Trabajo Fin de Máster
Máster Universitario en Ingeniería de Telecomunicación

Diseño Back-end VLSI para Redes Neuronales Convolucionales basadas en Computación Estocástica en Tecnología de 40 nm

Autor: Iván Díez de los Ríos Luis

Tutores: M. Mar Elena Pérez

Alejandro Barriga-Rivera

Tutor externo: Luis A. Camuñas Mesa

Dptos. Ingeniería Electrónica y Física Aplicada III
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Máster
Máster Universitario en Ingeniería de Telecomunicación

Diseño Back-end VLSI para Redes Neuronales Convolucionales basadas en Computación Estocástica en Tecnología de 40 nm

Autor:
Iván Díez de los Ríos Luis

Tutores:
M. Mar Elena Pérez
Profesora contratada doctora
y
Alejandro Barriga-Rivera
Profesor contratado doctor interino

Tutor externo:
Luis A. Camuñas Mesa
Profesor contratado doctor interino

Dptos. Ingeniería Electrónica y Física Aplicada III
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Máster: Diseño Back-end VLSI para Redes Neuronales Convolucionales basadas en Computación Estocástica en Tecnología de 40 nm

Autor: Iván Díez de los Ríos Luis
Tutores: M. Mar Elena Pérez y Alejandro Barriga-Rivera
Tutor externo: Luis A. Camuñas Mesa

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Cerrar otra etapa de estudios, suele ser un momento ansiado y deseado, en especial, cuando el último paso parece no tener fin. Por eso, me es muy gratificante terminar este trabajo, el cual no habría podido realizar solo.

En primer lugar, quiero agradecer a Macarena que ha estado día a día a mi lado y que me ha dado las fuerzas necesarias para sentarme a escribir cada una de estas páginas. Es ella la que es capaz de conseguir la mejor versión de mí mismo y que cada día siga progresando. A mis padres, que me han dado todo para que sea quien soy. Agradecer también al resto de mi familia, tanto a los de siempre como a los nuevos. También quiero recordar a mis amigos a los que tantas veces les he pospuesto algún plan por falta de tiempo.

Agradezco también a aquellos que han recorrido este camino al mismo tiempo que yo, José Manuel, Pablo, Eykis, Ángela, Edgardo y Francisco, que fueron grandes compañeros y amigos cuando, además, nos encontramos en la tesitura de tener que afrontar la carga del máster desde casa. Por supuesto, también quiero recordar a los profesores que nos guiaron y acompañaron durante ese período y en especial a mis tutores en este trabajo, Mar y Alejandro, sin cuyo apoyo y confianza hubiese tomado un camino diferente a este. Ellos me descubrieron el mundo de la investigación y abrieron una ventana de oportunidades.

A mis compañeros de trabajo tanto de Applus como del GRVC de los cuales guardo buenos recuerdos y amistad.

Aunque este trabajo cierra un ciclo dentro de mi vida académica, aún me quedan caminos por recorrer. En este trayecto me acompañan un buen número de grandes compañeros y compañeras, algunos con los que comparto edificio y otros que estoy conociendo a través del mundo. Destacar, sin duda, a las personas que me han dado la oportunidad y me ayudan a sacar lo mejor de mí, Luis, Teresa y Bernabé, y a todo el Instituto de Microelectrónica de Sevilla.

Si bien muchas personas me acompañan y me han acompañado en muchos ámbitos de mi vida, también quiero dedicar un pensamiento hacia una parte muy importante de mi vida que ha sido, es y será la música. A todas las notas que han sonado mientras estaba pensando frente al ordenador y los buenos momentos que paso fantaseando en el mundo de las 88 teclas rodeado de compases y acordes.

Por último, agradecer a todas las personas que pudieran estar leyendo estas líneas, pues son ellas las que hacen que estas palabras encerradas entre páginas, o documentos digitales, broten fuera de su medio y cobren algún valor.

Iván Díez de los Ríos Luis
Investigador Predoctoral IMSE,CNM,CSIC-US

Sevilla, 2023

Resumen

Para la implementación de circuitos integrados digitales se requiere un complejo proceso que abarca desde el planteamiento del problema a alto nivel, continuando con el diseño a nivel de comportamiento y el diseño de *Back-end*, para dar paso al proceso de fabricación, donde se convierte un pequeño trozo de silicio en un dispositivo capaz de procesar enormes cantidades de datos a altas velocidades.

Una de las fases que se requieren para obtener un microchip digital es el proceso de síntesis. En este trabajo se expone cómo se sintetizaron una serie de diseños digitales para redes neuronales convolucionales y los resultados que se obtuvieron de la solución de alto nivel previamente propuesta. Asimismo, se pretende explicar el contexto donde se enmarca el circuito planteado y el proceso de fabricación de un microchip digital *Application-Specific Integrated Circuit* (ASIC) con tecnología de 40 nm de TSMC. Por último, se explican los pasos realizados, los resultados obtenidos y la ventaja del uso de estas técnicas para conseguir elaborar diseños digitales de forma mucho más rápida y eficiente.

Abstract

The implementation of digital integrated circuits is a complex process that spans from high-level problem formulation, continuing with behavioral-level design and back-end design, to enter the manufacturing process, where a small piece of silicon is turned into a device capable of processing huge amounts of data at high speeds.

One of the phases required to obtain a digital microchip is the synthesis process. This master thesis describes how a series of digital designs for convolutional neural networks were synthesized and the results obtained from the previously proposed high-level solution. Likewise, this work aims to explain the context in which the proposed circuit is framed and the process of manufacturing a 40 nm TSMC digital *Application-Specific Integrated Circuit* (ASIC) microchip. Finally, it explains the steps taken, the results obtained, and the advantage of using these techniques to achieve much faster and more efficient digital designs.

Prólogo

Este Trabajo Fin de Máster explica una parte concreta de un proyecto más amplio sobre redes neuronales convolucionales que nace de la colaboración de un grupo de investigación de la Universitat de les Illes Balears (UIB), dirigido por Josep Lluis Rosselló, y el Instituto de Microelectrónica de Sevilla (IMSE).

En el Instituto de Microelecrónica de Sevilla se ha contado con la supervisión de Luis Camuñas Mesa, Teresa Serrano Gotarredona y Bernabé Linares Barranco al comienzo de mi estancia en dicho centro. Ellos han sido los responsables de guiarme y dirigirme para adquirir los conocimientos necesarios para la realización de las tareas de síntesis digital y comprender el contexto del proyecto en el que se engloban estas actividades.

Además, he contado con la ayuda de mis tutores y profesores dentro de la Escuela Técnica Superior de Ingenieros de la Universidad de Sevilla, Mar Elena Pérez y Alejandro Barriga-Rivera, los cuales han colaborado en la supervisión de este documento.

Para presentar este trabajo he decidido hacer un primer capítulo de Introducción con el objetivo de contextualizar el proyecto general entre la UIB y el IMSE. Ahí se tratan la Inteligencia Artificial, la necesidad de un nuevo Hardware orientado a la Inteligencia Artificial y una antigua rama de la computación, la Computación Estocástica, que en su día no generó un gran impacto. Esta Computación Estocástica ha sido rescatada en varias ocasiones por investigadores[1] para proponer su uso como un método para obtener eficiencia en sistemas de Inteligencia Artificial.

Como he comentado antes, la labor que hemos realizado desde nuestro grupo, y la mía en particular, es la de asistir en la parte más técnica del diseño. En el segundo capítulo se detalla de forma general el flujo de trabajo para el diseño de circuitos VLSI, allí se trata la creación de un chip digital desde la idea hasta su empaquetado y testado. Los primeros pasos del flujo (*front-end*) forman parte del trabajo realizado por el grupo de la UIB; la síntesis, el diseño físico y la verificación del diseño físico (*back-end*) son labores que hemos o estamos llevando a cabo en el IMSE; por supuesto, el resto tendrá que hacerse por una empresa especializada.

Así pues este trabajo recoge los detalles del proceso de síntesis igital para un caso concreto, circuitos VLSI para Redes Neuronales Convolucionales basadas en Computación Estocástica, y usando una Tecnología de 40 nm de TSMC.

Iván Díez de los Ríos Luis

[1] En la sección 1.3 se encuentran las referencias a algunos trabajos que han usado dichas técnicas.

Índice Abreviado

Índice

1 Introducción

Los grandes conocimientos engendran las grandes dudas.

ARISTÓTELES

Desde hace muchos años, el ser humano viene soñando y fantaseando con la Inteligencia Artificial o *Artificial Intelligence* (*AI*) en inglés. Son muchas las novelas que se han redactado a lo largo de los años que describen un tipo de vida artificial que intenta imitar al ser humano. Ya en la antigüedad, Herón de Alejandría escribió un tratado titulado *Los Autómatas*, considerado los inicios de la robótica. Basta pensar un poco para encontrar muchos ejemplos de seres artificiales que intentan imitar al *homo sapiens*, seres construidos a base de madera, metal u otros organismos vivos. Seres inanimados que transgreden las normas de la naturaleza para convertirse en "seres vivos".

Si bien se han planteado muchas formas para crear estas "vidas", una parte de las mismas ha acaparado más la atención del ser humano durante mucho tiempo. Son múltiples los tipos de seres vivos, ya sean del reino vegetal, animal, protoctistas, hongos o móneras, sin embargo, uno de los más intrigantes misterios acerca de esos seres que cobran vida es su inteligencia, su mente o su espíritu. Precisamente el punto que más nos separa del resto de animales, la inteligencia, la capacidad de hablar, razonar o tomar decisiones, es el tema más tratado dentro de los seres artificiales. Tal vez sea la soledad de la especie humana, en cuanto a inteligencia se refiere, la fuente de motivación de la que beben todos estos relatos.

1.1 Sobre Inteligencia Artificial

Aunque muchos escritores gustan dejándose llevar por sus sueños y fantasías, el mundo de la investigación tiende más a la búsqueda del entendimiento del mundo que le rodea y aprovechar esos conocimientos para intentar moldearlo.

La biología y la medicina han dedicado un arduo trabajo a la comprensión del sistema nervioso. Desde los primeros estudios fisiológico hasta la comprensión de las neuronas destacan nombres como Ramón y Cajal, Sherrington, Hodgkin y Huxley[15], entre otros.

El nacimiento de lo que se conoce a día de hoy como *AI* ocurre en 1958 con Frank Rosenblatt y la creación del Perceptrón[37]. Este invento intenta imitar el funcionamiento de las neuronas reales. Visto desde una perspectiva del funcionamiento y la transmisión de información, la neurona se divide en tres partes: el cuerpo de la célula, llamado soma; las dendritas, que son las entradas de la neurona; y el axón, que es la salida de la neurona [4]. Véase la figura 1.1 donde se muestran las dendritas, el soma y el axón.

Las neuronas son células especializadas que forman el sistema nervioso y son responsables de la transmisión de información. El funcionamiento de una neurona biológica se puede describir, de forma simplificada, en tres etapas: recepción, integración y transmisión.

En la etapa de recepción, la neurona recibe señales eléctricas y químicas de otras neuronas o de células sensoriales a través de sus dendritas, que son prolongaciones ramificadas que se conectan con las terminales de otras neuronas. Las señales eléctricas son transformadas en señales químicas mediante la liberación de neurotransmisores en la sinapsis.

En la etapa de integración, la neurona procesa y suma la información recibida para determinar si debe enviar una señal eléctrica a otras neuronas o músculos. Esta integración se lleva a cabo en el soma de la

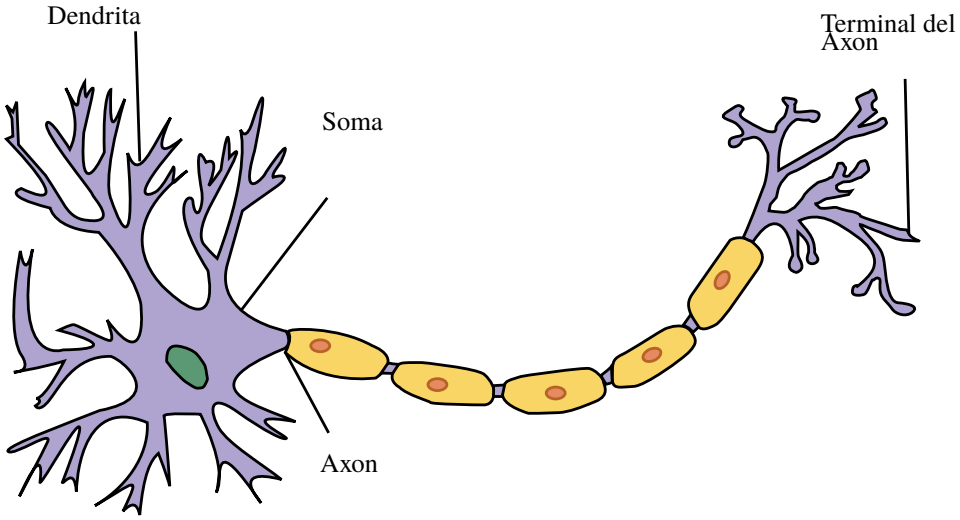


Figura 1.1 La neurona y sus partes. Principalmente, las partes más destacadas a nivel funcional son las dendritas, el axón y el soma. Imagen modificada, original obtenida de Wikimedia Commons bajo licencia CC BY-SA 3.0. Fuente: *https://commons.wikimedia.org/wiki/File:Neurona.svg*.

neurona, donde se concentran las dendritas y se produce el potencial de acción, que es la señal eléctrica que viaja a lo largo del axón.

Finalmente, en la etapa de transmisión, la señal eléctrica se propaga por el axón de la neurona, que es una prolongación larga y delgada que se conecta con otras neuronas o músculos. La transmisión de la señal se lleva a cabo mediante la liberación de neurotransmisores en las terminales axónicas, que activan o inhiben a las neuronas o músculos conectados.

Si bien el comportamiento del Perceptrón y lo que aquí se explica difiere un poco del funcionamiento real, existe otra rama de la *AI*, los *Sistemas Neuromórficos*[41, 21, 59, 13, 11, 55], que sí trabaja de forma más análoga a los sistemas orgánicos mediante redes neuronales de *spiking* conocidas como *Spiking Neural Network* (*SNN*).

Ahora bien, si las neuronas reales reciben y emiten pequeños pulsos eléctricos, el Perceptrón recibe y devuelve números, y las sinapsis neuronales son modeladas mediante unos pesos w_i. La salida que se produce en el Perceptrón, $y[n]$, responde a las ecuaciones siguientes:

$$g[n] = w_0 + w_1 \cdot x_1[n] + w_2 \cdot x_2[n] + \cdots + w_i \cdot x_i[n] \tag{1.1}$$

$$y[n] = f_{act}(g[n]) \tag{1.2}$$

donde w_0 es un término que no depende de ninguna entrada y se conoce como sesgo o *bias*, las x_i son los valores de entrada en cada dentrita i, y f_{act} es llamado la función de activación.

La función de activación es de especial importancia y su principal atractivo es producir no linealidades en la salida. El lector puede comprobar que el resultado de $g[n]$ es lineal para (1.1), sin embargo las neuronas biológicas tienen salidas que no lo son, la función de (1.2) es modular este efecto, para ello pueden usarse un gran número de funciones no lineales entre las que destacan funciones *Rectified Linear Unit* (*ReLU*), que se define como $f_{act}(x) = max(0,x)$, cuya salida es igual a x para cualquier valor de $x > 0$ y 0 en otro caso, la tangente hiperbólica $\tanh(x)$, la función sigmoide $\sigma(x)$ o la función escalón $u(x)$.

1.1.1 Redes Neuronales

Como ocurriría en un organismo vivo, una neurona aislada no podría considerarse inteligente, la capacidad de aprender, pensar o tomar decisiones reside en el conjunto de todas las neuronas interconectadas entre sí. Dentro del *Machine Learning* (*ML*) y la *AI* también existe la interconexión entre varias neuronas (perceptrones), para crear una red, la cual recibe comúnmente el nombre de *Artificial Neural Network* (*ANN*). En los diseños más sencillos la red se estructura en una serie de capas, cada una con un número diferente o no de neuronas, un ejemplo se muestra en la figura 1.2. La interconexión se genera entre una capa y la posterior, de forma que, en principio, la salida de cada neurona de una capa irá conectada a cada una de las neuronas de la siguiente

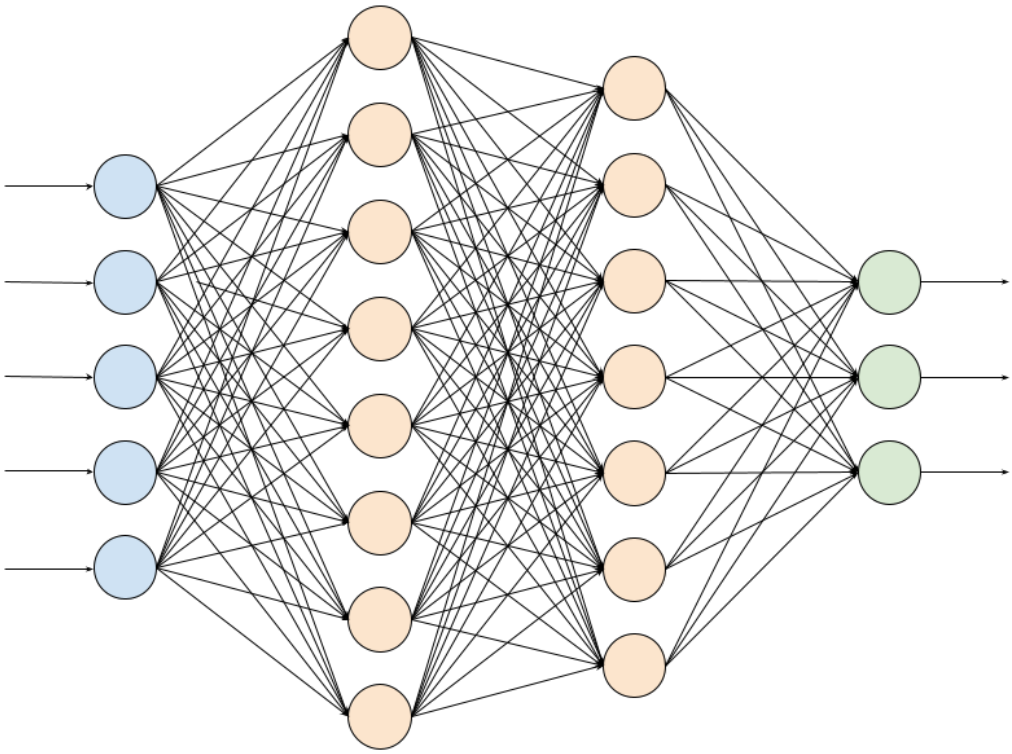


Figura 1.2 Esquema de una red neuronal con 5 neuronas de entrada y tres de salida. La red se organiza por capas que pueden tener distinto número de neuronas y entre cada capa, todas las neuronas de una capa están conectadas con todas las de la siguiente.

capa. Existen redes mucho más complejas donde las neuronas no están conectadas por capas, pero en este trabajo se simplifica a este modelo.

Gracias a los pesos de cada neurona puede ajustarse la importancia que cada entrada representa para la salida de dicha neurona. Se considera la primera capa de neuronas a los mismos sensores o dispositivo que recoge el valor. Visto desde el punto de vista del sistema, si se considerase una imagen, cada uno de los píxeles de dicha imagen formarían la primera capa, la cual recibiría el nombre de capa de entrada. En la última capa puede haber diferentes configuraciones, pero una de las más sencillas podría ser usar una sola neurona que discrimine si el resultado de la red neuronal es afirmativo o positivo, lo cual podría traducirse con que exista salida mayor que cero o no[1].

Las *ANN* representan una de las arquitecturas más sencillas de implementar y son capaces de generar el resultado de una regresión de funciones no lineal con alto nivel de complejidad, tareas de clasificación u obtener la probabilidad de que una imagen de entrada represente un número, por ejemplo. Pero lo más importante de este paradigma de computación es la forma de conseguir el resultado, pues las *ANN* pueden aprender estas tareas a base de entrenamiento.

Existen diferentes métodos de entrenamiento o aprendizaje en *ML*, pero aquí sólo se hablará de *aprendizaje supervisado* y *aprendizaje no supervisado*. En el primero se realiza de forma que la red recibe unas entradas y una etiqueta de salida, la etiqueta puede ser equivalente a la salida de la *ANN*. Por ejemplo, si se entrenase el sistema para que detecte si una imagen muestra un globo rojo o verde, dicha etiqueta podría indicar si esa imagen concreta tiene un globo de un color u otro. Por tanto, en esta modalidad la red recibe las entradas y las salidas, el proceso de entrenamiento debe ayudarle a ser capaz de generar dicha salida con unas entradas similares. En entrenamiento no supervisado no es necesario entregar ninguna etiqueta de salida, este tipo de entrenamiento suele utilizarse para tareas de *clusterización*, entre otras.

Para conseguir entrenar las redes neuronales usando entrenamiento supervisado, es necesario definir una función de coste que medirá el error de la predicción y se intentará minimizar dicho error en base a formulas de optimización de forma reiterativa. Los parámetros a optimizar serán los pesos de cada sinapsis. Una vez entrenadas, se llama *inferencia* al proceso de obtener una predicción por la *ANN* ya entrenada.

[1] Puede el lector observar que esta salida es equivalente a la de la función *ReLU* comentado con anterioridad.

Redes neuronales con unas pocas capas son capaces de inferir con una alta precisión en una gran cantidad de entornos diferentes.

1.1.2 Deep Learning

Muchas técnicas de *AI* fueron desarrolladas hace varias décadas, sin embargo, la capacidad de cómputo del momento no permitía que pudieran explotarse adecuadamente. Es por ello que la explosión en el desarrollo y uso de todas estas tecnologías ha florecido en los últimos años[27].

Las mejoras en cuánto a capacidad de procesamiento no sólo han permitido el uso de dichas técnicas, sino que han facilitado que crezcan de forma exponencial, dando lugar al *Deep Learning*[25, 39]. Esta rama del *ML* y la *AI* comprende modelos de redes neuronales de un cierto tamaño con capacidad para obtener un aprendizaje más profundo. Actualmente, existen redes neuronales con miles de millones de parámetros[2].

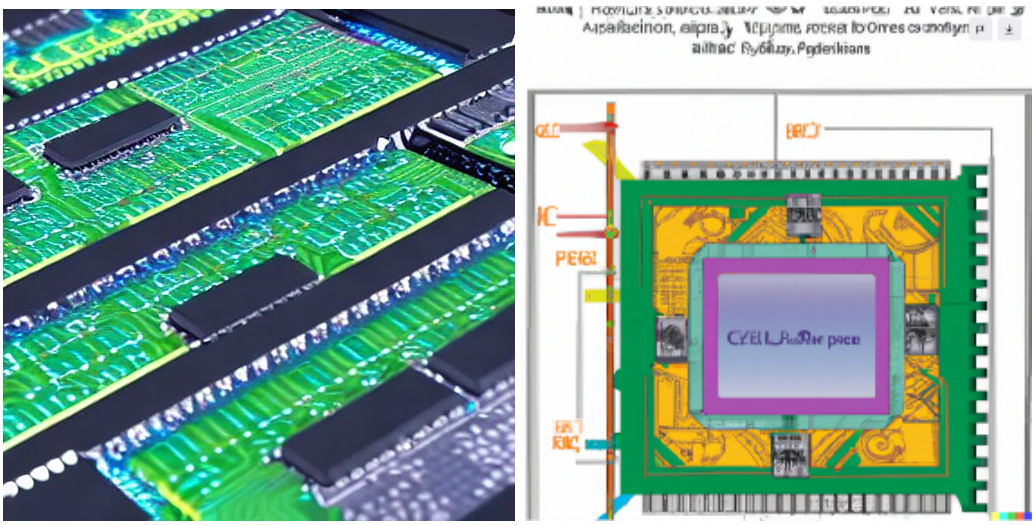


Figura 1.3 Imágenes generada por Inteligencia Artificial. A la izquierda Stable Diffusion y a la derecha DALL·E. Estas *AI* generan imagenes a partir de un texto dado, en este caso se ha usado el título de este trabajo como entrada para cada una de las *AI*.

La tecnología permite que existan *AIs* capaces de reconocer personas en una imagen, generar imágenes a partir de texto, o generar códigos de programación en base a la descripción de las especificaciones. La figura 1.3 han sido obtenidas con dos redes distintas y usando el título de este trabajo como entrada. Sin duda puede hablarse de que todas estas *AI* están causando una revolución tecnológica.

1.1.3 Redes Neuronales Convolucionales

Dentro del *Deep Learning*, una de las primeras arquitecturas que aparecieron fueron las redes neuronales convolucionales o *Convolutional Neural Network* (*CNN*). Este tipo de redes han sido desarrolladas para procesar información con muchas entradas[3], la *CNN* implementa unas técnicas de convolución de imágenes que permite reducir su tamaño y obtener características de la misma, dichas características podrían ser los bordes de los objetos que aparecen en la imagen, determinadas formas geométricas y un sin fin de otras posibilidades. Por tanto, se consigue una reducción del número de conexiones neuronales en las primeras capas y un mayor entendimiento de las entradas.

Además de las *CNN*, existen otras arquitecturas diferentes en el mundo de las redes neuronales, tales como redes neuronales recurrentes, en inglés *Recurrent Neural Network* (RNN)[20, 19], redes generativas adversarias, en inglés *Generative Adversarial Network* (GAN)[18], o las llamadas *Transformer*[52] responsables de muchos de los avances actuales, como pueden ser los modelos de procesamiento de lenguaje natural (NLP) como GPT o LaMDA. La figura 1.3 muestra dos ejemplos de las capacidades de estas tecnologías, que a partir de un simple texto, son capaces de generar imágenes propias. La imagen de la izquierda se obtuvo con Stable Diffusion[42] y muestra algo parecido a una placa de circuito impreso con algunos integrados.

[2] Los parámetros o pesos pueden considerarse equivalentes a las sinapsis neuronales en los sistemas nerviosos de los seres vivos.

[3] Véase una imagen como ejemplo. Una imagen con una resolución FullHD tiene 1920x1080 píxeles y al menos tres capas de colores, para una red neuronal podrían suponer más de 6.000.000 neuronas de entrada.

A su derecha, imagen generada con DALL·E[31], que parece intentar representar un chip con anotaciones ininteligibles.

1.1.4 LeNet-5

En 1998, Yan LeCun y sus colegas plantearon la arquitectura de una *CNN* llamada *LeNet-5*[26]. Esta red está compuesta por siete capas: tres capas convolucionales, dos capas de submuestreo o *Max-Pooling* (*MP*) y dos capas completamente conectadas o *Fully Connected* (*FC*).

Esta arquitectura fue utilizada para detectar números escritos de forma manual. Para ello se utilizó el *dataset Modified National Institute of Standards and Technology* (*MNIST*).

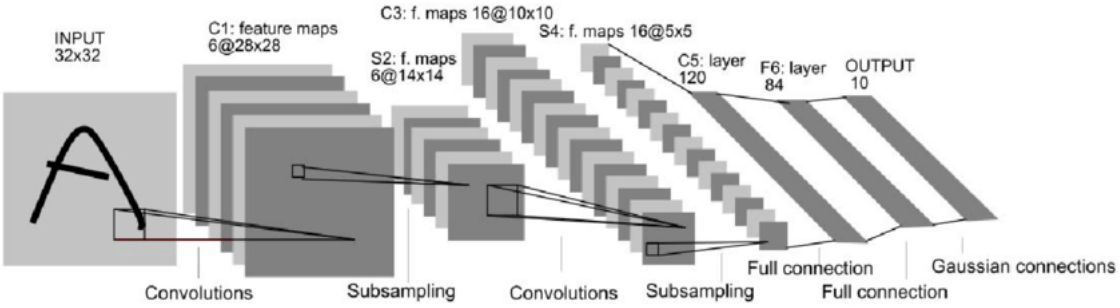


Figura 1.4 Arquitectura de LeNet-5, imagen tomada del artículo original de Yann LeCun et al[26]. La arquitectura está formada por siete capas, tres de ellas convolucionales: C1, C3 y C5, dos de submuestreo: S2 y S4, y dos *FC*: F6 y Output.

1.1.5 Datasets

Una de las partes más importantes del *aprendizaje supervisado* es el entrenamiento. Para ello se optimizan los pesos de forma reiterada, pero para ello se necesita de un conjunto de entradas y salidas de referencia de la que el modelo debe de aprender. Este conjunto es llamado *dataset* y suelen estar formada por una enorme cantidad de entradas, cuánto mayor y más representativo sea el *dataset*, mayor será la capacidad para generalizar.

Además de las entradas de entrenamiento, existe otro subconjunto de datos usados para testear el sistema. Estas entradas son únicas y no están incluidas en las muestras de entrenamiento, pues la red podría ser capaz de adaptarse a los datos de entrenamiento y sin embargo no tener un buen funcionamiento con otros datos no incluidos. A este fenómeno se le conoce como sobreentranamiento u *overfitting*.

Existen muchos *datasets* diferentes, de naturalezas totalmente diversas. Aquí se tratará sólo un par de ellos, ya que fueron utilizados por los colaboradores de este proyecto en el diseño de redes neuronales.

MNIST

Algunas de las muestras contenidas en este *dataset* se han mostrado en la figura 1.5. En cada una de las filas se representan los 10 dígitos usados para representar los números en el sistema occidental, cada columna muestra diferentes escrituras, puede apreciarse que las muestras del *dataset* son bastante diversas. En concreto, esta compilación está formada por 60.000 muestras de entrenamiento y 10.000 de test, cada muestra tiene una resolución de 28x28 píxeles y han sido manuscritas por distintas personas. Este *dataset* es de libre acceso: *http://yann.lecun.com/exdb/mnist/*. En su página se explican más detalles así como resultados obtenidos y publicados con diferentes arquitecturas.

CIFAR-10

Otro *dataset* destacado y mucho más complejo es *CIFAR-10* elaborado por el Canadian Institute For Advanced Research, que puede encontrarse en la dirección: *https://www.cs.toronto.edu/~kriz/cifar.html*. Al igual que en el caso anterior, son conjuntos sencillos de utilizar porque sólo tienen diez clases para clasificar.

El conjunto está formado por 50.000 imágenes de entrenamiento y 10.000 de test. Cada una de las muestras tiene 3 capas de colores y una resolución de 32x32 píxeles. En concreto, las muestran representan seis tipos de animales y 4 tipos de vehículos:

- Airplane: aviones, el dataset incluye desde avionetas a aviones comerciales.

Figura 1.5 Ejemplo de las imágenes contenidas en el *dataset MNIST* para cada uno de los números. Fuente *https://commons.wikimedia.org/wiki/File:MnistExamples.png*.

- Automobile: esta clase está formada por todo tipo de coches de décadas bien distintas.
- Bird: incluye muchas especies de aves, desde un gorrión a un avestruz pasando por gallinas y rapaces.
- Cat: diferentes fotografías de gatos domésticos.
- Deer: contiene diversas imágenes de ciervos.
- Dog: se incluyen una gran variedad de razas de perros, hay que destacar lo diferentes que pueden ser dos perros de razas distintas entre sí.
- Frog: diversa colección de fotografías de ranas de diferentes colores y tamaños.
- Horse: esta clase incluye imágenes de caballos algunos con arreos de montar e incluso jinetes.
- Ship: diferentes tipos de embarcaciones, incluye desde pequeños barcos y lanchas a grandes barcos de carga o embarcaciones militares.
- Truck: camiones de toda índole.

En la figura 1.6 se muestran algunas de las imágenes recogidas en este *dataset*, puede apreciarse que la diversidad de las imágenes es muy alta y mucho más difícil de clasificar que en el caso del *MNIST*. No sólo se trata de imágenes diversas, sino que algunas clases tienen fotografías tomadas desde diferentes ángulos y tienen objetos o seres muy diferentes entre sí. Entrenar un modelo para ser capaz de reconocer estas muestras requiere un mayor número de parámetros que complican y hace que el tamaño de las *CNN* crezcan exponencialmente.

También pueden encontrarse sitios webs que recogen diferentes arquitecturas entrenadas con estos *datasets*. Este es el caso de Kaggle, donde puede verse un *ranking* de los modelos con las mejores puntuaciones para Cifar-10: *https://www.kaggle.com/competitions/cifar-10/leaderboard* o los resultados de *MNIST*: *https://www.kaggle.com/competitions/digit-recognizer/leaderboard* .

1.2 Sobre Hardware para Inteligencia Artificial

Es interesante hacer notar la forma en que todos estos datos son procesados. La mayoría de las redes neuronales suelen funcionar sobre algún tipo de procesador o microcontrolador basado en el modelo de Von Neumann. Es decir, existe una memoria y una unidad aritmético-lógica para hacer todos los cálculos de forma secuencial. Dentro de este paradigma de computación, las operaciones matemáticas suelen realizarse con sumadores de números enteros, esto implica que cada multiplicación de números no enteros requiere un número considerable de operaciones del procesador. Si se observa en (1.1), el cálculo de la salida de cada neurona necesitará un número de multiplicaciones que depende directamente del número de entradas. De

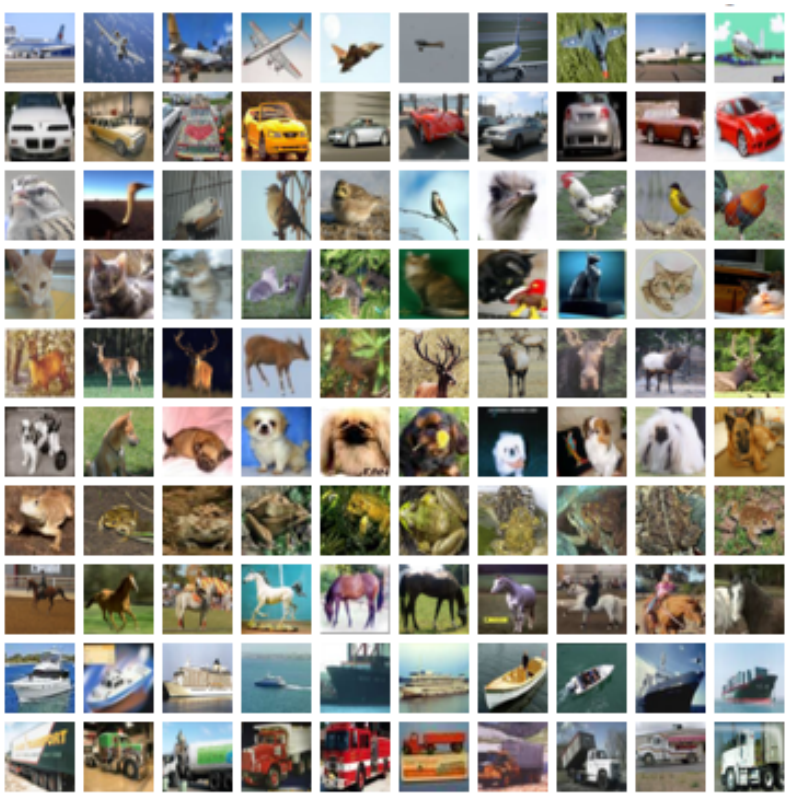


Figura 1.6 Ejemplo de las muestras contenidas en el *dataset* CIFAR-10. Cada una de las filas muestra cada una de las diez clases que configuran el *dataset*. Fuente: *https://www.cs.toronto.edu/~kriz/cifar.html*.

igual manera, las capas convolucionales también requieren de un gran número de multiplicaciones. Por tanto, obtener una sola inferencia de una red neuronal dentro del mundo del *Deep Learning* puede requerir un tiempo considerable, lo que impide que se pueda obtener una tasa de inferencia alta en muchas situaciones[27]. Además, el entrenamiento de una *CNN* no sólo requiere realizar un gran número de inferencias, sino que también se necesitan muchos ciclos de reloj para la optimización de los parámetros.

Todas estas dificultades hicieron en el pasado que esto fuese irrealizable. A día de hoy estas han sido resueltas, sin embargo, a medida que las redes neuronales crecen en complejidad y tamaño, crecen las necesidades de computación. Todas estas vicisitudes han llevado a que se busquen soluciones para reducir este problema. Entre dichas soluciones se encuentra el uso de coprocesadores basados en operaciones con coma flotante, como pueden ser las GPU de los ordenadores o el uso de aceleradores hardware basados en *Field-Programmable Gate Array* (*FPGA*).

Si bien estas soluciones permiten reducir el tiempo de ejecución de cada multiplicación, siguen funcionando de forma secuencial, por lo que el número de parámetros del sistema sigue haciendo que se incremente el tiempo de cálculo de cada inferencia.

Dentro del mundo del diseño de circuitos *Very Large-Scale Integration* (*VLSI*), integración a muy gran escala en español, uno de los principales objetivos es el de reducir el tiempo de computación para el cálculo de esa gran cantidad de parámetros. Para ello se busca una paralelización parcial o completa de las operaciones matemáticas, lo cual implica un aumento considerable del número de transistores del circuito integrado, *Integrated Circuit* (*IC*) en inglés, y la falta de flexibilidad de la arquitectura de la red neuronal.

Como se ha comentado con anterioridad, no existe una arquitectura de red neuronal estandarizada para todos los sistemas, sino que se han presentado muy distintos y diversos tipos de arquitecturas para implementar *ANNs* y *CNNs* las cuales son fáciles de realizar en software, pero necesitan muchas horas de estudio y desarrollo en el terreno del hardware para adaptarse a cada uno de estos diseños.

La falta de uniformidad y normalización es, a día de hoy, uno de los mayores problemas para que la *AI* puede incrementar su capacidad de cómputo en varios órdenes de magnitud. Además, el desarrollo hardware busca también reducir el consumo energético que la *AI* genera a día de hoy. La cantidad de energía consumida y sus efectos en el medio ambiente es notable[44, 43], y todo parece indicar que en los próximos años la cantidad de energía eléctrica destinada a este campo tendrá un aumento considerable.

1.3 Sobre Computación Estocástica

Tal y como se ha visto previamente, la gran cantidad de operaciones más complejas es uno de los grandes desafíos que los diseñadores de hardware deben resolver para conseguir unas implementaciones hardware de arquitecturas de *CNNs* capaces de cambiar el paradigma de computación actual.

Este es el desafío que muchos desarrolladores *VLSI* se han propuesto. La intención es dejar el modelo de von Neumann, basado en la ejecución secuencial de instrucciones y algoritmos, y remplazarlo por un dispositivo al que se le pueda hacer preguntas o dar algún tipo de entrada y recibir una salida acorde a lo demandado, como si de un humano se tratase. Todo ello, además, sin necesidad de programación y algoritmos, y únicamente basándose en el aprendizaje del dispositivo. Tal como lo hubiese hecho una persona.

Una de las aproximaciones planteadas para resolver algunas operaciones matemáticas costosas es la computación estocástica, *Stochastic Computing* (*SC*) en inglés. En los años 50, John von Neumann sienta las bases teóricas de la lógica probabilística[54]. Posteriormente, Poppelbaum y Mars[33, 28] desarrollan la teoría que dará vida al primer computador estocástica, *RASCEL*[14]. Esta técnica tuvo tanto puntos fuertes como débiles. Por un lado, se destaca la facilidad para realizar operaciones matemáticas como multiplicaciones con poco coste y sin necesidad de tanta lógica. Pero entre sus inconvenientes, la naturaleza estadística del método reduce la fiabilidad de las operaciones.

1.3.1 Ejemplo de cómputo

Para explicar el funcionamiento de algunas operaciones, en primer lugar es necesario hablar de cómo se representan los números en *SC*. Estos se representan normalizados en un rango de $[0,1]$ para la versión unipolar y entre $[-1,1]$ para la versión bipolar.

Unipolar

Supóngase que se quiere representar el número $a = 0{,}75$, el número se definiría como una secuencia de n números binarios en serie y cuya probabilidad de estar a valor alto equivaldría a a:

$$P(X = 1) = a = 0.75 \tag{1.3}$$

para el caso de $n = 4$, podría representarse el pulso de la secuencia como $\{1,0,1,1\}$ o cualquiera de las otras combinaciones. Puede verse por tanto que el valor de n define la resolución del número, con cuatro bits se podrían representar los números $0; 0{,}25; 0{,}5; 0{,}75; 1$. El factor estadístico de este método radica en el orden de los bits, si bien se ha indicado cuantas porciones de tiempo la señal está a nivel alto o bajo, no existe un orden exacto, y en concreto, el orden debería ser siempre aleatorio y que no exista correlación entre el orden de los diferentes números con los que se opera.

Supóngase que se definen ahora los números $a = 0{,}75$ y $b = 0{,}375$ con una resolución $n = 8$:

$$P(X_a = 1) = a = 0{,}75 = \frac{6}{8} \tag{1.4}$$

y

$$P(X_b = 1) = b = 0{,}5 = \frac{4}{8} \tag{1.5}$$

por lo que, una de las formas de a podría ser $\{1,1,0,1,1,1,1,0\}$. Igualmente, se podría representar b como $\{0,0,1,1,1,0,0,1\}$. Supóngase ahora que ambas secuencias se colocan en las dos entradas de una puerta lógica *AND*.

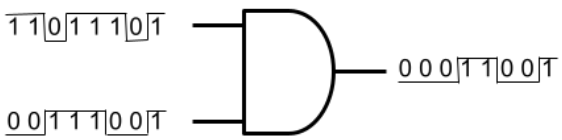


Figura 1.7 Multiplicador de números estocásticos unipolares construido con una puerta AND. El resultado obtenido a la salida es $P(X_o = 1) = P(X_a = 1) \cdot P(X_b = 1)$.

En la figura 1.7 se muestran las dos entradas y el resultado de la operación es la secuencia $\{0,0,0,1,1,0,0,1\}$. Que cumple:

$$P(X_o = 1) = \frac{3}{8} = 0{,}375 \tag{1.6}$$

que coincide con el producto de a por b:

$$a \cdot b = 0{,}5 \cdot 0{,}75 = 0{,}375 = P(X_o = 1) \tag{1.7}$$

Bipolar

Para la versión bipolar, en la que se representan números entre $[-1,1]$, se obtiene:

$$a = 2 \cdot P(X = 1) - 1 \tag{1.8}$$

ya que la probabilidad se define entre 0 y 1, el valor de a sólo podrá variar entre -1 y 1. Al ser el rango de valores el doble que para el caso unipolar, con n bits sólo podrían representarse la mitad de valores. Por ejemplo, con $n = 4$, los valores que podrían representarse son $-1; -0{,}5; 0; 0{,}5; 1$. Supóngase $n = 8$ y dos números $a = -0{,}5$ y $b = 0$.

Para conseguir a y b se necesitará que el primero sea $P(X_a = 1) = 2/8$ y $P(X_b = 1) = 4/8$ para el segundo. Para obtener la multiplicación es posible obtenerla con una puerta *XNOR*. Para ello, se definen las dos secuencias ordenadas de forma aleatoria $\{0,0,1,0,1,0,0,0\}$ y $\{1,0,0,1,1,0,0,1\}$ para a y b, respectivamente.

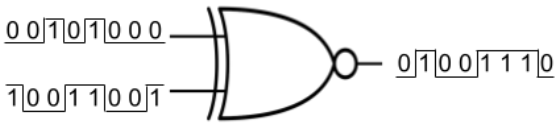


Figura 1.8 Multiplicador de números estocásticos bipolares construido con una puerta XNOR. El resultado obtenido a la salida es $P(X_o = 1) = P(X_a = 1) \cdot P(X_b = 1)$.

En la figura 1.8 se observa la secuencia resultante tras realizar la multiplicación estocástica. Sabiendo que el resultado obtenido se consigue de la expresión 1.8, se tiene un resultado:

$$2 \cdot P(X_o = 1) - 1 = 2 \cdot \frac{4}{8} - 1 = 0 \tag{1.9}$$

de modo que, al multiplicar a y b:

$$a \cdot b = -0{,}5 \cdot 0 = 0 \tag{1.10}$$

efectivamente, se comprueba que las expresiones 1.9 y 1.10 tienen el mismo valor.

Si bien se ha visto que se pueden obtener multiplicaciones de números con codificación unipolar y bipolar con unas simples puertas lógicas, es importante notar que, a veces, el resultado exacto no puede obtenerse con una baja resolución de bits, ya que el número puede no ser representable. A medida que n crece, más números son representables y por tanto menor el error obtenido.

A la hora de implementar estos sistemas de forma práctica aparece el problema de la pseudo-aleatoriedad de las señales. Normalmente suelen usarse *Linear-Feedback Shift Register* (*LFSR*) para generar números aleatorios[17, 50]. Sin embargo, los números conseguidos no pueden considerarse completamente aleatorios, pues suelen generarse a partir de un número previo y acaban incurriendo en una secuencia pseudo-aleatoria que se repite después de un determinado número de muestras. Este tipo de problemas hace que aumente la probabilidad de fallo en las operaciones calculadas, disminuyendo la fiabilidad.

No obstante, la rapidez para realizar multiplicaciones ha suscitado el interés de investigadores para su uso en redes neuronales desde hace ya varias décadas[51, 22, 23, 7].

1.4 Sobre este trabajo

Estas páginas son el fruto del aprendizaje conseguido durante el transcurso del Máster Universitario de Ingeniería de Telecomunicación en la *Universidad de Sevilla* (*US*) y mis comienzos en el *Instituo de Microelectrónica de Sevilla* (*IMSE*). Durante los dos años de duración de estos estudios, se han podido adquirir

conocimientos propios de una maestría, reforzamiento de los conocimientos previos y una ampliación y actualización en otras áreas.

El periodo de realización de unos estudios avanzados suponen también una oportunidad para aprender otras habilidades *blandas*, también conocidas como *soft-skills* para los angloparlantes. Por todo ello, este trabajo no es simplemente el fruto de dos años de cursos universitarios, sino que forma parte de la experiencia humana y la adaptación al entorno, y la iniciación al mundo de la investigación.

Estas páginas son, por tanto, el fruto obtenido a partir de una gran cantidad de vivencias. Por suerte, tampoco son un fin en sí mismas, pues aunque representan el fin de unos estudios, este trabajo también es el principio de los siguientes pasos, en el mundo académico, profesional, y humano.

En este trabajo se unen los caminos de dos campos muy importantes de la ingeniería. Por un lado, la *AI* y el *ML*, que representan uno de los caminos más disruptivos durante los próximos años. Y por otro, la electrónica y los *IC*, un camino que sigue siendo el medio necesario para la evolución de la tecnología. Si bien estos caminos no discurren muy alejados entre sí, cada vez son más las publicaciones que tratan estas temáticas con el objeto de potenciar y aumentar la eficiencia de la *AI*.

La estructura de este Trabajo Fin de Máster (TFM) se apoya en seis capítulos. Este capítulo 1, que introduce unos conocimientos mínimos sobre las materias tratadas, la introducción está enfocada en el contexto global del proyecto. El capítulo 2, donde se expone un resumen del camino que ha de recorrer un microchip desde que se concibe la idea hasta que se obtiene un producto y sirve como guía inicial para las materias que se tratan en este trabajo. El capítulo 3 define de forma concisa las metas que se buscaron para realizar el diseño *back-end*. El capítulo 4 explica cómo se desarrolló el trabajo realizado. El capítulo 5, donde se exponen los resultados obtenidos en el proceso de síntesis. El capítulo 6 cierra el trabajo exponiendo las ideas más interesantes extraídas. Por último, unos apéndices recogen algunos detalles de este trabajo que no tuvieron espacio en los primeros capítulos.

2 Flujo de trabajo en diseño digital de circuitos integrados

- ¿Puedes inventar tu propia cita acerca de "flujo de trabajo para diseño de circuitos integrados"?
*-"El flujo de trabajo en el diseño de circuitos integrados es como navegar en un océano de posibilidades, requiere una brújula precisa y una tripulación experimentada para llegar al destino final de un circuito funcional y eficiente".**

*Cita generada mediante ChatGPT[30]

2.1 Introducción

La tecnología actual está permitiendo la creación de gran cantidad de dispositivos de circuitos integrados para aplicaciónes específicas, *Application-Specific Integrated Circuit* (*ASIC*) en inglés, que inundan el panorama actual. La gran cantidad de diseños desarrollados hoy en día está consiguiendo una reducción de los costes de fabricación, lo que desemboca en un aumento de la demanda, cerrando un bucle en el que este fenómeno se retroalimenta.

Durante los años de crecimiento de la tecnología de dispositivos micro y nanoelectrónicos han ido apareciendo un refinamiento de las técnicas para el desarrollo de los dispositivos *ASIC*. Desde los primeros diseños hasta hoy la cantidad de transistores incluidos en cada dado ha ido aumentando de forma exponencial. La necesidad de automatizar y simplificar los procesos de diseño ha sido fundamental para este desarrollo. En este capítulo se pretende hacer un esbozo del flujo de trabajo, la complejidad y amplitud de conocimientos que existen sobre esta materia podría convertirse en una ardua lectura que se escapa de los objetivos de este trabajo.

Se pueden englobar las diferentes tareas en dos grandes áreas, *front-end* y *back-end*. La primera engloba las primeras etapas hasta la descripción del comportamiento del sistema. La segunda cubre desde el proceso de sintetizar el diseño hasta las últimas fases dónde se fabrica y distribuyen los *IC*.

2.2 Front-end

La aparición de los lenguajes lenguajes de descripción hardware, *Hardware Description Language* (*HDL*) en inglés, obedece a la intención de simplificar la tarea de describir el comportamiento de los sistemas electrónicos. Si bien estos lenguajes permiten describir el comportamiento de los circuitos digitales, también lo hacen con las estructuras de las mismas, por tanto, estos lenguajes aparecen tanto en las etapas de *front-end* como *back-end*. Dos de los más importantes lenguajes son *VHSIC Hardware Description Language* (*VHDL*)[36, 10] y *Verilog* (*Verilog*)[35], aunque no son las únicas, sí son unas de las primeras herramientas de los desarrolladores.

2.2.1 Especificaciones del sistema

El punto de partida de cualquier diseño radica en conocer las especificaciones y requisitos del mismo. Es una tarea fundamental, pues un fallo en los requisitos podría dar lugar a un *IC* inútil.

Un diseñador necesitará conocer las entradas y salidas del sistema, y el funcionamiento del mismo. Sin embargo, puede que existan unos requisitos más específicos de las entradas y salidas. Es necesario tener todos estos factores en cuenta, pues el diseño hardware obtiene un producto rígido y costoso, por lo que los errores pueden ser inasumibles.

2.2.2 Diseño arquitectónico

Los sistemas complejos pueden requerir de una gran cantidad de detalles que pueden describirse en subsistemas llamados módulos. Dividir la tarea del sistema en módulos ayuda al desarrollo del sistema.

Esta fase del flujo se basa en subdividir el sistema en los diferentes módulos y estudiar la compatibilidad y conectividad de los mismos entre sí. Cada *módulo* contará con una serie de entradas y salidas, las cuales podrán ser salidas o entradas de otro módulo o del conjunto del sistema. Se necesita prestar atención a las interfaces para asegurar la compatibilidad de los módulos, así como cerciorarse de que el conjunto cumple con las especificaciones de partida.

2.2.3 Diseño lógico

El diseño lógico consiste en describir el comportamiento dentro de cada módulo. Si bien los lenguajes *HDL* lucen similares a lenguajes de programación para software clásicos, su uso es mucho más restrictivo. No todo lo que se puede programar en un procesador corriente es implementable a nivel hardware, por lo que las restricciones a la hora de desarrollar códigos hardware son muy altas.

Para ayudar en estas tareas existen simuladores y otras herramientas que ayudan a conocer cómo será el comportamiento de lo descrito en el código. En general, el tiempo de cómputo en algunas simulaciones puede ser muy pesado ya que se calculan una gran cantidad de datos y transiciones que ocurren en tiempos de pico o nanosegundos. Los simuladores permiten trabajar a varios niveles, el nivel más alto es el del comportamiento funcional. En este caso el simulador no tiene en cuenta los retrasos que se producen por elementos lógicos del mismo. A medida que se baja de nivel se realizan simulaciones con las características de las puertas lógicas o incluso a nivel de transistores, no obstante, los niveles bajos suelen realizarse en fases de *back-end*, mientras que los funcionales son tareas de *front-end*.

2.3 Back-end

Una vez que ha sido definido el comportamiento y la relación de los módulos de un diseño, debe trasladarse todo esto al dispositivo físico. Después de que haya sido testados los diseños *front-end* en simulación o con plataformas FPGA para conocer como será el comportamiento del diseño, es el turno para el diseñador de *back-end* de convertir el código en algún formato conocido por los fabricantes[46]. Esta parte del diseño es de especial importancia. Un diseño incorrecto podría dar lugar a un dispositivo totalmente inútil. No obstante, existen métodos y pruebas para poder comprobar e intentar predecir el correcto comportamiento del sistema.

Uno de los detalles importantes que se necesitan para poder desarrollar estas tareas y conocer el resultado, es la elección del fabricante y conocer sus reglas. En la descripción del comportamiento, las variables de las tecnologías carecen de importancia, sin embargo, en esta etapa del proceso sí es necesario conocer los detalles de los dispositivos lógicos y los transistores que el fabricante obtiene de sus tecnologías.

En la figura 2.1 se muestra una vista tridimensional de una parte de un circuito realizado con *Innovus Implementation System*[48].

2.3.1 Síntesis

La *síntesis* es un proceso que convierte un código *HDL* de tipo comportamental a otro de tipo estructural. El código comportamental describe lo que el diseñador espera del circuito, mientras que el código estructural describe la relación entre dispositivos lógicos. A estos últimos códigos se les denominan *Register-Transfer Level* (*RTL*), nivel de transferencia de registros.

Para generar la síntesis los fabricantes ofrecen sus *Process Design Kit* (*PDK*). Estos *PDK* consisten en unas librerías de archivos, preparados para distintas herramientas, en el que se detallan gran parte de las especificaciones de los dispositivos que se fabricarán, incluyen distintos tipos de dispositivos lógicos, como

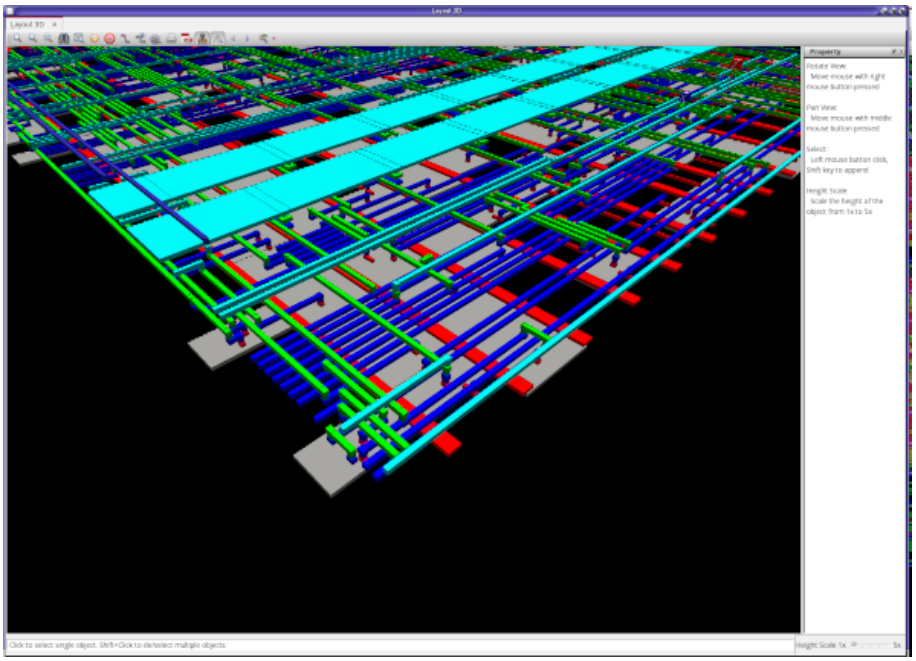


Figura 2.1 Detalle en 3D de un circuito integrado con Innovus Implementation System. Las capas de metal se extienden de forma perpendicular entre sí para permitir el enrutado de los dispositivos. En gris se muestra el área ocupada en el diseño por cada una de las celdas. En rojo se muestra la primera capa de metal, usada para las conexiones directas con los terminales de las celdas y para los carriles de alimentación. En azul oscuro la segunda capa de metal que se extiende de forma perpendicular a la primera capa de metal. En verde se muestra la tercera capa de metal, también perpendicular a la anterior. En cian la cuarta capa, en esta capa puede verse dos grandes carriles usados para llevar la alimentación y la tierra a lo largo de todo el circuito. Por último, también en azul oscuro se puede ver la quinta capa de metal..

pueden ser puertas *AND*, *OR*, *biestables*, *pads*, etc. Además de estos dispositivos, se indican datos como los retrasos de cada puerta, para distintos casos, por ejemplo, tiempos mínimos, máximos o medios. Para poder hacer pruebas de alto nivel y conocer si una puerta es capaz de activar a otras varias, también conocido *fan-out*, también se incluyen datos de capacitancia, resistividad, etc. En definitiva, los *PDK* suelen incluir una gran colección de información, normalmente de carácter confidencial, para realizar tareas de síntesis, cálculos y verificaciones.

Como se ha comentado con anterioridad, la función de la *síntesis* es devolver un fichero *HDL* en el cual se convierte todo el diseño a nivel de puertas lógicas o transistores. Para ello existen herramientas software que se encargan de esta tarea partiendo de la base de datos que el fabricante ofrece.

Aunque las herramientas podrían pasar todo el diseño directamente a nivel de transistores, las *foundries* (fabricantes de IC) ofrecen librerías en las que ellos mismos han optimizado el diseño y cuyos datos de la caracterización de los dispositivos lógicos es facilitada por dichos fabricantes, permitiendo un comportamiento del mismo más predecible. Se ofrecen también una cantidad de módulos propietarios llamados *Intellectual Property* (*IP*) de diversa naturaleza como podrían ser memorias RAM, microcontroladores o microprocesadores, dispositivos de comunicación y un largo etcétera. Los detalles de la implementación de los módulos *IP* no son proporcionados por los fabricantes para evitar usos fraudulentos y proteger sus intereses.

2.3.2 Diseño físico

La realización del diseño físico puede separarse en varios pasos:

Floorplanning

El primero de los pasos consiste en definir el tamaño del dado[1] y los *pads* del diseño, este proceso recibe el nombre de *floorplanning*.

Gracias al proceso de *síntesis* puede calcularse una aproximación del área mínima necesaria para alojar el diseño. Sabiendo ese dato, puede establecerse el tamaño del dado y repartir las entradas sobre la superficie.

La figura 2.2 muestra un ejemplo de *floorplanning* en *Innovus Implementation System*. En el área circunscrita por el cuadrado interior se alojan las celdas lógicas y las conexiones. A su alrededor, los diferentes *pads* rodean el circuito y sirven como base de las futuras conexiones con el encapsulado. Las esquinas también son reconocidas como *pads* y pueden ser usadas. En ocasiones, en el área ocupada por los *pads* se coloca algo de electrónica que puede servir para adaptar las tensiones de entradas y salidas del *IC* con la circuitería interna.

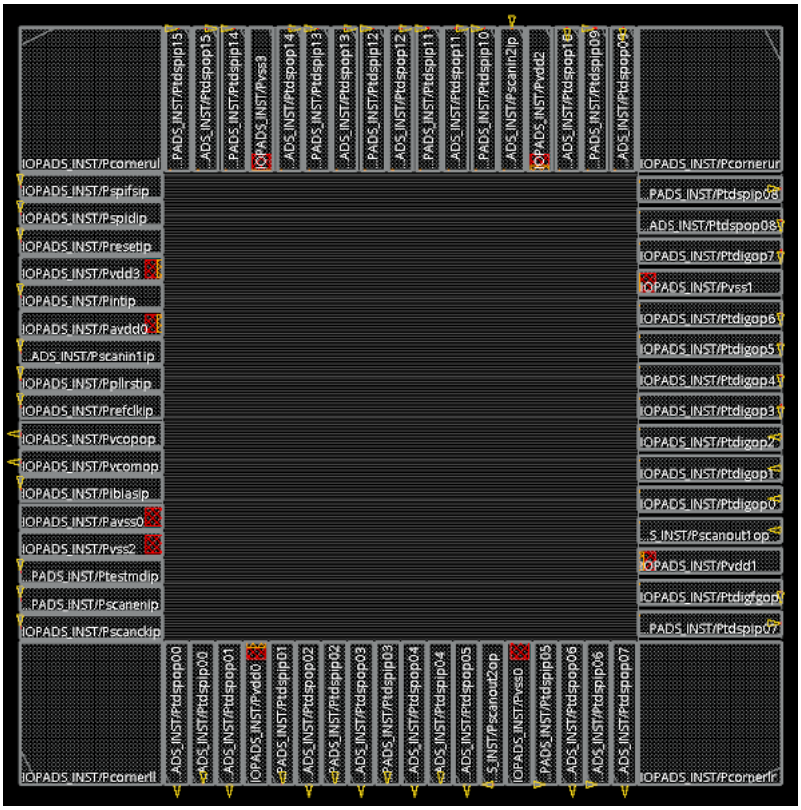


Figura 2.2 Ejemplo de floorplanning realizado durante ejercicios de entrenamiento con Innovus Implementation System. La electrónica del diseño debería ocupar el cuadrado central, mientras que los los rectángulos que lo rodean representan a los distintos *pads* con las entradas y salidas del circuito. Estas terminaciones también pueden ser propiedad intelectual del fabricante e incorporar electrónica para evitar sobretensiones o cambio de voltaje para la adaptación entre los níveles usados fuera del chip, 3,3V y el voltaje interno que puede ser inferior a 2V dependiendo de la tecnología..

Place and Route

El proceso de *place and route* (emplazar y enrutar) consiste en definir la colocación de los dispositivos y transistores así como las capas de metal que conectarán, a nivel de conductividad, el circuito. Aunque en diseño de circuitos analógicos predomina la creación de *layout* de forma manual, en digital está más extendido el uso de herramientas que automatizan este proceso.

[1] Dado: es la unidad mínima de fabricación que aloja un diseño completo. Para la fabricación de circuitos integrados se usan unos discos llamados obleas, normalmente de silicio, sobre la que se ejecutan algunos procesos y se añaden capas de diversos materiales, en dicha oblea suelen ponerse varios circuitos completos, estas subdivisiones son los llamados dados. Para más información se invita al lector a buscar más sobre el proceso de fabricación CMOS[56, 24, 49, 8].

En circuitos complejos y con grandes buses de datos, este proceso se vuelve de especial importancia ya que puede ocurrir que el enrutado consuma una gran cantidad de área del dado. Por tanto, se presta una gran atención al *place and route* y al *floorplanning* para evitar distancias innecesarias

Como se ha comentado anteriormente, en diseños analógicos se opta por el desarrollo de *layout* manuales. Sin embargo, en las tecnologías digitales *Complementary Metal-Oxide-Semiconductor* (*CMOS*), se aprovecha la característica de que la mayoría de los dispositivos cuentan con transistores de canal P y N, creando filas de celdas con altura fija para todos los dispositivos, en la parte superior e inferior se colocan vías a Vdd y Vss, de esta forma, y teniendo en cuenta que los tamaños de los transistores suelen ser lo más reducidos en área posible, se hace más fácil ajustar en cada línea una cantidad de dispositivos sin desperdiciar superficie útil de la oblea[2].

Un ejemplo de lo explicado en esta sección, hecho con *Innovus Implementation System*, se muestra en la figura 2.3. Si se observa con detenimiento, en especial en la parte superior derecha, en azul se muestran las vías para alimentar las celdas[3].

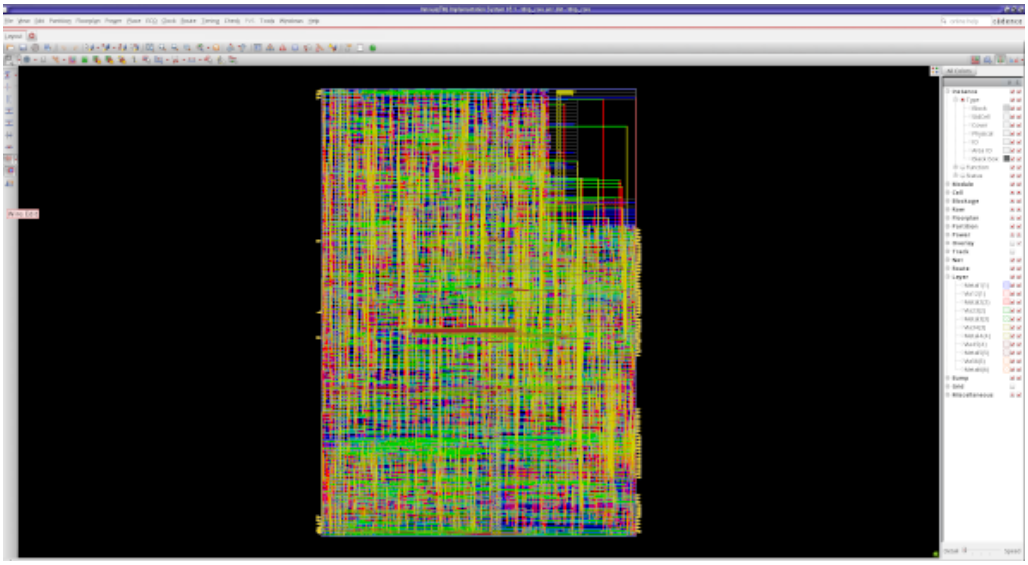


Figura 2.3 Ejemplo sencillo de un diseño después del Place and Route realizando durante un curso de entrenamiento. Puede observarse que los elementos son bastante pequeños y el número de interconexiones elevado. Gracias a la ayuda de las herramientas, esta gran tarea se encuentra automatizada, ahorrando horas de trabajo en el diseño y en la búsqueda de errores.

2.3.3 Verificación física y Aprobación

Una vez se ha finalizado el diseño físico, es necesario realizar estudios que prevean cómo será el comportamiento del *IC* diseñado. Para ello, se hacen análisis con las capacidades parásitas y los retrasos que se producen en cada etapa de transistores que la señal debe atravesar.

Los circuitos digitales acostumbran a llevar una gran cantidad de lógica secuencial, por lo que es necesario un estudio minucioso del tiempo que tardará una señal en cada etapa del circuito. Sin embargo, el diseño de micro y nanoelectrónica conlleva una gran variabilidad de algunos parámetros de los transistores.

Esta variación se debe al *mismatching* que se produce por las imperfecciones de los procesos de fabricación. Los *PDK* incorporan datos sobre el funcionamiento de los dispositivos en diversas condiciones y a distintas temperaturas. Gracias a dicha información, pueden realizarse *análisis de esquina* y *análisis de Montecarlo*, lo que permite obtener una aproximación del comportamiento del circuito en las condiciones típicas, mejores o peores.

[2] Por no ser objeto de este trabajo y evitar exceso de información irrelevante, se remite a los lectores interesados a que busquen más detalles sobre la optimización del área mediante la literatura, cursos de los desarrolladores de las herramientas u otros materiales de divulgación[8, 32, 34, 38, 40, 49, 47, 45, 48]

[3] Para más información de los pasos más básicos, se deja una referencia a un sencillo tutorial[12], además de los cursos que *Cadence Design Systems* ofrece [48].

Una vez conocidas las irregularidades del diseño, pueden calcularse las frecuencias máximas a las que la lógica secuencial puede funcionar sin esperar ningún problema. A veces es necesario incluir retrasos en algunos relojes y crear un árbol de relojes para evitar problemas de *fan-out* en dichos relojes.

Con toda esta información, es posible, hacer análisis estáticos de tiempo o dinámico, si se proporciona unas entradas simuladas.

Otras pruebas interesantes pueden ser *Layout vs Schematic* (*LVS*), *Design Rule Checking* (*DRC*), geometrías, antena[4]. El uso de estas herramientas permite testear el funcionamiento del sistema físico y su similitud con el diseño comportamental esperado.

2.3.4 Preparación de Máscaras

Si bien el paso anterior suele revisarse tanto por los diseñadores del circuito como por el fabricante, la preparación de máscaras y fabricación de las obleas, suele ser responsabilidad de los fabricantes.

Para la fabricación es necesario crear las máscaras para cada proceso al que se someterá la oblea. Existe mucha literatura que explica el proceso de fabricación *CMOS* donde se hablan de las diferentes máscaras y procesos. Si bien se conocen a grandes rasgos detalles de éstos, mucha de la información es celosamente guardada por las *foundries*. Puede encontrarse más detalles sobre este tema en libros especializados[49, 8, 32].

La preparación de las máscaras y la fabricación de dispositivos nanométricos, ha de hacerse en entornos limpios y controlados, ya que cualquier pequeña partícula no deseada podría ocasionar grandes errores de fabricación. En la figura 2.4 se muestra un ejemplo de cómo son las salas blancas donde se realizan estas tareas y el tipo de protecciones usuales.

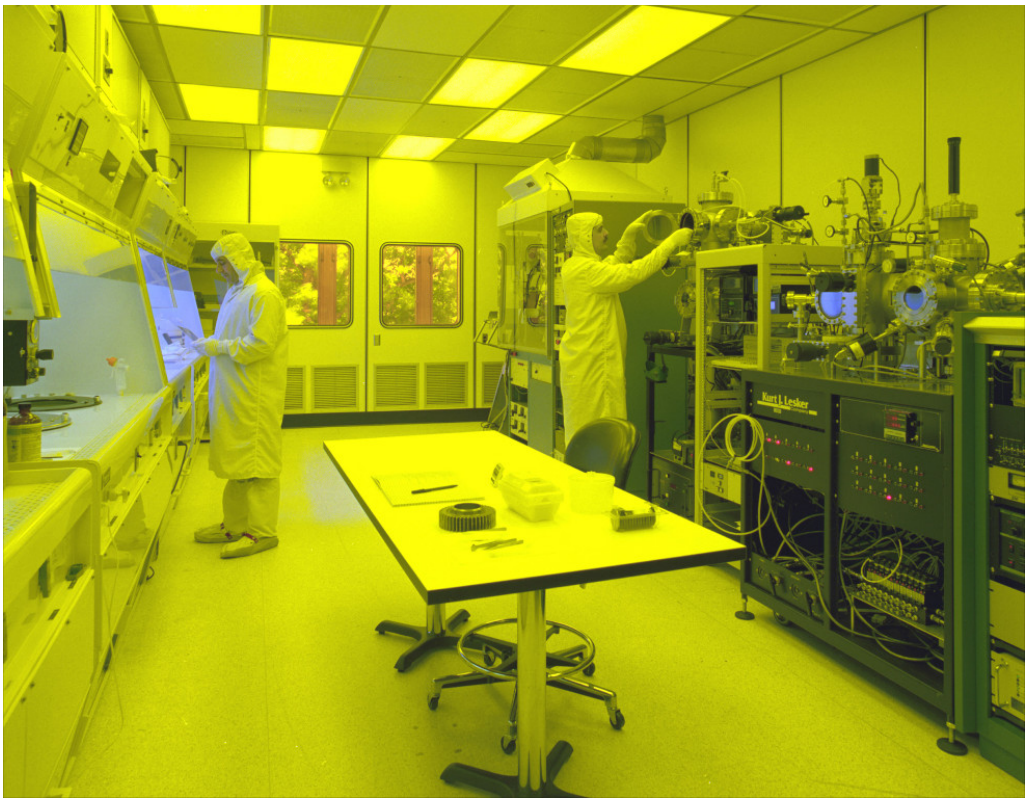


Figura 2.4 Sala blanca de la NASA donde se fabrican *IC*. Imágenes obtenidas de WikiMedia Commons bajo licencia Creative Commons. Fuente:*https://commons.wikimedia.org/wiki/File:Clean_room.jpg*.

2.3.5 Fabricación de obleas

El proceso de fabricación parte de un oblea de silicio que se va sometiendo a diferentes procedimientos para definir las regiones de dopados en la oblea, añadir la capa de aislante de los transistores y las diferentes capas

[4] Los problemas de antena que se pueden originar dentro del chip no son los que se acostumbran de transmisión de ondas de radio. Este problema deriva de la posibilidad de que la puerta de un transistor pueda ser activada por la carga de una conexión cercana al transistor, por lo que es necesario tener en cuenta la distancia a la que pasan las vías del diseño.

de metales que interconectarán los transistores. Existen diferentes técnicas para algunos de los pasos y van evolucionando con el tiempo para obtener dispositivos mejores y más eficientes que, sin embargo, se escapan del alcance de este trabajo. Al igual que en el apartado anterior, se invita al lector interesado a consultar información sobre el proceso de fabricación *CMOS* para conocer más sobre este paso[49, 8, 32].

En la figura 2.5 se muestran dos imágenes con el resultado del proceso de fabricación de obleas.

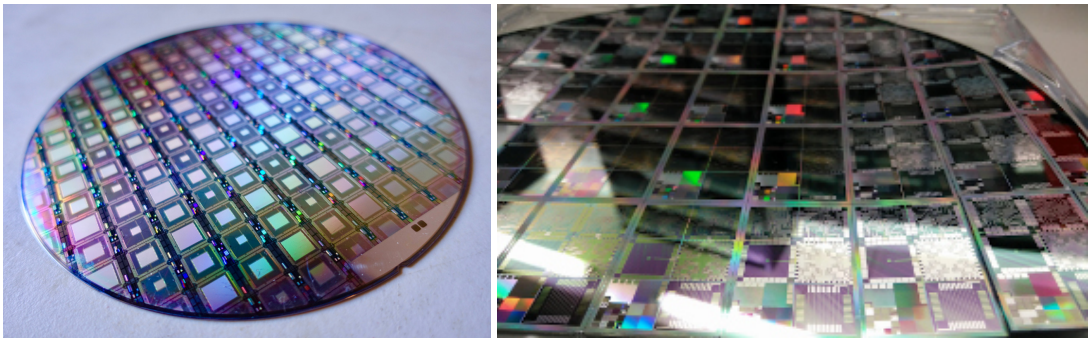


Figura 2.5 Diferentes obleas con circuitos después del proceso de fabricación. Imágenes obtenidas de WikiMedia Commons bajo licencia Creative Commons. Fuente:*https://commons.wikimedia.org/wiki/File:A_Wafer_of_the_Latest_D-Wave_Quantum_Computers_(39188583425).jpg* y *https://commons.wikimedia.org/wiki/File:Semiconductor_Wafer_of_Microelectronics.jpg*.

2.3.6 Empaquetado

Una vez obtenidas las obleas, se cortan y dividen en dados. Sin embargo, estos *dados* son pequeños pedazos de silicio que pueden medir micrómetros en ambos lados. Manejar estos tamaños para la fabricación de *Printed Circuit Board* (*PCB*) haría el proceso de colocación y soldadura una tarea difícil. Por ello y para, además, poder proteger el chip, se empaquetan en diferentes soportes que podrán ser trabajados en otros ambientes sin depender de maquinaria tan precisa.

El empaquetado también puede permitir que en chips como procesadores de alta velocidad, se pueda disipar el calor generado por el funcionamiento del chip.

Para unir los pines o *pads* del empaquetado con los *pads* del dado, se usan máquinas de *bonding* que generan microconexiones de alta precisión.

2.3.7 Test y Documentación

Una vez se tienen los chips ya fabricados y empaquetados, es necesario probar que el comportamiento del diseño responde a las especificaciones esperadas durante el diseño.

Es importante caracterizar el circuito para conocer su comportamiento real y generar la documentación necesaria con la información relativa al circuito que necesitarán conocer todos aquellos que tengan que trabajar posteriormente con el chip.

2.4 Procesos de fabricación de semiconductores

La fabricación de *IC* ha ido variando a lo largo de los años. Las técnicas se han ido refinando y mejorando, se han buscado automatizar tareas del diseño para mejorar las prestaciones de los circuitos. Así pues, se ha conseguido minimizar el tamaño de los transistores, a medida que se disminuyen la longitud del canal y la capa de óxido que separa la puerta del canal, se obtienen una serie de ventajas: al disminuir el tamaño de los transistores, es posible integrar una mayor cantidad de componentes un solo chip; mejora la velocidad de conmutación, los electrones tienen un menor recorrido; al ser el canal más corto, también disminuye la resistencia interna del canal, por lo que la energía consumida disminuye.

Obviamente, cuando se optimizan el tamaño de los transistores también aparecen desventajas: la ganancia del transistor disminuye, por lo que no son capaces de manejar señales grandes y la tensión de rotura del óxido de la puerta disminuye, haciendo que los dispositivos sean menos robustos; la variabilidad con la temperatura aumenta, lo cual hace más difícil diseñar y simular los diseños. Por tanto, los cambios en los procesos de fabricación han ido escalando poco a poco, a medida que se han ido resolviendo conflictos

y problemas. En las tecnologías avanzadas, a partir de 22 nm, se ha empezado a cambiar los transistores clásicos por dispositivos FinFET. Estos cambios han sido más sencillos de adaptar en electrónica digital, pero en el mundo de las señales analógicas, algunos de estos problemas hace muy difícil seguir reduciendo los tamaños[34, 49, 8, 32, 40, 9, 58]. En la tabla 2.1 se muestra la evolución de los nodos de fabricación desde 1972 a 2022 y la previsión para 2024[57].

Tabla 2.1 Nodos de fabricación de semiconductores desde 1971 a 2022 y previsión para el año 2024. Fuente: Wikipedia[57].

10.000 nm	1971
3.000 nm	1975
1.500 nm	1982
1.000 nm	1985
800 nm	1989
600 nm	1994
350 nm	1995
250 nm	1997
180 nm	1999
130 nm	2002
90 nm	2004
65 nm	2006
45 nm	2008
32 nm	2010
22 nm	2012
14 nm	2014
10 nm	2016
7 nm	2018
5 nm	2020
3 nm	2022
2 nm	2024

El flujo de trabajo descrito para el diseño de circuitos micro y nanoelectrónicos no es más que un simple esbozo de lo que realmente conlleva. En él intervienen ingenieros, físicos y técnicos con mucha experiencia y conocimiento en sus respectivos campos para optimizar estos procesos. Por lo que, aunque no represente una guía clara de cómo se diseña un *IC*, sí puede suponer unos conocimientos básicos para asomarse a la comprensión del flujo de diseño.

3 Objetivos

El mayor peligro para la mayoría de nosotros no radica en establecer unos objetivos demasiado altos y fracasar pronto, sino en establecer unos objetivos demasiado bajos, y lograrlos.

MIGUEL ÁNGEL

El trabajo que se refleja en estas páginas, ha sido realizado en el *IMSE*, centro mixto perteneciente al Centro Nacional de Microelectrónica (CNM) del *Consejo Superior de Investigaciones Científicas* (*CSIC*) y la *US*, y nace de la colaboración entre el *IMSE* y la Universitat de les Illes Balears (UIB).

3.1 Introducción

Como se comentó en el prólogo, estas páginas recogen información de cómo se llevó el proceso de *síntesis* para los diseños previamente realizados por la UIB. Por tanto, se reflejan en este capítulo los objetivos que fueron planteados para el proceso de *back-end*, teniendo en cuenta que el objetivo de estos circuitos era hacer una prueba de conceptos de redes neuronales basadas en *CNN* haciendo uso de *SC* para conseguir un sistema completamente paralelo[1].

Los objetivos iniciales del proyecto se encuentran fuera del ámbito de este trabajo, por lo que en este capítulo solamente se hablarán de los objetivos fijados para el diseño *back-end* y más concretamente para el proceso de *síntesis*. Para poder conocer la escalabilidad de este tipo de sistemas, se propone obtener información de cómo podría ser el tamaño y las prestaciones de los *IC*. Se establece el uso de una tecnología de 40 nm de TSMC.

Sobre el diseño *front-end*, puede encontrarse más información en la publicación de *Frasser et al.* publicada en Transactions on Neural Network and Learning Systems[16].

3.2 Objetivos

Dentro de este contexto y para este trabajo se establecieron los siguientes objetivos:

1. Hacer uso del *PDK* de 40 nm de TSMC para todos los diseños.
2. Realizar el proceso de *síntesis* con el diseño *front-end* de una *CNN* que implemente una arquitectura básica de 64 neuronas dada por la UIB.
3. Realizar el proceso de *síntesis* con el diseño *front-end* de una *CNN* que implemente la arquitectura de LeNet-5 dada por la UIB, con resolución de 2 y 4 bits.
4. Realizar el proceso de *síntesis* con el diseño *front-end* de una *CNN* que implemente la arquitectura de Cifar-10 dada por la UIB. Este diseño se hará subdividiendo el diseño:
    - 6 capas: 4 capas convolucionales y 2 *FC* con resolución de 4 bits.

[1] Un sistema completamente paralelo hace referencia en este contexto a la capacidad de obtener la multiplicación de los pesos y las entradas de las neuronas de forma simultánea.

- 6 capas: 4 capas convolucionales y 2 *FC* con resolución de 8 bits.

5. Realizar el proceso de *síntesis* con el diseño *front-end* de una *CNN* que implemente la arquitectura de Cifar-10 dada por la UIB de forma completa.
6. Hacer uso de un reloj de referencia de 6 ns. Podrá modificarse si el diseño no fuese realizable debido a este punto.
7. Hacer uso de las herramientas de *Cadence Design Systems* (*CDS*), compatible con la tecnología.
8. Obtener una estimación de la potencia consumida para cada uno de los diseños.
9. Obtener el número de celdas y el área de silicio consumida para cada diseño.
10. Obtener una estimación de los retrasos máximos en el camino más largo.

Al tratarse de un diseño hecho como prueba de concepto, y teniendo en cuenta el enorme tiempo de cómputo para algunas de las pruebas, los tiempos máximos y las potencias consumidas se harán para valores típicos y no para esquinas[2].

El diseño del *layout* y los consiguientes pasos de la fabricación, explicados en el capítulo 2, no han sido finalizados aún y exceden las posibilidades de este trabajo, y por tanto, no forman parte del mismo.

[2] Normalmente las *foundries* ofrecen unos valores típicos para los parámetros de su tecnología, los más esperables, así como los valores mínimos y máximos para los mismos parámetros que pueden producirse por las imperfecciones y no linealidades de los procesos de fabricación. En las fases de verificación de diseño de circuitos integrados es frecuente hacer análisis de esquinas para conocer la variabilidad que puede sufrir el diseño así cómo sus puntos débiles.

4 Desarrollo

El desarrollo del hombre depende fundamentalmente de la invención. Es el producto más importante de su cerebro creativo. Su objetivo final es el dominio completo de la mente sobre el mundo material y el aprovechamiento de las fuerzas de la naturaleza a favor de las necesidades humanas.

NIKOLA TESLA

Para cumplir con los objetivos establecidos en el capítulo 3 de este trabajo se ha de hacer uso de las herramientas de *CDS* disponibles en el centro de computación del *IMSE*.

Dentro de la gran cantidad de herramientas que proporciona esta empresa, *Genus Synthesis Solution* ofrece la posibilidad de generar el código *RTL* de forma potente. Como se vio en el capítulo 2, el objetivo de la *síntesis* es convertir un diseño de alto nivel a bajo nivel, mientras que los diseños de bucles de alto nivel usan operadores, variables y estructuras condicionales, el código que genera la *síntesis* es más similar a un esquemático, pues lo que contiene es una descripción de celdas básicas e *IP*, y las conexiones que los relaciona. Entre las principales características de esta herramienta destacan:

- Incrementa hasta diez veces la productividad durante el diseño *RTL*.
- Tiempo de cálculo cinco veces más rápido que otros softwares para sistemas con más de diez millones de instancias.
- Reducción de al menos dos veces el tiempo entre iteraciones.
- El cálculo de las estimaciones de tiempos de retrasos y de la longitud de los cables están dentro del 5 % de lo obtenido con *Cadence Innovus Implementation System*.
- Reducción del área consumida por las conexiones hasta un 20 % sin perjuicio para el rendimiento.

Además de lo mencionado, esta solución es valida tanto para códigos en *VHDL*, *Verilog* o *System Verilog*; funciona de forma paralela en sistemas con múltiples CPUs y máquinas; capacidad de crear y ajustar jerarquía de registros de relojes *RTL*; etc.

4.1 Introducción

Es algo común en la ingeniería que la obtención de la solución requiera una inversión mayor de tiempo para el aprendizaje de la técnica y estudio del problema que para la resolución del mismo. El proceso de síntesis es similar.

Para la realización de este proyecto se ha invertido alrededor del 30 % del tiempo en el aprendizaje y dominio de la técnica y el análisis de los códigos diseñados en la parte *front-end*; un 15 % en la elaboración de los códigos a ejecutar; un 40 % en la ejecución de la síntesis y otro 15 % del tiempo en el análisis, reajustes y correcciones de los problemas que fueron ocurriendo. Y para el caso que ocupa este proyecto, el total fue de alrededor de 45 días de trabajo o 360 horas.

Es interesante notar que el trabajo que se realiza para la síntesis de *IC* es similar al trabajo para crear una *CNN*. Ambas son tareas con un alto coste de computación que ahorra al ingeniero gran parte del trabajo y

que requiere, por tanto, de una buena preparación para evitar errores y repetir largos tiempos de espera de computación.

4.2 Flujo básico de síntesis

La mayor parte del trabajo de síntesis se realiza mediante programación con lenguaje de *scripting TCL*. La tarea más importante que realiza el software es convertir un código escrito de manera funcional a uno estructural, por lo que lo primero que necesitará es el código donde se describe el comportamiento del sistema y las librerías con los elementos disponibles en la tecnología[1]. A continuación se detallan los pasos básicos a realizarse en el flujo de síntesis:

1. Seleccionar las librerías de la tecnología y ajustar las condiciones iniciales.
2. Lectura de los archivos *HDL*.
3. Elaboración del diseño.
4. Inicializar el diseño.
5. Ajustar las restricciones de tiempo y diseño.
6. Aplicar directivas de optimización.
7. Ejecución del proceso de síntesis.
8. Obtener reporte y analizar.
9. Generar archivos de salida.
10. Evaluar si el diseño cumple con las especificaciones, en caso contrario volver al paso 6.

4.3 Ejecución

Genus Synthesis Solution, en principio, se presenta como un programa en línea de comando, si bien es posible obtener una interfaz gráfica(GUI) para algunas de las tareas que se pueden realizar desde el software, lo normal es empezar sin dicha interfaz.

Los comandos del flujo mencionados anteriormente pueden ejecutarse de uno en uno desde el terminal de *Genus* o directamente desde un fichero *TCL* que reúna todos los comandos necesarios. Este último método es más interesante pues permite al diseñador lanzar el conjunto de los comandos sin tener que estar pendiente de que terminen cada uno de los pasos, ya que alguno de ellos puede durar un tiempo considerable.

4.3.1 Estructura de carpetas

Debido al alto número de ficheros de entrada y salida, es más que recomendable separarlos por secciones. A continuación se explica la separación sugerida por *CDS*:

- QRC/ Contiene los archivos QRC de la tecnología.
- constraints/ Contiene los archivos de requisitos (SDC).
- def/ Contiene archivos para el floorplan (DEF).
- libraries/ Contiene las librerías de la tecnología.
- rtl/ Contiene los códigos de descripción funcional.
- sim/ Contiene archivos de simulación (TCF).
- tcl/ Contiene los *scripts*.
- work/ Directorio donde se ejecuta el código y se reúnen las salidas.

[1] Los *PDK* de los fabricantes ofrecen librerías con una gran cantidad de elementos como puertas OR, AND, NAND, Biestables, etc. para, por un lado, evitar que el software tenga que hacer una síntesis al más bajo nivel, y además conseguir mejores resultados de optimización ya que estos elementos han sido ya caracterizados por el fabricante. Por ejemplo, si el diseño implementa unas puertas NAND, no será necesario calcular los retrasos o las capacidades de dicho elemento, sino que el *PDK* ya ofrece los datos de dicha puerta NAND.

4.3.2 Cargando librerías y diseños

El primer paso del *script* desarrollado para este proyecto es la definición de las carpetas para obtener los archivos necesarios, selección del lenguaje usado, en este caso *VHDL*, elegir el módulo principal o *Top* de la jerarquía.

Además, deben describirse los ajustes que se quieran hacer, en este caso, se han ignorado algunos mensajes de advertencia que no son muy relevantes y otros ajustes recomendados por el desarrollador del software.

4.3.3 Lectura de los archivos HDL

Una vez establecido los directorios, los ajustes y conocidos los archivos que se quieren utilizar, declarados en algunas variables ($), se da la orden de que el software los cargue en memoria, primero las librerías de la tecnología, librerías físicas y por último, los códigos diseñados en la fase de *front-end*. Esto se muestra en el código 4.1.

Código 4.1 Ejemplo de instrucciones de lectura de librerías y códigos fuentes del diseño.

```
#set_db library $LIB_LIST
read_libs $LIB_LIST
## PLE
#set_db lef_library $LEF_LIST
read_physical -lef $LEF_LIST

# set_db cap_table_file $CAP_TABLE_FILE
# set_db qrc_tech_file $QRC_TECH_FILE

read_hdl $RTL_LIST
```

4.3.4 Elaboración

La elaboración tiene como objetivos construir la estructura de datos e inferir los registros en el diseño. Realiza una optimización de alto nivel para eliminar bloques muertos que no son necesarios e identifica los relojes candidatos para crear la jerarquía.

Si el modulo Top tiene algún parámetro, debe pasarse durante este comando para que la síntesis sea correcta. El código 4.2 muestra el comando usado.

Código 4.2 Instrucción básica para la ejecución del proceso de elaboración del diseño.

```
elaborate $DESIGN
```

La ejecución de este comando suele devolver información interesante a la hora de depurar los códigos del diseño. En la realización de este proyecto aparecieron varios problemas por incompatibilidades de funciones de alto nivel. Si bien *VHDL* es reconocido por el software, existen funciones de alto nivel que *Genus Synthesis Solution* no fue capaz de resolver, como funciones para resolver logaritmos. Estas funciones son usadas para ajustar de forma automática el número de bits que necesitará una cierta entrada de algunos de los modulos en función de un valor definido en un archivo que contiene las constantes del sistema. Para resolverlo se necesitó reajustar los códigos y poner de forma manual los valores necesarios para la implementación.

La menor flexibilidad de *Verilog* frente a *VHDL* hace del primero un lenguaje mucho más cercano al hardware que el segundo. Este tipo de situaciones se ven reflejadas en los números cuando se analiza la cantidad de personas que usan *Verilog* para el diseño de *VLSI* frente a la cantidad de usuarios de *VHDL* en el campo de las *FPGA*[53, 29].

4.3.5 Inicialización del diseño

Durante este paso, el software incializa la base de datos y se asegura de que todas las herramientas están preparadas para la síntesis. Para ello se usa el comando del código 4.3.

Código 4.3 Instrucción para ejecución de la inicialización del diseño.

```
init_design
```

4.3.6 Restricciones de tiempo y diseño

Los requisitos y restricciones de un diseño suelen ser una pieza fundamental a la hora de resolverlo. En el campo del diseño digital pueden existir muchas, sin embargo, casi todas las piezas están supeditadas a la velocidad que les marca un reloj.

Un *circuito secuencial* necesitará de al menos un reloj para trabajar. Diseños más complejos y sofisticados pueden conllevar el uso de varios relojes y circuitería capaz de generar árboles diferentes con velocidades programables. En el caso de este proyecto sólo se ha necesitado de un reloj para delimitar el funcionamiento del *IC* cuyo periodo es de 6*ns*.

Gracias a estos parámetros, la herramienta es capaz de conocer el retraso máximo que puede haber en un camino para ajustar el número de puertas lógicas y transistores del mismo. Si bien habrá circuitos que cumplan holgadamente estas restricciones, puede, darse el caso de que el software sea incapaz de generar un código *RTL* que cumpla estos requisitos. Todo esto lo notifica el programa a través del *log* y los mensajes del *prompt*.

Las restricciones pueden venir en un archivo SDC, como se muestra en el código 4.4.

Código 4.4 Ejemplo de instrucción para cargar las restricciones de tiempo mediante un archivo SDC.

```
read_sdc ../tcl/clock.sdc
```

4.3.7 Directivas de optimización

Antes de comenzar la síntesis, resulta de interés definir el esfuerzo que se espera que realice el programa para conseguir la síntesis. En fases previas puede ser interesante obtener una síntesis preliminar para buscar errores o tener una primera aproximación de la potencia o el área que serán necesarias. Si el circuito tiene un tamaño considerable puede que el proceso se alargue durante unas semanas e interese hacer primero una ejecución con el mínimo esfuerzo.

También puede darse el caso de que el resultado de la síntesis con un esfuerzo medio y alto no varíe demasiado, por lo que *CDS* suele recomendar configurar el esfuerzo en medio para la mayoría de los diseños. En el código 4.5 se muestra un ejemplo de configuración usando un esfuerzo medio para todas las etapas de la síntesis.

Código 4.5 Ejemplo de instrucciones para configurar el esfuerzo de síntesis a nivel medio (recomendado por el desarrollador del software).

```
set_db syn_generic_effort medium
set_db syn_map_effort medium
set_db syn_opt_effort medium
```

4.3.8 Síntesis

El proceso de la síntesis está dividido en tres etapas:

Genérica

Durante esta etapa, *Genus Synthesis Solution* convierte el código a nivel de puertas lógicas. En el código 4.6 se muestra el comando para iniciar el proceso de síntesis genérica.

Código 4.6 Instrucción básica para la ejecución de la síntesis genérica.

```
syn_generic
```

Mapping

Mapea las celdas específicas del diseño y realiza una optimización lógica. En el código 4.7 se muestra el comando para iniciar el proceso de síntesis mapping.

Código 4.7 Instrucción básica para la ejecución de la síntesis de mapeado.

```
syn_map
```

Optimización

En esta última étapa se realiza una optimización mediante iteración de todos los parámetros. En el código 4.8 se muestra el comando para iniciar el proceso de optimización de la síntesis.

Código 4.8 Instrucción básica para la ejecución de la síntesis de optimización.

```
syn_opt
```

4.3.9 Reportes y analisis

Una vez sintetizado el circuito, pueden obtenerse diferentes reportes con la información del proceso. En general puede encontrarse información detallada sobre cada área en los reportes específicos o encontrar información más general en un reporte de resumen (report_summary o report_qor). El código 4.9 se detallan comandos para obtener diferentes reportes con información sobre la síntesis.

Código 4.9 Comandos para generar reportes.

```
report_area         // Contiene información sobre el área necesaria
report_dp           // Contiene información de todos los datapaths, su TNS, WNS
report_design_rules // Contiene información de las reglas de diseño
report_gates        // Contiene información sobre las puertas lógicas
report_hierarchy    // Contiene información sobre la jerarquí de las instancias
report_instance     // Contiene información sobre las instancias
report_memory       // Contiene información de la memoria usada en la síntesis
report_messages     // Contiene
report_power        // Contiene información de la potencia consumida estimada
report_qor          // Contiene información sobre potencia, área, tiempos, etc.
report_timing       // Contiene información de los tiempos
report_summary      // Contiene un resumen de la información más importante
```

Para este proyecto se ha priorizado la información que se obtiene sobre área, tiempos y potencias.

Información de áreas

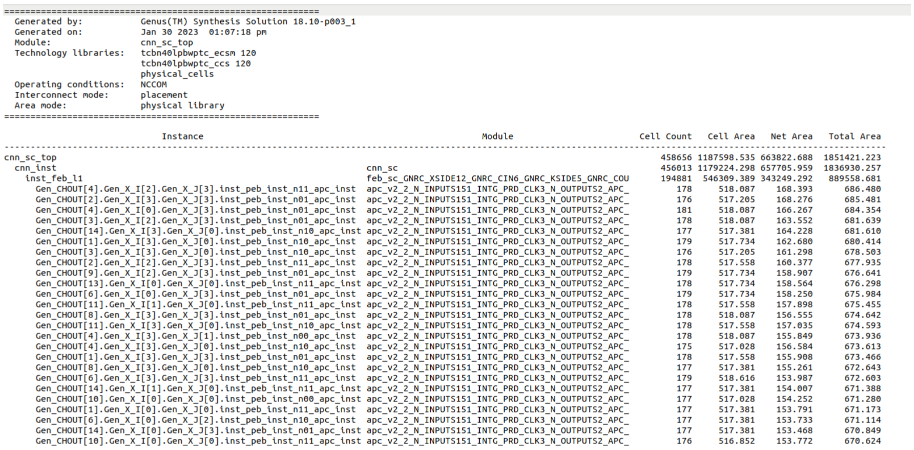
El reporte de áreas muestra una lista de forma jerárquica de todos las instancias del diseño y para cada uno de ellos devuelve el nombre del módulo que se implementa, el número de celdas, el área de las celdas, el área necesaria para el enrutado y el área total. Véase figura 4.1.

En la figura 4.2 se muestra un fragmento de uno de los circuitos implementados en este proyecto. La figura 4.3 recoge el *layout* completo dónde, además, pueden verse las celdas que pertenecen a algunas de las diferentes capas de neuronas del circuito.

Las imágenes de las figuras 4.2 y 4.3 no forman parte del reporte generado, sino que son obtenidas posteriormente usando la interfaz gráfica del software para tareas previas a la creación del *layout*. Sin embargo, son muy ilustrativas para conocer cómo el software funciona y calcula las áreas.

Información de tiempos

Al igual que para los reportes de área, los reportes de tiempos también se presentan en formato de texto y muestran información a través de tablas. Sin embargo, en este caso la información viene marcada por el camino más largo que existe en el circuito y los detalles al respecto. En primer lugar, se informa del periodo del reloj (en pico segundos) y sus detalles para el caso de que deriven de otros relojes. A continuación, se



```
============================================================
  Generated by:          Genus(TM) Synthesis Solution 18.10-p003_1
  Generated on:          Jan 30 2023  01:07:18 pm
  Module:                cnn_sc_top
  Technology libraries:  tcbn40lpbwptc_ecsm 120
                         tcbn40lpbwptc_ccs 120
                         physical_cells
  Operating conditions:  NCCOM
  Interconnect mode:     placement
  Area mode:             physical library
============================================================

                       Instance                                                      Module                          Cell Count  Cell Area   Net Area   Total Area
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
cnn_sc_top                                                                                                              458656 1187598.535 663822.688  1851421.223
  cnn_inst                                            cnn_sc                                                            456013 1179224.298 657705.959  1836930.257
    inst_feb_l1                                       feb_sc_GNRC_XSIDE12_GNRC_CIN6_GNRC_KSIDE5_GNRC_COU                194881  546309.389 343249.292   889558.681
      Gen_CHOUT[4].Gen_X_I[2].Gen_X_J[3].inst_peb_inst_n11_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     178     518.087    168.393      686.480
      Gen_CHOUT[2].Gen_X_I[3].Gen_X_J[3].inst_peb_inst_n01_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     176     517.205    168.276      685.481
      Gen_CHOUT[4].Gen_X_I[0].Gen_X_J[3].inst_peb_inst_n01_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     181     518.087    166.267      684.354
      Gen_CHOUT[3].Gen_X_I[2].Gen_X_J[3].inst_peb_inst_n01_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     178     518.087    163.552      681.639
      Gen_CHOUT[14].Gen_X_I[3].Gen_X_J[0].inst_peb_inst_n10_apc_inst apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     177     517.381    164.228      681.610
      Gen_CHOUT[1].Gen_X_I[3].Gen_X_J[0].inst_peb_inst_n10_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     179     517.734    162.680      680.414
      Gen_CHOUT[3].Gen_X_I[3].Gen_X_J[0].inst_peb_inst_n10_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     176     517.205    161.298      678.503
      Gen_CHOUT[2].Gen_X_I[2].Gen_X_J[3].inst_peb_inst_n11_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     178     517.558    160.377      677.935
      Gen_CHOUT[9].Gen_X_I[2].Gen_X_J[3].inst_peb_inst_n01_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     179     517.734    158.907      676.641
      Gen_CHOUT[13].Gen_X_I[0].Gen_X_J[0].inst_peb_inst_n11_apc_inst apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     178     517.734    158.564      676.298
      Gen_CHOUT[6].Gen_X_I[0].Gen_X_J[3].inst_peb_inst_n01_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     179     517.734    158.250      675.984
      Gen_CHOUT[11].Gen_X_I[1].Gen_X_J[0].inst_peb_inst_n11_apc_inst apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     178     517.558    157.898      675.455
      Gen_CHOUT[8].Gen_X_I[3].Gen_X_J[3].inst_peb_inst_n01_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     178     518.087    156.555      674.642
      Gen_CHOUT[11].Gen_X_I[3].Gen_X_J[0].inst_peb_inst_n10_apc_inst apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     178     517.558    157.035      674.593
      Gen_CHOUT[4].Gen_X_I[3].Gen_X_J[1].inst_peb_inst_n00_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     178     518.087    155.849      673.936
      Gen_CHOUT[4].Gen_X_I[3].Gen_X_J[0].inst_peb_inst_n10_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     175     517.028    156.584      673.613
      Gen_CHOUT[1].Gen_X_I[3].Gen_X_J[3].inst_peb_inst_n01_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     178     517.558    155.908      673.466
      Gen_CHOUT[8].Gen_X_I[3].Gen_X_J[0].inst_peb_inst_n10_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     177     517.381    155.261      672.643
      Gen_CHOUT[6].Gen_X_I[3].Gen_X_J[3].inst_peb_inst_n11_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     179     518.616    153.987      672.603
      Gen_CHOUT[14].Gen_X_I[1].Gen_X_J[0].inst_peb_inst_n11_apc_inst apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     177     517.381    154.007      671.388
      Gen_CHOUT[10].Gen_X_I[0].Gen_X_J[0].inst_peb_inst_n00_apc_inst apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     177     517.028    154.252      671.280
      Gen_CHOUT[1].Gen_X_I[0].Gen_X_J[0].inst_peb_inst_n11_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     177     517.381    153.791      671.173
      Gen_CHOUT[6].Gen_X_I[0].Gen_X_J[2].inst_peb_inst_n10_apc_inst  apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     177     517.381    153.733      671.114
      Gen_CHOUT[14].Gen_X_I[0].Gen_X_J[3].inst_peb_inst_n01_apc_inst apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     177     517.381    153.468      670.849
      Gen_CHOUT[10].Gen_X_I[0].Gen_X_J[0].inst_peb_inst_n11_apc_inst apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_     176     516.852    153.772      670.624
```

Figura 4.1 Reporte de área generado en el diseño de una de las implementaciones del circuito para la *CNN LeNet-5*.

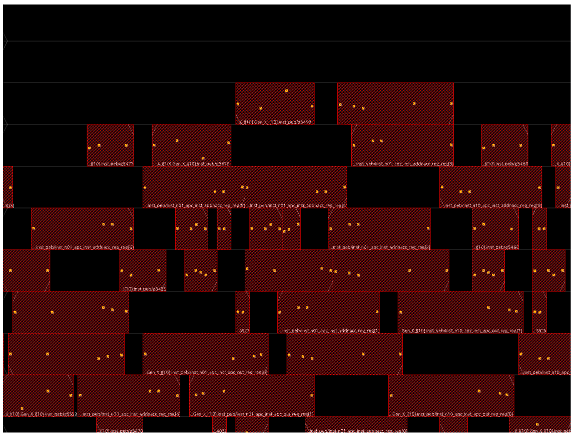


Figura 4.2 Detalle de las celdas generadas en el diseño de una de las implementaciones del circuito para la *CNN LeNet-5*. Cada una de las celdas representa algún tipo de dispositivo lógico (puertas *AND*,*OR*, *biestables*, etc.), los puntos muestran los pines y pads de conexión para las entradas y salidas de cada celda.

calcula el tiempo sobrante basándose en el tiempo de *setup*, y el tiempo que sobra del reloj tras el *setup*. Desde ese valor y restando el tiempo que se necesita para recorrer todas las puertas lógicas necesarias, aparece la holgura (*slack*), la franja de tiempo que sobraría antes del siguiente ciclo de reloj.

Además de esta información, se ofrece una tabla con todos los detalles del camino más largo, en dicha tabla se ofrece una lista con cada uno de los bloques combinacionales que se van cruzando. En el resto de columnas se informa de la dirección, si la señal cambia en eje de subida o bajada, la celda o puerta lógica concreta, el *fan-out*[2], la capacidad (en fentofaradios) que se encuentra a la salida de la celda, el tiempo de la transición, el retraso y el tiempo total en el que el cambio aparecería en la salida de esa celda desde el inicio del camino.

En este contexto aparecen dos parámetros importantes para medir los problemas de *timing*, *Worse Negative Slack* (*WNS*) y *Total Negative Slack* (*TNS*)[3]. Cuanto más negativos sean estos valores, peor será el rendimiento del *ASIC* que se está diseñando, si los valores son positivos, indican que el circuito tiene margen para establecer el reloj del mismo. Pueden verse más detalle en la figura 4.4.

Las figuras 4.5 y 4.6 muestran ventanas que se pueden obtener desde la interfaz gráfica con la información de los tiempos del circuito. En ellas es posible ver la distribución de los trayectos y la cantidad de trayectos según su *slack*, así como la información completa de cada uno de los trayectos en el "Timing Path Analyzer".

[2] El *fan-out* indica el número máximo de puertas lógicas que pueden ser alimentadas desde la salida de otra puerta lógica.

[3] *Worse Negative Slack* y *Total Negative Slack* hace referencia a la porción de tiempo que excede los requisitos de tiempo para un camino concreto (WNS) o el total de ellos (TNS). Los valores negativos indican exceso de tiempo, mientras que los positivos indican tiempo sobrante.

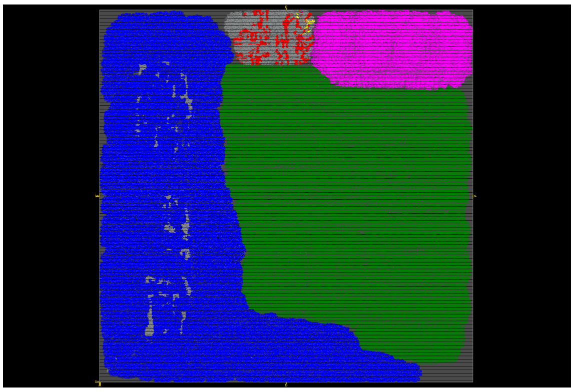


Figura 4.3 Vista superior del *layout* generado en el diseño de una de las implementaciones del circuito para la *CNN LeNet-5*. En azul, se muestran todas las celdas que pertenecen a la implementación de la primera capa convolucional del sistema. En verde, las celdas pertenecientes a la segunda capa convolucional del sistema. En rosa, las celdas de una de las capas *FC*. En rojo, las instancias seleccionadas en el *Design Browser* (explorador de instancias en la izquierda).

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 18.10-p003_1
  Generated on:           Jan 30 2023  01:07:21 pm
  Module:                 cnn_sc_top
  Operating conditions:   NCCOM
  Interconnect mode:      placement
  Area mode:              physical library
============================================================


Path 1: MET (147 ps) Setup Check with Pin cnn_inst/inst_feb_l0/Gen_CHOUT[1].Gen_X_I[1].Gen_X_J[6].inst_peb_inst_n00_apc_inst_apc_out_reg_reg[0]/CP->D
          Group: clk
     Startpoint: (R) cnn_inst/sinc_inst_out_sel_inputs_mux_pc_reg/CP
          Clock: (R) clk
       Endpoint: (R) cnn_inst/inst_feb_l0/Gen_CHOUT[1].Gen_X_I[1].Gen_X_J[6].inst_peb_inst_n00_apc_inst_apc_out_reg_reg[0]/D
          Clock: (R) clk

                     Capture       Launch
        Clock Edge:+    6000            0
       Src Latency:+       0            0
       Net Latency:+       0 (I)        0 (I)
           Arrival:=    6000            0

             Setup:-     101
     Required Time:=    5899
      Launch Clock:-       0
         Data Path:-    5751
             Slack:=     147

#-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
#                                            Timing Point                                                  Flags    Arc   Edge    Cell       Fanout  Load Trans Delay Arrival    Instance
#                                                                                                                                                    (fF)  (ps)  (ps)   (ps)      Location
#-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
  cnn_inst/sinc_inst_out_sel_inputs_mux_pc_reg/CP                                                           -       -      R     (arrival)    21773    -     0     -      0            (-,-)
  cnn_inst/sinc_inst_out_sel_inputs_mux_pc_reg/Q                                                            -       CP->Q  R     DFQD1BWP         1   3.7    52   115    115     (32.480,44.100)
  cnn_inst/FE_OFC1319_sel_inputs_mux_pc/ZN                                                                  -       I->ZN  F     INVD4BWP         6 212.5   113    87    202     (41.440,45.360)
  cnn_inst/FE_OFC1323_sel_inputs_mux_pc/ZN                                                                  -       I->ZN  R     INVD3BWP         2   9.4   294   940   1142  (136.220,560.700)
  cnn_inst/FE_OFC1331_sel_inputs_mux_pc/ZN                                                                  -       I->ZN  F     CKND8BWP         3  90.4   119   131   1272  (132.020,559.440)
  cnn_inst/FE_OFC1347_sel_inputs_mux_pc/ZN                                                                  -       I->ZN  R     INVD3BWP         2  36.3   144   159   1431  (220.780,585.900)
  cnn_inst/inst_feb_l0/FE_OFC1397_sel_inputs_mux_pc/ZN                                                      -       I->ZN  F     INVD4BWP         8  56.0   104   122   1553  (320.880,544.320)
  cnn_inst/inst_feb_l0/FE_OFC1547_sel_inputs_mux_pc/ZN                                                      -       I->ZN  R     CKND4BWP        71  63.6   186   150   1704  (347.760,545.580)
  cnn_inst/inst_feb_l0/FE_OFC1853_sel_inputs_mux_pc/ZN                                                      -       I->ZN  F     INVD4BWP         5  37.3    96   112   1816  (345.940,561.960)
  cnn_inst/inst_feb_l0/FE_OFC2334_sel_inputs_mux_pc/ZN                                                      -       I->ZN  R     INVD6BWP        37  64.5   103   106   1922  (260.400,631.260)
  cnn_inst/inst_feb_l0/g42491/ZN                                                                            -       I->ZN  F     INVD6BWP        37  82.7   100   119   2041  (310.380,653.940)
  cnn_inst/inst_feb_l0/g42475/Z                                                                             -       A1->Z  F     OR2D2BWP         2  11.4    56   112   2154  (315.840,648.900)
  cnn_inst/inst_feb_l0/FE_OFC321_n_1051/ZN                                                                  -       I->ZN  R     INVD6BWP         5  81.5    74    63   2216  (308.000,656.460)
  cnn_inst/inst_feb_l0/FE_OFC330_n_1051/ZN                                                                  -       I->ZN  F     INVD1BWP         9  11.8   115   221   2438  (272.860,939.960)
  cnn_inst/inst_feb_l0/FE_OFC338_n_1051/ZN                                                                  -       I->ZN  R     INVD1BWP         2   3.9    71    74   2511  (269.080,980.280)
  cnn_inst/inst_feb_l0/FE_OFC349_n_1051/ZN                                                                  -       I->ZN  F     INVD4BWP        26  54.1   103    87   2598  (266.700,984.060)
  cnn_inst/inst_feb_l0/FE_DBTC1_n_1051/ZN                                                                   -       I->ZN  R     INVD6BWP        26 106.2   175   144   2742  (244.020,987.840)
  cnn_inst/inst_feb_l0/FE_OFC5541_FE_DBTN1_n_1051/Z                                                         -       I->Z   R     CKBD6BWP        22  71.5    88   123   2866  (244.160,946.260)
  cnn_inst/inst_feb_l0/FE_OFC5542_FE_DBTN1_n_1051/Z                                                         -       I->Z   R     CKBD3BWP        15  51.5   163   208   3073  (277.200,750.960)
  cnn_inst/inst_feb_l0/FE_OFC5543_FE_DBTN1_n_1051/Z                                                         -       I->Z   R     CKBD3BWP        15  60.2   172   193   3267  (277.480,641.340)
  cnn_inst/inst_feb_l0/FE_OFC5544_FE_DBTN1_n_1051/Z                                                         -       I->Z   R     CKBD3BWP        22  52.2   199   242   3508  (312.760,367.920)
  cnn_inst/inst_feb_l0/FE_OFC5545_FE_DBTN1_n_1051/Z                                                         -       I->Z   R     CKBD3BWP        19  54.0   190   188   3696  (324.380,297.360)
  cnn_inst/inst_feb_l0/FE_OFC5546_FE_DBTN1_n_1051/Z                                                         -       I->Z   R     CKBD3BWP        28  45.8   156   178   3874  (250.880,246.960)
  cnn_inst/inst_feb_l0/g39239/ZN                                                                            -       A1->ZN F     CKND2D0BWP       2   1.1    68    84   3958  (200.480,225.540)
  cnn_inst/inst_feb_l0/g39102/Z                                                                             -       B->Z   F     OA21D0BWP        2   9.0   129   130   4088  (201.040,225.540)
  cnn_inst/inst_feb_l0/g38776/ZN                                                                            -       A1->ZN R     CKND2D0BWP       3   8.8   244   182   4271  (253.680,228.060)
  cnn_inst/inst_feb_l0/g38320/S                                                                             -       A->S   R     FA1D0BWP         1   1.7    49   205   4476  (237.020,240.660)
  cnn_inst/inst_feb_l0/g38160/S                                                                             -       B->S   F     FA1D0BWP         1   1.7    40   160   4636  (233.520,240.660)
  cnn_inst/inst_feb_l0/g23262/CO                                                                            -       D->CO  F     CMPE42D1BWP      1   0.9    29   120   4756  (233.520,244.440)
  cnn_inst/inst_feb_l0/g37966/CO                                                                            -       A->CO  F     FA1D0BWP         1   0.8    29   153   4908  (234.500,243.180)
  cnn_inst/inst_feb_l0/g37906/S                                                                             -       CI->S  F     FA1D0BWP         2   2.0    43    96   5005  (234.780,241.920)
  cnn_inst/inst_feb_l0/g20341/CO                                                                            -       A->CO  F     FA1D0BWP         1   1.1    32   162   5166  (230.720,241.920)
  cnn_inst/inst_feb_l0/g20232/CO                                                                            -       CI->CO F     FA1D0BWP         1   2.3    46    98   5264  (229.600,239.400)
  cnn_inst/inst_feb_l0/g20133/CO                                                                            -       B->CO  F     HA1D0BWP         1   2.1    37    66   5330  (227.080,238.140)
  cnn_inst/inst_feb_l0/g19914/S                                                                             -       B->S   R     HA1D0BWP         3   2.7    67    95   5425  (226.100,239.400)
  cnn_inst/inst_feb_l0/g19889/Z                                                                             -       A1->Z  R     AN3D0BWP         2   1.4    46    96   5520  (227.780,240.660)
  cnn_inst/inst_feb_l0/g11213/Z                                                                             -       B->Z   R     AO21D0BWP        1   1.4    42    62   5582  (225.680,240.660)
  cnn_inst/inst_feb_l0/g11212/ZN                                                                            -       A1->ZN F     XNR2D0BWP        2   1.5    38    86   5668  (223.860,241.920)
  cnn_inst/inst_feb_l0/g11211/ZN                                                                            -       B->ZN  R     AOI31D0BWP       1   1.3    96    83   5751  (226.660,241.920)
  cnn_inst/inst_feb_l0/Gen_CHOUT[1].Gen_X_I[1].Gen_X_J[6].inst_peb_inst_n00_apc_inst_apc_out_reg_reg[0]/D <<<   -      R     EDFQD1BWP        1     -     -     0   5751  (223.860,244.440)
#-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Figura 4.4 Reporte de tiempo generado en el diseño de una de las implementaciones del circuito para la *CNN LeNet-5*.

Información de potencias

Para los reportes de potencia se presenta la información de forma parecida al área. Se muestra una lista con todas las instancias del diseño y la información relativa para cada una de ellas.

Las diferentes columnas muestran el números de celdas dentro de la instancia, el consumo estático o *leakage*, consumo de potencia dinámica y el total.

La figura 4.7 representa uno de los informes obtenidos durante un proceso de síntesis. Para este caso, donde

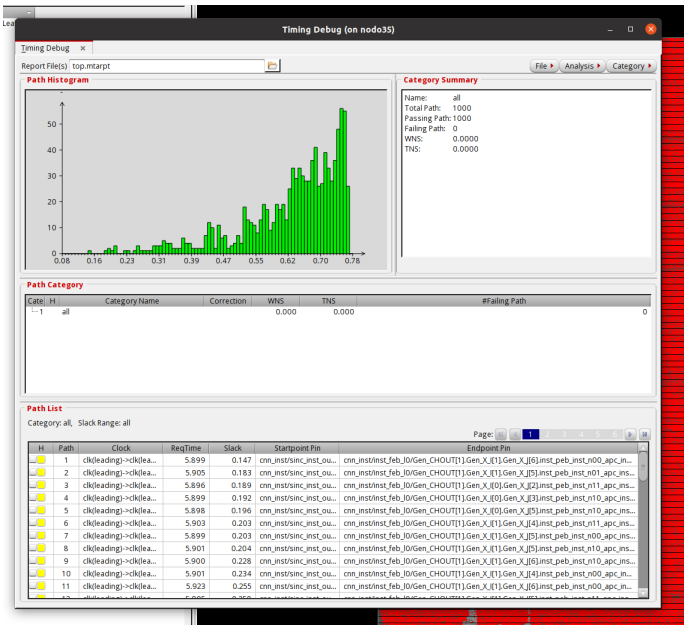


Figura 4.5 Ventana Timing Debug del diseño obtenido en la interfaz gráfica de Genus en el diseño de una de las implementaciones del circuito para la *CNN LeNet-5*. Se muestran todos los caminos que se recorren en el circuito así como sus tiempos y un histograma que muestra el número de caminos que tienen cada *WNS*.

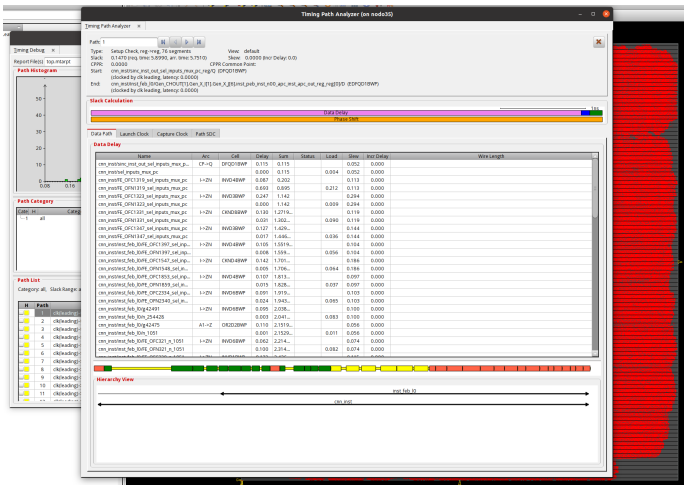
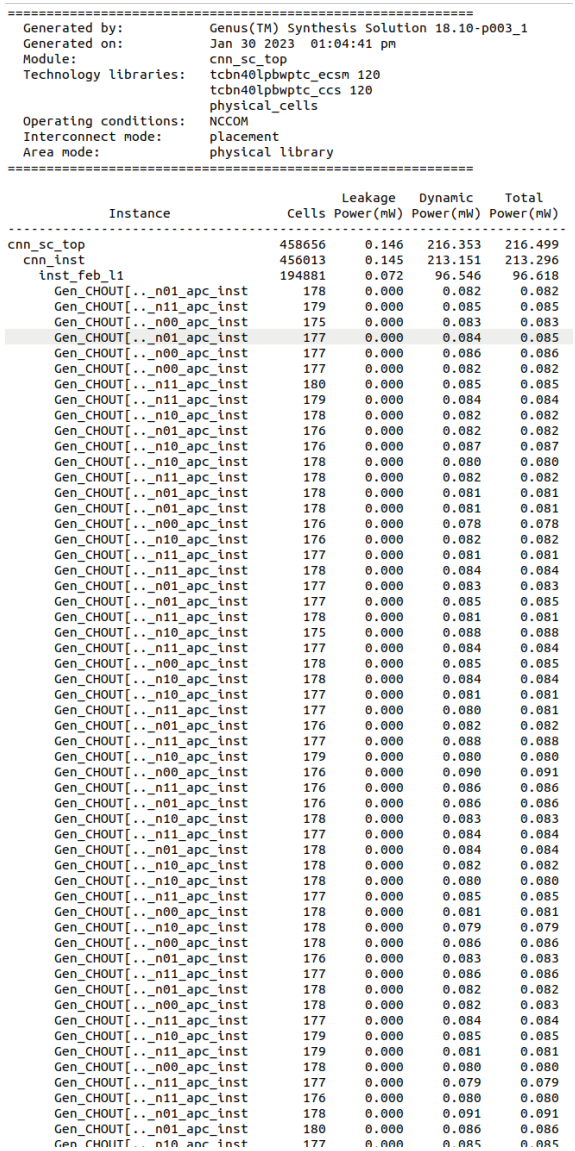


Figura 4.6 Ventana Timing Path Analyzer del diseño obtenido en la interfaz gráfica de Genus en el diseño de una de las implementaciones del circuito para la *CNN LeNet-5*. Se muestra una lista con todas las celdas combinacionales que se recorren en el trayecto, se muestra también de forma gráfica dónde es consumido el tiempo.

el circuito es muy homogéneo y está formado por una gran cantidad de módulos similares, se observa que el consumo está muy distribuido. Sin embargo, en otros diseños con subsistemas más heterogéneos, conocer la potencia de cada instancia puede ayudar a entender qué partes del circuito tienen un mayor consumo y tal vez buscar maneras de optimizar la potencia consumida por dichos módulos.

4.3.10 Evaluación del diseño y reajuste, si es necesario

Como se comentó con anterioridad, estos resultados necesitan ser evaluados y analizados al finalizar el proceso. Si los resultados se ajustan a los esperados, pueden generarse los archivos de salida y continuar el proceso para generar el *floorplanning* y diseñar el *layout* que habría de fabricarse después. A estas alturas del diseño, es importante recordar al lector que el proceso de *síntesis* genera un código con las celdas para el



```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 18.10-p003_1
  Generated on:           Jan 30 2023  01:04:41 pm
  Module:                 cnn_sc_top
  Technology libraries:   tcbn40lpbwptc_ecsm 120
                          tcbn40lpbwptc_ccs 120
                          physical_cells
  Operating conditions:   NCCOM
  Interconnect mode:      placement
  Area mode:              physical library
============================================================

                                        Leakage   Dynamic    Total
           Instance              Cells Power(mW) Power(mW) Power(mW)
-------------------------------------------------------------------
cnn_sc_top                      458656     0.146   216.353   216.499
  cnn_inst                      456013     0.145   213.151   213.296
    inst_feb_l1                 194881     0.072    96.546    96.618
      Gen_CHOUT[..._n01_apc_inst   178     0.000     0.082     0.082
      Gen_CHOUT[..._n11_apc_inst   179     0.000     0.085     0.085
      Gen_CHOUT[..._n00_apc_inst   175     0.000     0.083     0.083
      Gen_CHOUT[..._n01_apc_inst   177     0.000     0.084     0.085
      Gen_CHOUT[..._n00_apc_inst   177     0.000     0.086     0.086
      Gen_CHOUT[..._n00_apc_inst   177     0.000     0.082     0.082
      Gen_CHOUT[..._n11_apc_inst   180     0.000     0.085     0.085
      Gen_CHOUT[..._n11_apc_inst   179     0.000     0.084     0.084
      Gen_CHOUT[..._n10_apc_inst   178     0.000     0.082     0.082
      Gen_CHOUT[..._n01_apc_inst   176     0.000     0.082     0.082
      Gen_CHOUT[..._n10_apc_inst   176     0.000     0.087     0.087
      Gen_CHOUT[..._n10_apc_inst   178     0.000     0.080     0.080
      Gen_CHOUT[..._n11_apc_inst   178     0.000     0.082     0.082
      Gen_CHOUT[..._n01_apc_inst   178     0.000     0.081     0.081
      Gen_CHOUT[..._n01_apc_inst   178     0.000     0.081     0.081
      Gen_CHOUT[..._n00_apc_inst   176     0.000     0.078     0.078
      Gen_CHOUT[..._n10_apc_inst   176     0.000     0.082     0.082
      Gen_CHOUT[..._n11_apc_inst   177     0.000     0.081     0.081
      Gen_CHOUT[..._n11_apc_inst   178     0.000     0.084     0.084
      Gen_CHOUT[..._n01_apc_inst   177     0.000     0.083     0.083
      Gen_CHOUT[..._n01_apc_inst   177     0.000     0.085     0.085
      Gen_CHOUT[..._n11_apc_inst   178     0.000     0.081     0.081
      Gen_CHOUT[..._n10_apc_inst   175     0.000     0.088     0.088
      Gen_CHOUT[..._n11_apc_inst   177     0.000     0.084     0.084
      Gen_CHOUT[..._n00_apc_inst   178     0.000     0.085     0.085
      Gen_CHOUT[..._n10_apc_inst   178     0.000     0.084     0.084
      Gen_CHOUT[..._n10_apc_inst   177     0.000     0.081     0.081
      Gen_CHOUT[..._n11_apc_inst   177     0.000     0.080     0.081
      Gen_CHOUT[..._n01_apc_inst   176     0.000     0.082     0.082
      Gen_CHOUT[..._n11_apc_inst   177     0.000     0.088     0.088
      Gen_CHOUT[..._n10_apc_inst   179     0.000     0.080     0.080
      Gen_CHOUT[..._n00_apc_inst   176     0.000     0.090     0.091
      Gen_CHOUT[..._n11_apc_inst   176     0.000     0.086     0.086
      Gen_CHOUT[..._n01_apc_inst   176     0.000     0.086     0.086
      Gen_CHOUT[..._n10_apc_inst   178     0.000     0.083     0.083
      Gen_CHOUT[..._n11_apc_inst   177     0.000     0.084     0.084
      Gen_CHOUT[..._n01_apc_inst   178     0.000     0.084     0.084
      Gen_CHOUT[..._n10_apc_inst   178     0.000     0.082     0.082
      Gen_CHOUT[..._n10_apc_inst   178     0.000     0.080     0.080
      Gen_CHOUT[..._n11_apc_inst   177     0.000     0.085     0.085
      Gen_CHOUT[..._n00_apc_inst   178     0.000     0.081     0.081
      Gen_CHOUT[..._n10_apc_inst   178     0.000     0.079     0.079
      Gen_CHOUT[..._n00_apc_inst   178     0.000     0.086     0.086
      Gen_CHOUT[..._n01_apc_inst   176     0.000     0.083     0.083
      Gen_CHOUT[..._n11_apc_inst   177     0.000     0.086     0.086
      Gen_CHOUT[..._n01_apc_inst   178     0.000     0.082     0.082
      Gen_CHOUT[..._n00_apc_inst   178     0.000     0.082     0.083
      Gen_CHOUT[..._n11_apc_inst   177     0.000     0.084     0.084
      Gen_CHOUT[..._n10_apc_inst   179     0.000     0.085     0.085
      Gen_CHOUT[..._n11_apc_inst   179     0.000     0.081     0.081
      Gen_CHOUT[..._n00_apc_inst   178     0.000     0.080     0.080
      Gen_CHOUT[..._n11_apc_inst   177     0.000     0.079     0.079
      Gen_CHOUT[..._n11_apc_inst   176     0.000     0.080     0.080
      Gen_CHOUT[..._n01_apc_inst   178     0.000     0.091     0.091
      Gen_CHOUT[..._n01_apc_inst   180     0.000     0.086     0.086
      Gen_CHOUT[..._n10_apc_inst   177     0.000     0.085     0.085
```

Figura 4.7 Reporte de potencia generado en el diseño de una de las implementaciones del circuito para la *CNN LeNet-5*.

diseño del circuito, pero la colocación definitiva y el enrutado forman parte de las siguientes fases del diseño, resultados mostrados como los de la figura 4.3 no muestran un *layout* ya terminado, sino una aproximación al diseño final.

Para el caso que ocupa a este trabajo, donde no se pretende optimizar el diseño, sino obtener una aproximación numérica de los requisitos de área y de reloj, no existe la necesidad de volver atrás a nivel de diseño *front-end* o *back-end* para optimizar el diseño. Para este proyecto, se tuvo que ajustar algunos códigos para evitar errores y reajustar el periodo del reloj en uno de los circuitos. En el siguiente capítulo se desarrollará un poco los detalles.

4.4 Entorno de ejecución

Aunque la automatización puede ayudar al diseñador a maximizar su producción gracias a las herramientas de cálculo, no está exento de resolver otras dificultades.

La ejecución de la síntesis se llevó a cabo en el clúster de computación del *IMSE*, el cual dispone de 20 nodos de computación, cada uno de ellos formados por dos procesadores Intel Xeon de 10 núcleos y 20 hilos, 128GB de RAM entre otras características. Este tipo de arquitecturas de computación son de gran ayuda para

el diseño, simulación y test de diferentes circuitos pues permiten realizar tareas muy pesadas en un tiempo considerablemente inferior.

Código 4.10 Opciones de ejecución para hacer uso de varias máquinas en paralelo.

```
set_db max_cpus_per_server 20
set_db super_thread_servers "node1 node2 node3 node4"
set_db super_thread_rsh_command rsh
```

En el código 4.10 se muestran las opciones de configuración necesarias para poder paralelizar de forma masiva las operaciones de cálculo en el clúster. Para los circuitos basados en *CNN* y el *dataset CIFAR-10* fueron necesarias varias semanas en algunos casos. El sistema completo se dividió en varios circuitos para las capas convolucionales y otros para las capas planas *FC*. Para las capas convolucionales, hubo de aumentarse la memoria de intercambio del sistema a grandes niveles para que las máquinas fuesen capaz de manejar la cantidad de información.

Sin embargo, y después de varios intentos de síntesis del *CIFAR-10* completo, no hubo manera de finalizar el proceso de síntesis y obtener los datos buscados. El considerable tamaño del circuito y la falta de tiempo obligaron a que se diese, este último diseño, por fracasado. No obstante, el diseño modular obtenido con el sistema dividido es suficiente para conocer las necesidades e incluso para poder unirlos como diferentes bloques para las futuras fases de diseño del *layout*.

5 Resultados

La calidad nunca es un accidente; siempre es el resultado de un esfuerzo de la inteligencia

JOHN RUSKIN

Recoger los frutos del trabajo realizado suele ser uno de los grandes momentos tras largas jornadas de esfuerzo. Y así fue también para este proyecto. Ha de tenerse en cuenta que cada proceso de síntesis es complejo y tiene una componente estocástica, por lo que dos ejecuciones del mismo código no devuelven siempre los mismos resultados, aunque sí suelen encontrarse muchas similitudes. En algunos de los casos se repitieron las ejecuciones para volver a obtener ciertos informes o debido a que el sistema no podía cumplir las restricciones de tiempo.

Salvo en algunos casos, los relojes se configuraron con un periodo de 6000 ps ó 6 ns, sin embargo, en algunos casos se aumento a 10*ns* para que el resultado fuese implementable. *Genus Synthesis Solution* suele mostrar resultados de *WNS* en valores negativos cuando no se cumplen los requisitos, al no ser unas pruebas definitivas y en modo *Multi-Mode Multi-Corner* (*MMMC*)[1], no se puede saber la precisión que se obtendría al fabricarse el diseño. No obstante, la intención de este proyecto no es desarrollar un producto final, sino demostrar las capacidades de la computación estocástica para el diseño de hardware para *ML*. A continuación se detalla información de los resultados que se obtuvieron para los diferentes casos de estudios.

El objeto de estudio de este trabajo es el diseño *back-end* de distintos diseños proporcionados, por lo que no se entrará en detalles en cuanto a los resultados y conclusiones, sobre *AI* y *SC*, para ello puede acudirse al artículo publicado[16].

5.1 Red neuronal de 64 neuronas

Para este caso se recibieron dos códigos de prueba, uno con el nombre de “No-opt” y otro con el de “Optimized” con un número similar de neuronas.

5.1.1 No-opt

Las tablas 5.1 y 5.2 muestran los resultados para el modelo “no-opt” de 64 neuronas. Se trata de un circuito bastante sencillo con menos de mil celdas y 45 μm x 45 μm. El trayecto máximo es de 2,12 ns, por lo que podría trabajar con un reloj superior a 450MHz. El consumo de potencia estática es despreciable y en dinámica y total sería de 220 μW.

Tabla 5.1 Tabla de resultados del modelo “no-opt”: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
661	232	429	0,0015 mm²	0,0005 mm²	0,002 mm²

[1] El modo *MMMC* realiza el proceso para múltiples esquinas, esto es, realiza el proceso usando los mejores y peores valores de los parámetros del fabricante para obtener una previsión de cuales podrían ser los rangos de funcionamiento de los circuitos debido a las imperfecciones del silicio y la fabricación.

Tabla 5.2 Tabla de resultados del modelo "no-opt": retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
2,12 ns	0 mW	0,22 mW	0,22 mW

5.1.2 Optimized

Para el caso de la red neuronal "optimized", las tablas 5.3 y 5.4 contienen los valores de celdas, áreas, tiempos y potencia. El número de celdas sigue siendo bastante pequeño. El área crece ligeramente hasta los 59 μm x 59 μm. El retraso máximo es ligeramente inferior y sigue siendo por debajo del periodo del reloj establecido, por lo que, en este caso también podría funcionar con relojes bastante rápidos. En cuanto a potencia, de nuevo se puede despreciar el consumo en estática y el consumo total duplica el del caso "no-opt".

Tabla 5.3 Tabla de resultados del modelo "optimized": número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
774	368	406	0,0021 mm²	0,0006 mm²	0,0026 mm²

Tabla 5.4 Tabla de resultados del modelo "optimized": retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
1,37 ns	0 mW	0,43 mW	0,43 mW

5.2 LeNet-5

A continuación se muestran los resultados obtenidos con la implementación de la red *LeNet-5* en *SC*. Entre los objetivos esperados estaba la necesidad de obtener los datos con un modelo de 2 bits de resolución y otro de 4 bits.

5.2.1 2 bits

El paso de una red neuronal de 64 neuronas a un modelo complejo basado en *CNN* es, sin duda, una gran diferencia en cuanto a capacidad y complejidad, y por supuesto, en tiempo de computación. Esta red tiene del orden de sesenta mil parámetros y varios miles de neuronas.

Como cabe esperar, los números de la tabla 5.5 y 5.6 reflejan el cambio. El número de celdas aumenta un orden de magnitud, por encima del medio millón. El tamaño del dado aumentaría hasta 1,39 mm x 1,39 mm. Así como su potencia que pasa a ser de 204 mW. Sin embargo, aunque los retrasos son mayores, 4,175 ns, sigue permitiendo alimentar el reloj a unas tasas de 200 MHz.

Tabla 5.5 Tabla de resultados del modelo LeNet-5 con 2 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
549K	51.029	497.967	1,426 mm²	0,491 mm²	1,917 mm²

Tabla 5.6 Tabla de resultados del modelo LeNet-5 con 2 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
4,175 ns	0,172 mW	203,57 mW	203,744 mW

5.2.2 4 bits

Usando una resolución de 4 bits para la misma arquitectura puede apreciarse que el número de celdas aumenta un 20 % y la potencia consumida se duplica, sin embargo, los tiempos y el área, aunque aumentan,

se mantienen en un margen considerablemente óptimo. En concreto, pueden verse lo resultados en las tablas 5.7 y 5.8.

El número de celdas alcanza la cifra de 663 mil celdas, casi un 90 % de ellas de lógica combinacional y el resto de lógica secuencial. El área total alcanza los 2,137mm², lo que equivale a un tamaño del dado de 1,46 mm x 1,46 mm, el área para las pistas sigue siendo de un 20 % del total. Los tiempos se acercan al límite de los requisitos impuestos con un retraso máximo de 5.63 ns. Por su parte, la potencia estática es de 0,173 mW y la dinámica de 405,74 mW, obteniéndose un total de 405,91 mW.

Tabla 5.7 Tabla de resultados del modelo LeNet-5 con 4 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
663K	76.847	586.334	1,521 mm²	0,617 mm²	2,137 mm²

Tabla 5.8 Tabla de resultados del modelo LeNet-5 con 4 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
5,636 ns	0,173 mW	405,74 mW	405,91 mW

5.3 CNN para CIFAR-10

El tamaño de la *CNN* necesaria para procesar los datos de un *dataset* de este tamaño, sin duda, requiere de una gran cantidad de recursos. Como se ha ido viendo a lo largo de los resultados anteriores, las redes neuronales no escalan de forma lineal, sino que lo hacen de forma exponencial, por lo que cada vez que se aumenta la complejidad de la red aumenta el número de parámetros. Para este diseño en concreto se pasan de los 60.000 parámetros de LeNet-5 a 30.000.000 de parámetros para esta arquitectura.

Este número de parámetros hace necesario que la red se divida en diferentes partes a través de las diferentes capas. En concreto, el diseño se dividió en seis partes: cuatro capas convolucionales y otras dos *FC*. Cada una de estas partes se sintetizará por separado. Por desgracia, *Genus Synthesis Solution* no fue capaz de sintetizar el modelo completo con esta tecnología, sin embargo, se pueden extrapolar los datos que se obtienen de las diferentes partes.

5.3.1 4 bits

En las siguientes tablas se muestran los resultados de todas las capas utilizadas para la *CNN*, puede verse que las capas convolucionales son considerablemente mayores que las planas, en especial la segunda, que alcanza un mayor tamaño. El número de celdas por cada *dado* aumenta al orden de los millones de unidades, en concreto, las celdas de tipo combinacional. Al aumentar el número de celdas, lógicamente, el tamaño del *dado* crece hasta un máximo de 7,3 mm x 7,3 mm para la segunda capa convolucional, y la potencia alcanza valores de decenas de vatios, por lo que aumenta más de 20 veces el consumo. No obstante, los retrasos producidos en el circuito se reducen drásticamente por debajo de los 3 ns, lo que permitiría usar relojes de 333MHz.

De todas las capas sintetizadas, las de mayor consumo y tamaño son la segunda y la cuarta, mientras la primera y la tercera mantienen un término medio y, como puede esperarse, las dos capas *FC* son significativamente más pequeñas.

Primera capa convolucional

En las tablas 5.9 y 5.10 se muestran los resultados para la primera capa convolucional:

Tabla 5.9 Tabla de resultados del modelo CIFAR-10 capa 1 con 4 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
3,04M	432.001	2.605.358	6,82 mm²	2,44 mm²	9,25 mm²

Tabla 5.10 Tabla de resultados del modelo CIFAR-10 capa 1 con 4 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
1,3 ns	0,76 mW	1.710 mW	1.711 mW

Segunda capa convolucional

En las tablas 5.11 y 5.12 se muestran los resultados para la Segunda capa convolucional:

Tabla 5.11 Tabla de resultados del modelo CIFAR-10 capa 2 con 4 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
16M	451.584	16.430.225	38,29 mm²	14,45 mm²	52,74 mm²

Tabla 5.12 Tabla de resultados del modelo CIFAR-10 capa 2 con 4 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
2,241 ns	4,84 mW	11.766 mW	11.771 mW

Tercera capa convolucional

En las tablas 5.13 y 5.14 se muestran los resultados para la tercera capa convolucional:

Tabla 5.13 Tabla de resultados del modelo CIFAR-10 capa 3 con 4 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
6,02M	156.673	6.040.392	13,9 mm²	5,28 mm²	19,18 mm²

Tabla 5.14 Tabla de resultados del modelo CIFAR-10 capa 3 con 4 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
2,35 ns	1,75 mW	4.187 mW	4.189 mW

Cuarta capa convolucional

En las tablas 5.15 y 5.16 se muestran los resultados para la cuarta capa convolucional:

Tabla 5.15 Tabla de resultados del modelo CIFAR-10 capa 4 con 4 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
8,13M	115.201	8.015.442	18,93 mm²	7,05 mm²	25,98 mm²

Tabla 5.16 Tabla de resultados del modelo CIFAR-10 capa 4 con 4 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
2,82 ns	2,4 mW	6.109 mW	6.112 mW

Primera capa FC

En las tablas 5.17 y 5.18 se muestran los resultados para la primera capa *FC*:

Tabla 5.17 Tabla de resultados del modelo CIFAR-10 capa 5 con 4 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
1,75M	10.753	1.742.357	4,1 mm²	1,73 mm²	5,83 mm²

Tabla 5.18 Tabla de resultados del modelo CIFAR-10 capa 5 con 4 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
2,14 ns	0,53 mW	1.409 mW	1.409 mW

Segunda capa FC

En las tablas 5.19 y 5.20 se muestran los resultados para la segunda capa *FC*:

Tabla 5.19 Tabla de resultados del modelo CIFAR-10 capa 6 con 4 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
10,9K	181	10.731	0,026 mm²	0,011 mm²	0,037 mm²

Tabla 5.20 Tabla de resultados del modelo CIFAR-10 capa 6 con 4 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
2,72 ns	0 mW	8,49 mW	8,49 mW

5.3.2 8 bits

Al igual que para el caso de 4 bits, desde la tabla 5.21 hasta la tabla5.32 se muestran los resultados para las distintas capas en el modelo de 8 bits. Los números obtenidos son del mismo orden que para el caso de 4 bits. Al final del capítulo se hablará un poco más de ellos.

Primera capa convolucional

En las tablas 5.21 y 5.22 se muestran los resultados para la primera capa convolucional:

Tabla 5.21 Tabla de resultados del modelo CIFAR-10 capa 1 con 8 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
4,06M	662.401	3.394.944	8,7 mm²	8,52 mm²	17,22 mm²

Tabla 5.22 Tabla de resultados del modelo CIFAR-10 capa 1 con 8 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
2,02 ns	0,97 mW	1.995 mW	1.996 mW

Segunda capa convolucional

En las tablas 5.23 y 5.24 se muestran los resultados para la Segunda capa convolucional:

Tabla 5.23 Tabla de resultados del modelo CIFAR-10 capa 2 con 8 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
17,1M	652.288	16.371.956	39,9 mm²	14,9 mm²	54,8 mm²

Tabla 5.24 Tabla de resultados del modelo CIFAR-10 capa 2 con 8 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
3,1 ns	5 mW	11.847 mW	11.852 mW

Tercera capa convolucional

En las tablas 5.25 y 5.26 se muestran los resultados para la tercera capa convolucional:

Tabla 5.25 Tabla de resultados del modelo CIFAR-10 capa 3 con 8 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
6,34M	230.401	6.114.013	14,41 mm²	5,38 mm²	19,79 mm²

Tabla 5.26 Tabla de resultados del modelo CIFAR-10 capa 3 con 8 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
2,79 ns	1,79 mW	4.311 mW	4.313 mW

Cuarta capa convolucional

En las tablas 5.27 y 5.28 se muestran los resultados para la cuarta capa convolucional:

Tabla 5.27 Tabla de resultados del modelo CIFAR-10 capa 4 con 8 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
8,77M	166.401	8.605.909	19,71 mm²	7,59 mm²	27,3 mm²

Tabla 5.28 Tabla de resultados del modelo CIFAR-10 capa 4 con 8 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
3,68 ns	2,49 mW	6.208 mW	6.210 mW

Primera capa FC

En las tablas 5.29 y 5.30 se muestran los resultados para la primera capa *FC*:

Tabla 5.29 Tabla de resultados del modelo CIFAR-10 capa 5 con 8 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
1,76M	12.801	1.747.291	4,12 mm²	1,73 mm²	5,85 mm²

Tabla 5.30 Tabla de resultados del modelo CIFAR-10 capa 5 con 8 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
2,32 ns	0,53 mW	1.409 mW	1.409 mW

Segunda capa FC

En las tablas 5.31 y 5.32 se muestran los resultados para la segunda capa *FC*:

Tabla 5.31 Tabla de resultados del modelo CIFAR-10 capa 6 con 8 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
11,0K	221	10.761	0,026 mm²	0,011 mm²	0,037 mm²

Tabla 5.32 Tabla de resultados del modelo CIFAR-10 capa 6 con 8 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
2,88 ns	0 mW	8,59 mW	8,59 mW

5.3.3 Estimación CIFAR-10 completo

Como se ha mencionado anteriormente no se pudo obtener el resultado de la síntesis completa. Sin embargo, sería posible diseñar los módulos por separados y unirlos como bloques, por lo que una aproximación del resultado se puede obtener con los datos obtenidos. El error de la aproximación que se va a hacer a continuación viene del área de las pistas necesarias para unir los módulos, por lo que hay que tener claro que estos datos son solamente una aproximación.

Para obtener estos números, se ha sumado el número de celdas, las áreas y las potencias, para el retraso máximo del trayecto, se ha tomado el mayor.

4 bits

En las tablas 5.33 y 5.34 se muestran los datos extrapolados del circuito completo para 4 bits. Las capas convolucionales contienen 34 millones de celdas del total reflejado en estas tablas y en concreto, casi la mitad pertenece a la segunda capa convolucional. Por supuesto, se encuentra la misma relación en los consumos. El área de las pistas es de alrededor del 27 % para cada modulo y del total.

Tabla 5.33 Tabla de datos estimados del modelo CIFAR-10 completo con 4 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
36M	1.166.393	34.844.505	82,1 mm²	30,96 mm²	113,02 mm²

Tabla 5.34 Tabla de datos estimados del modelo CIFAR-10 completo con 4 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
2,82 ns	10,28 mW	25.190 mW	25.201 mW

8 bits

En las tablas 5.35 y 5.36 se muestran los datos extrapolados del circuito completo para 8 bits. Las capas convolucionales contienen 37 millones de celdas del total reflejado en estas tablas y en concreto, casi la mitad pertenece a la segunda capa convolucional. De nuevo, se encuentra la misma relación en los consumos. El área

de las pistas es de alrededor del 30 % del total, sin embargo, en este caso, la primera capa convolucional tiene repartido el área en un poco más de un 50 % para celdas y un algo menos para las pistas que las interconectan.

Tabla 5.35 Tabla de datos estimados del modelo CIFAR-10 completo con 8 bits de resolución: número de celdas y área.

Núm. celdas	C. secuencial	C. combinacional	Área celdas	Área pistas	Área total
37M	1.724.513	36.244.874	82,87 mm²	38,12 mm²	125 mm²

Tabla 5.36 Tabla de datos estimados del modelo CIFAR-10 completo con 8 bits de resolución: retraso máximo de trayectos y potencias.

Máx retraso trayecto	Pot. estática	Pot. dinámica	Pot. total
3,68 ns	10,78 mW	25.779 mW	25.789 mW

5.4 Comparativas

En esta sección se hará una breve comparativa de los resultados presentados aquí.

5.4.1 LeNet-5: 2 vs 4 bits

En cuanto al número de celdas total, la versión de 2 bits necesita un 83 % que su hermana mayor, un 66 % de celdas secuenciales y un 85 % de celdas combinacionales, por lo que una mayor resolución no implica mucha más lógica combinacional, pero sí hay una gran diferencia en cuanto a número de biestables y componentes secuenciales.

Comparando áreas, los datos muestran que la versión de 2 bits necesita un 93 % de área para celdas comparada a la de 4 bits. La diferencia de área para pistas es mayor, ya que la versión de 2 bits sólo necesita un 0,79 % que la versión de 4 bits. En total, la relación de 2 bits con la de 4 bits es del 90 %.

Para los tiempos, el retraso máximo de la versión de 2 bits es del 75 % frente a la de 4 bits. La potencia estática es casi la misma en ambos casos, sin embargo, el consumo de *LeNet-5* 2 bits es la mitad del de 4 bits.

5.4.2 CIFAR-10: 4 vs 8 bits

Para la *CNN* de *CIFAR-10* se obtienen unos números muy parecidos, excepto en potencia:

Celdas totales: 94 %, secuenciales: 67 % y combinacional: 96 %. La relación entre las celdas secuenciales se mantiene en la relación 2bits/4bits de LeNet-5 y la relación 4 bits/8 bits de *CIFAR-10*. Sí que cambia en cuanto a celdas combinacionales, donde puede verse que la versión de 4 y 8 bits de *CIFAR-10* necesitan de un número muy similar de celdas combinacionales.

En términos de área, el área consumida para celdas en la versión de 4 bits es del 95 % en la versión de 8 bits y del 81 % para el área de las pistas. El total, por tanto, es de un 90,4 % del área total. Los tiempos tienen una relación del 76,6 % de retraso en la versión de 4 bits frente a la de 8.

El consumo de potencia estática sigue siendo muy similar en ambos modelos, con una relación del 95 %. Para el total y la dinámica, que sigue representando el grueso del total, las relaciones son del 97,7 % en ambos casos.

5.4.3 LeNet-5 4bits vs CIFAR-10 4bits

Para terminar este capítulo, se explica la relación entre *LeNet-5* de 4 bits y *CIFAR-10* de 4 bits, con el objetivo de observar el efecto que produce el aumento de parámetros en cuanto a electrónica:

Primero, hay que hacer notar que *LeNet-5* tiene 60.000 parámetros, mientras que *CIFAR-10* tiene 30.000.000 de parámetros, viendo la relación, el tamaño de *LeNet-5* es del 0,2 % del tamaño de *CIFAR-10*, o dicho de otra forma, *CIFAR-10* es 500 veces mayor que *LeNet-5*.

La relación de celdas secuenciales entre *LeNet-5* y *CIFAR-10* es del 6,6 %, 1,7 % de celdas combinacionales y en total, 1,8 % de celdas entre *LeNet-5* y *CIFAR-10*.

La relación entre las áreas, tanto para celdas como para pistas y el total, es del 2 %.

No obstante, el retraso que se produce en *LeNet-5* es del 200 % de *CIFAR-10*.

En cuanto a potencia, las relaciones son muy similares para potencia estática, dinámica y, por supuesto, para el total, las cuales son del orden del 1,6 %.

Estos números demuestran que esta tecnología escala bien cuando crecen los parámetros del sistema, ya que las capacidades del circuito aumentan mucho más que los requisitos de hardware necesarios para su implementación.

6 Conclusiones

A veces creo que hay vida en otros planetas, y a veces creo que no. En cualquiera de los dos casos la conclusión es asombrosa.

Carl Sagan

Mayoritariamente se han conseguido los objetivos que se establecieron en el capitulo 3. Este trabajo trata de cubrir varias áreas muy relevantes de la ingeniería en los tiempos que corren, si bien cualquiera de las áreas aquí tratadas podrían dar lugar a varios libros, como si de unos cimientos se tratasen, se ha puesto la cantidad de información adecuada para levantar estas páginas y conseguir que los lectores tengan un entendimiento aceptable de lo aquí expuesto.

6.1 Sobre el diseño electrónico

La era de la información ha sido posible gracias al desarrollo de dispositivos electrónicos que han revolucionado el mundo. Desde hace décadas, los investigadores de este campo han ido desarrollando nuevos diseños, dispositivos y tecnologías que se reinventan a sí mismos de forma constante.

Desde los primeros transistores al primer *IC*, desde las primeras puertas lógicas al primer computador, no sólo se ha trabajado en esos diseños sino que han generado un nuevo mundo. Un mundo digital y no tangible que ha sido capaz de comunicarnos con otras personas a miles de kilómetros de distancia. Ha sido capaz de solucionar cálculos que tomarían años para un ser humano en tan solo unos minutos. Horas de entretenimiento, arte y cultura en la palma de la mano.

Las herramientas que *CDS* ofrece para el diseño de circuitos *VLSI* digitales ha demostrado ser efectiva y potente. Sin ser demasiado compleja, se necesitan bastantes conocimientos de electrónica y de la propia herramienta para poder resolver los problemas que surgen durante la síntesis. Además, al ser un software tan potente y con tanta necesidad de cálculos, imposibilita a muchos usuarios el acceso a la herramienta, pues se necesita de potentes servidores para conseguir resultados y en unos tiempos competitivos.

A pesar de los requisitos, el proceso de diseño de síntesis de *Genus Synthesis Solution* es enormemente más rápido que el diseño manual, el cual sería una tarea titánica cuando se habla de cientos de millones de transistores.

6.2 Líneas futuras

El diseño *back-end* no sólo consiste en sintetizar el diseño de la parte *front-end*, queda aún un largo camino hasta llegar al producto final.

Para este proyecto se ha planteado la realización física de alguna *CNN* con esta tecnología. Desgraciadamente, los costes de fabricación de un *ASIC* pueden rondar las decenas de millares de Euros. La fabricación de *IC* no es cara cuando se produce a gran escala, pero sí cuando no se necesitan miles de unidades. A pesar de esto, en estos momentos se está trabajando en terminar el resto del diseño usando el diseño de *LeNet-5*.

6.2.1 Otras líneas de investigación

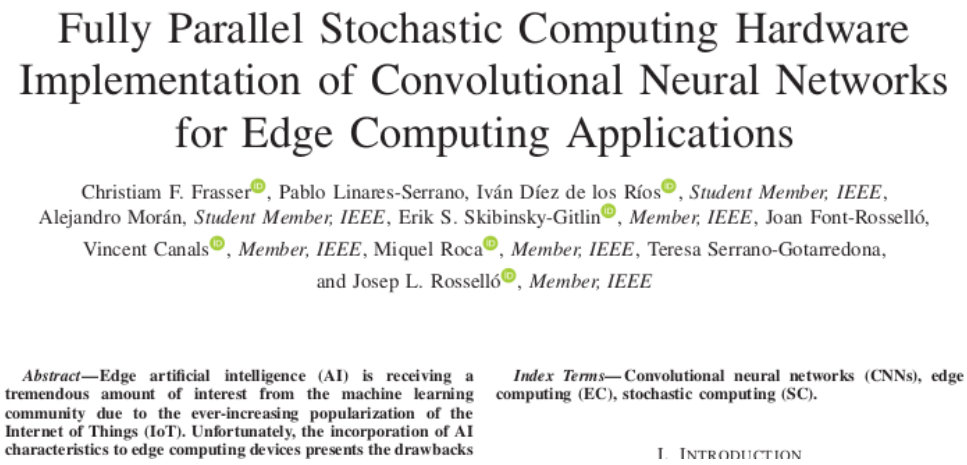
Por otro lado, y fuera de este proyecto pero dentro de la línea del mismo, se están llevando a cabo más investigaciones enfocadas en *Computación Neuromórfica*, tanto con sistemas digitales como analógicos. Para ello se hace uso de *memristores* o *Resistive Random-Access Memory* (*RRAM*) como sinápsis neuronales en redes *SNN* analógicas continuando trabajos previos de compañeros[5, 6].

Además, también se está continuando otras investigaciones que agrupan el terreno Neuromórfico y la biología en la búsqueda de tratamientos regenerativos para la epilepsia[1, 2, 3]. Entre las principales tareas planteadas se encuentran el diseño analógico, digital, test y caracterización de *IC* o programación de diferentes dispositivos digitales como microcontroladores o *FPGA*. Todas estás líneas de trabajo se están llevando a cabo en el *IMSE* y la *US* como estudiante de doctorado en la misma universidad.

Apéndice A
Publicaciones

Dentro de la colaboración llevada a cabo entre la UIB y el *IMSE*, se publicó un artículo en *IEEE Transactions on Neural Network and Learning Systems* titulado *Fully Parallel Stochastic Computing Hardware Implementation of Convolutional Neural Networks for Edge Computing Applicationes*[16]. Este artículo detalla las técnicas que usaron para el diseño *front-end*, las características de las distintas capas usadas, así como los resultados obtenidos de su implementación en *FPGA*. Para terminar, el artículo muestra los resultados de la sintetización con la implementación en circuitos *VLSI*, los cuales fueron extraídos de los resultados de este trabajo, en concreto, la parte IV. B. habla sobre la implementación que se explica en este trabajo.

Pueden encontrarse en este artículo las conclusiones sobre el uso del diseño propuesto y algunas explicaciones más extensas sobre el diseño *front-end*. La figura A.1 muestra el encabezado del artículo a modo ilustrativo.



Fully Parallel Stochastic Computing Hardware Implementation of Convolutional Neural Networks for Edge Computing Applications

Christiam F. Frasser, Pablo Linares-Serrano, Iván Díez de los Ríos, *Student Member, IEEE*, Alejandro Morán, *Student Member, IEEE*, Erik S. Skibinsky-Gitlin, *Member, IEEE*, Joan Font-Rosselló, Vincent Canals, *Member, IEEE*, Miquel Roca, *Member, IEEE*, Teresa Serrano-Gotarredona, and Josep L. Rosselló, *Member, IEEE*

***Abstract*—Edge artificial intelligence (AI) is receiving a tremendous amount of interest from the machine learning community due to the ever-increasing popularization of the Internet of Things (IoT). Unfortunately, the incorporation of AI characteristics to edge computing devices presents the drawbacks**

***Index Terms*— Convolutional neural networks (CNNs), edge computing (EC), stochastic computing (SC).**

I. Introduction

Figura A.1 Encabezado del artículo publicado por *Frasser et al.* en la revista *IEEE Transactions on Neural Network and Learning Systems*.

Apéndice B
Tabla comparativa

En la figura B.1 de la página siguiente se muestra una tabla con todos los datos recogidos de los resultados durante la *síntesis*

Circuit	Total numer of Cells	Seq. Cells	Combin. Cells	Area Cells (mm2)	Area Nets (mm2)	Total Area (mm2)	Max Data Path delay (ns)	Leakage Power (mW)	Dynamic Power (mW)	Total Power (mW)
PEB-64 No-opt	661	232	429	0,0015	0,0005	0,002	2,12	0	0,22	0,22
PEB-64 optimized	774	368	406	0,0021	0,0006	0,0026	1,37	0	0,43	0,43
LeNet-5 2bits	549K	51029	497967	1,426	0,491	1,917	4,175	0,172	203,57	203,744
LeNet-5 4bits	663K	76847	586334	1,521	0,617	2,137	5,636	0,173	405,74	405,91
CIFAR10_4bits feb_layer0_sc	3.04M	432001	2605358	6,82	2,44	9,25	1,3	0,76	1710	1711
CIFAR10_4bits feb_layer1_sc	16M	451584	16430225	38,29	14,45	52,74	2,241	4,84	11766	11771
CIFAR10_4bits feb_layer2_sc	6.20M	156673	6040392	13,9	5,28	19,18	2,35	1,75	4187	4189
CIFAR10_4bits feb_layer3_sc	8.13M	115201	8015442	18,93	7,05	25,98	2,82	2,4	6109	6112
CIFAR10_4bits fc_layer0_sc	1.75M	10753	1742357	4,1	1,73	5,83	2,14	0,53	1409	1409
CIFAR10_4bits fc_layer1_sc	10.9K	181	10731	0,026	0,011	0,037	2,72	0	8,49	8,49
CIFAR10_4bits full	36010898	1166393	34844505	82,066	30,961	113,017	2,82	10,28	25189,49	25201
CIFAR10_8bits feb_layer0_sc	4.06M	662401	3394944	8,7	8,52	17,22	2,02	0,97	1995	1996
CIFAR10_8bits feb_layer1_sc	17.1M	652288	16371956	39,9	14,9	54,8	3,1	5	11847	11852
CIFAR10_8bits feb_layer2_sc	6.34M	230401	6114013	14,41	5,38	19,79	2,79	1,79	4311	4313
CIFAR10_8bits feb_layer3_sc	8.77M	166401	8605909	19,71	7,59	27,3	3,68	2,49	6208	6210
CIFAR10_8bits fc_layer0_sc	1.76M	12801	1747291	4,12	1,73	5,85	2,32	0,53	1409	1409
CIFAR10_8bits fc_layer1_sc	11.0K	221	10761	0,026	0,0011	0,037	2,88	0	8,59	8,59
CIFAR10_8bits full	37969387	1724513	36244874	86,866	38,1211	124,997	3,68	10,78	25778,59	25789

Figura B.1 Tabla comparativa con todos los resultados obtenidos (CIFAR-10 FULL 4 y 8 bits estimados).

Apéndice C
Código TCL para síntesis

A continuación, el código C.1 muestra un ejemplo de código usado para la ejecución de la síntesis. La estructura básica usada se basa en la recomendada por *CDS* con las modificaciones oportunas para el caso concreto de este proyecto y las librerías del fabricante. El flujo básico de este código es el descrito en el capítulo 4, con algunas modificaciones realizadas para mejorar la ejecución o solventar errores.

Código C.1 Código TCL utilizado en Genus.

```
################################################################################
##
##  Script for Cadence Genus
##
################################################################################

date

set_db hdl_language {vhdl}

set LOCAL_DIR "[exec pwd]/.."
set SYNTH_DIR ${LOCAL_DIR}/work
set TCL_PATH   "${LOCAL_DIR}/tcl $LOCAL_DIR/constraints"
set REPORTS_PATH      "${LOCAL_DIR}/work/reports"
set LIB_PATH   "${LOCAL_DIR}/libraries/"
set RTL_PATH   "$LOCAL_DIR/rtl/src $LOCAL_DIR/rtl/src/05-pc_tree $LOCAL_DIR/rtl
    /src/packages"

set DESIGN   "cnn_sc_top"

set MSGS_TO_BE_SUPRESSED {LBR-58 LBR-40 LBR-41 VLOGPT-35}

set LIB_LIST { \
  tcbn40lpbwptc_ecsm.lib \
}

set LEF_LIST { \
  tcbn40lpbwp_7lm4X2ZRDL.lef \
}

set RTL_LIST "[exec ls ${LOCAL_DIR}/rtl/src/packages] [exec ls ${LOCAL_DIR}/rtl
    /src/05-pc_tree] [exec ls ${LOCAL_DIR}/rtl/src | grep vhd]"

suppress_messages {LBR-30 LBR-31 LBR-40 LBR-41 LBR-72 LBR-77 LBR-162}
```

```
set_db hdl_track_filename_row_col true
set_db lp_power_unit mW
set_db init_lib_search_path $LIB_PATH
set_db script_search_path $TCL_PATH
set_db init_hdl_search_path $RTL_PATH
set_db error_on_lib_lef_pin_inconsistency true
set_db lp_insert_clock_gating true
set_db leakage_power_effort low
set_db delete_unloaded_insts false
set_db hdl_preserve_unused_registers true
set_db delete_unloaded_seqs false


#set_db library $LIB_LIST
read_libs $LIB_LIST

## PLE
#set_db lef_library $LEF_LIST
read_physical -lef $LEF_LIST

read_hdl $RTL_LIST

::legacy::set_attribute hdl_max_loop_limit 500000 /

elaborate $DESIGN

init_design

check_design -unresolved

report_ple > ple.rpt

read_sdc ../tcl/clock.sdc

set_db innovus_executable /lustre/cad/CADENCE_18/INNOVUS181/bin/innovus

set_db syn_generic_effort medium
set_db syn_map_effort medium
set_db syn_opt_effort medium

#Starting the synthesis with:

syn_generic
syn_map
syn_opt

report_qor > qor.txt

read_saif ../libraries/salida.saif

report_power > a_power.txt

report_area > a_area.txt

report_timing > a_timing.txt
```

```
#Generating extra reports
report_dp > a_dp.txt
report_design_rules > a_desrul.txt
report_gates > a_gates.txt
report_summary > a_summary.txt
report_instance > a_instance.txt
```

Apéndice D
Algunos reportes

Aquí se muestran algunos ejemplos más de reportes generados durante la *síntesis* de los circuitos propuestos. Algunos informes han sido explicados en el capítulo 4, área, potencia y tiempo, figuras D.1, D.2 y D.3 respectivamente.

El resto: *datapath*, figura D.4; comprobación de reglas de diseño, figura D.5; *quality of results*, figura D.6; puertas empleadas, figura D.7; y reporte *PLE*, figura D.8. Por razones de espacio, algunos de los reporte muestran solamente las primeras líneas del mismo, ya que algunos pueden ser de cientos o miles de líneas.

```
====================================================================
  Generated by:           Genus(TM) Synthesis Solution 18.10-p003_1
  Generated on:           Jan 31 2023  02:14:53 pm
  Module:                 cnn_sc_top
  Technology libraries:   tcbn40lpbwptc_ecsm 120
                          tcbn40lpbwptc_ccs 120
                          physical_cells
  Operating conditions:   NCCOM
  Interconnect mode:      global
  Area mode:              physical library
====================================================================
```

Instance	Module	Cell Count	Cell Area	Net Area	Total Area
cnn_sc_top		548996	1426279.201	491034.554	1917313.756
cnn_inst	cnn_sc	546469	1418085.774	489239.496	1907325.270
inst_feb_l1	feb_sc_GNRC_XSIDE12_GNRC_CIN6_GNRC_XSIDE5_GNRC_COU	247812	666501.469	225193.460	891694.929
Gen_CHOUT[11].Gen_X_I[0].Gen_X_J[2].inst_peb	peb_sc_v1_GNRC_XSIZE5_GNRC_CIN6_GNRC_N_W_PEB151_GN	989	2657.642	778.613	3436.256
inst_n01	neurona_sc_v2_2_NEURON_N_INPUTS150_NEURON_OUTPUT_S	245	660.618	159.818	820.436
apc_inst	apc_v2_2_N_INPUTS151_INTG_PRD_CLK3_N_OUTPUTS2_APC_	173	536.080	124.651	660.731
n_inputs_151_gen.pc_tree_61_inst_pc_2_inst_pc_up_inst_FA3_inst	full_adder_77278	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_2_inst_pc_up_inst_FA2_inst	full_adder_77277	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_2_inst_pc_up_inst_FA1_inst	full_adder_77276	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_2_inst_pc_up_inst_FA0_inst	full_adder_77275	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_2_inst_pc_dwn_inst_FA3_inst	full_adder_77282	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_2_inst_pc_dwn_inst_FA2_inst	full_adder_77281	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_2_inst_pc_dwn_inst_FA1_inst	full_adder_77280	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_2_inst_pc_dwn_inst_FA0_inst	full_adder_77279	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_2_inst_fa2_inst	full_adder_77285	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_2_inst_fa1_inst	full_adder_77284	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_2_inst_fa0_inst	full_adder_77283	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_pc_up_inst_FA3_inst	full_adder_77267	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_pc_up_inst_FA2_inst	full_adder_77266	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_pc_up_inst_FA1_inst	full_adder_77265	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_pc_up_inst_FA0_inst	full_adder_77264	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_pc_dwn_inst_FA3_inst	full_adder_77271	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_pc_dwn_inst_FA2_inst	full_adder_77270	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_pc_dwn_inst_FA1_inst	full_adder_77269	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_pc_dwn_inst_FA0_inst	full_adder_77268	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_fa2_inst	full_adder_77274	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_fa1_inst	full_adder_77273	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_fa0_inst	full_adder_77272	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_1_inst_pc_up_inst_FA3_inst	full_adder_77256	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_1_inst_pc_up_inst_FA2_inst	full_adder_77255	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_1_inst_pc_up_inst_FA1_inst	full_adder_77254	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_1_inst_pc_up_inst_FA0_inst	full_adder_77253	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_1_inst_pc_dwn_inst_FA3_inst	full_adder_77260	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_1_inst_pc_dwn_inst_FA2_inst	full_adder_77259	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_1_inst_pc_dwn_inst_FA1_inst	full_adder_77258	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_1_inst_pc_dwn_inst_FA0_inst	full_adder_77257	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_1_inst_fa2_inst	full_adder_77263	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_1_inst_fa1_inst	full_adder_77262	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_1_inst_fa0_inst	full_adder_77261	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_2_inst_pc_up_inst_FA3_inst	full_adder_77245	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_2_inst_pc_up_inst_FA2_inst	full_adder_77244	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_2_inst_pc_up_inst_FA1_inst	full_adder_77243	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_2_inst_pc_up_inst_FA0_inst	full_adder_77242	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_2_inst_pc_dwn_inst_FA3_inst	full_adder_77249	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_2_inst_pc_dwn_inst_FA2_inst	full_adder_77248	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_2_inst_pc_dwn_inst_FA1_inst	full_adder_77247	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_2_inst_pc_dwn_inst_FA0_inst	full_adder_77246	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_2_inst_fa2_inst	full_adder_77252	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_2_inst_fa1_inst	full_adder_77251	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_2_inst_fa0_inst	full_adder_77250	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_1_inst_pc_up_inst_FA3_inst	full_adder_77234	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_1_inst_pc_up_inst_FA2_inst	full_adder_77233	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_1_inst_pc_up_inst_FA1_inst	full_adder_77232	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_1_inst_pc_up_inst_FA0_inst	full_adder_77231	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_1_inst_pc_dwn_inst_FA3_inst	full_adder_77238	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_1_inst_pc_dwn_inst_FA2_inst	full_adder_77237	1	4.410	0.000	4.410
n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_1_inst_pc_1_inst_pc_dwn_inst_FA1_inst	full_adder_77236	1	4.410	0.000	4.410

Figura D.1 Reporte de área generado después del proceso de síntesis.

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 18.10-p003_1
  Generated on:           Jan 31 2023  02:09:45 pm
  Module:                 cnn_sc_top
  Technology libraries:   tcbn40lpbwptc_ecsm 120
                          tcbn40lpbwptc_ccs 120
                          physical_cells
  Operating conditions:   NCCOM
  Interconnect mode:      global
  Area mode:              physical library
============================================================

                                          Leakage   Dynamic    Total
             Instance              Cells Power(mW) Power(mW) Power(mW)
-------------------------------------------------------------------------------
cnn_sc_top                        548996     0.172   203.572   203.744
  cnn_inst                        546469     0.171   200.418   200.589
    inst_feb_l1                   247812     0.083    83.063    83.146
      Gen_CHOUT[..J[0].inst_peb      987     0.000     0.333     0.334
        inst_n00                     243     0.000     0.078     0.078
          apc_inst                   171     0.000     0.072     0.072
            n_inputs_1..inst_fa0_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_fa1_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_fa2_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA0_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA1_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA2_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA3_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA0_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA1_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA2_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA3_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_fa0_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_fa1_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_fa2_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA0_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA1_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA2_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA3_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA0_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA1_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA2_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA3_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_fa0_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_fa1_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_fa2_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA0_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA1_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA2_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA3_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA0_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA1_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA2_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA3_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_fa0_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_fa1_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_fa2_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA0_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA1_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA2_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA3_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA0_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA1_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA2_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA3_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_fa0_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_fa1_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_fa2_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA0_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA1_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA2_inst  1     0.000     0.001     0.001
            n_inputs_1..inst_FA3_inst  1     0.000     0.000     0.000
            n_inputs_1..inst_FA0_inst  1     0.000     0.001     0.001
```

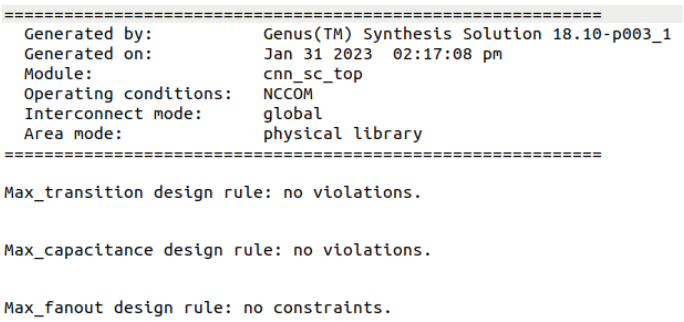
Figura D.2 Reporte de potencia generado después del proceso de síntesis.

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 18.10-p003_1
  Generated on:           Jan 31 2023  02:17:06 pm
  Module:                 cnn_sc_top
  Operating conditions:   NCCOM
  Interconnect mode:      global
  Area mode:              physical library
============================================================


Path 1: MET (1826 ps) Setup Check with Pin cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/apc_out_reg_reg[0]/CP->D
          Group: clk
     Startpoint: (R) cnn_inst/inst_lfsrIn_lfsr_reg[2]/CP
          Clock: (R) clk
       Endpoint: (F) cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/apc_out_reg_reg[0]/D
          Clock: (R) clk

                     Capture       Launch
        Clock Edge:+    6000            0
       Src Latency:+       0            0
       Net Latency:+       0 (I)        0 (I)
           Arrival:=    6000            0

             Setup:-      87
     Required Time:=    5913
      Launch Clock:-       0
         Data Path:-    4087
             Slack:=    1826

#-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
#                                                                                     Timing Point                                                                        Flags    Arc   Edge    Cell       Fanout Load Trans Delay Arrival Instance
#                                                                                                                                                                                                          (fF)  (ps)  (ps)   (ps)  Location
#-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
  cnn_inst/inst_lfsrIn_lfsr_reg[2]/CP                                                                                                                                       -       -      R     (arrival)   21773    -     0     -       0    (-,-)
  cnn_inst/inst_lfsrIn_lfsr_reg[2]/Q                                                                                                                                        -       CP->Q  R     EDFQD4BWP      13 16.0    55   114     114    (-,-)
  cnn_inst/inst_feb_l0/g5859/Z                                                                                                                                              -       I->Z   R     CKBD0BWP        3  3.5    78    80     195    (-,-)
  cnn_inst/inst_feb_l0/g5858/ZN                                                                                                                                             -       I->ZN  F     CKND0BWP        3  5.3    92    81     276    (-,-)
  cnn_inst/inst_feb_l0/g5857/ZN                                                                                                                                             -       I->ZN  R     CKND1BWP       21 22.8   270   184     460    (-,-)
  cnn_inst/inst_feb_l0/g5856/ZN                                                                                                                                             -       I->ZN  F     CKND0BWP        2  2.5    98   101     562    (-,-)
  cnn_inst/inst_feb_l0/g5855/ZN                                                                                                                                             -       I->ZN  R     CKND0BWP       13 14.3   300   205     766    (-,-)
  cnn_inst/inst_feb_l0/g5854/Z                                                                                                                                              -       I->Z   R     CKBD0BWP        5  5.8   124   150     917    (-,-)
  cnn_inst/inst_feb_l0/g5853/Z                                                                                                                                              -       I->Z   R     CKBD0BWP        1  1.6    42    78     994    (-,-)
  cnn_inst/inst_feb_l0/g6251/Z                                                                                                                                              -       I->Z   R     CKBD0BWP        4  4.8   102    90    1084    (-,-)
  cnn_inst/inst_feb_l0/Gen_CHOUT[3].Gen_X_I[6].Gen_X_J[6].inst_peb/inst_n01/g165/Z                                                                                          -       B->Z   R     AO21D0BWP       1  1.8    49    84    1168    (-,-)
  cnn_inst/inst_feb_l0/Gen_CHOUT[3].Gen_X_I[6].Gen_X_J[6].inst_peb/g28/Z                                                                                                    -       A1->Z  R     OR4D1BWP        1  3.3    49    79    1247    (-,-)
  cnn_inst/inst_feb_l0/Gen_CHOUT[3].Gen_X_I[6].Gen_X_J[6].inst_peb/drc_bufs/Z                                                                                               -       I->Z   R     CKBD12BWP      39 44.8    53    68    1315    (-,-)
  cnn_inst/inst_feb_l1/g1763/Z                                                                                                                                              -       I->Z   R     CKBD0BWP        7  8.4   172   130    1446    (-,-)
  cnn_inst/inst_feb_l1/g1762/Z                                                                                                                                              -       I->Z   R     CKBD2BWP       42 53.0   308   229    1675    (-,-)
  cnn_inst/inst_feb_l1/g1761/Z                                                                                                                                              -       I->Z   R     CKBD0BWP        8 10.6   216   201    1875    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/g1357/ZN                                                                                        -       A1->ZN F     XNR2D0BWP       1  1.6    41   130    2005    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/g2002/Z                                                                                -       I1->Z  F     MUX2D0BWP       1  1.8    49    94    2099    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_pc_dwn_inst_FA0_inst/g74/S -       CI->S  F     FA1D0BWP        1  1.8    41   103    2201    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_pc_dwn_inst_FA3_inst/g86/CO -      A->CO  F     FA1D0BWP        1  1.8    40   167    2368    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_pc_dwn_inst_FA2_inst/g74/S -       CI->S  R     FA1D0BWP        1  1.8    50   109    2477    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/n_inputs_151_gen.pc_tree_61_inst_pc_1_inst_pc_2_inst_pc_2_inst_fa1_inst/g74/S              -       A->S   F     FA1D0BWP        1  1.8    41   162    2639    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/csa_tree_add_242_33_groupi_g564/CO                                                     -       CI->CO F     FA1D0BWP        1  2.4    47   102    2741    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/csa_tree_add_242_33_groupi_g558/S                                                      -       B->S   R     FA1D0BWP        1  2.4    62   149    2890    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/g2058/S                                                                                -       D->S   F     CMPE42D1BWP     1  1.8    33   207    3097    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/csa_tree_add_242_33_groupi_g549/CO                                                     -       CI->CO F     FA1D0BWP        1  2.4    47    98    3196    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/g2/CO                                                                                  -       D->CO  F     CMPE42D1BWP     1  1.8    33   128    3324    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/csa_tree_add_242_33_groupi_g547/CO                                                     -       CI->CO F     FA1D0BWP        1  1.8    40    93    3417    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/csa_tree_add_242_33_groupi_g546/CO                                                     -       CI->CO F     FA1D0BWP        1  1.8    40    96    3513    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/csa_tree_add_242_33_groupi_g545/CO                                                     -       CI->CO F     FA1D0BWP        1  3.0    54   107    3620    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/csa_tree_add_242_33_groupi_g544/S                                                      -       B->S   R     HA1D0BWP        2  2.7    67   101    3721    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/g819/ZN                                                                                -       A1->ZN F     ND3D0BWP        2  2.5   118    94    3814    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/g806/ZN                                                                                -       A1->ZN F     IAO21D0BWP      1  2.1    49   104    3919    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/g805/Z                                                                                 -       A1->Z  R     XOR2D0BWP       2  2.7    66   102    4020    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/g804/ZN                                                                                -       B->ZN  F     OAI21D0BWP      1  1.7    68    67    4087    (-,-)
  cnn_inst/inst_feb_l1/Gen_CHOUT[0].Gen_X_I[2].Gen_X_J[2].inst_peb/inst_n11/apc_inst/apc_out_reg_reg[0]/D                                                                   <<<     -      F     EDFQD1BWP       1    -     -     0    4087    (-,-)
#-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Figura D.3 Reporte de *timing* generado después del proceso de síntesis.

```
Command: report datapath > a_dp.txt
============================================================
  Generated by:           Genus(TM) Synthesis Solution 18.10-p003_1
  Generated on:           Jan 31 2023  02:17:06 pm
  Module:                 cnn_sc_top
  Operating conditions:   NCCOM
  Interconnect mode:      global
  Area mode:              physical library
============================================================


      Type        CellArea  Percentage
--------------------------------------
datapath modules      0.00        0.00
external muxes        0.00        0.00
others          1426279.20      100.00
--------------------------------------
total           1426279.20      100.00
```

Figura D.4 Reporte de *datapath* generado después del proceso de síntesis.



```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 18.10-p003_1
  Generated on:           Jan 31 2023  02:17:08 pm
  Module:                 cnn_sc_top
  Operating conditions:   NCCOM
  Interconnect mode:      global
  Area mode:              physical library
============================================================

Max_transition design rule: no violations.


Max_capacitance design rule: no violations.


Max_fanout design rule: no constraints.
```

Figura D.5 Reporte de comprobación reglas de diseño generado después del proceso de síntesis.

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 18.10-p003_1
  Generated on:           Jan 31 2023  01:58:23 pm
  Module:                 cnn_sc_top
  Technology libraries:   tcbn40lpbwptc_ecsm 120
                          tcbn40lpbwptc_ccs 120
                          physical_cells
  Operating conditions:   NCCOM
  Interconnect mode:      global
  Area mode:              physical library
============================================================

Timing
--------

Clock Period
-------------
clk   6000.0


        Cost          Critical         Violating
       Group         Path Slack  TNS     Paths
-------------------------------------------------
cg_enable_group_clk      3032.7   0.0          0
clk                      1825.7   0.0          0
default                 No paths   0.0
-------------------------------------------------
Total                              0.0          0

Instance Count
--------------
Leaf Instance Count             548996
Physical Instance count              0
Sequential Instance Count        51029
Combinational Instance Count    497967
Hierarchical Instance Count     119908

Area
----
Cell Area                          1426279.201
Physical Cell Area                 0.000
Total Cell Area (Cell+Physical)    1426279.201
Net Area                           491034.555
Total Area (Cell+Physical+Net)     1917313.756

Power
-----
Leakage Power                      0.172 mW
Dynamic Power                      203.572 mW
Total Power                        203.744 mW

Number of Clock Gating Logic       3467
Max Fanout                         21773 (clk)
Min Fanout                         0 (cnn_i_x[0][0])
Average Fanout                     1.9
Terms to net ratio                 2.9022
Terms to instance ratio            3.9077
Runtime                            12451.907618 seconds
Elapsed Runtime                    15668 seconds
Genus peak memory usage            74380.42
Innovus peak memory usage          no_value
Hostname                           nodo35
```

Figura D.6 Reporte de calidad de resultados generado después del proceso de síntesis. Este reporte muestra un resumen del proceso completo con información relativa a la *síntesis* y a la misma ejecución del proceso.

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 18.10-p003_1
  Generated on:           Jan 31 2023  02:17:31 pm
  Module:                 cnn_sc_top
  Operating conditions:   NCCOM
  Interconnect mode:      global
  Area mode:              physical library
============================================================


    Gate      Instances     Area            Library
-------------------------------------------------------------
AN2D2BWP             97      119.776    tcbn40lpbwptc_ecsm
AN2D4BWP             17       35.986    tcbn40lpbwptc_ecsm
AN3D0BWP             11       13.583    tcbn40lpbwptc_ecsm
AO21D0BWP          4382     5410.894    tcbn40lpbwptc_ecsm
AO21D1BWP            15       18.522    tcbn40lpbwptc_ecsm
AOI21D0BWP         3466     3668.414    tcbn40lpbwptc_ecsm
AOI31D0BWP          120      148.176    tcbn40lpbwptc_ecsm
BUFFD0BWP             2        1.411    tcbn40lpbwptc_ecsm
BUFFD12BWP           55      213.444    tcbn40lpbwptc_ecsm
BUFFD1BWP          3497     2467.483    tcbn40lpbwptc_ecsm
BUFFD2BWP            16       16.934    tcbn40lpbwptc_ecsm
BUFFD3BWP           642      792.742    tcbn40lpbwptc_ecsm
BUFFD4BWP          1151     1827.328    tcbn40lpbwptc_ecsm
BUFFD6BWP           556     1176.941    tcbn40lpbwptc_ecsm
CKAN2D0BWP            6        6.350    tcbn40lpbwptc_ecsm
CKAN2D1BWP           32       33.869    tcbn40lpbwptc_ecsm
CKAN2D4BWP           30       47.628    tcbn40lpbwptc_ecsm
CKAN2D8BWP          155      410.130    tcbn40lpbwptc_ecsm
CKBD0BWP           2066     1457.770    tcbn40lpbwptc_ecsm
CKBD12BWP           137      507.503    tcbn40lpbwptc_ecsm
CKBD16BWP            14       66.679    tcbn40lpbwptc_ecsm
CKBD1BWP             98       69.149    tcbn40lpbwptc_ecsm
CKBD2BWP           2851     3017.498    tcbn40lpbwptc_ecsm
CKBD3BWP            538      664.322    tcbn40lpbwptc_ecsm
CKBD4BWP           1267     2011.489    tcbn40lpbwptc_ecsm
CKBD6BWP             21       44.453    tcbn40lpbwptc_ecsm
CKBD8BWP             27       76.205    tcbn40lpbwptc_ecsm
CKLNQD1BWP         3467    11619.997    tcbn40lpbwptc_ecsm
CKND0BWP          12542     6637.226    tcbn40lpbwptc_ecsm
CKND12BWP            33       98.960    tcbn40lpbwptc_ecsm
CKND16BWP            17       65.974    tcbn40lpbwptc_ecsm
CKND1BWP            296      156.643    tcbn40lpbwptc_ecsm
CKND2BWP            140       98.784    tcbn40lpbwptc_ecsm
CKND2D0BWP         3812     2689.747    tcbn40lpbwptc_ecsm
CKND2D3BWP            1        1.588    tcbn40lpbwptc_ecsm
CKND3BWP           1866     1974.974    tcbn40lpbwptc_ecsm
CKND4BWP           1102     1360.750    tcbn40lpbwptc_ecsm
CKND6BWP             13       20.639    tcbn40lpbwptc_ecsm
CKND8BWP             13       27.518    tcbn40lpbwptc_ecsm
CMPE42D1BWP        5460    47194.056    tcbn40lpbwptc_ecsm
DCCKBD12BWP          12       88.906    tcbn40lpbwptc_ecsm
DCCKND12BWP           2       11.995    tcbn40lpbwptc_ecsm
DCCKND4BWP           27       66.679    tcbn40lpbwptc_ecsm
DEL025D1BWP          28       24.696    tcbn40lpbwptc_ecsm
DEL050D1BWP           7        7.409    tcbn40lpbwptc_ecsm
DFKCNQD1BWP       36595   142017.876    tcbn40lpbwptc_ecsm
DFQD1BWP              7       24.696    tcbn40lpbwptc_ecsm
DFQD2BWP              2        7.762    tcbn40lpbwptc_ecsm
EDFKCNQD1BWP          1        5.292    tcbn40lpbwptc_ecsm
EDFQD1BWP         10956    52181.237    tcbn40lpbwptc_ecsm
EDFQD4BWP             1        5.645    tcbn40lpbwptc_ecsm
FA1D0BWP         128112   564973.920    tcbn40lpbwptc_ecsm
HA1D0BWP          20410    61205.508    tcbn40lpbwptc_ecsm
IAO21D0BWP         1108     1368.158    tcbn40lpbwptc_ecsm
IND2D0BWP          3477     3680.057    tcbn40lpbwptc_ecsm
IND2D1BWP            10       10.584    tcbn40lpbwptc_ecsm
```

Figura D.7 Reporte de puertas usadas generado después del proceso de síntesis. Se muestra una lista de los dispositivos lógicos incluidos en las librerías físicas del fabricante así como el número de veces que se instancia, el área que ocupan y la librería de donde proceden.

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 18.10-p003_1
  Generated on:           Jan 31 2023  11:56:25 am
  Module:                 cnn_sc_top
  Technology libraries:   tcbn40lpbwptc_ecsm 120
                          tcbn40lpbwptc_ccs 120
                          physical_cells
  Operating conditions:   NCCOM
  Interconnect mode:      global
  Area mode:              physical library
============================================================

Aspect ratio        : 1.00
Shrink factor       : 1.00
Scale of res/length : 1.00
Scale of cap/length : 1.00
Net derating factor : 1.00
Thermal factor      : 1.00
Via Resistance      : 4.79 ohm (from lef_library)
Site size           : 1.40 um (from lef [tech+cell])

                                  Capacitance
  Layer                            / Length         Data source:
Name        Direction Utilization  (pF/micron)      lef_library
-------------------------------------------------
M1              V         0.00        0.000178
M2              H         1.00        0.000170
M3              V         1.00        0.000170
M4              H         1.00        0.000170
M5              V         1.00        0.000170
M6              H         1.00        0.000207
M7              V         1.00        0.000212
AP              H         1.00        0.000241

                                  Resistance
  Layer                            / Length         Data source:
Name        Direction Utilization  (ohm/micron)     lef_library
-------------------------------------------------
M1              V         0.00          4.400000
M2              H         1.00          3.971429
M3              V         1.00          3.971429
M4              H         1.00          3.971429
M5              V         1.00          3.971429
M6              H         1.00          0.055000
M7              V         1.00          0.055000
AP              H         1.00          0.010500

                                    Area
  Layer                            / Length         Data source:
Name        Direction Utilization    (micron)       lef_library
-------------------------------------------------
M1              V         0.00          0.070000
M2              H         1.00          0.070000
M3              V         1.00          0.070000
M4              H         1.00          0.070000
M5              V         1.00          0.070000
M6              H         1.00          0.400000
M7              V         1.00          0.400000
AP              H         1.00          2.000000
```

Figura D.8 Reporte PLE generado antes del proceso de síntesis. Este informe previo muestra información relativa a las librerías del fabricante y las capas de metal que serán utilizadas.

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] J. Ahmadi-Farsani, B. Linares-Barranco, and T. Serrano-Gotarredona, *Auxiliary pulse-extender and current-attenuator circuits for flexible interaction with memristive crossbars in snns*, 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2020, pp. 1–4.

[2] J. Ahmadi-Farsani, S. Ricci, S. Hashemkhani, D. Ielmini, B. Linares-Barranco, and T. Serrano-Gotarredona, *A cmos-memristor hybrid system for implementing stochastic binary spike timing-dependent plasticity*, Phil. Trans. R. Soc. A **380** (2022), 20210018.

[3] J. Ahmadi-Farsani, S. Ricci, S. Hashemkhani, D. Ielmini, B. Linares-Barranco, and T. Serrano-Gotarredona, *A hybrid memristor/cmos snn for implementing one-shot winner-takes-all training*, 2022 IEEE International Symposium on Circuits and Systems (ISCAS), 2022, pp. 210–214.

[4] T. H. Bullock, M. V. L. Bennett, D. Johnston, R. Josephson, E. Marder, and R. D. Fields, *The neuron doctrine, redux*, Science **310** (2005), no. 5749, 791–793.

[5] A. Camuñas-Mesa, L, E. Vianello, C. Reita, T. Serrano-Gotarredona, and B. Linares-Barranco, *A cmol-like memristor-cmos neuromorphic chip-core demonstrating stochastic binary stdp*, IEEE Journal on Emerging and Selected Topics in Circuits and Systems **12** (2022), no. 4, 898–912.

[6] L. A. Camuñas-Mesa, B. Linares-Barranco, and T. Serrano-Gotarredona, *Neuromorphic spiking neural networks and their memristor-cmos hardware implementations*, Materials **12** (2019), no. 17, 10–37.

[7] V. Canals, A. Morro, A. Oliver, M. L. Alomar, and J. L. Rosselló, *A new stochastic computing methodology for efficient neural network implementation*, IEEE Transactions on Neural Networks and Learning Systems **27** (2016), no. 3, 551–564.

[8] C.Y. Chang and S.M. Sze, *Ulsi technology*, Electrical engineering series, McGraw-Hill, 1996.

[9] Y. Chen, Z. Shu, S. Zhang, P. Zeng, H. Liang, M. Zheng, and H. Duan, *Sub-10 nm fabrication: methods and applications*, International Journal of Extreme Manufacturing **3** (2021), no. 3, 032002.

[10] P.P. Chu, *Fpga prototyping by vhdl examples: Xilinx microblaze mcs soc*, Wiley, 2018.

[11] L. Chua, *Memristor, hodgkin–huxley, and edge of chaos*, Nanotechnology **24** (2013), no. 38, 383001.

[12] Digital System Design and S. Roy, *Placement and routing using innovus*, *https://digitalsystemdesign.in/placement-and-routing-using-innovus/*, 2020, Visitado en 2023.

[13] S. A. El-Sayed, T. Spyrou, A. Pavlidis, E. Afacan, L. A. Camuñas-Mesa, B. Linares-Barranco, and H.-G. Stratigopoulos, *Spiking neuron hardware-level fault modeling*, 2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS), 2020, pp. 1–4.

[14] J. W. Esch, *— rascel — a programmable analog computer based on a regular array of stochastic computing element logic*, Ph.D. thesis, University of Illinois Urbana-Champaign, 1969, p. 108.

[15] X. Fan and H. Markram, *A brief history of simulation neuroscience*, Frontiers in Neuroinformatics **13** (2019), 32.

[16] C. F. Frasser, P. Linares-Serrano, I. Díez de los Ríos, A. Morán, E. S. Skibinsky-Gitlin, J. Font-Rosselló, V. Canals, M. Roca, T. Serrano-Gotarredona, and J. L. Rosselló, *Fully parallel stochastic computing hardware implementation of convolutional neural networks for edge computing applications*, IEEE Transactions on Neural Networks and Learning Systems (2022), 1–11.

[17] J. E. Gentle, *Random number generation and monte carlo methods*, Statistics and Computing, Springer New York, NY, 2003.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial networks*, Commun. ACM **63** (2020), no. 11, 139–144.

[19] A. Graves, A.-R. Mohamed, and G. Hinton, *Speech recognition with deep recurrent neural networks*, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6645–6649.

[20] S Hochreiter and J Schmidhuber, *Long short-term memory*, Neural Computation **9** (1997), no. 8, 1735–1780.

[21] G. Indiveri, B. Linares-Barranco, T. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. SAÏGHI, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, *Neuromorphic silicon neuron circuits*, Frontiers in Neuroscience **5** (2011), 23.

[22] C.L. Janer, J.M. Quero, and L.G. Franquelo, *Fully parallel summation in a new stochastic neural network architecture*, IEEE International Conference on Neural Networks, 1993, pp. 1498–1503 vol.3.

[23] C.L. Janer, J.M. Quero, J.G. Ortega, and L.G. Franquelo, *Fully parallel stochastic computation architecture*, IEEE Transactions on Signal Processing **44** (1996), no. 8, 2110–2117.

[24] S.M. Kang and Y. Leblebici, *Cmos digital integrated circuits analysis & design*, McGraw-Hill Series in Electrical and Computer Engineering Series, McGraw-Hill Education, 2003.

[25] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature **521** (2015), 436–444.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), no. 11, 2278–2324.

[27] Yann LeCun, *1.1 deep learning hardware: Past, present, and future*, 2019 IEEE International Solid-State Circuits Conference - (ISSCC), 2019, pp. 12–19.

[28] P. Mars and W. J. Poppelbaum, *Stochastic and deterministic averaging processors / [by] p. mars [and] w. j. poppelbaum.*, Institute of Electrical Engineers digital electronics and computing series ; 1, Peregrinus on behalf of the Institution of Electrical Engineers, New York ;, 1981 (eng).

[29] Nandland, *Vhdl vs. verilog. which language should you use for your fpga and asic designs?*, https://nandland.com/lesson-16-vhdl-vs-verilog-which-language-should-you-learn-first/, 2022, Visitado en 2023.

[30] OpenAI, *Chatgpt*, https://chat.openai.com/, 2023, Visitado entre noviembre 2022 y enero 2023.

[31] ———, *Dall·e*, https://labs.openai.com/, 2023, Visitado entre noviembre 2022 y enero 2023.

[32] J.D. Plummer, M.D. Deal, and P.B. Griffin, *Silicon vlsi technology: Fundamentals, practice and modeling*, Prentice Hall electronics and VLSI series, Prentice Hall, 2000.

[33] W. J. Poppelbaum, C. Afuso, and J. W. Esch, *Stochastic computing elements and systems*, Proceedings of the November 14-16, 1967, Fall Joint Computer Conference (New York, NY, USA), AFIPS '67 (Fall), Association for Computing Machinery, 1967, p. 635–644.

[34] B. Razavi, *Design of analog cmos integrated circuits*, Electrical Engineering Series, McGraw-Hill, 2001.

[35] B.C. Readler, *Verilog by example: A concise introduction for fpga design*, Full Arc Press, 2011.

[36] ———, *Vhdl by example: A concise introduction for fpga design*, Full Arc Press, 2014.

[37] F. Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain.*, Psychological review **65 6** (1958), 386–408.

[38] C. Saint and J. Saint, *Ic mask design: Essential layout techniques*, McGraw-Hill professional engineering, McGraw-Hill, 2002.

[39] J. Schmidhuber, *Deep learning in neural networks: An overview*, Neural Networks **61** (2015), 85–117.

[40] K. Shubham and A. Gupta, *Integrated circuit fabrication*, Manakin Press, 2021.

[41] T. Spyrou, S. A. El-Sayed, E. Afacan, L. A. Camuñas-Mesa, B. Linares-Barranco, and H.-G. Stratigopoulos, *Neuron fault tolerance in spiking neural networks*, 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2021, pp. 743–748.

[42] StabilityAI, *Stable diffusion online*, *https://stablediffusionweb.com/#demo*, 2023, Visitado entre noviembre 2022 y enero 2023.

[43] E. Strubell, A. Ganesh, and A. McCallum, *Energy and policy considerations for deep learning in NLP*, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Florence, Italy), Association for Computational Linguistics, July 2019, pp. 3645–3650.

[44] _____, *Energy and policy considerations for modern deep learning research*, Proceedings of the AAAI Conference on Artificial Intelligence **34** (2020), no. 09, 13693–13696.

[45] Cadence Design Systems, *Advanced synthesis with genus stylus common ui training*, *https://www.cadence.com/en_US/home/training/all-courses/86249.html*, 2021-2022.

[46] _____, *Cadence rtl-to-gdsii flow training*, *https://www.cadence.com/en_US/home/training/all-courses/86136.html*, 2021-2022.

[47] _____, *Genus synthesis solution with stylus common ui training*, *https://www.cadence.com/en_US/home/training/all-courses/86220.html*, 2021-2022.

[48] _____, *Innovus block implementation with stylus common ui training*, *https://www.cadence.com/en_US/home/training/all-courses/86222.html*, 2021-2022.

[49] S.M. Sze, *Vlsi technology*, Electronics and electronic circuits, McGraw-Hill, 1983.

[50] R. C. Tausworthe, *Random numbers generated by linear recurrence modulo two*, Mathematics of Computation **19** (1965), no. 90, 201–209.

[51] A. Torralba, F. Colodro, E. Ibanez, and L.G. Franquelo, *Two digital circuits for a fully parallel stochastic neural network*, IEEE Transactions on Neural Networks **6** (1995), no. 5, 1264–1268.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, Advances in Neural Information Processing Systems (I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[53] VHDLwhiz, *Should i learn vhdl if verilog is becoming more popular?*, *https://vhdlwhiz.com/should-i-learn-vhdl-if-verilog-is-becoming-more-popular/*, 2022, Visitado en 2023.

[54] J. von Neumann, *Probabilistic logics and the synthesis of reliable organisms from unreliable components*, pp. 43–98, Princeton University Press, Princeton, 1956.

[55] W. Wan, R. Kubendran, C. Schaefer, S. B. Erylmaz, W. Zhang, D. Wu, S. Deiss, P. Raina, H. Qian, B. Gao, S. Joshi, H. Wu, H.-S. Philip Wonw, and G. Cauwenberghs, *A compute-in-memory chip based on resistive random-access memory*, Nature **608** (2022), 504–512.

[56] N.H.E. Weste and D.M. Harris, *Cmos vlsi design: A circuits and systems perspective*, Addison Wesley, 2011.

[57] Wikipedia, *Semiconductor device fabrication*, *https://en.wikipedia.org/wiki/Semiconductor_device_fabrication*, 2023, Visitado en febrero de 2023.

[58] B. Yu, L. Chang, S. Ahmed, H. Wang, S. Bell, C. Y. Yang, C. Tabery, C. Ho, Q. Xiang, T.J. King, J. Bokor, C. Hu, M. R. Lin, and D. Kyser, *Finfet scaling to 10 nm gate length*, Digest. International Electron Devices Meeting,, 2002, pp. 251–254.

[59] C. Zamarreño-Ramos, L. Camuñas-Mesa, J. Perez-Carrasco, T. Masquelier, T. Serrano-Gotarredona, and B. Linares-Barranco, *On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex*, Frontiers in Neuroscience **5** (2011), 22.

Índice alfabético

Glosario

AI *Artificial Intelligence*. 1, 2, 4, 7, 10, 31, 59

ANN *Artificial Neural Network*. 2, 3, 7

ASIC *Application-Specific Integrated Circuit*. 11, 26, 41

CDS *Cadence Design Systems*. 20–22, 24, 41, 47

CMOS *Complementary Metal-Oxide-Semiconductor*. 15–17

CNN *Convolutional Neural Network*. 4–8, 19–21, 26–30, 32, 33, 38, 41, 60

CSIC *Consejo Superior de Investigaciones Científicas*. 19

DRC *Design Rule Checking*. 16

FC *Fully Connected*. 5, 19, 20, 27, 30, 33, 35–37, 59, 60

FPGA *Field-Programmable Gate Array*. 7, 23, 42, 43

HDL *Hardware Description Language*. 11–13, 22

IC *Integrated Circuit*. 7, 10–12, 14–19, 21, 24, 41, 42, 60

IMSE *Instituo de Microelectrónica de Sevilla*. 9, 19, 21, 29, 42, 43

IP *Intellectual Property*. 13, 21

LFSR *Linear-Feedback Shift Register*. 9

LVS *Layout vs Schematic*. 16

ML *Machine Learning*. 2–4, 10, 31

MMMC *Multi-Mode Multi-Corner*. 31

MNIST *Modified National Institute of Standards and Technology*. 5, 6, 59

MP *Max-Pooling*. 5

PCB *Printed Circuit Board*. 17

PDK *Process Design Kit*. 12, 13, 15, 19, 22

RRAM *Resistive Random-Access Memory*. 42

RTL *Register-Transfer Level*. 12, 21, 24

ReLU *Rectified Linear Unit*. 2, 3

SC *Stochastic Computing*. 8, 19, 31, 32

SNN *Spiking Neural Network*. 2, 42

TNS *Total Negative Slack*. 26

US *Universidad de Sevilla*. 9, 19, 42

VHDL *VHSIC Hardware Description Language*. 11, 21, 23

VLSI *Very Large-Scale Integration*. 7, 8, 23, 41, 43

Verilog *Verilog*. 11, 23

WNS *Worse Negative Slack*. 26, 28, 31, 60