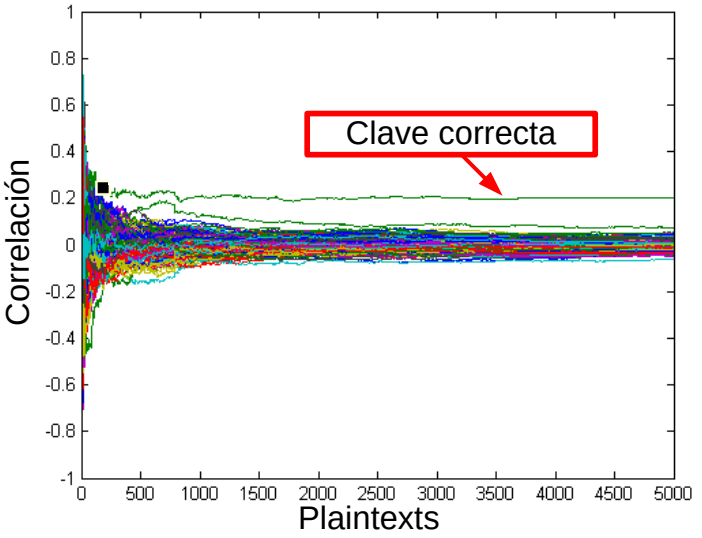

Figura 2.16.: Resultados del ataque al AES para el byte 0 en función de los *plaintext* aplicados.

del número de *plaintexts* aplicados. Al aplicar pocos patrones de entrada, no somos capaces de determinar cuál es la clave correcta. Sin embargo, aplicando en torno a los 200 *plaintexts*, conseguimos que la clave correcta se separe de las incorrectas mostrando un ataque satisfactorio.

Esto nos demuestra que, aunque el algoritmo AES es matemáticamente seguro, su implementación física en la FPGA nos da información a través del consumo de potencia durante la encriptación que nos sirve para determinar la clave secreta. Además de medir el consumo de potencia, se puede tener acceso a otro tipo de información como pueden ser los tiempos de ejecución o emisión electromagnética, entre otros. Como ejemplo de la efectividad de los ataques SCA, se muestra a continuación un breve ejemplo de ataque DEMA experimental sobre la misma implementación del AES.

2.3.3. Ejemplo de ataque DEMA

Si bien en el caso del ataque DPA medíamos el consumo de potencia de la FPGA principal, en este caso simplemente vamos a medir la emisión electromagnética durante la encriptación. El setup de medida y los scripts en Python y Matlab serán igualmente válidos en este ataque. Las únicas diferencias serán la forma y el punto de medida. Donde antes se medía con una sonda a la salida del amplificador de la SAKURA, ahora la vamos a reemplazar por una sonda electromagnética y la situaremos sobre la FPGA principal, tal y como se muestra en la Figura 2.17. Se modificarán ligeramente los datos de configuración del canal del osciloscopio para adecuarlos a la sonda electromagnética y se realizará el mismo ataque promediando en este caso las trazas de emisión electromagnética en el osciloscopio para mayor efectividad en el ataque.

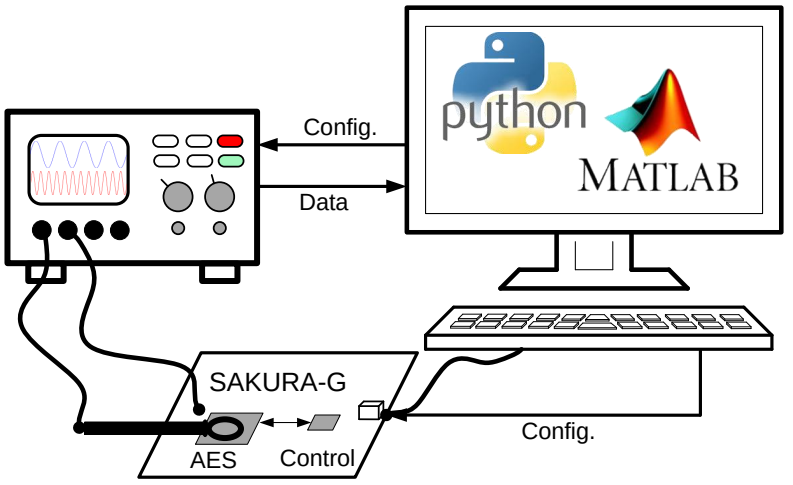

Figura 2.17.: Esquemático de ataque experimental con sonda electromagnética.

Aplicando los mismos patrones de entrada (5000), los resultados se muestran en las Figuras 2.18 y 2.19. Se puede apreciar nuevamente que el ataque es satisfactorio para el caso de ataque electromagnético. Es decir, los ataques SCA suponen un alto riesgo para la seguridad de nuestros dispositivos.

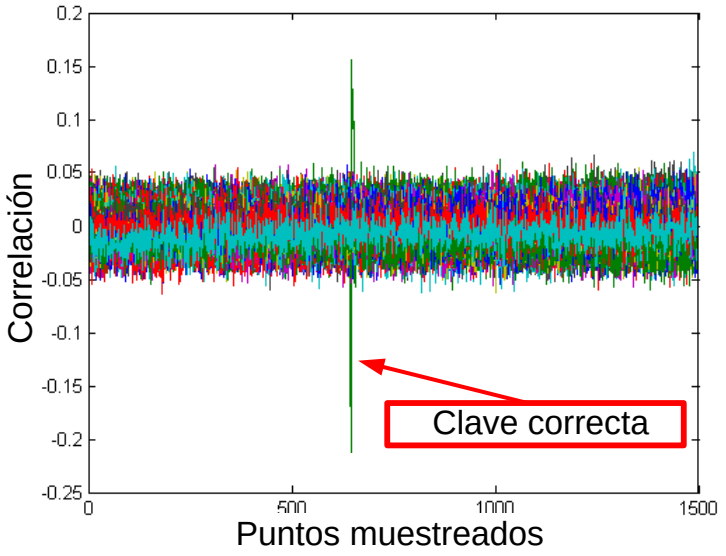

Figura 2.18.: Resultados del ataque EM al AES para el byte 0.

Con el desarrollo realizado, se ha cumplido el primer objetivo de la Tesis que consistía en implementar de forma experimental un ataque DPA viendo las vulnerabilidades que existen en los dispositivos criptográficos. Además, se ha comprobado que con una ligera modificación en el ataque, se pueden implementar ataques DEMA. Aunque el cumplimiento de este objetivo no supone un avance en el estado del arte, si nos permite contar con una metodología de medida de vulnerabilidad que se empleará a lo largo de la Tesis y en futuros trabajos.

2.3.4. Fortaleza comparativa del sistema de ataques al AES: variación de la temperatura

El estudio de los efectos de la variación de la temperatura en ataques DPA se muestran en CV. 8.

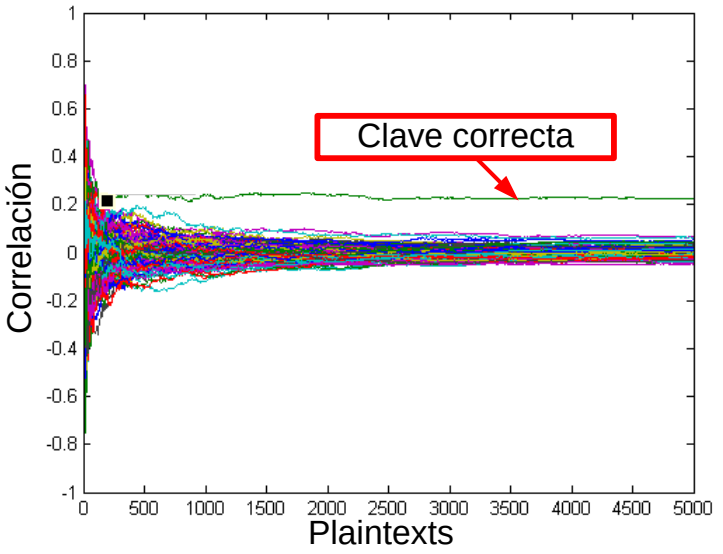

Figura 2.19.: Resultados del ataque EM al AES para el byte 0 en función de los *plaintext* aplicados.

2.3.5. Otros mecanismos de medida de vulnerabilidad: t-test

Los ataques DPA y DEMA permiten medir el nivel de seguridad alcanzado por los criptocircuitos frente ataques de canal lateral. Además, permiten comparar dos implementaciones diferentes entre si, y determinar cual de las dos es más segura. Sin embargo, la tarea de implementar el ataque es costosa en comparación con otras técnicas como es el t-test. El t-test es un análisis estadístico que nos permite determinar si una implementación tiene fugas de información [57, 64]. Al contrario de los ataques, no nos permite comparar dos implementaciones entre sí, ya que solo indica si hay fugas de información, no determina el nivel de dificultad que tiene un ataque, ni nos da indicaciones de cómo aprovechar esas fugas.

El t-test compara dos conjuntos de datos (pueden ser por ejemplo trazas de consumo de potencia ó electromagnéticas) y determina si son distinguibles entre si. Si el resultado de la comparación de los dos conjuntos de datos superan el umbral t fijado

a un valor de ±4.5, determinado para obtener una confianza de al menos un 99.99 %, quiere decir que los dos grupos de datos son distinguibles entre sí, y por tanto el criptocircuito es atacable al existir fuga de información.

Dentro del análisis de un t-test nos encontramos con dos tipos de test:

1. Test no específico, *"Fixed vs Random"*: se comparan dos conjuntos de datos, uno medido con un mismo patrón y clave fijas, y otro conjunto de datos con patrones aleatorios. Es el t-test más potente y determina si el dispositivo tiene fugas de información.
2. Test específicos: este caso se compone a su vez de diferentes tests, donde se comparan dos conjuntos de datos en una ronda de encriptación específica para patrones prefijados. Cada uno de los resultados de estos tests, da información más detallada del punto de fuga del criptocircuito.

Como ejemplo de t-test, vamos a implementar el *Fixed vs Random* con curvas de consumo de potencia, ya que solo buscamos probar la efectividad de este tipo de medidas, midiendo si la implementación del AES en FPGA muestra fugas de información sin determinar el punto exacto de la fuga. Para ello, comenzamos a medir dos conjuntos de datos definidos como [57]:

- DATA-SET1:
  - La clave K de 128 bits se fija a "0x0123456789abcdef123456789abcdef0"
  - Se van a realizar 2n encriptaciones, donde n es el número de trazas para cada test (100<n<5000 para tener una confianza del 99.99 %). Se tomará como primer *plaintext* el valor pt="0x00000000000000000000000000000000", siendo la salida cifrada la siguiente entrada al criptocircuito.
- DATA-SET2:
  - La clave K de 128 bits se fija a "0x0123456789abcdef123456789abcdef0"
  - Se medirán n encriptaciones fijando el *plaintext* siempre al valor "0xda39a3ee5e6b4b0d3255bfef95601890"

Tal y como se indica en [65], las trazas del DATA-SET2 se medirán intercaladas dentro de las medidas del DATA-SET1 para una n=5000 en nuestro caso. Una vez capturadas las trazas de consumo, se implementa el test *fixed vs random* de la siguiente forma. Se realizan dos test independientes:

- Test1: se realiza el t-test comparando las primeras n/2 trazas del conjunto de datos DATA-SET1 (A para el grupo1) vs las primeras n/2 trazas del DATA-SET2 (B para el grupo 1).
- Test2: se realiza el t-test comparando las últimas n/2 trazas del conjunto de datos DATA-SET1 (A para el grupo 2) vs las últimas n/2 trazas del DATA-SET2 (B para el grupo 2).

Para calcular el valor de t aplicamos lo siguiente a cada punto de la traza muestreada [57, 65]:

$$t = \frac{X_A - X_B}{\sqrt{\frac{S_A^2}{N_A} + \frac{S_B^2}{N_B}}} \tag{2.7}$$

Si el valor t obtenido a lo largo de la traza, tantos valores de t como puntos muestreados en la traza de consumo, supera en algún punto el valor de ± 4.5 significa que el test ha fallado y por tanto existe fuga de información. Si el resultado t estuviera dentro de los márgenes del umbral, significa que la implementación del AES es segura.

Como resultado, t tendrá una fila de longitud igual al número de puntos muestreados por traza, donde N_A y N_B son la longitud de los subgrupos A y B de cada uno de los test, X_A y X_B son el promedio del valor de los subgrupos A y B, y finalmente S_A y S_B es la desviación estándar de cada grupo A y B. Se calcula el valor de t en el test1 y en el test2, que se representan en las Figuras 2.20 y 2.21 respectivamente.

En la Figura 2.22 se muestran ambas gráficas superpuestas, viendo que en ambos casos el valor de t supera los umbrales de ± 4.5. Esto nos indica que la implementación del AES tiene fugas de información y por tanto no es segura. Tal y como habíamos determinado en los apartados anteriores, nuestra implementación del AES es vulnerable a ataques DPA y DEMA, por lo que resulta lógico el resultado obtenido en el t-test.

Como podemos ver, los resultados experimentales obtenidos sobre las medidas de vulnerabilidad sobre un cifrador AES implementado en una FPGA, muestran que la implementación física de los algoritmos criptográficos tienen fugas de información

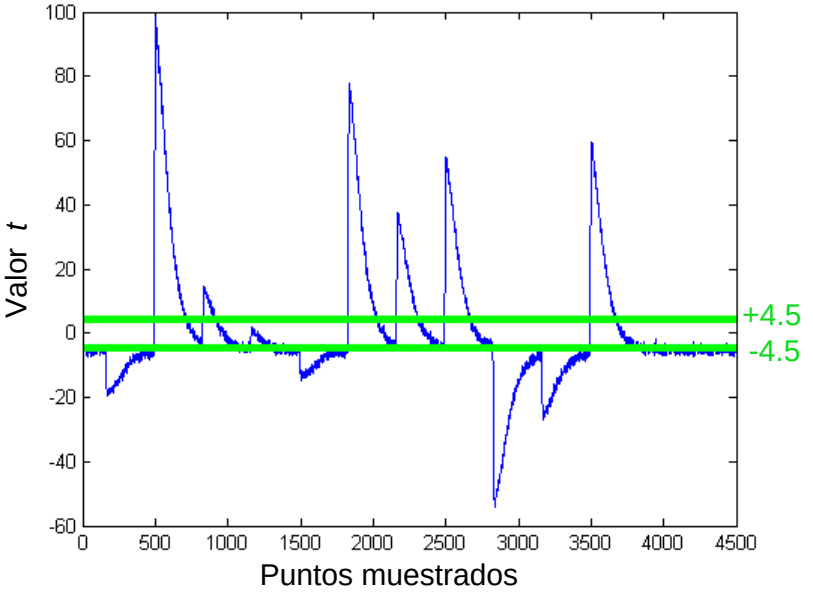

Figura 2.20.: Resultado del t-test para el test1 "*fixed vs random*".

que pueden utilizarse para revelar la clave secreta. Se han mostrado diferentes técnicas de ataque (DPA y DEMA), así como una medida de vulnerabilidad alternativa, el t-test. A la vista de los resultados hemos decidido que para comparar diferentes implementaciones vamos a optar por ataques DPA.

De la misma forma que se realiza un ataque experimental, se puede trasladar la estrategia de ataque a un entorno de simulación. De esta forma, se cambiaría el mecanismo de adquisición de las trazas de consumo, modificándolas por trazas de consumo provenientes de una simulación eléctrica, y se lanzaría el mismo script de ataque en Matlab para la recuperación de la clave. Esto nos va a permitir medir el nivel de seguridad de nuestras implementaciones en fases de diseño previas a la implementación física del dispositivo.

Por lo tanto, una vez analizado en profundidad un cifrador de bloque ampliamente estudiado, para seguir avanzando en el diseño de ataques DPA vamos a escoger un

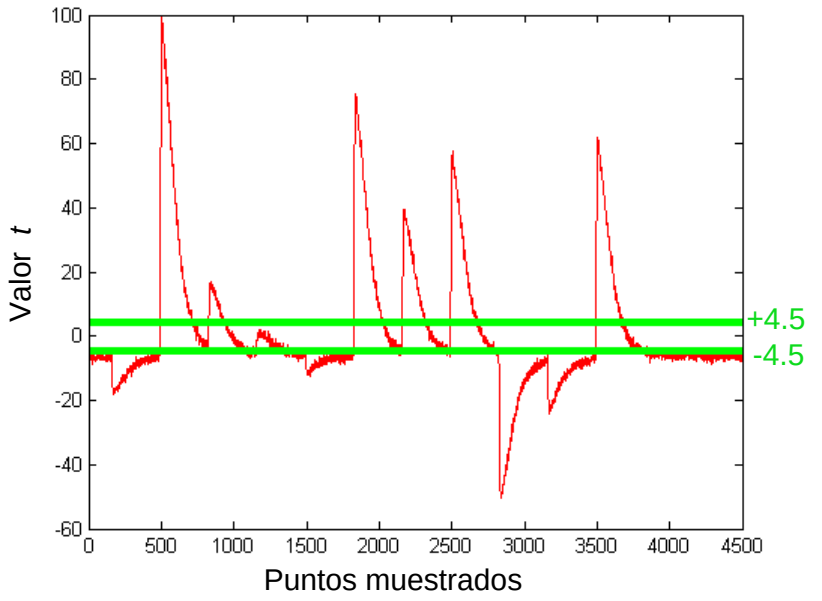

Figura 2.21.: Resultado del t-test para el test1 "*fixed vs random*".

nuevo cifrador que nos permita aprender más sobre este tipo de ataques. Como el AES es un cifrador de bloque, vamos a optar en este caso por un cifrador de flujo, Trivium, del que no se han reportado ataques experimentales y que además no tiene ataques teóricos que recuperan la clave completa sin tener que realizar ninguna hipótesis final [15, 53, 66].

2.4. Ataque DPA al cifrador de flujo Trivium

El estudio presentado en esta sección se muestra en CV. 16.

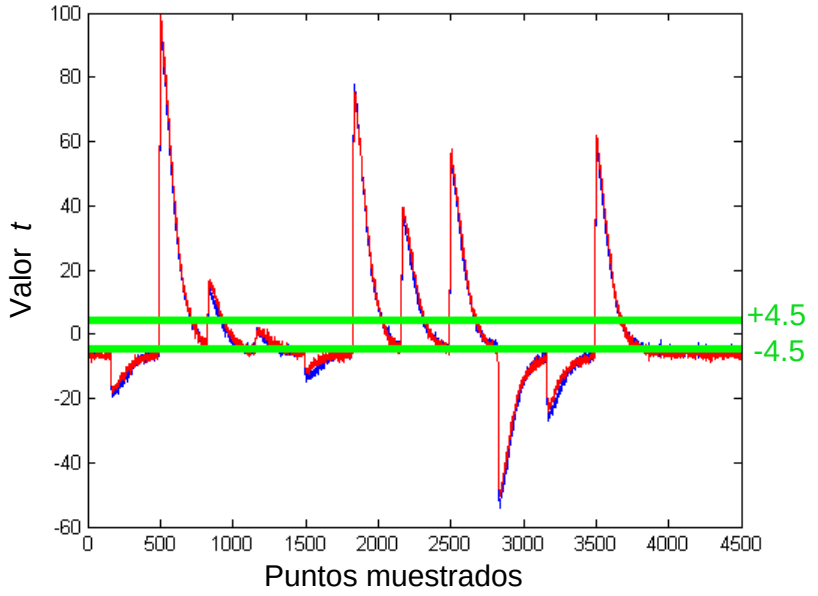

Figura 2.22.: Resultado del t-test para el test1 y test 2 "*fixed vs random*" superpuestos.

2.5. Optimización del ataque DPA al cifrador Trivium

El estudio presentado en esta sección se muestra en CV. 13.

2.6. Conclusiones

Para medir el nivel de seguridad de los circuitos criptográficos existen diferentes métricas y técnicas de evaluación. Para una primera fase de diseño se utilizarán medidas indirectas como son el NED y NSD que nos permiten tener una aproximación del nivel de seguridad esperado a través de una pequeña simulación eléctrica. Sin embargo, para determinar de forma precisa el nivel de seguridad alcanzado, es mejor

implementar otras técnicas como pueden ser ataques DPA/DEMA. El inconveniente de estas medidas es que no son inmediatas, hay que implementar el cifrador completo o la parte crítica, y además diseñar la estrategia de ataque. Si nos encontramos con un número elevado de posibles propuestas, esta tarea puede resultar inabordable y tendremos que optar en primer lugar por medidas como NED y NSD para realizar un primer filtrado de potenciales implementaciones seguras. El t-test nos ayuda a determinar si las propuestas son vulnerables o no.

Para entender las medidas de vulnerabilidad presentadas, se han propuesto de forma experimental un ataque DPA, un ataque DEMA y un t-test sobre el cifrador AES implementado sobre una FPGA. Tanto los ataques DPA como DEMA han resultado efectivos recuperando la clave en ambos casos. A su vez, el t-test nos ha mostrado que la implementación del AES no era segura ya que sobrepasaba los valores umbrales fijados. Aunque las tres medidas son válidas para determinar si un diseño es seguro, vamos a optar por implementar a partir de ahora ataques DPA por varios motivos. El primero de ellos es que los ataques DPA se pueden realizar de forma experimental o por simulación de forma relativamente sencilla, sin embargo los ataques DEMA por simulación serían muy costosos e inabordables. Por otra parte los ataques DPA nos permiten comparar dos implementaciones y determinar cual de ellas es más segura. En cambio, los t-test no son adecuados para implementarlos por simulación y solo determinan si una implementación es segura, es decir, no indican el nivel de seguridad alcanzado frente a un ataque y, por tanto, no permiten comparar una implementación con otra.

Para evaluar la robustez de los ataques se han realizado, a diferentes temperaturas y con variación de la clave, ataques DPA y DEMA sobre una implementación AES en FPGA, variando la temperatura en 10, 25, 50 y 70^{o}C, para dos claves K1 y K2 diferentes. Los resultados han mostrado que los ataques DPA son más efectivos que los DEMA, mientras que las variaciones de temperatura y clave no muestran una influencia decisiva. Hemos podido apreciar que para una misma temperatura, el nivel de seguridad alcanzado para dos claves diferentes varía, por lo que se aconseja que las medidas de seguridad se determinen para el mayor número de claves posibles promediando finalmente el resultado.

Una vez estudiados los ataques DPA en un cifrador de bloque ampliamente utilizado y atacado, pasamos a proponer nuevas estrategias de ataque, en este caso para

un cifrador de flujo: Trivium. Se ha planteado una nueva estrategia de ataque donde modelando exclusivamente el consumo de la puerta XOR previa al estado S_{94}, se consigue revelar la clave secreta sin realizar ninguna hipótesis. El ataque se ha implementado por simulación recuperando la clave correcta para varios ejemplos.

Finalmente, se ha propuesto una mejora para optimizar el ataque al Trivium, basada en la identificación del patrón resultante en las curvas de correlación. Con esta técnica, se consigue reducir el número de patrones aplicados en un 91.25 % si lo comparamos con la técnica original de detección del valor de máxima correlación.

3. Diseño de contramedidas DPA a nivel de transistor

La seguridad de un criptocircuito viene determinada por la elección del algoritmo criptográfico y la longitud de la clave, ya que los algoritmos tienen que ser matemáticamente seguros y utilizar una clave lo suficientemente larga para evitar ataques por fuerza bruta. Por ejemplo, en algoritmos *lightweight*, el NIST indica que claves con menos de 56 bits no son seguras debido al nivel computacional existente hoy en día y recomienda que el nivel de seguridad mínimo expresado en numero de bits sea de 112 bits para alcanzar un compromiso adecuado entre complejidad hardware y nivel de seguridad [10]. Pero ya se ha visto que debido a su implementación física puede haber fugas de información que pueden ser utilizadas por un atacante para revelar la clave secreta.

Desde el año 1999, cuando Kocher mostró que los ataques de potencia podían revelar la clave secreta de un dispositivo criptográfico de forma muy efectiva [30], se ha realizado un gran estudio sobre el diseño de diferentes contramedidas a diferentes niveles de abstracción. Debido a que los ataques de potencia son los más efectivos, han sido numerosas las propuestas de contramedidas presentadas frente a este tipo de ataques [2, 59, 67–70].

En este capítulo de la Tesis, vamos a proponer contramedidas frente a ataques DPA a nivel de transistor, tal y como se muestra en la Figura 3.1. Los resultados obtenidos en este capítulo se presentaron en [CV.1], [CV.5], [CV.17] y [CV.18].

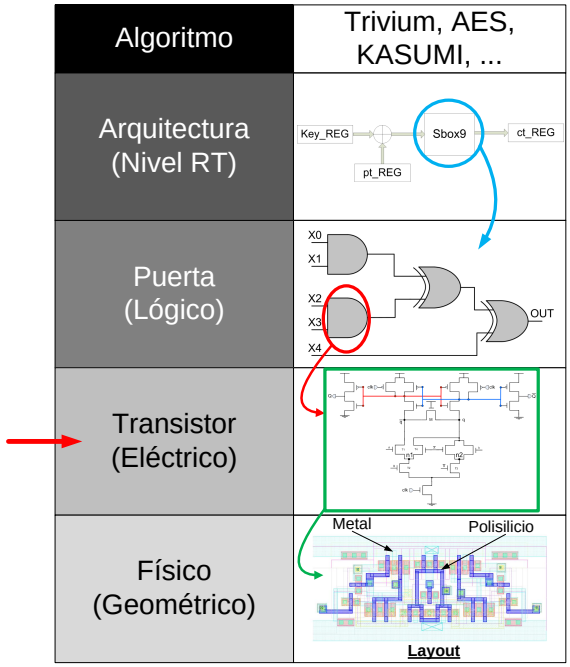

Figura 3.1.: Niveles de abstracción en cryptocircuitos: nivel de transistor.

3.1. Estado del arte de contramedidas DPA

El objetivo de las contramedidas DPA es desvincular el consumo de potencia del circuito con el dato procesado durante el encriptado/desencriptado, haciendo difícil (o imposible) adivinar la clave oculta con un esfuerzo razonable.

Hay numerosas formas de clasificar las contramedidas DPA, en esta Tesis las dividiremos esencialmente en dos grupos: técnicas de enmascaramiento *masking* y de ocultación *hiding* [2]. Dentro de cada uno de los grupos, se pueden clasificar a su vez en contramedidas *software* o contramedidas *hardware*, sin embargo, debido a la naturaleza de este estudio, dejaremos de lado las contramedidas *software* y nos centraremos únicamente en contramedidas *hardware*. A su vez, tanto las técnicas de

masking como las de *hiding* se podrán aplicar a nivel de arquitectura o a nivel e celda.

La idea básica de las técnicas de **masking** es aleatorizar los valores intermedios que son procesados por el dispositivo criptográfico. Para ejecutar las operaciones intermedias del criptocircuito, se utilizan máscaras m que se mezclan con los datos reales a procesar, desvinculando por tanto el consumo de potencia con el dato real procesado.

Estas técnicas permiten obtener consumos de potencia independientes del dato procesado sin tener que cambiar las características de consumo del circuito, ya que lo que se consigue es enmascarar el dato real procesado.

Las técnicas de *masking* se pueden aplicar a su vez a nivel de arquitectura o a nivel de celda, tal y como se muestra en la Figura 3.2. Las contramedidas de *masking* a nivel de arquitectura consisten en seleccionar una operación intermedia del dispositivo, donde el dato a procesar se mezcla con la mascara, de tal forma que la siguiente operación que se realice opere con un dato "transformado" en vez de procesar el dato real a cifrar, teniendo un consumo de potencia diferente al esperado. Las técnicas a nivel de celda sin embargo, no se focalizan en una operación en concreto, sino que se aplican a más bajo nivel, utilizando los *Masked Logic Styles* (estilos lógicos "enmascarados") [2] como pueden ser MDPL (*Masked Dual-rail Precharged Logic)* [71] e iMDPL (*improved Masked Dual-rail Precharged Logic)* [67]. En este caso, el criptocircuito se implementa utilizando celdas lógicas que utilizan una máscara asociada a cada una de las señales que procesa. La principal ventaja del uso de este tipo de celdas frente a contramedidas aplicadas a nivel de arquitectura, es que no son dependientes del algoritmo que se esté implementando, siendo aplicables de la misma forma independientemente del algoritmo utilizado pudiéndose automatizar.

Para mostrar un ejemplo del uso de técnicas de *masking* en resistencia frente a ataques DPA, el Apéndice A muestra un ataque DPA realizado al AES con contramedidas de *masking*, realizado colateralmente al desarrollo de esta Tesis.

Por otra parte, nos encontramos con las contramedidas de *hiding*, que también las podemos dividir en técnicas aplicadas a nivel de arquitectura o a nivel de celda, tal y como se muestra en la Figura 3.3.

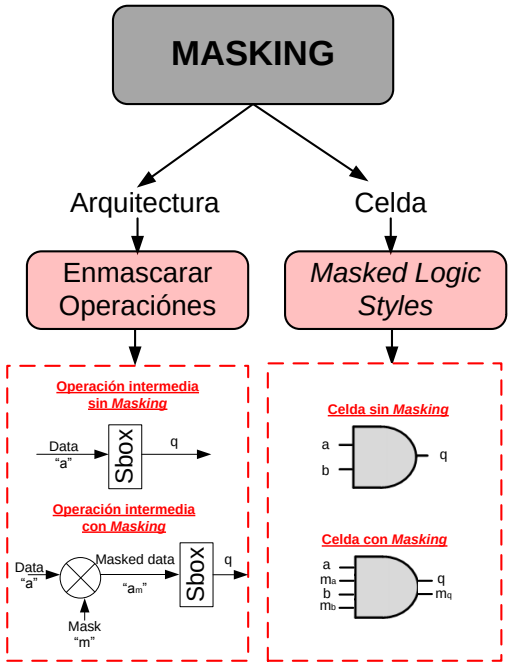

Figura 3.2.: Clasificación de contramedidas *masking*.

Las contramedidas de **hiding** buscan esconder el consumo real del criptocircuito haciéndolo independiente del dato procesado. Esto se puede lograr de dos formas: haciendo que el circuito consuma potencia de forma aleatoria o haciendo que el criptocircuito consuma siempre lo mismo en cada ciclo de reloj .

Para conseguir estos objetivos, dentro de las contramedidas de *hiding* a nivel de arquitectura, podemos realizar dos cambios en el criptocircuito: variar la temporización o variar la potencia. Para variar la temporización, por ejemplo, se pueden añadir operaciones aleatorias dentro del algoritmo, donde ya no se procesen las mismas operaciones que en el algoritmo original, obteniendo por tanto un consumo aleatorio en función de las operaciones realizadas. Para realizar variaciones en la potencia, podemos añadir por ejemplo un generador de ruido aleatorio que enmas-

care el consumo de potencia del algoritmo, haciendo que el consumo aportado por el generador de ruido sea superior al del algoritmo. Estas contramedidas sin embargo, son dependientes del algoritmo implementado o elevan enormemente el consumo de potencia y área del criptocircuito.

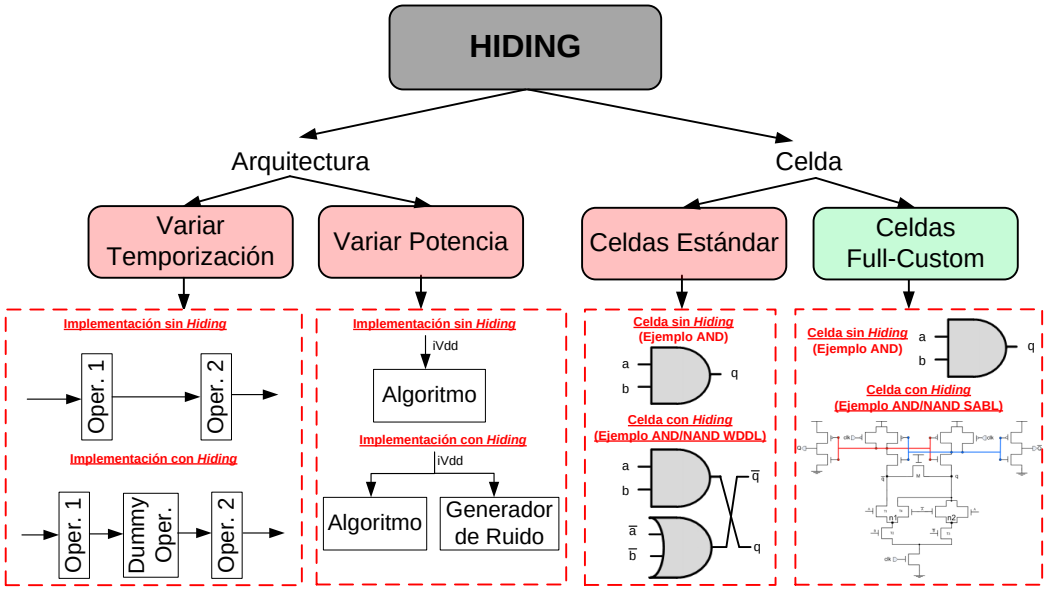

Figura 3.3.: Clasificación de contramedidas *hiding*.

Por otra parte, las técnicas de *hiding* aplicadas a nivel de celda se basan en la implementación de celdas lógicas con consumo de energía teóricamente independiente de los datos que se procesan. El diseño de este tipo de celdas seguras ha sido una obsesión permanente en la criptocomunidad, porque se pueden usar para la implementación de hardware asociado a cualquier tipo de algoritmo criptográfico, tanto para criptosistemas de clave pública o de clave privada, independientemente de la aplicación específica y de la técnica de circuito empleada. Hay varios enfoques para crear contramedidas de *hiding* a nivel de celda con codificación complementaria y consumo de energía independiente de los datos procesados. Aquellos basados en lógica adiabática, como por ejemplo [72], ofrecen seguridad con muy bajos niveles de consumo de potencia, pero los diseños adiabáticos requieren una temporización

precisa y necesitan un mayor desarrollo. Por otra parte, para maximizar los niveles de seguridad utilizando estilos lógicos más convencionales, se han propuesto las familias lógicas con doble raíl y precarga, conocidas como *Dual-Rail Precharge Logic* (DPL). Las familias DPL utilizan dos estados de funcionamiento, el de precarga (alcanzar un valor fijo después de cada evaluación) y evaluación (realización de la función lógica dependiendo de las entradas), y lógica de doble raíl (cada entrada y salida se codifica con dos bits: la señal a y su complementaria $\bar{a}$). Con las lógicas DPL se consigue tener siempre una conmutación a la salida en cada ciclo de reloj mostrando por tanto la misma actividad de conmutación independientemente del dato procesado [71, 73]. En este tipo de lógicas, un aspecto a tener en cuenta en la fase de diseño es la generación de un rutado diferencial para cada par de señales complementarias, tal y como se muestra en [42, 48, 60]

Dentro de las soluciones DPL podemos distinguir aquellas que se realizan con puertas CMOS standard y aquellas que emplean diseños *full-custom* [2, 74].

En la primera categoría, la de celdas CMOS estándar, la implementación clásica más utilizada es la llamada *Wave Dynamic Differential Logic* (WDDL) [68]. Este estilo lógico emplea pares de puertas NAND/NOR como primitivas para realizar las funciones combinacionales. En WDDL, los elementos secuenciales son los que están conectados a la señal de reloj, por lo que son los encargados de proporcionar la fase de precarga que se propaga desde las entradas hacia las salidas como una onda. La ventaja que proporciona este estilo lógico es el uso del flujo de diseño de celdas estándar, lo que facilita la síntesis del proceso. Se han presentado posteriormente mejoras sobre la WDDL, como pueden ser STTL (*Secure Triple Track Logic*) [75] y BCDL (*Balanced Cell-based Differential Logic*) [76]. Básicamente, las celdas seguras diseñadas utilizando este tipo de estilo lógico se incorporan fácilmente en diseños de criptocircuitos sobre FPGAs, pero ofrecen un nivel de mejora en cuanto a seguridad del dispositivo bastante baja. Incluso el uso de un proceso de back-end muy cuidadoso en el diseño de un ASIC no asegura que el dispositivo diseñado sea seguro. Para mayor referencia sobre implementaciones WDDL en ASIC y sus resultados frente ataques de canal lateral, ver [77].

Dentro del diseño de celdas diferenciales que requieren metodologías *full-custom*, aquellas basadas en estilos lógicos diferenciales son las que mejores niveles de seguridad ofrecen. Aunque estos estilos son originarios de los años 80, se ha documentado

que si tanto el diseño como el proceso de *place & route* se hace de forma correcta, estas soluciones brindan los mejores resultados para aplicaciones criptográficas. Las propuestas de familias lógicas más relevantes han sido la *Dynamic Current Mode Logic* (DyCML) [69], *Low-Swing Current Mode Logic* (LSCML) [70], *Sense Amplifier Based Logic* (SABL) [59], *Three-Phased Dual-Rail Pre-charge Logic* (TDPL) [78] y *Delay-Based Dual-Rail Precharge Logic* (DDPL) [79]. Los circuitos diferenciales explotan su simetría inherente para garantizar consumos similares para las evaluaciones de '0' y '1', ya que tanto las salidas como sus complementadas se generan simultáneamente.

La Figura 3.4 muestra un esquema simplificado para un estilo de lógica diferencial dinámica. Dichos estilos lógicos se componen de una red DPDN (*Differential Pull Down Network*) y una red DPUN (*Differential Pull Up Network*) que proporcionan tanto la salida Q como su complementada $\overline{Q}$ en cada ciclo de reloj, con solo un evento de carga, pero se debe tener cuidado para garantizar que se ponga en juego la misma cantidad de carga en cada transición:

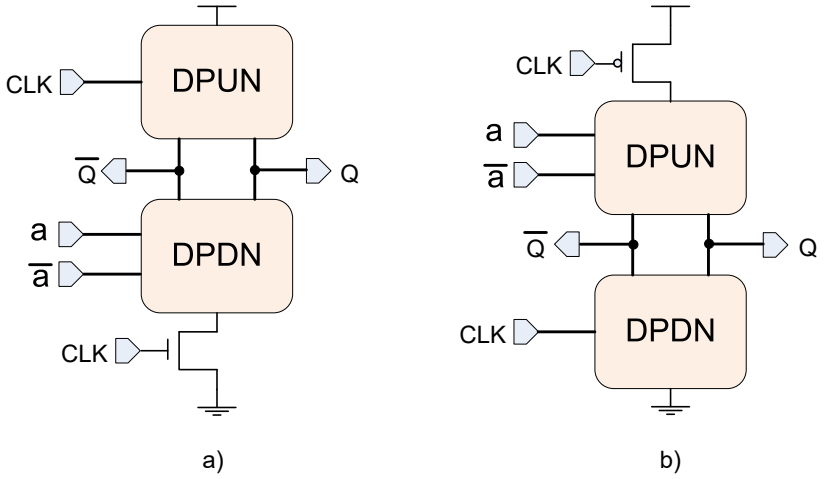

Figura 3.4.: Estilo lógico dinámico y de doble raíl: a) función lógica DPDN en transistores NMOS, y b) función lógica DPUN en transistores PMOS.

- **Bloque de control de las fases de precarga y evaluación:** es un bloque que consta de transistores NMOS y PMOS controlados por CLK que permiten fijar ambas salidas Q y $\overline{Q}$ siempre al mismo valor en la fase de precarga, y conectan cada una de las ramas diferenciales a la salida en la fase de evaluación. En la Figura 3.4-a, este bloque de control se identifica con el bloque DPUN. En este caso, las salidas Q y $\overline{Q}$ se fijan a valor Vdd en la fase de precarga. En el caso de la Figura 3.4-b, el bloque que implementa esta funcionalidad es el DPDN, que fija las salidas a valor Gnd en la fase de precarga.
- **Bloque funcional:** este bloque es el que implementa la función lógica diferencial de la celda. En el caso de la Figura 3.4-a, el bloque se corresponde con el DPDN y se implementa utilizando transistores NMOS. En el caso de la Figura 3.4-b, es el DPUN quien implementa este bloque y está compuesto por transistores PMOS.

Como las dos estructuras de la Figura 3.4 son igualmente válidas para el desarrollo de celdas seguras, hemos escogido una de ellas, desarrollando en este proyecto la estructura de la Figura 3.4-a.

Tabla 3.1.: Características y rendimiento de lógicas DPL full-custom.

	#trans.	**XOR/XNOR2**		**AND/NAND2**		**Inconvenientes**
	DPUN	E_{avg}	σ_E	E_{avg}	σ_E	
SABL [59]	12	1	1	1	1	Sensible a salidas desbalanceadas
DyCML [69]	13	1.976	$\approx$ 1e8	1.996	126.70	Sensible a salidas desbalanceadas Output swing reducido
LSCML [70]	13	1.508	$\approx$ 1e5	1.553	131.80	Sensible a salidas desbalanceadas Output swing reducido
TDPL [78]	14	1.405	1.28	1.333	0.80	Generación y routing de señales de control adicionales
DDPL [79]	8	1.301	0.66	1.251	0.24	Necesita convertidores de nivel y una temporización precisa

La comparación entre los estilos lógicos es compleja. En la Tabla 3.1 se pueden ver datos comparativos y un resumen de pros y contras de cada una de las familias DPL más destacadas. Las lógicas TDPL y DDPL son insensibles a las diferencias en las capacidades de rutado, pero la TDPL necesita una tercera fase de reloj, y la DDPL necesita un esquema de codificación de temporización que requiere una generación de retrasos precisa. Por otro lado, la SABL es un estilo de lógica diferencial que cumple con ambos requisitos: tiene un único evento de carga y carga la misma capacidad en cada transición. SABL logra mejores resultados porque su estructura interna suprime la influencia de las capacidades internas mejor que las lógicas DyCML y LSCML. Por lo tanto, vamos a tomar la lógica SABL como la estructura de referencia para el diseño de celdas DPL realizadas en esta Tesis.

3.2. Métricas para evaluación de seguridad y rendimiento de celdas DPL

Antes de plantear el diseño de celdas DPL resistentes a ataques de canal lateral, vamos a determinar qué métricas utilizar para medir el nivel de seguridad y rendimiento alcanzado. En la Figura 3.5 se muestran las medidas aplicables en cada fase de diseño. Para la medida del nivel de seguridad alcanzado por los circuitos, vamos a utilizar el NED y NSD en una primera fase al diseñar celdas individuales, mientras que una vez tengamos un bloque más complejo utilizaremos un ataque DPA con los resultados de MTD obtenidos.

Por otra parte, las métricas de rendimiento serán las mismas independientemente de la fase del diseño en la que nos encontremos. Tanto si son medidas experimentales como de simulación, o si se trata de medidas obtenidas sobre una celda individual o sobre un bloque más complejo, podremos medir el consumo de potencia (*Power* en la Figura 3.5), el retraso (*Delay* en la Figura 3.5) o la figura PDP (*Power Delay Product*), que se define como:

$$PDP = Power \cdot Delay \tag{3.1}$$

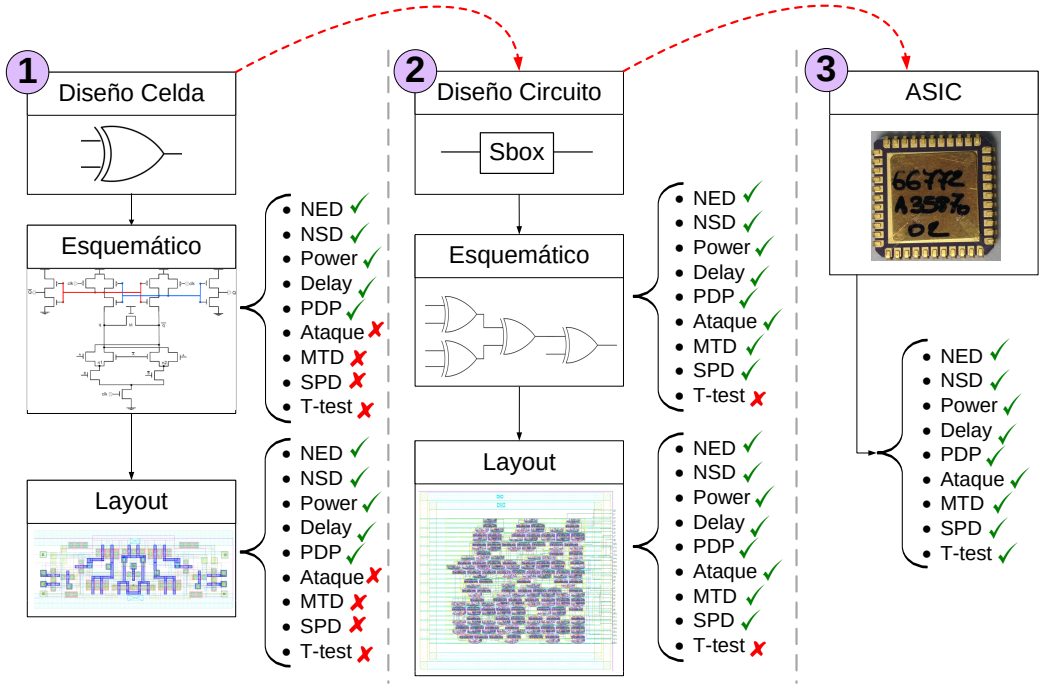

Figura 3.5.: Medidas de celdas DPL posibles de realizar según la fase de diseño. ✓ significa realizable y X no realizable.

En la literatura, las métricas calculaban la seguridad por un lado y el rendimiento por otro. En esta Tesis, se presenta una nueva métrica con la que poder obtener una relación de seguridad y rendimiento, la cual nos permite comparar diferentes implementaciones entre sí para alcanzar la mejor relación entre seguridad y prestaciones. Introducimos el SPD (*Security-Power-Delay* metric) [CV.1], que se define de la siguiente forma:

$$SPD = \frac{MTD}{Power \cdot Delay} \tag{3.2}$$

Tabla 3.2.: Métricas utilizadas para las propuestas incluidas en esta Tesis en cada fase de diseño

	NED	NSD	Potencia	Retraso	PDP	Ataque	MTD	SPD	T-test
Celda	✓	✓	✓	✓	✓	✗	✗	✗	✗
Circuito	✗	✗	✓	✓	✓	✓	✓	✓	✗
ASIC	✗	✗	✓	✓	✓	✓	✓	✓	✗

A mayor SPD, mejores prestaciones y seguridad alcanza nuestro diseño. En la Tabla 3.2 se muestra un resumen de las métricas utilizadas en esta Tesis para evaluar el nivel de seguridad y rendimiento alcanzado por nuestras propuestas para cada fase de diseño.

3.3. Diseño de celdas DPL resistentes a DPA

Para proponer y diseñar una celda lógica *full-custom* que sea resistente a ataques DPA, vamos a realizar un estudio de las características que debería cumplir en cuanto al consumo de potencia. Para ello realizamos el siguiente análisis. Se puede aproximar el consumo de potencia dinámica de una celda lógica como:

$$P_D = \alpha \cdot C_L \cdot f_{CLK} \cdot Vdd^2 \tag{3.3}$$

En la ecuación 3.3 tenemos dos parámetros que no dependen del dato que se esté procesando: la tensión de alimentación Vdd y la frecuencia de reloj f_{CLK}. Los otros dos parámetros, α (actividad de conmutación) y C_L (capacidades de la celda concentradas a la salida, donde tenemos capacidad de los nodos internos, conexionado y las capacidades de entrada de la etapa siguiente), sí dependen del dato procesado. Para conseguir que el consumo sea independiente del dato procesado, tenemos que intentar que los valores de α y C_L sean iguales para todas las transiciones posibles. Para ello se necesitan cumplir dos condiciones:

- La puerta tiene que tener exactamente un evento de carga por ciclo de reloj ($\alpha = 1$).
- La celda lógica tiene que cargar la misma capacidad en todos los eventos de carga.

Con un estilo lógico dinámico y diferencial se logra alcanzar $\alpha = 1$, teniendo dos fases de operación alternadas, una de precarga y otra de evaluación, y lógica de doble raíl. En DPL, cada variable booleana a está representada por un par de cables a, $\bar{a}$. Cada cómputo comprende una precarga (donde $a = \bar{a}$), seguida de una evaluación (donde $a \neq \bar{a}$, siendo complementarias). En la fase de precarga, las salidas Q y $\overline{Q}$ toman el mismo valor durante toda la precarga. En la evaluación, debido a que la lógica es diferencial, una de las ramas de salida conmutará de acuerdo con la función lógica que se implemente, quedando la otra con el valor complementario. Es decir, cualquiera que sea la situación de entrada, exactamente uno y solo uno de a y $\bar{a}$, y Q y $\overline{Q}$ cambiará. El número de conmutaciones por ciclo de reloj es, por lo tanto, constante. De esta forma en cada ciclo de reloj siempre habrá una conmutación a la salida y $\alpha = 1$.

Para cumplir con el segundo punto, hay que asegurar que la capacidad en cada nodo sea la misma en cada uno de los eventos de carga de cada ciclo de reloj. Para ello, la celda tiene que ser completamente simétrica teniendo los nodos de salida balanceados. Es decir, la cantidad de carga puesta en juego en cada transición tiene que ser igual, independientemente de la rama diferencial que conmute, para toda combinación de entradas. Las estructuras diferenciales tienen dos salidas, a las que les corresponden una carga capacitiva total que es debida a la capacidad de salida de la puerta, la capacidad de las interconexiones y las capacidades internas de la puerta. El objetivo es conseguir la máxima simetría en las redes DPDN y DPUN.

3.3.1. Desarrollo del bloque DPUN

El estilo lógico que consideramos más idóneo debido a las características que ofrece para aplicaciones criptográficas tiene que ser dinámico y diferencial. Pero para que sea realmente útil para este tipo de aplicaciones, hay que añadir que los nodos de salida estén balanceados, necesitándose una máxima simetría y por tanto la capacidad C_L sea la misma en la ecuación 3.3 para toda transición de entrada. Esto es

necesario porque si los nodos de las ramas diferenciales que unen la red DPDN con la DPUN están directamente conectados con los nodos de salida, como se muestra en la Figura 3.4, hacen que la capacidad de carga en los nodos de salida dependa de la red DPDN. En el caso de que las capacidades totales de los nodos de cada rama diferencial fuesen iguales, independientemente de la rama que conmute, el consumo de potencia siempre sería el mismo sean cuales sean las entradas que se estén procesando. Como las ramas diferenciales de la red DPDN no van a poder ser siempre simétricas debido a la complejidad de la función que implementen, hay que desarrollar algún mecanismo que iguale la tensión de los nodos de salida de la red DPDN al finalizar cada fase de precarga y evaluación, y evitar vestigios de carga que provoquen diferencias entre los consumos de cada transición de entrada.

Estilo lógico SABL

El estilo lógico denominado SABL [59], basado en el flip-flop StrongArm110 [80], es el que mejor cumple con los requisitos expuestos: es un estilo lógico dinámico y diferencial, y tiene los nodos de salida diferenciales balanceados.

En la Figura 3.6 se muestra la estructura a nivel de transistor de una celda SABL. Su funcionamiento es el siguiente y se muestra el caso de una simulación de un inversor con estructura SABL (DPDN mostrado en la Figura 3.7) realizado en una tecnología TSMC de 90nm, a 1.2V en la Figura 3.8.

Para la fase de precarga tendremos el valor de clk='0', en este caso el transistor T11 está cortado mientras que los transistores T6 y T8 están conduciendo fijando a '1' los nodos q, $\overline{q}$, azúl ($\overline{w}$) y rojo (w) de la Figura 3.6. Ambas salidas Q y $\overline{Q}$ tomarán el valor de '0' debido a que las entradas de los inversores de salida (T1 y T2 para la salida $\overline{Q}$, y T3 y T4 para la salida Q) estan a valor '1'.

Con clk='1', estaremos en la fase de evaluación. En este caso el transistor NMOS T11 conectado a tierra estará conduciendo, haciendo que solo una de las ramas de la red DPDN se descargue dependiendo de la función lógica que implemente y del valor de las entradas, que serán efectivamente complementadas, ya que estas son diferenciales. En este punto vemos que el estilo lógico SABL cumple con las condiciones de ser

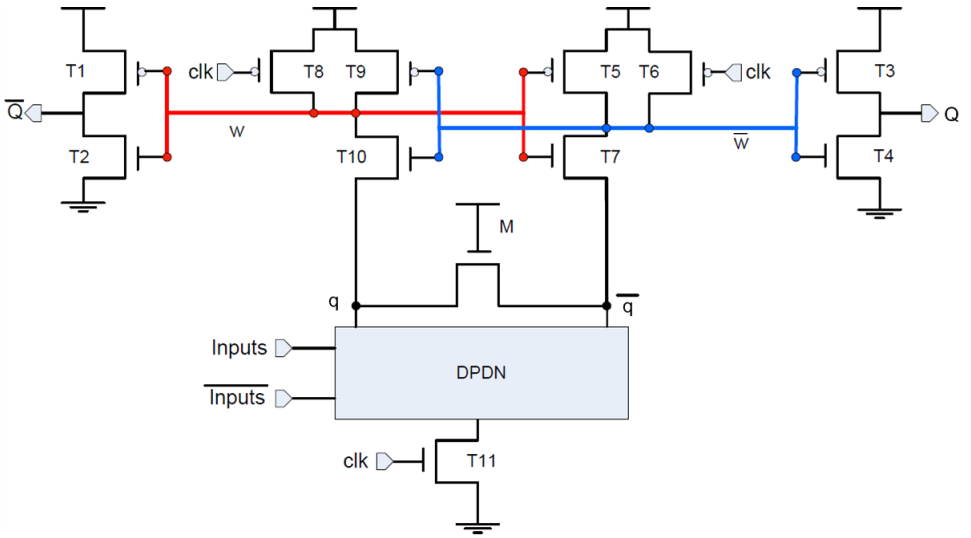

Figura 3.6.: DPUN del estilo lógico SABL.

un estilo lógico dinámico y diferencial. Para cumplir con la condición de simetría, nos fijamos en el transistor M que une los nodos q y $\overline{q}$, los que conectan la red DPDN con la DPUN, que siempre está encendido, permite crear un camino que evita tener nodos flotantes garantizando que todos los nodos internos se descarguen. Independientemente de qué rama diferencial de la red DPDN esté conduciendo, todos los nodos internos y sus respectivas capacidades estarán conectadas a través de M y se descargarán junto con uno de los nodos de salida. Con esto conseguimos eliminar la información acerca de la conmutación que tuvo lugar con anterioridad eliminando un posible efecto memoria que pueda ser usado por un atacante. De esta forma tendremos a la salida siempre $Q\overline{Q}$ = ”01” ó $Q\overline{Q}$ = ”10” en la fase de evaluación. Podemos ver que a la salida se incluyen dos inversores, de forma que puedan funcionar en modo dominó a la hora de conectar varias puertas en cascada.

De todas las propuestas existentes en la literatura, hemos escogido el estilo lógico denomidado SABL por ser el que mejores resultados muestra en cuanto a seguridad. En cambio, debido a la complejidad del estilo lógico nos encontramos con un aumento considerable en área y consumo de potencia respecto a otras estructuras diferenciales

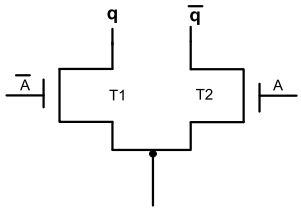

Figura 3.7.: Esquemático de la red DPDN del Inversor.

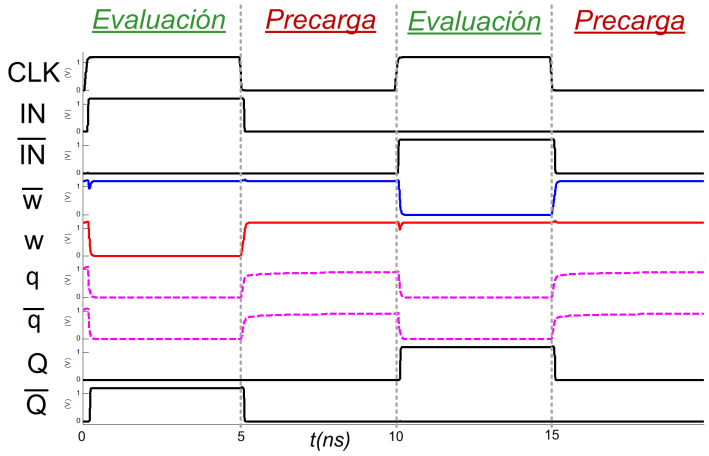

Figura 3.8.: Simulación del inversor SABL.

y, por supuesto, respecto a la lógica estática convencional. En [81], se estudiaron alternativas a la lógica SABL para poder implementar celdas con la misma seguridad que la SABL pero reduciendo el consumo de potencia y área, proponiendo como alternativa la *Dynamic Differential Cascode Voltage Switch Logic* (DDCVSL) [82].

Estilo lógico DDCVSL

En la Figura 3.9 se muestra la estructura de la DDCVSL. El funcionamiento de la DDCVSL es el siguiente y se muestra en la Figura 3.10 para el caso de un inversor DDCVSL. Al igual que en el caso SABL, nos encontramos en la fase de precarga con

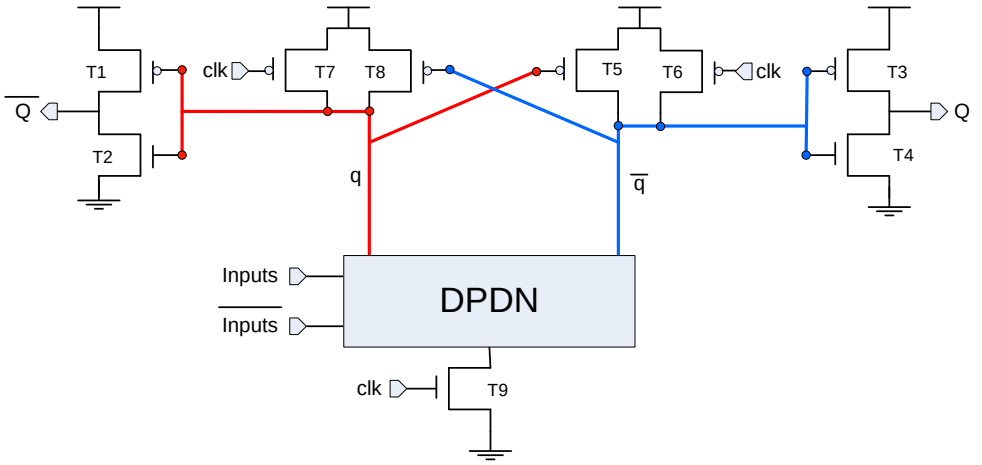

Figura 3.9.: DPUN del estilo lógico DDCVSL.

clk=’0’. En este caso, los transistores T6 y T7 están conduciendo fijando las ramas roja (q) y azul ($\overline{q}$) de la Figura 3.9 a ’1’. De esta forma las salidas Q y $\overline{Q}$ estarán a ’0’ debido a los inversores de salida formados por T1-T4. Como en la fase de precarga el transistor T9 está cortado, no habrá ninguna rama conduciendo a tierra. En caso de la evaluación, nos encontramos con clk=’1’. Ahora, el transistor T9 está conduciendo, haciendo que una de las ramas diferenciales se descargue dependiendo de la función lógica que implemente y obteniendo por tanto los valores $Q\overline{Q}$ = ”01” ó $Q\overline{Q}$ = ”10” en la fase de evaluación.

Debido a que la implementación DDCVSL es más simple que la SABL, se consiguen reducir tanto el consumo como el área, pero a costa de perder características necesarias para garantizar la seguridad.

Se puede apreciar cómo con esta estructura no tenemos los transistores T7, T10 y M de la estructura SABL. Esto supone el no tener salidas balanceadas y tener las ramas diferenciales del DPDN directamente conectadas a la salida. Esto implica que la capacidad de carga dependerá del dato procesado siempre que la red DPDN no sea completamente simétrica. Además, al eliminar el transistor M, perdemos el

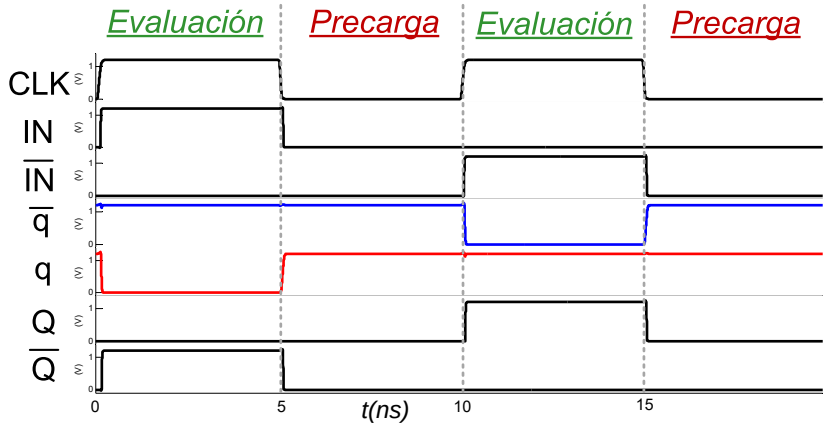

Figura 3.10.: Simulación del inversor FLDS.

camino de descarga que permitía conectar los nodos internos pudiendo quedar nodos flotantes.

En esta Tesis vamos a trabajar con las redes DPUN SABL y DDCVSL. Con la primera opción buscamos la mayor seguridad posible a costa de un mayor consumo de potencia y área, mientras que con la red DDCVSL, queremos analizar el nivel de seguridad alcanzado reduciendo el área y consumo proporcionado, en relación a la SABL.

3.3.2. Desarrollo del bloque DPDN

Aunque la red DPUN sea óptima en cuanto a seguridad, hay que eliminar cualquier variación de consumo debidas a asimetrías de las ramas diferenciales de la red DPDN. Además, como es el caso de la DDCVSL, el contar con el diseño de una red DPDN totalmente simétrica es de vital importancia para alcanzar los niveles de seguridad requeridos por las aplicaciones criptográficas resistentes a ataques de canal lateral [1].

En el DPDN, una y sólo una de las dos ramas se activará conectando a tierra uno de los nodos q o $\overline{q}$, por lo que una de las ramas genera el valor de la función de la celda, mientras que la otra implementa la función complementaria. Para que la descarga

de ambas ramas sean idénticas, la capacidad equivalente de la rama que descarga q tiene que ser la misma que la de $\overline{q}$. Por ello, todas las capacidades que son internas de la propia puerta tienen que ser iguales en las dos ramas diferenciales. Por lo que se deben intentar cumplir los siguientes requisitos:

- Tener la mayor simetría posible entre las dos ramas diferenciales.
- Tener el mismo número de transistores en serie por rama, independientemente de las entradas que se procesen en cada evaluación.
- Conectar los nodos internos de las ramas diferenciales a los nodos de salida q y $\overline{q}$ mediante transistores en ON.
- Tener la salida de cada rama conectada al mismo número de transistores.

Cumpliendo esto, conseguimos tener la misma equivalencia RC en cada una de las ramas, independientemente de las entradas que se estén procesando en el circuito.

El caso ideal sería la solución alcanzada en el diseño del inversor. La red DPDN se muestra en la Figura 3.11-a, donde al solo tener un transistor por rama, atacados en su puerta por la entrada y su complementada, tenemos total simetría entre las dos ramas diferenciales, ademas de contar con el mismo número de transistores en serie y las dos ramas de salida están cargadas por el mismo número de transistores, no teniendo en este caso ningún nodo interno que descargar. Por tanto, para explicar mejor la complejidad del desarrollo de la red DPDN, vamos a escoger las funciones XOR/XNOR y AND/NAND de dos entradas mostradas en la Figura 3.11-b y Figura 3.11-c respectivamente, que son celdas básicas que se usan habitualmente en el diseño de dispositivos criptográficos.

El bloque DPDN de la puerta XOR/XNOR en la Figura 3.11-b cumple con los requisitos que hemos mencionado anteriormente: las dos ramas son simétricas, tiene el mismo número de transistores por rama independientemente de las entradas que se procesen en cada ejecución (2), las dos salidas q y $\overline{q}$ están cargadas por 2 transistores y los nodos internos n1 y n2 siempre tienen un camino de conexión en conducción con q y $\overline{q}$. En la Tabla 3.3 se muestra detallado el funcionamiento de la red DPDN XOR/XNOR. Se puede apreciar que:

$$q_{XOR} = A \cdot B + \overline{A} \cdot \overline{B} \tag{3.4}$$

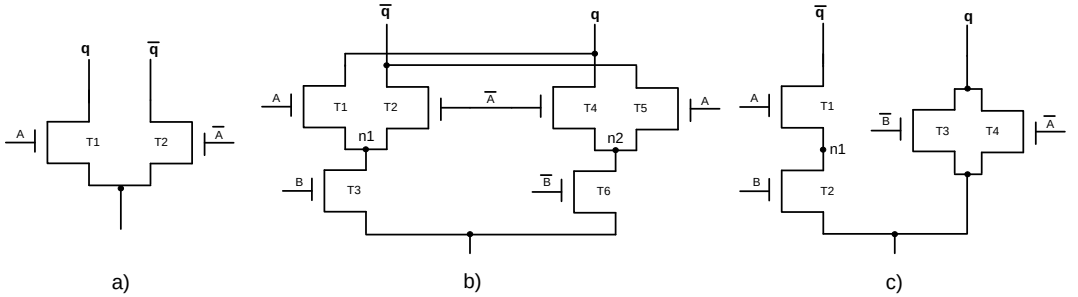

Figura 3.11.: Red DPDN inicial para los casos: a) inversor/buffer, b) XOR/XNOR, y c) AND/NAND.

$$\overline{q}_{XOR} = \overline{A} \cdot B + A \cdot \overline{B} \tag{3.5}$$

Tabla 3.3.: Funcionamiento de la red DPDN XOR/XNOR

A	**B**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**n1**	**n2**	**q**	$\overline{\mathbf{q}}$
0	0	OFF	ON	OFF	ON	OFF	ON	$\overline{q}$	q	0	1
0	1	OFF	ON	ON	ON	OFF	OFF	$\overline{q}$	q	1	0
1	0	ON	OFF	OFF	OFF	ON	ON	q	$\overline{q}$	1	0
1	1	ON	OFF	ON	OFF	ON	OFF	q	$\overline{q}$	0	1

Pero en el caso de la puerta AND/NAND, mostrada en la Figura 3.11-c, apreciamos asimetría entre la dos ramas. Originalmente la función AND/NAND se construye con solo 4 transistores y se puede constatar el incumplimiento de todas las normas planteadas anteriormente. Las ramas no son simétricas, dependiendo del dato de entrada nos encontramos con que tenemos 1 ó 2 transistores en serie en ON por rama, el nodo interno n1 queda flotante en algunas combinaciones de datos de entrada, y por último la salida q está conectada a 2 transistores, mientras que la salida $\overline{q}$ está

conectada a 1. En la Tabla 3.4 se muestra detallado el funcionamiento de la red DPDN AND/NAND de la Figura 3.11-c. En este caso tenemos:

$$q_{AND} = \overline{A} + \overline{B} \tag{3.6}$$

$$\overline{q}_{AND} = A \cdot B \tag{3.7}$$

Tabla 3.4.: Funcionamiento de la red DPDN AND/NAND de la Figura 3.11-c

A	**B**	**T1**	**T2**	**T3**	**T4**	**n1**	**q**	$\overline{\mathbf{q}}$
0	0	OFF	OFF	ON	ON	-	0	1
0	1	OFF	ON	OFF	ON	-	0	1
1	0	ON	OFF	ON	OFF	$\overline{q}$	0	1
1	1	ON	ON	OFF	OFF	$\overline{q}$	1	0

Para mejorar la estructura del DPDN en los casos de mayor asimetría, en [1] se propone una metodología que permite mejorar el diseño de la red DPDN para aplicaciones criptográficas. Primero hay que conseguir que todos los nodos internos, en este caso n1, esté conectado a una de las ramas q o $\overline{q}$. Para ello se hace la modificación mostrada en la Figura 3.12 aplicando la ley de absorción del álgebra de Boole a la rama q_{AND} dejando igual la rama $\overline{q}_{AND}$:

$$q_{AND} = \overline{A} + \overline{B} = \overline{A} \cdot B + \overline{B} \tag{3.8}$$

A continuación, se añaden los transistores *dummy* necesarios para que las ramas diferenciales tengan el mismo número de transistores en serie por transición. Para ello, se añadirá una llave de paso en el camino de descarga que tenga un menor

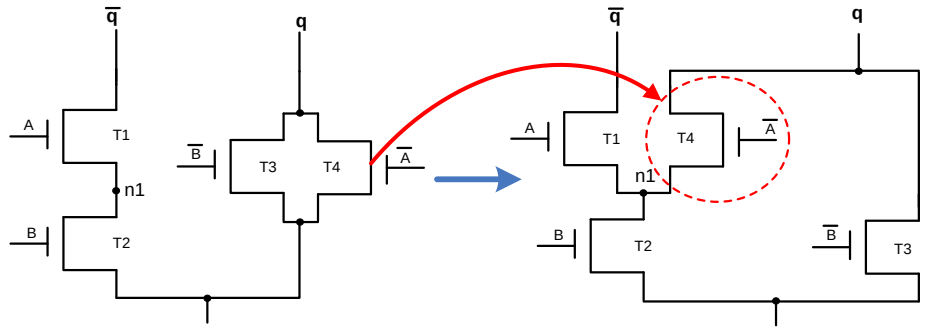

Figura 3.12.: Red DPDN AND/NAND modificada para conectar el nodo n1 a q [1] .

número de transistores en serie. En el caso de la puerta AND/NAND se aplicará el siguiente cambio a la rama q_{AND}, tal y como se muestra en la Figura 3.13:

$$q_{AND} = \overline{A} + \overline{B} = \overline{A} \cdot B + (A + \overline{A}) \cdot \overline{B} \tag{3.9}$$

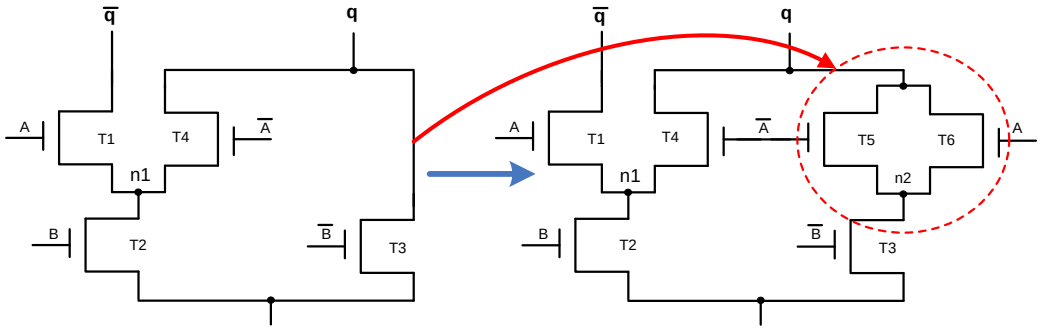

Figura 3.13.: Red DPDN AND/NAND modificada para tener el mismo número de transistores en serie por cada rama diferencial [1] .

En la Tabla 3.5 se muestra detallado el funcionamiento de la red DPDN AND/NAND de la Figura 3.13.

Tabla 3.5.: Funcionamiento de la red DPDN AND/NAND de la Figura 3.13

A	**B**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**n1**	**n2**	**q**	$\overline{\mathbf{q}}$
0	0	OFF	OFF	ON	ON	ON	OFF	q	q	0	1
0	1	OFF	ON	OFF	ON	ON	OFF	q	q	0	1
1	0	ON	OFF	ON	OFF	OFF	ON	$\overline{q}$	q	0	1
1	1	ON	ON	OFF	OFF	OFF	ON	$\overline{q}$	q	1	0

Estas son las mejores soluciones en cuanto a simetría del circuito planteadas hasta la fecha, aunque como se puede observar en el caso de la puerta AND/NAND, no se puede conseguir que el número de transistores en la rama q_{AND} sea igual que en la rama $\overline{q}_{AND}$.

Aunque se consiga la máxima simetría, se ha constatado que las 4 características exigibles al bloque DPDN no son suficientes para obtener el máximo nivel de seguridad. En los DPDN de la Figura 3.11-b y 3.13 los nudos internos de las dos ramas (n1 y n2) pueden guardar información de la evaluación anterior, aún teniendo un diseño totalmente simétrico de las dos ramas diferenciales. Esta información puede ser usada por un atacante para así revelar la clave secreta de nuestro dispositivo criptográfico. Es por ello por lo que cualquier tipo de información almacenada, o memoria del circuito, tiene que ser eliminada tras cada evaluación.

3.4. Vulnerabilidades de las celdas DPL

El estudio desarrollado en esta sección se ha presentado en CV. 5, CV. 17 y CV. 18.

3.5. Metodología de optimización del DPDN

El estudio desarrollado en esta sección se ha presentado en CV. 5, CV. 17 y CV. 18.

3.6. Metodología de diseño para celdas de más de dos entradas

El estudio desarrollado en esta sección se ha presentado en CV. 1.

3.7. Medida del nivel de seguridad de celdas DPL seguras frente ataques DPA

En las secciones anteriores se han presentado dos metodologías de diseño de celdas DPL: una para eliminar el efecto memoria en los nodos internos de las ramas diferenciales del DPDN, y otra para el diseño de celdas DPL con fan-in superior a 2.

Una vez escogidas las implementaciones óptimas de cada metodología, el número de diseños de los bloques criptográficos se reduce lo suficiente para hacer un análisis abordable. Como el uso de puertas de *fan-in* superior a 2 está vinculado a la optimización lógica que se analiza en el Capítulo 4, dejaremos para ese capítulo la verificación de la seguridad mediante ataques DPA. Es por ello por lo que en este apartado vamos a presentar los resultados del ataque DPA para las implementaciones propuestas y determinar el nivel de seguridad alcanzado por las nuevas propuestas respecto a las implementaciones de referencia.

3.7.1. Selección del cifrador

Un ataque DPA se puede realizar sobre el algoritmo completo, como se ha presentado en el Capítulo 2, o sobre una parte de él. Como nuestro objetivo actual no es avanzar en nuevas técnicas de ataque, sino probar el nivel de seguridad alcanzado con las nuevas propuestas, vamos a implementar sólo parte del algoritmo. Para ello, en la literatura se ha optado por el ataque a bloques Sbox aislados, al ser la parte más vulnerable de un cifrador de bloque frente ataques DPA. Si conseguimos que la Sbox aislada sin tener ninguna componente adicional de ruido en las medidas de corriente, que sería el escenario ideal para un atacante, sea segura, conseguiremos

que esta robustez se extienda a la hora de implementar el cifrador completo. Además, el diseño de las Sbox es mucho menos costoso que el diseño del cifrador completo, ya que son bloques combinacionales sencillos. Dependiendo del algoritmo, se utilizan diferentes construcciones de Sbox, así como diferentes números de entradas y salidas.

Para probar nuestros diseños se escogió la Sbox9 que utiliza el algoritmo KASUMI [83] como caso de estudio. En la Figura 3.14 se muestran las ecuaciones de la Sbox9.

$$
\begin{aligned}
y0 =& x0x2 \oplus x3 \oplus x2x5 \oplus x5x6 \oplus x0x7 \oplus x1x7 \oplus x2x7 \oplus x4x8 \oplus x5x8 \oplus 1\\
y1 =& x1 \oplus x0x1 \oplus x2x3 \oplus x0x4 \oplus x1x4 \oplus x0x5 \oplus x3x5 \oplus x6 \oplus x1x7 \oplus x2x7 \oplus x5x8 \oplus 1\\
y2 =& x1 \oplus x0x3 \oplus x3x4 \oplus x0x5 \oplus x2x6 \oplus x3x6 \oplus x5x6 \oplus x4x7 \oplus x5x7 \oplus x6x7 \oplus x8 \oplus x0x8 \oplus 1\\
y3 =& x0 \oplus x1x2 \oplus x0x3 \oplus x2x4 \oplus x5 \oplus x0x6 \oplus x1x6 \oplus x4x7 \oplus x0x8 \oplus x1x8 \oplus x7x8\\
y4 =& x0x1 \oplus x1x3 \oplus x4 \oplus x0x5 \oplus x3x6 \oplus x0x7 \oplus x6x7 \oplus x1x8 \oplus x2x8 \oplus x3x8\\
y5 =& x2 \oplus x1x4 \oplus x4x5 \oplus x0x6 \oplus x1x6 \oplus x3x7 \oplus x4x7 \oplus x6x7 \oplus x5x8 \oplus x6x8 \oplus x7x8 \oplus 1\\
y6 =& x0 \oplus x2x3 \oplus x1x5 \oplus x2x5 \oplus x4x5 \oplus x3x6 \oplus x4x6 \oplus x5x6 \oplus x7 \oplus x1x8 \oplus x3x8 \oplus x5x8 \oplus x7x8\\
y7 =& x0x1 \oplus x0x2 \oplus x1x2 \oplus x3 \oplus x0x3 \oplus x2x3 \oplus x4x5 \oplus x2x6 \oplus x3x6 \oplus x2x7 \oplus x5x7 \oplus x8 \oplus 1\\
y8 =& x0x1 \oplus x2 \oplus x1x2 \oplus x3x4 \oplus x1x5 \oplus x2x5 \oplus x1x6 \oplus x4x6 \oplus x7 \oplus x2x8 \oplus x3x8
\end{aligned}
$$

Figura 3.14.: Ecuaciones de la Sbox9.

La Sbox9 se puede construir como un bloque combinacional con 9 bits de entrada y 9 bits de salida, que está construido con operaciones lógicas AND y XOR. En las ecuaciones podemos ver los bits de entrada definidos por X (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) y los bits de salida Y (y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8) .

Una vez vista la Sbox que se va a utilizar como caso de estudio, se van a analizar cada una de las implementaciones presentadas.

3.7.2. Planteamiento del ataque DPA a Sbox9

El ataque DPA a la Sbox9 sigue los pasos presentados en el Capítulo 2, que se muestra detallado en la Figura 3.15. En primer lugar seleccionamos un punto intermedio del algoritmo donde intervengan el plaintext y la clave. En este caso, el punto de ataque es la salida de la Sbox, debido a que el valor lógico en este punto depende de un plaintext conocido y de la clave que queremos averiguar. El esquema implementado para la simulación es el mostrado en la Figura 13 en CV. 5, donde la entrada a la Sbox es el resultado de una operación XOR del plaintext y la clave, elemento que componen la mayoría de cifradores de bloque.

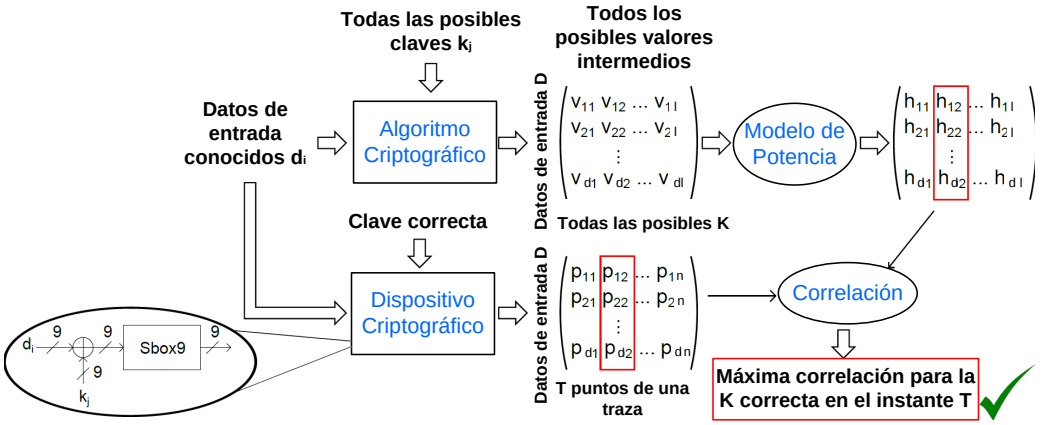

Figura 3.15.: Flujo de ataque DPA sobre la SBox.

A continuación, se miden las trazas de consumo del sistema (en este caso la corriente i_{Vdd} de la Sbox) durante la encriptación de los patrones de entrada. Almacenando el valor de los patrones de entrada aplicados en simulación, se calcula teóricamente el hipotético consumo de potencia que tendría nuestra Sbox si se encriptaran dichos plaintexts para todas las posibles claves usando un modelo basado en la distancia de Hamming. Una vez que se tienen los valores medidos del consumo de potencia y los calculados teóricamente para todas las posibles claves, se realiza una función de correlación entre los dos conjuntos de datos, dando como resultado la clave correcta aquella con mayor valor de correlación. Es decir, la traza de consumo hipotético que mayor correlación tenga con la medida por simulación será aquella que compute con la misma clave secreta.

Para comparar el nivel de seguridad alcanzado por cada implementación se utilizará como métrica el MTD.

3.7.3. Ataque DPA a las optimizaciones *single-switch* y *dual-switch*

El estudio desarrollado en esta subsección se ha presentado en CV. 5, CV. 17 y CV. 18.

3.8. Conclusiones

En este capítulo se ha realizado una revisión y propuestas de mejora de contramedidas basadas en lógicas DPL frente ataques DPA, además de presentar las métricas para la evaluación del nivel de seguridad y prestaciones de los criptocircuitos. En una primera fase de diseño se ha optado por medidas como el NED y NSD para poder descartar aquellas soluciones que no cumplan un mínimo de seguridad, mientras que para fases del diseño más avanzadas se ha optado por la implementación de ataques DPA sobre cifradores completos o partes críticas de ellos. Se presenta como nueva métrica el SPD que nos permite evaluar el compromiso entre seguridad y prestaciones (potencia y retraso).

Las soluciones que mejores niveles de seguridad alcanzan dentro de las lógicas DPL, siendo la SABL la que mejor compromiso ofrece entre seguridad y prestaciones, deben cumplir los siguientes requisitos:

- La puerta tiene que tener exactamente un evento de carga por ciclo de reloj.
- La celda lógica tiene que cargar la misma capacidad en ese evento de carga.

Por otra parte, el DPUN tiene que tener las siguientes características:

- Tener la mayor simetría posible entre las dos ramas diferenciales.
- Tener el mismo número de transistores en serie por rama, independientemente de las entradas que se procesen en cada ejecución.
- Conectar los nodos internos de las ramas diferenciales a los nodos q y $\overline{q}$.
- Tener la salida de cada rama cargada por el mismo número de transistores.

Sin embargo, se ha propuesto también la lógica DDCVSL para la parte del DPUN, que aún no cumpliendo con todos los puntos anteriores (sobre todo en los casos en las que las ramas diferenciales del DPDN no sean totalmente simétricas como es el caso de la celda AND/NAND), ofrece mejoras en cuanto a área y consumo al utilizar las soluciones *Dual/Single-switch* en la red DPDN.

Se ha demostrado que aún cumpliendo los requisitos anteriores para las redes DPUN y DPDN, existen fugas de información debidas a los nodos internos de la red DPDN. Para solucionar este problema se propone una metodología de diseño que busca

igualar los voltajes en estos nodos internos en la fase de precarga. Las dos soluciones propuestas son:

- Solución *Single-switch*: usar un transistor PMOS que una los nodos internos de la red DPDN igualando el voltaje entre los dos nodos.
- Solución *Dual-switch*: usar un transistor PMOS en cada nodo interno que en la fase de precarga fije los nodos a valor Vdd.

En los resultados de simulación de las medidas del nivel de seguridad a través de ataques DPA, se ha mostrado que las soluciones *single-switch* y *dual-switch* aumentan el nivel de seguridad de las celdas respecto a las de referencia en un factor de al menos x29.07. El inconveniente es un incremento en área y consumo. Si bien las SABL con soluciones *single-switch* para el caso de la celda AND/NAND y la *dual-switch* para el caso de la XOR/XNOR muestran niveles superiores de seguridad respecto a las DDCVSL, el uso de esta red DPUN puede resultar de gran interés si se busca un consumo menor con un nivel de seguridad menor (los resultados para la celda DDCVSL AND/NAND no mostraron en la primera fase de diseño valores que mejoraran a la SABL Clásica).

Por otra parte, se ha presentado una metodología de diseño del DPDN para celdas DPL con más de 2 entradas, estudiando qué implementaciones son las más adecuadas para cada caso considerando un nivel (o más) de puertas. Tras realizar simulaciones eléctricas de las celdas de 2, 3 y 4 entradas para diferentes combinaciones, se ha podido observar que a medida que aumentamos el *fan-in* de las puertas, el retraso y consumo de potencia aumenta, tal y como era de esperar. Sin embargo, con el uso de celdas de 3 y 4 entradas el número de transistores utilizados, y por ende el área ocupada, se reduce, siendo preferible el uso de celdas de *fan-in* de 3 y 4 entradas en vez de la combinación de celdas de 2 entradas en la medida de lo posible para obtener mejoras en cuando a prestaciones se refiere.

4. Diseño de contramedidas DPA a nivel lógico y mejora de prestaciones

El diseño de contramedidas para circuitos criptográficos frente ataques de canal lateral puede aplicarse a diferentes niveles de abstracción. Si bien en el capítulo anterior nos centrábamos en aplicar la seguridad a nivel de transistor, ahora pasamos a estudiar el nivel de seguridad alcanzado con propuestas aplicadas a nivel lógico o de puerta, ver Figura 4.1.

En las últimas décadas, se ha realizado un esfuerzo enorme para aumentar y analizar la seguridad de las implementaciones *lightweight* de algoritmos criptográficos, incorporando también contramedidas SCA [CV.2, CV.3]. A pesar de la enorme gama de contramedidas desarrolladas, está claro que se debe trabajar más para proporcionar soluciones más seguras y eficientes desde el punto de vista del consumo de potencia y seguridad, dado el continuo avance en las estrategias y efectividad de los ataques. Cuando se diseñan criptocircuitos de bajos recursos, se debe garantizar una maximización de la seguridad y las prestaciones con un coste razonable. Sin embargo, estos tres aspectos, seguridad, prestaciones y coste, son fáciles de cumplir de dos en dos, pero muy difíciles de cumplir juntos.

En este capítulo, se proponen diferentes métodos para intentar cumplir la mayor parte de los requisitos necesarios para implementaciones seguras, intentando mejorar tanto el consumo de potencia, como la frecuencia de operación, sin degradar otros aspectos importantes de los criptocircuitos, como la seguridad. Los resultados obtenidos en este capítulo se presentaron en [CV.1], [CV.10].

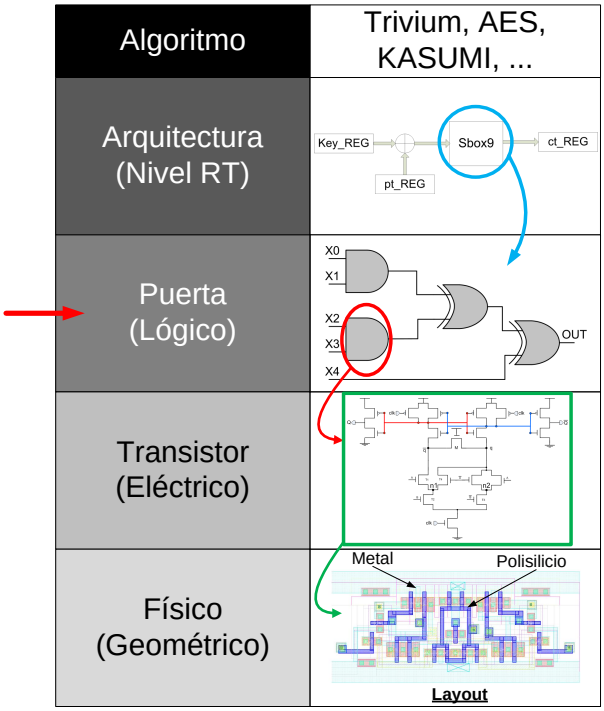

Figura 4.1.: Niveles de abstracción en criptocircuitos: nivel de puerta.

4.1. Trabajo previo

Las contramedidas propuestas en el capítulo anterior consistían en la implementación de celdas lógicas DPL que conseguían teóricamente el mismo consumo de potencia con independencia del dato procesado, en cada ciclo de reloj. De esta forma, procesando las trazas de consumo no se revela información alguna de la clave que está utilizando el criptocircuito. Estas soluciones tienen como objetivo principal reducir la correlación entre el consumo con los datos procesados focalizándose en el consumo entre ciclos de ejecución (en ingles "*inter-cycle*"), es decir, haciendo que en cada ciclo de reloj el criptocircuito consuma la misma cantidad de energía. Sin

embargo, la variación del consumo de potencia dentro de un mismo ciclo (en ingles "*intra-cycle*") no se contempla.

En [84], se estudia el efecto de variaciones *intra-cycle* en el consumo de potencia y su dependencia con los datos procesados. Se afirma que la resistencia de un circuito criptográfico frente a ataques DPA aumenta con la desalineación entre las diferentes transiciones de nodos de salida. Cuando se implementa físicamente un dispositivo criptográfico, las señales de salida no conmutan simultáneamente porque cada transición se produce en un instante de tiempo que depende de factores como las conmutaciones y pendientes de las señales de entrada, las capacidades de rutado y capacidades de salida, el tamaño de la puerta y el número de puertas por *path*, entre otros. La seguridad alcanzada por el criptocircuito frente a ataques de canal lateral se ve afectada por la hipótesis de la potencia consumida simultáneamente por el circuito tras conmutar varias salidas a la vez (por ejemplo, 9 bits de salida en la Sbox9 y 7 bits de salida en la Sbox7). Por ese motivo, la desalineación de las transiciones de los bits de salida hacen que el consumo de potencia de las mismas se vea distribuido a lo largo del tiempo, y la hipótesis realizada no sea la correcta debido a que esta contempla que el consumo de todos los bits de salida se realiza en un mismo instante de tiempo. Además, debido a la naturaleza de esta desalineación, el atacante no puede determinar los instantes de tiempo en que tendrán lugar las transiciones de cada bit de salida ya que no tiene información sobre la implementación física del algoritmo criptográfico.

Estas desalineaciónes se pueden inducir de forma externa como es el caso de [84], donde incluyen elementos externos que retrasan las señales de los bits de salida de forma aleatoria para provocar una desalineación en los tiempos de llegada, o pueden deberse a retrasos inducidos por las técnicas de optimización utilizadas por la mayoría de las herramientas de síntesis automática, como son la minimización lógica o el uso de puertas con diferente *fan-in*. La primera opción implica introducir elementos externos a la implementación original que incrementan tanto el consumo como el área del criptocircuito. Por ese motivo, vamos a centrarnos en las técnicas de minimización lógica y el uso de celdas de diferente *fan-in*, para ver cuál es el impacto que provocan en los niveles de seguridad alcanzados.

4.2. Impacto de técnicas de minimización lógica y uso de puertas con diferente *fan-in*

Las técnicas de minimización lógica se aplican en la mayoría de las herramientas de síntesis automática de diseño electrónico (del inglés *Electronic Design Automation*, EDA), optimizando las figuras de area-retraso-potencia para el diseño de ASICs de propósito general. Sin embargo, el diseño optimizado de circuitos con celdas DPL queda excluido en las herramientas EDA y, lo que es más importante, dichas herramientas no contemplan el aspecto más importante a considerar en los criptocircuitos: la seguridad. Las técnicas de optimización habituales pueden tener efectos negativos o positivos en la seguridad de las funciones implementadas frente a ataques DPA, por ese motivo, vamos a estudiar de forma independiente el impacto causado por la minimización lógica y el uso de puertas de diferente *fan-in*, como las diseñadas en el Capítulo 3.

El estudio desarrollado en esta sección se ha presentado en CV. 1.

4.3. Arquitectura de criptocircuitos seguros para aplicaciones de alta velocidad

Aunque la frecuencia máxima de operación o *throughput* no suele ser el factor limitante en los diseños criptográficos, vemos interesante el estudio de arquitecturas maximizando estas características de velocidad de cifrado. Además de esto, hay que mantener los niveles de seguridad requeridos para aplicaciones criptográficas seguras. Por esta razón, es interesante el realizar un estudio de la utilización de técnicas de implementación de alto rendimiento temporal así como diseños seguros con contramedidas frente a ataques de canal lateral [85–87].

Centrándonos en implementaciones de dispositivos electrónicos de altas prestaciones, hay numerosos trabajos que estudian el uso de *pipelining* muy fino para aumentar la frecuencia de operación, siendo esta un área de tremendo interés [88–91]. Estos estudios se centran en circuitos lógicos que por sí solos, pueden implementar arquitecturas *pipeline* sin el uso de elementos de memoria, como por ejemplo la lógica dinámica con sus fases de precarga y evaluación. Por ejemplo, en [88] se analiza el

uso de lógica domino aplicando técnicas de *pipelining*, donde se utiliza un esquema de reloj multifásico, y sin utilizar elementos de memoria adicionales.

Mediante el uso de **celdas lógicas dinámicas**, podemos construir circuitos lógicos con ***pipeline* sin elementos de memoria** adicionales, obteniendo beneficios considerables en velocidad, gracias al uso de *pipelining* muy agresivo.

En este subapartado de la Tesis, nos centraremos en el diseño de hardware de alta velocidad de dispositivos criptográficos con contramedidas para prevenir ataques pasivos no invasivos, donde señalaremos algunos aspectos importantes a tener en cuenta para poder conseguir el mejor compromiso en términos de seguridad y prestaciones.

El estudio desarrollado en esta sección se ha presentado en CV. 10.

4.4. Medida del nivel de seguridad de contramedidas aplicadas a nivel lógico

La variación en la temporización de los criptocircuitos tras aplicar técnicas de minimización lógica, el uso de celdas de *fan-in* elevado o el nivel de *pipelining*, implica cambios en las prestaciones del circuito e, indudablemente, en los niveles de seguridad frente ataques DPA. En general, si el circuito resultante tiene una mayor sincronización el los tiempos de llegada de los bits de salida, nos encontraremos con resultados de seguridad peores que en los casos en que las llegadas se produzcan en diferentes instantes de tiempo, a tenor de lo sugerido en [84].

Para poder evaluar de forma independiente cada una de estas técnicas, vamos a proponer diferentes casos de estudio, donde se implementaran de forma experimental parte de los criptocircuitos para ver cuál es el impacto real en el nivel de seguridad alcanzado. Para estas implementaciones, utilizaremos diferentes celdas propuestas en el capítulo anterior, ya que las contramedidas propuestas a nivel de transistor se pueden superponer a las técnicas de minimización lógica, *fan-in* o *pipelining*, ya que están a diferentes niveles de abstracción siendo por tanto técnicas complementarias.

4.4.1. Ataque DPA a las implementaciones con minimización lógica y diferente *fan-in*

El estudio desarrollado en esta subsección se ha presentado en CV. 1.

4.4.2. Ataques DPA a las implementaciones de alta velocidad

El estudio desarrollado en esta subsección se ha presentado en CV. 10.

4.5. Conclusiones

Tras evaluar diferentes contramedidas a nivel de transistor en el capítulo 3, donde nos centrábamos en aumentar la seguridad de las puertas diseñadas a costa de aumentar ligeramente el consumo, área y retraso, pasamos a intentar aumentar la seguridad a nivel lógico así como la mejora de prestaciones. Como las técnicas propuestas en ambos capítulos se aplican a diferentes niveles de abstracción, estas pueden aplicarse en un mismo diseño, siendo ambas complementarias.

En primer lugar se han estudiado los efectos de la minimización lógica y el uso de celdas con *fan-in* superior a 2. Se ha demostrado cómo las variaciones en la distribución de retrasos en los caminos de los criptocircuitos puede afectar a los niveles de seguridad frente a ataques DPA. Esto se debe a que la desincronización de los tiempos de llegada de las transiciones de los bits de salida afecta negativamente a la seguridad, ya que en los ataques DPA *multi-bit* el atacante hace una hipótesis del consumo del criptocircuito para un instante de tiempo en concreto, suponiendo que todos los bits a los que ataca cambian o hacen un aporte al consumo de potencia en el mismo instante de tiempo. Si el aporte al consumo de potencia se dispersa a lo largo del tiempo debido a los diferentes instantes de llegada de las transiciones de los bits de salida, la efectividad del ataque disminuye, y por tanto la seguridad aumenta.

El uso de minimización lógica y celdas con *fan-in* superior a 2, provoca variaciones en la temporización que pueden afectar de forma negativa o positiva a la seguridad y dependerán del circuito a implementar. Sin embargo, se consiguen en ambos casos

Tabla 4.1.: Resumen de los factores de multiplicación respecto a la Sbox9.

Sbox	MTD	Potencia Avg.	Retraso	Duty Cycle	SPD	Area
Sbox9_Minimized	x2.15	x0.62	x1.13	x0.75	x3.06	x0.61
Sbox9_FanIn	x1.92	x0.84	x1.10	x0.88	x2.09	x0.86
Sbox9_All	x6.58	x0.51	x1.17	x0.75	x10.89	x0.53

Tabla 4.2.: Resumen de los factores de multiplicación respecto a la Sbox7.

Sbox	MTD	Potencia Avg.	Retraso	Duty Cycle	SPD	Area
Sbox7_FanIn	x0.60	x0.86	x1.01	x0.94	x0.70	x0.9
Sbox7_All	x2.04	x0.55	x1.23	x0.39	x3.02	x0.57

mejoras en cuanto a área y consumo de potencia. Para evaluar estos efectos, se han realizado diferentes implementaciones de la Sbox9 y Sbox7 que nos permiten evaluar de forma independiente el uso de celdas XOR de *fan-in* superior a 2, el uso de celdas AND con *fan-in* superior a 2, el uso de la minimización lógica, y la aplicación de ambas técnicas a la vez, junto con el uso de celdas DPL diseñadas en el Capítulo 3. Para evaluar los niveles de seguridad, se han realizado ataques DPA a todas las implementaciones. Como resumen, en la Tabla 4.1 y 4.2 se muestran el factor de multiplicación para cada característica (MTD, potencia, retraso, *duty-cycle*, SPD y área) respecto a las implementaciones de referencia (*Sbox*9 y *Sbox*7) respectivamente.

Se puede observar cómo la mayor seguridad se obtiene al combinar ambas técnicas tanto para la Sbox9 como Sbox7. Se puede apreciar que el uso de celdas de *fan-in* superior a dos en la Sbox7 afecta negativamente debido a la sincronización temporal derivada en la etapa AND. Por otra parte, los valores de consumo de potencia se ven reducidos en todos los casos, siendo más notable al aplicar ambas técnicas a la vez en las Sbox_All. Como dato a destacar tenemos el SPD, donde podemos ver el compromiso entre seguridad y prestaciones, donde llegamos a tener una mejora de hasta un factor de x10.89 en el caso de la Sbox9_All.

Como la característica que mayor degradación obtiene es la frecuencia de operación, pasamos a evaluar dos métodos para técnicas de alta velocidad: la inserción de retrasos locales en la señal de reloj y el uso de un reloj trifásico con una sola celda por etapa del *pipeline*. En este caso, aunque se consiguen excelentes resultados en la frecuencia de operación y robustez de la celda, haciendo que no se estreche el ancho del pulso del dato de la señal de salida, se obtiene una degradación en la seguridad debido a la sincronización en los tiempos de ocurrencia de las transiciones de los bits de salida del criptocircuito. Si la aplicación requiere una alta frecuencia de operación, se tendrá que evaluar el nivel de seguridad alcanzado y ajustar ambos parámetros para obtener una buena relación seguridad-velocidad.

5. Diseño del demostrador criptográfico ASIC_CESAR

Una vez analizadas las vulnerabilidades de los criptocircuitos, y tras proponer varias contramedidas a diferentes niveles de abstracción, pasamos a la implementación física de estas contramedidas. Por tanto ahora nos encontramos en el nivel físico (ver Figura 5.1), donde realizaremos los diseños *full-custom* de nuestras celdas e implementaremos los casos de estudio para poder incluirlos en un ASIC y así comprobar experimentalmente la efectividad de nuestras propuestas.

El objetivo del diseño de este ASIC (llamado "CESAR" al igual que el proyecto donde está enmarcado), es la caracterización y verificación experimental de los niveles de seguridad y prestaciones alcanzados por cada contramedida. El alcanzar una alta frecuencia de operación no será objeto primordial de este diseño, centrándonos especialmente en las medidas de seguridad, área y consumo de potencia de los bloques criptográficos implementados, por lo tanto, no se incluirá el uso de técnicas de mejora de la frecuencia de operación del circuito ya que degradaba considerablemente la seguridad. El ASIC CESAR contiene un conjunto de bloques criptográficos usados como demostradores de las propuestas realizadas en la Tesis. Mencionar que, enmarcado dentro del proyecto CESAR, hay otra linea de investigación que utilizará parte el espacio del ASIC para diseñar sus bloques digitales ("Diseño 2"). Por ello, el espacio del ASIC se ha compartido, siendo aproximadamente el 60 % perteneciente al trabajo desarrollado en esta Tesis (ver Figura 5.2).

Para la integración se escogió la tecnología CMOS de 90nm de TSMC, en la que se han realizado todos los diseños y demostradores de esta Tesis. Se trata de una tecnología con un buen compromiso entre prestaciones y precio de integración, ampliamente utilizada con éxito en nuestro equipo de investigación en otros proyectos, como CriptoBio, MOBY-DIC o CITIES.

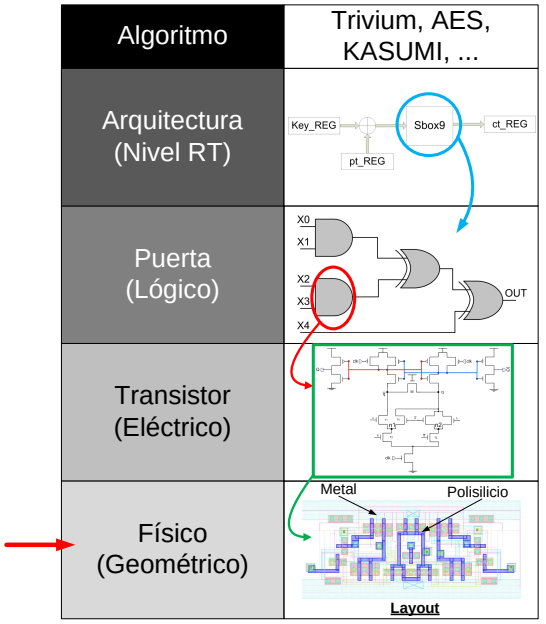

Figura 5.1.: Niveles de abstracción: nivel físico.

Pendiente de publicación.

Figura 5.2.: División del ASIC CESAR.

Este capítulo se va a dividir en los siguientes subapartados. En primer lugar, vamos a presentar los bloques criptográficos seleccionados para incluir en el ASIC y el flujo de diseño seguido. A continuación, se expone la metodología de diseño de las celdas de librería seguras, seguida por el diseño de los bloques criptográficos. Una vez expuestos los bloques criptográficos, presentamos el bloque digital que controlará el entorno de test. Finalmente mostraremos el diseño final del ASIC y setup de medida.

5.1. Especificaciones y flujo de diseño del prototipo ASIC CESAR

El ASIC CESAR ha incluido el diseño de diferentes bloques criptográficos con y sin contramedidas, con el fin de poder evaluar los niveles de seguridad, consumo y área alcanzados por cada implementación. Como bloque criptográfico demostrador se ha escogido la Sbox9, el bloque más vulnerable del algoritmo KASUMI [83]. El motivo para no implementar un algoritmo completo es el siguiente: el peor escenario para un diseñador es aquel en el que el atacante tenga las mejores condiciones de ataque posibles, es decir, que el atacante tenga accesible un punto para medir el consumo de potencia única y exclusivamente debido al bloque criptográfico que está atacando, así como acceso a los *plaintexts* o *ciphertexts*. Si conseguimos que las celdas sean seguras en este entorno, a la hora de plantear un ataque sobre un algoritmo completo, que esté implementado junto con otros bloques dentro de un mismo diseño, la dificultad o duración del ataque aumentará. Además, como queremos comparar los niveles de seguridad alcanzados con las contramedidas propuestas, necesitamos realizar ataques satisfactorios para determinar el número de patrones necesarios para recuperar la clave en cada caso, por lo que un entorno de ataque óptimo facilitará esta tarea sin perjudicar las medidas de comparación que queremos realizar.

Resumiendo las contramedidas estudiadas en los capítulos anteriores tenemos:

- A nivel de transistor:
  1. Optimización *dual-switch* y *single-switch* para la red DPDN.
  2. Uso de red DPUN tipo SABL.
  3. Uso de red DPUN tipo DDCVSL.
- A nivel lógico:
  1. Uso de celdas con *fan-in* superior a 2.
  2. Minimización lógica.

Las contramedidas aplicadas a ambos niveles añaden un nivel de seguridad adicional en cada uno de ellos. Por lo tanto, para poder evaluar de forma independiente los efectos causados por cada contramedida vamos a utilizar los siguientes casos de estudio:

1. **Sbox9 SABL**: Sbox9 implementada con celdas XOR/XNOR y AND/NAND tipo SABL de 2 entradas, sin minimización lógica.
2. **Sbox9 SABL_2P/P:** Sbox9 implementada con celdas XOR/XNOR tipo SABL de 2 entradas con solución *dual-switch*, celdas AND/NAND tipo SABL de 2 entradas con solución *single-switch*, y sin minimización lógica.
3. **Sbox9 DDCVSL**: Sbox9 implementada con celdas XOR/XNOR tipo DDCVSL de 2 entradas con solución *dual-switch*, celdas AND/NAND tipo DDCVSL de 2 entradas con solución *single-switch*, y sin minimización lógica.
4. **Sbox9 SABL_Shared**: Sbox9 implementada con celdas XOR/XNOR y AND/NAND tipo SABL de 2 entradas, y con minimización lógica.
5. **Sbox9 SABL_FanIn:** Sbox9 implementada con celdas XOR/XNOR tipo SABL de 2, 3 y 4 entradas, celdas AND/NAND tipo SABL de 2 entradas, y sin minimización lógica.
6. **Sbox9 SABL_All:** Sbox9 implementada con celdas XOR/XNOR tipo SABL de 2, 3 y 4 entradas, celdas AND/NAND tipo SABL de 2 entradas, y con minimización lógica.
7. **Sbox9 Std**: Sbox9 implementada con celdas XOR/XNOR y AND/NAND CMOS estándar, sin minimización lógica.

Para poder implementar las Sbox9 en el ASIC CESAR, lo ideal sería que cada una de las Sbox9 tuviera sus pines de entrada, salida y alimentación independientes. Debido a la limitación en el número de *pads* en el ASIC, vamos a tener que compartir ciertos recursos entre las Sbox9. Vamos a priorizar que cada una de las Sbox9 tenga sus *pads* de alimentación independientes, para aislar todo lo posible el consumo de potencia de cada Sbox9 individualmente. Por otra parte, el tener cada uno de los bits de entrada y salida de las Sbox9 conectados a su propio *pad* es imposible debido al gran número de bits, por lo tanto tenemos que incluir una unidad de control que sea la encargada de recibir los datos de entrada y controlar la activación de las Sbox9 de forma independiente.

Una vez seleccionados los casos de estudio, vamos a dividir los bloques de trabajo que se llevaron a cabo para el diseño del ASIC CESAR:

1. **Diseño de celdas de librería**: diseño *full-custom* de las celdas de librería con las que vamos a implementar las Sbox9.

2. **Diseño de las Sbox9**: Implementación de las Sbox9 con las celdas de librería.
3. **Diseño del bloque de control:** Diseño convencional del bloque digital que controla la operación independiente de los 7 demostradores.
4. **Diseño final**: Una vez obtenidos los layout de las Sbox9 y el bloque de control, el conexionado y el emplazamiento final con los pads se realizará manualmente, realizándose las verificaciones necesarias para poder generar el fichero para su envío a fabricación.
5. **Diseño de PCB y setup de medida**: para el testado del ASIC, se diseña una PCB y se adecúa el entorno de test para realizar las medidas necesarias.

La metodología de diseño de un ASIC digital convencional (ver Figura 5.3-a) comienza con una fase de *front-end*, donde, utilizando lenguajes de descripción hardware, se obtiene la descripción y especificación del circuito a implementar (descripción RTL, *Register Transfer Level*). A continuación, se realiza la síntesis lógica a nivel de puertas, donde se obtiene un fichero con la descripción de nuestro circuito a nivel de puertas. Utilizando este fichero, en la herramienta de *place&route* se realiza el emplazamiento y conexionado de todas las celdas hasta llegar a la implementación física o layout. Finalmente, se realiza el emplazamiento y conexionado con los *pads*.

En nuestro caso (ver el flujo de diseño seguido en la Figura 5.3-b), para la implementación de las Sbox9 vamos a tener que incluir en esta metodología una librería propia de celdas. Por lo tanto, el primer paso es la creación de una librería de puertas seguras, a la que llamaremos librería CESAR, conteniendo todas las vistas de las celdas necesarias para su uso, y se obtendrán todos los ficheros necesarios para poder utilizar la herramienta de *place&route* para el conexionado de las celdas. Tras la obtención de los ficheros de librería, pasamos al diseño de las Sbox9, que utilizan las celdas de la librería CESAR, por lo que no se podrá realizar una síntesis lógica automática de una descripción HDL ya que no tenemos una caracterización completa de las celdas de la librería. Otra alternativa semi-automática para la implementación de las Sbox9, consiste en la obtención de la descripción a nivel de puertas (archivo Verilog) a partir de los esquemáticos implementados en el entorno de Virtuoso, Cadence. Se simulan eléctricamente las celdas de la librería, pero el *place&route* se hará de forma automática con las herramientas del entorno de Cadence. Tal y como hemos realizado en los capítulos anteriores, las Sbox9 se implementaban directamente en Virtuoso instanciando las celdas necesarias e interconectando unas con

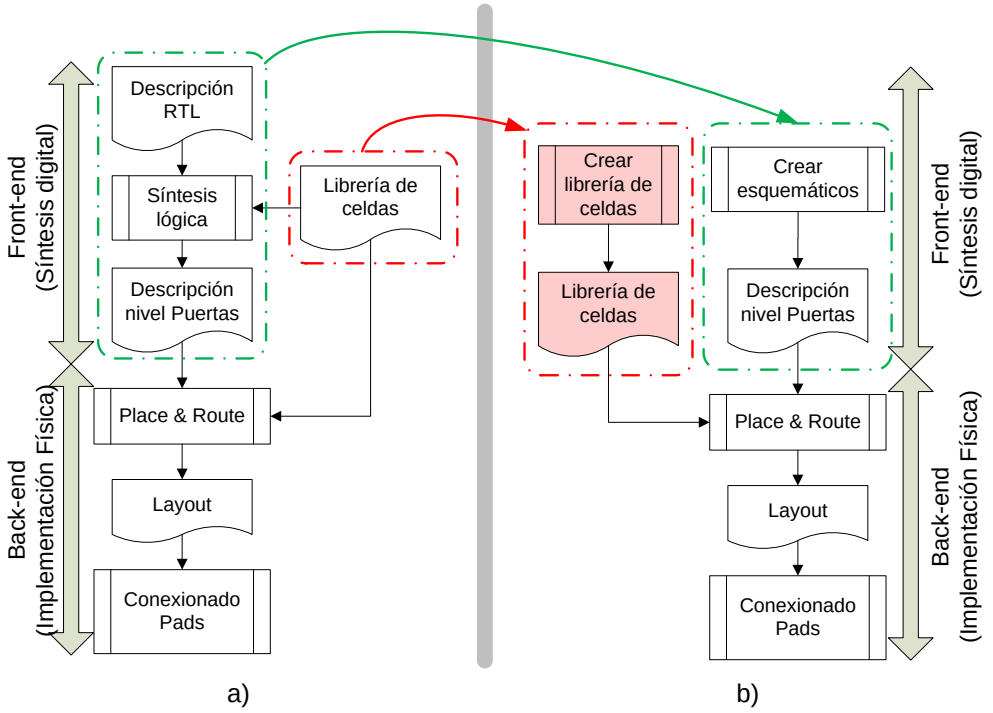

Figura 5.3.: Flujo de diseño digital: a) convencional y b) el seguido en esta Tesis.

otras. Siguiendo el mismo procedimiento, realizamos las nuevas implementaciones de todas las Sbox9 propuestas, partiendo así de una descripción a nivel de puertas. Por lo tanto, no es necesario realizar el paso de la síntesis lógica, y podemos pasar directamente al *place&route*. La parte final de *place&route* se hará siguiendo el flujo de diseño digital convencional. El bloque de control se implementará con las celdas de librería de TSMC 90nm siguiendo un flujo de diseño digital convencional. Tras la obtención de los layout de las Sbox9 y el bloque de control, el conexionado entre bloques y los *pads* del ASIC se realizará manualmente.

5.2. Diseño de las celdas de la librería CESAR

Para la implementación de las Sbox9, vamos a crear una librería que contenga las celdas seguras necesarias. Las celdas que se han implementado para el diseño de las Sbox9 propuestas son las siguientes:

1. Inversor/Buffer tipo SABL.
2. AND/NAND de 2 entradas tipo SABL.
3. AND/NAND de 2 entradas con solución single-switch tipo SABL.
4. AND/NAND de 2 entradas con solución single-switch tipo DDCVSL.
5. XOR/XNOR de 2 entradas tipo SABL.
6. XOR/XNOR de 2 entradas con solución dual-switch tipo SABL.
7. XOR/XNOR de 2 entradas con solución dual-switch tipo DDCVSL.
8. XOR/XNOR de 3 entradas tipo SABL.
9. XOR/XNOR de 4 entradas tipo SABL.

La creación de una nueva librería conlleva el seguimiento sistemático de una metodología, sobre todo si la queremos utilizar para un flujo de diseño digital completo para el diseño de cualquier ASIC. Para cada una de las celdas de librería se han de obtener unas vistas y archivos que incluyen no solo las correspondientes al diseño físico (abstract, layout, LEF (*Library Exchange Format*)), sino también las vistas a nivel de esquemático, e información temporal para la verificación funcional.

La caracterización de la librería supone un arduo trabajo, sobre todo en el conocimiento y uso de la herramienta de caracterización donde un primer paso importante es la identificación de la estructura de la celda para poder determinar su funcionalidad y caracterización temporal. La estructura diferencial de nuestras celdas y la utilización de la señal de reloj, hace muy difícil que la herramienta de caracterización pueda determinar y caracterizar de forma automática nuestra librería. Por ese motivo, la caracterización de la librería se dejará como trabajo futuro no siendo objeto de esta Tesis y plantearemos una estrategia diferente para la implementación de nuestras Sbox9.

Para poder realizar de forma automática el *place&route*, se necesitará la descripción de alto nivel de las celdas de librería y los ficheros de configuración para la herramienta de *place&route.*

5.2.1. Metodología de diseño de la librería

La metodología de diseño seguida para la creación de la librería CESAR es la mostrada en la Figura 5.4.

Como ejemplo, vamos a mostrar el proceso de diseño de la puerta XOR/XNOR2 tipo SABL. El punto de partida es el esquemático de la celda a nivel de transistor, ver Figura 5.5. El proceso de dimensionado de los transistores se ha discutido en el Capítulo 2 y se incluye en el Apéndice B. Los esquemáticos y dimensiones de los transistores de todas las celdas diseñadas se muestran en el Apéndice D. Una vez verificada su correcta funcionalidad tras varias simulaciones eléctricas pasamos a crear el layout. Para la realización del layout hay que cumplir con dos requisitos fijados por la herramienta automática de *place&route*: que las celdas tengan una altura en concreto y que la anchura sea múltiplo de un valor fijo[1]. Esto se debe a que en caso de tener que realizar un diseño donde se deban utilizar tanto las celdas de la librería estándar como las nuestras, estas tienen que ser compatibles para que la herramienta de *place&route* pueda emplazarlas y conectar correctamente los raíles de alimentación entre ambas. El layout para el caso de ejemplo se muestra en la Figura 5.6.

Para la realización del layout se han utilizado 3 niveles de metal (el metal-1 se corresponde con el color celeste, metal-2 amarillo y metal-3 verde oscuro) para todas las celdas diseñadas. El acceso a los pines de entrada y salida se realiza a través del metal-3, dejando los raíles de alimentación a metal-1. Para la creación de los layout, se ha mantenido en la parte central el diseño correspondiente a la estructura común entre las celdas, es decir, en el caso del ejemplo los transistores T1-T11 y transistor M. Los transistores que implementan la funcionalidad de la celda (en el ejemplo T12-T17) se emplazan en los laterales del layout manteniendo la mayor simetría posible y utilizando los caminos idénticos de metal en cada una de las ramas diferenciales.

[1]No se indica en esta Tesis el valor del múltiplo de la anchura, ni la altura de las celdas, por motivos de confidencialidad.

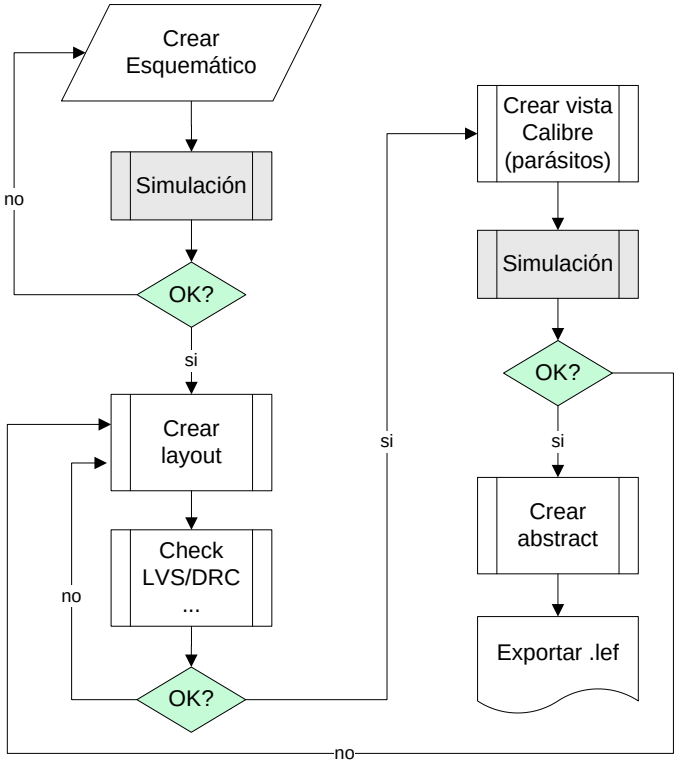

Figura 5.4.: Flujo de diseño de librería CESAR.

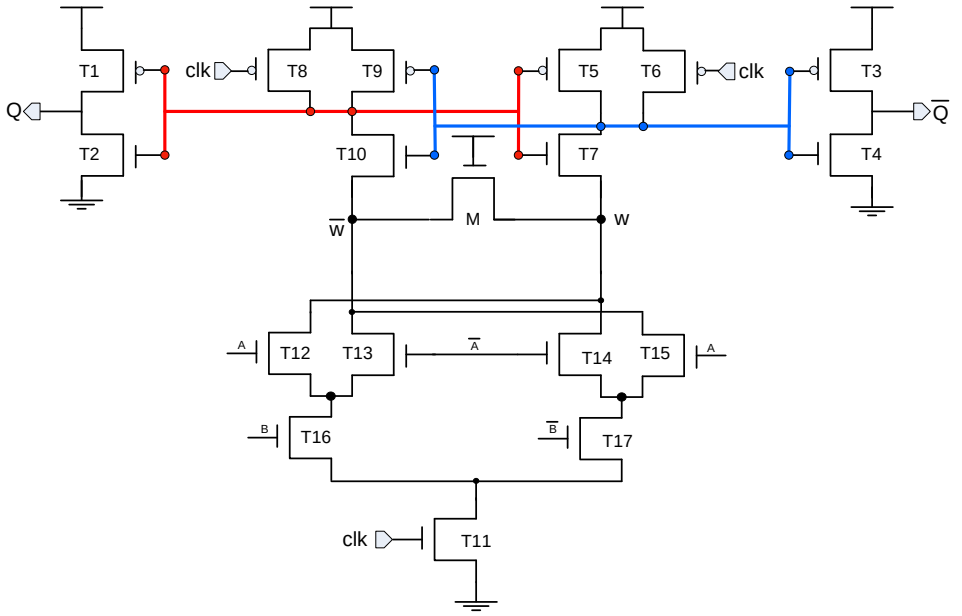

Figura 5.5.: Esquemático de la celda XOR/XNOR2 tipo SABL.

Pendiente de publicación.

Figura 5.6.: Layout de la celda XOR/XNOR2 tipo SABL.

Para conseguir un diseño lo más compacto posible de las ramas diferenciales y poder realizar conexiones directas entre drenador y fuente de los transistores, se ha hecho uso de los grafos de Euler (en la teoría de grafos, un camino euleriano es un camino que pasa por cada arista una y solo una vez). En nuestro caso específico, cada uno de los transistores de la red DPDN serán una arista del grafo, mientras que los vértices serán las conexiones entre los nodos de los transistores. Como ejemplo se muestra la Figura 5.7. Se puede ver que si realizamos el camino ${}_4\bar{A}_1A_2B_3\bar{B}_4A_5\bar{A}_2$ pasamos por cada uno de los caminos, una y solo una vez. Podremos compartir drenador/fuente entre los transistores que compartan el mismo nodo, por ejemplo, los transistores A y $\bar{A}$ comparten drenador/fuente en la Figura 5.8.

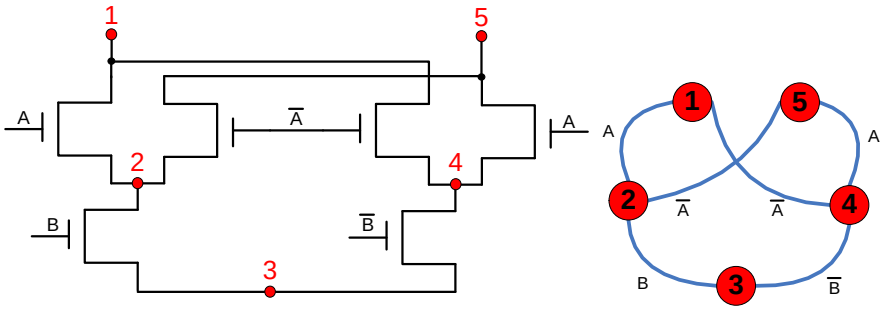

Figura 5.7.: Grafo de Euler correspondiente al DPDN de la celda XOR/XNOR2.

Pendiente de publicación.

Figura 5.8.: Ejemplo de emplazamiento de los transistores A y $\bar{A}$ en XOR/XNOR2 tipo SABL.

Siguiendo esta metodología de diseño de los layout, conseguimos reducir el área de las puertas, sobre todo en los casos de 3 y 4 entradas de las celdas XOR/XNOR. La información de los layout generados se muestran en la Tabla 5.1.

Se puede apreciar que los layout con menor área se corresponden con las puertas DDCVSL, reduciéndose el área un 31.03 % en el caso AND/NAND y un 24.14 % en

Tabla 5.1.: Información de los layout de la librería CESAR.

Puerta	Entradas	Salidas	Transistores	Ancho(μm)
Inv/Buff SABL	3	2	14	7.00
AND/NAND2 SABL	5	2	18	8.12
AND/NAND2_P SABL	5	2	19	8.12
AND/NAND2 DDCVSL	5	2	16	5.60
XOR/XNOR2 SABL	5	2	18	8.12
XOR/XNOR2_2P SABL	5	2	20	8.12
XOR/XNOR2 DDCVSL	5	2	16	6.16
XOR/XNOR3 SABL	7	2	26	9.24
XOR/XNOR4 SABL	9	2	42	15.04

el caso XOR/XNOR respecto a las SABL clásicas. Por otra parte, podemos ver que las puertas SABL clásicas tienen la misma anchura que las SABL con las soluciones *dual/single-switch*. Esto se debe a que al incluir los transistores de la red DPDN en el layout, en el caso de la SABL clásica necesitamos tener esa anchura para poder mantener la simetría de la red DPDN, quedando un hueco en la parte superior a ambos lados del layout. La parte superior del layout está reservada para los transistores PMOS, mientras que en la parte inferior están emplazados los transistores NMOS, por lo que se aprovechan los huecos superiores a los lados de las implementaciones SABL clásicas para incluir los transistores P en el caso de las soluciones *dual/single-switch*. Finalmente, podemos observar que añadir una entrada a la celda XOR/XNOR aumenta su área un 13.78 %, siendo en el caso de la XOR/XNOR de 4 entradas un 89.66 %.

Tras crear el layout, se realizan diferentes chequeos como el LVS (*Layout vs Schematic*), donde se comprueba que el layout se corresponde con el esquemático diseñado, así como pasar el DRC (*Design Rule Check*) para comprobar que se cumplen las normas de diseño impuestas por el fabricante. Una vez superados estos chequeos, el siguiente paso es generar una vista del diseño con la extracción de los parásitos (vista

formato Calibre). De esta forma, podremos realizar una simulación post-layout verificando su funcionalidad con los parásitos obtenidos tras la implementación física. Cuando las simulaciones satisfagan los requisitos necesarios para nuestra implementación, podemos pasar a extraer el fichero LEF, donde figura la información física de nuestra celda (vista *abstract* del extraído del layout, ver Figura 5.9). Los datos que se indican en el fichero son por ejemplo los niveles de metal utilizados y su posicionamiento, pines de entrada y salida, etc.

Pendiente de publicación.

Figura 5.9.: Abstract de la celda XOR/XNOR2 tipo SABL.

Estos pasos se realizarán para cada una de las celdas diseñadas (ver Apéndice D), obteniendo finalmente un único archivo LEF que utilizaremos para la herramienta de *place&route*.

5.3. Diseño de las Sbox9

Para realizar el diseño de las Sbox9, vamos a partir del esquemático de cada implementación, del cual extraeremos la descripción a nivel de puertas (netlist Verilog) dentro de la herramienta Virtuoso de Cadence. Los pasos a seguir en esta parte del diseño, que se corresponderían con el *back-end* de un diseño digital convencional, son los mostrados en la Figura 5.10.

En primer lugar, dentro de la herramienta de *place&route* automático, Encounter de Cadence, cargamos el netlist y el fichero LEF con la información sobre la librería de celdas CESAR. A continuación, creamos el anillo con las alimentaciones para el bloque Sbox9. Después añadimos las celdas de librería y hacemos el routing automático del diseño. Tras realizar las verificaciones del diseño y estando todo correcto, generamos el archivo GDSII para importar la información del layout obtenido a la siguiente herramienta.

Para continuar con las verificaciones, dentro de la herramienta Virtuoso de Cadence, importamos el layout de la Sbox9. Tras comprobar las reglas DRC y confirmar que el LVS es correcto, al igual que hacíamos con las celdas individuales, se extraen

los parásitos del layout generando a su vez una vista Calibre para su simulación. Con la vista calibre, se realizan simulaciones eléctricas donde comprobamos que la funcionalidad de la celda es la esperada.

Tras realizar este procedimiento con todas las Sbox9 propuestas, podemos dar por concluida la etapa de diseño de los bloques criptográficos.

La información del número de celdas utilizadas en cada caso, así como las dimensiones finales del layout de cada Sbox9 implementada se muestran en la Tabla 5.2. Los layout finales de cada implementación se muestran en la Figura 5.11, donde se ha mantenido la relación de tamaños entre las diferentes Sbox9 para apreciar el área ocupada por cada implementación. Para más detalles del diseño y simulación de las Sbox9, consultar el Apéndice E.

Se puede apreciar cómo las Sbox9 SABL y la SABL_2P/P son las que más área ocupan. Si le asignamos el 100 % de área a las Sbox9_SABL y SABL_2P/P, tenemos que le siguen la SABL_FanIn y SABL_DDCVSL con un 84.29 % y un 80.38 % respectivamente. Es interesante el dato obtenido por la propuesta DDCVSL, ya que

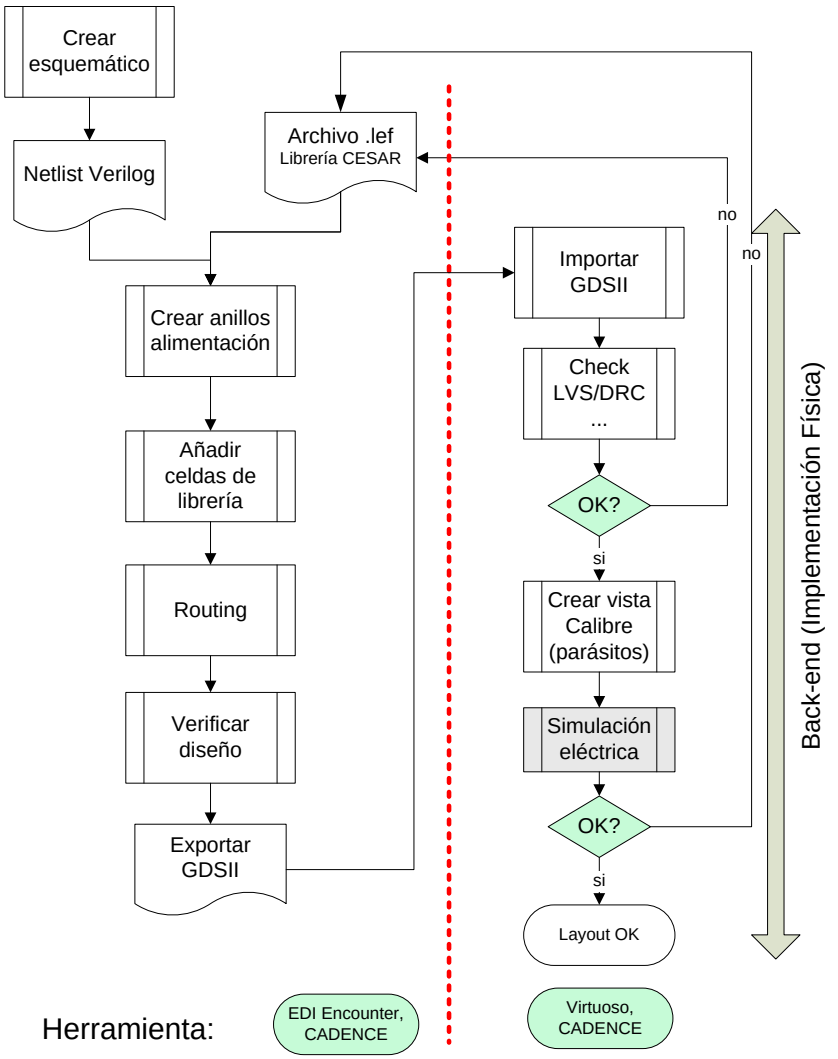

Figura 5.10.: Flujo de diseño del demostrador Sbox9.

comparada con la SABL_2P/P se consigue reducir un 19,62 % el área con el uso del DPUN DDCVSL. Si seguimos analizando los datos, tenemos que la SABL_Shared y SABL_All se reducen a un 72.37 % y a un 62.27 % respectivamente. Finalmente, como esperábamos, la implementación CMOS con celdas estándar es aquella que menos área ocupa.

Tabla 5.2.: Numero de celdas y dimensiones de las Sbox9.

	SABL					DDCVSL	STD
	Classic	**2P/P**	**Shared**	**FanIn**	**All**		
INV. STD	-	-	-	-	-	-	5
INV. SABL	-	-	2	-	-	-	-
AND2 STD	-	-	-	-	-	-	86
AND2 SABL	86	-	36	86	36	-	-
AND2 SABL_P	-	86	-	-	-	-	-
AND2 DDCVSL	-	-	-	-	-	86	-
XOR2 STD	-	-	-	-	-	-	92
XOR2 SABL	92	-	73	-	20	-	-
XOR2 SABL_2P	-	92	-	-	-	-	-
XOR2 DDCVSL	-	-	-	-	-	92	-
XOR3 SABL	-	-	-	40	16	-	-
XOR4 SABL	-	-	-	4	7	-	-
Ancho (μm)	106.60	106.60	88.51	97.35	83.54	95.36	69.13
Alto (μm)	98.16	98.16	85.56	90.60	78.00	88.20	65.40
Area (μm^2)	10463.86	10463.86	7572.92	8819.91	6516.12	8410.75	4521.10
Area(%)	100 %	100 %	72.37 %	84.29 %	62.27 %	80.38 %	43.21 %

Pendiente de publicación.
(a) Sbox9 SABL.

Pendiente de publicación.
(b) Sbox9 SABL_2P/P.

Pendiente de publicación.
(c) Sbox9 SABL_Shared.

Pendiente de publicación.
(d) Sbox9 SABL_FanIn.

Pendiente de publicación.
(e) Sbox9 SABL_All.

Pendiente de publicación.
(f) Sbox9 DDCVSL.

Pendiente de publicación.
(g) Sbox9 STD.

Figura 5.11.: Layout de las Sbox9.

5.4. Diseño del bloque de control

Para poder controlar y generar los estímulos para las siete Sbox9, vamos a necesitar un bloque de control (topASIC). Este módulo será el encargado de recibir las entradas del exterior del ASIC, generar las entradas y señales de control para las Sbox9, recibir las salidas de las Sbox9 y después mostrar la salida al exterior. El módulo de más alta jerarquía que representa al ASIC CESAR es el mostrado en la Figura 5.12. Mencionar que, aunque la Sbox está incluida en el esquema de la Figura 5.12, esta se encuentra fuera del bloque digital ya que es el diseño *full-custom* realizado con las celdas de la librería CESAR y queremos medir su consumo de forma aislada, por lo que no lo podemos incluir dentro del mismo flujo de diseño. Por lo tanto, antes y después de la Sbox tendremos pines de entrada y salida XSbox e YSbox que irán/vendrán de los bloques *full-custom*.

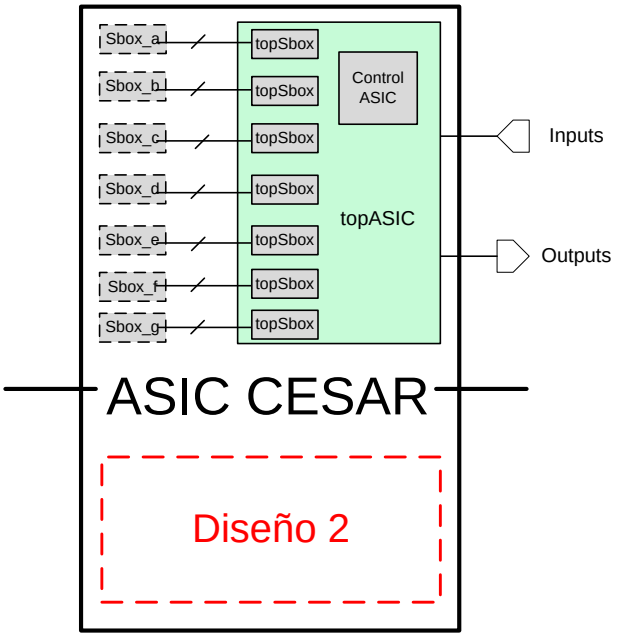

Figura 5.12.: Esquema del ASIC CESAR.

El bloque de control digital consta de 3 bloques:

1. **topSbox:** Unidades que proporcionan la entrada X y leen la salida Y de cada una de las Sbox9. Cada Sbox9 tiene asignado un bloque "topSbox" independiente.
2. **controlASIC**: es la máquina de estados que se encarga de leer los datos de entrada del exterior y procesarlos, así como generar todas las señales de control para habilitar o deshabilitar los diferentes componentes de los bloques "topSbox".
3. **topASIC**: dentro de este bloque se incluye la lógica necesaria para el conexionado entre "topSbox" y "controlASIC", así como de seleccionar la habilitación de cada una de las Sbox9 y mostrar a la salida el dato deseado. Además, es el encargado de habilitar el modo test implementado en el ASIC en caso de tener que detectar algún error en la implementación.

5.4.1. Bloque "topSbox"

Para realizar la descripción de todos los bloques que componen la unidad de control, vamos a comenzar por el bloque superior que controla cada una de las Sbox9, al que llamaremos "topSbox". Este bloque está basado en [92], donde muestran el esquema a seguir para el control de unidades Sbox para el testado de implementaciones con contramedidas. El esquema del bloque "topSbox" se muestra en la Figura 5.13 y la descripción de las entradas y salidas del bloque se muestran en la Tabla 5.3.

Como se muestra en la Figura 5.13, se añaden sendos registros antes y después del bloque Sbox. Esto lo hacemos para aislar la parte combinacional y evitar la propagación de *glitches*. Cada registro de 9 bits puede habilitarse por separado, mientras que a todos los registros ataca la misma señal de reloj *clk*, excepto para el registro de salida *Reg_SboxOUT* al que ataca *clk2*, sincronizando todo el procesado de las señales. El reloj de muestreo de salida tiene que ser tal y como se muestra en la Figura 5.14, muestreando con el flanco de subida de *clk2*, para que se pueda muestrear el dato válido de las salidas de las Sbox9 que empleen lógica DPL. Como tanto *clk* como *clk2* se van a generar externamente, tendremos control completo para poder ajustar los flancos de subida de las dos señales para un muestreo correcto.

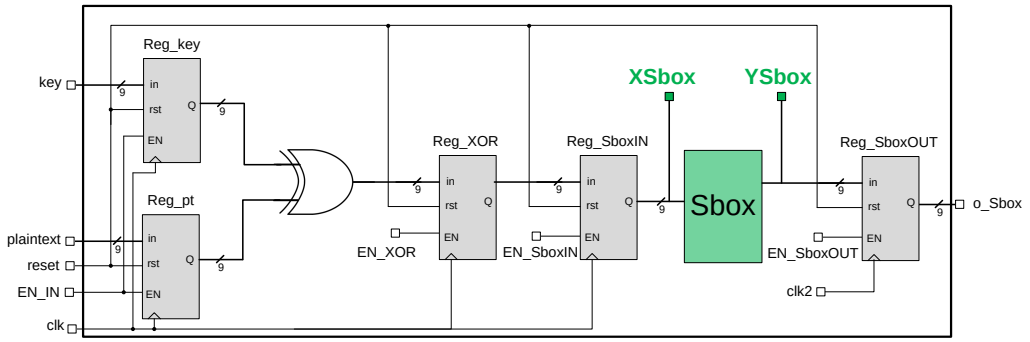

Figura 5.13.: Esquema del circuito de test "topSbox".

Tabla 5.3.: Entradas y salidas del bloque *topSbox* del ASIC CESAR.

Nombre	Dirección	Descripción y comentarios
EN_IN	Entrada	Enable de los registros Reg_key y Reg_pt
EN_XOR	Entrada	Enable del registro Reg_XOR
EN_SboxIN	Entrada	Enable del registro Reg_SboxIN
EN_SboxOUT	Entrada	Enable del registro Reg_SboxOUT
clk	Entrada	Señal de reloj
clk2	Entrada	Señal de reloj de muestreo
reset	Entrada	Señal de reset
plaintext<8:0>	Entrada	9 bits del *plaintext*
key<8:0>	Entrada	9 bits de clave
YSbox<8:0>	Entrada	Dato de salida de la Sbox9
XSbox<8:0>	Salida	Dato de entrada a las Sbox9
o_Sbox<8:0>	Salida	Salida muestreada de la Sbox9

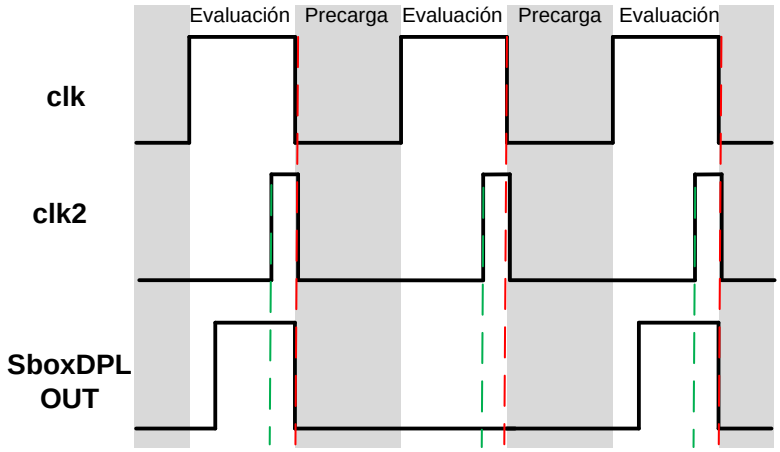

Figura 5.14.: Esquema de relojes empleado.

Por otra parte, la clave *key* y el *plaintext pt,* que los proporciona el bloque de control, se almacenan en los registros de 9 bits *Reg_key* y *Reg_pt* respectivamente. Al igual que en el caso de la Sbox, el banco de puertas XOR está precedido por los registros de *Reg_key* y *Reg_pt*, mientras que a la salida tenemos el registro *Reg_XOR* que almacena la salida de la operación XOR entre la *key* y el *pt*. De esta forma, nuevamente aislamos la parte combinacional. Si continuamos analizando el esquemático, nos encontramos que a la entrada del bloque Sbox tenemos el registro *Reg_SboxIN*. Poniendo los registros *Reg_XOR* y *Reg_SboxIN* en serie se consigue poder diferenciar entre el consumo debido a la parte combinacional de la Sbox y de la parte debida a los registros, como por ejemplo *Reg_XOR*. Para finalizar, se puede observar que la salida del bloque Sbox se almacena en el registro de salida *Reg_SboxOUT*.

5.4.2. Bloque "controlASIC"

Este bloque es el encargado de leer/enviar los datos de entrada/salida del ASIC y generar las señales de control para las Sbox9, así como los bit para el control del

test, tal y como muestra la Figura 5.15. Básicamente, este bloque es una máquina de estados en el que vamos generando las señales de control para activar las diferentes Sbox9 del ASIC.

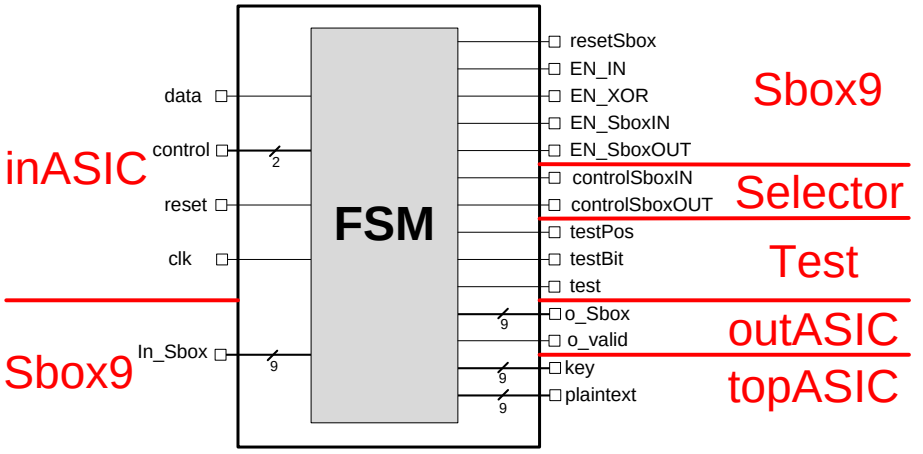

Figura 5.15.: Esquema del bloque "controlASIC".

Debido al limitado número de *pads* de los que disponemos, la carga de los datos de la clave, *plaintext*, señales de habilitación de las Sbox9 y las señales de control del modo test tienen que enviarse en modo serie. Por este motivo, nos encontramos con las dos señales de entrada *data* y *control*. En función de la señal de *control* de 2 bits, vamos a proceder a leer los datos de la entrada *data*, y a clasificarlos para obtener las señales que atacan a los otros bloques. En función del valor de la señal control, realizaremos las operaciones mostradas en la Tabla 5.4.

Tabla 5.4.: Selección de las fases de operación.

control<1:0>	**Descripción y comentarios**
01	Fase de carga
10	Fase de encriptado
otro caso	Espera

Por otra parte, vamos a cargar 40 bits de datos de entrada en un registro al que llamaremos *dataIn* donde la correspondencia con las demás variables se muestra en la Tabla 5.5 y en la Figura 5.16.

Tabla 5.5.: División de los datos de entrada serie.

Señal	Correspondencia
key<8:0>	dataIn[39:31]
plaintext<8:0>	dataIn[30:22]
resetSbox	dataIn[21]
controlSboxOUT<6:0>	dataIn[20:14]
controlSboxIN<6:0>	dataIn[13:7]
test	dataIn[6]
testPos<1:0>	dataIn[5:4]
testBit<3:0>	dataIn[3:0]

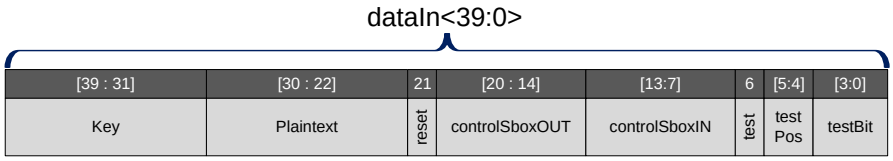

Figura 5.16.: División de los datos de entrada serie.

El diagrama de estados se muestra en la Figura 5.17, donde se pueden ver los 9 estados que la componen. Comenzamos desde un estado de espera "IDLE". Para iniciar la carga serie, tenemos que introducir los datos de entrada mediante la señal *data* fijando a su vez la señal de control en modo carga *control="01"*. Una vez finalizada la carga, pasamos a un estado "READY" donde el bloque está preparado para iniciar la fase de encriptado.

Para iniciar la encriptación, fijaremos la señal *control="10"*. Durante el encriptado, pasaremos por 4 estados independientes donde activaremos los diferentes registros de los bloques "topSbox". Primero se realiza la carga de *key* y *pt* activando la señal *EN_IN*, seguida de la activación de *EN_XOR* en el estado "ENCRYPT_XOR". En este punto, tenemos almacenado en el registro *Reg_XOR* el valor de la operación XOR entre el *pt* y *key*. Finalmente, activamos *EN_SboxIN* para que las entradas pasen a la Sbox9 mientras que en el siguiente estado se activa la señal *EN_SboxOUT* que realiza el muestreo de la salida de las Sbox9.

Una vez finalizada la fase de encriptado, se leen los datos de salida de la Sbox9 y se envían al exterior activando la señal *o_valid* que nos mostrará que los datos de salida en *o_Sbox* son datos válidos.

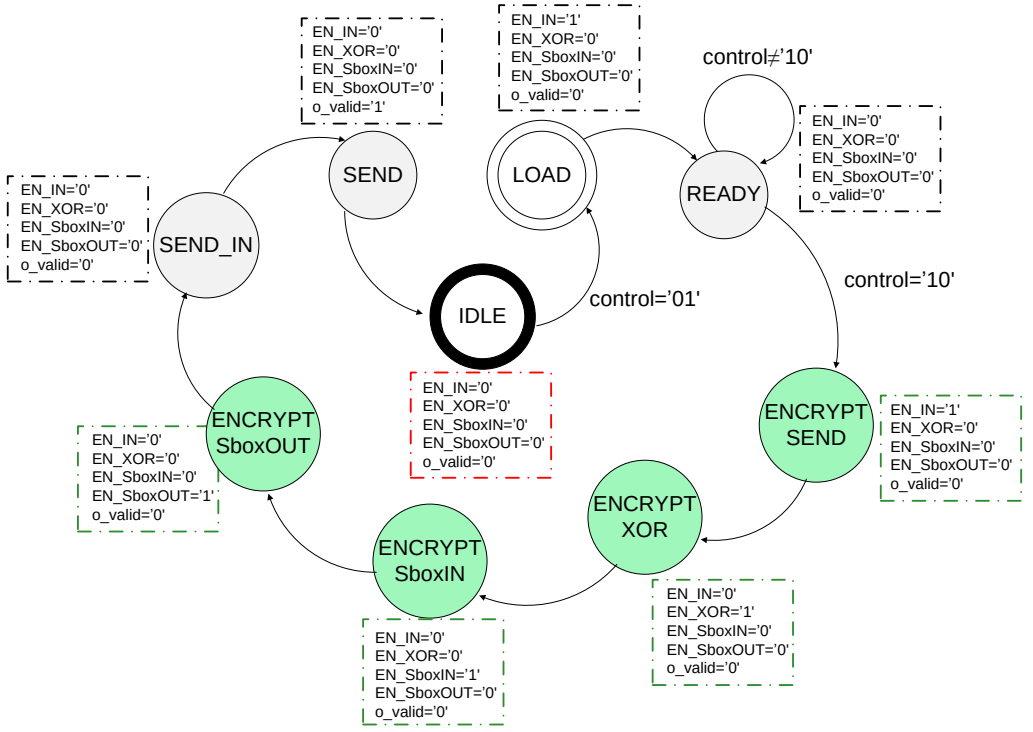

Figura 5.17.: Diagrama de estados del bloque "controlASIC".

En la Tabla 5.6 se muestra la descripción de las entradas y salidas del bloque.

Tabla 5.6.: Entradas y salidas del bloque *controlASIC* del ASIC CESAR.

Nombre	Dirección	Descripción y comentarios
data	Entrada	Entrada serie de datos
control<1:0>	Entrada	Señal de control
clk	Entrada	Señal de reloj
reset	Entrada	Señal de reset
in_Sbox<8:0>	Entrada	Dato de salida de la Sbox9
resetSbox	Salida	Señal de reset para las Sbox
EN_IN	Salida	Enable de los registros Reg_key y Reg_pt
EN_XOR	Salida	Enable del registro Reg_XOR
EN_SboxIN	Salida	Enable del registro Reg_SboxIN
EN_SboxOUT	Salida	Enable del registro Reg_SboxOUT
controlSboxIN<6:0>	Salida	Señales de activación de las entradas a las Sbox9
controlSboxOUT<6:0>	Salida	Señales de activación de las salidas a las Sbox9
testPos<1:0>	Salida	Señales de control de la posición a testar
testBit<3:0>	Salida	Señales de control del bit a testar
test	Salida	Señales de control de activación del modo test
o_Sbox<8:0>	Salida	Salida de la Sbox9
o_valid<8:0>	Salida	Señal de activación del dato válido
key<8:0>	Salida	9 bits de clave
plaintext<8:0>	Salida	9 bits de plaintext

5.4.3. Bloque "topASIC"

El bloque superior es "topASIC", en el cual se instancian los bloques "topSbox" y "controlASIC", y es el encargado de interconectar cada uno de los bloques, así como gestionar la funcionalidad de test y habilitar cada una de las Sbox9 en función de las señales de control. La Figura 5.18 muestra el esquemático del bloque "topASIC", donde las Sbox9 se muestran en verde no siendo estas parte del bloque.

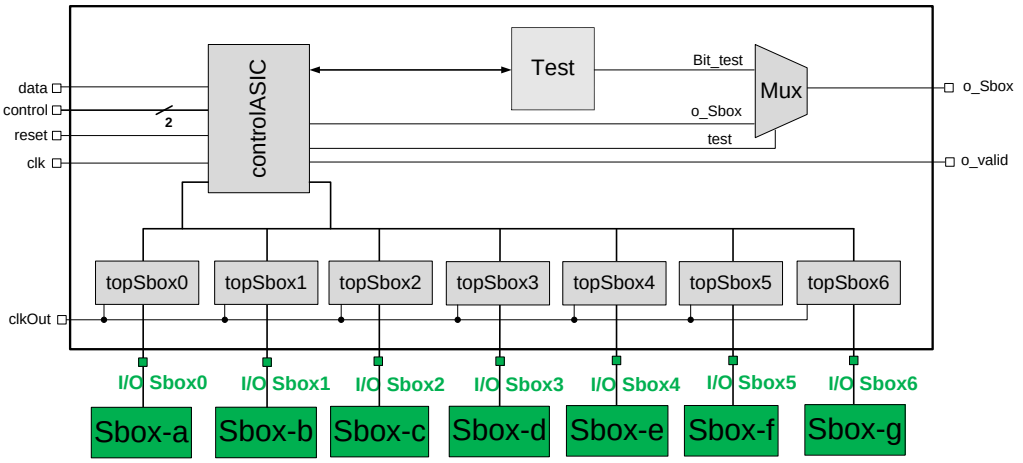

Figura 5.18.: Esquemático del bloque "topASIC".

El bloque contiene entradas y salidas que se conectarán finalmente a los *pads* de entrada/salida del ASIC y a las Sbox9 *full-custom* (ver Tabla 5.7). Aclarar que las señales de entrada a las Sbox9 son los 9 bits de entrada (*XSbox*) y sus complementadas (*XSboxc*), además de la señal de *clk*. Por otra parte, tenemos que los datos que recibe de la Sbox9 son las salidas *YSbox* y sus complementadas *YSboxc*. Por lo tanto, tenemos 19 bits de entrada y 18 bits de salida para cada Sbox9. En la Tabla 5.7 se muestran las entradas y salidas a las Sbox9 como *I/O_Sbox*.

Tabla 5.7.: Entradas y salidas del bloque *topASIC* del ASIC CESAR.

Nombre	Dirección	Descripción y comentarios
data	Entrada	Entrada serie de datos
control<1:0>	Entrada	Señal de control
clk	Entrada	Señal de reloj
clkOUT	Entrada	Señal de reloj de muestreo
reset	Entrada	Señal de reset
o_Sbox<8:0>	Salida	Salida de la Sbox9
o_valid<8:0>	Salida	Señal de activación del dato válido
I/O_Sbox	Entrada/Salida	Entradas y salidas de la Sbox9

Este bloque a su vez controla qué Sbox9 se va a habilitar en función de las señales de control procesadas desde el bloque "controlASIC". Para los ataques DPA, vamos a activar una sola Sbox9 en cada encriptación, aislando de esta forma el consumo de potencia a medir. Por lo tanto, tendremos que habilitar únicamente las entradas de la Sbox objetivo. Sin embargo, para tener más versatilidad en el circuito, hemos planteado la posibilidad de poder activar tantas Sbox9 como sean necesarias para el test, de esta forma podremos medir el consumo de una, dos o incluso todas las Sbox en funcionamiento. Esto se controla a través de la señal *controlSboxIN* de 7 bits. Cada uno de los bits de *controlSboxIN* (Tabla 5.5 y 5.6) se corresponde con una Sbox9, el bit menos significativo o bit 0 habilitaría la Sbox_a, el bit 1 habilitaría la Sbox_b y así sucesivamente. Por ejemplo, con *controlSboxIN* igual a "0000100" estaríamos habilitando la Sbox_c, mientras que con *controlSboxIN* igual a "0100011" estaríamos habilitando la Sbox_a, Sbox_b y Sbox_f.

De la misma forma, tenemos la señal *controlSboxOUT* para fijar qué datos se mostrarán a la salida. A la salida únicamente vamos a poder observar el resultado encriptado de una Sbox9. Por lo que *controlSboxOUT* tendrá sólo un bit activado. Al igual que en el caso de *controlSboxIN*, el bit 0 se corresponde con la Sbox_a, el bit 1 con la Sbox_b y así sucesivamente.

Por otra parte tenemos el modo test, donde la señal de salida *test* del bloque "controlASIC" determina si estamos en modo de funcionamiento normal (*test* igual a 0) o en modo test (*test* igual a 1). Si *test* está fijada a '0', a la salida veremos los datos de la salida de la Sbox9 testada. En caso de estar en modo test, el bloque "topASIC" determina qué valor fijar a la salida. Para ello, tenemos dos señales de control del test que tenemos que valorar: *testPos* y *testBit*. La primera de ellas determina qué posición queremos ver a la salida, mientras que la segunda determina el bit que queremos observar.

Para el testado vamos a tener 4 posiciones diferentes: valor del *plaintext*, valor de la clave, dato de entrada a la Sbox9 (XSbox) y dato de salida de la Sbox9 (YSbox). De esta forma, en caso de recibir datos erróneos tras el encriptado, vamos a poder chequear si la clave y *plaintext* son los fijados desde el exterior y se reciben correctamente. En caso de que esta comprobación sea correcta, podemos determinar si el error se encuentra antes de la Sbox9 comprobando que el valor de la operación XOR entre la *key* y *plaintext* es el esperado. Finalmente, podremos comprobar que si la entrada a la Sbox9 (XSbox) es la correcta y la salida (YSbox) es erronea, se trata de un error en la Sbox9 o su conexionado, mientras que si nuevamente este dato es el esperado, se trataría de un error de muestreo o conexionado final. Esta posición se determina con el valor de la señal *testPos* de 2 bit (ver Figura 5.19), siendo "00" para testar el *plaintext*, "01" para la clave, "10" para la entrada a la Sbox9 y "11" para la salida de la Sox9.

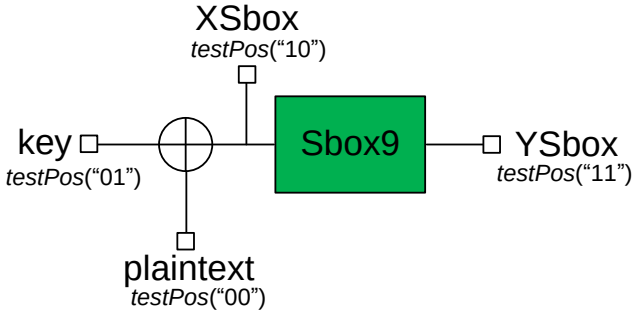

Figura 5.19.: Posición de test en función de la señal *testPos*.

Tabla 5.8.: Bit de salida en modo test en función de *testBit*.

testBit	**Bit testado**
"0000"	Bit 0
"0001"	Bit 1
"0010"	Bit 2
"0011"	Bit 3
"0100"	Bit 4
"0101"	Bit 5
"0110"	Bit 6
"0111"	Bit 7
"1000"	Bit 8

Una vez fijada la posición a testar, seleccionamos mediante la señal de *testBit* el bit que queremos observar a la salida. En este caso, esta señal consta de 4 bits, y el bit que se mostrará en cada caso a la salida es el mostrado en la Tabla 5.8.

Con los bloques presentados hasta ahora, tenemos control completo sobre las Sbox9 a testar, incluyendo además un modo test para poder detectar posibles errores en el diseño o en la fabricación, así como versatilidad para la medición del aporte en el consumo de potencia de las diferentes Sbox9, pudiendo activar tantas como sea necesario.

5.4.4. Flujo de diseño del circuito de control

Para realizar con éxito la implementación del circuito de control "topASIC", es necesario seguir rigurosamente la metodología de diseño de nuestro bloque digital, que es la mostrada en la Figura 5.20. En este caso, el circuito se ha implementado siguiendo una metodología de diseño *semi-custom* con herramientas de diseño automático comerciales.

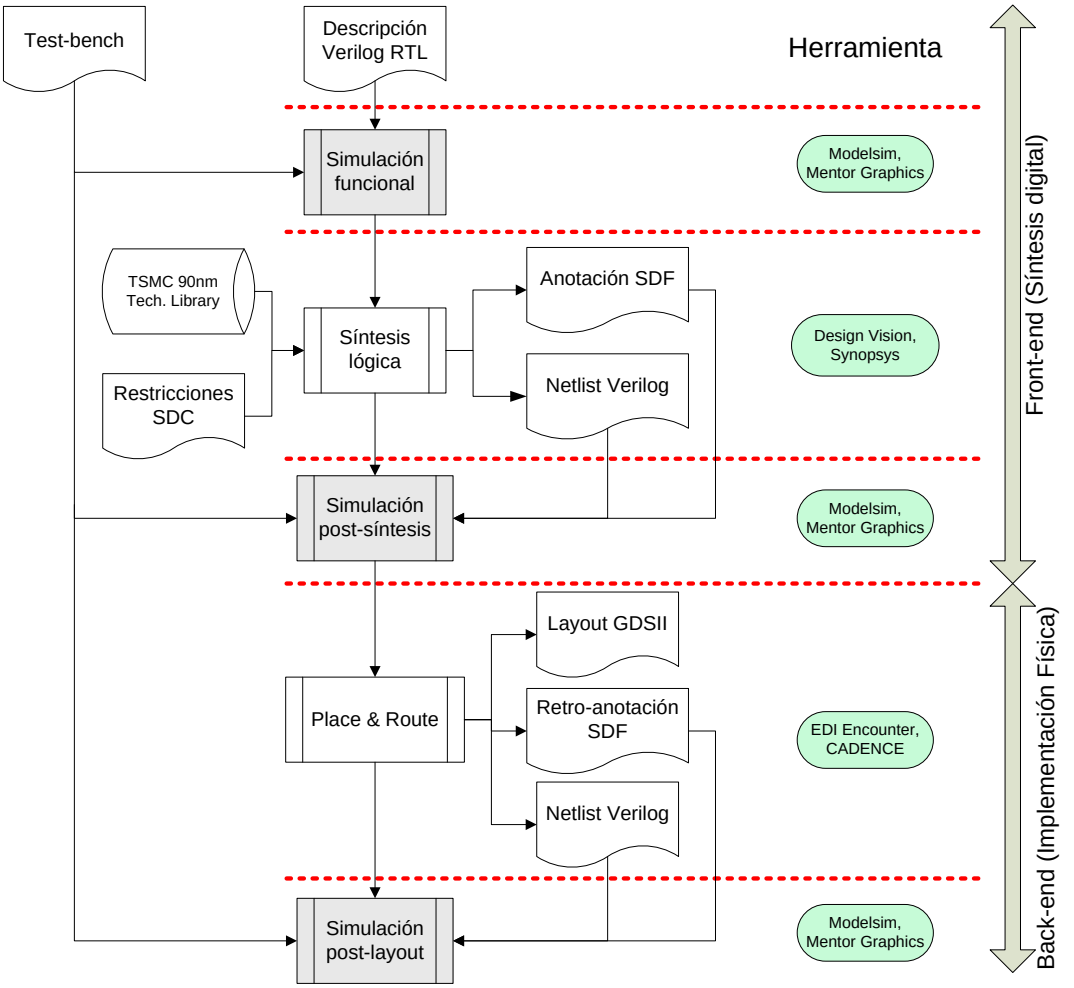

Figura 5.20.: Flujo de diseño digital.

El diseño completo se puede dividir en dos fases: *front-end* (diseño lógico) y *back-end* (implementación física). Dentro de la parte de *front-end* se encuentra el diseño lógico a nivel RTL, realizando la descripción hardware del circuito en lenguaje Verilog. Una vez descrito el bloque, se comprueba su correcto funcionamiento a través de una simulación funcional en Modelsim, de Mentor Graphics. A continuación, tras seleccionar la tecnología de TSMC90nm, se realiza la síntesis lógica en la herramienta Design Vision de Synopsys. Tras anotar en un fichero la información temporal obtenida y el netlist Verilog a nivel de puerta, se realiza nuevamente la misma simulación, pero esta vez incluyendo los ficheros temporales obtenidos con la tecnología seleccionada. Los datos de implementación obtenidos tras la síntesis son los siguientes:

- Número de puertos: 1407
- Número de interconexiones: 2742
- Número de celdas: 1502
- Área combinacional: 4705.646492
- Área Inversores-Buffers: 232.848008
- Área no combinacional: 7182.302595
- Consumo de potencia interno de las celdas: 130.8064 uW (99 %)
- Consumo de potencia de las interconexiones: 1.1220 uW (1 %)
- Consumo de potencia dinámico total: 131.9284 uW (100 %)
- Consumo de potencia de *leakage* de las celdas: 600.1616 nW

Una vez comprobada la correcta funcionalidad del sistema, pasamos a la parte de implementación física o back-end, donde realizamos la creación de los anillos de alimentación, el emplazamiento, la generación del árbol de reloj y rutado de las celdas. Para ello, utilizamos la información proporcionada por la herramienta de síntesis lógica, en la herramienta de *place&route* (EDI Encounter de Cadence). Tras realizar este paso, obtenemos el layout de nuestro circuito. Se realizan nuevamente simulaciones post-layout realizando verificaciones funcionales y temporales con los parásitos obtenidos de la implementación física. Una vez comprobada la funcionalidad de nuestro circuito, tenemos que el layout definitivo del bloque de control es el mostrado en la Figura 5.21 y tiene un área total de 181.32x142.45μm^2.

Pendiente de publicación.

Figura 5.21.: Layout del bloque de control.

5.5. Diseño final

Para realizar el diseño final del ASIC, el paso restante es el emplazamiento y conexionado de todos los bloques Sbox9, el bloque de control y el anillo de *pads*. En primer lugar, comentar que el anillo de *pads* del ASIC CESAR (de 68 pines en total), así como la configuración de los *pads* de salida (en esta tecnología el fabricante proporciona únicamente *pads* bidireccionales, siendo tarea del diseñador configurarlos como *pads* de entrada o salida), fue realizada por otros diseñadores, por lo que no ha sido objeto de esta Tesis.

Tras realizar el emplazamiento y conexionado de todos los bloques, se realizaron las verificaciones descritas por el fabricante (reglas DRC) y se verificaron que las descripciones a nivel de esquemático del circuito y el layout definitivo (LVS) eran correctas. Una vez finalizadas todas las comprobaciones, se generó el fichero en formato GDSII para su envío a fábrica, a través del programa miniASIC de Europractice. En la Figura 5.22 ser muestra el layout final con su conexionado entre los bloques y los *pads*. Se puede observar en la parte superior del ASIC nuestro diseño, mientras que en la parte inferior está el diseño digital correspondiente al otro proyecto.

Pendiente de publicación.

Figura 5.22.: Layout del ASIC CESAR.

El *bonding* final del ASIC con el encapsulado (JLCC-68) se muestra en la Figura 5.23. La relación entre los pines y las señales de entrada/salida y alimentación se muestra en la Tabla 5.9.

Pendiente de publicación.

Figura 5.23.: *Bonding* ASIC CESAR.

Tabla 5.9.: Relación de pines y señales de entrada/salida.

Pin	Nombre	Dirección	Descripción y comentarios
1	VDD6	Alimentación/Tierra	Alimentación del core Sbox-g (1.2V).
68	VSS6	Alimentación/Tierra	Masa del core Sbox-g.
67	VDD5	Alimentación/Tierra	Alimentación del core Sbox-f (1.2V).
66	VSS5	Alimentación/Tierra	Masa del core Sbox-f.
65	VDD2	Alimentación/Tierra	Alimentación del core Sbox-c (1.2V).
64	VSS2	Alimentación/Tierra	Masa del core Sbox-c.
63	VDD4	Alimentación/Tierra	Alimentación del core Sbox-e (1.2V).
62	VDD_RING_1	Alimentación/Tierra	Alimentación del anillo 1 (2.5V) parte Crypto.
59	VSS_RING_1	Alimentación/Tierra	Masa del anillo 1 parte Crypto.
58	VSS4	Alimentación/Tierra	Masa del core Sbox-e.
57	VDD3	Alimentación/Tierra	Alimentación del core Sbox-d (1.2V).
56	VSS3	Alimentación/Tierra	Masa del core Sbox-d.
55	VDD1	Alimentación/Tierra	Alimentación del core Sbox-b (1.2V).
54	VSS1	Alimentación/Tierra	Masa del core Sbox-b.
53	VDD0	Alimentación/Tierra	Alimentación del core Sbox-a (1.2V).
52	VSS0	Alimentación/Tierra	Masa del core Sbox-a.
51	VSS_CORE_1	Alimentación/Tierra	Masa del core 1 parte Crypto.
50	VDD_CORE_1	Alimentación/Tierra	Alimentación del core 1 (2.5V) parte Crypto.
49	VDD_CORE_2	Alimentación/Tierra	Alimentación del core 2 (2.5V) parte Control.
48	VSS_CORE_2	Alimentación/Tierra	Masa del core 2 parte Control.
47	data	Input	Entrada *data* al bloque de Control
46	reset	Input	Entrada *reset* al bloque de Control
45	VSS_RING_2	Alimentación/Tierra	Masa del anillo 2 parte Control.
41	VDD_RING_2	Alimentación/Tierra	Alimentación del anillo 2 (2.5V) parte Control.
40	control<1>	Input	Bit 1 de la entrada *control* al bloque de Control
39	control<0>	Input	Bit 0 de la entrada *control* al bloque de Control
38	clk	Input	Entrada *clk* al bloque de Control
37	clkOUT	Input	Entrada *clkOUT* al bloque de Control
36	o_Sbox	Output	Salida *o_Sbox* al bloque de Control
35	o_valid	Output	Salida *o_valid* al bloque de Control
Resto	-	Input/output	No conectado o del bloque diseño 2.

Tras su fabricación, se recibieron 10 muestras con el *bonding* ya realizado: siendo 8 de ellas selladas y 2 sin sellar. La fotografía del ASIC se muestra en la Figura 5.24 junto con una moneda de un céntimo para apreciar el tamaño del circuito.

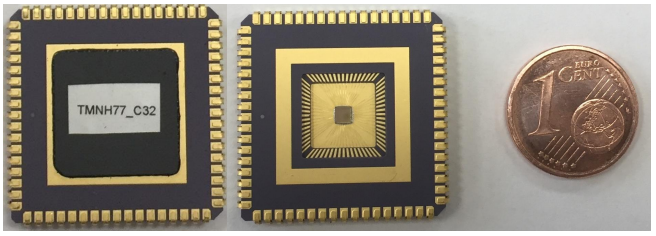

Figura 5.24.: ASIC CESAR.

En la Figura 5.25 se muestra una microfotografía del ASIC CESAR, donde se pueden apreciar las zonas ampliadas, hasta poder llegar a ver el logo, así como el *bounding* con los *pads* de salida.

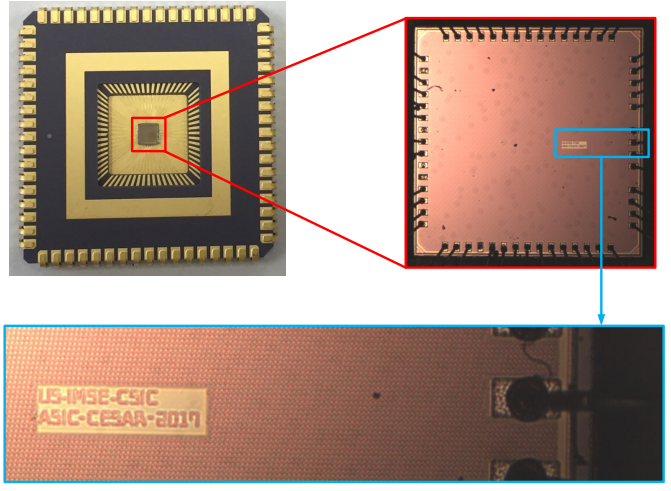

Figura 5.25.: Layout del ASIC CESAR.

5.6. Diseño de PCB y setup de medida

Para realizar el testado del ASIC en el laboratorio, se ha diseñado una PCB llamada CESAR. El esquemático de la placa se muestra en la Figura 5.27, mientras que en la Figura 5.26 se muestra una fotografía de la placa. La PCB CESAR se alimenta con 5V y a través de varios reguladores alimentamos los diferentes anillos y cores del ASIC. La selección de las Sbox9 a alimentar se realiza a través de unos jumpers, debiendo coincidir la configuración de los jumpers activados con las señales de control de las Sbox9 enviadas al bloque de control digital.

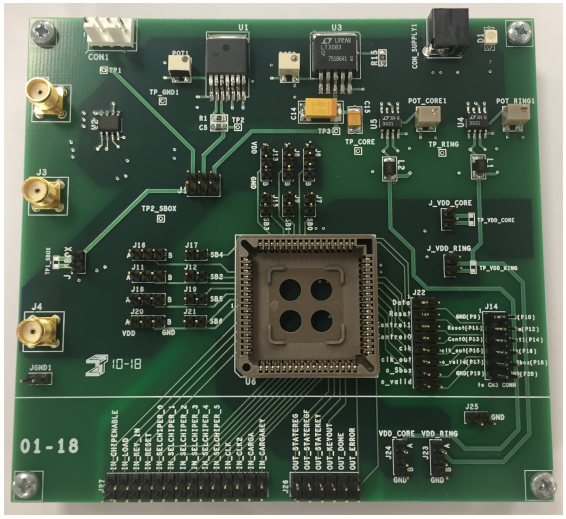

Figura 5.26.: Placa CESAR.

Para las medidas de consumo de potencia, fundamentales para el ataque DPA, vamos a tener varias opciones: medir la caída de potencial a través de una resistencia o medir directamente a la salida de un amplificador posicionado para la medida de consumo, tal y como muestra el esquemático de la Figura 5.28.

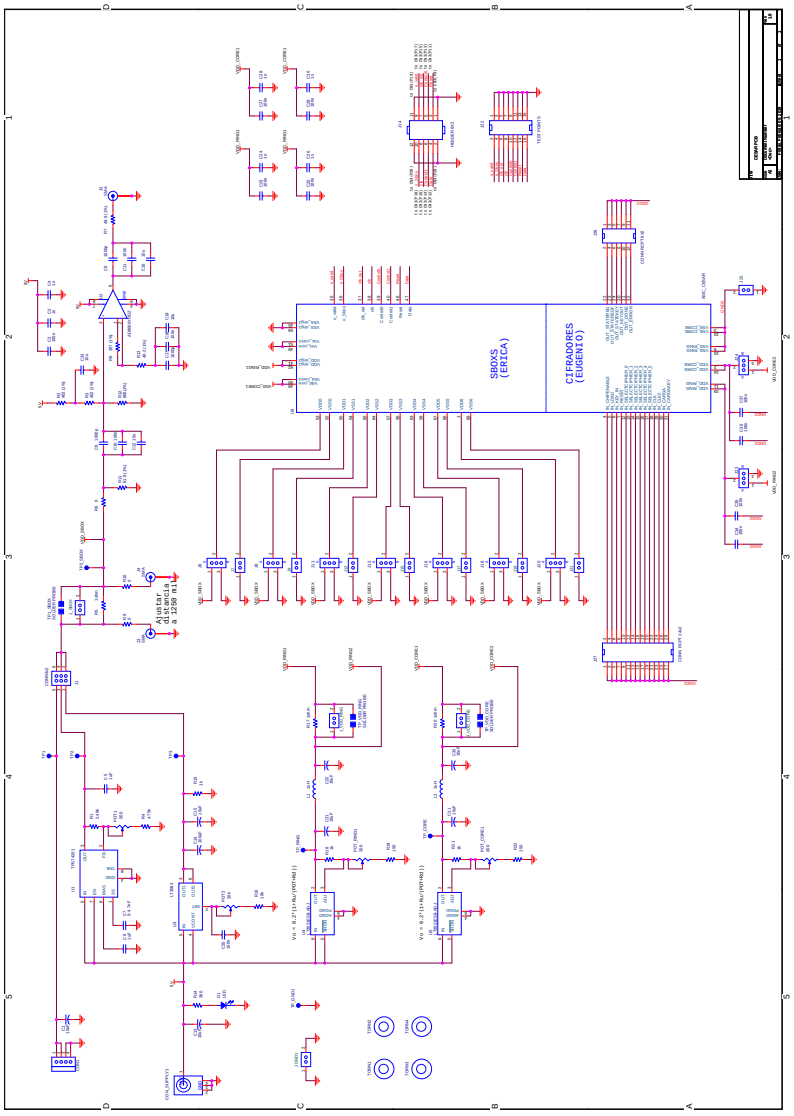

Figura 5.27.: Esquemático de la placa CESAR.

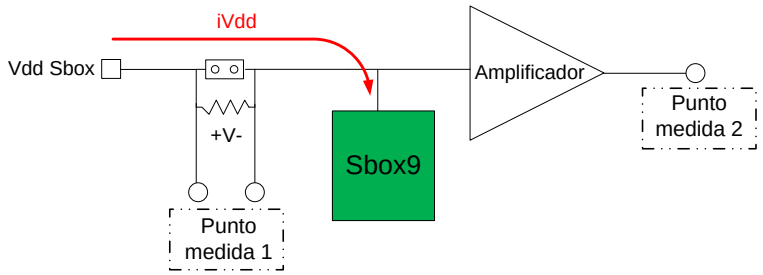

Figura 5.28.: Esquemático de la parte de medida de consumo.

Los instrumentos a utilizar en el setup experimental del ataque son:

- **PCB:** placa diseñada para alimentar el ASIC, así como para interactuar con los diferentes equipos del laboratorio
- **Fuente de alimentación**: servirá para alimentar a la PCB y ASIC.
- **Osciloscopio**: utilizado para la adquisición de las trazas de consumo o trazas electromagnéticas si fuera necesario.
- **Sondas electromagnéticas**: en caso de necesitar realizar medidas de emisión electromgnética.
- **Placa SAKURA-G**: utilizaremos la SAKURA-G para generar los estímulos del ASIC y para servir de intermediaria entre la PCB CESAR y PC.
- **PC:** utilizaremos un PC con MATLAB para controlar todos los equipos de laboratorio y el experimento.
- **ASIC**.

El plan de test para la caracterización del ASIC CESAR es el siguiente. En primer lugar se harán unas primeras medidas para comprobar la correcta funcionalidad del ASIC (test *go/no-go*), así como el correcto funcionamiento de cada una de las Sbox9 por separado. A continuación, se preparará el setup y se calibrará para poder medir las trazas de consumo de potencia o electromagnéticas para realizar tanto ataques DPA como DEMA. En el caso del ataque DEMA, se tendrá que realizar además un estudio previo para determinar la posición correcta donde colocar la sonda electromagnética, ya que tal y como se comentó en el Capítulo 2, la efectividad del

ataque depende enormemente de la precisión con la que posicionemos la sonda sobre el bloque a analizar. Tras capturar las trazas, se realizará el procesado en Matlab para determinar el MTD. El proceso de captura de trazas y obtención del MTD se realizará para tantas claves como sea posible. Finalmente, se realizarán medidas de consumo, retraso y duty-cycle para completar las medidas sobre las Sbox.

5.7. Conclusiones

En este capítulo, se ha diseñado un ASIC llamado CESAR en una tecnología de TSMC de 90nm, 1.2V y 9 metales, donde se incluyen diferentes bloques criptográficos implementados con las contramedidas propuestas en los capítulos anteriores. Para ello, se ha abordado el diseño en diferentes fases.

La primera fase ha sido la realización de una librería *full-custom* de las celdas presentadas en el capítulo 3. Se han diseñado un total de 9 celdas, utilizando los DPUN tipo SABL y DPUN tipo DDCVSL. Por otra parte, los DPDNs incluyen en alguno de los casos las soluciones *dual-switch* y *single-switch*. De esta forma, implementamos las celdas seguras que mejores prestaciones han mostrado, algunas de ellas en cuanto a seguridad (por ejemplo SABL con solución *dual/single-switch*) y otras en cuanto a área y consumo de potencia (por ejemplo, la DDCVSL con solución *dual/single-switch*).

Una vez generada la librería, se han implementado y rutado de forma semi-automática las Sbox9 utilizando la librería desarrollada. Así, se tiene un layout individual para cada una de las Sbox9, que será conectado con el bloque de control.

Para el control de todos los casos de estudio, se ha desarrollado un bloque digital que es el encargado de recibir los datos del exterior del ASIC, y transmitir los datos de las Sbox9 a la salida. Además, tras la recepción de los datos, este bloque se encarga de la activación y generación de las señales de control necesarias para activar cada una de las Sbox9. El diseño de este bloque ha seguido, a diferencia de las Sbox9, un flujo de diseño digital convencional.

El emplazamiento y conexionado *full-custom* de los bloques Sbox9 y del bloque digital se ha realizado en Virtuoso. Finalmente, tenemos el diseño del ASIC completo, realizando nuevamente todas las conexiones con los *pads* de salida de forma manual.

Una vez enviado el ASIC a fabricación a través del programa miniASIC de Europractice, se ha desarrollado una placa PCB para su testado, así como la preparación de un setup de medida para su caracterización y evaluación de la seguridad.

Aunque el test no se ha podido incluir dentro de esta Tesis, se ha detallado el plan de test, que conllevará unas primeras medidas para comprobar la correcta funcionalidad del ASIC, así como la realización de ataques DPA y DEMA, y finalmente medidas de caracterización.

6. Conclusiones finales y líneas futuras

El principal objetivo de esta Tesis ha sido aumentar el nivel de seguridad alcanzado por los criptocircuitos, analizando sus vulnerabilidades frente ataques DPA, y plantear tras la detección de las fugas de información, diferentes contramedidas a distintos niveles de abstracción para hacerlos más seguros.

Para ello, se han estudiando diferentes técnicas para la medida de la seguridad de los circuitos en función de la etapa de diseño en la que nos encontremos, y se han presentado diferentes experimentos tanto en el laboratorio como por simulación para poder determinar la efectividad de los ataques DPA y otras alternativas de evaluación de la seguridad, como el t-test. Tras estos experimentos, se puede concluir que:

- Las medidas como el NED y NSD nos permiten tener una aproximación del nivel de seguridad esperado a través de una pequeña simulación eléctrica. Sin embargo, no son válidas para determinar de forma precisa el nivel de seguridad alcanzado en un criptocircuito. Por ello, se utilizarán únicamente en una fase preliminar del diseño donde nos encontremos con numerosas alternativas a evaluar y busquemos descartar aquellas propuestas que peores resultados muestren y quedarnos solo son aquellas que son potencialmente seguras.
- Para determinar de forma precisa el nivel de seguridad de un criptocircuito, hay que llevar a cabo medidas más exhaustivas, como son la propia realización de los ataques DPA. Estas medidas son más costosas, ya que hay que implementar la estrategia de ataque y llevarla a cabo.
- Los ataques DPA y DEMA son efectivos y es posible revelar la clave secreta de un algoritmo criptográfico considerado matemáticamente seguro, monitorizando exclusivamente el consumo de potencia o emisión electromagnética y el

dato cifrado a la salida tras aplicar un número de patrones de entrada suficiente. Esto se ha demostrado realizando ataques DPA y DEMA experimentales sobre una implementación del algoritmo AES en FPGA.

- Los t-test también consiguen demostrar si un sistema tiene fugas de información, tal y como se ha demostrado al aplicar esta técnica sobre una implementación del AES en FPGA. Sin embargo, no sirven para determinar el nivel de seguridad alcanzado. Con los ataques DPA, por otro lado, podemos comparar varias implementaciones entre sí y determinar cuál de ellas es más segura.
- Tras realizar ataques DPA y DEMA a diferentes temperaturas y claves, se ha determinado: que la variación de la clave muestra variaciones en los niveles de seguridad aplicando el mismo experimento, y que las variaciones en la temperatura no producen cambios significativos en los niveles de seguridad alcanzados. Como conclusión, hay que promediar los resultados obtenidos del ataque para tantas claves como sea posible.
- Se han realizado aportaciones novedosas en el ataque al cifrador de flujo Trivium. Se ha presentado una nueva metodología de ataque DPA donde se consigue revelar la clave secreta del Trivium sin realizar ninguna hipótesis en los bits de la clave.
- Se ha presentado una optimización del ataque DPA por simulación sobre el cifrador Trivium basada en la forma de curvas de la correlación, obteniendo la clave secreta con menos patrones de entrada que con el ataque original.

Una vez estudiadas las diferentes técnicas de evaluación de la seguridad y tras llevar a cabo diferentes ataques de canal lateral, se han desarrollado contramedidas a nivel de transistor. Los estudios y propuestas de mejora presentadas a nivel de transistor han sido:

- La lógica SABL para el DPUN demuestra tener una buena relación entre seguridad y prestaciones, por lo que se selecciona como lógica de referencia.
- La lógica DDCVSL, muestra valores aceptables de seguridad con una reducción de consumo importante.
- Los nodos internos de la red DPDN muestran un efecto memoria que favorecen los ataques DPA. Se presenta una metodología de diseño seguro donde se

proponen las soluciones *dual-switch* y *single-switch* para eliminar este efecto, y los resultados demuestran una mejora de un factor al menos de x29.04 respecto a la original en los niveles de seguridad alcanzados.

- Se ha desarrollado una metodología de diseño de celdas seguras de más de dos entradas y se ha evaluado su seguridad, obteniendo diferentes resultados en la seguridad en función de la implementación resultante, pero reduciendo siempre el consumo y área ocupada.

Se han presentado contramedidas aplicadas a nivel lógico y mejora de prestaciones, determinándose que:

- Las variaciones en los tiempos de llegada de las señales en las que se focaliza el ataque DPA, elevan el nivel de seguridad.
- El uso de minimización lógica y celdas con fan-in superior a 2, provoca variaciones en la temporización que pueden afectar de forma negativa o positiva a la seguridad y dependerán del circuito a implementar. Sin embargo, se consiguen en ambos casos mejoras en cuanto a área y consumo de potencia.
- La sincronización de las señales tras utilizar técnicas para aumentar la frecuencia de operación, como pueden ser la inserción de retrasos locales o el uso de tres fases de reloj en estructuras *pipeline*, degradan los niveles de seguridad de las celdas, aunque se obtengan notables mejoras en la frecuencia de operación.

Para evaluar de forma experimental las contramedidas presentadas a ambos niveles de abstracción, se ha diseñado el ASIC CESAR donde se han llevado a cabo las siguientes tareas:

- Diseño de la librería segura CESAR, con la implementación full-custom de 9 celdas y generación de ficheros necesarios para el rutado automático en la herramienta Encounter de CADENCE.
- Diseño de 7 Sbox9 con diferentes contramedidas para evaluar las propuestas.
- Diseño de un bloque de digital para el control y activación de las Sbox9.
- Diseño de una PCB y setup de medida para el ASIC CESAR.

Como trabajo futuro a corto plazo, se realizará la caracterización completa del ASIC CESAR, así como los ataques DPA y medidas de t-test necesarias para determinar los niveles alcanzados para cada propuesta:

- Test de funcionalidad.
- Obtención de datos de consumo.
- Ataques DPA.
- Ataques DEMA.
- T-test.

Una vez determinada qué contramedida es la que mejores prestaciones ofrece, pasaremos a diseñar una librería completa que pueda utilizarse junto con otras celdas de librería estándar, caracterizándola para poder incluirla en un flujo de diseño automático. Además, se estudiará la posibilidad de incluir el rutado diferencial en la herramienta Encounter de CADENCE. De esta forma, junto con el desarrollo del setup de medida experimental, dispondremos de una librería segura y tendremos a nuestra disposición un laboratorio de caracterización criptohardware para seguir investigando en la misma linea, cuyo recorrido aún es largo y con muchos puntos por desarrollar.

Final conclusions and future work

The main objective of this thesis has been to increase the security level achieved by cryptocircuits, analyzing their vulnerabilities against DPA attacks, and to propose after the detection of information leakage, different countermeasures at different levels of abstraction to make them safer.

To this end, different techniques have been studied to measure the security level of the circuits depending on the design stage, and different experiments have been presented both in the laboratory and by simulation to determine the effectiveness of DPA attacks and other alternative security evaluation metrics, such as the t-test. After these experiments, it can be concluded that:

- Metrics such as the NED and NSD allow us to have an approximation of the expected security level through a small electrical simulation. However, they are not valid to determine precisely the level of security reached in a cryptocircuit. Therefore, they will only be used in a preliminary phase of the design where we find numerous alternatives to evaluate and discard those proposals that show worse results and remain only those that are potentially safe.
- In order to determine precisely the security level of a cryptocircuit, it is necessary to carry out more exhaustive measurements, such as carrying out DPA attacks. These measurements are more costly, since the attack strategy must be implemented and carried out.
- DPA and DEMA attacks are effective and it is possible to reveal the secret key of a cryptographic algorithm considered mathematically secure, monitoring only the power consumption or electromagnetic radiation and the encrypted data at the output after applying a sufficient number of input patterns. This has been demonstrated by performing experimental DPA and DEMA attacks on an implementation of the AES algorithm in FPGA.

- The t-tests also manage to demonstrate if a system has information leakages, as it has been demonstrated when applying this technique on an implementation of the AES in FPGA. However, they do not serve to determine the level of security achieved. With DPA attacks, on the other hand, we can compare several implementations with each other and determine which of them is more secure.
- After carrying out DPA and DEMA attacks at different temperatures and keys, it has been determined: that the variation of the key shows variations in the security levels applying the same experiment, and that the variations in the temperature do not produce significant changes in the security levels reached. As a conclusion, it is necessary to average the results obtained from the attack for as many keys as possible.
- Innovative contributions have been made to the Trivium stream cipher attack. A new DPA attack methodology has been introduced where the Trivium secret key is revealed without making any hypotheses in the key bits.
- An optimization of the DPA attack has been presented by simulation on the Trivium cipher based on the form of correlation curves, obtaining the secret key with less input patterns than with the original attack.

Once the different security evaluation techniques had been studied and after carrying out different side channel attacks, countermeasures at transistor level were developed. The studies and improvement proposals presented at transistor level have been:

- The SABL logic for the DPUN proves to have a good relationship between safety and performance, so it is selected as reference logic.
- The DDCVSL logic shows acceptable security values with a significant reduction in power consumption.
- The internal nodes of the DPDN network show a memory effect that favors DPA attacks. A secure design methodology is presented where dual-switch and single-switch solutions are proposed to eliminate this effect, and the results show an improvement of at least a factor of x29.04 with respect to the original in the security levels reached.
- A methodology for designing secure gates with more than two inputs has been developed and their safety has been evaluated, obtaining different security

results depending on the resulting implementation, but always reducing power consumption and area.

Countermeasures applied at a logic level and performance improvement have been presented, it being determined that:

- Variations in the arrival time of the signals on which the DPA attack is focused raise the level of security.
- The use of logic minimization and gates with fan-in superior to 2, causes variations in the timing that can affect in a negative or positive way the safety and will depend on the implemented circuit. However, improvements are achieved in both cases in terms of area and power consumption.
- The synchronization of the signals after using techniques to increase the operating frequency, such as the insertion of local delays or the use of three clock phases in pipelined structures, degrade the safety levels of the gates, although notable improvements in the operating frequency are obtained.

In order to evaluate in an experimental way the countermeasures presented at both levels of abstraction, the ASIC CESAR has been designed where the following tasks have been carried out:

- Design of the secure CESAR library, with the full-custom implementation of 9 gates and generation of files necessary for automatic routing in CADENCE's Encounter tool.
- Design of 7 Sbox9 with different countermeasures to evaluate the proposals.
- Design of a digital block for the control and activation of the Sbox9.
- Design of a PCB and measurement setup for the ASIC CESAR.

As a short-term future work, the complete characterization of the ASIC CESAR will be carried out, as well as the DPA attacks and t-test measurements necessary to determine the security levels reached for each proposal:

- Functionality test.
- Power consumption measurements.
- DPA attacks.

- DEMA attacks.
- T-test.

Once it has been determined which countermeasure offers the best features, we will design a complete library that can be used together with other standard library gates, characterising it so that it can be included in an automatic design flow. In addition, the possibility of including differential routing in CADENCE's Encounter tool will be studied. In this way, together with the development of the experimental measurement setup, we will have a secure library and we will have at our disposal a cryptohardware characterization laboratory to continue researching in the same line, whose route is still long and with many points to develop.

Apéndices

A. Ataque DPA a AES con contramedidas de *masking*

En el Capítulo 2 se ha mostrado de forma experimental un ataque DPA sobre el AES implementado en una FPGA Spartan6 sin contramedidas. En este apéndice, vamos a mostrar cómo el uso de contramedidas puede dificultar el ataque de forma tal que sea inviable debido al tiempo que se necesita para poder recuperar la clave o al gran coste computacional. Para ello, se va a realizar un ataque DPA sobre un AES implementado sobre la misma FPGA pero incluyendo contramedidas basadas en *masking*. En este caso, la parte del diseño del algoritmo, así como la inclusión de las contramedidas ha sido realizada por el investigador predoctoral Sadiel de la Cruz (UAB) como parte de su Tesis Doctoral y no han sido objeto de esta Tesis. Explicaremos brevemente en qué consiste la técnica de *masking* implementada y nos centraremos en la parte del estudio que ha abarcado esta Tesis: el ataque DPA experimental al AES con contramedidas.

A.1. Contramedida: Masking

Como los ataques DPA aprovechan las fugas de información relacionadas con el consumo de potencia, es decir, utilizan la relación existente entre los valores de consumo dependiendo del dato procesado, las técnicas de *masking* emplean una máscara para romper la relación lineal entre las operaciones lógicas vinculada a los datos críticos puestos en juego durante la ejecución del AES. Al ser el AES un algoritmo iterativo, esta máscara se va modificando en cada ronda dependiendo de los valores intermedios de la ejecución del algoritmo, una máscara prefijada y un número

aleatorio, tal y como se muestra en la Figura A.1. De esta forma, es difícil poder determinar exactamente el aporte del consumo debido a los datos de entrada, ya que los componentes del circuito que modelan los ataques DPA no tienen información directa de la máscara ni de cómo se está "mezclando" con los datos intermedios.

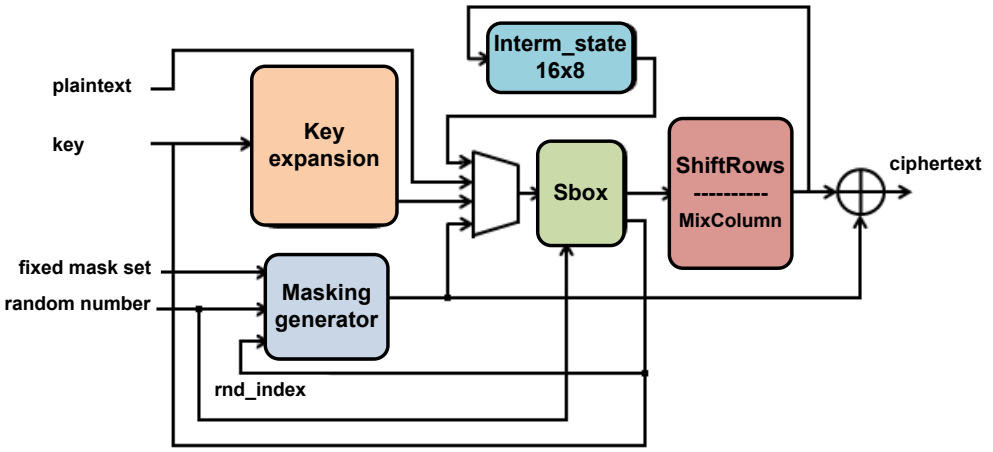

Figura A.1.: Diagrama de bloques del AES implementado con masking.

A.2. Resultados obtenidos de forma experimental

Para evaluar la mejora de la seguridad lograda con la propuesta, se ha realizado experimentalmente un ataque DPA de primer orden sobre dos implementaciones del AES: una sin contramedidas y otra protegida con *masking*. Ambos diseños se han implementado en la Spartan6 de la FPGA SAKURA-G presentada en el Capítulo 2.

El ataque se realizó en dos pasos: primero se capturaron todas las trazas de consumo correspondientes a cada *plaintext* procesado para una misma clave aleatoria, usando un script en Python ejecutado en un ordenador con Windows 7 con un procesador Intel core i7 y 8GB RAM. En una segunda fase, las trazas son procesadas con un *script* en Matlab para realizar el ataque DPA y recuperar la clave desconocida. Encripta-

mos 39500 mensajes de entrada aleatorios usando la misma clave para ambas implementaciones. La clave seleccionada fue "E401020304050607080 90A0B0C0D0E0F". El tiempo de adquisición en Python de todas las trazas es de unos 279 minutos utilizando un osciloscopio con una frecuencia de muestreo de 500 MS/s. Todos los datos son procesados en Matlab con un *script* cuya ejecución dura unos 20 segundos.

Los resultados de los ataques a ambas implementaciones se muestran a continuación. En la Figura A.2 se puede ver en el eje X el punto muestreado de la traza de potencia y en el eje Y los valores de correlación, para todas las posibles claves (256 en total para una subclave, es decir, como atacamos 8 bits tenemos 2^8 posibles claves) para la implementación sin contramedidas. En este caso, la subclave correcta se obtiene claramente alrededor del punto 3000 de la muestra en verde. En la Figura A.3, se muestran los valores de correlación (eje Y) frente al número de patrones de entrada aplicados (eje X) para el punto de correlación máximo y para todas las posibles claves. Como se puede apreciar claramente en esta figura, el MTD es aproximadamente 685, lo que significa que después de aplicar 685 patrones de entrada, el valor máximo de correlación corresponderá a la clave correcta para la implementación desprotegida de AES.

En las Figuras A.4 y A.5, se puede apreciar que no se recupera la subclave correcta para la implementación protegida después de aplicar 39500 patrones de entrada. Esto supone que el nivel de seguridad alcanzado respecto a la implementación no protegida es de al menos de un factor x57. Por lo tanto, esta técnica de *masking* consigue aumentar de forma significativa el nivel de seguridad del criptocircuito frente ataques DPA de primer orden.

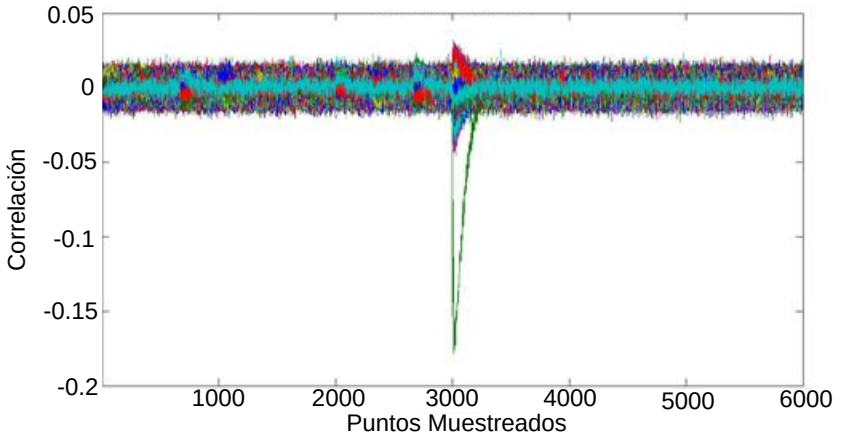

Figura A.2.: Valores de correlación para el AES sin contramedidas.

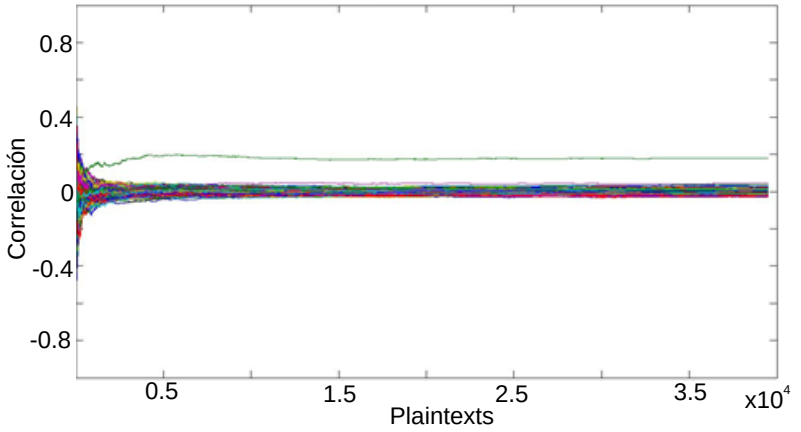

Figura A.3.: Correlación para cada clave vs patrones de entrada aplicados para el AES sin contramedidas.

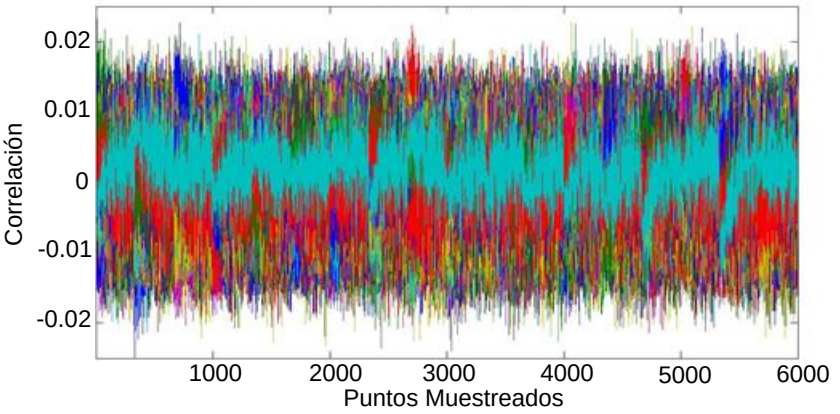

Figura A.4.: Valores de correlación para el AES con *masking*.

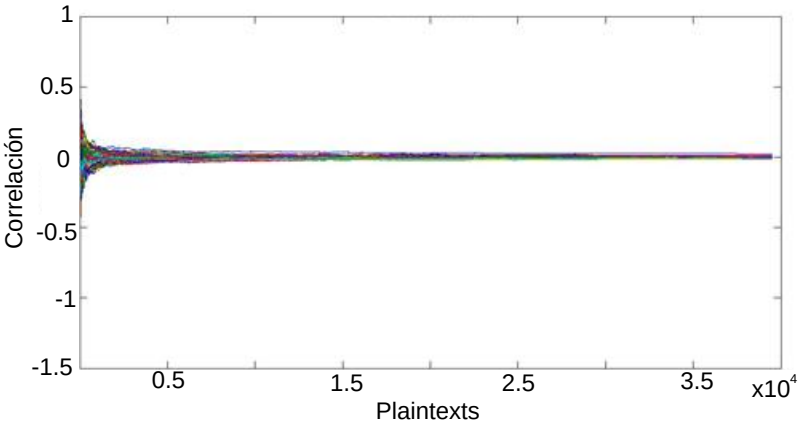

Figura A.5.: Correlación para cada clave vs patrones de entrada aplicados para el AES con *masking*.

B. Análisis de dimensiones de los transistores de las celdas SABL

El dimensionamiento de los transistores ejerce una influencia enorme en las prestaciones de las puertas. Para las celdas SABL empleadas en esta Tesis, se han seleccionado las longitudes mínimas proporcionadas por la tecnología TSMC de 90 nm (100nm) y las anchuras de canal consideradas en [81]. Las dimensiones del transistor M de las Figuras B.1 y B.2 serán objeto de optimización.

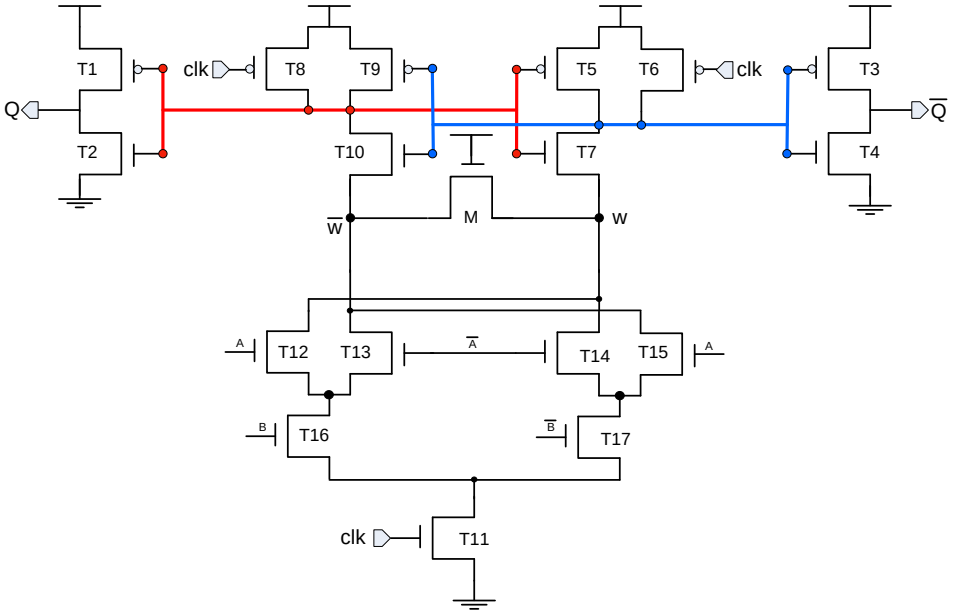

Figura B.1.: Esquemático de la celda SABL XOR/XNOR2.

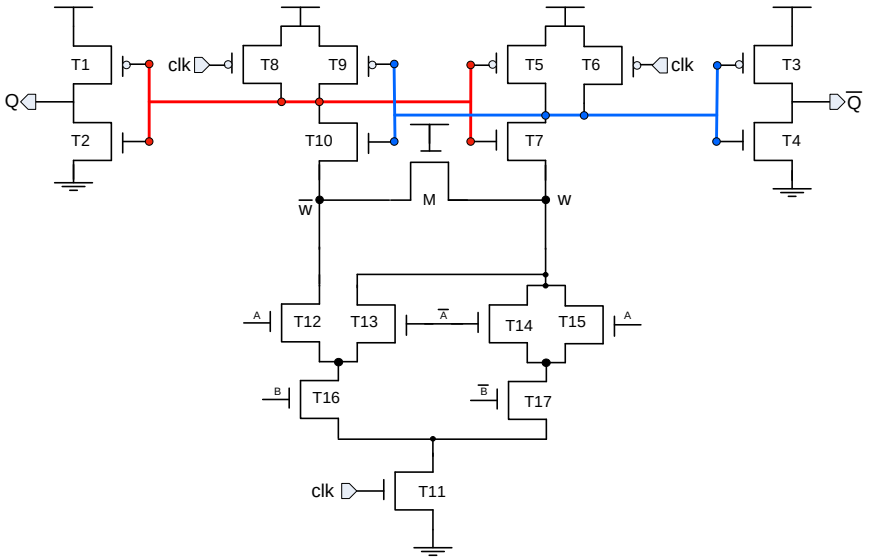

Figura B.2.: Esquemático de la celda SABL AND/NAND2.

Para optimizar las dimensiones del transistor M en lo que a su influencia en la seguridad y prestaciones se refiere, se han realizado simulaciones eléctricas de las celdas SABL XOR/XNOR2 (Figura B.1) y SABL AND/NAND2 (Figura B.2). El análisis paramétrico se ha realizado desde la anchura de 200nm, hasta los 750nm, con pasos de 50nm. Para determinar el nivel de seguridad, hemos calculado el NED y NSD para cada una de las implementaciones para las dimensiones fijadas. Los resultados se muestran en las Figuras B.3, B.4, B.5 y B.6.

En el caso de la puerta SABL XOR/XNOR2, podemos observar que los valores mínimos de NED y NSD los tenemos para la anchura W=200nm. En cambio para el caso de la puerta SABL AND/NAND2 tenemos que las variaciones del NED y NSD respecto a la anchura del transistor presentan valores menores del NED y NSD en la parte central de la gráfica (entre los 300nm y 500nm), es por ello por lo que la dimensión de este transistor la fijamos a 400nm. Los resultados finales seleccionados para las anchuras de los transistores para cada una de las celdas son las mostradas en la Tabla B.1.

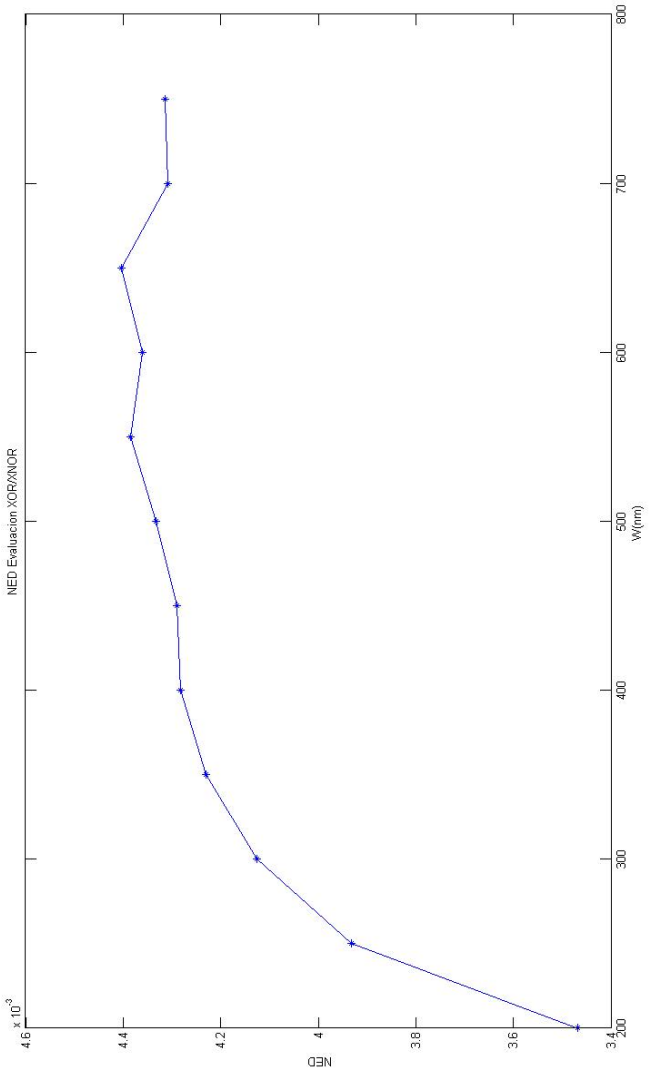

Figura B.3.: Análisis paramétrico del NED respecto de la anchura del transistor M de la puerta SABL XOR/XNOR2.

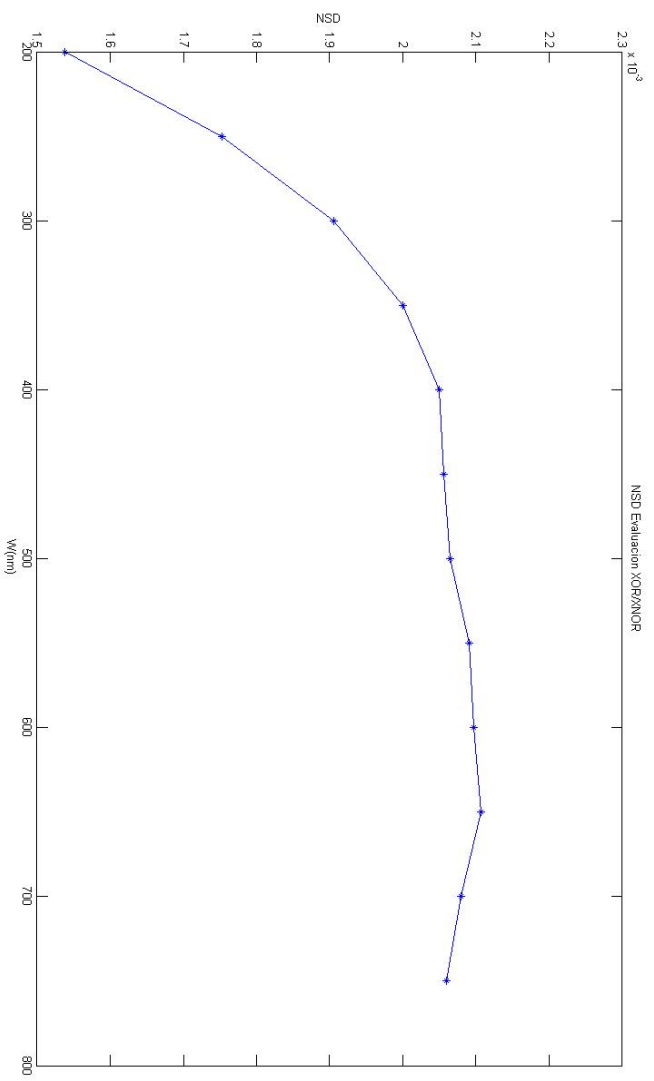

Figura B.4.: Análisis paramétrico del NSD respecto de la anchura del transistor M de la puerta SABL XOR/XNOR2.

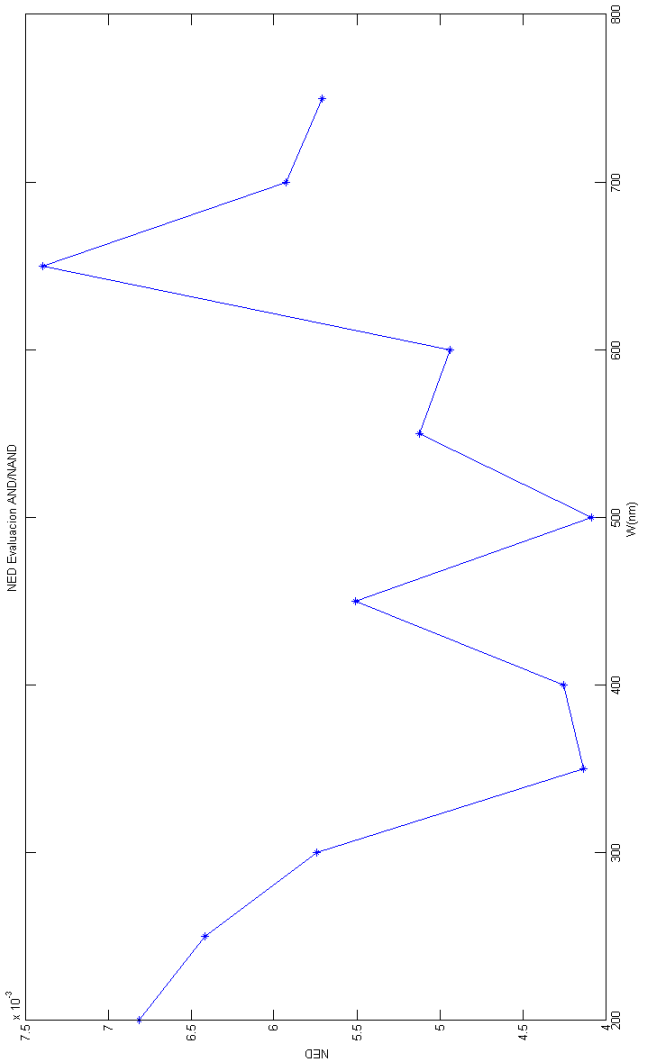

Figura B.5.: Análisis paramétrico del NED respecto de la anchura del transistor M de la puerta SABL AND/NAND2.

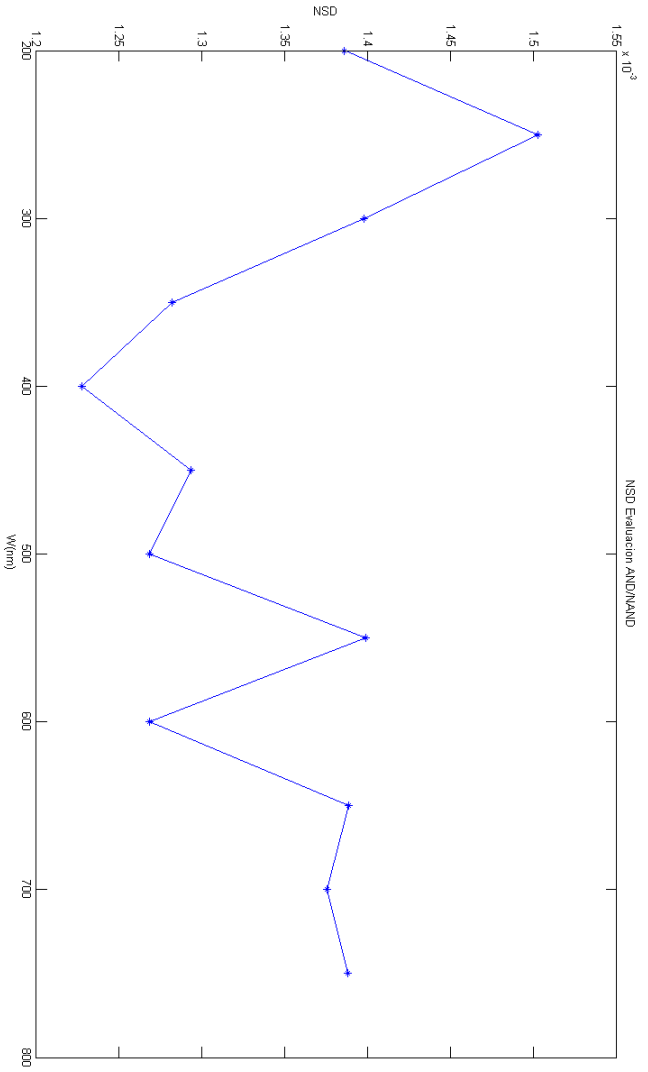

Figura B.6.: Análisis paramétrico del NSD respecto de la anchura del transistor M de la puerta SABL AND/NAND2.

Tabla B.1.: Dimensiones de los transistores de las celdas SABL_XOR/XNOR y SABL_AND/NAND(L=100nm)

Transistor	**W(nm)**	
	XOR/XNOR	AND/NAND
T1	300	300
T2	200	200
T3	300	300
T4	200	200
T5	400	400
T6	200	200
T7	800	800
T8	200	200
T9	400	400
T10	800	800
T11	800	800
T12-T17	200	200
M	200	400

C. Efecto de la variación de la temperatura en ataques DPA por simulación a Sbox9

El estudio desarrollado en este apéndice se ha presentado en la sección VI del artículo CV. 5.

D. Librería CESAR

Pendiente de publicación.

E. Implementación de las Sbox9

Pendiente de publicación.

CV Abreviado

Erica Tena Sánchez recibió el título de Ingeniera Técnica de Telecomunicaciones Especializada en Sistemas Electrónicos en el año 2010 por la Universidad de Cantabria. En el año 2012 recibió el título de Ingeniera en Electrónica por la Universidad de Sevilla, recibiendo el Premio Extraordinario Fin de Carrera. En el año 2013, obtuvo el título de Máster Universitario en Microelectrónica: Diseño y Aplicaciones de Sistemas Micro/Nanométricos por la Universidad de Sevilla.

Desde el año 2011 se encuentra desarrollando su actividad investigadora en el Instituto de Microelectrónica de Sevilla (CSIC/Universidad de Sevilla) dentro del grupo TIC-180, donde su principal linea de investigación es el diseño de criptocircuitos CMOS seguros.

Publicaciones en Revista:

CV.1 E. Tena-Sánchez, A. J. Acosta, "Logic Minimization and Wide Fan-In Issues in DPL-Based Cryptocircuits Against Power Analysis Attacks", *International Journal of Circuit Theory and Applications*, Aceptado en Noviembre 2018.

CV.2 A. J. Acosta, E. Tena-Sánchez, C. J. Jiménez, J. M. Mora, "Power and energy issues on lightweight cryptography", *Journal of Low Power Electronics*, vol. 13, no. 3, pp. 326-337, 2017.

CV.3 A. J. Acosta, T. Addabbo, E. Tena-Sánchez, "Embedded electronic circuits for cryptography, hardware security and true random number generation: an overview", *International Journal of Circuit Theory and Applications*, vol. 45, no. 2, pp. 145-169, 2017.

CV.4 P. Brox, M. C. Martínez-Rodríguez, E. Tena-Sánchez, I. Baturone, A. J. Acosta, "Application specific integrated circuit solution for multi-input multi-output

piecewise-affine functions", *International Journal of Circuit Theory and Applications*, vol. 44, no. 1, pp. 4-20, 2016.

CV.5 E. Tena-Sánchez, J. Castro, A. J. Acosta, "A Methodology for Optimized Design of Secure Differential Logic Gates for DPA Resistant Circuits", *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 2, pp. 203-215, Jun. 2014.

CV.6 P. Brox, J. Castro, M. C. Martinez-Rodríguez, E. Tena-Sánchez, C. J. Jimenez, I. Baturone, A. J. Acosta, "A Programmable and Configurable ASIC to Generate Piecewise-Affine Functions Defined Over General Partitions", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 12, pp. 3182-3194, Dec. 2013.

Publicaciones en Congresos y Workshops:

CV.7 E. Tena-Sánchez, I. M. Delgado-Lozano, J. Núñez and A. J. Acosta, "Benchmarking of nanometer technologies for DPA-resilient DPL-based cryptocircuits", *Conference on Design of Circuits and Integrated Systems (DCIS'18)*, Nov. 2018.

CV.8 E. Tena-Sánchez and A. J. Acosta, "Effect of temperature variation in experimental DPA and DEMA attacks", *International Conference on Power and Timing Modeling, Optimization and Simulation (PATMOS'18)*, Jul., 2018.

CV.9 E. Tena-Sánchez, I. Durán, S. Canas and A. J. Acosta, "Vulnerability Evaluation and Secure Design Methodology of Cryptohardware for ASIC-embedded Secure Applications to Prevent Side-Channel Attacks", *in Workshop on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE'16)*, Nov., 2016.

CV.10 E. Tena-Sánchez, A. J. Acosta, and J. Nuñez, "Secure Cryptographic Hardware Implementation Issues for High-Performance Applications", *International Conference on Power and Timing, Modeling, Optimization and Simulation (PATMOS'16)*, Sept., 2016.

CV.11 S. Canas, E. Tena-Sánchez, and A. J. Acosta, "A low-cost FPGA-based platform to perform fast Power/Electromagnetic Attacks on cryptographic circuits", *Conference on Design of Circuits and Integrated Systems (DCIS'16)*, Nov., 2016.

CV.12 E. Tena-Sánchez, "Cryptohardware design for ASIC-embedded secure applications", *International Workshop on Cryptography, Robustness, and Provably Secure Schemes for Female Young Researchers (CrossFyre'15)*, Nov., 2015.

CV.13 E. Tena-Sánchez, A. J. Acosta, "Optimized DPA attack on Trivium using correlation shape distinguishers", *Conference on Design of Circuits and Integrated Systems (DCIS'15)*, Nov., 2015.

CV.14 E. Tena-Sánchez, A. J. Acosta, "Design and characterization of cryptohardware for ASIC-embedded secure applications to prevent power analysis attacks", *Workshop on Cryptographic Hardware and Embedded Systems (CHES'15)*, Sep., 2015.

CV.15 M. C. Martínez-Rodríguez, P. Brox, E. Tena-Sánchez, A. J. Acosta, I. Baturone, "Programmable ASICs for Model Predictive Control", *IEEE International Conference on Industrial Technology (ICIT'15)*, Mar., 2015.

CV.16 E. Tena-Sánchez, A. J. Acosta, "DPA Vulnerability Analysis on Trivium Stream Cipher Using an Optimized Power Model", *IEEE International Symposium on Circuits and Systems (ISCAS'15)*, May, 2015.

CV.17 E. Tena-Sánchez, J. Castro, A. J. Acosta, "Design and Test of a Low-Power 90nm Xor/Xnor Gate for Cryptographic Applications", *International Conference on Power and Timing Modeling, Optimization and Simulation (PATMOS'14)*, Sep., 2014.

CV.18 E. Tena-Sánchez, J. Castro, A. J. Acosta, "Low-Power Differential Logic Gates for DPA Resistant Circuits", *Euromicro Conference on Digital System Design (DSD'14)*, Aug., 2014.

CV.19 E. Tena, J. Castro, A. J. Acosta, "Automatic And Systematic Control of Experimental Data Measurements on ASICs", *IMEKO Symposium Measurements of Electrical Quantities (IMEKO'13)*, Jul. 2013.

CV.20 E. Tena, J. Castro, A. J. Acosta, "Automatic and Systematic Test Toolset for Digital ASICs", *Conference on Design of Circuits and Integrated Systems (DCIS'13)*, Nov. 2013.

CV.21 S. Eiroa, J. Castro, M. C. Martinez-Rodríguez, E. Tena-Sánchez, P. Brox, I. Baturone, "Reducing bit flipping problems in SRAM physical unclonable functions

for chip identification," *19th IEEE International Conference on Electronics, Circuits and Systems (ICECS'12)*, Dec., 2012.

CV.22 M.C. Martinez-Rodríguez, P. Brox, J. Castro, E. Tena-Sánchez, A. J. Acosta, I. Baturone, "ASIC-in-the-loop methodology for verification of piecewise affine controllers," *19th IEEE International Conference on Electronics, Circuits and Systems (ICECS'12)*, Dec., 2012.

Participación en los proyectos:

CV.23 MOBY-DIC: Model-based synthesis of digital electronic circuits for embedded control (FP7-ICT-2009-4-248858), Comisión Europea.

CV.24 CITIES: Circuitos Integrados para Transmisión de Información Especialmente Segura (TEC2010-16870), Ministerio de Ciencia e Innovación.

CV.25 CESAR: Circuitos Microelectrónicos Seguros Frente a Ataques Laterales (TEC2013-45523-R), Ministerio de Economía y competitividad.

CV.26 INTERVALO: Integración y validación en laboratorio de contramedidas frente a ataques laterales en criptocircuitos microelectrónicos (TEC2016-80549-R), Ministerio de Economía y Competitividad.

CV.27 CRIPTO-BIO: Diseño microelectrónico para autenticación criptobiométrica (P08-TIC-03674), Junta de Andalucía.

CV.28 MISAL: Microelectrónica segura frente a ataques laterales (201450E034), Consejo Superior de Investigaciones Científicas (CSIC).

CV.29 LACRE: LAboratorio de Criptoanálisis HardwarE (201S50E039), Consejo Superior de Investigaciones Científicas (CSIC).

CV.30 V Plan Propio de Investigación de la Universidad de Sevilla.

Bibliografía

[1] K. Tiri and I. Verbauwhede, "Design method for constant power consumption of differential logic circuits," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'05)*. IEEE, 2005, pp. 628–633.

[2] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer Science & Business Media, 2008, vol. 31.

[3] S. Vaudenay, *A classical introduction to cryptography*. Springer, 2006.

[4] J. Katz and K. Lindell, *Introduction to modern cryptography*. CRC Press, 2015.

[5] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (iot) security: Current status, challenges and prospective measures," in *10th International Conference for Internet Technology and Secured Transactions (ICITST'15)*. IEEE, 2015, pp. 336–341.

[6] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of iot systems: Design challenges and opportunities," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'14)*. IEEE Press, 2014, pp. 417–423.

[7] NIST: National Institute of Standards and Technology. Accedido: 26-07-2018. [Online]. Available: https://www.nist.gov/

[8] FIPS PUB 46, 1977 Jan 15, available from NTIS; Springfield, VA 22151 USA. , "Data encryption standard."

[9] N. F. Pub, "197: Advanced encryption standard (AES)," *Federal information processing standards publication*, vol. 197, no. 441, p. 0311, 2001.

[10] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, *Report on lightweight cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2017.

[11] M. Katagi and S. Moriai, "Lightweight cryptography for the internet of things," *Sony Corporation*, pp. 7–10, 2008.

[12] ECRYPT, "The eSTREAM project," http://www.ecrypt.eu.org/stream/, accedido: 29-07-2018.

[13] M. Hell, T. Johansson, and W. Meier, "Grain: a stream cipher for constrained environments," *International Journal of Wireless and Mobile Computing*, vol. 2, no. 1, pp. 86–93, 2007.

[14] S. Babbage and M. Dodd, "The stream cipher mickey 2.0," *ECRYPT Stream Cipher*, 2006.

[15] C. De Cannière, "Trivium: A stream cipher construction inspired by block cipher design principles," in *International Conference on Information Security (ISC'06)*. Springer, 2006, pp. 171–186.

[16] NIST: Lightweight Cryptography. Accedido: 04-09-2018. [Online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography

[17] R. L. Rivest *et al.*, "Rfc 1321: The md5 message-digest algorithm," *Internet activities board*, vol. 143, 1992.

[18] P. FIPS, "180-1. secure hash standard," *National Institute of Standards and Technology*, vol. 17, p. 45, 1995.

[19] G. S. Vernam, "Secret signaling system," Jul. 22 1919, US Patent 1,310,719.

[20] J. Daemen and P. Kitsos, "The self-synchronizing stream cipher moustique," in *New Stream Cipher Designs*. Springer, 2008, pp. 210–223.

[21] F. Arnault, T. P. Berger, and C. Lauradoux, "Update on f-fcsr stream cipher," in *State of the Art of Stream Ciphers Workshop (SACS'06)*. Citeseer, 2006, pp. 267–277.

[22] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES'07)*. Springer, 2007, pp. 450–466.

[23] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, Nov 1976.

[24] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[25] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[26] V. S. Miller, "Use of elliptic curves in cryptography," in *International Cryptogy Coference (CRYPTO'85)*. Springer, 1985, pp. 417–426.

[27] José Miguel Mora Gutiérrez, "Implementaciones VLSI de cifradores de flujo Trivium de bajo consumo," *Tesis Doctoral, Universidad de Sevilla*, 2011.

[28] Des cracker project. Accedido: 04-09-2018. [Online]. Available: https://web.archive.org/web/20170507231657/https://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html

[29] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *International Cryptology Conference (CRYPTO'96)*. Springer, 1996, pp. 104–113.

[30] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *International Cryptology Conference (CRYPTO'99)*. Springer, 1999, pp. 388–397.

[31] D. Boneh, R. DeMillo, and R. Lipton, "On the importance of checking cryptographic protocols for faults," *Advances in Cryptology (EUROCRYPT'97)*, 1997.

[32] M. Joye and M. Tunstall, *Fault Analysis in Cryptography*. Springer, 2012, vol. 7.

[33] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 370–382, Feb 2006.

[34] D. Karaklajic, J.-M. Schmidt, and I. Verbauwhede, "Hardware designer's guide to fault attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 12, pp. 2295–2306, 2013.

[35] O. Kömmerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors." *Smartcard*, vol. 99, pp. 9–20, 1999.

[36] M. Hojsik and B. Rudolf, "Differential fault analysis of trivium," in *Fast Software Encryption*. Springer, 2008, pp. 158–172.

[37] F. E. Potestad-Ordóñez, C. J. Jiménez-Fernández, and M. Valencia-Barrero, "Fault attack on fpga implementations of trivium stream cipher," *IEEE International Symposium on Circuits and Systems (ISCAS'16)*, pp. 562–565, 2016.

[38] F. Potestad-Ordóñez, C. Jiménez-Fernández, and M. Valencia-Barrero, "Vulnerability analysis of trivium fpga implementations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, no. 12, pp. 3380–3389, 2017.

[39] T. Fukunaga and J. Takahashi, "Practical Fault Attack on a Cryptographic LSI with ISO/IEC 18033-3 Block Ciphers," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'09)*, Sept 2009, pp. 84–92.

[40] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems, "A practical implementation of the timing attack," in *Smart Card Research and Applications*. Springer, 1998, pp. 167–182.

[41] A. C. Aldaya, A. J. C. Sarmiento, and S. Sánchez-Solano, "Spa vulnerabilities of the binary extended euclidean algorithm," *Journal of Cryptographic Engineering*, vol. 7, no. 4, pp. 273–285, 2017.

[42] W. He, A. Otero, E. de la Torre, and T. Riesgo, "Customized and automated routing repair toolset towards side-channel analysis resistant dual rail logic," *Microprocessors and Microsystems*, vol. 38, no. 8, pp. 899–910, 2014.

[43] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES'01)*. Springer, 2001, pp. 251–261.

[44] Y. Hayashi, N. Homma, T. Mizuki, T. Aoki, H. Sone, L. Sauvage, and J.-L. Danger, "Analysis of electromagnetic information leakage from cryptographic devices with different physical structures," *IEEE Transactions on Electromagnetic Compatibility*, vol. 55, no. 3, pp. 571–580, 2013.

[45] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *Smart Card Programming and Security*. Springer, 2001, pp. 200–210.

[46] E. De Mulder, S. Ors, B. Preneel, and I. Verbauwhede, "Differential electromagnetic attack on an fpga implementation of elliptic curve cryptosystems," in *World Automation Congress (WAC'06)*. IEEE, 2006, pp. 1–6.

[47] M. Yoshikawa, Y. Nozaki, and K. Asahi, "Electromagnetic analysis attack for a lightweight block cipher twine," in *International Conference on Wireless Information Technology and Systems (ICWITS'16) and Applied Computational Electromagnetics (ACES'16)*. IEEE, 2016, pp. 1–2.

[48] W. He, A. Otero, E. de la Torre, and T. Riesgo, "Automatic generation of identical routing pairs for fpga implemented dpl logic," in *International Conference on Reconfigurable Computing and FPGAs (ReConFig'12)*. IEEE, 2012, pp. 1–6.

[49] D. Genkin, A. Shamir, and E. Tromer, "Acoustic cryptanalysis," *Journal of Cryptology*, vol. 30, no. 2, pp. 392–443, 2017.

[50] ——, "Rsa key extraction via low-bandwidth acoustic cryptanalysis," in *International cryptology conference (CRYPTO'14)*. Springer, 2014, pp. 444–461.

[51] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked AES hardware implementations," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES'05)*. Springer, 2005, pp. 157–171.

[52] J. Lano, N. Mentens, B. Preneel, and I. Verbauwhede, "Power analysis of synchronous stream ciphers with resynchronization mechanism," in *ECRYPT workshop SASC-The state-of-the-art of stream ciphers*, 2004.

[53] W. Fischer, B. M. Gammel, O. Kniffler, and J. Velten, "Differential power analysis of stream ciphers," in *Cryptographers Track at the RSA Conference*. Springer, 2007, pp. 257–270.

[54] C.-H. Yen and B.-F. Wu, "Simple error detection methods for hardware implementation of advanced encryption standard," *IEEE Transactions on Computers*, vol. 55, no. 6, pp. 720–731, 2006.

[55] M. Weiner, S. Manich, R. Rodríguez-Montañés, and G. Sigl, "The low area probing detector as a countermeasure against invasive attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 2, pp. 392–403, 2018.

[56] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541–552, 2002.

[57] B. J. Gilbert Goodwill, J. Jaffe, and P. Rohatgi, "A testing methodology for side-channel resistance validation," in *NIST non-invasive attack testing workshop*, vol. 7, 2011, pp. 115–136.

[58] Y. Wang and Y. Ha, "FPGA-based 40.9-Gbits/s masked AES with area optimization for storage area network," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 1, pp. 36–40, 2013.

[59] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Proceedings of the 28th European Solid-State Circuits Conference (ESSCIRC'02)*. IEEE, 2002, pp. 403–406.

[60] K. Tiri, D. Hwang, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "Prototype IC with WDDL and differential routing–DPA resistance assessment," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES'05)*. Springer, 2005, pp. 354–365.

[61] K. Tiri and I. Verbauwhede, "A VLSI design flow for secure side-channel attack resistant ICs," in *Proceedings of the conference on Design, Automation and Test in Europe (DATE'05)*, vol. 3. IEEE Computer Society, 2005, pp. 58–63.

[62] M. Alioto, S. Bongiovanni, M. Djukanovic, G. Scotti, and A. Trifiletti, "Effectiveness of leakage power analysis attacks on DPA-resistant logic styles under process variations," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 2, pp. 429–442, 2014.

[63] H. Guntur, J. Ishii, and A. Satoh, "Side-channel attack user reference architecture board sakura-g," in *IEEE 3rd Global Conference on Consumer Electronics (GCCE'14)*. IEEE, 2014, pp. 271–274.

[64] T. Schneider and A. Moradi, "Leakage assessment methodology," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES'15)*. Springer, 2015, pp. 495–513.

[65] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi *et al.*, "Test vector leakage assessment (tvla) methodology in practice," in *International Cryptographic Module Conference (ICMC'13)*, vol. 1001, 2013, p. 13.

[66] Y. Jia, Y. Hu, F. Wang, and H. Wang, "Correlation power analysis of trivium," *Security and Communication Networks*, vol. 5, no. 5, pp. 479–484, 2012.

[67] M. Kirschbaum and T. Popp, "Evaluation of a DPA-resistant prototype chip," in *Annual Computer Security Applications Conference (ACSAC'09)*. IEEE, 2009, pp. 43–50.

[68] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Proceedings of the conference on Design, Automation and Test in Europe (DATE'04)*, vol. 1. IEEE, 2004, pp. 246–251.

[69] M. W. Allam and M. I. Elmasry, "Dynamic current mode logic (DyCML): A new low-power high-performance logic style," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 3, pp. 550–558, 2001.

[70] I. Hassoune, F. Macé, D. Flandre, and J.-D. Legat, "Low-swing current mode logic (LSCML): A new logic style for secure and robust smart cards against power analysis attacks," *Microelectronics Journal*, vol. 37, no. 9, pp. 997–1006, 2006.

[71] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES'05)*. Springer, 2005, pp. 172–186.

[72] B.-D. Choi, K. E. Kim, K.-S. Chung, and D. K. Kim, "Symmetric adiabatic logic circuits against differential power analysis," *ETRI journal*, vol. 32, no. 1, pp. 166–168, 2010.

[73] D. Suzuki and M. Saeki, "Security evaluation of dpa countermeasures using dual-rail pre-charge logic style," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES'05)*. Springer, 2006, pp. 255–269.

[74] J.-L. Danger, S. Guilley, S. Bhasin, M. Nassar, and L. Sauvage, "Overview of dual rail with precharge logic styles to thwart implementation-level attacks on hardware cryptoprocessors," 2009.

[75] R. Soares, N. Calazans, V. Lomné, P. Maurine, L. Torres, and M. Robert, "Evaluating the robustness of secure triple track logic through prototyping," in *Proceedings of the 21st annual symposium on Integrated circuits and system design (SBCCI'08)*. ACM, 2008, pp. 193–198.

[76] M. Nassar, S. Bhasin, J.-L. Danger, G. Duc, and S. Guilley, "BCDL: a high speed balanced DPL for FPGA with global precharge and no early evaluation," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'10)*. European Design and Automation Association, 2010, pp. 849–854.

[77] K. Tiri and I. Verbauwhede, "A digital design flow for secure integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 7, pp. 1197–1208, 2006.

[78] M. Bucci, L. Giancane, R. Luzzi, and A. Trifiletti, "Three-phase dual-rail precharge logic," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES'06)*. Springer, 2006, pp. 232–241.

[79] M. Bucci, L. Giancane, R. Luzzi, G. Scotti, and A. Trifiletti, "Delay-based dual-rail precharge logic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 7, pp. 1147–1153, 2011.

[80] B. Nikolic, V. G. Oklobdzija, V. Stojanovic, W. Jia, J. K.-S. Chiu, and M. M.-T. Leung, "Improved sense-amplifier-based flip-flop: Design and measurements," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 6, pp. 876–884, 2000.

[81] Javier Castro Ramírez, "Desarrollo y aplicaciones de t'ecnicas de control de corriente de alimentación en circuitos integrados digitales CMOS," *Tesis Doctoral, Universidad de Sevilla*, 2017.

[82] P. Ng, P. T. Balsara, and D. Steiss, "Performance of CMOS differential circuits," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 6, pp. 841–846, Jun 1996.

[83] *Specification of the 3GPP confidentiality and integrity algorithms; Document 2: KASUMI specification, ETSI/SAGE*, 2007.

[84] I. Levi, O. Keren, and A. Fish, "Data-dependent delays as a barrier against power attacks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 8, pp. 2069–2078, 2015.

[85] I. Hammad, K. El-Sankary, and E. El-Masry, "High-speed AES encryptor with efficient merging techniques," *IEEE Embedded Systems Letters*, vol. 2, no. 3, pp. 67–71, 2010.

[86] P. Abhijith, M. Goswami, S. Tadi, and K. Pandey, "Optimized Architecture for AES," *IACR Cryptology ePrint Archive*, vol. 2014, p. 540, 2014.

[87] U. Hussain and H. Jamal, "An efficient high throughput FPGA implementation of AES for multi-gigabit protocols," in *10th International Conference on Frontiers of Information Technology (FIT'12)*. IEEE, 2012, pp. 215–218.

[88] D. Harris and M. A. Horowitz, "Skew-tolerant domino circuits," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, pp. 1702–1711, 1997.

[89] R. Hossain, *High performance ASIC design: using synthesizable domino logic in an ASIC flow*. Cambridge University Press Cambridge, 2008.

[90] S. Horne, D. Glowka, S. McMahon, P. Nixon, M. Seningen, and G. Vijayan, "Fast/sub 14/technology: design technology for the automation of multi-gigahertz digital logic," in *International Conference on Integrated Circuit Design and Technology (ICICDT'04)*. IEEE, 2004, pp. 165–173.

[91] W. Belluomini, D. Jamsek, A. Martin, C. McDowell, R. Montoye, T. Nguyen, H. Ngo, J. Sawada, I. Vo, and R. Datta, "An 8GHz floating-point multiply," in *IEEE International Solid-State Circuits Conference (ISSCC'05)*. IEEE, 2005, pp. 374–604.

[92] A. Gornik, A. Moradi, J. Oehm, and C. Paar, "A hardware-based countermeasure to reduce side-channel leakage: Design, implementation, and evaluation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1308–1319, 2015.