Diseño de un circuito integrado con tecnología de 65 nm utilizando librerías de diseño de TSMC: pruebas de LVS, ERC y extracción de parásitos

Diana Sofía Alvarado Mota

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un circuito integrado con tecnología de 65 nm utilizando librerías de diseño de TSMC: pruebas de LVS, ERC y extracción de parásitos

Trabajo de graduación presentado por Diana Sofía Alvarado Mota para optar al grado académico de Licenciada en Ingeniería Electrónica

Guatemala,

2025

Vo.Bo.:

(f) 

M. Sc. Carlos Esquit

Tribunal Examinador:

(f) 

M.Sc. Carlos Esquit

(f) 

M. Sc. Miguel Enrique Zea Arenales

(f) 

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

Este trabajo de graduación está dedicado a mi familia, en especial a mi madre, cuyo esfuerzo ha sido el pilar de mi camino, y a mi hermano, por su constante apoyo. Agradezco profundamente a mis amigos y compañeros de equipo por su colaboración y por los momentos inolvidables que compartimos.

Asimismo, extiendo mi gratitud al Msc. Jonathan de los Santos, cuya mentoría ha sido invaluable a lo largo de este año, y al Msc. Carlos Esquit, por compartir su vasto conocimiento sobre nanoelectrónica e inspirar a las nuevas generaciones de ingenieros electrónicos en esta fascinante área.

Finalmente, agradezco a la Fundación Juan Bautista Gutiérrez por brindarme la oportunidad de continuar mis estudios superiores en la Universidad del Valle de Guatemala.

Índice

Lista de figuras

Lista de cuadros

Resumen

En el presente trabajo, se abordó el proceso de la verificación de reglas eléctricas y las pruebas de *layout versus schematic* (ERC y LVS), fundamentales en el diseño de *layout* de circuitos integrados. Este flujo de diseño tuvo como propósito la mejora de los procesos desarrollados en iteraciones anteriores, abarcando desde el posicionamiento e interconexión de componentes hasta la corrección de errores encontrados durante las verificaciones, con el fin de lograr un diseño más eficiente y fluido para futuros proyectos de circuitos personalizados.

Para las pruebas se utilizaron herramientas de Synopsys, como *IC Compiler II*, que permitió la síntesis de *layouts* en silicio, y *IC Validator*, que se empleó para verificar que los resultados fueran fabricables. Se realizaron diversas pruebas en circuitos básicos como un *NOT* y circuitos de mayor complejidad como un *carry look ahead*, un contador de 16 bits y el circuito principal diseñado por el grupo de trabajo. Los resultados fueron satisfactorios, cumpliendo todos los objetivos propuestos.

En cuanto a la verificación física mediante *layout versus schematic* (LVS), este trabajo definió un flujo eficiente y actualizado, adaptando las nuevas versiones de las herramientas como *IC Validator*. Siguiendo guías de investigaciones previas y la *user guide* de TSMC, se logró ajustar archivos de versiones anteriores a los formatos actuales, lo que permitió ejecutar el LVS en circuitos desde una compuerta *NOT* hasta circuitos más complejos, como una ALU. Además, se realizó una extracción parásita de resistencias utilizando *StarRC*, lo cual permitió identificar las resistencias parásitas en el diseño y evaluar su impacto en el rendimiento del circuito, sin que la complejidad del *deck* fuera demasiado elevada para identificar las conexiones.

En trabajos futuros, se plantea la automatización del flujo de verificación mediante el desarrollo de *scripts* o herramientas personalizadas que permitan automatizar ciertas etapas del proceso, reduciendo la intervención manual y optimizando los tiempos de prueba. Además, se propone la automatización de la creación de celdas para el mapeo en los *decks* de extracción parásita, lo cual mejoraría la eficiencia en la asignación de celdas y permitiría una mayor precisión. Este enfoque contribuiría a una metodología más eficiente, con menor probabilidad de errores humanos.

Palabras Clave: nanoelectrónica, LVS, ERC, extracción de parásitos, HSPICE.

Abstract

In this work, the process of performing electrical rule checking (ERC) and layout versus schematic (LVS) testing, which are essential in the layout design of integrated circuits, has been systematically addressed. This design flow builds on improvements from previous iterations, covering everything from the placement and interconnection of components to error correction during verification, aiming to support more efficient custom circuit designs in future projects.

To conduct the tests, Synopsys tools, including IC Compiler II for silicon layout synthesis and IC Validator for manufacturability verification, were utilized. A series of tests were carried out on both basic circuits like a NOT gate and on more complex designs, such as a carry look-ahead, a 16-bit counter, and the main circuit developed by the team. The results were positive, meeting all established objectives.

For physical verification through layout versus schematic (LVS), an efficient and updated flow was established by adapting new versions of tools such as IC Validator. Using insights from prior studies and the TSMC User Guide, we adjusted older files to align with current formats, facilitating LVS testing on a range of circuits from a single NOT gate to more complex designs, such as an ALU. Additionally, parasitic resistance extraction was conducted with StarRC, allowing for the identification of parasitic resistances and assessment of their impact on circuit performance without over-complicating the deck for connection identification

For future development, it is proposed to automate the verification process through custom scripts or tools to reduce manual tasks and improve testing efficiency. Another suggestion includes automating cell creation for mapping in parasitic extraction decks, which would enhance cell allocation accuracy and optimize precision. These improvements aim to create a faster, more reliable methodology with reduced chances of human error.

Keywords: nanoelectronics, LVS, ERC, parasite extraction, HSPICE.

CAPÍTULO 1

Introducción

En la era actual de la tecnología, el desarrollo de circuitos integrados ha transformado radicalmente la forma en que los sistemas electrónicos cumplen con las exigencias de rendimiento, miniaturización y eficiencia energética. A medida que los nodos tecnológicos se hacen más pequeños, la necesidad de garantizar la confiabilidad y precisión en el diseño de estos circuitos se vuelve fundamental.

Este documento se centra en el diseño y validación de algunos circuitos integrados utilizando la tecnología de 180 nm de TSMC. Para asegurar su correcto funcionamiento, se llevaron a cabo pruebas de *layout versus schematic* (LVS), verificación de reglas eléctricas (ERC) y extracción de parásitos. Las pruebas de LVS permiten verificar la consistencia entre el diseño físico y esquemático, mientras que el ERC garantiza el cumplimiento de las normas eléctricas esenciales para evitar posibles fallos de diseño. Además, la extracción de parásitos permite identificar el comportamiento del circuito logrando simulaciones de funcionamiento más precisas y representativas de la realidad. El objetivo no fue solo asegurar que el circuito cumpliera con los estándares de funcionalidad, sino también elaborar documentación detallada que sirva como guía en futuros proyectos de diseño con esta tecnología.

CAPÍTULO 2

Antecedentes

Este proyecto de graduación se basará en los trabajos de años pasados, que realizaron el diseño y crearon el avance en distintas etapas del primer nanochip creado por una universidad centroamericana, llamado “El Gran Jaguar”. En primer lugar, se replicarán los proyectos de graduación de años anteriores con el fin de comprender el proceso de diseño del nanochip. La documentación proporcionada es especialmente útil para comprender cada etapa del proceso.

El proyecto del nanochip dio inició en el 2019, sin embargo, hay teoría importante que existe desde unos años antes. De los Santos (2014) menciona que la nanoelectrónica, como campo de estudio, se ha destacado por su enfoque en el diseño e implementación de circuitos a muy gran escala, con impacto tanto en la industria como en la vida cotidiana. En el trabajo universitario se introduce a la Universidad del Valle de Guatemala en este campo, utilizando tecnología de vanguardia y software proporcionado por Synopsys. Se partió desde cero instalando las aplicaciones necesarias en el laboratorio de electrónica, siguiendo meticulosamente las instrucciones de Synopsys. Se elaboraron guías de uso para facilitar el aprendizaje de los diseñadores, mientras que la librería proporcionada por Synopsys, avalada por TSMC, permitió llevar a cabo con éxito un proyecto de diseño de un sumador/restador de 32 bits [1].

Nájera y Rubio (2019) se centran en los primeros pasos del flujo de diseño para la creación de circuitos nanométricos con tecnología CMOS, buscando elaborar circuitos en lenguaje descriptivo de hardware y mapearlos a un *netlist* de compuertas lógicas mediante síntesis, con verificación de su funcionamiento previo y posterior a este proceso. Los autores utilizaron la herramienta Design Vision junto con librerías de 90nm proporcionadas por Synopsys, así como las herramientas Formality y VCS para asegurar la correcta ejecución de la síntesis y simular el comportamiento del circuito. El trabajo incluye pruebas de síntesis con distintos circuitos para comprender mejor la herramienta, y se resume la obtención del software, librerías y su uso, con la intención de servir de guía para futuros estudiantes en el diseño de circuitos nanométricos. Además, se propone un esquema detallado para abordar ambas partes del proceso del "flujo de diseño", desde el diseño de la funcionalidad hasta la generación de los planos necesarios para la fabricación del Chip o Circuito Integrado. Se

desarrollaron guías de instalación y uso de herramientas para cada etapa del proceso, junto con consideraciones para la revisión y detección de errores en la generación de archivos en cada etapa del flujo, así como para la preparación de futuras variaciones de este [2] [3].

Sibrian (2020) se concentró en llevar a cabo el proceso de Diseño de Reglas de Comprobación para finalizar el diseño de el circuito. Este sistema está concebido para generar un mensaje en codificación ASCII que, al ser procesado por un decodificador, reproduzca de forma audible los nombres de las personas involucradas en el proyecto. La revisión de errores por reglas de diseño se efectuó utilizando la herramienta Custom Compiler de Synopsys, ejecutando un *Runset* proporcionado por TSMC, el fabricante del circuito integrado. Durante la síntesis de este circuito, no se realizaron optimizaciones específicas para variables como la potencia o el *timing*. Es importante destacar que el proceso de DRC empleó el diseño real generado en pasos anteriores del flujo de diseño, entregando un diseño libre de errores en formato GDSII para permitir la continuación con la extracción de parásitos y la comparación del Layout Versus Schematic (LVS) [4].

En el trabajo de graduación de Ricardo Girón (2020) se establece una base sólida en uno de los módulos de verificación física dentro del flujo de diseño de chips a nanoescala, conocido como LVS. El proceso de LVS se encarga de garantizar la integridad del *layout* de un circuito integrado. Este flujo comprende dos procesos fundamentales. En primer lugar, está el proceso de extracción, donde se genera un netlist al extraer dispositivos y sus interconexiones de una base de datos de un *layout*. Posteriormente, tras la síntesis lógica, se obtiene un *netlist* del esquemático original. Este *netlist* se traduce a un formato compatible con la herramienta IC Validator de Synopsys mediante NetTran, lo que permite su comparación, constituyendo el segundo proceso del flujo de LVS. Durante la comparación, se realizan ajustes globales en los netlists y se generan puntos de equivalencia (celdas equivalentes) para su posterior comparación. Este proceso se basa en la comparación de un par de celdas equivalentes, una del *netlist* del *layout* y otra del *netlist* del esquemático. Finalmente, la herramienta IC Validator produce un conjunto de archivos de resultados interpretables, que facilitan la toma de decisiones y la corrección de errores en el diseño del chip [5].

Flores (2020) menciona que el flujo de diseño de un circuito integrado, comprendido como una secuencia de pasos para obtener los planos necesarios para su fabricación a partir del diseño en un lenguaje descriptor de hardware, se divide en síntesis lógica y síntesis física. Este trabajo se centra en la fase de implementación del anillo de Entradas/Salidas, así como en la realización de pruebas de Antena y Reglas de diseño eléctrico, cruciales para la viabilidad del circuito integrado. Durante la implementación del anillo, se seleccionan los *pads* de Entrada/Salida apropiados y se crean los *pads* de voltaje y tierra necesarios, evitando los efectos adversos de Antena que podrían afectar la integridad de los transistores durante los pasos de grabado de plasma. Las pruebas de Antena se ejecutan mediante archivos específicos en herramientas como IC Validator y IC Compiler. Finalmente, se verifica el cumplimiento de las reglas de diseño eléctrico (ERC), que incluyen seis reglas principales, durante el proceso de LVS, generando archivos de resultados que permiten interpretar los errores de *layout* y abordar cualquier problema en el diseño eléctrico del circuito [6].

Cardona (2021) se enfoca principalmente en la reconfiguración de la etapa de síntesis lógica como parte del flujo de diseño para la creación del chip. Se adopta un enfoque detallado, aprovechando la herramienta Design Vision de Synopsys para llevar a cabo pruebas exhaustivas y verificar el funcionamiento del script utilizado en la fase inicial del proyecto,

mientras se realizan pruebas con diversos circuitos, desde compuertas básicas hasta sumadores de múltiples bits. La síntesis lógica se divide en dos etapas, abarcando la generación de archivos Verilog y la asignación de pines de entrada y salida. Todo el proceso se documenta meticulosamente, desde las pruebas realizadas hasta la traducción del *netlist* a nivel transistor, con el objetivo de proporcionar una guía comprensible tanto para el uso de la herramienta como para el proceso de síntesis [7].

El trabajo de Elmer Torres (2021) se centra en el desarrollo de la etapa de síntesis lógica. Esta fase inicial es crucial dentro del flujo de diseño del circuito integrado y, por ende, se enfoca en garantizar resultados satisfactorios y la continuidad del flujo sin interrupciones. El objetivo principal es realizar y verificar que la síntesis lógica cumpla con todos los requisitos necesarios para las siguientes etapas, sin errores. Para llevar a cabo esta síntesis, se empleó la herramienta Design Vision de Synopsys, que permite la lectura de archivos HDL y su síntesis utilizando librerías proporcionadas por el fabricante para obtener un diseño utilizable con celdas reales. Se utilizaron diversos circuitos, como compuertas NOT, XOR, contadores síncronos, sumadores completos, memorias ROM/RAM y una ALU, para corroborar este proceso. Finalmente, se utilizó la herramienta VCS y su complemento DVE para verificar el funcionamiento de los diferentes circuitos, mediante la ejecución de simulaciones con un *test bench* diseñado para simular el comportamiento de las entradas y salidas de los circuitos [8].

Letona (2022) se enfocó en alcanzar un dominio avanzado en el uso de StarRC para identificar comandos esenciales en el desarrollo de circuitos integrados, ya que esta herramienta de Synopsys ofrecía una amplia gama de funcionalidades que aún no habían sido exploradas por completo. A pesar de la extensa documentación disponible, se buscó identificar procesos, opciones y archivos cruciales para la culminación del nanochip. Aquí se pretendía generar un CMD que facilitara la generación de archivos con parásitos mediante StarRC, aprovechando la documentación de promociones pasadas para replicar resultados y comprender los procedimientos necesarios. Además, se documentaron opciones relevantes para realizar análisis específicos con la herramienta y se identificaron comandos útiles que simplificaron su uso. Además, se llevó a cabo la identificación de las compuertas empleadas en el diseño del nanochip y se crearon decks individuales para cada compuerta, asegurando que cumplieran con su comportamiento lógico para simular el archivo de salida de la extracción de parásitos. Para concluir, se propusieron distintas estrategias de extracción y diversos flujos para la generación de bases de datos [9].

En el trabajo de Estuardo Mancio (2022) se tiene el enfoque de evaluar el funcionamiento del código Verilog obtenido durante la síntesis lógica. Se optó por implementar un entorno de simulación utilizando un FPGA Xilinx *Geneys*. Tras familiarizarse con el entorno de trabajo ISE, se realizaron correcciones en las librerías y se crearon guías de instalación y operación. Se desarrollaron códigos en Verilog para pruebas físicas en los puertos I/O, incluyendo un bus de datos de 8 bits. Luego, se generaron códigos para cajas negras no proporcionadas por TSMC y se realizaron pruebas de funcionamiento en el entorno de simulación Isim. Se intentó finalizar el proyecto utilizando un microcontrolador como traductor de la salida del nanochip para conectarlo a una computadora y mostrar su funcionamiento [10].

Luis Gómez (2022) se enfocó en el funcionamiento en un FPGA Digilent Genesys Board, con el propósito de demostrar la síntesis lógica y obtener resultados esperados. Se inició con la instalación del software ISE Design Suite 14.7 en sistemas operativos Windows 11 y Linux, específicamente distribución CentOS 7, para luego crear un nuevo proyecto y importar

el archivo Verilog de la síntesis lógica proporcionado por el equipo de automatización del proceso de creación del nanochip. Además, se desarrollaron los *blackboxes* o módulos de compuertas necesarios para que el FPGA Digilent Genesys Board realice la síntesis sin errores y se ponga en práctica la estrategia de trabajo. Se empleó un microcontrolador (Tiva C) para recibir el bus de datos de 8 bits y transmitirlo a una computadora mediante comunicación serial (UART). Por último, se diseñó una aplicación en Python utilizando las librerías de PySerial para recibir los datos del microcontrolador y, con la ayuda de la librería Pyttsx3, convertir la cadena de texto a audio, logrando así el sintetizador digital deseado [11].

CAPÍTULO 3

Justificación

El desarrollo de nanochips constituye un avance tecnológico crucial en el panorama actual, dado su potencial para revolucionar una amplia gama de aplicaciones, desde dispositivos electrónicos de consumo hasta aplicaciones médicas y científicas. Los nanochips representan una innovación tecnológica significativa en el campo de la microelectrónica, permitiendo la integración de una gran cantidad de componentes en un espacio reducido y, por ende, la creación de dispositivos más pequeños, rápidos y eficientes energéticamente.

Además, la versatilidad de los nanochips los hace indispensables en numerosas industrias, desde dispositivos móviles hasta sistemas de comunicación, sistemas médicos y sensores. La colaboración entre IMEC, un centro de investigación líder en nanotecnología, subraya la calidad y la rigurosidad técnica del proyecto, aprovechando la experiencia y los recursos de ambas organizaciones para alcanzar resultados óptimos.

Del mismo modo, contribuirá al avance del conocimiento científico en el campo de la microelectrónica y la nanotecnología, con posibles aplicaciones más amplias en la investigación científica y tecnológica. En sí, el proyecto de diseño y fabricación de "El Gran Jaguar" es crucial dada su contribución al avance tecnológico, sus diversas aplicaciones, la colaboración internacional, y su potencial para impulsar la investigación científica.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Realizar pruebas LVS, ERC y extracción de parásitos a un circuito integrado diseñado con tecnología de 180 nm de TSMC para garantizar su funcionalidad y confiabilidad.

4.2. Objetivos específicos

- Replicar el progreso del proyecto del circuito integrado llevado a cabo en años previos.
- Ejecutar pruebas de Verificación de *Layout vs. Schematic* (LVS) para verificar la consistencia entre el diseño esquemático y el diseño físico del circuito integrado.
- Realizar pruebas de *Electrical Rules Checking* (ERC) para garantizar el cumplimiento de las normativas eléctricas y evitar posibles problemas de diseño.
- Realizar la extracción de parásitos para identificar y obtener un *Deck* que tenga influencia de resistencias y capacitancias parásitas para que las simulaciones finales sean lo más apegadas a la realidad posible.
- Analizar los resultados de las pruebas y realizar las modificaciones necesarias en el diseño para mejorar la funcionalidad y la confiabilidad del circuito integrado.
- Elaborar documentación sobre el uso del flujo de diseño específicamente del *Layout vs. Schematic*, *Electrical Rules Checking* y extracción parásita que proporcione una guía clara y concisa.

CAPÍTULO 5

Alcance

Esta tesis se centra en la realización de pruebas de Verificación de *Layout vs. Schematic* (LVS), verificación de Reglas Eléctricas (ERC) y extracción de parásitos en un circuito integrado diseñado con tecnología de 65 nm de TSMC. El proyecto se llevará a cabo de enero a noviembre de 2024. El objetivo es garantizar la funcionalidad y confiabilidad del circuito integrado. Se replicará el progreso del proyecto del circuito integrado llevado a cabo en años previos, actualizando la tecnología a 65 nm de TSMC y sus librerías de diseño. Posteriormente, se ejecutarán pruebas de LVS para verificar la consistencia entre el diseño esquemático y el diseño físico del circuito, así como pruebas de ERC. Adicionalmente, se llevará a cabo la extracción de parásitos para identificar y obtener un deck que incluya la influencia de resistencias y capacitancias parásitas, lo que permitirá que las simulaciones finales sean lo más precisas y realistas posible. Finalmente, se elaborará documentación detallada sobre el uso del flujo de diseño, específicamente en las áreas de LVS, ERC y extracción parásita. Esta documentación proporcionará una guía clara y concisa para futuras referencias y proyectos, sirviendo como recurso valioso. Para la ejecución de este proyecto se utilizará *IC Compiler 2* (ICC2), *IC Validator*, herramientas de *Synopsys*, librerías de diseño de TSMC y una máquina virtual *Rocky Linux*. El proyecto se centrará prinipalmente en el circuito ."El Gran Jaguar"pero también se incluye la realización de pruebas de LVS en otros circuitos adicionales. Del mismo modo, se busca que las pruebas de LVS salgan limpias con las reglas establecidas. Estas pruebas permitirán validar la metodología aplicada para poder ver su generalización. Con lo cual, el manual final puede ser aplicado a otros diseños. Por último, la extracción parásita se realizará con un enfoque detallado, abarcando las resistencias y capacitancias, siendo un RC.

CAPÍTULO 6

Marco teórico

6.1. *Design Compiler*

Esta solución permite a los usuarios resolver los desafíos de diseño actuales con una optimización simultánea de tiempo, área, potencia y prueba. Este incluye tecnología tipográfica que permite un flujo predecible, dando así resultados más rápidos. La tecnología tipográfica proporciona así una predicción de tiempo y área dentro del 10 % de los resultados vistos después del diseño, permitiendo así reducir las iteraciones entre la síntesis y la implementación física. *Design Compiler* también incluye una infraestructura escalable que ofrece un tiempo de ejecución dos veces más rápido en plataforma de cuatro núcleos.

Integrar *Design Compiler* en el flujo de diseño es esencial para asegurar una implementación exitosa del diseño electrónico. Desde la codificación inicial en HDL hasta la implementación física en *IC Compiler*, cada paso del proceso tiene su lugar en la estructura general del flujo de diseño. La capacidad de realizar tareas de exploración y síntesis de alto nivel facilita la iteración rápida y la mejora continua del diseño a lo largo del ciclo de desarrollo. [12]

6.2. *VCS Funtional Verification*

La *VCS Funtional Verification* de *Synopsys* es una de las soluciones principales de verificación. Proporciona los motores de resolución de restricciones y simulación de mayor rendimiento. Este aprovecha el máximo de forma nativa los procesadores multinúcleos con tecnología de paralelismo de grano fino (FGP por sus siglas en inglés) de última generación. Esto permite acelerar las pruebas de ciclo largo y alta actividad asignando más núcleos en tiempo de ejecución. Del mismo modo, posee características para lograr un mayor rendimiento y permitir desplazar los flujos de verificación a la izquierda al principio del ciclo de diseño para detectar errores adicionales [13].

6.3. *IC Compiler*

Este incluye herramientas para la planificación de diseño plano y jerárquico, exploración temprana del diseño, ubicación y optimización consciente de la congestión, síntesis del árbol de reloj, convergencia avanzada de enrutamiento de nodos, cumplimiento de fabricación y cierre de aprobación. Este esta diseñado para abordar las presiones agresivas de rendimiento, potencia, área y tiempo de comercialización de diseños. Las tecnologías que este incluye son marco de optimización omnipresente en paralelo, ubicación global multiobjetivo, optimización de ubicación impulsada por enrutamiento, optimización de datos y reloj concurrente basada en Arc de flujo completo, optimización de energía total, flujo consciente de múltiples patrones y *FinFET* y optimización impulsada por aprendizaje automático (ML) para cierre de diseño rápido y predictivo [14].

6.4. *IC Validator*

La solución de aprobación de alto rendimiento de verificación física *Synopsys IC Validator* mejora la productividad en todos los nodos de proceso. *Synopsys IC Validator* ofrece escalabilidad de procesamiento distribuido de la industria a más de 4000 núcleos de CPU. El rendimiento y la escalabilidad de la herramienta permitieron algunos de los chips de límite de retícula más grandes de la industria con miles de millones de transistores, verificación de reglas de diseño (DRC) el mismo día, diseño versus esquema (LVS) y tiempo de respuesta de llenado.

IC Validator se integra perfectamente con la solución Synopsys Fusion Compiler™ RTL a GDSII y el sistema de ubicación y ruta IC Compiler® II de la familia de diseño digital Synopsys. Esta tecnología de fusión integrada acelera el cierre del diseño para la fabricación al permitir un análisis independiente de la calidad y la reparación automática dentro del entorno de implementación [15].

6.5. *Custome WaveView*

Custome WaveView representa una herramienta fundamental en el ámbito del diseño de circuitos integrados analógicos y mixtos. Al ofrecer una interfaz gráfica intuitiva y potente, permite visualizar y analizar las formas de onda generadas por las simulaciones de diseño. Esta capacidad de visualización es crucial para comprender el comportamiento de los circuitos en diferentes condiciones de funcionamiento, lo que ayuda a identificar posibles problemas de diseño y a optimizar el rendimiento de los circuitos.

Una de las características destacadas de *Custome WaveView* es su capacidad para manejar grandes volúmenes de datos de simulación de manera eficiente. Esto es especialmente importante en el diseño de chips, donde las simulaciones pueden generar enormes cantidades de información. La eficiencia en la gestión de estos datos permite explorar y analizar múltiples aspectos del diseño de manera rápida y efectiva.

Además de su capacidad para visualizar formas de onda, *Custome WaveView* también

ofrece herramientas avanzadas de análisis. Estas herramientas permiten extraer parámetros clave del diseño, como el rendimiento de la señal, la potencia consumida o la estabilidad del circuito. Esta información es esencial para tomar decisiones informadas durante el proceso de diseño y garantizar que el chip final cumpla con los requisitos de rendimiento y especificaciones [16].

6.6. *StarRC*

StarRC es una herramienta de análisis de resistencia y capacitancia parasitaria utilizada en el diseño de circuitos integrados. Su función principal es calcular las resistencias y capacitancias parasitarias que surgen en un diseño de chip debido a la interconexión de los componentes y la estructura del circuito. Estas resistencias y capacitancias pueden afectar significativamente el rendimiento y la fiabilidad del circuito, especialmente en diseños de alta velocidad y densidad.

La importancia de *StarRC* radica en su capacidad para predecir con precisión y eficiencia las características eléctricas de un diseño de chip. Al modelar con precisión las resistencias y capacitancias parasitarias, se puede identificar y mitigar posibles problemas de señal, como retrasos en la señal, interferencias y pérdidas de potencia. Esto es crucial para garantizar que el chip funcione según lo previsto y cumpla con los requisitos de rendimiento y especificaciones.

Además, *StarRC* proporciona herramientas avanzadas de análisis y visualización que permiten explorar y comprender mejor el comportamiento eléctrico de los diseños. Esto permite tomar decisiones informadas durante el proceso de diseño y optimizar el diseño para cumplir con los objetivos de rendimiento, consumo de energía y costo [17].

6.7. *HSpice*

HSpice es una herramienta de simulación de circuitos analógicos extremadamente poderosa y ampliamente utilizada en el diseño de circuitos integrados. Se utiliza para simular y analizar el comportamiento de circuitos analógicos complejos, incluidos circuitos mixtos, digitales y analógicos, antes de la fabricación física del chip. Este se basa en el método de simulación de dominio del tiempo y utiliza modelos precisos de dispositivos y componentes para predecir el rendimiento eléctrico de un circuito en condiciones reales.

La importancia de *HSpice* en el diseño de chips radica en su capacidad para predecir con precisión el comportamiento eléctrico de un circuito antes de que se fabrique físicamente. Esto permite identificar posibles problemas de diseño, como inestabilidades, oscilaciones, tiempos de respuesta lentos o violaciones de las especificaciones, y realizar las correcciones necesarias antes de la fabricación. Además, facilita la optimización del diseño al permitir explorar diferentes configuraciones de circuitos y parámetros para lograr los mejores resultados en términos de rendimiento, consumo de energía y costo.

Aparte de la simulación de circuitos, *HSpice* ofrece herramientas avanzadas de análisis

y visualización que permiten comprender mejor el comportamiento eléctrico de los diseños. Permitiendo tomar decisiones informadas durante el proceso de diseño y garantizar que este cumpla con los requisitos de rendimiento y especificaciones [18].

6.8. *Electrical rule checking (ERC)*

El *ERC* es una herramienta crucial en el flujo de diseño de circuitos integrados que se utiliza para verificar automáticamente si el diseño cumple con las reglas eléctricas establecidas para el proceso de fabricación. Estas reglas eléctricas se centran en aspectos como la correcta polarización de los transistores, la existencia de conexiones eléctricas adecuadas y la prevención de posibles errores que puedan afectar el funcionamiento del circuito.

Su importancia radica en su capacidad para detectar y corregir problemas eléctricos en las etapas tempranas del diseño, lo que ayuda a prevenir fallos durante la operación del circuito. Entre los problemas que puede detectar el *ERC* se incluyen cortocircuitos, conexiones flotantes, niveles de voltaje incorrectos y otras configuraciones eléctricas que podrían comprometer la funcionalidad o la fiabilidad del diseño.

Al realizar el *ERC* durante la fase de diseño, se pueden identificar y corregir problemas eléctricos de manera temprana, asegurando que el diseño cumpla con los estándares de calidad y las especificaciones eléctricas antes de avanzar hacia etapas posteriores del proceso, como la simulación y la verificación de *layout vs. schematic* (LVS).

Además de verificar las reglas eléctricas estándar, el *ERC* también puede personalizarse para incluir reglas específicas del diseño o del proceso de fabricación. Esto permite adaptar el flujo de diseño a los requisitos y las restricciones únicos de cada proyecto, mejorando así la eficiencia y la calidad del diseño final del circuito integrado [19].

6.9. *Layout vs. schematic (LVS)*

El LVS es una herramienta fundamental en el flujo de diseño de chips que se utiliza para verificar la correspondencia entre el diseño del *layout* del circuito integrado y su representación esquemática. Esta verificación es crucial para garantizar que el diseño físico del chip coincida fielmente con su diseño lógico o esquemático, evitando así errores que podrían afectar el funcionamiento del dispositivo una vez fabricado.

Su importancia está en su capacidad para detectar discrepancias entre el diseño del *layout* y el diseño esquemático del chip. Estas discrepancias pueden incluir conexiones faltantes, componentes no coincidentes, errores de polaridad y otros problemas que podrían comprometer la funcionalidad y la integridad del diseño.

Al realizar esta verificación durante la fase de diseño, se puede identificar y corregir discrepancias entre el diseño del *layout* y el diseño esquemático antes de avanzar hacia etapas posteriores del proceso de diseño. Esto ayuda a minimizar los riesgos asociados con errores de diseño y a garantizar la viabilidad y la calidad del diseño final del chip.

Además de verificar la correspondencia entre el *layout* y el esquemático, puede detectar problemas de conectividad, como cortocircuitos y circuitos abiertos, al igual que errores en la topología del diseño. Esto garantiza que el diseño físico del chip sea coherente con su diseño lógico y que cumpla con los requisitos de funcionalidad y rendimiento establecidos por el diseñador [20].

CAPÍTULO 7

Proceso *layout vs schematic* (LVS)

Para entender y describir de manera integral cómo se lleva a cabo el proceso de verificación de LVS, es esencial ubicar en qué parte del flujo de diseño se sitúa esta etapa. La ejecución de LVS requiere varios archivos que provienen de dos fases anteriores en el proceso de diseño siendo estas la síntesis lógica y la síntesis física. En ciertos casos, estos archivos necesitan ser ajustados para cumplir con los formatos requeridos por las herramientas y software que se usan en la verificación LVS. Del mismo modo, se requieren un conjunto de archivos proporcionados por TSMC.

7.1. Archivos previos necesarios

Dentro de los archivos generados en la síntesis lógica, se obtienen varios archivos a su salida. Sin embargo el más importante es el *.v* que es un archivo *verilog* pues es usado junto con otros archivos para extraer una salida en formato *.icv*, el cual es el formato compatible con *IC Validator*. Sin embargo, este archivo se modifica automáticamente en la síntesis física para que el *verilog* tenga las conexiones físicas.

Ahora, los archivos generados en la síntesis física son más variados, pero el más importante aquí es el *.gds*. Pues este es el que nos ayudará en la elaboración de la prueba del LVS, además de que es el que contiene la *top cell* del circuito.

Adicional a estos archivos mencionados, existen otros archivos necesarios para la ejecución correcta del LVS. Estos archivos son archivos proporcionados por TSMC. Llevan por nombre *CORE.v*, *IO.v* y *LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a*. Los primeros dos archivos *verilog* contienen las librerías estándar y las librerías de *inputs* y *outputs* de los circuitos a analizar. El último archivo con extensión `.4a` es el archivo de *runset*; en este se deben verificar los parámetros introducidos en el *environment setup* y la adición de las

celdas que fueron utilizadas en el diseño de cada circuito. Esto quiere decir que el archivo tiene que modificarse para poder cumplir los parámetros de cada circuito.

7.2. Archivos a generar

7.2.1. Archivo *ICV*

Para poder generar el archivo *ICV* se necesitan de los archivos *verilog* antes mencionados (*CORE.v* y *IO.v*). Este archivo es necesario para poder ejecutar el LVS. Estos dos archivos *verilog* se deben concatenar para así obtener como salida un solo archivo con todas las librerías juntas. Para esto se utiliza el siguiente comando:

Cuadro 1
Comando para concatenar los archivos *verilog*

```
cat CORE.v IO.v> headers.v
```

Nota. Elaboración propia.

En este caso el archivo de salida se nombró como *headers.v* y es un formato *verilog* de igual modo. Después de obtener este archivo, se utiliza la herramienta *NetTran*, pues permite una "traducción"de *verilog* a un formato *scpice* o *CDL*, que es un formato compatible para extraer el archivo *ICV*. A continuación se muestra el comando para la traducción del archivo:

Cuadro 2
Comando para generar el archivo *SPICE*

```
icv_nettran -verilog headers.v -outType SPICE -outName headers.sp
```

Nota. Elaboración propia.

El último paso previo a la extracción del archivo *ICV* es modificar el archivo *.sp*, *headers.sp*. La modificación que se debe realizar es agregar las siguientes líneas de texto al inicio del archivo.

Cuadro 3
Líneas a agregar en el archivo *SPICE*

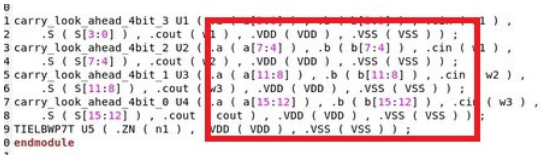
```
.GLOBAL VDD VSS VDDPST VSSPST
*.EQUATION
*.SCALE METER
*.MEGA
.PARAM
.INCLUDE /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   nueva_integracion/source.added
```

Nota. Elaboración propia.

La última línea variará en cada caso, pues se debe indicar la dirección donde se encuentre el archivo. Este es brindado por TSMC y contiene la delcaración de los subcircuitos de los dispositivos. El proceso anterior, que involucra a los archivos *CORE.v*, *IO.v*, *headers.v* y *headers.sp* se puede ejecutar una sola vez, pues el archivo final (*headers.sp*) es el que se utilizará para todos los circuitos y puede copiarse y ser utilizado en cada circuito a analizar. Sin embargo, antes de ejecutar el comando es necesario realizar algunas modificaciones al archivo *verilog* si se requieren. La primera verificación es comprobar si las compuertas poseen conexiones a VDD y VSS como se muestra en la figura 1. Si es así, es necesario borrar esas conexiones. Del mismo modo hay que borrar las instancias de componentes puramente físicos, esto incluye todos los *PCORNER*, *PFILLER* y *FILL* que posee el archivo. Estos se pueden ver en las figuras 2 y 3.

Figura 1

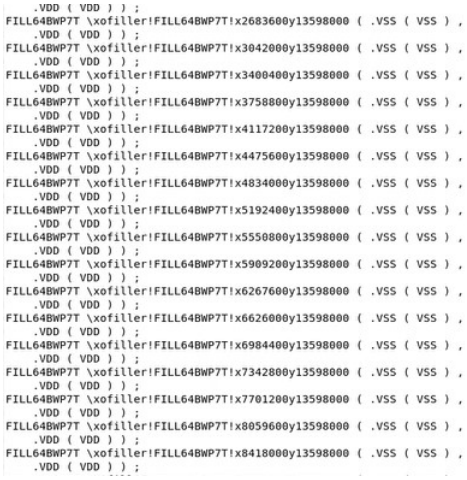
Celdas conectadas a VDD y VSS en el *verilog*



Nota. Elaboración propia.

Figura 2

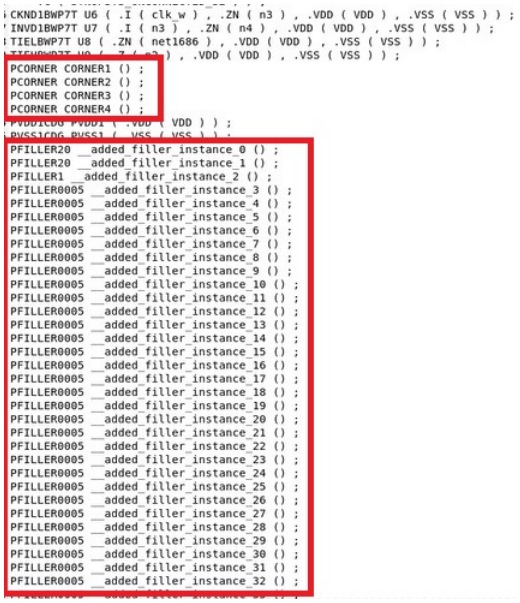
Instancias físicas I



Nota. Elaboración propia.

Figura 3

Instancias físicas II



Nota. Elaboración propia.

Finalmente se debe ejecutar el comando para poder exportar el archivo en formato *ICV*. Para esto se debe utilizar el archivo *headers.sp* y el *netlist* en formato *verilog* (que se obtuvo en la parte de la síntesis física). El comando se muestra a continuación:

Cuadro 4

Comando para generar el archivo *ICV*

```
icv_nettran -verilog netlist_circuito.v -sp headers.sp -outType ICV -outName
    CDL_netlist.icv
```

Nota. Elaboración propia.

La salida *CDL_ netlist.icv* será utilizada más adelante para poder ejecutar la prueba LVS. Donde *netlist_ circuito.v* hace referencia a la *netlist* en formato *verilog*.

7.2.2. Archivo *runset*

Luego de obtener este archivo y tener el archivo *gds* es necesario modificar el *runset*. En este, es importante ajustar la configuración de entorno (*environment setup*) que se muestra en la figura 4. Primero se modifica el primer recuadro en el cual se coloca en *library_ name* la ubicación del archivo *gds* del circuito. Mientras que en *cell* se coloca el nombre de la *top cell* del circuito, ese mismo nombre se coloca también en *SCHEMATIC_ TOPCELL* en el segundo recuadro. Los otros campos que se modifican son *schematic_ file* y *schematic_ library_ file*. En el primero se coloca el directorio del archivo *icv* y en el otro el directorio de un archivo llamado *unit.cdl*, el cual es un archivo proporcionado por TSMC. En el último recuadro, se muestran las configuraciones necesarias para ejecutar el *LVS* correctamente.

Figura 4

Enviroment setup

```
//////////////////////
// ENVIRONMENT SETUP //
//////////////////////

library(
        library_name = "/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_mult32/Outputs/IO/chip.gds",
        cell = "multiplier_16bit_IO",
        format = GDSII
);

SCHEMATIC_TOPCELL : string = "multiplier_16bit_IO";            // Set schematic top cell name here
sch_db = schematic(
        schematic_file = {{"/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_mult32/Outputs/LVS/CDL_Not_5.icv", ICV}},
        schematic_library_file = {{"/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/SCRIPTS_NUEVOS/20191128-124344/unit.cdl", SPICE}},
        expand_multiple_devices = true,
        spice_settings = {slash_is_space = false}

//#define USER_EQUIV_FILE                       // Turn on for user-specified equivalent point file flow
#ifdef USER_EQUIV_FILE
#include "./user.equiv"                         // Set equivalent point file here
#endif




/* EDIT: The following section contains all of the runset variables for RC extraction tools. */
#define RC_DECK                             // Turn on for LPE/RC extraction
//#define CROSS_REFERENCE                   // Turn on for source cross reference in LPE extraction
//#define ZERO_NRS_NRD                      // Turn off when this deck would calculate NRS and NRD
//#define FILTER_DGS_TIED_MOS               // Turn on to filter MOS with D, G and S tied together (default filter MOS with all pins tied)

#define WELL_TO_PG_CHECK                    // Default is on. Turn on to highlight if nwell connects to ground or psub connects to power.
//#define GATE_TO_PG_CHECK                  // Default is off. Turn on to highlight if a mos gate directly connects to power or ground.
//#define PATH_CHECK                        // Default is off. Turn on to highlight if
                                            // (1) nodes have a path to power but no path to ground
                                            // (2) nodes have a path to ground but no path to power
                                            // (3) nodes have no path to power or ground
                                            // (4) nodes have no path to any label net
#define DS_TO_PG_CHECK                      // Default is on. Turn on to highlight if drain connects to power and source connects to ground.
#define FLOATING_GATE_CHECK                         // Default is on. Turn on to highlight if there are floating gates.
#define FLOATING_WELL_CHECK                 // Default is on. Turn on to highlight if well does not connect to power or ground.
                                            // The nwell of moscaps and nwell-resistor are excluded
//#define NW_RING                           // Turn on to enable NW ring to separate the node from BULK


// #define MACRO
//By default, macro models are not netlisted. Please turn on this switch to enable "_mac" models
```

Nota. Elaboración propia.

Otra cosa que se tiene que modificar en el *runset* son las *black boxes*. Estas se tienen que agregar dependiendo del circuito. Estas se encuentran en el apartado *ICV OPTIONS*. Varían para cada circuito y a medida que el circuito tenga mayor dificultad, tendrá más *black boxes*. En la figura 5 se observa un ejemplo de como agregar las *black boxes* necesarias para cada circuito. Este formato las permite visualizar de tal forma en la cual no se sabe lo que ocurre dentro y solo se especifican las entradas y salidas de la misma.

Figura 5

Ejemplo del formato de *black boxes*

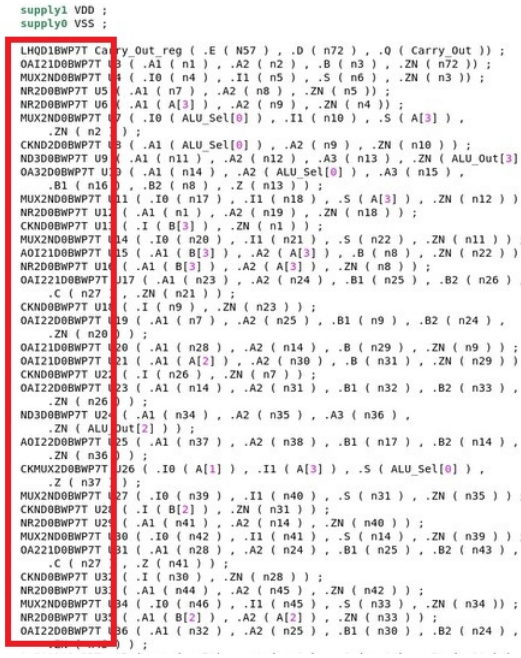
```
lvs_black_box_options(
        equiv_cells = {{schematic_cell = "CKND0BWP7T", layout_cell = "CKND0BWP7T"}},
        remove_schematic_ports = {"I", "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "CKND1BWP7T", layout_cell = "CKND1BWP7T"}},
        remove_schematic_ports = {"I", "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "INVD1BWP7T", layout_cell = "INVD1BWP7T"}},
        remove_schematic_ports = {"I", "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "NR2D1BWP7T", layout_cell = "NR2D1BWP7T"}},
        remove_schematic_ports = {"A1", "A2" , "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "OAI21D0BWP7T", layout_cell = "OAI21D0BWP7T"}},
        remove_schematic_ports = {"A1", "A2", "B", "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "OA31D0BWP7T", layout_cell = "OA31D0BWP7T"}},
        remove_schematic_ports = {"A1", "A2", "A3" , "B", "Z"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "INR2XD0BWP7T", layout_cell = "INR2XD0BWP7T"}},
        remove_schematic_ports = {"A1", "B1", "ZN"}
);

lvs_black_box_options(
        equiv_cells ={{schematic_cell = "ND3D0BWP7T", layout_cell = "ND3D0BWP7T"}},
        remove_schematic_ports = {"A1", "A2", "A3" , "ZN"}
);

lvs_black_box_options(
        equiv_cells ={{schematic_cell = "NR2D0BWP7T", layout_cell = "NR2D0BWP7T"}},
        remove_schematic_ports = {"A1", "A2" , "ZN"}
);
```

Nota. Elaboración propia.

Para saber cuales son las *black boxes* que hay que agregar se observa en el archivo *verilog* las celdas que se usan para describir el circuito como se muestra en la siguiente figura 6:

Figura 6

Black boxes que se usan en los circuitos



```
supply1 VDD ;
supply0 VSS ;

LHQD1BWP7T Carry_Out_reg ( .E ( N57 ) , .D ( n72 ) , .Q ( Carry_Out )) ;
OAI21D0BWP7T U3 ( .A1 ( n1 ) , .A2 ( n2 ) , .B ( n3 ) , .ZN ( n72 )) ;
MUX2ND0BWP7T U4 ( .I0 ( n4 ) , .I1 ( n5 ) , .S ( n6 ) , .ZN ( n3 )) ;
NR2D0BWP7T U5 ( .A1 ( n7 ) , .A2 ( n8 ) , .ZN ( n5 )) ;
NR2D0BWP7T U6 ( .A1 ( A[3] ) , .A2 ( n9 ) , .ZN ( n4 )) ;
MUX2ND0BWP7T U7 ( .I0 ( ALU_Sel[0] ) , .I1 ( n10 ) , .S ( A[3] ) ,
    .ZN ( n2 ) ) ;
CKND2D0BWP7T U8 ( .A1 ( ALU_Sel[0] ) , .A2 ( n9 ) , .ZN ( n10 ) ) ;
ND3D0BWP7T U9 ( .A1 ( n11 ) , .A2 ( n12 ) , .A3 ( n13 ) , .ZN ( ALU_Out[3] ) ) ;
OA32D0BWP7T U10 ( .A1 ( n14 ) , .A2 ( ALU_Sel[0] ) , .A3 ( n15 ) ,
    .B1 ( n16 ) , .B2 ( n8 ) , .Z ( n13 ) ) ;
MUX2ND0BWP7T U11 ( .I0 ( n17 ) , .I1 ( n18 ) , .S ( A[3] ) , .ZN ( n12 ) ) ;
NR2D0BWP7T U12 ( .A1 ( n1 ) , .A2 ( n19 ) , .ZN ( n18 ) ) ;
CKND0BWP7T U13 ( .I ( B[3] ) , .ZN ( n1 ) ) ;
MUX2ND0BWP7T U14 ( .I0 ( n20 ) , .I1 ( n21 ) , .S ( n22 ) , .ZN ( n11 ) ) ;
AOI21D0BWP7T U15 ( .A1 ( B[3] ) , .A2 ( A[3] ) , .B ( n8 ) , .ZN ( n22 ) ) ;
NR2D0BWP7T U16 ( .A1 ( B[3] ) , .A2 ( A[3] ) , .ZN ( n8 ) ) ;
OAI221D0BWP7T U17 ( .A1 ( n23 ) , .A2 ( n24 ) , .B1 ( n25 ) , .B2 ( n26 ) ,
    .C ( n27 ) , .ZN ( n21 ) ) ;
CKND0BWP7T U18 ( .I ( n9 ) , .ZN ( n23 ) ) ;
OAI22D0BWP7T U19 ( .A1 ( n7 ) , .A2 ( n25 ) , .B1 ( n9 ) , .B2 ( n24 ) ,
    .ZN ( n20 ) ) ;
OAI21D0BWP7T U20 ( .A1 ( n28 ) , .A2 ( n14 ) , .B ( n29 ) , .ZN ( n9 ) ) ;
OAI21D0BWP7T U21 ( .A1 ( A[2] ) , .A2 ( n30 ) , .B ( n31 ) , .ZN ( n29 ) ) ;
CKND0BWP7T U22 ( .I ( n26 ) , .ZN ( n7 ) ) ;
OAI22D0BWP7T U23 ( .A1 ( n14 ) , .A2 ( n31 ) , .B1 ( n32 ) , .B2 ( n33 ) ,
    .ZN ( n26 ) ) ;
ND3D0BWP7T U24 ( .A1 ( n34 ) , .A2 ( n35 ) , .A3 ( n36 ) ,
    .ZN ( ALU_Out[2] ) ) ;
AOI22D0BWP7T U25 ( .A1 ( n37 ) , .A2 ( n38 ) , .B1 ( n17 ) , .B2 ( n14 ) ,
    .ZN ( n36 ) ) ;
CKMUX2D0BWP7T U26 ( .I0 ( A[1] ) , .I1 ( A[3] ) , .S ( ALU_Sel[0] ) ,
    .Z ( n37 ) ) ;
MUX2ND0BWP7T U27 ( .I0 ( n39 ) , .I1 ( n40 ) , .S ( n31 ) , .ZN ( n35 ) ) ;
CKND0BWP7T U28 ( .I ( B[2] ) , .ZN ( n31 ) ) ;
NR2D0BWP7T U29 ( .A1 ( n41 ) , .A2 ( n14 ) , .ZN ( n40 ) ) ;
MUX2ND0BWP7T U30 ( .I0 ( n42 ) , .I1 ( n41 ) , .S ( n14 ) , .ZN ( n39 ) ) ;
OA221D0BWP7T U31 ( .A1 ( n28 ) , .A2 ( n24 ) , .B1 ( n25 ) , .B2 ( n43 ) ,
    .C ( n27 ) , .Z ( n41 ) ) ;
CKND0BWP7T U32 ( .I ( n30 ) , .ZN ( n28 ) ) ;
NR2D0BWP7T U33 ( .A1 ( n44 ) , .A2 ( n45 ) , .ZN ( n42 ) ) ;
MUX2ND0BWP7T U34 ( .I0 ( n46 ) , .I1 ( n45 ) , .S ( n33 ) , .ZN ( n34 )) ;
NR2D0BWP7T U35 ( .A1 ( B[2] ) , .A2 ( A[2] ) , .ZN ( n33 ) ) ;
OAI22D0BWP7T U36 ( .A1 ( n32 ) , .A2 ( n25 ) , .B1 ( n30 ) , .B2 ( n24 ) ,
```

Nota. Elaboración propia.

Otra forma de hacerlo es realizar todo el proceso (con los comandos que se relatan a continuación) y ver el archivo con extensión *_lvs.log* y ver los errores,, como se ve en la figura 7.

Figura 7

Black boxes que se usan en los circuitos desde los errores

```
+------------------------------------------------------------------------+
|                          Preprocessing Stage                           |
+------------------------------------------------------------------------+

Loading netlists ...

Deleted schematic cells:
    [none].

Deleted layout cells:
    [none].
Loading netlists Time=0:00:00  User=0.00 Sys=0.01 Mem=0.018 GB

Checking netlists ...
ERROR:  Found schematic empty cell AO22D0BWP7T not defined as device.
ERROR:  Found schematic empty cell NR4D0BWP7T not defined as device.
ERROR:  Found schematic empty cell CKMUX2D0BWP7T not defined as device.
ERROR:  Found schematic empty cell INVD2BWP7T not defined as device.
ERROR:  Found schematic empty cell DFCND0BWP7T not defined as device.
ERROR:  Found schematic empty cell OAI31D1BWP7T not defined as device.
ERROR:  Found schematic empty cell INR2D0BWP7T not defined as device.
ERROR:  Found schematic empty cell IOA21D0BWP7T not defined as device.
ERROR:  Found schematic empty cell CKND2D0BWP7T not defined as device.
ERROR:  Found schematic empty cell XNR2D1BWP7T not defined as device.
ERROR:  Found schematic empty cell CKXOR2D0BWP7T not defined as device.
ERROR:  Found schematic empty cell MOAI22D0BWP7T not defined as device.
ERROR:  Found schematic empty cell AOI22D0BWP7T not defined as device.
```

Nota. Elaboración propia.

Ahora, para saber cuales son las entradas y salidas de cada *black box* se puede buscar cada celda en el archivo *CORE.v* antes mencionado. Cabe recalcar que el orden de las entradas y salidas tiene que ser el que se utiliza en ese archivo.

7.3. Comando del LVS

Una vez que se tienen esos archivos, ya se puede realizar la verificación *LVS*. Para esto es recomendable hacer una carpeta exclusiva para que todos los archivos generados se guarden en ella. Luego de esto se recomienda colocar en esa carpeta el archivo *icv* y el *runset* del circuito en esa misma carpeta. Para realizar la prueba se necesita abrir la terminal en la carpeta creada y ejecutar el siguiente comando:

Cuadro 5
Comando para correr el *LVS*

```
icv -i archivo_gds.gds -c nombre_topcell -s cdl_netlist.icv -sf ICV -vue
    runset
```

Nota. Elaboración propia.

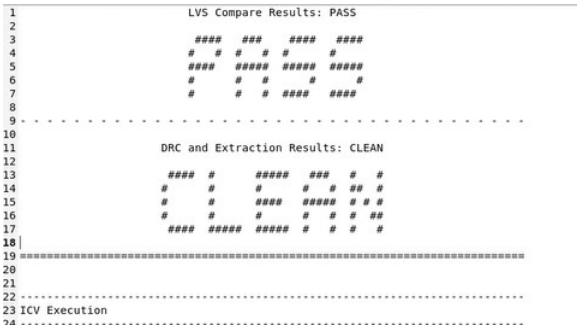
Donde el *archivo_gds.gds* hace referencia al archivo *gds*. Si este no se encuentra en la misma carpeta, es necesario especificar la ubicación completa del archivo. El *nombre_topcell* corresponde, como su nombre lo indica, al nombre de la *topcell*; el *cdl_netlist.icv* hace referencia al archivo *icv* del circuito creado en los pasos anteriores; *ICV* se queda exactamente de ese modo siempre y el *runset* hace referencia al archivo antes mencionado.

Si todo sale bien, en el archivo con extensión *RESULTS* (ver figura 9) debería salir el mensaje que se muestra en la figura 8. Si hay algún error, en lugar de *PASS* aparecerá

FAIL. Si esto sucede, se puede consultar el archivo con extensión *LVS_ERRORS* para ver el detalle del error (ver figura 9).

Figura 8

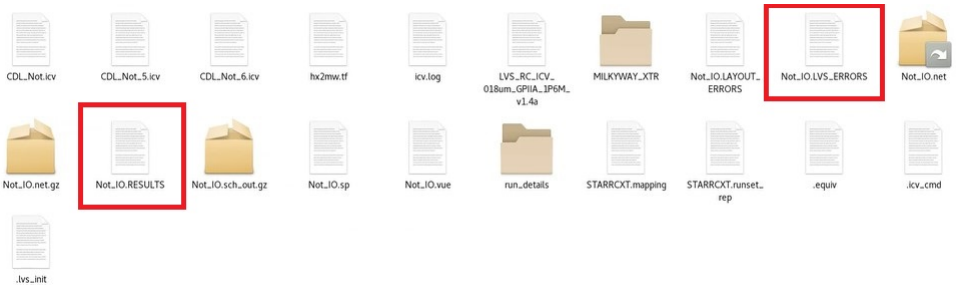
Texto que se muestra si el LVS es exitoso



Nota. Elaboración propia.

Figura 9

Archivos de resultados y errores en LVS



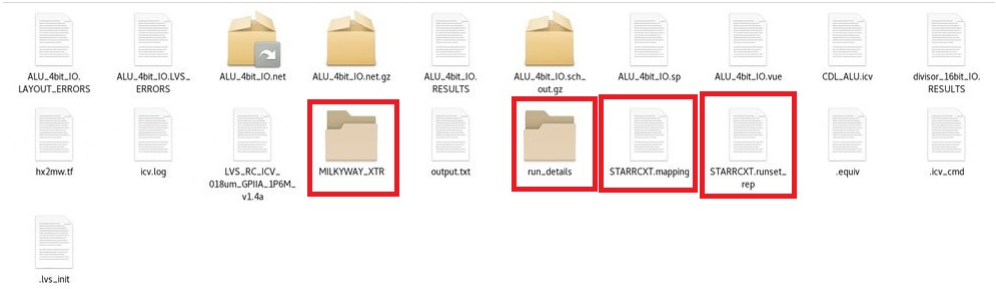
Nota. Elaboración propia.

7.4. Archivos generados

Al momento de ejecutar el LVS se obtienen varios archivos de salida. Aunque esto dependerá siempre de los archivos que se definieron para que este exporte en el archivo del *runset* descrito en capítulos anteriores (ver figura 4) En la figura 10 se muestra los distintos archivos generados y las carpetas de salida. Los archivos más importantes son los que que muestran los errores (si es que existen) para poder corregirlos lo antes posible. Adicional a esto, se puede observar en la figura 10 que hay archivos y carpetas resaltadas en rojo, estos son los archivos que son utilizados para poder generar la extracción de parásitos. Del mismo modo, se utilizan los archivos icv, gds y *runset*.

Figura 10

Archivos generados por el LVS



Nota. Elaboración propia.

7.5. Circuitos analizados para el LVS

Para poder realizar la prueba de LVS de manera adecuada se efectuaron pruebas en distintos circuitos, permitiendo analizar y desarrollar todas las partes del flujo de diseño. Estos circuitos fueron diversos y de distintas complejidades para asegurar que el proceso definido en cada etapa fuera correcto y se obtuvieran los resultados esperados en cada uno. A continuación se presentan los resultados y los layouts de los circuitos trabajados, ordenados en orden ascendente de complejidad. Todas las visualizaciones se realizaron con *IC Compiler II*, el proceso para exportar se explica más adelante.

7.5.1. Proceso para visualizar el *layout*

En la carpeta donde se encuentran los archivos generados en la síntesis física es necesario abrir una terminal y correr el siguiente comando:

Cuadro 6

Comando para iniciar *IC Compiler II*

```
icc2_shell
```

Nota. Elaboración propia.

Posteriormente aparecerá el texto que se muestra en la figura 11 diciendo que se está ejectuando *IC Compiler II*. Luego, en la terminal donde se esta ejecutando se ejecuta el siguiente comando para abrir la interfaz del programa, la cual se verá como en la figura 12.

Cuadro 7

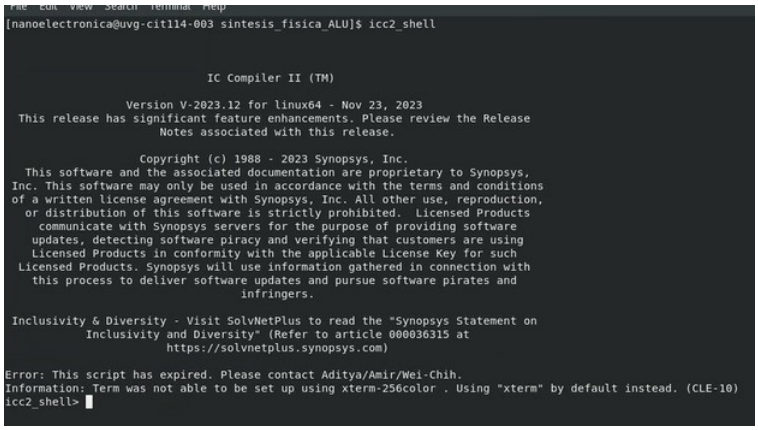
Comando para abrir la interfaz de *IC Compiler II*

```
start_gui
```

Nota. Elaboración propia.

Figura 11

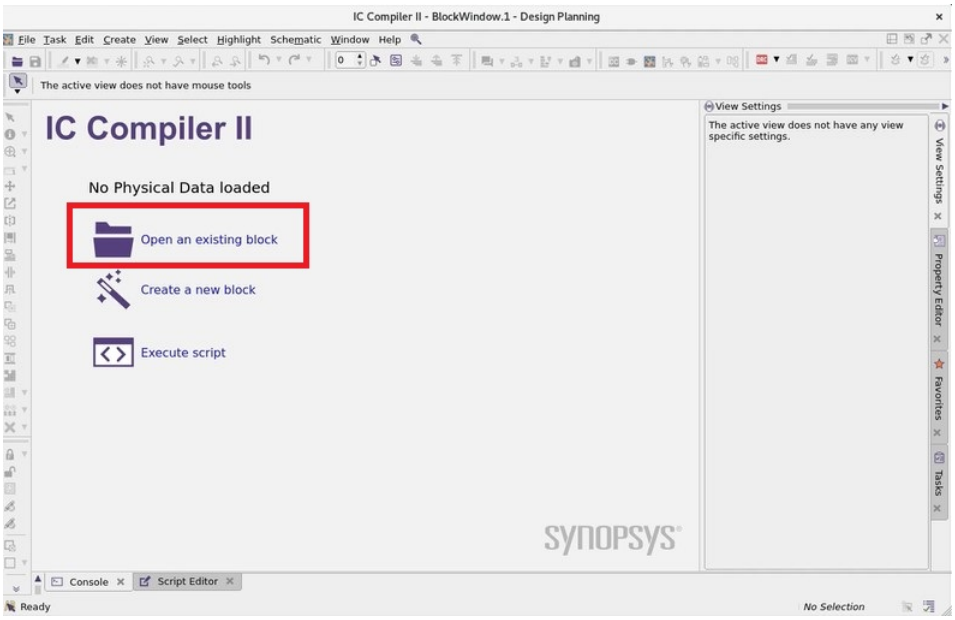
Ejecución de ICC2



Nota. Elaboración propia.

Figura 12

Interfaz de ICC2

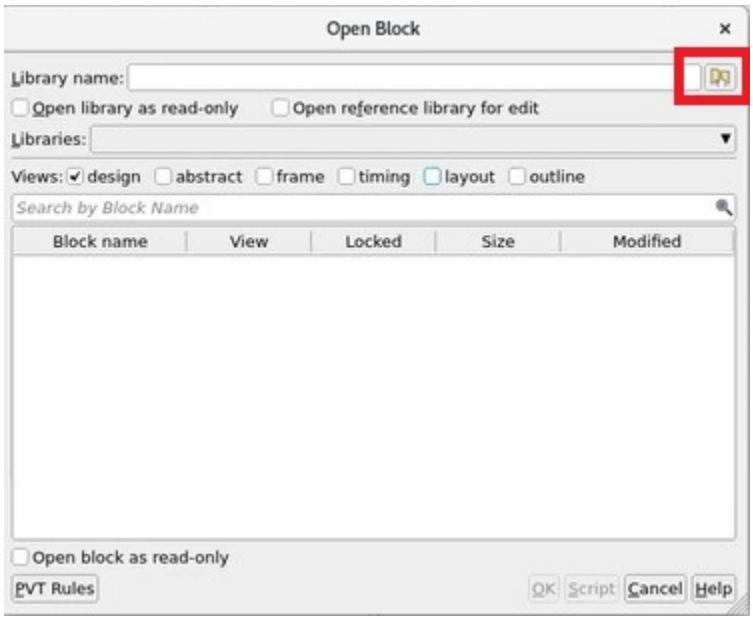


Nota. Elaboración propia.

Luego de abrir la interfaz se presiona donde se indica en la figura 12. La cual abrirá la ventana que se muestra en la figura 13. En esta ventana, es necesario dirigirse al lugar señalado en la figura anterior. Con esto se abrirá la ventana para poder escoger el archivo del bloque. Aquí se debe elegir el archivo *LIB_ TEST* como se muestra en la figura 14, verificar que también esté en el apartado *library_ name*, si se encuentra ahí, presionar el botón *choose*.

Figura 13

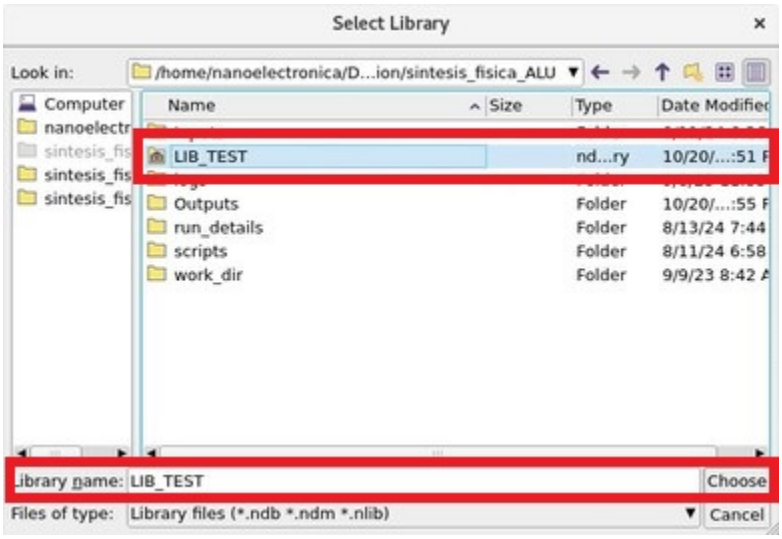
Ventana para seleccionar el bloque



Nota. Elaboración propia.

Figura 14

Explorador de archivos para seleccionar la librería

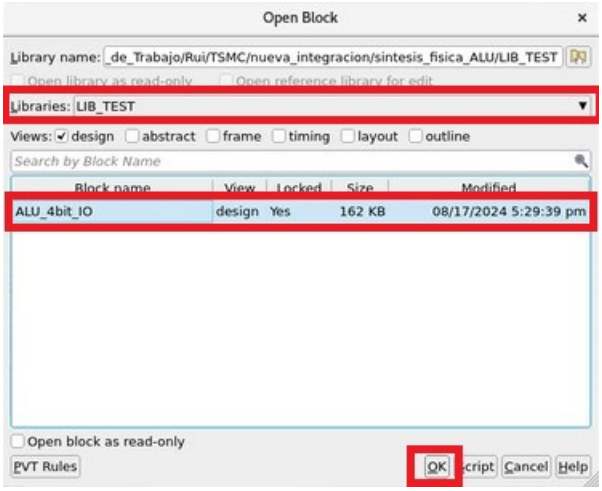


Nota. Elaboración propia.

Luego es necesario comprobar que, en la ventana, se muestre en *libraries* la librería *LIB_TEST* y que en *Block_name* aparezca el nombre de la *topcell*. Si todo está correcto, se presiona el botón *Okay* como se muestra en la figura 15. De manera siguiente, se abrirá el *layout* del circuito.

Figura 15

Librería y bloque seleccionados



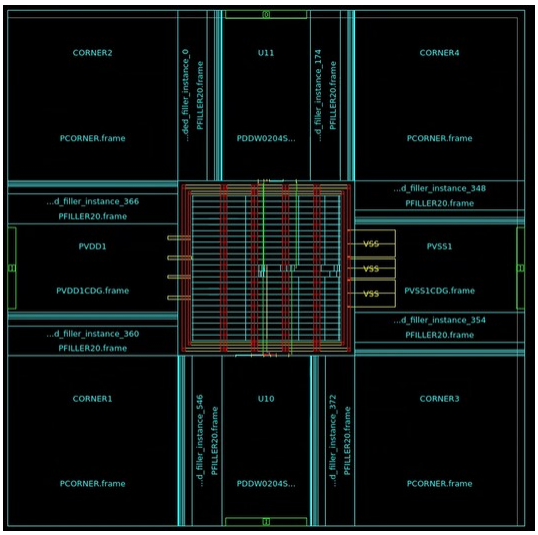
Nota. Elaboración propia.

7.5.2. Compuerta *NOT*

Esta fue la primera compuerta que pasó el proceso de síntesis física y la validación LVS. Se decidió iniciar con una compuerta *NOT* debido a que es un circuito sencillo ideal para familiarizarse con el entorno de *IC Compiler II*. Es recomendable realizar una primera prueba con este circuito debido a que es el circuito más fácil de analizar, permitiendo así un mejor entendimiento de las herramientas y haciendo que el proceso sea mucho más fácil de entender. En las figuras 16 y 17 se puede observar el *layout* y el resultado de la prueba LVS.

Figura 16

Layout compuerta *NOT*



Nota. Elaboración propia.

Figura 17

Resultado de la prueba LVS

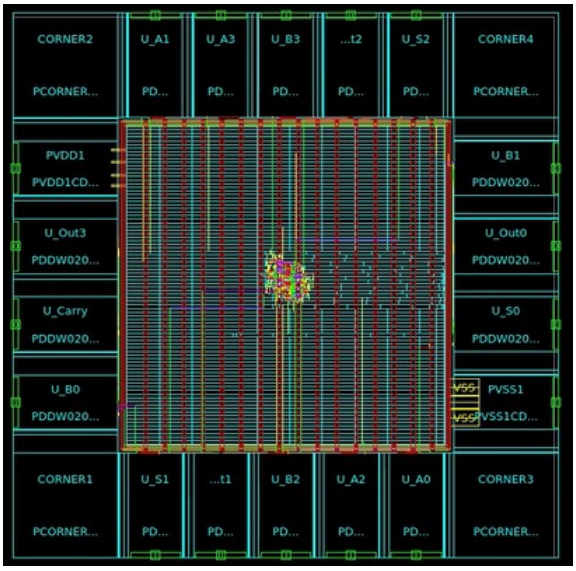
Nota. Elaboración propia.

7.5.3. Unidad aritmética lógica (ALU)

Luego de realizar la compuerta *NOT* se realizó el circuito de una unidad aritmética básica, ALU por sus siglas en inglés. Este es un circuito más complejo, con más entradas, haciendo que el *core* sea más grande que el del circuito anterior. A continuación se muestra el *layout* y el resultado exitoso de la prueba LVS.

Figura 18

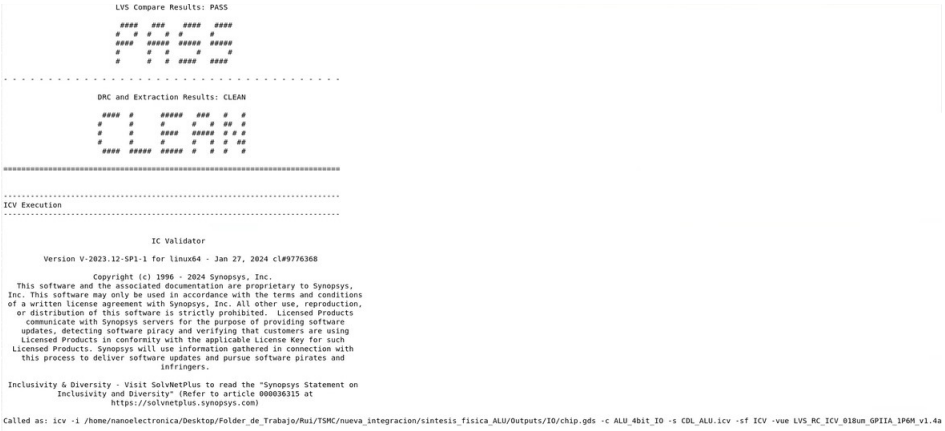
Layout ALU



Nota. Elaboración propia.

Figura 19

Resultado de la prueba LVS ALU



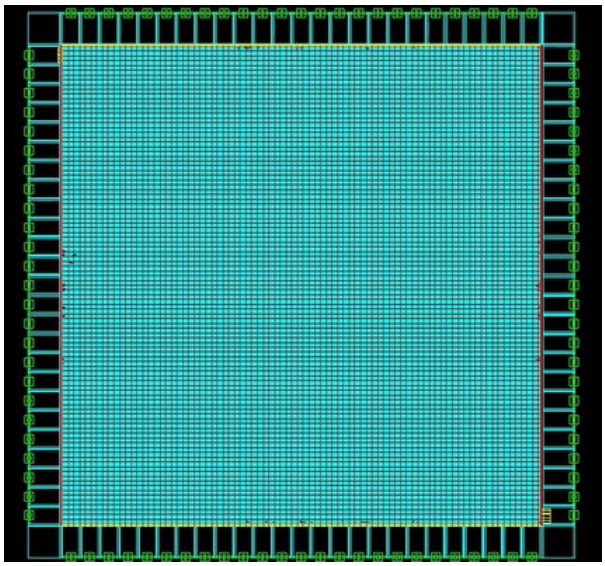
Nota. Elaboración propia.

7.5.4. *Carry look ahead*

El siguiente circuito que fue sometido a la prueba LVS fue un *carry look ahead* de 16 *bits*. Este circuito aumentó más la cantidad de entradas y salidas que se tenían, además agregó más compuertas al *core*, incrementando así la complejidad del circuito a sintetizar. En las figuras 20 y 21 se puede observar el *layout* y el resultado exitoso de la prueba LVS.

Figura 20

Layout carry look ahead



Nota. Elaboración propia.

Figura 21

Resultado de la prueba LVS del *carry look ahead*

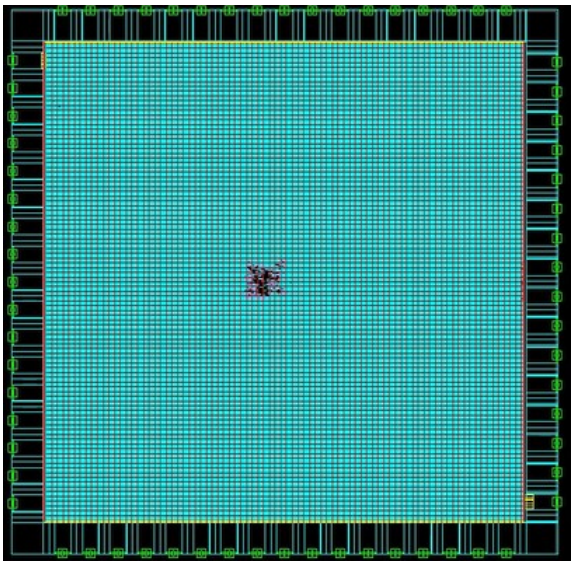
Nota. Elaboración propia.

7.5.5. Multiplicador de 16 *bits*

Posteriormente se sintetizó un multiplicador de 16 *bits*, aumentando más que todo las compuertas que se utilizaban en el *core*, pues como entradas y salidas tenían casi la misma cantidad que el sumador. Sin embargo por poseer más compuertas, aumentó del mismo modo la complejidad. En las figuras 22 y 23 se puede observar el *layout* y el resultado exitoso de la prueba LVS.

Figura 22

Layout multiplicador de 16 *bits*



Nota. Elaboración propia.

Figura 23

Resultado de la prueba LVS del multiplicador de 16 *bits*

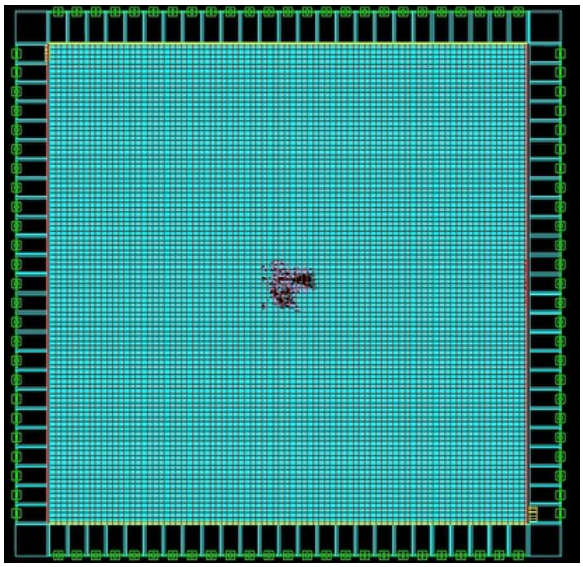
Nota. Elaboración propia.

7.5.6. Divisor de 16 *bits*

El último circuito a sintetizar, antes del circuito diseñado por el grupo de trabajo, fue un divisor de 16 *bits*, aumentando así más la complejidad. En las figuras 24 y 25 se puede observar el *layout* y el resultado exitoso de la prueba LVS.

Figura 24

Layout del divisor de 16 *bits*



Nota. Elaboración propia.

Figura 25

Resultado de la prueba LVS del divisor de 16 *bits*

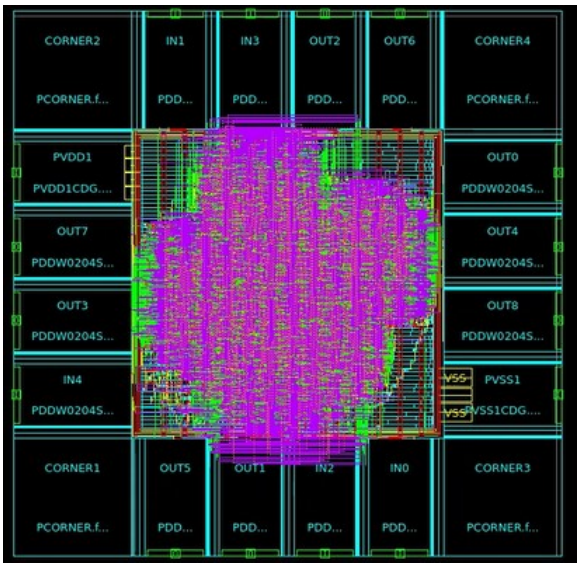
Nota. Elaboración propia.

7.5.7. El Gran Jaguar

Como etapa final se replicó la síntesis física de .[El] Gran Jaguar", con la modificación de los nombres de las personas que estuvieron involucradas en el proyecto en el año 2024. El circuito está diseñado para cumplir dos procesos. El primero es un *ring oscillator*, un circuito adicional agregado al chip completo, como un extra al funcionamiento principal. Mientras que el segundo, y más completo, es un circuito con ocho salidas. Estas están controladas por un contador para que a medida que este aumente, se enviarán configuraciones distintas en los pines de salida. Cada *set* de *bits* es específico para una letra del abecedario. El contador llegaría al número de caracteres necesarios para completar una frase completa, que se traducirá a caracteres ASCII para reproducir la frase completa en formato de audio. En las figuras 26 y 27 se puede observar el *layout* y el resultado exitoso de la prueba LVS.

Figura 26

Layout de El Gran Jaguar



Nota. Elaboración propia.

Figura 27

Resultado de la prueba LVS de El Gran Jaguar

LVS Compare Results: PASS

DRC and Extraction Results: CLEAN

ICV Execution

IC Validator

Version V-2023.12-SP1-1 for linux64 - Jan 27, 2024 cl#9776368

Copyright (c) 1996 - 2024 Synopsys, Inc.
This software and the associated documentation are proprietary to Synopsys, Inc. This software may only be used in accordance with the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, or distribution of this software is strictly prohibited. Licensed Products communicate with Synopsys servers for the purpose of providing software updates, detecting software piracy and verifying that customers are using Licensed Products in conformity with the applicable License Key for such Licensed Products. Synopsys will use information gathered in connection with this process to deliver software updates and pursue software pirates and infringers.

Inclusivity & Diversity - Visit SolvNetPlus to read the "Synopsys Statement on Inclusivity and Diversity" (Refer to article 000036315 at https://solvnetplus.synopsys.com)

Called as: icv -i /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_GRAN_JAGUAR/Outputs/IO/chip.gds -c circuit -s CDL_GRAN_JAGUAR.icv -sf ICV -vue LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a

Nota. Elaboración propia.

CAPÍTULO 8

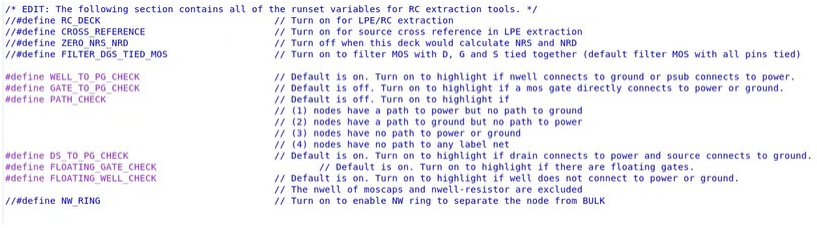
Proceso *Electrical Rule Check* (ERC)

8.1. Archivos previos

Para realizar el *Electrical Rule Check* (o ERC por sus siglas en inglés), también se necesitan los archivos *.gds*, *.icv* y el *runset* que se utilizaron para el proceso *LVS*. Sin embargo, aquí hay que realizar una modificación en el último archivo.En este es necesario volver a buscar el *ENVIRONMENT SETUP* que posee el archivo y descomentar y comentar ciertas opciones para así habilitar el proceso de ERC. Esta modificación se muestra en la figura 28.

Figura 28

Configuración a cambiar en el *runset*



```
/* EDIT: The following section contains all of the runset variables for RC extraction tools. */
//#define RC_DECK                       // Turn on for LPE/RC extraction
//#define CROSS_REFERENCE               // Turn on for source cross reference in LPE extraction
//#define ZERO_NRS_NRD                  // Turn off when this deck would calculate NRS and NRD
//#define FILTER_DGS_TIED_MOS           // Turn on to filter MOS with D, G and S tied together (default filter MOS with all pins tied)

#define WELL_TO_PG_CHECK                // Default is on. Turn on to highlight if nwell connects to ground or psub connects to power.
#define GATE_TO_PG_CHECK                // Default is off. Turn on to highlight if a mos gate directly connects to power or ground.
#define PATH_CHECK                      // Default is off. Turn on to highlight if
                                        // (1) nodes have a path to power but no path to ground
                                        // (2) nodes have a path to ground but no path to power
                                        // (3) nodes have no path to power or ground
                                        // (4) nodes have no path to any label net
#define DS_TO_PG_CHECK                  // Default is on. Turn on to highlight if drain connects to power and source connects to ground.
#define FLOATING_GATE_CHECK                   // Default is on. Turn on to highlight if there are floating gates.
#define FLOATING_WELL_CHECK             // Default is on. Turn on to highlight if well does not connect to power or ground.
                                        // The nwell of moscaps and nwell-resistor are excluded
//#define NW_RING                       // Turn on to enable NW ring to separate the node from BULK
```

Nota. Elaboración propia.

Como se ve en la figura, lo que hay que cambiar en el runset son tres cosas. La primera es comentar la línea que dice *#define RC_ DECK* y lo otro que hay que cambiar es descomentar las opciones *#define GATE_ TO_ PG_ CHECK* y *#define PATH_ CHECK*

8.2. Comando ERC

Ya que este proceso utiliza los mismos archivos que el proceso de LVS, el comando es el mismo para este proceso, siendo este:

Cuadro 8
Comando para correr *ERC*

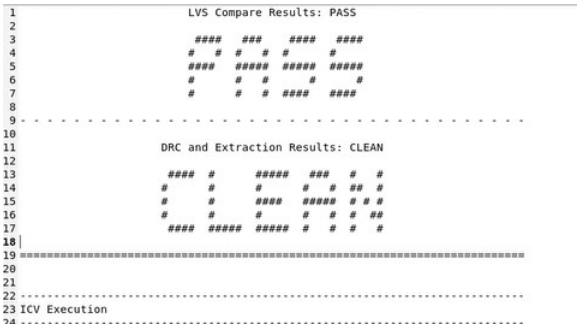
```
icv -i archivo_gds.gds -c nombre_topcell -s cdl_netlist.icv -sf ICV -vue
    runset
```

Nota. Elaboración propia.

Donde el *archivo_ gds.gds* hace referencia al archivo *gds*. Si este no se encuentra en la misma carpeta, se debe especificar el directorio donde se encuentra. El *nombre_ topcell* corresponde, como su nombre lo indica, al nombre de la *top cell*; el *cdl_ netlist.icv* hace referencia al archivo *icv* del circuito creado en los pasos anteriores del LVS; *ICV* siempre se mantiene de ese modo y el *runset* hace referencia al archivo antes mencionado.

Si todo sale bien, en el archivo con extensión *RESULTS* (ver figura 30) debería salir el mensaje que se muestra en la figura 29. Si este no se completa de foma adecuada, en el lugar donde dice *PASS*, saldría un *FAIL*. Si sucede esto último, se puede consultar el archivo con extensión *LVS_ ERRORS* para ver el detalle de los errores (ver figura 30).

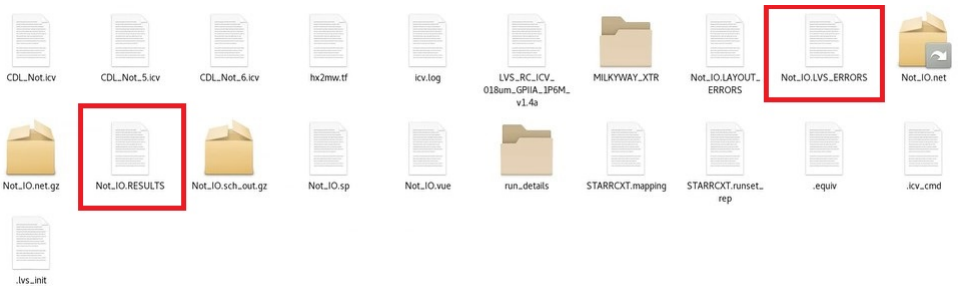
Figura 29
Texto que se muestra si el ERC es exitoso



Nota. Elaboración propia.

Figura 30

Archivos de resultados y errores



Nota. Elaboración propia.

8.3. Circuitos analizados para el ERC

Del mismo modo que el LVS, para las pruebas de ERC también se analizaron los mismos circuitos antes sintetizados. Empezando con el de menor complejidad (compuerta *NOT*) hasta llegar al de mayor complejidad ('El Gran Jaguar'). A continuación se muestran los resultados de cada circuito para la prueba de ERC.

8.3.1. Compuerta *NOT*

Figura 31

Resultado de la prueba ERC *NOT*

Nota. Elaboración propia.

8.3.2. Unidad aritmética lógica (ALU)

Figura 32

Resultado de la prueba ERC ALU

Nota. Elaboración propia.

8.3.3. *Carry look ahead*

Figura 33

Resultado de la prueba ERC del *carry look ahead*

Nota. Elaboración propia.

8.3.4. Multiplicador de 16 *bits*

Figura 34

Resultado de la prueba ERC del multiplicador de 16 *bits*

Nota. Elaboración propia.

8.3.5. Divisor de 16 *bits*

Figura 35

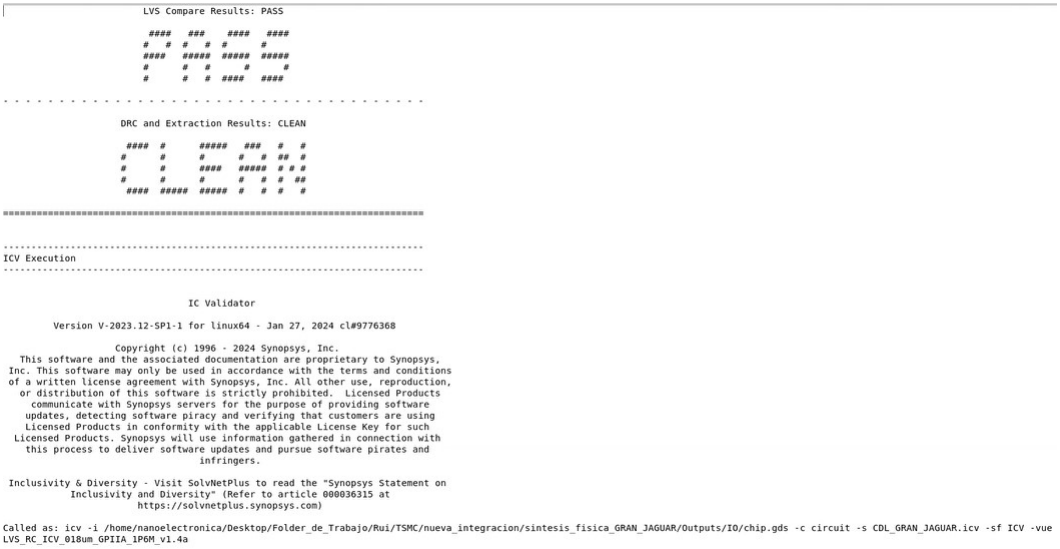
Resultado de la prueba ERC del divisor de 16 *bits*

Nota. Elaboración propia.

8.3.6. El Gran Jaguar

Figura 36

Resultado de la prueba ERC de El Gran Jaguar



Nota. Elaboración propia.

CAPÍTULO 9

Proceso de extracción parásita

9.1. Archivos previos

Para realizar la extracción parásita se necesitan los archivos generados en LVS, siendo estos los que se muestran en la figura 10. Además de esos archivos son necesarios otros dos, el primero llamado *cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd* y el otro llamado *LPE_script.cmd* (ver figura 37). El primer archivo es proporcionado por TSMC y no es un archivo de texto, por lo que no es necesario abrirlo. El segundo archivo es un script utilizado en proyectos previos, especialmente en el trabajo [9]. Este es un script que tiene las opciones que se utilizarán en la ejecución de la extracción parásita y su contenido se muestra en el cuadro 9.

La primera parte del *script* es la que indica las entradas, siendo esta el bloque la base de datos *MILKYWAY*, el archivo *mapping*, el *runset_rep*, el archivo icv (creado en la elaboración de la verificación del LVS) y el archivo proporcionado por TSMC. Luego es necesario modificar la parte de las salidas, esto mayormente solo para que las salidas sean en la extensión deseada (en este caso un sp) y con el nombre deseado, siendo recomendable ponerle el nombre del circuito realizado. Por último, lo que se puede modificar es lo de obtener una extracción con resistencias y capacitores, solo con resistencias o solo con capacitores (depende de las necesidades del circuito) y si se desea obtener un circuito más compacto, se puede utilizar la opción *HIGH* en el campo de *REDUCTION*.

En todos los años siempre ha surgido el problema de que, al trabajar con *blackboxes*, la extracción parásita no incluye pines de entrada y salida. Para resolverlo, se intenta reducir el *deck* al máximo posible para facilitar la identificación de las conexiones faltantes en las instancias de estos pines. Por este motivo, el *script* realiza una extracción solo de resistencias (sin considerar capacitores) y con una reducción alta, buscando simplificar el análisis y mapeo.

Cuadro 9
Script LPE

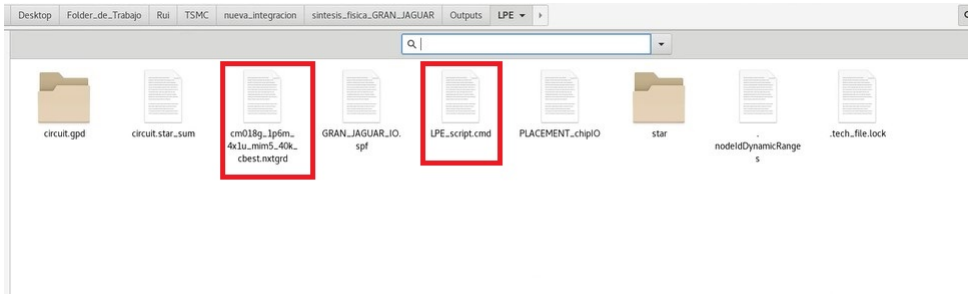
```
\*-------------------------------------------------------------\*
 ** Name: LPE_script.cmd
 ** Date: 11 Nov 2022
 ** Author: Carlos Letona
 ** Modificado: Diana Alvarado
 ** Description: The following is a command file utilized to run StarRC to
     extract
 ** 4 bit ALU characteristics to be simulated and tested.
 **
 ** Based on TSMC template made for customer reference.
 **--------------------------------INPUT-------------------------**
 BLOCK: Not_IO
 MILKYWAY_DATABASE: /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC
     /nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/LVS/MILKYWAY\
     _XTR
 TCAD_GRD_FILE: /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
     nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/LVS/STARRCXT.
     mapping
 ICV_RUNSET_REPORT_FILE: /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui
     /TSMC/nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/LVS/
     STARRCXT.runset_rep


SPICE_SUBCKT_FILE: /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
    nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/LVS/CDL_Not_6.icv


 **--------------------------------OUTPUT----------------------**
 NETLIST_FORMAT: SPF
 NETLIST_PASSIVE_PARAMS: YES
 NETLIST_FILE: Not_IO.spf
 COUPLING_REPORT_FILE: Salidas/ejercicio_cc.rep
 **-------------------------------OPTIONS-------------------------**

 PLACEMENT_INFO_FILE: YES
 PLACEMENT_INFO_FILE_NAME: PLACEMENT_chipIO
 MILKYWAY_USE_CELL_PINS: YES
 NDM_USE_DESIGN_PINS: NO
 *NET XREF WRITE LNXF

 CASE_SENSITIVE: NO
 HIERARCHICAL_SEPARATOR: /

 *MILKYWAY_EXTRACT_VIEW: YES
 *** Metal fill extraction
```

```
 * METAL_FILL_POLYGON_HANDLING: FLOATING
 * METAL_FILL_GDS_FILE:
 * GDS_LAYER_MAP_FILE:

 *** RC Extraction options
 * NETLIST_DEVICE_LOCATION_ORIENTATION : COMMENT

COUPLE_TO_GROUND: YES
 EXTRACTION: R
 REDUCTION: HIGH
 DENSITY_BASED_THICKNESS: YES
 *** For 90nm and below process
 *EXTRACT_VIA_CAPS: YES
 *** For 0.13um and above process
 EXTRACT_VIA_CAPS: NO
 *** DataBase Processing

REMOVE_FLOATING_NETS: YES
REMOVE_DANGLING_NETS: YES

 ****************************************************************
 *Original:
 POWER_NETS: VDD VSS
 * MAGNIFICATION_FACTOR : 0.9
 * MAGNIFY_DEVICE_PARAMS : NO
 SKIP_PCELLS : !*
```

Nota. Elaboración propia.

Figura 37

Archivos necesarios LPE



Nota. Elaboración propia.

9.2. Comando de extracción parásita

Una vez que se tienen esos archivos y se han realizado las modificaciones deseadas al *script*, se puede realizar la extraccción parásita. Para esto es recomendable crear una carpeta exclusiva donde se guarden todos los archivos generados, donde también se recomienda

colocar el archivo *cm018g_ 1p6m_ 4x1u_ mim5_ 40k_ cbest.nxtgrd* y el *script*. Para realizar la prueba, se necesita abrir la terminal en la carpeta creada y ejecutar el siguiente comando:

Cuadro 10
Comando para correr la extracción parásita

```
StarXtract LPE_script.cmd
```

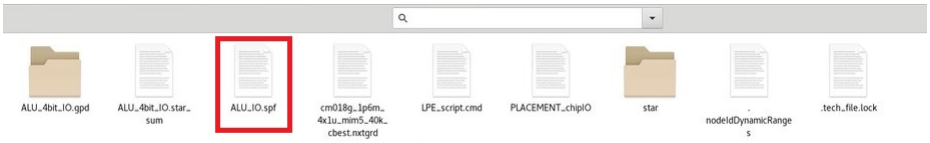
Nota. Elaboración propia.

Donde *LPE_ script.cmd* hace referencia al archivo *script* antes mencionado que si no se encuentra en la misma carpeta, se tiene que colocar la dirección donde se encuentra.

9.3. Archivos generados

Al momento de que se termina de ejecutar la extracción se obtienen algunos archivos de salida. En la figura 38 se muestran los distintos archivos obtenidos y las carpetas de salida. Sin embargo, el archivo que más nos importa de la salida es el que tiene extensión sp. Como se muestra en la figura anterior, es el archivo que esta en un rectangulo rojo.

Figura 38
Archivos generados por la extraccción parásita



Nota. Elaboración propia.

9.4. Circuitos analizados para la extraccción parásita

Del mismo modo que el LVS y el ERC, para la extarcción parásita también se analizaron los mismos circuitos antes sintetizados. Empezando con el de menor complejidad (compuerta *NOT*) hasta llegar al de mayor complejidad ('El Gran Jaguar'). En este trabajo se buscó minimizar la longitud del deck, lo cual se logró para todos los circuitos, sin embargo, se logró realizar solo un *deck* simulable, pues había que crear también los subcircuitos de cada celda que utiliza el propio circuito en *HSPICE*. En las siguientes partes se muestran todas o partes de las instancias obtenidas en la extraccción por cada circuito.

9.4.1. Compuerta NOT

El *deck* de este circuito empezó teniendo una longitud de 2415 líneas, pero se redujo a 131 con la modificación del CMD en el apartado de *REDUCTION: HIGH* en lugar de

usar *REDUCTION: YES*. Esto implica una reducción del 94.57 % en el *deck*. La parte de las instancias del *deck* es la siguiente.

Cuadro 11
Instancias obtenidas en el *deck* de la *NOT*

```
*
* Instance Section
*
XI_589437A31 N_6 I_589437A31:N_3 I_589437A31:N_4 I_589437A31:N_5 I_589437A31
    :N_6 I_589437A31:N_7 I_589437A31:N_8 PDDW0204SCDG
XI_589437A32 N_7 I_589437A32:N_3 I_589437A32:N_4 I_589437A32:N_5 I_589437A32
    :N_6 I_589437A32:N_7 N__generated_12 PDDW0204SCDG
XI_589437A33 I_589437A33:N_2 I_589437A33:N_3 I_589437A33:N_4 TIEHBWP7T
XI_589437A34 N_8 I_589437A34:N_3 I_589437A34:N_4 I_589437A34:N_5 PVSS1CDG
XI_589437A35 I_589437A35:N_2 I_589437A35:N_3 I_589437A35:N_4 I_589437A35:N_5
     CKND0BWP7T
XI_589437A36 I_589437A36:N_2 I_589437A36:N_3 I_589437A36:N_4 TIELBWP7T
XI_589437A37 N_9 I_589437A37:N_3 I_589437A37:N_4 I_589437A37:N_5 I_589437A37
    :N_6 PVDD1CDG
```

Nota. Elaboración propia.

9.4.2. Unidad aritmética lógica (ALU)

El *deck* de este circuito empezó teniendo una longitud de 22636 líneas, pero se redujo a 3039. Esto implica una reducción del 86.58 % en el *deck*. Ya que este *deck* es mucho más grande, la parte de las instancias también, así que solo se mostrará una parte. De aquí en adelante en las próximas secciones solo se pondrán partes de las instancias pues los *deck* son bastante grandes debido a la complejidad de los circuitos.

Cuadro 12
Parte de las instancias obtenidas en el *deck* de la ALU

```
*
* Instance Section
*
XI_5E585F151 I_5E585F151:N_2 I_5E585F151:N_3 I_5E585F151:N_4 I_5E585F151:N_5
     I_5E585F151:N_6 I_5E585F151:N_7 I_5E585F151:N_8 MOAI22D0BWP7T
XI_5E585F1510 I_5E585F1510:N_2 I_5E585F1510:N_3 I_5E585F1510:N_4
    I_5E585F1510:N_5 I_5E585F1510:N_6 I_5E585F1510:N_7 I_5E585F1510:N_8
    MOAI22D0BWP7T
XI_5E585F15100 I_5E585F15100:N_2 I_5E585F15100:N_3 N__generated_21224
    N__generated_21218 I_5E585F15100:N_6 N_3950 N__generated_21217
    N__generated_21227 N__generated_21221 N__generated_21215 I_5E585F15100:
    N_12 I_5E585F15100:N_13 N__generated_21214 N__generated_21226
    N__generated_21220 N__generated_21213 N__generated_21225
```

```
    N__generated_21219 N__generated_21228 N__generated_21223 I_5E585F15100:
    N_22 I_5E585F15100:N_23 N__generated_21222 N__generated_21216
    PDDW0204SCDG
XI_5E585F15101 N__generated_21235 N__generated_21241 N__generated_21237
    N__generated_21233 N__generated_21240 N_11750 N__generated_21232
    I_5E585F15101:N_9 I_5E585F15101:N_10 N__generated_21231 I_5E585F15101:
    N_12 I_5E585F15101:N_13 N__generated_21230 N__generated_21239
    N__generated_21236 N__generated_21229 N__generated_21238
    N__generated_21234 I_5E585F15101:N_20 I_5E585F15101:N_21 I_5E585F15101:
    N_22 I_5E585F15101:N_23 I_5E585F15101:N_24 I_5E585F15101:N_25
    PDDW0204SCDG
```

Nota. Elaboración propia.

9.4.3. *Carry look ahead*

El *deck* de este circuito empezó teniendo una longitud de 380447 líneas, pero se redujo a 19022. Esto implica una reducción del 95.00 % en el *deck*. La parte de algunas instancias del *deck* es la siguiente.

Cuadro 13
Parte de las instancias obtenidas en el *deck* del adder

```
XI_85CC4A6110 I_85CC4A6110:N_2 I_85CC4A6110:N_3 I_85CC4A6110:N_4
    I_85CC4A6110:N_5 I_85CC4A6110:N_6 N__generated_21724 I_85CC4A6110:N_8
    DFCND0BWP7T
XI_85CC4A6111 I_85CC4A6111:N_2 I_85CC4A6111:N_3 I_85CC4A6111:N_4
    I_85CC4A6111:N_5 I_85CC4A6111:N_6 I_85CC4A6111:N_7 I_85CC4A6111:N_8
    DFCND0BWP7T
XI_85CC4A61110 I_85CC4A61110:N_2 I_85CC4A61110:N_3 I_85CC4A61110:N_4
    I_85CC4A61110:N_5 I_85CC4A61110:N_6 CKXOR2D0BWP7T
XI_85CC4A61111 I_85CC4A61111:N_2 I_85CC4A61111:N_3 I_85CC4A61111:N_4
    I_85CC4A61111:N_5 I_85CC4A61111:N_6 CKXOR2D0BWP7T
XI_85CC4A61112 I_85CC4A61112:N_2 I_85CC4A61112:N_3 I_85CC4A61112:N_4
    I_85CC4A61112:N_5 I_85CC4A61112:N_6 CKXOR2D0BWP7T
XI_85CC4A61113 I_85CC4A61113:N_2 I_85CC4A61113:N_3 I_85CC4A61113:N_4
    I_85CC4A61113:N_5 I_85CC4A61113:N_6 CKXOR2D0BWP7T
```

Nota. Elaboración propia.

9.4.4. Multiplicador de 16 *bits*

El *deck* de este circuito empezó teniendo una longitud de 387534 líneas, pero se redujo a 23824. Esto implica una reducción del 93.85 % en el *deck*. La parte de las instancias del *deck* es la siguiente.

Cuadro 14
Parte de las instancias obtenidas en el *deck* del multiplicador

```
*
* Instance Section
*
XI_86BD02CF1 I_86BD02CF1:N_2 I_86BD02CF1:N_3 I_86BD02CF1:N_4 I_86BD02CF1:N_5
     CKND0BWP7T
XI_86BD02CF10 I_86BD02CF10:N_2 I_86BD02CF10:N_3 I_86BD02CF10:N_4
    I_86BD02CF10:N_5 CKND0BWP7T
XI_86BD02CF100 N__generated_153 N_34 N__generated_157 N__generated_151
    N__generated_160 N__generated_155 N__generated_149 N__generated_159
    N__generated_154 I_86BD02CF100:N_11 I_86BD02CF100:N_12 I_86BD02CF100:N_13
     I_86BD02CF100:N_14 I_86BD02CF100:N_15 I_86BD02CF100:N_16
    N__generated_148 N__generated_158 N__generated_152 N__generated_161
    N__generated_156 N__generated_150 PDDW0204SCDG
XI_86BD02CF101 N__generated_167 N_35 N__generated_171 N__generated_165
    N__generated_174 N__generated_169 N__generated_163 N__generated_173
    N__generated_168 I_86BD02CF101:N_11 I_86BD02CF101:N_12 I_86BD02CF101:N_13
     I_86BD02CF101:N_14 I_86BD02CF101:N_15 I_86BD02CF101:N_16
    N__generated_162 N__generated_172 N__generated_166 N__generated_175
    N__generated_170 N__generated_164 PDDW0204SCDG
XI_86BD02CF102 I_86BD02CF102:N_2 N_36 N__generated_182 N__generated_178
    N__generated_187 I_86BD02CF102:N_7 I_86BD02CF102:N_8 N__generated_186
    I_86BD02CF102:N_10 N__generated_177 N__generated_185 N__generated_180
    N__generated_176 N__generated_184 N__generated_179 I_86BD02CF102:N_17
    N__generated_183 I_86BD02CF102:N_19 I_86BD02CF102:N_20 N__generated_181
    I_86BD02CF102:N_22 PDDW0204SCDG
XI_86BD02CF103 N__generated_191 N_37 N__generated_195 N__generated_190
    N__generated_199 I_86BD02CF103:N_7 I_86BD02CF103:N_8 N__generated_198
    N__generated_194 N__generated_189 N__generated_197 N__generated_193
    N__generated_188 N__generated_196 N__generated_192 I_86BD02CF103:N_17
    I_86BD02CF103:N_18 I_86BD02CF103:N_19 I_86BD02CF103:N_20 I_86BD02CF103:
    N_21 I_86BD02CF103:N_22 PDDW0204SCDG
```

Nota. Elaboración propia.

9.4.5. Divisor de 16 *bits*

El *deck* de este circuito empezó teniendo una longitud de 391773 líneas, pero se redujo a 36383. La parte de las instancias del *deck* es la siguiente.

Cuadro 15
Parte de las instancias obtenidas en el *deck* del divisor

```
*
* Instance Section
```

```
*
XI_0426A8E71 I_0426A8E71:N_2 I_0426A8E71:N_3 I_0426A8E71:N_4 I_0426A8E71:N_5
    I_0426A8E71:N_6 I_0426A8E71:N_7 I_0426A8E71:N_8 IIND4D0BWP7T
XI_0426A8E710 I_0426A8E710:N_2 I_0426A8E710:N_3 I_0426A8E710:N_4
   I_0426A8E710:N_5 I_0426A8E710:N_6 DFQD1BWP7T
XI_0426A8E7100 I_0426A8E7100:N_2 N__generated_22950 N__generated_22942
   N__generated_22936 N__generated_22948 N__generated_22940 N_22004
   N__generated_22946 N__generated_22938 N__generated_22933
   N__generated_22944 N__generated_22937 N__generated_22932
   N__generated_22943 I_0426A8E7100:N_16 I_0426A8E7100:N_17 I_0426A8E7100:
   N_18 I_0426A8E7100:N_19 N__generated_22949 N__generated_22941
   N__generated_22935 N__generated_22947 N__generated_22939
   N__generated_22934 N__generated_22945 PDDW0204SCDG
XI_0426A8E7101 N__generated_22956 N__generated_22969 N__generated_22962
   N__generated_22955 N__generated_22967 N__generated_22960 N_22005
   N__generated_22965 N__generated_22958 N__generated_22952
   N__generated_22963 N__generated_22957 N__generated_22951 I_0426A8E7101:
   N_15 I_0426A8E7101:N_16 I_0426A8E7101:N_17 I_0426A8E7101:N_18
   I_0426A8E7101:N_19 N__generated_22968 N__generated_22961
   N__generated_22954 N__generated_22966 N__generated_22959
   N__generated_22953 N__generated_22964 PDDW0204SCDG
```

Nota. Elaboración propia.

9.4.6. El Gran Jaguar

El *deck* de este circuito empezó teniendo una longitud de 76355 líneas, pero se redujo a 56248. Esto implica una reducción del 26.33 % en el *deck*. La parte de las instancias del *deck* es la siguiente.

Cuadro 16
Parte de las instancias obtenidas en el *deck* de El Gran Jaguar

```
*
* Instance Section
*
XI_68D25EAA1 I_68D25EAA1:N_2 I_68D25EAA1:N_3 I_68D25EAA1:N_4 I_68D25EAA1:N_5
    I_68D25EAA1:N_6 I_68D25EAA1:N_7 OA21D0BWP7T
XI_68D25EAA10 I_68D25EAA10:N_2 I_68D25EAA10:N_3 I_68D25EAA10:N_4
   I_68D25EAA10:N_5 I_68D25EAA10:N_6 I_68D25EAA10:N_7 INR3D0BWP7T
XI_68D25EAA100 I_68D25EAA100:N_2 I_68D25EAA100:N_3 I_68D25EAA100:N_4
   I_68D25EAA100:N_5 I_68D25EAA100:N_6 I_68D25EAA100:N_7 I_68D25EAA100:N_8
   IND4D0BWP7T
XI_68D25EAA1000 I_68D25EAA1000:N_2 I_68D25EAA1000:N_3 I_68D25EAA1000:N_4
   I_68D25EAA1000:N_5 I_68D25EAA1000:N_6 INR2D1BWP7T
XI_68D25EAA1001 I_68D25EAA1001:N_2 I_68D25EAA1001:N_3 I_68D25EAA1001:N_4
   I_68D25EAA1001:N_5 I_68D25EAA1001:N_6 INR2D1BWP7T
```

```
XI_68D25EAA1002 I_68D25EAA1002:N_2 I_68D25EAA1002:N_3 I_68D25EAA1002:N_4
    I_68D25EAA1002:N_5 I_68D25EAA1002:N_6 INR2D1BWP7T
```

Nota. Elaboración propia.

Porcentaje de reducción

Como se puede apreciar, al reducir los *decks* se obtuvieron distintos porcentajes, pero en promedio se redujo un 80.67 % de los *decks*.

CAPÍTULO 10

Creación de un *deck* simulable

10.1. Mapeo de los pines

Como se observó en el capítulo anterior, la extracción parásita genera un *deck* sin pines de entrada y salida, lo que lo hace no simulable. Por ello, se buscó una forma de hacerlo simulable. Para hacer esto, se realizó un mapeo de los pines de manera manual. Dado que para la simulación solo se necesita el *core* de los circuitos, es necesario realizar la síntesis lógica y física únicamente con este. Luego de que ya se tiene el *deck* sin los pines de entrada y salida, es posible llevar a cabo el mapeo. Para esto, se puede usar el *verilog* que es la salida y observar las conexiones existentes para realizar los cambios necesarios en el *deck*, incluyendo la asignación de nombres a las conexiones de los pines. Si resulta útil, también se pueden graficar las celdas y sus conexiones para facilitar este proceso. Del mismo modo, se tiene que realizar un *deck* simulable borrando todas las resistencias y capacitores del mismo, pues algunas conexiones que se realizan en la extraccción no son adecuadas, lo que no permite realizar la simulación.

Además del *deck* con las instancias utilizadas para cada circuito, es necesario contar con otro archivo *.spf* con los subcircuitos de cada celda. Existe un archivo definido que incluye ciertas celdas, elaborado en el trabajo de [21]. Sin embargo, todavía existen otras celdas que necesitan ser creadas para simular cualquier circuito de manera correcta. También, se debe considerar que el mapeo y la asignación manual de pines puede ser sencillo para circuitos con una complejidad baja. Si la complejidad aumenta demasiado, realizar este proceso manualmente puede no ser una opción adecuada debido a la gran cantidad de instancias involucradas. Por lo cual, solo se realizaron *decks* de una compuerta *NOT*, un circuito *XOR* y un contador de 4 *bits*.

10.2. Circuitos exitosos

Como se mencionó anteriormente, se realizaron tres *decks* simulables, siendo estos una compuerta *NOT*, un circuito *XOR* y un contador de 4 *bits*. A continuación se describe cada uno.

10.2.1. Compuerta *NOT*

Este fue el primer circuito con el que se realizó el mappeo, debido a que es un circuito de una sola instancia. Fue el único circuito que funcionó con los componentes parásitos porque al ser solo una instancia, no existe el error humano para las conexiones manuales. En el cuadro 17 se muestra el *deck* original que no posee los pines de entrada y salida.

Cuadro 17
Deck de la compuerta *NOT* no simulable

```
*
*| DSPF 1.3
*| DESIGN Not_IO
*| DATE "Sat Nov 2 15:36:21 2024 "
*| VENDOR " Synopsys "
*| PROGRAM " StarRC "
*| VERSION "V -2023.12 - SP1"
*| DIVIDER /
*| DELIMITER :
** FORMAT SPF
*
** COMMENTS
** OPERATING_TEMPERATURE 25
** GLOBAL_TEMPERATURE 25
**
** TCAD_GRD_FILE /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
    nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/LPE/
    cm018g_1p6m_4x1u_mim5_40k_cbest . nxtgrd
** TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
** TCADGRD_VERSION 62
. SUBCKT Not_IO
*| GROUND_NET 0
*| NET N_10 0 PF
*| I ( I_589437A33 : N_3 I_589437A33 N_3 B 0 171.4800 168.1200)
*| I ( I_589437A34 : N_3 I_589437A34 N_3 B 0 232.1800 157.0000)
*| I ( I_589437A34 : N_4 I_589437A34 N_4 B 0 234.6800 173.8000)
*| I ( I_589437A34 : N_5 I_589437A34 N_5 B 0 232.1800 190.6000)
*| I ( I_589437A35 : N_2 I_589437A35 N_2 B 0 192.2000 175.9600)
*| I ( I_589437A36 : N_3 I_589437A36 N_3 B 0 173.7200 175.9600)
R2_1 I_589437A33 : N_3 I_589437A34 : N_5 16.3101
```

```
R2_2 I_589437A33 : N_3 I_589437A34 : N_3 15.18
R2_3 I_589437A33 : N_3 I_589437A34 : N_4 432.779
R2_4 I_589437A33 : N_3 I_589437A35 : N_2 10.5725
R2_5 I_589437A33 : N_3 I_589437A36 : N_3 13.6011
R2_6 I_589437A34 : N_3 I_589437A36 : N_3 24.0633
R2_7 I_589437A34 : N_3 I_589437A35 : N_2 17.638
R2_8 I_589437A34 : N_3 I_589437A34 : N_5 6.25994
R2_9 I_589437A34 : N_3 I_589437A34 : N_4 0.441559
R2_10 I_589437A34 : N_4 I_589437A36 : N_3 651.377
R2_11 I_589437A34 : N_4 I_589437A35 : N_2 476.655
R2_12 I_589437A34 : N_4 I_589437A34 : N_5 0.441555
R2_13 I_589437A34 : N_5 I_589437A36 : N_3 23.2844
R2_14 I_589437A34 : N_5 I_589437A35 : N_2 17.0112
R2_15 I_589437A35 : N_2 I_589437A36 : N_3 3.26711
R2_16 I_589437A35 : N_2 N_10 :7 0.278809
*| NET N_11 0 PF
*| I ( I_589437A33 : N_4 I_589437A33 N_4 B 0 171.4800 172.0400)
*| I ( I_589437A35 : N_3 I_589437A35 N_3 B 0 192.2000 172.0400)
*| I ( I_589437A36 : N_4 I_589437A36 N_4 B 0 173.7200 172.0400)
*| I ( I_589437A37 : N_3 I_589437A37 N_3 B 0 111.5000 194.4400)
*| I ( I_589437A37 : N_4 I_589437A37 N_4 B 0 111.5000 181.0000)
*| I ( I_589437A37 : N_5 I_589437A37 N_5 B 0 111.5000 168.1200)
*| I ( I_589437A37 : N_6 I_589437A37 N_6 B 0 111.5000 153.8400)
R3_1 I_589437A33 : N_4 I_589437A37 : N_5 33.2013
R3_2 I_589437A33 : N_4 I_589437A37 : N_3 30.8269
R3_3 I_589437A33 : N_4 I_589437A37 : N_4 33.211
R3_4 I_589437A33 : N_4 I_589437A35 : N_3 10.9164
R3_5 I_589437A33 : N_4 I_589437A37 : N_6 30.6617
R3_6 I_589437A33 : N_4 I_589437A36 : N_4 0.371745
R3_7 I_589437A35 : N_3 I_589437A37 : N_5 63.3604
R3_8 I_589437A35 : N_3 I_589437A37 : N_3 58.7625
R3_9 I_589437A35 : N_3 I_589437A37 : N_4 63.3424
R3_10 I_589437A35 : N_3 I_589437A37 : N_6 58.5485
R3_11 I_589437A35 : N_3 I_589437A36 : N_4 3.0669
R3_12 I_589437A37 : N_3 I_589437A37 : N_5 45.9763
R3_13 I_589437A37 : N_3 I_589437A37 : N_6 52.5342
R3_14 I_589437A37 : N_3 I_589437A37 : N_4 38.7333
R3_15 I_589437A37 : N_4 I_589437A37 : N_5 40.0611
R3_16 I_589437A37 : N_4 I_589437A37 : N_6 46.3492
R3_17 I_589437A37 : N_5 I_589437A37 : N_6 39.0396
*| NET N_2 0 PF
*| I ( I_589437A31 : N_4 I_589437A31 N_4 B 0 164.7600 114.5000)
*| I ( I_589437A31 : N_7 I_589437A31 N_7 B 0 181.5600 114.5000)
*| I ( I_589437A32 : N_7 I_589437A32 N_7 B 0 169.2400 233.5000)
*| I ( I_589437A33 : N_2 I_589437A33 N_2 B 0 172.8800 170.3600)
R4_1 I_589437A31 : N_4 N_2 :5 51.2485
R4_2 I_589437A31 : N_4 I_589437A31 : N_7 21.9558
R4_3 I_589437A31 : N_7 N_2 :5 53.0425
```

```
R4_4 I_589437A32 : N_7 N_2 :5 31.2247
R4_5 I_589437A33 : N_2 N_2 :5 19.278
*| NET N_3 0 PF
*| I ( I_589437A31 : N_3 I_589437A31 N_3 B 0 154.1200 114.5000)
*| I ( I_589437A31 : N_5 I_589437A31 N_5 B 0 173.1600 114.5000)
*| I ( I_589437A31 : N_6 I_589437A31 N_6 B 0 177.6400 114.5000)
*| I ( I_589437A32 : N_4 I_589437A32 N_4 B 0 186.0400 233.5000)
*| I ( I_589437A32 : N_5 I_589437A32 N_5 B 0 175.4000 233.5000)
*| I ( I_589437A32 : N_6 I_589437A32 N_6 B 0 173.1600 233.5000)
*| I ( I_589437A36 : N_2 I_589437A36 N_2 B 0 175.1200 174.2800)
R5_1 I_589437A31 : N_3 I_589437A31 : N_6 1796.49
R5_2 I_589437A31 : N_3 N_3 :8 9459.93
R5_3 I_589437A31 : N_3 I_589437A31 : N_5 19.8344
R5_4 I_589437A31 : N_5 I_589437A31 : N_6 15.9882
R5_5 I_589437A31 : N_5 N_3 :8 84.1906
R5_6 I_589437A31 : N_6 N_3 :8 73.272
R5_7 I_589437A32 : N_4 I_589437A32 : N_5 16.2921
R5_8 I_589437A32 : N_5 I_589437A36 : N_2 135.577
R5_9 I_589437A32 : N_5 N_3 :8 283.644
R5_10 I_589437A32 : N_5 I_589437A32 : N_6 14.9417
R5_11 I_589437A32 : N_6 I_589437A36 : N_2 123.822
R5_12 I_589437A32 : N_6 N_3 :8 259.05
R5_13 I_589437A36 : N_2 N_3 :8 21.9614
*| NET N_4 0 PF
*| I ( I_589437A31 : N_8 I_589437A31 N_8 B 0 189.9600 114.5000)
*| I ( I_589437A35 : N_4 I_589437A35 N_4 B 0 192.4800 173.7200)
R6_1 I_589437A31 : N_8 I_589437A35 : N_4 49.0987
*| NET N_5 0 PF
*| I ( I_589437A32 : N_3 I_589437A32 N_3 B 0 195.0000 233.5000)
*| I ( I_589437A35 : N_5 I_589437A35 N_5 B 0 193.5775 174.8400)
R7_1 I_589437A32 : N_3 I_589437A35 : N_5 49.1767
*
* Instance Section
*
XI_589437A31 N_6 I_589437A31 : N_3 I_589437A31 : N_4 I_589437A31 : N_5
    I_589437A31 : N_6 I_589437A31 : N_7 I_589437A31 : N_8 PDDW0204SCDG
XI_589437A32 N_7 I_589437A32 : N_3 I_589437A32 : N_4 I_589437A32 : N_5
    I_589437A32 : N_6 I_589437A32 : N_7 N__generated_12 PDDW0204SCDG
XI_589437A33 I_589437A33 : N_2 I_589437A33 : N_3 I_589437A33 : N_4 TIEHBWP7T
XI_589437A34 N_8 I_589437A34 : N_3 I_589437A34 : N_4 I_589437A34 : N_5
    PVSS1CDG
XI_589437A35 I_589437A35 : N_2 I_589437A35 : N_3 I_589437A35 : N_4
    I_589437A35 : N_5 CKNDOBWP7T
XI_589437A36 I_589437A36 : N_2 I_589437A36 : N_3 I_589437A36 : N_4 TIELBWP7T
.ENDS
```

Nota. Elaboración propia.

A fin de realizar la asignación de pines, se tomó como referencia el archivo *verilog* del cuadro 18. Este archivo, al contener una detallada especificación de las instancias y la secuencia de pines del diseño lógico, fue fundamental para validar y ajustar la correspondencia entre los nodos del archivo *.spf* y las definiciones del circuito extraído.

Cuadro 18
Verilog de la compuerta *NOT*

```
// IC Compiler II Version V -2023.12 Verilog Writer
// Generated on 11/9/2024 at 18:23:27
// Library Name : LIB_TEST
// Block Name : Not
// User Label :
// Write Command : write_verilog -include { all } ./Outputs/IO/Not.v
module Not ( A , Y , VDD , VSS ) ;
input A ;
output Y ;
input VDD ;
input VSS ;
supply1 VDD ;
supply0 VSS ;
CKNDOBWP7T U2 ( . I ( A ) , . ZN ( Y ) , . VDD ( VDD ) , . VSS ( VSS ) );
PVDD1CDG PVDD1 ( . VDD ( VDD ) ) ;
PVSS1CDG PVSS1 ( . VSS ( VSS ) ) ;
endmodule
```

Nota. Elaboración propia.

Luego del mapeo con la guía del *verilog*, el *deck* del *core* quedó de la suguiente forma:

Cuadro 19
Deck de la compuerta *NOT* simulable

```
*
*| DSPF 1.3
*| DESIGN Not
*| DATE "Sat Nov 9 18:46:14 2024 "
*| VENDOR " Synopsys "
*| PROGRAM " StarRC "
*| VERSION "V -2023.12 - SP1"
*| DIVIDER /
*| DELIMITER :
** FORMAT SPF
*
** COMMENTS
** OPERATING_TEMPERATURE 25
** GLOBAL_TEMPERATURE 25
**
** TCAD_GRD_FILE /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
```

```
    nueva_integracion/sintesis_fisica_not_noIO/Outputs/LPE/
    cm018g_1p6m_4x1u_mim5_40k_cbest . nxtgrd
** TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
** TCADGRD_VERSION 62
. SUBCKT Not A Y vsource
*| GROUND_NET 0
*| NET N_2 0.354649 PF
*| I ( I_C8A989271 : N_2 I_C8A989271 N_2 B 0 232.1800 157.0000)
*| I ( I_C8A989271 : Y I_C8A989271 Y B 0 234.6800 173.8000)
*| I ( I_C8A989271 : N_4 I_C8A989271 N_4 B 0 232.1800 190.6000)
*| I ( A I_C8A989273 N_2 B 0 173.7200 175.9600)
Cg2_1 I_C8A989271 : N_2 0 2.5463 e -14
Cg2_2 I_C8A989271 : N_4 0 7.21363 e -14
Cg2_3 N_2 :5 0 2.5705 e -13
R2_1 I_C8A989271 : N_2 A 25.215
R2_2 I_C8A989271 : N_2 N_2 :5 5.26195
R2_3 I_C8A989271 : N_2 I_C8A989271 : N_4 8.08502
R2_4 I_C8A989271 : N_2 I_C8A989271 : Y 0.441787
R2_5 I_C8A989271 : Y A 1386.28
R2_6 I_C8A989271 : Y N_2 :5 202.746
R2_7 I_C8A989271 : Y I_C8A989271 : N_4 0.441272
R2_8 I_C8A989271 : N_4 A 24586.8
R2_9 I_C8A989271 : N_4 N_2 :5 12.286
R2_10 A N_2 :5 3.72771
*
* Instance Section
*
XI_C8A989271 I_C8A989271 : N_2 I_C8A989271 : Y I_C8A989271 : N_4 N_4
    PVSS1CDG
XI_C8A989273 A Y vsource CKND0BWP7T
. ENDS
```

Nota. Elaboración propia.

Con esto, ya se tiene un *deck* simulable.

10.2.2. Compuerta *XOR*

El siguiente circuito al que se le realizó la extracción parásita con finalidad de realizar un *deck* simulable fue una compuerta *XOR*. Sin embargo, esa compuerta fue diseñada utilizando módulos básicos de compuertas *AND, NOT y OR*. Esto para practicar el proceso de mapeo de pines, pero con circuitos que involucraran múltiples instancias. Tras realizar el mapeo de pines en el deck con parásitos y las pruebas finales en este circuito, se observaron comportamientos extraños. A pesar de realizar revisiones en el archivo, no sé encontraron errores en el mapeo de los pines. Por esta razón, se tomó la decisión de eliminar todas los componentes parásitos y así solo validar el *core* del circuito.

En este circuito los pines también fueron mapeados empleando las instancias de los componentes definidos en el código *verilog*. Los inconvenientes detectados podrían atribuirse a múltiples factores, entre ellos, posibles errores en el archivo *CMD*, generado hace varios años. Este archivo, que contiene las instrucciones para la eliminación de ruido y las limitaciones del proceso, no fue validado en su momento debido a la escasez de pines en los circuitos, lo que imposibilitó la ejecución de simulaciones funcionales en *HSPICE* durante aquellas iteraciones.

Cuadro 20

Deck de la compuerta *XOR* simulable

```
*
*| DSPF 1.3
*| DESIGN xor_using_basic_gates
*| DATE "Tue Nov 12 12:59:40 2024 "
*| VENDOR " Synopsys "
*| PROGRAM " StarRC "
*| VERSION "V -2023.12 - SP1"
*
*| DIVIDER /
*| DELIMITER :
** FORMAT SPF
*
*
** COMMENTS
** OPERATING_TEMPERATURE 25
** GLOBAL_TEMPERATURE 25
**
** TCAD_GRD_FILE /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
    nueva_integracion/sintesis_fisica_xor_noIO/Outputs/LPE/
    cm018g_1p6m_4x1u_mim5_40k_cbest . nxtgrd
** TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
** TCADGRD_VERSION 62
. SUBCKT xor_using_basic_gates A B Y vsource
*| GROUND_NET 0
*
* Instance Section
*
XT1 NOTA B AND2 N4 N5 vsource AN2D0BWP7T
XT2 NOTB A AND1 N9 N10 vsource AN2D0BWP7T
XT3 A NOTA N13 N14 vsource CKND0BWP7T2
XT4 B NOTB N17 N18 vsource CKND0BWP7T2
XT5 AND1 AND2 Y N22 vsource OR2D0BWP7T
XT6 N23 N24 N25 N26 PVSS1CDG
. ENDS
```

Nota. Elaboración propia.

Cuadro 21
Verilog de la compuerta *XOR*

```
// IC Compiler II Version V -2023.12 Verilog Writer
// Generated on 11/12/2024 at 12:34:48
// Library Name : LIB_TEST
// Block Name : xor_using_basic_gates
// User Label :
// Write Command : write_verilog -include { all } ./ Outputs /IO/
// xor_using_basic_gates.v
module or_gate ( A , B , Y , VDD , VSS ) ;
input A ;
input B ;
output Y ;
input VDD ;
input VSS ;
supply1 VDD ;
supply0 VSS ;
OR2D0BWP7T U1 ( .A1 ( A ) , .A2 ( B ) , .Z ( Y ) , .VDD ( VDD ), .VSS ( VSS
    ) ) ;
endmodule
module and_gate_0 ( A , B , Y , VDD , VSS ) ;
input A ;
input B ;
output Y ;
input VDD ;
input VSS ;
supply1 VDD ;
supply0 VSS ;
AN2D0BWP7T U1 ( .A1 ( B ) , .A2 ( A ) , .Z ( Y ) , .VDD ( VDD ), .VSS ( VSS
    ) ) ;
endmodule
module and_gate_1 ( A , B , Y , VDD , VSS ) ;
input A ;
input B ;
output Y ;
input VDD ;
input VSS ;
supply1 VDD ;
supply0 VSS ;
AN2D0BWP7T U1 ( . A1( B ) , .A2 ( A ) , .Z ( Y ) , .VDD ( VDD ) , .VSS ( VSS
     ) ) ;
endmodule
module not_gate_0 ( A , Y , VDD , VSS ) ;
input A ;
output Y ;
input VDD ;
input VSS ;
```

```
supply1 VDD ;
supply0 VSS ;
CKND0BWP7T U1 ( . I( A ) , . ZN( Y ) , .VDD ( VDD ) , .VSS ( VSS) ) ;
endmodule
module not_gate_1 ( A , Y , VDD , VSS ) ;
input A ;
output Y ;
input VDD ;
input VSS ;
supply1 VDD ;
supply0 VSS ;
CKND0BWP7T U1 ( .I ( A ) , .ZN ( Y ) , .VDD ( VDD ) , .VSS ( VSS) ) ;
endmodule
module xor_using_basic_gates ( A , B , Y , VDD , VSS ) ;
input A ;
input B ;
output Y ;
input VDD ;
input VSS ;
wire not_A ;
wire not_B ;
wire and1 ;
wire and2 ;
supply1 VDD ;
supply0 VSS ;
not_gate_1 not_inst_A ( .A ( A ) , .Y ( not_A ) , .VDD ( VDD ) , .VSS ( VSS
    ) ) ;
not_gate_0 not_inst_B ( .A ( B ) , .Y ( not_B ) , .VDD ( VDD ) , .VSS ( VSS
    ) ) ;
and_gate_1 and_inst1 ( .A ( A ) , .B ( not_B ) , .Y ( and1 ) , .VDD ( VDD )
    , .VSS ( VSS ) ) ;
and_gate_0 and_inst2 ( .A ( not_A ) , .B ( B ) , .Y ( and2 ) , .VDD ( VDD )
    , .VSS ( VSS ) ) ;
or_gate or_inst ( .A ( and1 ) , .B ( and2 ) , .Y ( Y ) , .VDD (VDD ) , .VSS
    ( VSS ) ) ;
PVDD1CDG PVDD1 ( .VDD ( VDD ) ) ;
PVSS1CDG PVSS1 ( .VSS ( VSS ) ) ;
endmodule
```

Nota. Elaboración propia.

10.2.3. Sumador de 4 *bits*

Por último, se evaluó un sumador de 4 *bits carry look ahead* para aumentar la complejidad en el mapeo. Cabe resaltar que, al igual que la compuerta *XOR*, los resultados obtenidos con el *deck* que poseía los componentes parásitos eran extraños. Así pues, se decidió utilizar solo el *core* como en la compuerta *XOR*. Con esto los resultados si fueron los esperados.

Cuadro 22
Deck simulable sumador de 4 *bits*

```
*
*|DSPF 1.3
*|DESIGN carry_look_ahead_4bit
*|DATE "Wed Nov 13 11:57:05 2024"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "V-2023.12-SP1"
*|DIVIDER /
*|DELIMITER :
**FORMAT SPF
** COMMENTS
** OPERATING_TEMPERATURE 25
** GLOBAL_TEMPERATURE 25
**
** TCAD_GRD_FILE /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC
    /nueva_integracion/sintesis_fisica_adder4b_noIO/Outputs/LPE/
    cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
** TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
** TCADGRD_VERSION 62

.SUBCKT carry_look_ahead_4bit a0 a1 a2 a3 b0 b1 b2 b3 cin vsource s0 s1
    s2 s3 cout

*
* Instance Section
*
XI_D4A6F1241 n8 n9 s3 I_D4A6F1241:N_5 N__generated_22 vsource
    CKXOR2D0BWP7T
XI_D4A6F12410 n8 n9 b3 a3 cout I_D4A6F12410:N_7 I_D4A6F12410:N_8 vsource
    MOAI22D0BWP7T
XI_D4A6F12411 n12 n13 b1 a1 n10 I_D4A6F12411:N_7 I_D4A6F12411:N_8 vsource
     MOAI22D0BWP7T
XI_D4A6F12412 b3 a3 n9 I_D4A6F12412:N_5 I_D4A6F12412:N_6 vsource
    XNR2D1BWP7T
XI_D4A6F12413 b1 a1 n13 I_D4A6F12413:N_5 I_D4A6F12413:N_6 vsource
    XNR2D1BWP7T
XI_D4A6F1242 n11 n10 s2 I_D4A6F1242:N_5 N__generated_23 vsource
    CKXOR2D0BWP7T
XI_D4A6F1243 b2 a2 n11 I_D4A6F1243:N_5 I_D4A6F1243:N_6 vsource
    CKXOR2D0BWP7T
XI_D4A6F1244 n12 n13 s1 I_D4A6F1244:N_5 N__generated_24 vsource
    CKXOR2D0BWP7T
XI_D4A6F1245 cin n14 s0 I_D4A6F1245:N_5 N__generated_25 vsource
    CKXOR2D0BWP7T
XI_D4A6F1246 b0 a0 n14 I_D4A6F1246:N_5 I_D4A6F1246:N_6 vsource
```

```
       CKXOR2D0BWP7T
    XI_D4A6F1247 b2 a2 n10 n11 n8 I_D4A6F1247:N_7 I_D4A6F1247:N_8 vsource
       AOI22D0BWP7T
    XI_D4A6F1248 n14 cin a0 b0 n12 I_D4A6F1248:N_7 I_D4A6F1248:N_8 vsource
       AOI22D0BWP7T
    XI_D4A6F1249 N_4 I_D4A6F1249:N_3 I_D4A6F1249:N_4 I_D4A6F1249:N_5 PVSS1CDG
    .ENDS
```

Nota. Elaboración propia.

Cuadro 23
Verilog sumador de 4 *bits*

```
// IC Compiler II Version V -2023.12 Verilog Writer
// Generated on 11/13/2024 at 10:40:49
// Library Name : LIB_TEST
// Block Name : carry_look_ahead_4bit
// User Label :
// Write Command : write_verilog -include { all } ./Outputs/IO/
    carry_look_ahead_4bit.v

module carry_look_ahead_4bit ( a , b , cin , S , cout , VDD, VSS ) ;
input [3:0] a ;
input [3:0] b ;
input cin ;
output [3:0] S ;
output cout ;
input VDD ;
input VSS ;
wire n8 ;
wire n9 ;
wire n10 ;
wire n11 ;
wire n12 ;
wire n13 ;
wire n14 ;
supply1 VDD ;
supply0 VSS ;
MOAI22D0BWP7T U13 ( .A1 ( n8 ) , .A2 ( n9 ) , .B1 ( b [3] ) , .B2 ( a [3] )
    , .ZN ( cout ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
CKXOR2D0BWP7T U14 ( . A1 ( n8 ) , . A2 ( n9 ) , . Z ( S [3] ) , .VDD ( VDD )
     , .VSS ( VSS ) ) ;
XNR2D1BWP7T U15 ( . A1 ( b [3] ) , . A2 ( a [3] ) , . ZN ( n9 ) , .VDD ( VDD
     ) , .VSS ( VSS ) ) ;
AOI22D0BWP7T U16 ( . A1 ( b [2] ) , . A2 ( a [2] ) , . B1 ( n10 ) , .B2 (
    n11 ) , .ZN ( n8 ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
CKXOR2D0BWP7T U17 ( . A1 ( n11 ) , . A2 ( n10 ) , . Z ( S [2] ) , .VDD ( VDD
     ) , .VSS ( VSS ) ) ;
```

```
MOAI22D0BWP7T U18 ( . A1 ( n12 ) , . A2 ( n13 ) , . B1 ( b [1] ) , .B2 ( a
    [1] ) , .ZN ( n10 ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
CKXOR2D0BWP7T U19 ( . A1 ( b [2] ) , . A2 ( a [2] ) , . Z ( n11 ) , .VDD (
    VDD ) , .VSS ( VSS ) ) ;
CKXOR2D0BWP7T U20 ( . A1 ( n12 ) , . A2 ( n13 ) , . Z ( S [1] ) , .VDD ( VDD
     ) , .VSS ( VSS ) ) ;
XNR2D1BWP7T U21 ( . A1 ( b [1] ) , . A2 ( a [1] ) , . ZN ( n13 ) , .VDD (
    VDD ) , .VSS ( VSS ) ) ;
AOI22D0BWP7T U22 ( . A1 ( n14 ) , . A2 ( cin ) , . B1 ( a [0] ) , .B2 ( b
    [0] ) , .ZN ( n12 ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
CKXOR2D0BWP7T U23 ( . A1 ( cin ) , . A2 ( n14 ) , . Z ( S [0] ) , .VDD ( VDD
     ) , .VSS ( VSS ) ) ;
CKXOR2D0BWP7T U24 ( . A1 ( b [0] ) , . A2 ( a [0] ) , . Z ( n14 ) , .VDD (
    VDD ) , .VSS ( VSS ) ) ;
PVDD1CDG PVDD1 ( .VDD ( VDD ) ) ;
PVSS1CDG PVSS1 ( .VSS ( VSS ) ) ;
endmodule
```

Nota. Elaboración propia.

CAPÍTULO 11

Conclusiones

- Se logró replicar de manera efectiva el progreso realizado en años anteriores en el proyecto del circuito integrado, asegurando una base sólida para continuar el desarrollo.
- Las pruebas de verificación de *layout vs. schematic* (LVS) confirmaron la consistencia entre el diseño esquemático y el diseño físico.
- Las pruebas *Electrical Rules Checking* (ERC) fueron fundamentales para detectar y corregir posibles incumplimientos de normativas eléctricas.
- Se realizó un *deck* más pequeño en la extraccción parásita para un análisis de forma más fácil, reduciendo los *decks* en un 80.67 %.
- A partir del análisis de los resultados de las pruebas, se implementaron modificaciones clave en el diseño para cada citcuito, acercándolo a las especificaciones esperadas.
- Se elaboró una documentación detallada que cubre los flujos de trabajo relacionados con la verificación de *layout vs. schematic*, el *Electrical Rules Checking* y la extracción de parásitos. Este manual proporciona una guía práctica y comprensible para proyectos.

CAPÍTULO 12

Recomendaciones

- Revisar toda la documentación que sirvió de base para este trabajo, incluyendo manuales y guías relevantes.
- Mantener orden y consistencia en la organización de archivos y carpetas para facilitar el acceso y la identificación.
- Leer en su totalidad el trabajo actual para obtener una visión general del proceso y comprender el rol del LVS en el flujo de diseño.
- Actualizar el software del equipo para asegurar la compatibilidad con las herramientas y librerías utilizadas.
- Fomentar una comunicación efectiva dentro del equipo, pues todas las etapas del proyecto son importantes.
- Realizar un *script* para la automatización de la asignación de pines en los *decks* creados en la extraccción parásita.

CAPÍTULO 13

Referencias

[1] J. de los Santos, *Diseño de un sumador/restador completo de 32 bits con tecnología CMOS en un proceso de 28 nanómetros usando aplicaciones de diseño de la empresa Synopsys*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2014.

[2] L. Nájera, *Implementación de circuitos sintetizados a nivel netlist a partir de un diseño en lenguaje descriptivo de hardware como primer paso en el flujo de diseño de un circuito integrado*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2019.

[3] S. Rubio, *Definición del Flujo de Diseño para Fabricación de un Chip con Tecnologia VLSI CMOS*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2019.

[4] M. Sibrian, *Verificación de reglas de diseño (DRC) para el desarrollo de un flujo funcional de un circuito integrado con tecnología nanométrica*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2020.

[5] R. Girón, *Etapa de verificación física de Diseño en Silicio vs. Esquemático (LVS) en el flujo de diseño para un chip a nanoescala*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2020.

[6] M. Flores, *Corrección de anillo de entradas/salidas y pruebas de antenna y ERC para la definición del flujo de diseño del primer chip con tecnología nanométrica desarrollado en Guatemala*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2020.

[7] K. Cardona, *Mejoramiento del proceso de síntesis lógica llevada a cabo para la elaboración de un circuito integrado a escala nanométrica*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2021.

[8] E. Torres, *Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: Ejecución y simulación para la etapa de síntesis lógica*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2022.

[9] C. Letona, *Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: uso avanzado de StarRC para la generación de un archivo HSPICE con componentes parásitos para su correcta simulación*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2022.

[10] E. Mancio, *Implementación y verificación de la funcionalidad de código obtenido durante la síntesis lógica de arquitectura del nano chip Gran Jaguar utilizando una plataforma de desarrollo con un FPGA Xilinx Virtex-5 Genesys*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2022.

[11] L. Gómez, *Procesamiento de las señales generadas por un circuito integrado con tecnología de 180nm usando librerías de diseño de TSMC montado en un FPGA Digilent Genesys Board demostrando el funcionamiento mediante una aplicación y sintetizador digital*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2022.

[12] I. Synopsys, *Design Compiler User Guide*, Synopsys, USA, 2016, Acceso: Marzo. 26, 2024. [En línea]. Disponible: `https://picture.iczhiku.com/resource/eetop/WhIEDLIWLEUyevnv.pdf`.

[13] I. Synopsys, *VCS Funtional Verification*, Synopsys, USA, 2016, Acceso: Marzo. 26, 2024. [En línea]. Disponible: `https://www.synopsys.com/verification/simulation/vcs.html`.

[14] I. Synopsys, *IC Compiler Reference Manual*, Synopsys, USA, 2019, Acceso: Marzo. 26, 2024. [En línea]. Disponible: `https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html`.

[15] I. Synopsys, *IC Compiler Reference Manual*, Synopsys, USA, 2019, Acceso: Marzo. 26, 2024. [En línea]. Disponible: `https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/ic-validator-ds.pdf`.

[16] I. Synopsys, *Custom WaveView*, Synopsys, USA, 2013. Disponible: `https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html` (visitado 26-03-2024).

[17] I. Synopsys, *StarRC - Golden Signoff Extraction*, Synopsys, USA, 2023, Acceso: Marzo. 26, 2024. [En línea]. Disponible: `https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html`.

[18] I. Synopsys, *PrimeSim HSPICE: The Gold Standard for Accurate Circuit Simulation*, Synopsys, USA, 2023, Acceso: Marzo. 26, 2024. [En línea]. Disponible: `https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html`.

[19] I. Synopsys, *What is Electrical Rule Checking (ERC)?* Synopsys, USA, 2023, Acceso: Marzo. 26, 2024. [En línea]. Disponible: `https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html`.

[20] I. Synopsys, *What is Layout Versus Schematic Checking (LVS)?* Synopsys, USA, 2023, Acceso: Marzo. 26, 2024. [En línea]. Disponible: `https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html`.

[21] I. Arévalo, *Diseño de un circuito integrado con tecnología de 180 nm utilizando librerías de diseño de TSMC: Simulación en PrimeSim y análisis de errores*, Tesis de grado, Dpto. de Electr. Ciudad de Guatemala, Guatemala, 2022.

CAPÍTULO 14

Anexos

Cuadro 24
Celdas utilizadas en el runset para el Gran Jaguar

```
    lvs_black_box_options(
        equiv_cells ={{schematic_cell = "TIELBWP7T", layout_cell = "TIELBWP7T
            "}},
       remove_schematic_ports = {"ZN"}
);

lvs_black_box_options(
        equiv_cells ={{schematic_cell = "PVDD1CDG", layout_cell = "PVDD1CDG"
            }},
        remove_schematic_ports = {"VDD"}
);

lvs_black_box_options(
        equiv_cells ={{schematic_cell = "PVSS1CDG", layout_cell = "PVSS1CDG"
            }},
        remove_schematic_ports = {"VSS"}
);

lvs_black_box_options(
        equiv_cells ={{schematic_cell = "TIEHBWP7T", layout_cell = "TIEHBWP7T
            "}},
        remove_schematic_ports = {"Z"}
);

lvs_black_box_options(
        equiv_cells ={{schematic_cell = "PDDW0204SCDG", layout_cell = "
            PDDW0204SCDG"}},
```

```
        remove_schematic_ports = {"I", "DS", "OEN", "PAD", "C", "PE", "IE"}
);


lvs_black_box_options(
        equiv_cells = {{schematic_cell = "CKND0BWP7T", layout_cell = "
            CKND0BWP7T"}},
        remove_schematic_ports = {"I", "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "XNR2D1BWP7T", layout_cell = "
            XNR2D1BWP7T"}},
        remove_schematic_ports = {"A1", "A2","ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "IINR4D0BWP7T", layout_cell = "
            IINR4D0BWP7T"}},
        remove_schematic_ports = {"A1", "A2", "B1", "B2", "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "AN3D0BWP7T", layout_cell = "
            AN3D0BWP7T"}},
        remove_schematic_ports = {"A1", "A2", "A3" , "Z"}
);

lvs_black_box_options(
        equiv_cells ={{schematic_cell = "AO22D0BWP7T", layout_cell = "
            AO22D0BWP7T"}},
        remove_schematic_ports = {"A1", "A2" , "B1" , "B2" , "Z"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "INVD1BWP7T", layout_cell = "
            INVD1BWP7T"}},
        remove_schematic_ports = {"I", "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "MUX2ND0BWP7T", layout_cell = "
            MUX2ND0BWP7T"}},
        remove_schematic_ports = {"I0", "I1", "S", "ZN"}
);

lvs_black_box_options(
        equiv_cells ={{schematic_cell = "INR3D0BWP7T", layout_cell = "
```

```
           INR3D0BWP7T"}},
       remove_schematic_ports = {"A1", "B1" , "B2" , "ZN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "DFQD1BWP7T", layout_cell = "
           DFQD1BWP7T"}},
       remove_schematic_ports = {"D", "CP" , "Q" }
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OAI211D0BWP7T", layout_cell = "
           OAI211D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "B", "C", "ZN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "NR2D0BWP7T", layout_cell = "
           NR2D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2" , "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "AO221D0BWP7T", layout_cell = "
           AO221D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "B1", "B2" , "C" ,"Z"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "AN2D0BWP7T", layout_cell = "
           AN2D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "Z"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "IIND4D0BWP7T", layout_cell = "
           IIND4D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2" , "B1" , "B2" , "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "AOI21D0BWP7T", layout_cell = "
           AOI21D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2" , "B" , "ZN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "DFCND0BWP7T", layout_cell = "
```

```
           DFCND0BWP7T"}},
       remove_schematic_ports = {"D", "CP" , "CDN" , "Q" , "QN" }
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "INR2D0BWP7T", layout_cell = "
           INR2D0BWP7T"}},
       remove_schematic_ports = {"A1", "B1", "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OAI21D0BWP7T", layout_cell = "
           OAI21D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "B", "ZN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "NR3D0BWP7T", layout_cell = "
           NR3D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "AOI22D0BWP7T", layout_cell = "
           AOI22D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "B1", "B2", "ZN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "OAI222D0BWP7T", layout_cell = "
           OAI222D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "B1" , "B2" , "C1" , "C2" , "ZN
           "}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "AOI221D0BWP7T", layout_cell = "
           AOI221D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "B1" , "B2" , "C" , "ZN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "IND4D0BWP7T", layout_cell = "
           IND4D0BWP7T"}},
       remove_schematic_ports = {"A1", "B1" , "B2" , "B3" , "ZN"}
);

lvs_black_box_options(
```

```
        equiv_cells ={{schematic_cell = "INR4D0BWP7T", layout_cell = "
            INR4D0BWP7T"}},
        remove_schematic_ports = {"A1", "B1" , "B2" , "B3" , "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "IND3D0BWP7T", layout_cell = "
            IND3D0BWP7T"}},
        remove_schematic_ports = {"A1" , "B1", "B2" , "ZN"}
);

lvs_black_box_options(
        equiv_cells ={{schematic_cell = "AO222D0BWP7T", layout_cell = "
            AO222D0BWP7T"}},
        remove_schematic_ports = {"A1", "A2", "B1" , "B2" , "C1" , "C2" , "Z"
            }
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "CKND2D0BWP7T", layout_cell = "
            CKND2D0BWP7T"}},
        remove_schematic_ports = {"A1", "A2", "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "NR2D1BWP7T", layout_cell = "
            NR2D1BWP7T"}},
        remove_schematic_ports = {"A1", "A2" , "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "IND2D0BWP7T", layout_cell = "
            IND2D0BWP7T"}},
        remove_schematic_ports = {"A1", "B1", "ZN"}
);

lvs_black_box_options(
        equiv_cells = {{schematic_cell = "IOA21D0BWP7T", layout_cell = "
            IOA21D0BWP7T"}},
        remove_schematic_ports = {"A1", "A2", "B" , "ZN"}
);

lvs_black_box_options(
        equiv_cells ={{schematic_cell = "ND2D1BWP7T", layout_cell = "
            ND2D1BWP7T"}},
        remove_schematic_ports = {"A1", "A2" , "ZN"}
);
```

```
lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OAI22D0BWP7T", layout_cell = "
          OAI22D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "B1", "B2", "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OA31D0BWP7T", layout_cell = "
          OA31D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "B", "Z"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "NR4D0BWP7T", layout_cell = "
          NR4D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "A4" , "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "ND4D0BWP7T", layout_cell = "
          ND4D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3", "A4", "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "MOAI22D0BWP7T", layout_cell = "
          MOAI22D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2","B1", "B2", "ZN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "ND3D0BWP7T", layout_cell = "
          ND3D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "ZN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "AN4D0BWP7T", layout_cell = "
          AN4D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "A4" , "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "CKXOR2D0BWP7T", layout_cell = "
          CKXOR2D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2","Z"}
);
```

```
lvs_black_box_options(
       equiv_cells = {{schematic_cell = "HA1D0BWP7T", layout_cell = "
          HA1D0BWP7T"}},
       remove_schematic_ports = {"A", "B", "S" , "CO"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "AOI31D0BWP7T", layout_cell = "
          AOI31D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "B" , "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OA221D0BWP7T", layout_cell = "
          OA221D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2","B1", "B2", "C" , "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "CKAN2D1BWP7T", layout_cell = "
          CKAN2D1BWP7T"}},
       remove_schematic_ports = {"A1", "A2","Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "AOI33D0BWP7T", layout_cell = "
          AOI33D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "B1", "B2", "B3" , "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "MAOI22D0BWP7T", layout_cell = "
          MAOI22D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2","B1", "B2", "ZN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "OR4D0BWP7T", layout_cell = "
          OR4D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "A4" , "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "AOI32D0BWP7T", layout_cell = "
          AOI32D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "B1", "B2" , "ZN"}
);
```

```
lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OAI221D0BWP7T", layout_cell = "
          OAI221D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2","B1", "B2", "C" , "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OA22D0BWP7T", layout_cell = "
          OA22D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2","B1", "B2", "C" , "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "AOI222D0BWP7T", layout_cell = "
          AOI222D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2","B1", "B2", "C1" , "C2" , "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "INR2D1BWP7T", layout_cell = "
          INR2D1BWP7T"}},
       remove_schematic_ports = {"A1", "B1","ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OR2D0BWP7T", layout_cell = "
          OR2D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2","Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OA222D0BWP7T", layout_cell = "
          OA222D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2","B1", "B2", "C1" , "C2" , "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "AO31D0BWP7T", layout_cell = "
          AO31D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "B" , "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OA21D0BWP7T", layout_cell = "
          OA21D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2" , "B", "Z"}
);
```

```
lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OR3D0BWP7T", layout_cell = "
          OR3D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "IAO21D0BWP7T", layout_cell = "
          IAO21D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2" , "B", "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OA211D0BWP7T", layout_cell = "
          OA211D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2" , "B", "C" , "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "AO21D0BWP7T", layout_cell = "
          AO21D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2" , "B", "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "AO211D0BWP7T", layout_cell = "
          AO211D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2" , "B", "C" , "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "AOI211D0BWP7T", layout_cell = "
          AOI211D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2" , "B", "C" , "ZN"}
);


lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OAI31D0BWP7T", layout_cell = "
          OAI31D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "B" , "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "CKND1BWP7T", layout_cell = "
          CKND1BWP7T"}},
       remove_schematic_ports = {"I", "ZN"}
);
```

```
lvs_black_box_options(
       equiv_cells = {{schematic_cell = "INR2XD0BWP7T", layout_cell = "
          INR2XD0BWP7T"}},
       remove_schematic_ports = {"A1", "B1", "ZN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "DFCND1BWP7T", layout_cell = "
          DFCND1BWP7T"}},
       remove_schematic_ports = {"D", "CP" , "CDN" , "Q" , "QN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "DFCNQD1BWP7T", layout_cell = "
          DFCNQD1BWP7T"}},
       remove_schematic_ports = {"D", "CP" , "CDN" , "Q"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "CKMUX2D0BWP7T", layout_cell = "
          CKMUX2D0BWP7T"}},
       remove_schematic_ports = {"I0", "I1", "S", "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "INVD2BWP7T", layout_cell = "
          INVD2BWP7T"}},
       remove_schematic_ports = {"I", "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OAI31D1BWP7T", layout_cell = "
          OAI31D1BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "B" , "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "XNR3D0BWP7T", layout_cell = "
          XNR3D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3", "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OAI211D1BWP7T", layout_cell = "
          OAI211D1BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "B", "C", "ZN"}
);
```

```
lvs_black_box_options(
       equiv_cells = {{schematic_cell = "CKMUX2D1BWP7T", layout_cell = "
          CKMUX2D1BWP7T"}},
       remove_schematic_ports = {"IO", "I1", "S", "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "CKMUX2D0BWP7T", layout_cell = "
          CKMUX2D0BWP7T"}},
       remove_schematic_ports = {"IO", "I1", "S", "Z"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OAI32D0BWP7T", layout_cell = "
          OAI32D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "B1", "B2" , "ZN"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "LHQD1BWP7T", layout_cell = "
          LHQD1BWP7T"}},
       remove_schematic_ports = {"D" , "E" , "Q"}
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "OA32D0BWP7T", layout_cell = "
          OA32D0BWP7T"}},
       remove_schematic_ports = {"A1" , "A2" , "A3" , "B1", "B2" , "C" , "Z"
          }
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "OA32D0BWP7T", layout_cell = "
          OA32D0BWP7T"}},
       remove_schematic_ports = {"A1", "A2", "A3" , "B1" , "B2" , "Z"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "NR4D1BWP7T", layout_cell = "
          NR4D1BWP7T"}},
       remove_schematic_ports = {"A1", "A2" , "A3" , "A4" , "ZN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "DFCNQD1BWP7T", layout_cell = "
          DFCNQD1BWP7T"}},
       remove_schematic_ports = {"D", "CP" , "CDN" , "Q"}
```

```
);

lvs_black_box_options(
       equiv_cells = {{schematic_cell = "CKMUX2D0BWP7T", layout_cell = "
          CKMUX2D0BWP7T"}},
       remove_schematic_ports = {"I0", "I1", "S", "Z"}
);
lvs_black_box_options(
       equiv_cells ={{schematic_cell = "DFD1BWP7T", layout_cell = "DFD1BWP7T
          "}},
       remove_schematic_ports = {"D", "CP" , "Q" , "QN"}
);

lvs_black_box_options(
       equiv_cells ={{schematic_cell = "CKBD1BWP7T", layout_cell = "
          CKBD1BWP7T"}},
       remove_schematic_ports = {"I", "Z" }
);
```

Nota. Elaboración propia.