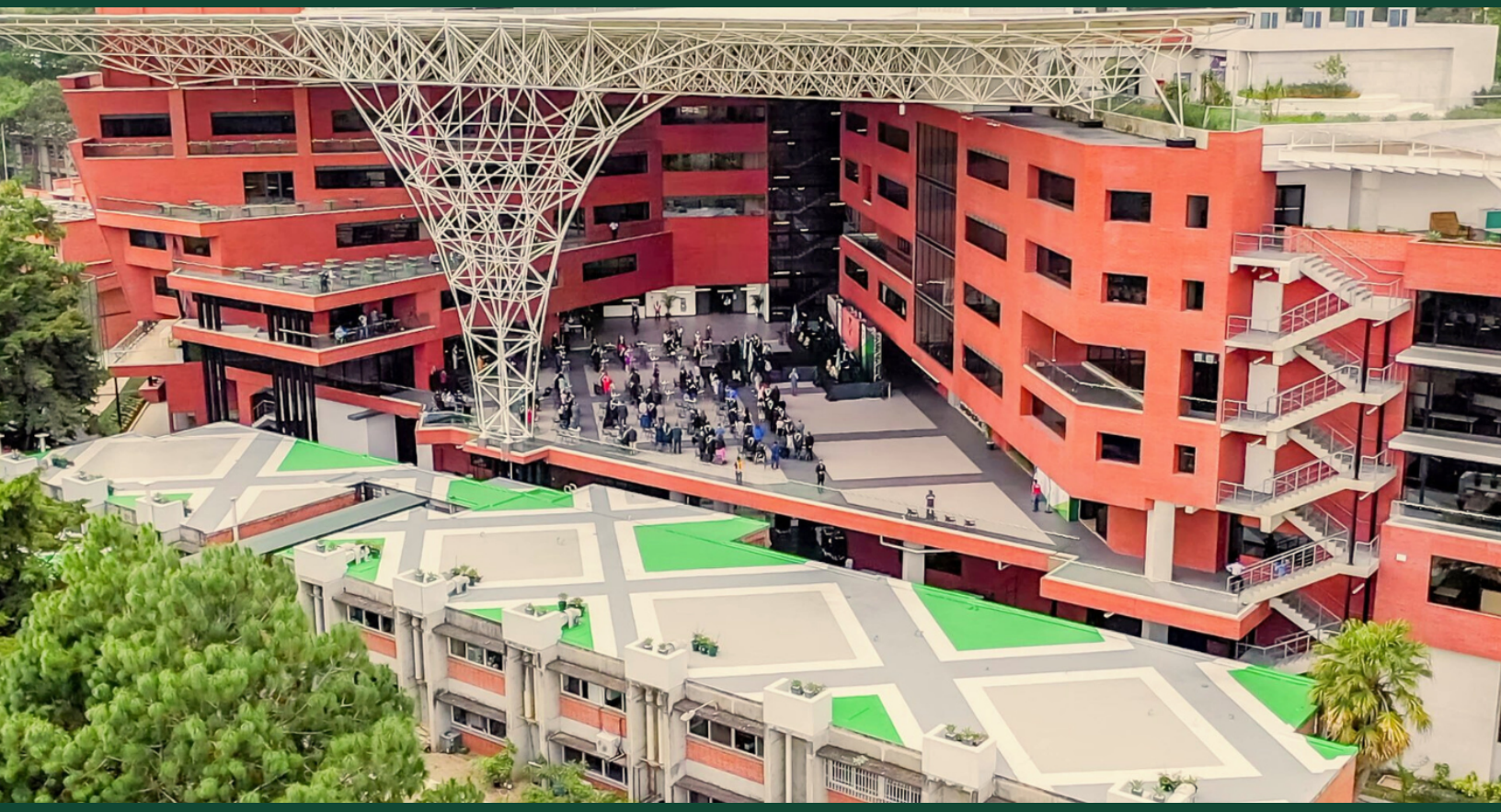
# Diseño de un circuito integrado con tecnología de 65 nm de TSMC: fase de síntesis física, verificación y generación de archivos para pruebas DRC, BND y de antena

Luis Pablo Gustavo Carranza Vásquez



UVG | UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería


# Diseño de un circuito integrado con tecnología de 65 nm de TSMC: fase de síntesis física, verificación y generación de archivos para pruebas DRC, BND y de antena

Trabajo de graduación presentado por Luis Pablo Gustavo Carranza Vásquez para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2025

Vo.Bo.:

(f) 

M.Sc. Jonathan de los Santos

(f) 

M.Sc. Carlos Esquit Hernández

Fecha de aprobación: Guatemala, 20 de noviembre de 2025.

# Prefacio

Trabajar en este proyecto ha representado para mí un desafío significativo pero también una gran oportunidad de aprendizaje y crecimiento. A lo largo de mi carrera universitaria he adquirido conocimientos en distintas áreas de la ingeniería electrónica y me considero afortunado de haber tenido la oportunidad de aplicar estos conocimientos en un proyecto tan relevante para el desarrollo tecnológico del país.

Primeramente quiero agradecer a Dios y a mi familia por el apoyo incondicional y la motivación constante que me han brindado durante todo este proceso. A mi papá, Luis Carranza, y mi mamá, Ana Cristina Vásquez, por ser mi ejemplo a seguir y por el amor incondicional que me han dado. A mis hermanas, Ana Lucía, Eliana María y Mayra Lucrecia, por siempre darme ánimo y también inspirarme a ser una mejor persona. Además, no quisiera dejar de lado al resto de mi familia que siempre han confiado en mí y han puesto sus expectativas que me impulsan a seguir buscando mi desarrollo personal y profesional.

De manera especial quisiera agradecer a Anyela González, que me ha acompañado y motivado durante gran parte de mi carrera universitaria. A mis amigos y compañeros de promoción, que se han preocupado por mí y me han permitido disfrutar de cada momento en la universidad, y también a las personas que desde fuera, me han inspirado a seguir adelante.

También quiero agradecer a la Universidad del Valle de Guatemala, por brindarme una educación de calidad, un acompañamiento constante y por darnos acceso a las herramientas y recursos necesarios para llevar a cabo este trabajo. Del mismo modo quiero expresar mi gratitud a los catedráticos que me han acompañado durante mi carrera y que han contribuido a mi formación profesional y personal; y de manera especial, a mi asesor, el Ing. Jonathan de los Santos, por su guía y apoyo durante la realización de este proyecto.

Y finalmente, quiero hacer mención de la Fundación Juan Bautista Gutiérrez, que, de no ser por su apoyo, no habría tenido la oportunidad de estudiar en tan prestigiosa universidad. Me siento honrado de haber tenido la oportunidad de adquirir la beca para mis estudios universitarios y por todo el aprendizaje, acompañamiento, apoyo y cariño que me han brindado desde el primer momento que entré en contacto con la institución.

Espero que este trabajo sirva como base para que futuros estudiantes puedan inspirarse

y comprender de mejor manera los conceptos importantes relacionados con el diseño de circuitos integrados. Y de la misma manera, que el presente trabajo pueda contribuir a la investigación y la aplicación de nuevas tecnologías que permitan el desarrollo del país.

# Índice general

# Índice de figuras

# Índice de cuadros

# Resumen

En la Universidad del Valle de Guatemala se está desarrollando el primer diseño de un nanochip, proyecto que lleva el nombre de "El Gran Jaguar". Hasta el año 2024 se trabajó con la tecnología de 180 nm de transistores de TSMC, y en el 2025 se realizó la transición a la tecnología de 65 nm.

El presente trabajo refleja una de las etapas de diseño, la cual corresponde a la síntesis física del circuito. En esta etapa se lleva a cabo la colocación de los componentes en un plano físico, el enrutamiento, pruebas de diseño (DRC y BND) y las verificaciones de antena. Esta etapa es crucial dado que en ella se genera el *layout* del circuito, y a partir de ello se hacen verificaciones para confirmar que el circuito sea fabricable y funcional. En el proyecto se establecieron los objetivos de replicar el proceso establecido en años anteriores y adaptar la síntesis física a la tecnología de 65 nm, de modo que se redujeran los errores para llevar el diseño a una etapa más cercana para la fabricación del circuito. Se realizó la réplica de los trabajos con las herramientas de IC Compiler II, Library Manager y IC Validator, y en el proceso de adaptación de la tecnología se observó que sería necesario realizar cambios en la mayor parte de los *scripts*.

Se tomaron como base los *scripts* de las metodologías de años anteriores y fue posible realizar adaptaciones y correcciones para generar resultados con la tecnología de 65 nm. De esta manera fue posible obtener los primeros resultados que contenían más de 34 mil errores debidos a la actualización de la tecnología, y al realizar una revisión de la documentación y generar cambios en los comandos y el flujo fue reducir los errores DRC del circuito del Gran Jaguar hasta en un 96 %. Además, se establecieron bases importantes para implementar las pruebas BND y generar cambios en las librerías Milkyway, lo cual también se recomienda seguir trabajando e investigando como trabajo a futuro para el desarrollo del proyecto. En el presente documento se detalla cada uno de los cambios efectuados para conocer los nuevos procesos para la ejecución de la síntesis física.

**Palabras clave:** VLSI, circuito integrado, síntesis física, *design rule checks* (DRC), *placement and routing* (P&R).

# Abstract

At the Universidad del Valle de Guatemala, the first nanochip design is being developed under the project name "El Gran Jaguar". Up until 2024, the project used TSMC's 180 nm transistor technology, and in 2025, it transitioned to 65 nm technology.

This work focuses on one of the design stages corresponding to the physical synthesis of the circuit. In this stage, the physical placement of components, routing, design checks (DRC and BND), and antenna verifications are carried out. This stage is crucial because it produces the circuit's layout, and based on it, verification steps are carried out to ensure that the circuit is manufacturable and functional. The project established the objectives of replicating the process used in previous years and adapting the physical synthesis to the 65 nm technology, with the goal of reducing errors and bringing the design closer to a stage suitable for fabrication. The replication of previous work was performed using the IC Compiler II, Library Manager, and IC Validator tools, and during the adaptation process it became clear that changes would be needed in most of the scripts.

The scripts from previous methodologies were used as a reference, and it was possible to make adaptations and corrections to obtain results using the 65 nm technology. In this way, the first results were obtained, which contained more than 34 thousand errors due to the technology update. By reviewing the documentation and applying changes to the commands and flow, it was possible to reduce El Gran Jaguar circuit's DRC errors by up to 96 %. Additionally, an important step was established bases for implementing BND checks and making changes to the Milkyway libraries, which is also recommended as future work for further advancement of the project. This document details each of the changes made in order to understand the new processes required for performing the physical synthesis.

**Keywords:** VLSI, integrated circuit, physical synthesis, design rule checks (DRC), placement and routing (P&R).

# CAPÍTULO 1

## Introducción

El diseño de circuitos electrónicos requiere de un alto nivel de complejidad, lo cual hace necesario la colaboración de un equipo multidisciplinario que permita abarcar todos los aspectos necesarios para garantizar el correcto funcionamiento de los circuitos. Actualmente, estos procesos no se realizan en Guatemala, lo cual implica un gran desafío y una gran oportunidad para el desarrollo tecnológico del país.

En el presente documento se detalla la realización de la fase de síntesis física del proyecto denominado "El Gran Jaguar". El diseño de un circuito integrado implica la realización de una serie de pasos que permiten definir todo lo necesario para garantizar que el circuito sea fabricable y que este funcione adecuadamente. En el 2019 se dio inicio al proyecto de diseño de un circuito, que corresponde a una máquina de estados finitos que almacenará la información de los nombres de cada uno de los integrantes del proyecto y una frase de presentación del proyecto para que se pueda escuchar el audio por medio de un circuito externo.

Hasta el año 2024 se trabajó con la tecnología brindada por IMEC de TSMC de 180 nanómetros, pero en este mismo año brindaron acceso a la tecnología de 65 nm, por lo cual se inició la transición de todo el trabajo que se había realizado en años anteriores. El grupo responsable del proyecto en el año 2025 formado por 5 integrantes, inició los procesos de migración de la tecnología. Se establecieron los objetivos de replicar el proceso anterior para comprender con mayor detalle cada una de las etapas y el funcionamiento de las herramientas de Synopsys, de modo que este fue el primer paso realizado y documentado. Sin embargo, durante la ejecución de esta réplica se observó que había partes del proceso que debían ser corregidas o que se podían mejorar para obtener mejores resultados.

En el año 2025 el objetivo primordial fue realizar los cambios para adecuar los procesos establecidos en años anteriores para que funcionen con la tecnología de 65 nm de TSMC, de manera que se obtuvieran resultados con la menor cantidad de errores posibles en las verificaciones DRC, antena y de LVS para acercar los trabajos a la etapa de fabricación del circuito. Es importante garantizar que el circuito tenga resultados libres de errores según los *runsets* brindados por IMEC y TSMC, dado que antes de manufacturar el circuito el

fabricante realiza sus propias pruebas para validar que se cumplan con los requisitos, y en caso no se cumplan con las reglas de diseño es posible que rechazen el proyecto y lo coloquen en una lista de espera que extiende el tiempo de manufactura. Por otro lado, también se estableció como objetivo realizar una serie de video tutoriales y manuales para permitir que los miembros futuros del equipo puedan tener una mejor base y dar seguimiento al proyecto.

Una vez culminada la realización de la réplica y finalizada la revisión bibliográfica, se dieron los preparativos para trabajar con la tecnología de 65 nm. Gracias a la relación establecida entre la Universidad del Valle de Guatemala y organizaciones como IMEC y Synopsys fue posible acceder a contenido exclusivo de las plataformas para aprender sobre la utilización de las herramientas de diseño. El equipo de trabajo pudo acceder a cursos de las herramientas de Synopsys, como Design Compiler, Library Manager, IC Compiler II, IC Validator entre otras. En este caso, los dos integrantes responsables de la fase de síntesis física se centraron en los cursos de Library Manager y de IC Compiler II. Esta parte fue fundamental en el desarrollo del proyecto, dado que la documentación de las herramientas de Synopsys incluye manuales con cantidades elevadas de páginas, lo cual puede dificultar la búsqueda de los comandos específicos para ejecutar partes puntuales del flujo de diseño.

Por medio de los cursos se determinó que era necesario realizar cambios en distintas secciones de los flujos desarrollados en años anteriores, como por ejemplo que, primero se determinó que no había que hacer ningún cambio en los valores específicos de los *runsets* de DRC y de los rellenos FEOL y BEOL, dado que estos documentos contienen datos brindados por el fabricante que se deben respetar. A partir de la investigación realizada se definió que, en lugar de realizar cambios en valores puntuales, solamente se deben hacer modificaciones en la sección de opciones y definiciones del *runset* para adecuarlos a las reglas de diseño brindadas por IMEC y TSMC. Para dar inicio al flujo de diseño es necesario seleccionar los archivos de *runsets* antes mencionados y también realizar adecuaciones a las librerías para que puedan ser utilizadas en el entorno de IC Compiler II. Una vez se establecieron las librerías y los archivos de tecnología se debe dar inicio al proceso de síntesis física, para el cual se desarrollaron 3 *scripts* principales en el proyecto que ejecutan las etapas de configuración del entorno, *floorplanning, placement, routing* y las verificaciones del diseño como DRC, BND y antena.

Es importante conocer a profundidad cuál es el orden con el que se ejecutan los comandos, dado que existen algunos que requieren de una variedad de opciones que se deben ejecutar previamente para que cumpla con su tarea de forma correcta. Además, la fase de síntesis física es crucial dado que en ella se generan las salidas del *layout* físico que se fabricará.

En esta fase del proyecto se tuvo dos principales enfoques distintos, de modo que uno de los integrantes dio inicio a la implementación de la "Metodología de Referencia" de IC Compiler II proporcionada por Synopsys que consiste en una serie de *scripts* automatizados que facilitan la realización de la síntesis física al contener todos los comandos necesarios que solamente requieren de configuraciones previas para adaptar los procesos a las recomendaciones del fabricante; y por otro lado, en el presente documento, se detalla el otro enfoque se relaciona con la adecuación de las librerías Milkyway para la utilización de los archivos brindados por IMEC en la herramienta de IC Compiler II.

Tomando estos principios como referencia, en este proyecto fue posible reducir la cantidad de errores con la tecnología de 65 nm, sin embargo, aún es necesario realizar modificaciones

para obtener resultados libres de errores y así garantizar que la fabricación del circuito sea exitosa. Se determinó que es muy importante para los próximos integrantes del proyecto que se realice una revisión exhaustiva de los cursos de Synopsys y de los manuales de las herramientas para poder realizar una implementación exitosa. En el presente documento se encuentra el detalle de todos los procedimientos realizados así como también una fundamentación teórica de los trabajos realizados y las recomendaciones para poder dar continuidad al proyecto.

# CAPÍTULO 2

# Antecedentes

En el año 2019 se realizaron las primeras investigaciones enfocadas en el desarrollo del proyecto "El Gran Jaguar" con los trabajos de graduación de los ingenieros Luis Arturo Nájera y Steven Rubio, quienes presentaron sus investigaciones tituladas "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" [1] y "Definición del flujo de diseño para fabricación de un chip con tecnología VLSI CMOS" [2], respectivamente. A partir de estas investigaciones se planteó una línea de investigación en el departamento de Ingeniería Electrónica de la Universidad del Valle de Guatemala que posteriormente abrió paso al desarrollo de un circuito integrado que implementa una máquina de estados finitos y el cual será el primer circuito integrado totalmente diseñado en Guatemala.

El proceso de diseño es realizado por diversidad de empresas y entidades, pero el presente documento se centra en los antecedentes realizados en Guatemala para el desarrollo del proyecto llamado "El Gran Jaguar".

## 2.1. Integración del diseño VLSI (*very large scale integration*)

Los primeros avances en la integración de programas curriculares relacionados con el diseño de circuitos a nanoescala en la Universidad del Valle de Guatemala se dieron a partir del año 2014, año en el que se realizó la primera investigación enfocada en el diseño VLSI realizada por el ingeniero Jonathan de los Santos [3] la cual se centró en el diseño de un sumador/restador de 32 bits completo con tecnología CMOS. El diseño VLSI es la metodología empleada para realizar el proceso de desarrollo de circuitos integrados a nanoescala, y estos avances sirvieron como base para la introducción de un curso nombrado "Introducción a sistemas de diseño VLSI", que posteriormente se dividió en dos cursos que pasaron a nombrarse "Nanoelectrónica 1 y 2" en los cuales los estudiantes adquieren los conocimientos necesarios para el uso de las herramientas de Synopsys, empresa referente en software de fabricación de circuitos de alta tecnología.

Los primeros flujos de diseño relacionados directamente con el proyecto "El Gran Jaguar" datan de las investigaciones antes mencionadas, en las cuales los ingenieros Rubio y Nájera plantean los primeros pasos del flujo de diseño de un circuito integrado [2] y el diseño de la máquina de estados que permitirá la reproducción de un mensaje con codificación ASCII [1].

En el año 2020, como recopila Kevin Hernández [4], se continuó con la implementación de más fases de diseño del circuito que son necesarias para poder manufacturar el circuito. Los trabajos de graduación realizados durante este año permitieron obtener avances significativos en la extracción de parásitos, simulaciones en HSPICE, implementación de anillos de entradas y salidas, verificaciones ERC (*electrical rule check*), DRC (*design rule check*) y verificaciones de antena.

## 2.2. Avances relacionados con la síntesis física

En el año 2021 se realizaron avances importantes en dos principales aspectos relacionados con la síntesis física y las verificaciones de diseño. Durante este año se realizaron dos trabajos de graduación en los cuales se lograron implementar mejoras en el proceso de verificación DRC, en una de ellas logrando así reducir los errores producidos durante la síntesis física [5], y en el segundo trabajo se logró seleccionar la celda adecuada para el anillo de entradas y salidas, dejando así documentación más rigurosa sobre este aspecto de la síntesis física [6].

En el año 2022 fue en el que se presentaron avances más rigurosos relacionados con esta etapa del flujo de diseño, dado que 6 estudiantes en total realizaron trabajos de graduación enfocados a la realización de la síntesis física. Los ingenieros Luis Abadía y Antonio Altuna trabajaron en el diseño riguroso del *floorplan* [7] y en el desarrollo de pruebas con distintos circuitos para las verificaciones de *design rule check, antenna rule check y electrical rule check*, de modo que probaron una compuerta NOT, una XOR, un *full adder*, una unidad lógica y aritmética (ALU), un contador de 4 bits, una memoria RAM y un circuito adicional personalizado [8].

Durante este año, el ingeniero José Ayala [9] centró su tesis en la implementación de los procesos en IC Compiler II, dado que en años anteriores habían utilizado IC Compiler. Uno de los objetivos de este trabajo era automatizar algunos de los avances obtenidos en años anteriores, así como también revisar los manuales brindados por la empresa Synopsys para identificar mejoras en el diseño. En el trabajo lograron identificar que era necesario realizar algunas modificaciones en el flujo de diseño que se había estado empleando, por lo cual en la etapa de síntesis física, además de utilizar el *floorplan, placement y routing*, añadieron procesos como lo fueron el *power planning* y la preparación de librerías.

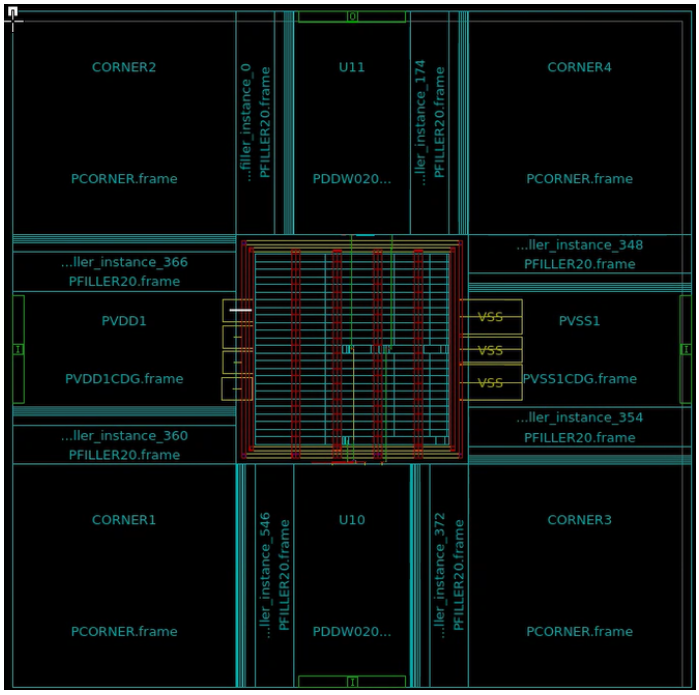
El ingeniero Julio Shin [10] obtuvo avances principalmente en las pruebas ERC, así como también en el desarrollo de los *scripts* que permitieron la automatización del proceso de la síntesis física. Por su parte Diego Esquité, se centró en la realización de un código en Python 3 que también está relacionado con la automatización, mientras que indica que fue posible también implementar el comando **signoff** el cual también permitió solucionar algunos de los errores de densidad que no se habían podido solucionar en años anteriores [11]. Finalmente, durante este año, Stefan Schwendener logró la automatización de más procesos, pero esta vez

enfocados a las verificaciones, de manera que detalla la generación de *scripts* relacionados con pruebas DRC, ARC (*antenna rule check*), ERC y LVS (*layout vs schematic*).

Durante el año 2023 fue principalmente uno de los trabajos de graduación el que se enfocó en procesos relacionados con la síntesis física, y fue el desarrollado por el ingeniero Carlos Letona [12], quien utilizó la herramienta StarRC para la generación de los *decks* resultantes del proceso de extracción de parásitos.

En el año 2024 fue publicado el trabajo de graduación del ingeniero Noel Prado, quien realizó pruebas con un *runset* enfocado para mejorar los problemas de errores de densidad; sin embargo, no logró obtener los resultados esperados ya que, si bien redujo la cantidad de errores, no los eliminó [13]. En su tesis también describe una mejora en la simplificación de la jerarquía de directorios utilizados en el proceso de diseño del circuito.

**Figura 1.** Resultado de la síntesis física de una compuerta NOT en IC Compiler II

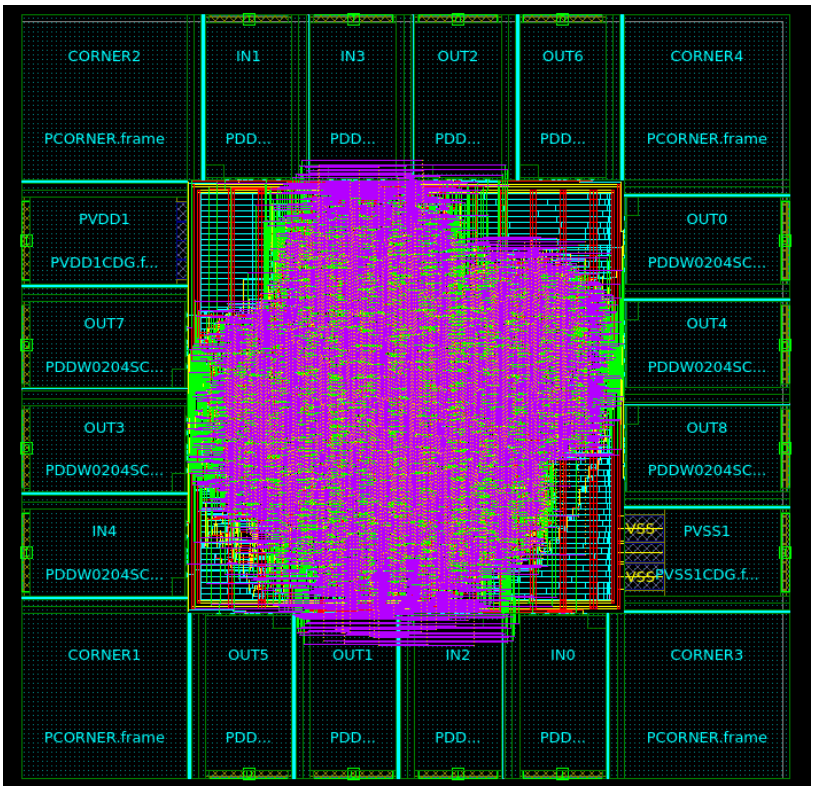



Nota. La imagen muestra el resultado de la síntesis física de una compuerta NOT, la cual es un circuito básico que sirve como base para la implementación de circuitos más complejos.

Finalmente, los avances más recientes realizados por estudiantes en sus trabajos de graduación se realizaron en el 2024. En este año 5 personas integraron el equipo que se enfocó en el desarrollo del proyecto de "El Gran Jaguar", y este año también representó desafíos importantes ya que dieron inicio a la transición del proyecto a la tecnología de 65 nm. De esta manera, los estudiantes que se centraron en la síntesis física y verificaciones fueron Diana Alvarado, que se enfocó en las pruebas ERC, LVS y extracción de parásitos [14], y Lourdes Ruiz, que tuvo como foco la implementación de la síntesis física, verificaciones DRC y de antena, iniciando también con el traslado a tecnología de 65 nm con una gran variedad de

circuitos [15]. Inicialmente realizaron los procesos de síntesis para circuitos simples como el de la Figura 1 en la cual se muestra el resultado de una compuerta NOT, para eventualmente familirizarse con el proceso y llegar al resultado que se muestra en la Figura 2 donde se muestra el resultado de la síntesis física del circuito integrado "El Gran Jaguar".

**Figura 2.** Síntesis física del circuito integrado "El Gran Jaguar"




Nota. La imagen muestra el resultado de la síntesis física del circuito integrado "El Gran Jaguar", el cual es el diseño más complejo que se ha realizado a la fecha en UVG.

Dado que el presente trabajo de graduación se presenta con el objetivo de implementar la síntesis física con tecnología de 65 nm, una de las mayores referencias es el trabajo de Lourdes Ruiz, dado que recopila y sintetiza los avances más importantes relacionados con este proceso hasta la actualidad, y también sentó bases para utilizar el diseño con la tecnología de 65 nm.

# CAPÍTULO 3

## Justificación

Los circuitos integrados han sido parte vital del desarrollo de la humanidad, siendo así que actualmente, es posible observar circuitos integrados en una gran variedad de aplicaciones, desde juguetes electrónicos hasta dispositivos de alta tecnología, como los utilizados en la exploración espacial.

Los circuitos integrados son distribuidos o bien adquiridos en todos los países del mundo; sin embargo, son pocos los países que tienen la capacidad de diseñarlos y producirlos a gran escala. Acceder a las tecnologías de diseño y fabricación de circuitos integrados presenta un gran desafío para países como Guatemala, tanto por los costos que implica como por el bajo conocimiento que existe actualmente en el país sobre este ámbito. Por este motivo, indagar y propagar el conocimiento adquirido gracias a las oportunidades brindadas en la Universidad del Valle de Guatemala por medio de IMEC y TSMC es de vital importancia para el desarrollo del país.

La Universidad del Valle de Guatemala ha sido pionera en la enseñanza y en el desarrollo de programas académicos relacionados con la fabricación y diseño de circuitos a nanoescala, siendo así que desde el año 2019 se ha realizado una serie de investigaciones y proyectos relacionados con el primer circuito integrado diseñado en Guatemala, el cual llevará el nombre de "El Gran Jaguar". Hasta el presente año se ha trabajado en el desarrollo de este proyecto con la tecnología de TSMC de 180 nm, por lo cual el siguiente paso es adecuar el proceso empleado para cambiar el diseño del circuito a una tecnología de 65 nm.

Es por ese motivo que el presente trabajo busca comprender cada fase del flujo de diseño de un circuito a nanoescala y se centra principalmente en el desarrollo del proceso de síntesis física, pruebas de antena, verificaciones DRC, BND y la generación de todos los archivos necesarios para proceder con los pasos finales de preparación para la manufactura con la tecnología de 65 nm de TSMC por medio del uso de las herramientas de Synopsys.

También es importante considerar que una de las pruebas que hasta el momento no se ha generado es la de BND (pruebas sobre la conexión con el empaquetado), y será parte importante del proyecto ya que según la documentación brindada por IMEC [16] se necesita

cumplir con este requisito para poder realizar la manufactura del circuito a la mayor brevedad posible. Esto presentará retos porque implica realizar cambios en los *scripts* que se habían desarrollado hasta el presente año y en los cuales ya se tenían automatizados varios procesos. Así como también se necesita tomar en cuenta que otros detalles como que, según el manual de tecnología brindado por IMEC, es necesario que los *corners* tengan cortes de 45° en sus esquinas dado que se necesita tener un área libre para que TSMC pueda colocar correctamente el anillo de protección *sealring* en el diseño para el proceso de manufactura. En caso no se cumplan con estos requisitos se obtendrán violaciones en las reglas de diseño que impedirán que TSMC pueda aprobar la fabricación del circuito.

Finalmente, cabe destacar que otro de los aspectos que se determinó relevante es la necesidad de hacer la transición de las librerías del entorno de Milkyway a IC Compiler II. Al realizar parte del proceso fue posible identificar que algunos de los documentos utilizados que fueron tomados del repositorio brindado por IMEC corresponde a una carpeta con nombre Milkyway. Al hacer la revisión de los archivos fue posible validar que son librerías que originalmente se diseñaron para el entorno de Milkyway, la cual es una herramienta predecesora de IC Compiler II. Esto tiene relevancia dado que puede ser la explicación de algunas de las advertencias que se generan durante la creación de librerías, por lo cual es necesario realizar la exploración de los comandos que permiten hacer la traducción de las librerías para el nuevo entorno y también implementar las nuevas librerías en el flujo de diseño de la síntesis física.

# CAPÍTULO 4

# Objetivos

## 4.1. Objetivo general

Comprender a profundidad y definir el proceso necesario para desarrollar la síntesis física del circuito "El Gran Jaguar" con la tecnología de 65 nm de TSMC, así como efectuar las pruebas necesarias como DRC, BND y verificaciones de antena que ayudarán a garantizar la viabilidad de fabricación con esta tecnología previo a la manufactura del circuito integrado.

## 4.2. Objetivos específicos

- Replicar los avances del circuito integrado realizados en años anteriores para aprender a utilizar las herramientas y tener claridad en los pasos del flujo de diseño.
- Definir e implementar los cambios necesarios en librerías y procesos del flujo de diseño empleado para adecuarlo a la tecnología de 65 nm.
- Aportar en la reducción de los errores obtenidos en el resultado de la prueba DRC usando nuevas técnicas de diseño recomendadas por IMEC y Synopsys.
- Desarrollar e implementar los *scripts* para realizar las verificaciones DRC y las mejoras en el *layout* del circuito para cumplir con los requisitos y recomendaciones brindados por IMEC.
- Adaptar las librerías Milkyway al entorno de IC Compiler II.
- Realizar la documentación necesaria para que el proceso sea replicable y fácilmente implementable en futuras investigaciones.

# CAPÍTULO 5

## Alcance

El proyecto se enfoca en el proceso de síntesis física de diferentes circuitos nanométricos como preparación para el desarrollo del proyecto de "El Gran Jaguar". Se tomará como base los procedimientos definidos en trabajos anteriores que se realizaron con las librerías de 180 nm de TSMC, pero se contemplan las modificaciones pertinentes tanto en los *scripts* como en las librerías para adaptarlos a la tecnología de 65 nm que fue proporcionada por IMEC.

Como se indica en la sección correspondiente, los objetivos del trabajo serán primeramente replicar los avances de trabajos de graduación de años anteriores, diseñar e implementar los cambios para adecuar el circuito a la tecnología de 65 nm, la obtención de resultados favorables en verificaciones DRC y BND para evitar errores futuros en la manufactura del circuito integrado, y en la documentación adecuada de los procesos para facilitar la replicación del proceso en futuras investigaciones. Es importante considerar que debido a que se realizarán cambios en la tecnología, no se puede garantizar que los resultados estén completamente libres de errores y estos también se limitan a las verificaciones que se pueden implementar en el laboratorio, por lo cual los resultados de IMEC posteriores al envío de los archivos para la manufactura, pueden variar respecto a los obtenidos en el laboratorio.

El desarrollo del proyecto se enfocará únicamente en sintetizar diversos circuitos para poder llevar a cabo posteriormente la síntesis del circuito de "El Gran Jaguar". La ejecución de los *scripts* permitirá la obtención de los archivos que servirán como base para procedimientos posteriores como lo pueden ser las pruebas de LVS y la generación de archivos finales para la manufactura del circuito integrado.

Para la ejecución del proyecto se utilizarán las herramientas de Synopsys como lo son IC Compiler II, IC Validator, y los archivos y documentación proporcionada por IMEC correspondientes a la tecnología de 65 nm de TSMC, los cuales son accesibles para los integrantes del proyecto a través de los recursos disponibles en los laboratorios de la Universidad del Valle de Guatemala. Además, cabe resaltar que la universidad cuenta con recursos como lo son las computadoras que poseen las máquinas virtuales de Rocky Linux que ya tienen integradas los programas de Synopsys y por medio de ellas tenemos acceso a las plataformas que cuentan con toda la documentación como lo puede ser SolveNet y la documentación

previamente descrita.

Dada la cantidad de documentación disponible, el proyecto contará con una fase profunda de investigación y aprendizaje para determinar los cambios necesarios en los procesos definidos en años anteriores, considerando que se espera terminar los procedimientos descritos en el presente año 2025.

# CAPÍTULO 6

## Marco teórico

## 6.1. Visión general del flujo de diseño de un circuito

El flujo de diseño VLSI es un proceso estructurado mediante el cual se diseña un circuito integrado complejo, el cual va desde la especificación funcional y estructura interna hasta su fabricación. En el contexto de la nanoelectrónica, este proceso adquiere una gran importancia debido a los desafíos adicionales que implican las escalas nanométricas, como lo pueden ser problemas relacionados con efectos cuánticos, la variabilidad del proceso o la disipación térmica y de potencia que se da en los circuitos integrados.

En 1965 Gorden Moore observó que la cantidad de transistores que poseía un *chip* se duplicaba aproximadamente cada 18 meses, lo cual seguía una tendencia semilogarítmica [17, p. 2]. Este enunciado procedió a llamarse la ley de Moore, y esto explica la necesidad del flujo de diseño VLSI, dado que pasó de utilizarse flujos denominados SSI (*small scale integration*) con circuitos de a penas 10 compuertas, a utilizarse flujos denominados MSI (*medium scale integration*), LSI *large scale integration* y VLSI (*very large scale integration*) con circuitos que contienen millones de compuertas [17, p. 3].

Este proceso por lo general es realizado por dos grandes grupos de trabajo, los cuales son el equipo de diseño y el equipo, empresa u organización que se encarga de la manufactura y los procesos de fabricación [18, p. 3]. El fabricante necesita brindar a sus clientes las reglas o requerimientos de diseño, dado que es necesario especificar las dimensiones como espacios mínimos entre rutas, metales, dopajes y todo lo necesario para garantizar el funcionamiento y correcta fabricación de un circuito.

Según [18] algunos aspectos importantes durante la etapa de diseño son la tecnología, el diseño funcional (*behavorial design*), el diseño estructural (*structural-architectural design*), el diseño lógico, el diseño físico y las simulaciones.

El proceso de manufactura de circuitos conlleva una gran serie de pasos, por lo que se suele utilizar la ideología conocida informalmente como "divide y vencerás", de modo que se

realizan varios grupos de trabajo que realizarán cada paso del diseño VLSI [18, p. 4].

## 6.2. Proceso de síntesis física

La síntesis física convierte el conjunto de *netlists* lógicas en una representación física del circuito que puede ser manufacturada en silicio. Este proceso se realiza con el fin de poder fabricar las máscaras que contienen las formas geométricas, ubicaciones y conexiones de los dopajes, conectores y metales [18].

En esta etapa se utilizan los resultados de la etapa anterior también denominada *front-end*, de modo que se obtiene un archivo denominado GDS el cual posee la información necesaria sobre las conexiones y componentes del circuito en silicio [10].

### 6.2.1. *Partitioning*

Este proceso consiste en dividir el diseño en módulos más simples y manejables debido a que son más pequeños y se pueden analizar de mejor manera [15, p. 11] sin perder la conectividad establecida en el circuito original y permitiendo optimizar el manejo de los circuitos. En circuitos integrados complejos, esta división ayuda a reducir la dificultad que conlleva realizar el enrutamiento y a mejorar la escalabilidad del diseño. Además, este proceso también es útil para soportar el diseño jerárquico y facilitar el trabajo paralelo de varios equipos de diseño.

### 6.2.2. *Floorplanning*

El *floorplanning* define la organización espacial de los bloques del diseño [15, p. 12] del circuito completo. En esta parte se define en dónde se colocarán los módulos principales, también dónde estarán las regiones o nodos de alimentación, pines de entrada y salida, la orientación y el tamaño de los bloques y todos los componentes necesarios para el funcionamiento del circuito.

Este paso del diseño es fundamental para evitar la congestión de rutas y asegurar un buen balance entre área, rendimiento y disipación térmica. De esta manera se puede optimizar el tamaño del chip y los largos de los *interconnects*. En la actualidad, existe una variedad de algoritmos de *floorplanning*, y el objetivo principal de estos métodos es reducir el costo de *floorplan* [19], para lo cual hay ramas específicas de investigación y desarrollo que han optimizado estos procesos.

### 6.2.3. *Placement*

Como el nombre de esta etapa lo indica, este procedimiento coloca o ubica físicamente las celdas estándar *standard cells* de forma óptima dentro del área de cada bloque [15, p. 12], sin conectar aún las rutas metálicas. El objetivo de esta etapa es minimizar la longitud total

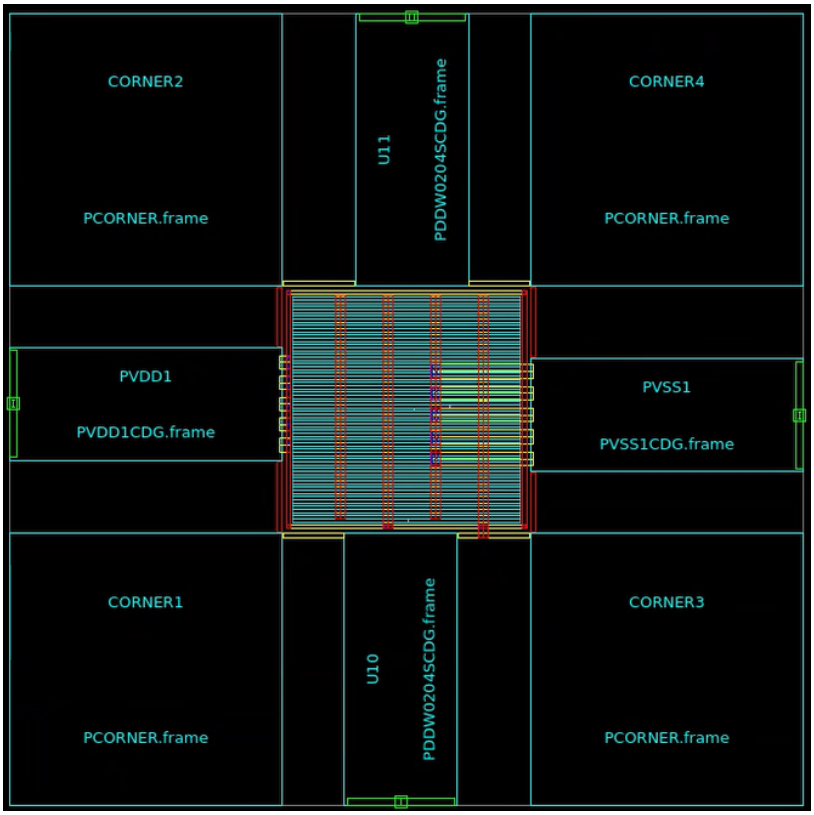
de los *interconnects* para reducir retardos y pérdidas o caídas de voltaje, impactos térmicos y de ruido y optimizar, considerando las restricciones físicas del circuito.

### 6.2.4. *Power and ground routing*

Esta etapa, como su nombre lo indica, se encarga del enrutamiento de las redes de alimentación y tierra. Se diseñan mallas de distribución, también conocidas como *(power grids)*, que aseguran una entrega estable de voltaje a todas las celdas, permiten que se minimicen las caídas de voltaje (*IR drop*) y que las rutas sean las más adecuadas para resistir en la mayor medida posible los picos de corriente.

En tecnologías de escala nanométrica las caídas de voltaje son un problema crítico, dado que los materiales presentan resistencia [20, p. 1] que aumenta con la disminución del tamaño de las tecnologías, por lo que establecer mallas de distribución de energía es de vital importancia para el correcto funcionamiento de los circuitos.

**Figura 3.** Fase de *floorplanning*, *placement*, y *power and ground routing* de una compuerta NOT




Nota. Esta imagen muestra el resultado de ejecutar los comandos de las etapas indicadas para una compuerta NOT en IC Compiler II con tecnología de 65 nm de TSMC.

### 6.2.5. *Clock tree synthesis* (CTS)

En este proceso se genera la red de distribución del reloj, el cual es uno de los recursos más sensibles e importantes del diseño. Este proceso busca garantizar que todas las señales de reloj lleguen a sus destinos con una mínima diferencia de tiempo y que se cumplan los requerimientos de latencia para el funcionamiento del circuito.

Como indica Chong en [21, p. 1], el desarrollo de esta fase es de vital importancia dado que en gran cantidad de circuitos, las señales del reloj representan entre 30 % y, en algunos circuitos especializados, hasta el 50 % de la potencia total que consume el circuito. Esto, además de representar una característica importante en el rendimiento del circuito, también influye significativamente en el consumo de energía y la vida útil del circuito.

**Cuadro 6.1.** Porción del código de la síntesis física de reloj

```
# Sintetizacion de relojes
if { $CLOCK_SYNTHESIS == "TRUE" } {
    puts "Sintetizando relojes..."
    create_clock -period 0.5 -name clk [get_nets -design [current_block] {clk}]
    check_clock_trees -clocks clk
    check_design -checks pre_clock_tree_stage
    synthesize_clock_trees -clocks clk -postroute -routed_clock_stage
    detail_with_signal_routes
    clock_opt -list_only
    check_design -checks cts_qor
} else {
    puts "Clock synthesis deshabilitada."
}
```

Nota. Este cuadro muestra una porción de código utilizado en la síntesis física de una compuerta NOT con tecnolgía de 180 nm de TSMC en distintas fases del desarrollo del proyecto de “El Gran Jaguar”.

### 6.2.6. *Routing*

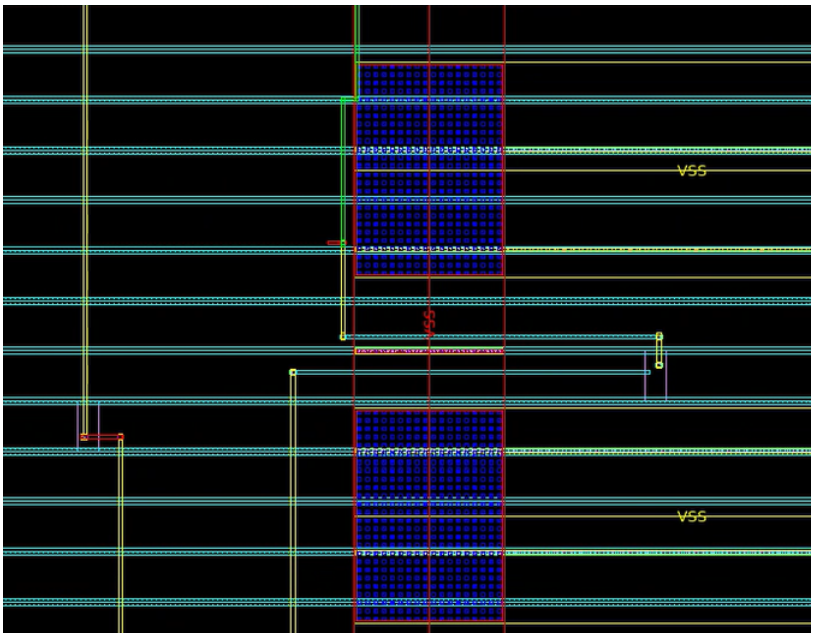
En esta etapa se realiza el enrutamiento de todas las conexiones eléctricas entre celdas y bloques. Se asignan capas metálicas y caminos específicos a cada señal, cumpliendo reglas de manufactura.

El proceso de *routing* es crucial en la etapa de diseño de un circuito integrado, y suele tener una duración prolongada dado que en la actualidad las tecnologías de transistores se han reducido considerablemente y han permitido la integración de millones de componentes en un solo circuito integrado. Esto presenta grandes desafíos, como la optimización para el establecimiento de las rutas más cortas de interconexión, los problemas generados en el enrutamiento global y también en los enrutamientos detallados que ya poseen restricciones geométricas para el correcto funcionamiento de los circuitos [22].

Dependiendo de la herramienta que se utilice para esta etapa, puede dividirse en distintas

fases, como lo es en el caso de IC Compiler II de Synopsys en donde el módulo denominado ZRoute utiliza la etapa de enrutamiento global, asignación de rutas, detalle de rutas, ECO *routing* (*engineering change order routing*), y la verificación de rutas [23].

**Figura 4.** Porción del *layout* de una compuerta NOT con las rutas creadas




Nota. En esta porción del *layout* se distinguen las rutas creadas a partir de los comandos de la herramienta de ICC2 en la etapa de *routing* de una compuerta NOT.

## 6.3. Verificaciones físicas: DRC, LVS, BND y verificaciones de antena

En los procesos de diseño de circuitos integrados es de vital importancia realizar pruebas que garantizan que estos se puedan fabricar y también ayudan a predecir el comportamiento. Para validar que el circuito sea fabricable, las empresas de manufactura generan sus propios *runsets* que brindan a los clientes para que realicen sus propias verificaciones, de modo que los *layouts* cumplan con las reglas y requisitos mínimos. Algunas de las pruebas que se realizan durante el proceso de diseño son DRC, LVS y BND, como se detalla a continuación.

### 6.3.1. DRC (*design rule check*)

Como se mencionó anteriormente, es importante que en el diseño de los circuitos integrados se tomen en cuenta factores geométricos y las restricciones que proporcionan los fabricantes, dadas las limitantes como el tamaño, posibles fallos durante la manufactura y factores físicos que pueden llegar a influir durante el proceso de fabricación.

Las reglas de diseño son parámetros que brindan los fabricantes a los diseñadores, con el fin de que las máscaras utilizadas en los procesos de litografía sean adecuadas para el proceso de fabricación [24]. Estos suelen ser parámetros geométricos como distancias mínimas entre metales, anchos mínimos de las rutas, densidad en las capas de metales, entre otros.

Existen diversas herramientas que pueden ser utilizadas para realizar las verificaciones DRC. La herramienta que se utiliza en la *suite* de Synopsys es IC Validator. Esta herramienta se encuentra integrada en IC Compiler 2, lo cual facilita los procesos de verificaciones. Esta herramienta posee diversas funciones, pero para verificaciones físicas permite validar si existen cruces o traslapes entre celdas, rellenos inapropiados, problemas de densidad, problemas en rutas y en sí permite la utilización de distintos *runsets* que ya contienen las reglas necesarias según la tecnología a utilizar [25]

### 6.3.2. LVS (*layout versus schematic*)

Esta es una prueba que se utiliza para verificar que el circuito diseñado en silicio corresponde y presenta el comportamiento deseado de un circuito en esquemático.

En el proceso usual de diseño de circuitos integrados, por lo general, se dibuja el diagrama eléctrico o esquemático por separado, dado que existen herramientas que permiten simular el comportamiento del circuito por medio de *netlist* como lo pueden ser por modelos SPICE. En ocasiones, estos programas son independientes o pueden estar integrados en un entorno CAD [26, p. 1].

Una vez se realizaron las simulaciones del circuito en esquemático de forma exitosa, se puede proceder a la realización de el *layout* [26, p. 1] que representa los componentes y dopajes del circuito a nivel de silicio. Al terminar esta etapa es en donde se procede a realizar las pruebas de LVS, las cuales permiten verificar si el comportamiento del circuito diseñado en silicio corresponden al esperado según las pruebas con esquemáticos.

Las pruebas de LVS se pueden dividir en 4 tareas principales que permiten comparar estos comportamientos, las cuales son el traslado del *layout* a esquemático, la simplificación de *netlist*, la comparación de *netlist* y la extracción de parámetros [26].

En herramientas como IC Validator, la comparación de LVS se considera limpia si todos los dispositivos y nodos del esquemático concuerdan con el *layout* del circuito, de modo que también es posible asignar tolerancias para comparar los dispositivos y determinar si existen diferencias aceptables para el propósito del circuito [25].

### 6.3.3. BND *decks*

El término *BND deck* se puede referir a dos elementos relacionados con el perímetro del *chip*. En un sentido, corresponde a un conjunto de reglas definidas dentro del DRC que verifican que los límites del diseño cumplan con las especificaciones del proceso de fabricación [27]. Estas reglas aseguran que el diseño no exceda el área permitida del *die*, que no existan metales, vías u otras capas demasiado próximas al borde, que se respeten los márgenes necesarios para evitar defectos durante el corte (*dicing*) y que se garantice la compatibilidad

con el empaquetado.

Por otro lado, BND también hacer referencia a un *bondpad deck*, es decir, una librería que contiene celdas de pads específicas para distintas funciones de entrada/salida (IO digitales, analógicas, *CUP* vs. *non-CUP* [16]). Este conjunto de celdas se emplea principalmente en la etapa de *floorplanning* para definir y ubicar correctamente los pads en el perímetro del chip. Estos archivos son provistos por los fabricantes dado que contienen información específica de su tecnología.

En el caso del presente proyecto, BND hace referencia a las pruebas o el conjunto de reglas que permiten validar que las celdas y el diseño en general cumpla con los requisitos mínimos para que el circuito sea fabricable según la tecnología disponible, que en este caso es *wirebonding*.

### 6.3.4. Pruebas de Antena

El efecto de antena es el que se produce dado los voltajes y corrientes que se producen en los metales durante los procesos de fabricación. Las cargas que se producen en los metales o en el policilicio pueden ser suficientes para dañar el óxido de las compuertas [28], lo cual es crucial evitar para garantizar el funcionamiento de los circuitos.

Estas pruebas y reglas permiten identificar las susceptibilidades de los diseños a efectos de antena que se podrían producir en la manufactura por medio de distintos pasos que involucran la definición de las capas, la extracción de capas intermedias que definen las estructuras de protección de dispositivos, la realización de pruebas posterior a realizar interconexiones y finalmente reportar las violaciones que se han producido durante las verificaciones [28].

## 6.4. Software necesario para la síntesis física

### 6.4.1. IC Compiler II

Como se indica en el manual de usuario de IC Compiler II [29], esta es una herramienta de la empresa Synopsys que permite realizar los procesos de síntesis física, síntesis de los árboles de relojes, *routing* para la implementación física y lógica de circuitos durante la etapa de diseño.

IC Compiler II ofrece la posibilidad de trabajar por medio de la consola de comandos, lo cual permite automatizar los procesos de diseño y realizar la síntesis física de forma más eficiente, dado que es posible generar *scripts* automatizados. Para realizar el proceso de síntesis física según la metodología empleada para el desarrollo del proyecto de "El Gran Jaguar", se utilizarán como base los *scripts* desarrollados por los estudiantes de años anteriores, principalmente los desarrollados por Lourdes Ruiz [15]. Los *scripts* y comandos principales de esta herramienta se listan a continuación.

Para iniciar la interfaz de comandos se debe utilizar el comando **icc2_shell** [29]. Una vez ejecutado el comando iniciará el proceso de verificación de licencias y posterior a esto se

podrán utilizar todos los comandos disponibles en la herramienta. Dado que en este proyecto se utilizará el apoyo de *scripts*, el comando **source** se podrá utilizar para ejecutarlos [29], de manera que se pueda automatizar el proceso de síntesis física.

Los *scripts* que se utilizarán para la síntesis física del circuito son los siguientes:

- **sintesis_fisica_top.tcl**: este es el *script* principal que se utiliza para realizar la síntesis física del circuito. En este documento se contienen los comandos que llaman a los demás *scripts* y también contiene las configuraciones necesarias para el proceso de síntesis física.

  Para ejecutar este *script* se debe utilizar el comando **source sintesis_fisica_top.tcl** una vez se haya iniciado la consola de IC Compiler II. Algunos de los comandos importantes que se encuentran en este *script* son:

  - **read_verilog**: este comando se utiliza para leer el archivo de descripción del circuito en lenguaje verilog [29] del circuito a sintetizar.
  - **route_auto**: se utiliza para realizar el enrutamiento automático de las señales del circuito, una vez se ha realizado la síntesis física y se han definido las conexiones entre las celdas.
  - **write_verilog**: este comando se utiliza para escribir el *netlist* resultante de la síntesis física en un archivo verilog, el cual se puede utilizar posteriormente para simulaciones o verificaciones adicionales.
  - **write_gds**: con él se genera el archivo GDSII [29], que es el formato estándar para la representación de diseños de circuitos integrados y se utiliza para la fabricación del circuito.

  Es importante mencionar que en este *script* se llaman a los demás que son secundarios, y existe uno el cual fue proporcionado por IMEC y TSMC y es el utilizado para definir las reglas de antena.

- **setup.tcl**: este *script* se encarga de realizar la configuración inicial del entorno de IC Compiler II, como la carga de librerías y la configuración de las variables necesarias para el proceso de síntesis física. En la metodología que se utilizará para el proyecto no será necesario llamar manualmente a este *script*, dado que se llama desde el *script* principal **sintesis_fisica_top.tcl**.

  Algunos de los comandos importantes que se encuentran en este *script* son:

  - **set**: este comando se utiliza para definir variables y parámetros [29] que se utilizarán durante el proceso de síntesis física. Por ejemplo, se pueden definir las rutas a las librerías de celdas estándar, los archivos de tecnología y otros parámetros necesarios para la síntesis.
  - **file**: se utiliza para manejar archivos y directorios dentro del entorno de IC Compiler II como por ejemplo para definir la ruta de salida donde se guardarán los archivos generados durante la síntesis física.
  - **puts**: para imprimir mensajes en la consola que pueden ser utilizados para informar sobre el progreso del proceso o para mostrar mensajes de error.

- **floorplan.tcl**: este es el encargado de realizar el *floorplan* del circuito, definiendo las áreas de los bloques y las regiones de alimentación y tierra. En este se definen las conexiones por medio de los comandos de IC Compiler II.

  En este *script* es en donde se utiliza una mayor variedad de comandos dado que se detallan las áreas de los bloques, las conexiones entre ellos y las regiones de alimentación y tierra. Algunos de los comandos importantes que se utilizarán son:

  - **create_cell**: se ejecuta para la creación de las céldas estándar que se utilizarán en el diseño del circuito [29]. Este comando permite definir las celdas y sus propiedades, como el tamaño y la orientación.
  - **create_net**: crear las rutas de interconexión entre las celdas del circuito.
  - **connect_pg_net**: se utiliza para conectar las redes de alimentación y tierra a las celdas del circuito, asegurando que todas las celdas tengan acceso a estos nodos.
  - **initialize_floorplan**: iniciar el proceso de *floorplan* del circuito [29], estableciendo las dimensiones y la organización espacial de los bloques.
  - **create_io_ring**: crea el anillo de entradas y salidas del circuito que permiten la conexión con el exterior del chip.
  - **merge_pg_mesh**: este comando se ejecuta para combinar las mallas o redes de alimentación y tierra, asegurando que estas estén correctamente conectadas y distribuidas en todo el circuito.

Es importante mencionar que estos son solamente algunos de los comandos más relevantes que se han empleado en el desarrollo de la síntesis física, ya que existe una gran variedad de comandos que se pueden utilizar con distintas opciones y parámetros.

### 6.4.2. IC Validator

Esta es otra herramienta de Synopsys que realiza verificaciones físicas de alto rendimiento que ayudan a garantizar que el diseño de un *chip* sea fabricable. Esta herramienta proporciona verificaciones como DRC, LVS y también otras pruebas como PERC (*programmable electrical rule check*)[31].

Existe un flujo de diseño denominado *in-design physical verification*, con el cual se pueden utilizar los comandos de IC Validator desde la herramienta de IC Compiler[32]. Esto da gran flexibilidad para el proceso dado que es posible utilizar los comandos directamente desde la consola luego de completar el proceso de síntesis de los circuitos. Algunos de los comandos más utilizados de esta herramienta para el desarrollo del proyecto son los que se muestran a continuación.

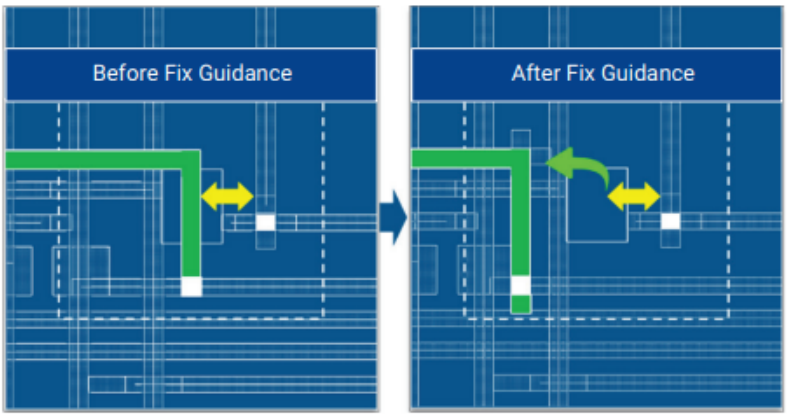
- **signoff_check_drc**: se utiliza para realizar la verificación de DRC a partir de un runset definido. La herramienta genera un informe de resultados con un resumen en donde se indican las violaciones encontradas, de manera que en la sección de DRC se muestran dos secciones, de las cuales la primera contiene estadísticas y nummero de errores, y en la segunda se muestra cada nombre de las reglas que fueron violadas, el número de errores y una descripción de la regla [25].

**Figura 5.** Herramientas que ofrece IC Validator para verificaciones de diseño


Nota. La gráfica muestra las herramientas y tareas que puede ejecutar IC Validator para las verificaciones físicas de un diseño sintetizado según la documentación oficial de Synopsys [30].

- **signoff_fix_drc**: corrige errores de DRC de forma automática si es posible.
- **signoff_create_metall_fill**: inserta metal de relleno para cumplir con requisitos de densidad [32].
- **check_lvs**: se utiliza para realizar una verificación de LVS preeliminar en esta etapa del flujo de diseño. En esta herramienta las pruebas de LVS se ejecutan en tres etapas, donde la primera hace modificaciones globales al *netlist* para facilitar la comparación, por ejemplo cuando dos puertos están conectados a un mismo noto, los unifica. En la segunda etapa se considera una lista de pares de celdas para determinar las equivalencias. Finalmente, se utiliza una serie de filtros y se compara el esquemático lógico con el *layout* del circuito [25].

**Figura 6.** Reparaciones automáticas de DRC de IC Validator




Nota. La imagen muestra un ejemplo del manual de IC Validator [31] sobre la funcionalidad de reparaciones automáticas de DRC.

# CAPÍTULO 7

## Preparación de librerías y *runsets* para la ejecución de la síntesis física

### 7.1. Elaboración de librerías para 65 nm con 9 capas de metal

En metodologías empleadas en trabajos realizados por los estudiantes de años anteriores se había tomado como base la utilización de los archivos de la tecnología de TSMC de 6 capas de metal. Sin embargo, según las recomendaciones del manual brindado por IMEC [16] se debe emplear la tecnología de 9 capas de metal con un esquema específico. Tomando esto como base, previo a realizar el proceso de síntesis física se ejecutaron los pasos para la generación de librerías a partir de los archivos de la tecnología (`.tf`), los archivos de con extensión `.lef` que contienen la información física de las celdas y también los archivos `.db` que manejan la información lógica de las celdas. Para esto se utilizó la herramienta de Library Manager Tool por medio de la cual se generó la carpeta correspondiente y a la cual se hace referencia en el *script* de `setup.tcl` denominada `Librerias`.

En el manual brindado por IMEC se indica que la recomendación sobre el esquema de metal para utilizar con la tecnología de 65 nm es la denominada 1P9M_6X1Z1U_MIM, lo cual indica que se deben utilizar 9 capas de metal y también los archivos de tecnología con las dimensiones 6X1Z1U. Sin embargo, debido que al realizar la búsqueda de los archivos en el repositorio compartido por IMEC no se encontraron los archivos específicos correspondientes a este esquema, se decidió partir con el esquema 6X2Z con 9 capas de metal. A partir de esta información se eligieron nuevamente los archivos de tecnología, las librerías de celdas estándar, las de entradas y salidas y también los archivos LEF.

En las primeras pruebas que se realizaron para la generación de librerías no se realizaron cambios en los procedimientos, sino que solamente cambiaron los archivos elegidos según las rutas que se muestran a continuación.

- Archivo de tecnología (TF): `65nm → CMOS → LP → 9T → tcbn65lp_set → techfiles`

→ tsmcn65_9lmT2.tf

- Archivo LEF (*Library Exchange Format*) de librerías estándar: 65nm → CMOS → LP → 9T → stdcell → T-N65LP-DR-200A → LEF → tcbn65lp_9lmT2.lef
- Archivo LEF de librearías de entradas y salidas: 65nm → CMOS → LP → IO 2.5V → LINEAR → T-PDN65LP-DR-140B → LEF → tpdn65lpnv2od3_9lm.lef
- Archivos DB correspondientes a las librerías estándar: 65nm → CMOS → LP → 9T → stdcell → T-N65LP-DR-220A → NLDM
- Archivos DB para las librerías de entradas y salidas: 65nm → CMOS → LP → IO 2.5V → LINEAR → T-PDN65LP-DR-200A → NLDM

Para generar las librerías con extensión NDM (*new data model*) que se utilizan en IC Compiler II se emplearon los pasos descritos en el video y documentación realizada por Lourdes Ruiz [15]. Si bien este procedimiento se realizaba por medio de la interfaz gráfica, al abrir la herramienta de Library Manager se observó que automáticamente se generaba un archivo de "*log*" de comandos que contenía todos los comandos ejecutados en la interfaz gráfica. Para simplificar y automatizar los procesos se tomó este archivo como referencia para generar un *script* que permitiera generar las librerías de forma automática que se muestra en los Cuadros 7.1 y 7.2.

**Cuadro 7.1.** *Script* final para la generación de librerías NDM

```
# Script para crear librerias
# Opcion para activar lectura de fillers
# set_app_options -name lib.physical_model.read_fill_cells -value true

# Borramos las librerias y archivos anteriores para crear los nuevos.
sh rm -rf command.log
sh rm -rf FillersWorkspace.ndm
sh rm -rf StandardWorkspace.ndm
sh rm -rf CornersWorkspace.ndm
sh rm -rf PadsWorkspace.ndm
sh rm -rf TSMCWorkspace.ndm
sh rm -rf check_workspace.ems
sh rm -rf PreFrameCheck

# Existen 4 workspaces, y 1 workspace general.
# Standard workspace (standard cells) y fillers workspace (fillers para core)
# Pads workspace (pads io) y corners worksapce ( fillers de corner o de ring io)

#Creando Standard Workspace ////////////////////////////////////////
create_workspace -flow normal -technology <ruta hacia tech file> StandardWorkspace

# Agregar db-techfile standard cells
read_db { <rutas hacia archivos .db>
}

# Agregar lef standard cells
read_lef /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t/Librerias
    /lib/ndm/physical-cells/standard-cells/tcbn65lp_9lmT2.lef
#check y commit de Standard workspace
current_workspace; check_workspace
current_workspace StandardWorkspace; commit_workspace  -output
    StandardWorkspace.ndm

#Creando Fillers Workspace ////////////////////////////////////////
create_workspace -flow physical_only -technology /home/nanoelectronica/Desktop/
    Folder_de_Trabajo/TSMC_65nm/9M_9t/Librerias/lib/ndm/techfile/tsmcn65_9lmT2.tf
    PhysicalOnlyWorkspace

# agregar db-techfiles
read_db { <ruta hacia archivos .db>
}

# Agregar lef
read_lef /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t/Librerias
    /lib/ndm/physical-cells/standard-cells/tcbn65lp_9lmT2.lef
```

Nota. El cuadro contiene el la primera parte del *script* que fue utilizado para la generación automatizada de librerías NDM.

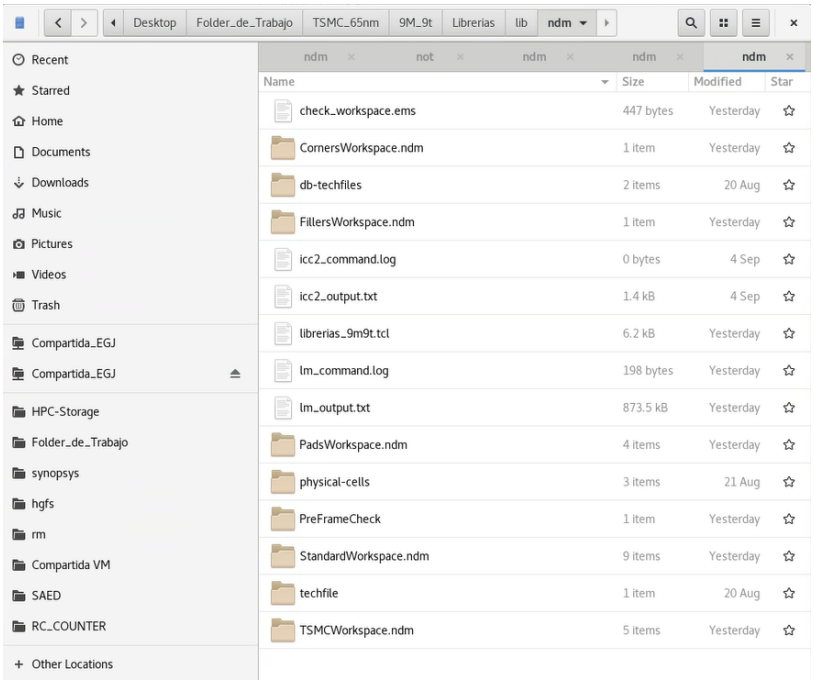
**Cuadro 7.2.** Segunda parte del *Script* final para la generación de librerías NDM

```
# check y commit de Fillers workspace
current_workspace; check_workspace
current_workspace PhysicalOnlyWorkspace; commit_workspace  -output
    FillersWorkspace.ndm

#Creando Pads Workspace ////////////////////////////////////////////
create_workspace -flow normal -technology <ruta hacia tech file> PadsWorkspace
# Agregar db-techfile i/o
read_db {<rutas hacia archivos .db de celdas de IO>}
# Agregar lef i/o
read_lef /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t/Librerias
    /lib/ndm/physical-cells/IO-cells/tpdn65lpnv2od3_9lm.lef
#check y commit de Pads workspace
current_workspace; check_workspace
current_workspace PadsWorkspace; commit_workspace  -output PadsWorkspace.ndm

# Creando Corners Workspace ////////////////////////////////////////////
create_workspace -flow physical_only -technology <ruta hacia tech file>
    PhysicalOnlyWorkspace
#agregar db-techfile I/O
read_db {<rutas hacia archivos .db de celdas de IO>}
# agregar lef I/O
read_lef /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t/Librerias
    /lib/ndm/physical-cells/IO-cells/tpdn65lpnv2od3_9lm.lef
# ceck y commit de Corners workspace
change_selection [get_workspaces {PhysicalOnlyWorkspace}]
current_workspace; check_workspace
change_selection [get_workspaces {PhysicalOnlyWorkspace}]
current_workspace PhysicalOnlyWorkspace; commit_workspace  -output
    CornersWorkspace.ndm

# Creando TSMC workspace ---------------------------
create_workspace -flow aggregate TSMCWorkspace
# Leer librerias ndm
read_ndm /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t/Librerias
    /lib/ndm/CornersWorkspace.ndm; read_ndm /home/nanoelectronica/Desktop/
    Folder_de_Trabajo/TSMC_65nm/9M_9t/Librerias/lib/ndm/FillersWorkspace.ndm;
    read_ndm /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t/
    Librerias/lib/ndm/PadsWorkspace.ndm; read_ndm /home/nanoelectronica/Desktop/
    Folder_de_Trabajo/TSMC_65nm/9M_9t/Librerias/lib/ndm/StandardWorkspace.ndm;
change_selection [get_workspaces {TSMCWorkspace}]
current_workspace; check_workspace
open_ems_database check_workspace.ems
change_selection [get_workspaces {TSMCWorkspace}]
current_workspace TSMCWorkspace; commit_workspace  -output TSMCWorkspace.ndm
```

Nota. El cuadro contiene la continuación del *script* que fue utilizado para la generación automatizada de librerías NDM.

Para poder emplear el *script* es necesario ubicar los archivos en las carpetas correctas, para lo cual se tomó como referencia la jerarquía de directorios descrita por Miguel Chacón [33] para colocar los archivos LEF, DB y TF en las carpetas correctas según el *script* de los Cuadros 7.1 y 7.2.

**Figura 7.** Librerías NDM generadas en Library Manager Tool




Nota. En la imagen se observan las libreraías generadas a partir del *script* automatizado

En este caso el *script* se ejecutó desde distintas carpetas dependiendo de la tecnología que se emplea, pero siempre se utilizó la jerarquía de directorios que se detalla en el trabajo descrito por Miguel Chacón [33]. En este caso se realizó una carpeta en el folder de trabajo denominada **TSMC_65nm**. Dentro del directorio se encuentran los distintos esquemas de metal empleados, por lo cual en este caso se debe ingresar a la carpeta `9M_9t`, para luego dirigirse a la ruta `./Librerias/lib/ndm/`. En esta carpeta se ejecuta el *script* denominado `librerias_9m9t.tcl` por medio de los siguientes comandos del Cuadro 7.3.

**Cuadro 7.3.** Comandos para ejecución de *script* de generación de librerías

```
[nanoelectronica@localhost ndm]$ lm_shell

...

lm_shell> source librerias_9m9t.tcl
```

Nota. Este cuadro muestra los comandos que hay que ingresar para que se ejecute el *script* de generación de librerías.

Como se observa en la Figura 7 se generaron 4 directorios con extensión NDM, las cuales contienen cada uno distinta información de las celdas dependiendo del flujo que se emplea en la creación del espacio de trabajo. En el *script* del Cuadro 7.1 se observan los siguientes flujos con el comando `create_workspace`:

- *Normal*: este flujo se utiliza para incluir librerías con información lógica y física de celdas, así como información de *timing* en diferentes puntos de operación. Usualmente este flujo se utiliza para las librerías estándar [34].
- *Physical_ only*: este flujo se utiliza con celdas que solo contienen información física y no están incluidas en las librerías lógicas. Un uso común de este tipo de celdas es en la generación de celdas de relleno (*fillers*) [34].
- *Aggregate*: en este caso se genera una única librería que combina la información de varias librerías de referencia [34].

De esta manera se generan las cuatro librerías NDM que se utilizarán en las siguientes etapas, las cuales son **CornersWorkspace.ndm**, que contiene la información de caracterización de las celdas; **FillersWorkspace.ndm**, que contiene las celdas de relleno; **PadsWorkspace.ndm**, que tiene las celdas de entradas y salidas; y finalmente, **TSMC-Workspace.ndm**, que combina la información de las librerías en una sola. En el trabajo de Miguel Chacón se encuentra un análisis más detallado sobre la generación de librerías empleada en las primeras fases de las pruebas que se realizaron en el año 2025 [33].

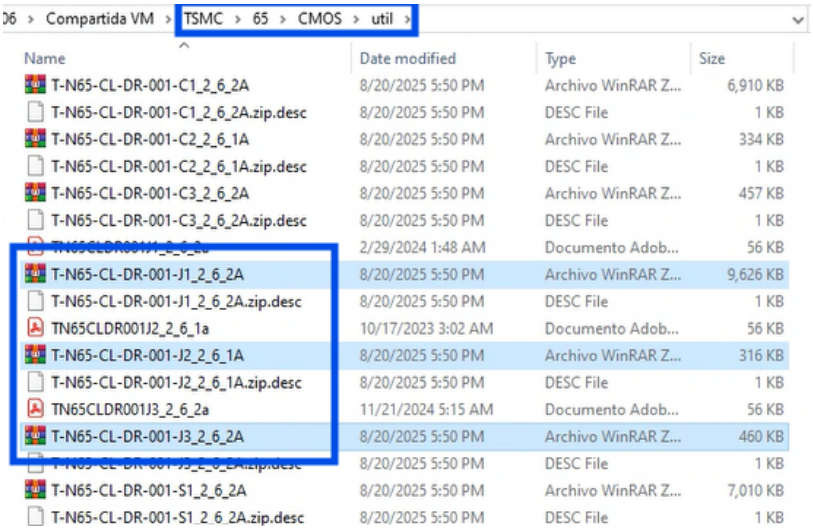
## 7.2. Selección de *runsets* para la síntesis física con 9 capas de metal

### 7.2.1. Archivos para pruebas DRC, antena y rellenos de metal

En las primeras pruebas efectuadas para la fase de síntesis física se había tomado como referencia el manual de diseño proporcionado por IMEC [16] según la Figura 8. Sin embargo, en esta ruta se determinó que no se encontraban disponibles los archivos específicos para la *suite* de Synopsys, por lo cual con el apoyo y comunicación con IMEC se actualizaron los directorios y se seleccionaron los archivos que se indican a continuación y que son equivalentes a los mostrados en la figura para la *suite* de Synopsys.

- T-N65-CL-DR-001-J1_2_6_2A: este archivo contiene los *runsets* de DRC y antena.
- T-N65-CL-DR-001-J2_2_6_1A: este archivo contiene los *runsets* relacionados con los rellenos OD/POLY (FEOL).
- T-N65-CL-DR-001-J3_2_6_2A: este archivo contiene los *runsets* correspondientes a los rellenos de metal BEOL.

**Figura 8.** Ruta del repositorio para acceder a los *runsets* de TSMC 65 nm

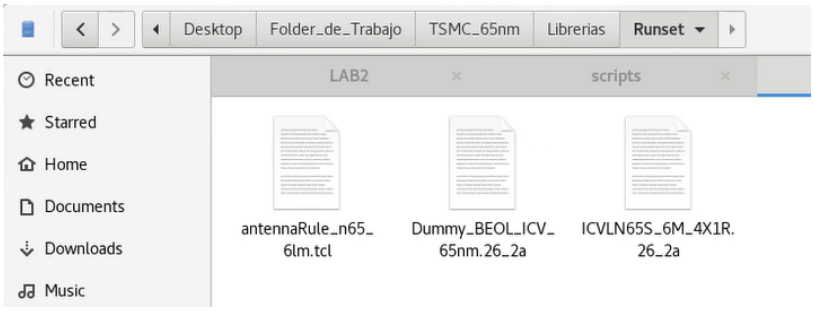



Nota. Esta imagen muestra la ubicación de los *runsets* en el directorio para las pruebas de DRC y rellenos de metal según el manual de mini@sic [16] para la *suite* de Synopsys.

En esta misma ubicación se encuentran otros archivos como los correspondientes a las pruebas de LVS, considerando que la letra "J" en el nombre del archivo indica que son archivos diseñados para la *suite* de Synopsys.

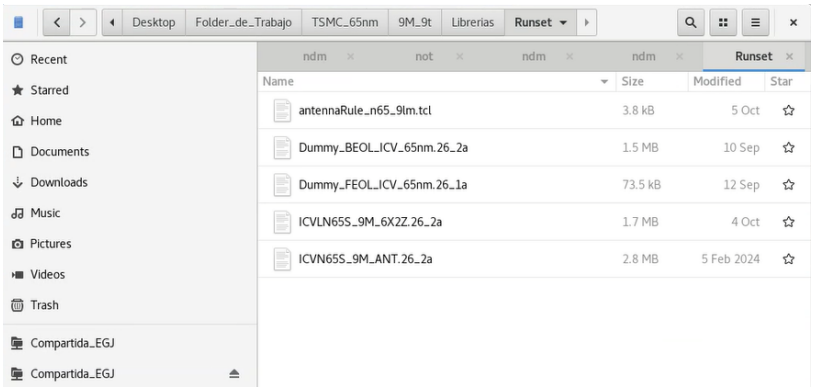
Originalmente se habían seleccionado los *runsets* que se muestran en la Figura 9, sin embargo luego de conocer el esquema adecuado de metales que se debía emplear se efectuaron cambios en la selección del archivo de DRC, antena y a diferencia de años anteriores, se trabajó con dos *runsets* relacionados con el proceso de colocación de rellenos, utilizando tanto el archivo denominado "FEOL"(*Front-End of Line*), como "BEOL"(*Back-End of Line*). De esta manera se logró llegar a la selección de los archivos que se muestran en la Figura 10

**Figura 9.** *Runsets* elegidos en la fase inicial de síntesis física




Nota. La imagen muestra los 3 *runsets* elegidos originalmente en las primeras pruebas de síntesis física con 65 nm.

**Figura 10.** *Runsets* finales elegidos para la ejecución de la síntesis física



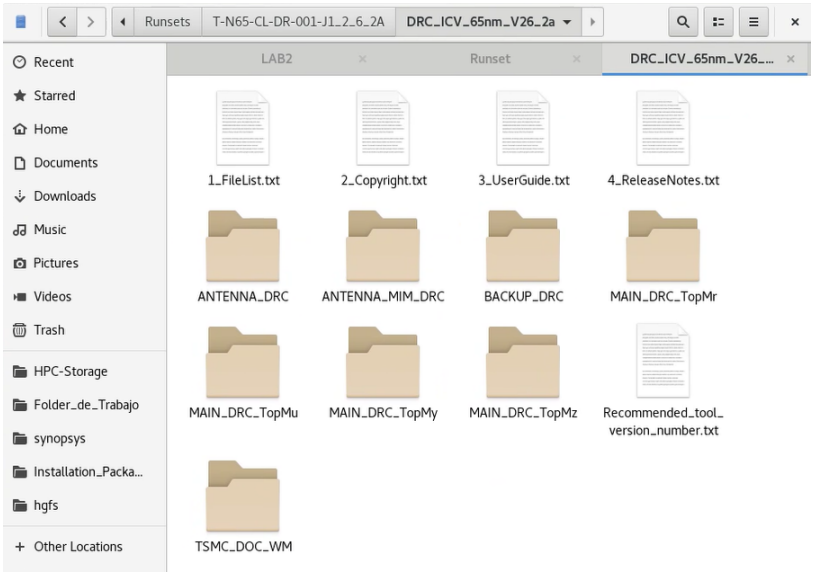

Nota. En esta imagen se muestran los *runsets* seleccionados finalmente para los procesos relacionados con la síntesis física de los circuitos.

Según el documento de Mini@sic [16], el manual de diseño (DRM - *Design Rule Manual*) se encuentra en la siguiente ruta del repositorio: `65nm → CMOS → doc → T-N65-CL-DR-001_-2_3.pdf`. En este documento se encuentran directrices generales que se deben utilizar para el manejo adecuado de la tecnología. Este archivo se tomó como referencia para tener una mejor comprensión de las recomendaciones de TSMC para la tecnología de 65 nm.

El equipo de trabajo verificó que en el manual [16] se describe la ruta para los archivos de PDK (*Process Design Kit*), sin embargo, en la versión de la tecnología brindada por IMEC, no están disponibles estos archivos para la *suite* de Synopsys. Esto implica que en lugar de utilizar los archivos y librerías prestablecidos, era necesario realizar las librerías a través de Library Manager Tool y diseñar manualmente los *scripts* para la ejecución de la síntesis física.

El primer *runset* que corresponde a las pruebas DRC se encuentra en la siguiente ruta `65nm → CMOS → util → T-N65-CL-DR-001-J1_2_6_2A`. En esta ruta se encontraron diversos archivos como se observa en la Figura 11.

**Figura 11.** *Runsets* relacionados con DRC para la *suite* de Synopsys




Nota. La imagen muestra las distintas carpetas con *runsets* relacionados con las pruebas DRC diseñados para las herramientas de Synopsys.

Según la recomendación brindada en el archivo con nombre `T-000-BP-DR-017_0_8.pdf` que se encuentra en la documentación del repositiorio, se hace la observación de que para tecnología *wire-bond* es necesario utilizar dos capas **Mz**. De modo que el runset seleccionado se encuentra en la carpeta denominada **MAIN_DRC_TopMz** y que corresponde a 9 capas de metal con el esquema indicado previamente.

Con los archivos seleccionados se procedió a ubicarlos dentro de la carpeta de la siguiente ruta: `home/Desktop/Folder_de_trabajo/TSMC_65nm/9M_9t/Librerias/Runset/` y en las rutas equivalentes para el resto de pruebas con otros esquemas de metal. A partir de haber seleccionado los archivos para la ejecución de las verificaciones, se decidió dar inicio a los procesos de preparación de las librerías para la síntesis física.

### 7.2.2. Archivos para extracción de parásitos

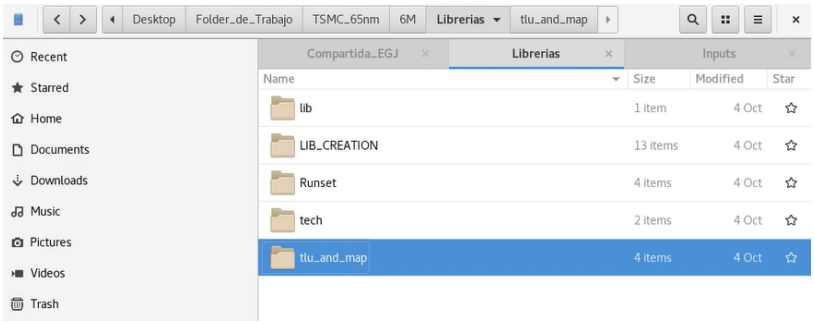
Además de los archivos utilizados para la elaboración de librerías y verificaciones de diseño se deben seleccionar los archivos para la extracción de parásitos, que en este caso corresponden a los TLUPLUS y MAP.

Los archivos TLUPLUS (*Table Look Up*) contienen la información de las capacitancias y resistencias parásitas de las celdas estándar y las capas de metal. El nombre del archivo TLUPLUS seleccionado es **cln65lp_1p09m+alrdl_typical_top2.tluplus**. Para la *suite* de Synopsys se utiliza otro archivo con el nombre **star.map_9M** (dependiendo de la cantidad de capas de metal varía el nombre), el cual sirve para hacer la traducción de los nombres de las capas de metal entre los archivos TLUPLUS y los de la tecnología.

La ruta para acceder a ambos archivos en el repositorio de IMEC es la siguiente:

65nm/CMOS/LP/stclib/9-track/tcbn65lp_set/tcbn65lp_220a_FE/tcbn65lp_200a_apf.→
tar.gz/TSMCHOME/digital/Back_End/milkyway/tcbn65lp_200a/techfiles/tluplus/

**Figura 12.** Ejemplo del directorio de librerías




Nota. En la imagen se muestra la carpeta de librerías para la tecnología de 6M en donde se ubicaron los archivos para la extracción de parásitos.

# CAPÍTULO 8

## Pruebas iniciales con tecnología de 65 nanómetros y 6 capas de metal

## 8.1. Pruebas con *scripts* originales para la síntesis física

Inicialmente se tomaron los *scripts* realizados para el proceso con la tecnología de 180 nm de TSCM. El primer paso fue validar todas las rutas, nombres de archivos y de celdas según los *runsets* y las librerías generadas para la tecnología de 65 nm. En el capítulo anterior se describió el procedimiento con 9 capas de metal, sin embargo, en las primeras pruebas se había utilizado la tecnología de 6 metales. Esto no tiene diferencia en los procesos de preparación de archivos, pero, se describió con mayor detalle con 9 capas de metal dado que este fue el esquema que se eligió finalmente.

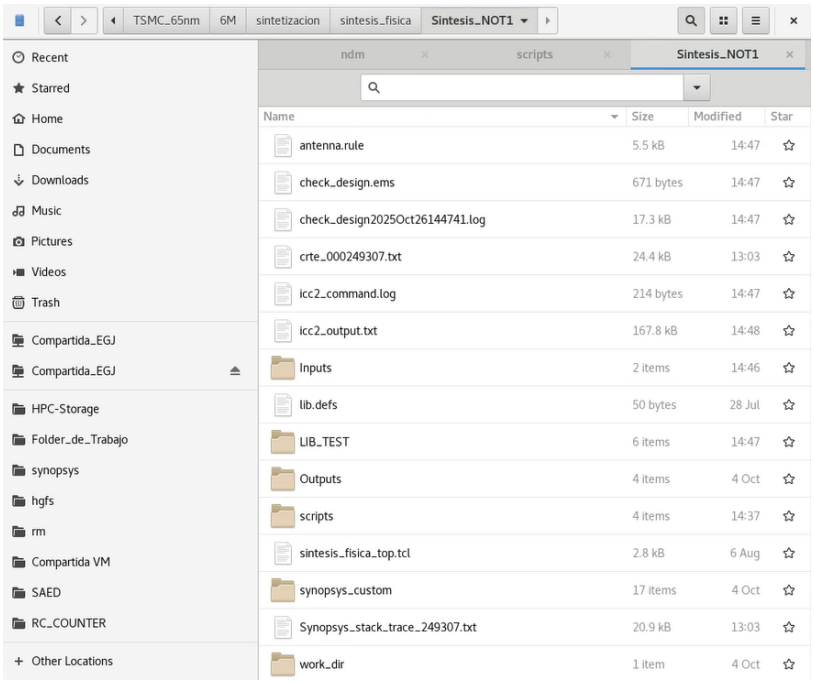
Antes de iniciar a realizar cambios en los comandos ejecutados y de realizar un análisis más detallado del procedimiento, se hicieron pruebas preliminares ejecutando los 3 *scripts* principales para validar que no existieran errores en la ejecución. Para ejecutar el proceso de síntesis física con 6 capas de metal se utilizó el procedimiento que se indica a continuación:

- Se tomó como referencia la jerarquía de directorios descrita en el trabajo de Miguel Chacón [33], de modo que se generó una carpeta denominada `6M` dentro del directorio `TSMC_65nm`. Al acceder a esta carpeta se creó la estructura de carpetas necesaria para la síntesis física, para lo cual en la ruta `TSMC_65nm/6M/sintetizacion/sintesis_fisica/` se creó la carpeta `Sintesis_NOT1`. En esta carpeta se ubicaron los *scripts* principales para la ejecución de la síntesis física según se muestra en la Figura 13.
- En el directorio principal se colocó el *script* denominado **sintesis_fisica_top.tcl**, y dentro de la carpeta `scripts` se ubicaron los otros dos archivos con nombres **setup.tcl** y **floorplan.tcl**.
- Dentro de la carpeta llamada **Inputs** se colocó el archivo del circuito en formato `.v`

que contiene la descripción en Verilog de la compuerta NOT, así como también el archivo de *constraints* en formato **.sdc**.

- Se ubicaron los archivos de librerías NDM según lo descrito en la sección 7.1 y se realizó el cambio en las rutas de los *scripts* para que coincidieran con la jerarquía de directorios empleada.

**Figura 13.** Directorio para ejecución de síntesis física de compuerta NOT con 6 capas de metal




Nota. En la figura se muestra la composición de la capeta en la que se ejecuta la síntesis física de la compuerta NOT con 6 capas de metal

En los Cuadros 8.1, 8.2 y 8.3 se muestran los comandos que fueron modificados para adecuar las rutas y nombres de archivos a la tecnología de 65 nm con 6 capas de metal. En los cuadros se pueden observar las rutas absolutas dado que hasta este punto no se realizó ninguna mejora en los *scripts*, sino que sirvió solo como prueba piloto para preparar los documentos y validar que funcionaran según lo esperado.

**Cuadro 8.1.** Cambios en el *script* `sintesis_fisica_top.tcl` de 6M

```
...

#antenna
source -verbose "../../../Librerias/Runset/antennaRule_n65_6lm.tcl"

...

#Creamos los filler del core y el IO ring
create_io_filler_cells -io_guides [get_io_guides {ANILLO_IO.top ANILLO_IO.right
    ANILLO_IO.left ANILLO_IO.bottom}] -reference_cells {PFILLER0005 PFILLER1
    PFILLER5 PFILLER05PFILLER10 PFILLER20}
create_stdcell_fillers -lib_cells [get_lib_cells {TSMCWorkspace|FillersWorkspace/
    FILL64BWP7T TSMCWorkspace|FillersWorkspace/FILL32BWP7T TSMCWorkspace|
    FillersWorkspace/FILL16BWP7T TSMCWorkspace|FillersWorkspace/FILL8BWP7T
    TSMCWorkspace|FillersWorkspace/FILL4BWP7T TSMCWorkspace|FillersWorkspace/
    FILL2BWP7T TSMCWorkspace|FillersWorkspace/FILL1BWP7T}]

...

set_app_options -list {signoff.check_design.runset {../../../Librerias/Runset/
    ICVLN65S_6M_3X2Z.26_2a}}
set_app_options -list {signoff.check_drc.runset {../../../Librerias/Runset/
    ICVLN65S_6M_3X2Z.26_2a}}

set_app_options -list {signoff.create_metal_fill.runset {../../../Librerias/Runset
    /Dummy_BEOL_ICV_65nm.26_2a}}

...
```

Nota. Este cuadro contiene las líneas de comandos que fueron modificadas en el *script* de síntesis física para adecuar las rutas y nombres de archivos a la tecnología de 65 nm con 6 capas de metal.

En el Cuadro 8.1 se puede distinguir que se realizaron configuraciones de los *runsets* que se utilizarán para hacer las pruebas de antena, DRC y rellenos. Conforme se hizo la revisión de estos documentos se tomaron estas observaciones para posterior efectuar los cambios necesarios para tener un mejor orden y estructura, ya que es ideal que estas opciones del programa se configuren en las etapas iniciales del proceso de síntesis física.

**Cuadro 8.2.** Modificaciones en el *script* de `setup.tcl` de 6M

```
...

set DESIGN_REF_PATH "/home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/6M/
    Librerias"

...

set TECH_FILE        "${DESIGN_REF_PATH}/tech/tsmcn65_6lmT2.tf"
set MAP_FILE         "${DESIGN_REF_PATH}/tlu_and_map/star.map_6M"
set TLUPLUS_FILE "${DESIGN_REF_PATH}/tlu_and_map/cln65lp_1p06m+
    alrdl_typical_top2.tluplus"

...
set CLOCK_SYNTHESIS FALSE  ;#  Configurar como TRUE o FALSE para habilitar o
    deshabilitar la sintesis de reloj
set CIRCUIT_NAME "Not_IO" ;# Definir el nombre del circuito

...

file copy -force "/home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/6M/
    sintetizacion/sintesis_logica/Outputs/NOT/Outputs_finales/not_sintesis_final.v
    " "./Inputs/"
file copy -force "/home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/6M/
    sintetizacion/sintesis_logica/Outputs/NOT/Outputs_finales/
    not_sintesis_final.sdc" "./Inputs/"
set INPUT_name "not_sintesis_final"

...

file copy -force "/home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/6M/
    Librerias/Runset/ICVLN65S_6M_3X2Z.26_2a" "./Outputs/Antena/"
```

Nota. Este cuadro contiene las partes del *script* de `setup.tcl` que fueron modificadas para ejecutar la síntesis física con 6 capas de metal.

Cabe destacar que en el Cuadro 8.2 se muestra que se debe hacer el cambio en la variable `CLOCK_SYNTHESIS` al valor `FALSE` dado que la compuerta NOT no requiere una red de reloj. Además, se debe definir el nombre del circuito en la variable `CIRCUIT_NAME` de modo que coincida con el nombre del módulo principal en el archivo Verilog. Finalmente para este *script* se hizo el cambio en la variable `INPUT_name` para que coincida con el nombre del archivo Verilog que contiene el *netlist* del circuito.

**Cuadro 8.3.** Modificaciones iniciales en el *script* del *floorplan*

```
...

create_pg_ring_pattern ring_pattern -horizontal_layer M2    -horizontal_width {2}
    -horizontal_spacing {2}    -vertical_layer M3 -vertical_width {2}
    -vertical_spacing {2}

...

create_pg_macro_conn_pattern hm_pattern -pin_conn_type scattered_pin -layers {M2
    M3} -nets {VDD VSS} -pin_layers {M2}

...

set_pg_strategy_via_rule macro_conn_via_rule -via_rule { { { {strategies:
    macro_conn}} { {existing: all} {layers: M3} } {via_master: default} }{{
    intersection: undefined}{via_master: NIL}}}

...

create_pg_mesh_pattern mesh_pattern -layers {{{ vertical_layer: M3} {width: 4.2}{
    pitch: 42} {spacing: interleaving }}}

...

merge_pg_mesh -nets {VDD VSS} -types {ring stripe} -layers {M2 M3}

...
```

Nota. En el cuadro se observan los comandos que requirieron modificaciones principalmente correspondientes a los nombres de las capas de metal según el archivo de tecnología de 65 nm de TSMC.

En el *script* del *floorplan* no fue necesario realizar mayores modificaciones además de los nombres de las capas de meta, ya que este archivo no contiene rutas hacia archivos externos.

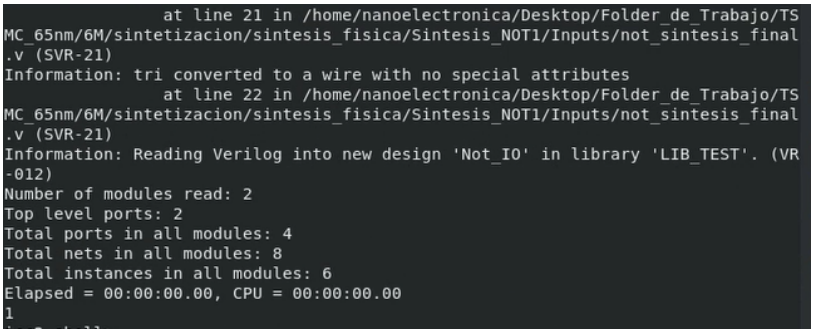
## 8.2. Resultados de las pruebas iniciales con 6 capas de metal

Para tener una mejor comprensión de los *scripts* utilizados en los trabajos anteriores, se decidió realizar las pruebas por segmentos de comandos para observar el comportamiento de cada etapa y verificar cómo se construye cada etapa del proceso de síntesis física.

Primero se ejecutó el *script* de configuración (**setup.tcl**). En este archivo se encuentran los comandos que preparan el entorno de trabajo de IC Compiler II, es decir que se configuran variables, rutas y valores constantes necesarios para las siguientes etapas. En esta primera parte se resalta el proceso de creación de la librería de trabajo a partir de los archivos NDM generados previamente.

Posteriormente se ejecutaron los comandos para la lectura del archivo Verilog como se observa en la Figura 14. También se ejecutaron los comandos para la lectura del archivo SDC con los *constraints* y el archivo con la información de parásitos de las celdas.

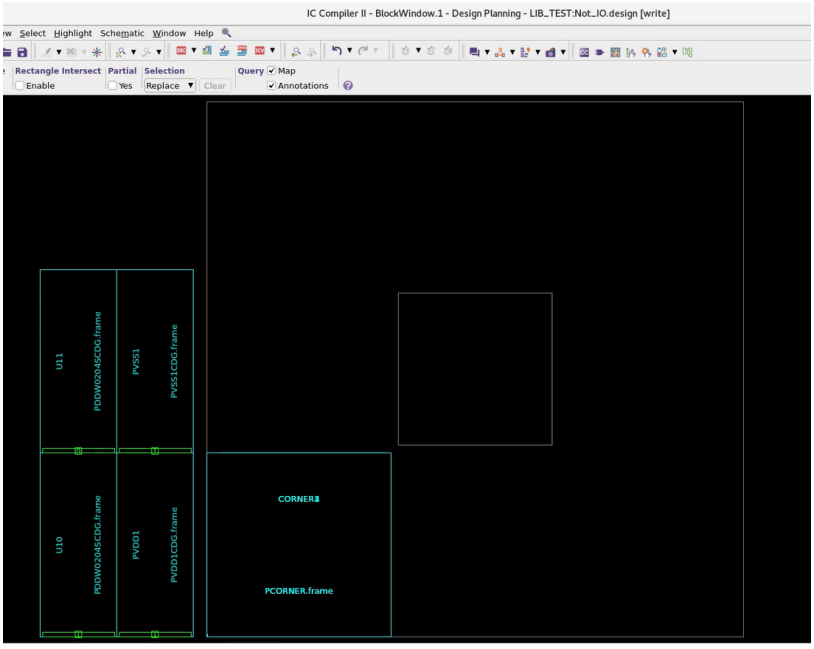
**Figura 14.** Lectura del archivo de Verilog de la compuerta NOT con 6 metales




Nota. La figura muestra la salida de la consola de IC Compiler II luego de hacer la lectura del archivo Verilog donde se observa que la herramienta reconoce la cantidad de puertos, módulos y *nets* del circuito.

El siguiente paso según el *script* principal es llamar al *script* del *floorplan*, de modo que corresponde crear celdas de alimentación e ingresar el comando `initialize_floorplan`, que en conjunto brindan el resultado que se observa en la Figura 15.

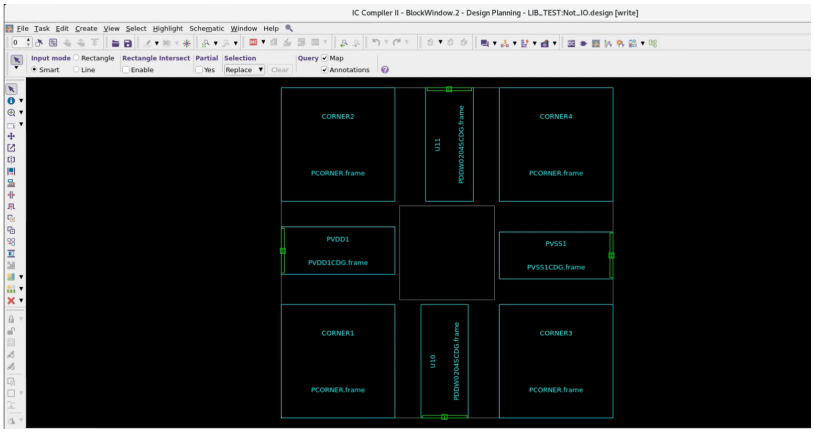
**Figura 15.** Creación del *floorplan* inicial de la compuerta NOT con 6 metales




Nota. Esta figura muestra el resultado en la interfaz gráfica de ICC2 al iniciar el *floorplan* del circuito sin realizar la colocación de las celdas en sus posiciones definitivas.

En la imagen de la Figura 16 se observa el resultado de la creación de anillo de entradas y salidas y la colocación de las celdas que corresponden al anillo.

**Figura 16.** Colocación de celdas de compuerta NOT con 6 capas de metal




Nota. En esta imagen se observa el resultado de la colocación de las celdas en el anillo de entradas y salidas.

Finalmente en el *script* del *floorplan* se realiza la creación de la malla de alimentación según el patrón definido y en las capas de metal indicadas en los comandos respectivos. En la Figura 17 se distingue que alrededor del perímetro del *core* del circuito hay redes de alimentación en color rojo y amarillo, que a su vez están conectadas a las líneas rojas y azules que forman la malla antes mencionada. Además, en el resultado de la figura anterior también se ejecutó la etapa de *placement* y de legalización de la colocación de las celdas.

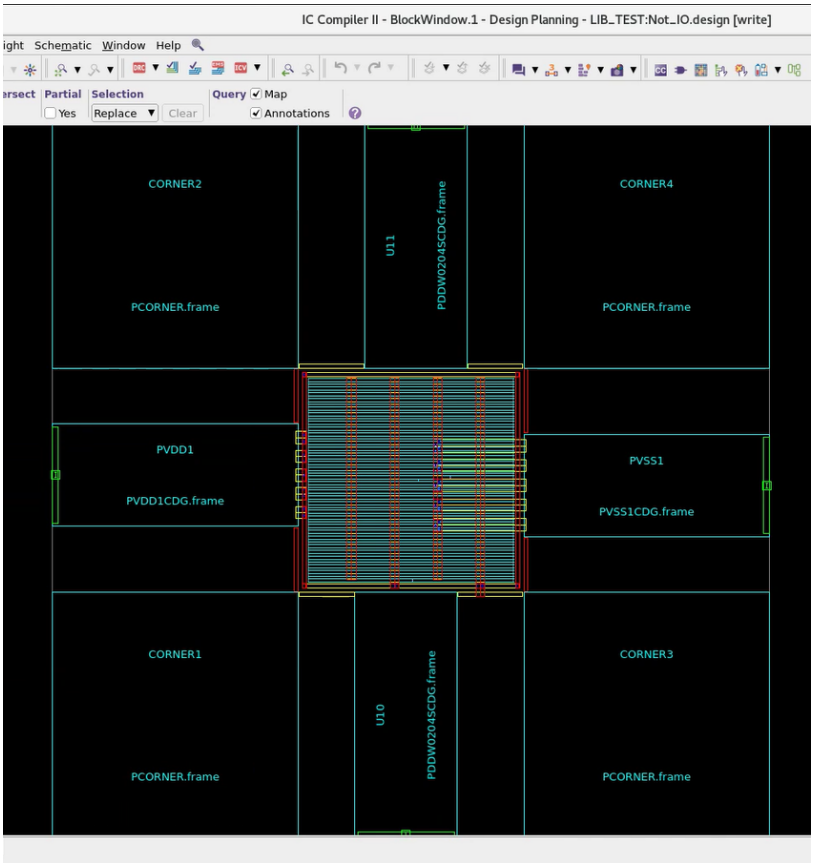
Para ver los scripts completos utilizados en esta etapa se puede consultar la documentación realizada por Lourdes Ruiz [15], ya que en estas pruebas iniciales se tomaron como referencia sin hacer cambios en los comandos ni opciones más allá de los cambios en las rutas y nombres de archivos que ya se detallaron en la sección 8.1.

En los *scripts* utilizados no se tiene una separación para el proceso de *routing* ni de las verificaciones DRC, colocación de celdas de relleno ni pruebas de antena, sino que están incluidas dentro del *script* principal de síntesis física. La Figura 18 presenta el *layout* del circuito con la ejecución del *routing* y la colocación de celdas de relleno que permiten aumentar la densidad de metal en las distinta capas y evitar problemas durante el proceso de fabricación del circuito.

Una vez creados los rellenos se realizaron las pruebas DRC y LVS. Entre los comandos que se ejecutaron en esta última fase de las pruebas iniciales con la compuerta NOT se incluye el comando de `signoff.fix_drc`. Este comando permite corregir automáticamente los errores DRC que se detectan en el diseño y que son solucionables por la herramienta. Sin embargo, IC Validator y IC Compiler no son capaces de solucionar todos los errores ya que pueden estar relacionados con otros factores. En la Figura 19 se puede ver la cantidad total de violaciones DRC que se detectaron en la primera prueba de la síntesis física con la compuerta NOT.

Si bien es cierto que la cantidad de errores DRC es considerablemente alta, es importante

**Figura 17.** Creación de la malla de alimentación de la compuerta NOT con 6 capas de metal

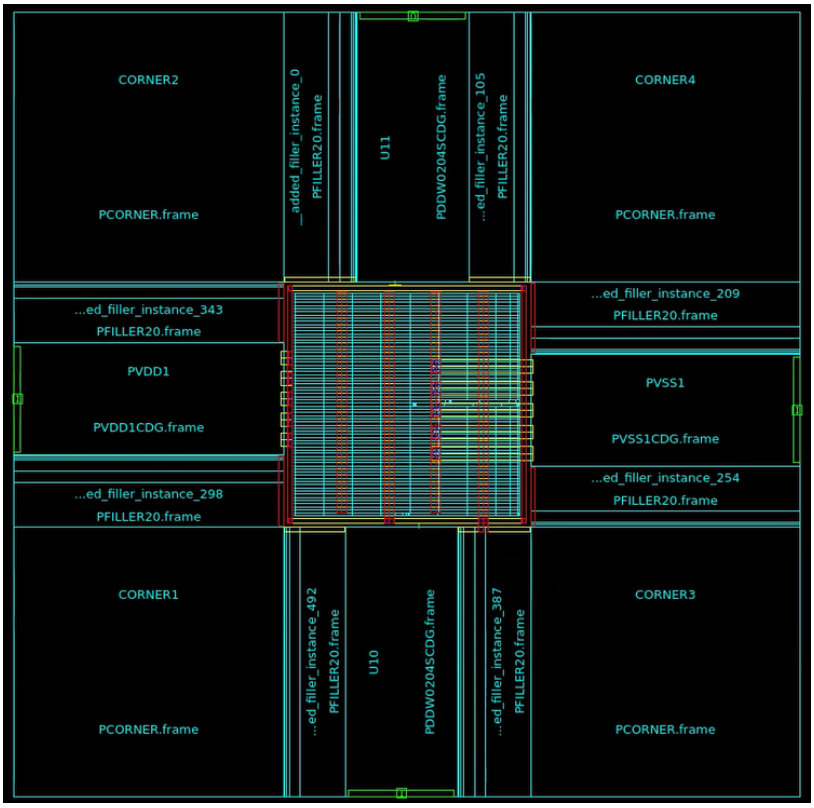



Nota. En esta figura se observa el resultado de la creación de la malla de alimentación para la compuerta NOT, mostrando cómo se distribuyen las conexiones de alimentación a través de las diferentes capas de metal.

destacar que fue la primera prueba sin realizar ninguna modificación en los *runsets* ni en los *scripts* que se emplearon con la tecnología de 180 nm. Sin embargo, ejecutar los *scripts* de esta manera permitió al equipo de trabajo identificar los puntos de mejora o incluso algunos errores que se generaban por el manejo inadecuado de las herramientas de Synopsys.

Si bien los resultados que se describen en esta sección son resumidos y no se profundiza en el análisis de cada etapa del proceso, permitieron al equipo de trabajo comprender de mejor manera el uso de las herramientas de Synopsys y sobre la metodología definida para la síntesis física. Durante el proceso de elaboración del resto de experimentos se eliminaron los archivos que quedaron obsoletos para no causar confusión a los futuros miembros que trabajen en el proyecto. En la carpeta con las versiones más recientes de la sintetización con el esquema de 6 capas de metal solo se encuentran los archivos que corresponden a una compuerta NOT sin incluir los *pads*. Además, como se indicó en secciones anteriores, este esquema de capas de metal no se utilizó en pruebas posteriores, dado que gracias a la comunicación con IMEC y a que el equipo de trabajo realizó una revisión más detallada de los documentos proporcionados respecto a la tecnología de fabricación de 65 nm de TSMC

**Figura 18.** *Routing* y colocación de rellenos de la compuerta NOT con 6 capas de metal

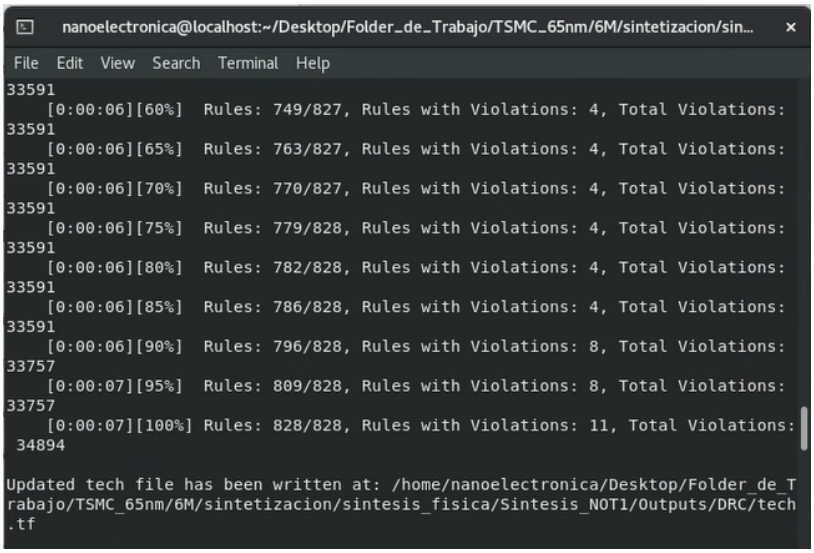



Nota. Esta figura muestra el resultado del *layout* del circuito luego de realizar el *routing* y la colocación de celdas de relleno.

[16], se concluyó que el esquema recomendado para la tecnología *flip-chip* es el de 9 capas de metal.

Para replicar este resultado es posible utilizar la carpeta que se creó que contiene los archivos necesarios e ingresar el comando `source sintesis_fisica_top.tcl` en la consola de IC Compiler II. Esto debería generar los mismos resultados que se muestran en esta sección.

**Figura 19.** Resultado de la prueba inicial de síntesis física de una compuerta NOT




Nota. En la figura se observa el resultado de la consola sobre la cantidad de errores DRC que se detectaron en la primera prueba de sintetización de la compuerta NOT con el esquema de 6 capas de metal.

# CAPÍTULO 9

## Cambio al esquema de 9 capas de metal de la tecnología de 65 nanómetros

Considerando que la cantidad de errores DRC que se obtuvieron en la sección anterior fue elevada, previo a realizar modificaciones en los *scripts* se decidió tomar la recomendación sobre la tecnología de 65 nm y se realizó el cambio al esquema de 9 capas de metal. Antes de llegar a la selección de archivos que se muestran en las secciones 7.1 y 7.2, se había seleccionado el esquema de 9 capas con 7 *tracks*. A continuación se describe un flujo de cambios efectuados entre los cuales se menciona que se realizó el cambio nuevamente al validar que existían errores en las librerías seleccionadas.

### 9.1. Pruebas para la implementación de mejoras a partir de la sintetización de una compuerta NOT

Para simplificar el procedimiento e identificar de manera más sencilla los errores al momento de ejecutar los comandos de ICC2 se utilizó la compuerta NOT como circuito base. En esta sección se describe una cronología de cambios realizados en los *scripts* y *runsets* que ayudaron a obtener resultados con una cantidad menor de errores DRC.

En la transición a 9 capas de metal se había decidido utilizar el esquema de 7 *tracks*, dado que era la misma cantidad de *tracks* que se utilizaba con las tecnologías que se trabajaron en versiones del proyecto anteriores. Debido a que para este punto del proyecto el equipo de trabajo ya había adquirido más conocimiento sobre los requerimientos y el funcionamiento de la tecnología, se decidió también realizar pruebas con el esquema de 9 *tracks*. Esta segunda opción fue la que se terminó empleando debido a que sus librerías fueron las que se adecuaron más a las necesidades de los circuitos trabajados.

### 9.1.1. Cambio en la elección de los archivos DB

Al principio, en la selección de archivos se tomó como referencia los archivos descritos en el manual que realizaron los estudiantes que participaron en el equipo de "El Gran Jaguar" en el año 2024 [33]. En este documento no se había indicado con claridad el esquema de metal escogido y para empezar con el desarrollo del proyecto el equipo había buscado rutas similares a las que se describen en el documento mencionado. Durante las primeras pruebas no se había logrado identificar que los archivos DB seleccionados no eran los adecuados, dado que en la etapa de síntesis lógica se estaban empleando archivos DB específicos que no eran congruentes con los que se seleccionaban en etapa de síntesis física.

A partir de esa información y teniendo más claridad sobre los significados de los nombres de los archivos DB se decidió seleccionar las siguientes librerías para el caso de las celdas estándar:

- tcbn65lpbc.db
- tcbn65lplt.db
- tcbn65lpml.db
- tcbn65lptc.db
- tcbn65lpwc.db
- tcbn65lpwc0d9.db

Mientras que para las librerías de entradas y salidas se seleccionaron los archivos siguientes:

- tpdn65lpnv2od3tc.db
- tpdn65lpnv2od3wc.db

Para estas librerías se resaltan lo siguientes significados según los nombres de los archivos:

- tcbn65lp: tecnología de 65 nm, *low power*.
- Terminación tc: caracterización típica de las celdas (*typical corner*).
- Terminación wc: caracterización del peor caso (*worst case corner*).
- Terminación bc: caracterización del mejor caso (*best case corner*).
- Terminación ml: caracterización del caso de carga media (*medium load corner*).
- Terminación lt: caracterización del caso de temperatura baja (*low temperature corner*).
- Terminación 0d9: caracterización a 0.9 V de voltaje de alimentación.

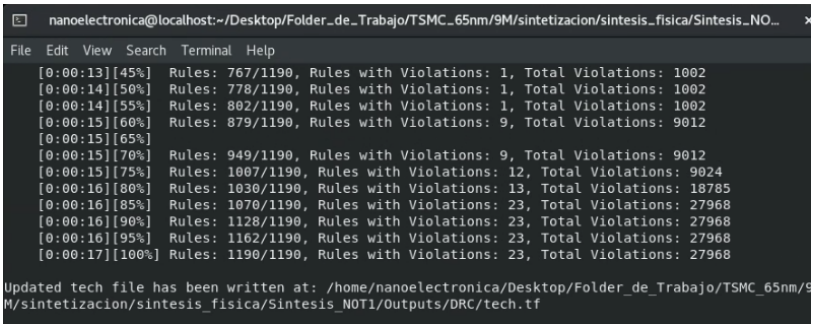
Si bien en el proceso de síntesis lógica se había seleccionado solo un archivo DB de cada tipo de celdas para la generación del *netlist*, en esta etapa se decidió añadir más archivos para que en caso de realizar cambios se tuvieran disponibles más opciones de celdas estándar y de entradas y salidas.

**Cuadro 9.1.** Comandos de lectura de librerías DB en Library Manager y IC Compiler II

```
1
2 # Comando para agregar el db-techfile de standard cells en Library Manager Tool
3 read_db {/home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t/Librerias
      /lib/ndm/db-techfiles/standard-cells/tcbn65lpbc.db}
4
5
6 # Comandos para agregar db-techfiles en IC Compiler II
7 lappend search_path "${DESIGN_REF_PATH}/lib/db_nldm"
8 set_app_var link_library [glob ${DESIGN_REF_PATH}/lib/db_nldm/*.db]
```

Nota. Este cuadro contiene los comandos utilizados para la lectura de los archivos DB en Library Manager Tool para la creación de librerías NDM y la en IC Compiler II.

**Figura 20.** Errores DRC con corrección de librerías DB y tecnología de 9 capas de metal




Nota. En esta imagen se puede observar la cantidad de errores DRC que se obtuvieron con las librerías DB corregidas y la tecnología de 9 capas de metal.

En la Figura 20 se observa que la cantidad de errores DRC disminuyó de una 34894 a 27968, lo cual fue la primera mejora significativa y esto permitió tener un indicio de que el esquema de 9 metales con 7 *tracks* era una mejor opción según las reglas de diseño.

### 9.1.2. Cambios efectuados en el *runset* de DRC y agregación del *runset* de antena en el *script* principal

Aún utilizando el esquema de 7 metales, se realizó una revisión del *runset* de DRC que se había seleccionado. En los experimentos de años anteriores se puede observar que

se realizaban modificaciones a valores específicos del *runset* [15], de manera que se buscaba reducir la cantidad de errores DRC pero modificando valores que no deberían tener cambios ya que son brindados por el fabricante de la tecnología.

A partir de esta observación se realizó una investigación, donde fue posible ubicar el documento que contiene las recomendaciones para la selección de opciones y definiciones en el *runset* de DRC. Este archivo se encuentra en el repositorio que contiene el *runset* descrito en la sección 7.2 en la carpeta con nombre `TSMC_DOC_WM`. El archivo contiene las directrices generales para determinar si las opciones del *runset* se deben comentar o no, lo cual no implica cambiar los valores de fábrica como se había manejado en las experimentaciones anteriores, sino que, dependiendo del diseño habilita o deshabilita las reglas que se tomarán en cuenta.

Las opciones finales que se habilitaron y deshabilitaron se muestran en el Cuadro 9.2. Estas opciones se seleccionaron considerando que el diseño elegido contiene celdas de las librerías del esquema LP (*Low power*), y también considerando las condiciones básicas del diseño. Algunas de las funciones que no se habilitaron requieren de mayor investigación dado que son funcionalidades más avanzadas que no se consideraron en los experimentos realizados en el presente trabajo. Además, en el cuadro se colocaron comentarios para orientar sobre el propósito de cada opción seleccionada.

**Cuadro 9.2.** Selección de opciones en el *runset* de DRC para la tecnología de 9 capas de metal

```
1
2  #define ICV_REPORT_DENSITY   // Activar para reportar densidad de metal
3  #define SNPSINDESIGN              // Activar para Synopsys InDesign
4
5
6  #define GUIDELINE_RES                    /* Activar para verificar las
       directrices de resistencia OD/PO */
7  //#define DISCONNECT_ALL_RESISTOR         /* Activar para desconectar todos los
       resistores entre el pad y el dispositivo */
8  //#define CONNECT_ALL_RESISTOR            /* Activar para conectar todos los
       resistores entre el pad y el dispositivo */
9  #define DEFINE_PAD_BY_TEXT              /* Activar para reconocer el PAD de E/S
       siguiendo el texto para las reglas de Latch-Up. */
10 #define GUIDELINE_LUP                    /* Activar para verificar Latch-up*/
11 #define GUIDELINE_ESD                    /* Activar para verificar las
       directrices ESD */
12 #define NW_SUGGESTED                     /* Activar para verificar las
       sugerencias de NW */
13 #define DATATYPE_WARNING                 /* Activar para mostrar advertencia del
       tipo de datos*/
14 #define MIXED_SCHEME                     /* Activar para permitir el uso de
       esquemas mixtos */
15 #define CHECK_LOW_DENSITY                /* Activar para revisar la densidad
       local baja  */
16 #define FRONT_END                        /* Activar para revisar reglas de Front-
       End */
17 #define BACK_END                         /* Activar para revisar reglas de Back-
       End */
18 #define FULL_CHIP                        /* Activar para revisar reglas de Full-
       Chip */
19 //#define WLCSP_2_MASK                   /* Activar para el nuevo proceso WLCSP 2
        */
20 //#define GP                             /* Activar para el proceso con celdas de
        GP */
21 //#define LPG                            /* Activar para el proceso con celdas
       LPG  */
22 #define LP                               /* Activar para el proceso con celdas LP
        */
23 //#define HALF_NODE                      /* Activar par el proceso N55 */
24 //#define _28K_AP                        /* Activar para un grosor de 28K en la
       capa AP */
25 //#define WLCSP_SEALRING                 /* Activar para utilizar el sealring
       WLCSP */
26 #define ChipWindowUsed                   /* Activar para especificar los bordes
       del chip directamente mediante las variables definidas por el usuario en el
       runset*/
```

Nota. Este cuadro contiene las opciones que se seleccionaron y comentaron en el *runset* que se emplea para las verificaciones DRC en tecnología de 9 capas de metal. Las líneas con una diagonal doble significa que están deshabilitadas.

### 9.1.3. Cambios efectuados en el *runset* de rellenos BEOL

Realizar cambios en el *runset* de verificaciones de DRC brindó un indicio de que se podía reducir la cantidad de errores al estudiar cada uno de los *runsets* y sus manuales correspondientes. Además, mientras se estaban ejecutando los *scripts* fue posible observar la advertencia que se muestra a continuación en el Cuadro 9.3 que está relacionada con los rellenos de metal BEOL que se estaban configurando.

**Cuadro 9.3.** Advertencia sobre configuración de rellenos BEOL en IC Compiler II

```
1
2    Rule: CSR.R.1:DUM1 : DUM1 is not allowed inside the empty area of chip corner.
              1002 errors
3    Rule: CSR.R.1:DUM2 : DUM2 is not allowed inside the empty area of chip corner.
              1001 errors
4    ...
```

Nota. Esta advertencia se muestra en la consola de IC Compiler II al momento de ejecutar el comando de DRC luego de crear los rellenos BEOL. Solamente se muestra una sección de ejemplo dado que hay más errores similares.

Dado que esta advertencia indica que hay celdas de relleno que están en un sitio no permitido, se acudió al manual de la tecnología que corresponde a las celdas de relleno BEOL que se encuentra en la misma ubicación del repositorio donde se puede acceder a este *runset* que se describe en la sección 7.2.

Uno de los puntos principales que se determinaron en la revisión, corresponde a las opciones `dmOnCorner` y `WithSealring` que se definen en el *runset*. La primera opción activa o desactiva la colocación de celdas de relleno en las esquinas del chip, mientras que la segunda opción habilita o deshabilita la colocación de rellenos en el área del *sealring*, es decir, en la orilla que sirve de protección en la fabricación. A partir del manual se determinaron las siguientes combinaciones:

- `dmOnCorner` desactivado y `WithSealring` desactivado: esta configuración no permite la colocación de rellenos en las esquinas pero sí en el área del *sealring*.
- `dmOnCorner` desactivado y `WithSealring` activado: con esta configuración no se permite la colocación de rellenos en esquinas ni en el *sealring*.
- `dmOnCorner` activado y `WithSealring` desactivado: con esta opción no existen restricciones para la colocación de rellenos, por lo cual se colocan en toda el área disponible del circuito

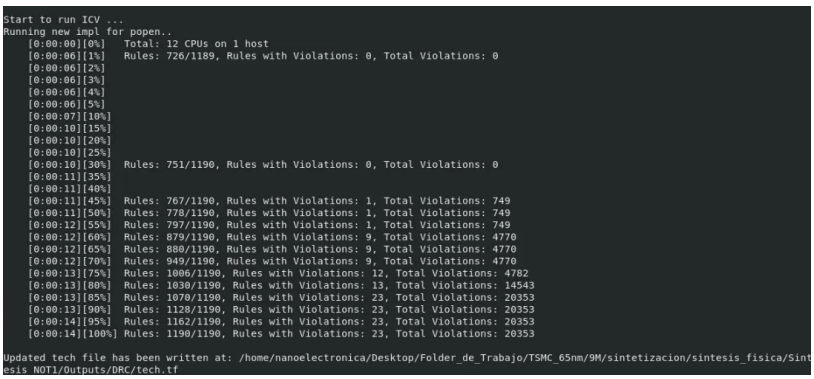
Estas opciones se pueden ver en los 3 casos que se describen en la imagen de la Figura 21, en donde corresponden al mismo orden en el que se describieron en la lista anterior.

**Figura 21.** Configuraciones de las opciones *dmOnCorner* y *WithSealring* en el *runset* de rellenos BEOL


Nota. Esta imagen está basada en el manual de fabricación de Mini@sic [16], y muestra las 3 configuraciones descritas de las combinaciones de las opciones `dmOnCorner` y `WithSealring`.

Por defecto el *script* tiene ambas opciones desactivadas (es decir que están comentadas), lo cual implica que se tenía una configuración similar a la selección 1 que se muestra en la figura. En la primera configuración que se realizó no se había seguido ninguna de las combinaciones mencionadas, ya que se habían activado ambas opciones. A pesar de que esta no es una configuración válida se había logrado reducir la cantidad de errores, como se muestra en la Figura 22.

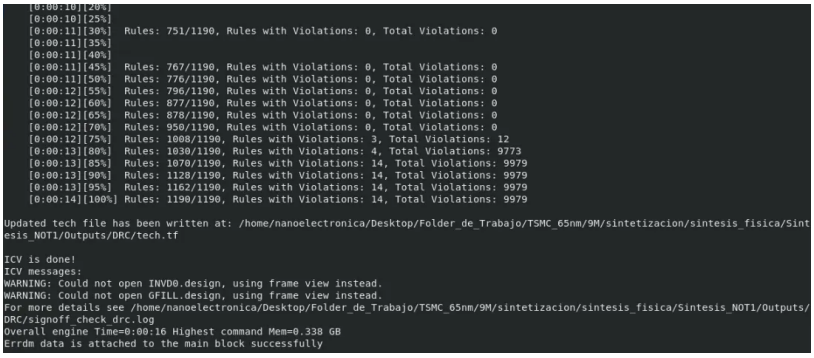
**Figura 22.** Pruebas iniciales con el primer cambio en las opciones del *runset* de rellenos BEOL




Nota. La imagen muestra una captura de de pantalla del resultado de la consola de IC Compiler II luego de realizar las pruebas DRC con los primeros cambios efectuados en el *runset* de rellenos BEOL.

Posterior a la investigación se determinó que la forma adecuada según los manuales de diseño disponibles y el *runset* de DRC, es la selección 2. Al probar esta segunda combinación se obtuvo una reducción notable en los errores, los cuales se muestran en la Figura 23.

**Figura 23.** Pruebas iniciales con el segundo cambio en las opciones del *runset* de rellenos BEOL




Nota. En esta figura se observa la reducción de errores que existió al elegir la configuración correcta del *runset* para el área de rellenos BEOL.

A continuación (en el Cuadro 9.4) se observan las definiciones utilizadas finalmente en el *runset* de rellenos BEOL que se empleó en las pruebas finales con la tecnología de 9 capas de metal. En el cuadro no se muestran las opciones completas, solo un segmento de las definiciones y un ejemplo de las configuraciones que corresponden a cada metal.

**Cuadro 9.4.** Listado de definiciones en el *runset* de rellenos BEOL para la tecnología de 9 capas de metal

```
1
2  // Primer apartado del runset de rellenos sobre espacios diponibles para colocar
       rellenos de metal
3  #define ChipWindowUsed
4  #define WithSealring
5  //#define dmOnCorner
6  //#define FILL_DAP_C
7
8  // Segundo apartado del runset sobre las de capas y tipos de rellenos
9  #define MIXED_SCHEME
10 //#define FILL_IN_SLOT
11 //#define N55HV
12 //#define N65_SIPH
13 //#define N65_SIPH_COUPE_EC
14 #define FILL_DM1
15 #define FILL_indDM1
16 #define FILL_OPCDM1
17 //#define _5K_THICK_M1
18 //#define _9K_THICK_M1
19 //#define _12K_THICK_M1
20 //#define _34K_THICK_M1
21 //#define CBM_OVER_M1
22 #define FILL_DM2
23 #define FILL_indDM2
24 #define FILL_OPCDM2
25 //#define _5K_THICK_M2
26 //#define _9K_THICK_M2
27 //#define _12K_THICK_M2
28 //#define _34K_THICK_M2
29 //#define CBM_OVER_M2
30
31 ...
32
33 ...
34
35 // Opciones relacionadas con la base de datos de entrada y salida
36 //#define MERGE_ORIGINAL_DESIGN
37 #define OUTPUT_GDS
38 #define AUTO_FILL_COMPRESSION
39
40 // Opciones para ICC2 y Synopsys In-Design
41 #define SNPSINDESIGN
42 #define USE_ICC2
43 //#define FILL_IN_MW_BLOCKAGE_LAYER
44 //#define EXTENDED_LAYERS
```

Nota. En el cuadro se listan las definiciones que se emplearon en el *runset* de rellenos BEOL que se utilizó para las pruebas con 9 capas de metal. Las opciones que aparecen comentadas fueron deshabilitadas en el archivo.

### 9.1.4. Implementación de *runset* de rellenos FEOL

En las primeras aproximaciones del equipo de trabajo solamente se escogió un *runset* de rellenos de metal, dado que de esa manera se había trabajado en las metodologías empleadas en años anteriores. Al revisar el material proporcionado por IMEC en el repositorio, fue posible identificar que existen dos *runsets* diferentes para la colocación de rellenos de metal, de modo que uno se nombra FEOL (que significa *Front End Of Line*) y otro BEOL (*Back End Of Line*). Estos dos archivos contienen funciones diferentes, lo cual implicó la necesidad de agregar comandos en los *scripts* para añadir este segundo archivo.

En el manual de comandos de ICC2 [34] se muestra la estructura de los comandos que permiten agregar las celdas de rellenos en el diseño. Originalmente se había utilizado el siguiente comando:

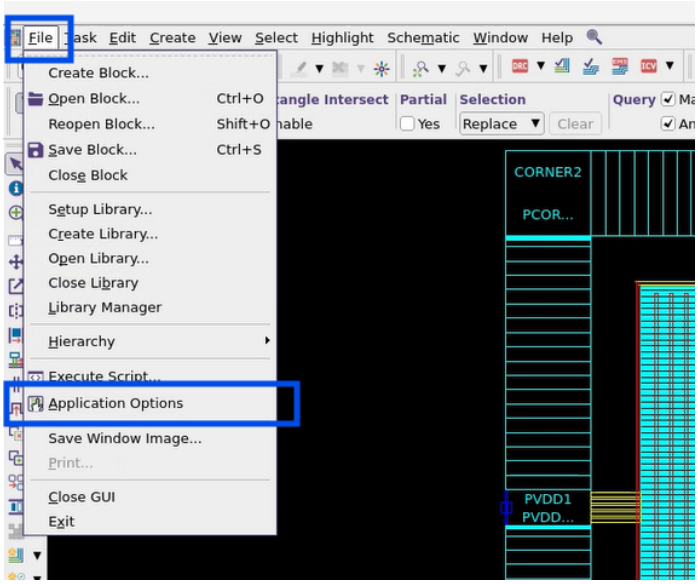
```
signoff_create_metal_fill -mode add
```

Como se distingue, este comando no indica información sobre el tipo de rellenos que se estaban insertando ni sobre los *runsets* seleccionados. En el manual de comandos de ICC2 fue posible conocer que para este comando existen opciones adicionales que permitieron hacer la integración de ambos tipos. La estructura del comando `signoff_create_metal_fill` según el manual [34] es la siguiente:

- `-mode`: con esta opción se define la acción que realizará el comando. En este caso se añaden celdas al diseño.
- `-foundry_fill_type`: con esta opción se define el tipo de relleno que se agregará, ya sea FEOL o BEOL.
- `-foundry_for_feol_fill`: con esta opción se define información de la tecnología para los rellenos FEOL. En este caso dado que se está utilizando tecnología de 65 nm y no hay una opción específica en las versiones más recientes de ICC2, se selecciona la opción *generic* [34].

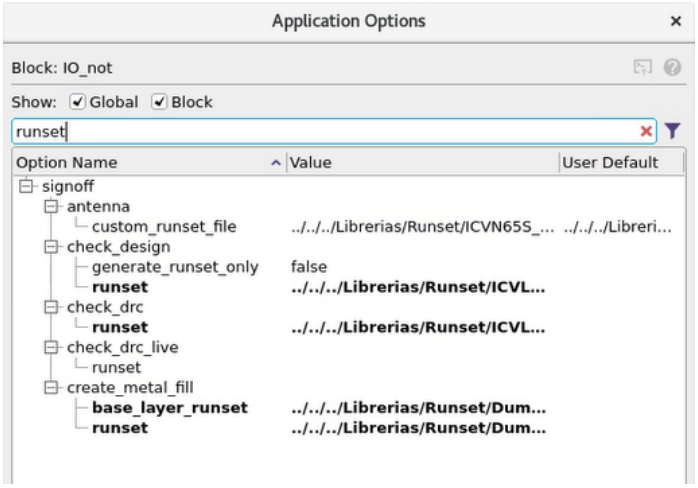
Antes de implementar correctamente este comando se habían ejecutado las pruebas sin considerar las opciones que se debían configurar. Dado que el comando presentaba errores se decidió explorar la interfaz gráfica de ICC2. En la Figura 24 se puede identificar cómo acceder a las opciones de la aplicación de ICC2, y en la Figura 25 se observa la ventana que permite aplicar los *runsets* a los procesos de verificaciones. Al mantener el cursor sobre las opciones la aplicación muestra la estructura de la opción de la aplicación, por lo cual usando estas funcionalidades se integró la configuración correcta de ambos *runsets* al *script* según se muestra en el Cuadro 9.5.

**Figura 24.** Abrir menú de opciones de la aplicación en ICC2




Nota. En la imagen se muestran los pasos que se deben seguir para abrir la ventana de las opciones de la aplicación de ICC2.

**Figura 25.** Ventana de selección de opciones de aplicación en ICC2




Nota. En la captura de pantalla se observa el resultado de abrir la ventana de opciones de la aplicación de ICC2 con un filtro aplicado para encontrar las definiciones de los *runsets*.

**Cuadro 9.5.** Implementación de *runsets* de rellenos FEOL y BEOL en IC Compiler II

```
1
2  set_app_options -list {signoff.create_metal_fill.runset {../../../Librerias/Runset
       /Dummy_BEOL_ICV_65nm.26_2a}}
3
4  set_app_options -list {signoff.create_metal_fill.base_layer_runset {../../../
       Librerias/Runset/Dummy_FEOL_ICV_65nm.26_1a}}
5
6  ...
7
8  signoff_create_metal_fill -mode add  -foundry_fill_type beol
9
10 signoff_create_metal_fill -mode add -foundry_fill_type feol -foundry_for_feol_fill
       generic
```

Nota. En este cuadro se distinguen los comandos que se agregaron al *script* principal de síntesis física para incluir los *runsets* de rellenos BEOL y FEOL.

Cabe destacar que una vez implementado el *runset* de rellenos FEOL se realizaron las modificaciones pertinentes como se muestra en el Cuadro 9.6 para adecuarlo según las recomendaciones brindadas por IMEC y TSMC. En este cuadro solamente se incluyeron las definiciones importantes, por lo cual cabe resaltar que el archivo completo contiene más opciones pero que no fueron manipuladas por el equipo del proyecto.

**Cuadro 9.6.** Listado de definiciones en el *runset* de rellenos FEOL para la tecnología de 9 capas de metal

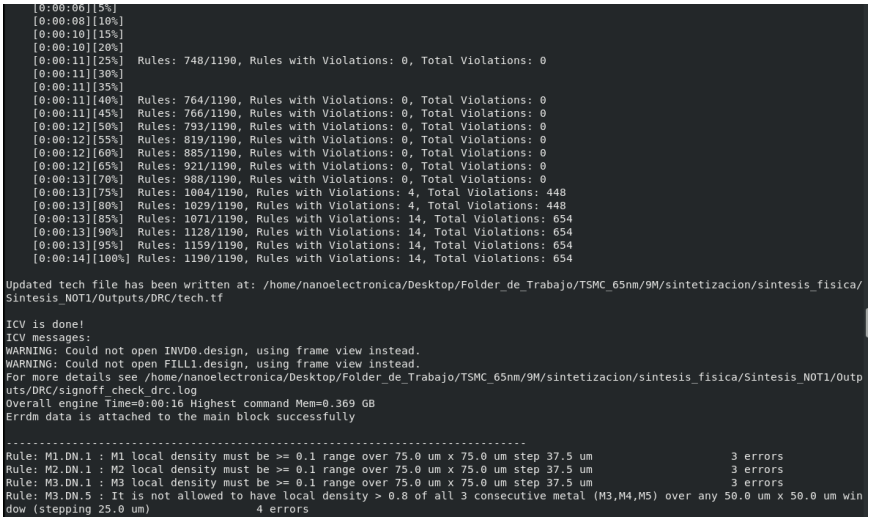
```
1
2  //******* Area del chip definida por el usuario **************************
3  //#define UseprBoundary        // usar la capa prBoundary(108) para definir la
       ventana del chip
4  #define ChipWindowUsed        // usar las variables siguientes para definir la
       ventana del chip
5  XLB : double = 0.0;           // Coordenadas de las equinas del chip
6  YLB : double = 0.0;
7  XRT : double = 1000.0;
8  YRT : double = 1000.0;
9
10 //******* Opciones para el area vacia en las esquinas del chip ***********
11 #define WithSealring          // indica que ya existe una estructura de sealring
12 //#define dmOnCorner          // habilita la insercion de DOD/DPO en las esquinas
13
14 //******* Seleccion de patrones a insertar ******************************
15  //#define FILL_TCD_PATTERN    // habilita la insercion de patron DummyTCD
16 #define FILL_DOD_DPO          // habilita la insercion de DOD/DPO
17
18 //***** Variables que no deben modificarse *****
19
20 //******* Opciones para entrada/salida de base de datos ******************
21 //#define MERGE_ORIGINAL_DESIGN
22 #define OUTPUT_GDS               // genera salida en formato GDSII (por defecto)
23 #define AUTO_FILL_COMPRESSION     // comprime automaticamente los datos de relleno
24
25 //******* Opciones para ICC/ICC2 *****************************
26 #define SNPSINDESIGN             // habilita el flujo ICC
27 #define USE_ICC2                 // habilita procesamiento de capas del sistema
       para ICC2
28 //#define FILL_IN_MW_BLOCKAGE_LAYER  // habilita relleno en la capa de bloqueo
       Milkyway
29 //#define EXTENDED_LAYERS            // habilita uso de capas extendidas
```

Nota. El cuadro contiene las definiciones importantes que se consideraron en el *runset* de rellenos FEOL previo a implementarlo en el flujo.

## 9.2. Resultados de la síntesis física de la compuerta NOT con los cambios efecutados

Al finalizar de implementar los cambios descritos en la sección anterior, se ejecutó la síntesis física de la compuerta NOT con la tecnología de 9 capas de metal. El resultado inicial fue el que se muestra a continuación en la Figura 26.

**Figura 26.** Resultado del DRC de la sintetización de la compuerta NOT con 9 capas de metal

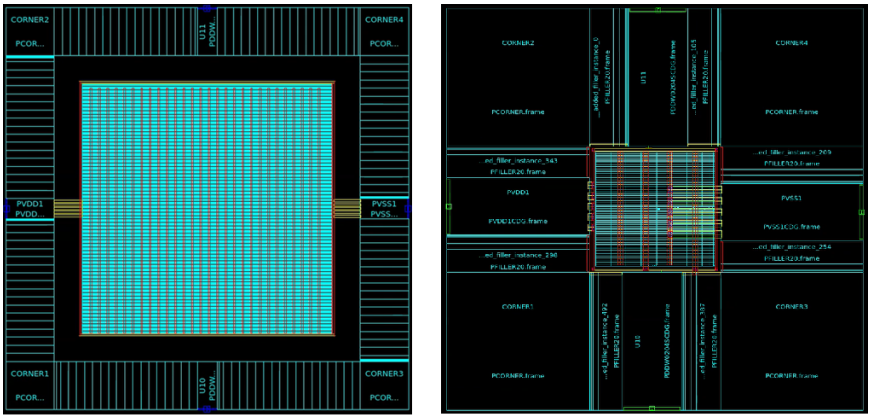



Nota. En la imagen se muestran los pasos que se deben seguir para abrir la ventana de las opciones de la aplicación de ICC2.

En la Figura 27 se distingue la diferencia entre los resultados de la sintetización de la compuerta NOT de la Figura 19 con 6 capas de metal y que utiliza los *scripts* empleados en las pruebas de versiones anteriores del proyecto, contra el resultado de las nuevas modificaciones que se implementaron en la sección anterior del documento.

A simple vista es posible observar que se realizaron cambios notables como el tamaño del *core* y del anillo de entradas y salidas, así como también se observa una diferencia notable en la aplicación de rellenos. Los rellenos del lado izquierdo se ven más consistentes y agrupados, lo cual es congruente con la solución de errores de densidad que presentaba el circuito de 6 capas de metal.

**Figura 27.** Comparación entre el *layout* de la compuerta NOT con 6 y 9 capas de metal




Nota. En el lado izquierdo de la figura se observa el resultado de la sintetización con 9 capas de metal y a la derecha el resultado con 6 capas de metal sin modificaciones en los *scripts* tomados de metodologías anteriores.

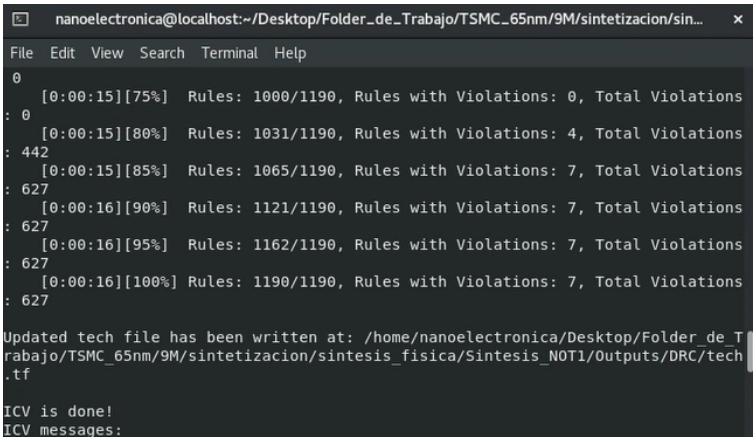
Finalmente se implementaron algunos cambios en los *scripts* que permitieron reducir en menor escala la cantidad de errores, de modo que se implementó el cambio en el archivo `star.map_9M` que se muestra en el Cuadro 9.7 y también en la utilización de comandos como `legalize_placement`, los cuales contribuyeron a reducir los errores como se observa en la Figura 28.

**Cuadro 9.7.** Arhivo de mapeo modificado

```
conducting_layers
PO     poly
M1       metal1
M2       metal2
M3       metal3
M4       metal4
M5       metal5
M6       metal6
M7       metal7
M8       metal8
M9       metal9
AP      metal10
via_layers
CO polyCont
VIA1       via1
VIA2       via2
VIA3       via3
VIA4       via4
VIA5       via5
VIA6       via6
VIA7       via7
VIA8       via8
RV        via9
```

Nota. El cuadro muestra el contenido del archivo `star.map_9M` modificado según los archivos de tecnología.

**Figura 28.** Resultado del DRC con los cambios completos de la sección en la compuerta NOT




Nota. Esta imagen presenta el resultado de la consola al sintetizar la compuerta NOT luego de terminar de aplicar los cambios que se describen en el presente capítulo del documento.

# CAPÍTULO 10

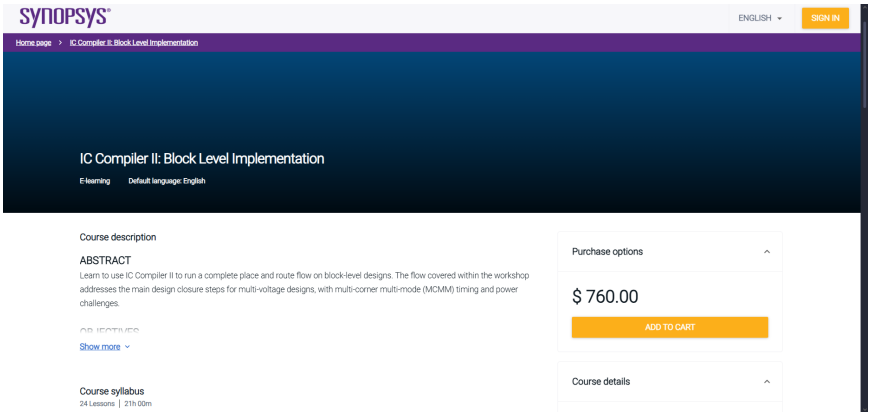
## Restructuración de los *scripts* de sintetización a partir de las recomendaciones de la documentación de IC Compiler II

## 10.1. Cursos de ICC2 de Synopsys

Durante el desarrollo del proyecto se tuvo acceso a la plataforma de aprendizaje de Synopsys, en donde se encuentran disponibles diversos cursos para cada una de las herramientas de la plataforma. En el apartado de ICC2 fue posible observar que existe un curso titulado ÏC Compiler II: Block Level Implementation"[35], el cual proporciona información valiosa sobre las mejores prácticas y recomendaciones para el desarrollo del proceso de síntesis en ICC2. Tomar este curso también representó una oportunidad del equipo de trabajo para familiarizarse con las herramientas y para tener una mejor comprensión de los conceptos fundamentales del *floorplaning*, *placement* y *routing* primordialmente, pero también permitió conocer otras utilidades como la sintetización del árbol de reloj, la generación de librerías y la utilización de herramientas integradas de IC Validator para las verificaciones DRC, antena y LVS.

El curso tiene una duración total de aproximadamente 21 horas (ver Figura 29), y en él se encuentra material disponible para la realización de laboratorios que muestran de mejor manera el orden de cada una de las etapas que se deben seguir según el flujo de diseño recomendado. En el presente capítulo se detallan las correcciones que se realizaron en los *scripts* a partir de recibir estos cursos y que fueron los documentos que permitieron obtener los resultados finales de la experimentación realizada en el año 2025.

**Figura 29.** Curso de IC Compiler II en la plataforma de aprendizaje de Synopsys




Nota. La captura de pantalla muestra la página principal del curso *Block Level Implementation* de IC Compiler II en la plataforma de aprendizaje de Synopsys.

## 10.2. Implementación de mejoras en los *scripts* de la compuerta NOT

### 10.2.1. Restructuración del *script* de configuración

Primero se realizaron cambios en el *script* de configuración llamado `setup.tcl`, ya que en él se deberían encontrar las definiciones generales del proyecto y la configuración de variables y opciones de la aplicación que se estarán utilizando en todo el proceso. Este fue uno de los puntos principales que se corrigieron, dado que en versiones anteriores las opciones se configuraban en diferentes archivos y por ejemplo, los comandos de selección de *runsets* se ejecutaban en las últimas fases del diseño.

**Cuadro 10.1.** Eliminación de archivos en el *script* de configuración

```
1
2 # ============================================================================
3 # Limpieza de archivos previos
4 # ============================================================================
5 sh clear
6 sh rm -f check_design*
7 sh rm -f default-*
8 sh rm -f *ems
```

Nota. Los comandos del presente cuadro se agregaron en el inicio del *script* de configuración `setup.tcl` para eliminar archivos innecesarios al reiniciar el proyecto.

En el Cuadro 10.1 se muestra el primer cambio implementado, y es la eliminación de archivos que quedan obsoletos al reiniciar el proceso de síntesis física. Se decidió agregar estos comandos para obtener un mejor orden y control en la carpeta del proyecto, ya que originalmente los archivos se acumulaban debido a que no se sobrescribían, lo cual generaba incomodidad al momento de buscar los archivos relevantes. Si es necesario revisar estos archivos, se pueden generar copias y almacenar en otras ubicaciones para no causar desorden en la carpeta principal del proyecto.

Para dar inicio al proceso de configuración del entorno del programa, se agregó la ruta de referencia con el comando `set DESIGN_REF_PATH`, el cual almacena la variable para facilitar la colocación de rutas absolutas que tienen un origen común. Posterior a este comando, se definen las rutas de los archivos de tecnología, mapeo y TLUPlus, así como también las rutas de las librerías NDM que se utilizarán en el proceso. La selección correcta de la librería NDM es fundamental para realizar un proceso adecuado, ya que estas se generan con la herramienta de Library Manager Tool pero necesitan ser adaptadas al entorno de ICC2.

**Cuadro 10.2.** Configuración del directorio de referencia y preparación de archivos de entrada

```
1
2  set command_log_file "./Logs/command.log"
3
4  set DESIGN_REF_PATH "/home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9
       M_9t/Librerias"
5
6  set TECH_FILE        "${DESIGN_REF_PATH}/<ruta relativa al techfile>"
7  set MAP_FILE         "${DESIGN_REF_PATH}/<ruta relativa al archivo de mapeo>"
8  set TLUPLUS_FILE "${DESIGN_REF_PATH}/<ruta relativa al archivo TLUPlus>"
9
10 set NDM_REFERENCE_LIB_DIRS  " \
11     ${DESIGN_REF_PATH}/lib/ndm/TSMCWorkspace.ndm"
12
13 set LIBRARY_FILES "${NDM_REFERENCE_LIB_DIRS}"
14
15 # Copiar archivos de entrada verilog y sdc
16 file copy -force "<ruta al archivo verilog del proceso front-end>" "./Inputs/"
17
18 file copy -force "<ruta del archivo sdc del proceso front-end>" "./Inputs/"
```

Nota. El cuadro contiene la primera sección del *script* del `setup.tcl` en donde se configuran las rutas de los archivos de referencia y se copian los archivos de entrada necesarios para la síntesis física.

El comando `file copy` permite realizar la copia del archivo obtenido en la síntesis lógica (proceso *front-end*) y se eliminó en versiones posteriores del *script* dado que se decidió trabajar de manera independiente los procesos de síntesis lógica y síntesis física. Esto se decidió ya que se utilizan diferentes máquinas para ejecutar cada uno de los procesos. Sin embargo, el comando es útil para copiar los archivos de entrada y no manipular en ningún momento los archivos originales.

**Cuadro 10.3.** Configuración de variables de potencia y capas

```

set MW_POWER_NET     "VDD"
set MW_POWER_PORT    "VDD"
set MW_GROUND_NET    "VSS"
set MW_GROUND_PORT   "VSS"

set MIN_ROUTING_LAYER "M1"
set MAX_ROUTING_LAYER "M9"
```

Nota. Esta sección del *script* de configuración contiene las definiciones de las variables de potencia y las capas mínimas y máximas de enrutamiento.

En el Cuadro 10.3 se observa la configuración de los nombres de los nodos y puertos de alimentación y las capas mínima y máxima de enrutamiento. En esta sección no se procesa directamente un comando, solamente se definen las variables para su posterior utilización.

La siguiente etapa que se muestra en el *script* de configuración corresponde a la creación de la librería principal que se utilizará en el flujo de síntesis física. Existen diversas opciones que se pueden seleccionar para la creación de la librería. La primera parte, que se muestra en el Cuadro 10.4, corresponde a la configuración de las rutas de los archivos DB que se utilizarán como referencia para la creación de la librería. Con el comando `lappend search_path` se define una ruta a la cual la herramienta de ICC2 acudrirá cuando no se le brinde la ruta específica de un archivo. Posterior a este comando, se utiliza el comando `set_app_var link_library` para definir los archivos DB a los cuales la herramienta puede acudir si es necesario utilizar una referencia de estas librerías.

En la segunda parte del código ya se muestra directamente la creación de la librería, de modo que se define el nombre de la librería con la variable `ndm_design_library` y se utiliza el comando `create_lib` para generar la librería con las referencias y tecnología seleccionadas.

Según la documentación de Synopsys [34], existen otras opciones que se pueden agregar al comando `create_lib`, pero en este caso se decidió utilizar las opciones básicas que corresponden al archivo de tecnología y las librerías de referencia. Estas librerías de referencia pueden ser archivos NDM, LEF, FRAME o librerías Milkyway. En este caso, la entrada del comando corresponde a las librerías NDM que se crearon con la herramienta de Library Manager según lo indicado en la sección 7.1.

**Cuadro 10.4.** Creación de librerías para el bloque de IC Compiler II

```
1
2  # Archivos DB de referencia
3  # ================================================================
4  lappend search_path "${DESIGN_REF_PATH}/<ruta hacia archivos DB>"
5
6  set_app_var link_library [glob ${DESIGN_REF_PATH}/<ruta hacia archivos DB>/*.db]
7
8  set_host_options -max_cores 12
9
10
11 # ================================================================
12 # Crear la lib
13 # ================================================================
14
15 set ndm_reference_library ${NDM_REFERENCE_LIB_DIRS}
16 set ndm_design_library LIB_TEST
17 puts "Creando libreria: $ndm_design_library"
18
19 sh rm -rf $ndm_design_library
20 create_lib -technology $TECH_FILE -ref_libs $LIBRARY_FILES ${ndm_design_library}
```

Nota. Estos comandos permiten generar la librería principal que se estará utilizando en el flujo y en el bloque principal creado en IC Compiler II.

Una vez realizada la creación de la librería se puede proceder a cargar el *netlist* del circuito como se puede verificar en el Cuadro 10.5 en donde se carga el archivo Verilog y SDC. Luego, se utiliza el comando `link_block` que realiza la vinculación entre el diseño y las librerías para luego generar un reporte donde se indique si existen inconsistencias en las celdas del diseño y la librería.

**Cuadro 10.5.** Lectura de archivos de entrada con información del diseño

```
1 set INPUT_name "not_mapped"
2
3 read_verilog ./Inputs/$INPUT_name.v
4 read_sdc -echo ./Inputs/$INPUT_name.sdc
5
6 link_block
7 report_design_mismatch -verbose
```

Nota. Esta sección del código realiza la lectura de los archivos de entrada con la información del diseño que se trabajará en el proceso de síntesis física.

La siguiente etapa del circuito corresponde a la configuración de los archivos que contienen la información parasitaria de la tecnología. Para esto se utiliza el comando `set_parasitic_tech` el cual recibe como entradas el archivo TLUPLUS y el archivo MAP que realiza la traducción de la terminología de las capas de metal en el archivo de tecnología ha-

cia la que se utiliza en el archivo TLUPLUS. Posterior a este comando, se utiliza el comando `set_parasitic_parameters` para definir los archivos que contienen la información parasítica en las condiciones de *early* y *late* (los valores mínimos y máximos) según la tecnología. Finalmente, se utiliza el comando `report_lib` para generar un reporte con la información parasítica de la librería que se creó previamente y que se utiliza para validar si los archivos se cargaron y configuraron correctamente.

**Cuadro 10.6.** Configuración de archivos con información de parásitos

```
read_parasitic_tech -tlup $TLUPLUS_FILE -layermap $MAP_FILE
set_parasitic_parameters -early_spec $TLUPLUS_FILE -late_spec $TLUPLUS_FILE

report_lib -parasitic_tech ${ndm_design_library}
```

Nota. Los comandos del cuadro permiten cargar información parasítica de las celdas y metales según la tecnología.

El el Cuadro 10.7 se muestra inicialmente un apartado que contiene opciones de la aplicación. Estas opciones corresponden primero a la configuración de vías redundantes en el enrutamiento, de modo que se puedan insertar vías adicionales en rutas críticas [23]. La segunda opción hace referencia a la revisión y corrección automática de errores DRC luego de finalizar el enrutamiento.

La segunda que se distingue en el cuadro es para la configuración de variables que permiten automatizar el flujo de diseño ya que se utilizan en varias secciones del proceso. Es importante que la variable `CIRCUIT_NAME` coincida con el nombre del módulo principal del archivo verilog. Otra de las secciones importantes es sobre el ajuste de las dimensiones del *core* y el *offset*, ya que la suma de dos veces el *offset* más el tamaño del *core* debe ser igual al tamaño total del *die*, que en este caso es de 1 um.

**Cuadro 10.7.** Configuración de variables de automatización y opciones de la aplicación

```
# ============================================================================
# Opciones del programa
# ============================================================================

set_app_options -list { \
    route.common.post_detail_route_redundant_via_insertion high \
    route.common.concurrent_redundant_via_mode insert_at_high_cost \
}

set_app_options -name signoff.fix_drc.check_drc -value global

# ============================================================================
# Variables para automatizar el flujo
# ============================================================================
set CLOCK_SYNTHESIS FALSE
set SIDE_LENGTH {620 620}
set CORE_OFFSET 190

set CIRCUIT_NAME "IO_not"
set ANILLO_NAME "ANILLO_IO"

# Ajuste de dimensiones
set x1 [expr {$CORE_OFFSET + 10}]
set y1 [expr {$CORE_OFFSET + 10}]
set adjusted_polygon_x [expr {[lindex $SIDE_LENGTH 0] + $CORE_OFFSET - 10}]
set adjusted_polygon_y [expr {[lindex $SIDE_LENGTH 1] + $CORE_OFFSET - 10}]
set POLYGON_DIMENSIONS "$adjusted_polygon_x $adjusted_polygon_y"

puts "Clock synthesis enabled: $CLOCK_SYNTHESIS"
puts "Circuit name: $CIRCUIT_NAME"
```

Nota. Los comandos del cuadro permiten cargar información parasitaria de las celdas y metales según la tecnología.

Finalmente, en el archivo de *setup.tcl* se utilizan comandos para la creación de los directorios donde se generarán las salidas de los reportes y también se configuran las opciones de la aplicación con las rutas de las carpetas creadas. Lo último configurado en el Cuadro 10.8 es para definir las rutas de los *runsets* que se utilizarán en las verificaciones físicas. Estos últimos comandos fueron los que se obtuvieron a partir de la exploración de la interfaz gráfica de ICC2 que se mostró en la Figura 25.

**Cuadro 10.8.** Configuración de carpetas de salida y *runsets*

```
1
2  # ========================================================================
3  # Carpetas de salida
4  # ========================================================================
5  file mkdir ./Outputs/IO/
6  file mkdir ./Outputs/DRC/
7  file mkdir ./Outputs/Fill/
8  file mkdir ./Outputs/Antena/
9
10 set_app_options -list {signoff.check_design.run_dir {./Outputs/DRC/}}
11 set_app_options -list {signoff.check_drc.run_dir {./Outputs/DRC/}}
12 set_app_options -list {signoff.fix_drc.run_dir {./Outputs/DRC/}}
13 set_app_options -list {signoff.create_metal_fill.run_dir {./Outputs/Fill/}}
14
15 # ========================================================================
16 # Runsets
17 # ========================================================================
18
19 set_app_options -list {signoff.check_design.runset {<ruta hacia runset DRC>}}
20 set_app_options -list {signoff.check_drc.runset {<ruta hacia runset DRC>}}
21
22 set_app_options -list {signoff.create_metal_fill.runset {<ruta hacia runset de
       rellenos BEOL>}}
23 set_app_options -list {signoff.create_metal_fill.base_layer_runset {<ruta hacia
       runset de rellenos FEOL>}}
24
25 set_app_options -list {signoff.antenna.custom_runset_file {<ruta hacia runset de
       antena>}}
26
27 set_app_options -list {signoff.create_metal_fill.fix_density_errors true}
```

Nota. Los comandos del cuadro permiten cargar información parasitaria de las celdas y metales según la tecnología.

### 10.2.2. Elaboración de las etapas de *floorplan* y *placement* en el archivo *floorplan.tcl*

En el archivo de configuración no se realizaron cambios tan significativos dado que en él no influyen directamente aspectos como el orden en la colocación de los comandos. Principalmente se vio afectado en cuanto a que se movieron configuración de opciones que estaban en los otros dos *scripts*. Sin embargo, el archivo de *floorplan.tcl* sí sufrió más modificaciones y restructuraciones para cumplir con el flujo recomendado en los cursos recibidos.

En el Cuadro 10.9 se distinguen los primeros comandos que se utilizan en la etapa de *floorplaning*, de modo que primero se utiliza el comando `set_app_options` para definir el estilo de macro que se utilizará en el diseño. Posterior a este comando, se utiliza el comando `initialize_floorplan` para iniciar la creación del *floorplan*, donde cabe destacar las opciones que se indican a continuación según el manual de ICC2 [34].

- `use_site_row`: se utiliza para definir el estilo de los sitios que se utilizarán para la colocación de las celdas.
- `site unit`: define el nombre del sitio en donde se ubicarán las celdas. Este sitio depende de la definición del atributo correspondiente en las celdas, el cual se puede observar tanto en el LEF como en la librería NDM.
- `keep_all`: esta opción permite mantener todas las áreas y celdas del diseño.
- `side_length`: define las dimensiones laterales del *core* del diseño.
- `core_offset`: define el espacio entre el *core* y el borde del *die*.

Finalmente en este segmento del *script* se utilizan los comandos `shape_blocks` y `set_attribute` para definir la simetría de las celdas en el sitio seleccionado.

**Cuadro 10.9.** Configuración inicial de floorplan

```

# Configura el estilo de macro (bloques grandes)
set_app_options -name plan.macro.style -value freeform

# Iniciar el floorplan
initialize_floorplan \
  -use_site_row \
  -site unit \
  -keep_all \
  -side_length $SIDE_LENGTH \
  -core_offset $CORE_OFFSET

shape_blocks

set_attribute [get_site_defs unit] symmetry {X Y}
```

Nota. Estos comandos muestran la configuración de opciones y creación inicial del *floorplan* del diseño.

La siguiente etapa corresponde a la colocación de las celdas (*placement*) y la colocación de los pines del bloque. En el Cuadro 10.10 se observa que primero se configuran opciones para la legalización avanzada de las celdas y para la creación automática de bloqueos en el *floorplan*. Posterior a esto, se utiliza el comando `create_placement` para realizar una colocación preliminar de las celdas en el *floorplan*, utilizando opciones que permiten optimizar la colocación en función de la congestión del diseño. Luego se utiliza el comando `check_legality` para verificar que la colocación realizada sea legal.

Después de esto, se definen las capas permitidas para la colocación de los pines del bloque con el comando `set_block_pin_constraints`, y finalmente se colocan los pines con el comando `place_pins`.

**Cuadro 10.10.** Colocación de celdas y creación de pines

```
1
2  # Configura opciones para legalizar y colocar macros
3  set_app_options -name place.legalize.enable_advanced_legalizer -value true
4  set_app_options -name place.coarse.fix_hard_macros -value false
5  set_app_options -name plan.place.auto_create_blockages -value auto
6
7  # Coloca de forma preliminar en el floorplan
8  create_placement -floorplan -congestion -effort high -congestion_effort high
9  check_legality
10
11 # Define las capas permitidas para los pines del bloque
12 set_block_pin_constraints -self -allowed_layers {M2 M3 M4 M5 M6}
13
14 # Coloca los pines
15 place_pins -self
```

Nota. El cuadro contiene la sección en la cual se realiza la colocación de celdas y pines en el *floorplan*. Esa sección ya contiene las modificaciones implementadas según el flujo recomendado.

El siguiente segmento del código contiene comandos que permiten la limpieza de estructuras previas de poder. Esto se realiza dado que se tomaron como referencia los cursos de ICC2, sin embargo, no es una parte fundamental del proceso, dado que, según la metodología y flujo empleados en el proyecto, no se utiliza un *floorplan* previo que contenga estructuras de poder. Sin embargo, se decidió mantener esta sección en el *script* para tener un mejor control en el proceso.

La siguiente fase que se observa en la Figura 10.11 corresponde a la creación del anillo de entradas y salidas del diseño, así como también la creación de las celdas que se utilizarán para los *pads* de alimentación. Aún no se colocan las celdas en esta sección, solamente se crean a partir de la información de las librerías. Para definir los nombres de las celdas, cabe destacar que es necesario realizar la documentación, dado que en futuras actualizaciones de las tecnologías se deberían buscar celdas equivalentes con otros nombres o aquellas que mejor se adapten al diseño.

**Cuadro 10.11.** Creación de anillo de entradas y salidas y celdas de alimentación

```
1
2  ###############################################################################
3  # ETAPA 2: LIMPIEZA DE ESTRUCTURAS PREVIAS DE POWER/GROUND
4  ###############################################################################
5
6  remove_pg_strategies -all
7  remove_pg_patterns -all
8  remove_pg_regions -all
9  remove_pg_via_master_rules -all
10 remove_pg_strategy_via_rules -all
11
12 ###############################################################################
13 # ETAPA 3: ANILLO IO Y PADS DE PODER
14 ###############################################################################
15
16
17 # Crea el anillo IO
18 create_io_ring -name $ANILLO_NAME -corner_height 120
19
20 # Crea las celdas de esquina (corners)
21 create_cell {CORNER1 CORNER2 CORNER3 CORNER4} PCORNER
22
23 # Pads para  VDD y VSS
24 create_cell {PVDD1} PVDD1CDG
25 create_cell {PVSS1} PVSS1CDG
```

Nota. En este cuadro se muestra el proceso para la creación del anillo de entradas y salidas del circuito así como también se crean las celdas que se utilizarán para los *pads* de alimentación.

Una vez que se crearon las celdas que servirán como *pads* de alimentación, se procede a configurar las redes de poder del diseño y a agregar los *pads* al anillo de entradas y salidas. Esto se observa en el Cuadro 10.12, en donde primero se utiliza el comando `resolve_pg_nets` para definir las redes de poder y tierra. Posterior a esto, se crean las redes de VDD y VSS con los comandos `create_net -power VDD` y `create_net -ground VSS` respectivamente, para finalmente agregar los *pads* al anillo de entradas y salidas con el comando `add_to_io_guide` y fijarlos con el comando `place_io`.

**Cuadro 10.12.** Configuración de las redes de alimentación del circuito

```

# Resuelve las redes de power/ground
resolve_pg_nets

# Crea las nets de VDD y VSS
create_net -power VDD
create_net -ground VSS

set_ignored_layers -min_routing_layer M1 -max_routing_layer M9

# Agrega los pads al anillo IO
add_to_io_guide [get_io_guides $ANILLO_NAME.left] PVDD*
add_to_io_guide [get_io_guides $ANILLO_NAME.right] PVSS*
place_io
```

Nota. Los comandos del cuadro permiten configurar las redes de alimentación del diseño y agregar los *pads* al anillo de entradas y salidas.

A pesar de que con los comandos que se han mostrado hasta el momento ya se realizó la creación de los *pads* y se colocaron en el anillo de entradas y salidas, aún no se ha realizado la conexión ni la creación del anillo de alimentación. En el Cuadro 10.13 se observa la creación del anillo de alimentación, en donde primero se define el patrón del anillo con el comando `create_pg_ring_pattern`, en el cual se definen las capas y dimensiones del anillo, así como también las capas donde se ubicará. Posteriormente, el comando `set_pg_strategy` permite definir la estrategia del anillo en el *core* del diseño, donde se indica el patrón creado previamente y las redes que se utilizarán (VDD y VSS). Finalmente, se compila el anillo de alimentación con el comando `compile_pg`.

**Cuadro 10.13.** Creación del anillo de alimentación

```

# Define el estilo del anillo PG (Power/Ground)
create_pg_ring_pattern ring_pattern \
 -horizontal_layer M2    -horizontal_width {2} -horizontal_spacing {2}
    -vertical_layer M3 -vertical_width {2} -vertical_spacing {2}

# Define la estrategia del anillo en el core
set_pg_strategy core_ring    -pattern {{name: ring_pattern}    {nets: {VDD VSS}} {
    offset: {1 1}}} -core

# Compila el anillo de PG
compile_pg -strategies core_ring
```

Nota. El cuadro contiene los comandos que permiten la creación del anillo de alimentación del diseño que se encuentra alrededor del *core* del diseño.

Posterior a la creación del anillo, ya se puede establecer la conexión entre este y los

*pads* de alimentación. En el Cuadro 10.14 se observa que primero se define el patrón de conexión entre los *pads* y el anillo con el comando `create_pg_macro_conn_pattern`, en el que se indica el tipo de conexión (en este caso, el estilo *scattered*), las capas y las redes que se utilizarán. Luego, se configura la herramienta para tratar los *pads* como macros con el comando `set_app_options`. Una vez definido esto, se configura la estrategia de conexión entre los *pads* y el anillo con el comando `set_pg_strategy`, en donde se indican las celdas que se utilizarán (PVDD y PVSS) y el patrón creado previamente.

En esta sección también se define la regla de vías que se utilizará para las conexiones con el comando `set_pg_strategy_via_rule`, en donde se indica que se utilizarán vías en la capa M3 para las conexiones existentes. Posterior a esto, se compila la estrategia de conexión con el comando `compile_pg`, y finalmente se conectan los pines de VDD y VSS a las redes correspondientes con el comando `connect_pg_net`. Al final de esta sección, se colocó el comando de generación de reportes para validar que el procedimiento se ejecutó de manera adecuada.

**Cuadro 10.14.** Conexión de celdas con el anillo de alimentación

```
1
2
3  # Define la forma de conectar los pads y el anillo (tipo scattered)
4  create_pg_macro_conn_pattern hm_pattern \
5      -pin_conn_type scattered_pin \
6      -layers {M2 M3} \
7      -nets {VDD VSS} \
8      -pin_layers {M2}
9
10 # Se configura para tratar pads como macros
11 set_app_options -name plan.pgroute.treat_pad_as_macro -value true
12
13 # Estrategia para conectar los pads al anillo
14 set_pg_strategy macro_conn \
15     -macros [get_cells {PVDD* PVSS*}] \
16     -pattern {{name: hm_pattern} {nets: {VDD VSS}}}
17
18 # Define regla de vias
19 set_pg_strategy_via_rule macro_conn_via_rule \
20     -via_rule { {{{strategies: macro_conn}} {{existing: all} {layers: M3}} {
       via_master: default}} {{intersection: undefined} {via_master: NIL}} }
21
22 # Realiza el proceso de conectar pads con el anillo
23 compile_pg -strategies macro_conn -via_rule macro_conn_via_rule -tag test
24
25 # Conecta los pines de VDD/VSS a las redes
26 connect_pg_net -net VDD [get_pins -physical_context */VDD*]
27 connect_pg_net -net VSS [get_pins -physical_context */VSS*]
28 connect_pg_net -automatic
29
30 # Reporte de celdas de poder
31 report_cells -power
```

Nota. En esta sección del *script* se configuran los parámetros y estrategias para las conexiones entre los *pads* de alimentación y el anillo de poder.

Además de definir el anillo de alimentación que rodea el *core* del diseño, se debe crear la malla de alimentación que servirá para conectar las celdas estándar. Para esto se puede observar el Cuadro 10.15, en donde se define el estilo de la malla con el comando `create_pg_mesh_pattern`, se define la estrategia de la malla, la estrategia de conexión de las celdas estándar y finalmente, se compilan las estrategias para crear el patrón. El último comando que se observa en esta sección es el que permite realizar la conexión entre el anillo de alimentación que rodea el *core* con la malla que fue creada para la alimentación de las celdas estándar.

**Cuadro 10.15.** Creación de la malla de alimentación en el *core*

```

# Define el estilo de malla PG (mesh)
create_pg_mesh_pattern mesh_pattern -layers {{{ vertical_layer: M3} {width: 4.2}{
    pitch: 42} {spacing: interleaving }}}

# Estrategia de malla dentro del core
set_pg_strategy mesh_strategy \
 -polygon "{190.000 190.000} {$adjusted_polygon_x $adjusted_polygon_y}" -pattern
    {{ pattern: mesh_pattern }{ nets: {VDD VSS }}} -blockage {macros: all}

# Estrategia para celdas del core
create_pg_std_cell_conn_pattern std_cell_pattern
set_pg_strategy std_cell_strategy -core -pattern {{pattern: std_cell_pattern} {
    nets: {VDD VSS}}}

# Compila las estrategias de malla
compile_pg

# Une la malla con el anillo PG
merge_pg_mesh -nets {VDD VSS} -types {ring stripe} -layers {M2 M3}
```

Nota. Este cuadro muestra los comandos utilizados para la creación de la malla de alimentación que se encuentra dentro del *core* y que brinda alimentación para las celdas estándar del diseño.

Finalmente, la última etapa del *floorplan* corresponde a la verificación y ajuste de la colocación final de las celdas. En el Cuadro 10.16 se observa que primero se realizan verificaciones de consistencia antes de finalizar la colocación con el comando `check_design -checks pre_placement_stage` y `check_design -checks physical_constraints`. Posteriormente, se utiliza el comando `legalize_placement` para legalizar la colocación final de las celdas en el diseño. Una vez ejecutados estos comandos, se da por finalizada esta etapa del flujo de diseño por lo cual se retorna al *script* principal de síntesis física, donde da continuidad con el enrutamiento y las verificaciones físicas.

**Cuadro 10.16.** Verificación y ajuste de la colocación final

```

# Revisa consistencia antes del placement
check_design -checks pre_placement_stage
check_design -checks physical_constraints

# Legaliza el placement final
legalize_placement
```

Nota. Este cuadro muestra la sección final del *script* del *floorplan.tcl* en donde se realizan las verificaciones y se legaliza la colocación final de las celdas.

### 10.2.3. Cambios implementados en el *script* de `sintesis_fisica_top.tcl`

Este archivo contiene la estructura principal del flujo de la síntesis física. Este *script* no sufró tantos cambios pero sí se eliminaron comandos para trasladarlos a los otros dos *scripts*. En este documento se encuentran los comandos que ejecutan los *scripts* de *setup* y *floorplan*, y luego se describen las etapas de síntesis del árbol de reloj, enrutamiento, creación de rellenos y verificaciones físicas. Es posible realizar modificaciones para que las últimas etapas se puedan separar en distintos *scripts*.

Los primeros comandos que están en el Cuadro 10.17 se utilizan para cargar la configuración y el *floorplan*, y luego eliminan las estrategias PG para limpiar nuevamente el entorno.

**Cuadro 10.17.** Configuración inicial del *script* de síntesis física

```
1
2 # Carga el script de setup
3 source scripts/setup.tcl
4
5 # Carga el script de floorplan
6 source scripts/floorplan.tcl -echo
7
8 # Elimina cualquier estrategia de PG previa para limpiar el entorno
9 remove_pg_strategies -all
```

Nota. En este cuadro se encuentran los comandos para generar la configuración inicial del entorno de ICC2 y para cargar el *floorplan* del diseño.

La siguiente sección del documento se muestra en el Cuadro 10.18, el cual contiene los comandos que ejecutan la síntesis del árbol de reloj del circuito. La estructura original ya contenía la opción de habilitar o deshabilitar esta etapa del flujo mediante la variable `CLOCK_SYNTHESIS`, por lo cual no se realizaron modificaciones significativas en esta sección. En el cuadro se observa que se genera un reloj con un período de 0.5 ns y se genera a partir de los *nets* con nombre “clk” disponibles en el diseño.

**Cuadro 10.18.** Síntesis del árbol de reloj

```

if { $CLOCK_SYNTHESIS == "TRUE" } {
    puts "Sintetizando relojes..."
    create_clock -period 0.5 -name clk [get_nets -design [current_block] {clk}]
    check_clock_trees -clocks clk
    check_design -checks pre_clock_tree_stage
    synthesize_clock_trees -clocks clk -postroute -routed_clock_stage
    detail_with_signal_routes
    clock_opt -list_only
    check_design -checks cts_qor
} else {
    puts "Clock synthesis deshabilitada."
}
```

Nota. Este cuadro contiene la sección del *script* principal en donde se realiza la síntesis del árbol de reloj si la variable `CLOCK_SYNTHESIS` está habilitada.

Antes de ejecutar la etapa de enrutamiento, es importante realizar verificaciones previas que se muestran en el Cuadro 10.19. Inicialmente, se verifica la ruteabilidad del diseño con el comando `check_routability`, incluyendo los puertos bloqueados por el anillo de alimentación. Por otro lado, se realizan verificaciones generales del diseño con el comando `check_design -checks pre_route_stage`.

Se incluyó un comando que permite cargar un archivo TCL de reglas de antena en caso de que se cuente con uno, para así realizar las verificaciones correspondientes durante el enrutamiento y generar un reporte que pueda ser consultado para corregir posibles violaciones. En el caso del proyecto, no se cuenta con un archivo que pueda ser ejecutado con ICC2 dado que en el repositorio solo se encontró el que corresponde a la herramienta de IC Compiler, por lo cual, este comando se encuentra comentado.

**Cuadro 10.19.** Verificaciones previas al enrutamiento

```

# Verifica la ruteabilidad, incluyendo puertos bloqueados por PG
check_routability -check_pg_blocked_ports true

# Ajusta el nivel de mensajes "verbose" durante el enrutamiento
set_app_options -name route.common.verbose_level -value 1
check_design -checks pre_route_stage
set_app_options -name route.common.verbose_level -value 0

# Archivo opcional de reglas de antena
# source -echo <ruta hacia tcl de antena>
# report_app_options route.detail.*antenna*
```

Nota. En estos comandos se realizan verificaciones previas al enrutamiento del diseño para asegurar la viabilidad y consistencia.

La próxima etapa del flujo ya ejecuta el enrutamiento. En el Cuadro 10.20 se observa que primero se ejecuta el enrutamiento global con el comando `route_auto`, luego se verifican las rutas generadas con el comando `check_routes`. Además, otra parte importante es agregar las vías redundantes con el comando `add_redundant_vias`, dado que existen reglas del DRC que implican la inserción de vías; y finalmente, se ejecuta el enrutamiento detallado incremental con DRC inicial utilizando el comando `route_detail`.

**Cuadro 10.20.** Comandos para ejecución de enrutamiento

```

# Enrutamiento global
route_auto

# Verifica rutas generadas
check_routes

# Agrega vias redundantes para mejorar confiabilidad
add_redundant_vias -effort high

# Enrutamiento detallado incremental con DRC inicial
route_detail -incremental true -initial_drc_from_input true
```

Nota. Los comandos del cuadro contienen la sección del *script* principal en donde se realiza el enrutamiento del diseño.

Antes de ejecutar la etapa de creación de rellenos, se verifica la legalidad del diseño. Este comando permite validar si el proceso se realizó de forma adecuada, de manera que, en el caso de que exista un error crítico, el diseño se detendrá en esta etapa.

En el Cuadro 10.21 se observa que, luego de validar la legalidad, se crean los rellenos en el anillo de entradas y salidas con el comando `create_io_filler_cells`, donde se indican las guías de entrada y salida correspondientes al anillo creado previamente, y también se indican los nombres de las celdas de relleno que se utilizarán según las librerías disponibles. Para poder observar los nombres de las celdas, se pueden utilizar los comandos que se encuentran comentados en la parte superior.

Además de crear los rellenos en el anillo de entradas y salidas, también se crean los rellenos en el *core* del diseño con el comando `create_stdcell_fillers`. En este caso, se utilizaron celdas de relleno de la librería *FillersWorkspace*, pero también se dejó un ejemplo alternativo comentado para utilizar celdas de relleno de la librería *StandardWorkspace*. En este caso, se seleccionan las celdas de distintos tamaños para que la herramienta tenga disponibles varias opciones al momento de insertar los rellenos en las distintas secciones del diseño. Es recomendable revisar los nombres de las celdas en los archivos LEF para comprobar que los nombres se seleccionen de manera adecuada.

Luego de colocar los rellenos, realiza nuevamente la conexión de los nodos de poder y tierra, y elimina los rellenos que se colocaron en ubicaciones que generan violaciones del diseño. Finalmente, hace una última verificación de legalidad para proseguir con la siguiente parte del flujo de diseño que ya corresponde a las verificaciones físicas.

**Cuadro 10.21.** Creación de rellenos en el diseño

```
1
2
3  # Verifica la legalidad antes de insertar fillers
4  check_legality
5
6  # Los comandos siguientes son ejemplos para obtener fillers disponibles:
7  # get_lib_cells LIB_TEST/* -filter "design_type==core"
8  # get_lib_cells FillersWorkspace/* -filter "design_type==filler"
9  # get_lib_cells TSMCWorkspace|StandardWorkspace/* -filter "design_type==filler"
10
11 # Crea fillers en el IO ring
12 create_io_filler_cells \
13   -io_guides [get_io_guides {ANILLO_IO.top ANILLO_IO.right ANILLO_IO.left
       ANILLO_IO.bottom}] \
14   -reference_cells {PFILLER0005 PFILLER05 PFILLER1 PFILLER5 PFILLER10 PFILLER20}
15
16 # Crea fillers en el core
17 # Fillers standard o core
18 create_stdcell_fillers -lib_cells [get_lib_cells {TSMCWorkspace|FillersWorkspace/
       FILL1 TSMCWorkspace|FillersWorkspace/FILL16 TSMCWorkspace|FillersWorkspace/
       FILL2 TSMCWorkspace|FillersWorkspace/FILL32 TSMCWorkspace|FillersWorkspace/
       FILL4 TSMCWorkspace|FillersWorkspace/FILL64 TSMCWorkspace|FillersWorkspace/
       FILL8}]
19
20 # Ejemplo alternativo (comentado):
21 # create_stdcell_fillers -lib_cells [get_lib_cells {TSMCWorkspace|
       StandardWorkspace/GFILL TSMCWorkspace|StandardWorkspace/GFILL2 TSMCWorkspace|
       StandardWorkspace/GFILL3 }]
22
23 # Conecta los fillers a las redes de power/ground
24 connect_pg_net -automatic
25
26 # Elimina fillers con violaciones
27 remove_stdcell_fillers_with_violation
28
29 # Revision final de legalidad tras insertar rellenos
30 check_legality
```

Nota. Este segmento del *script* principal muestra los comandos para la creación de las celdas de rellenos a partir de la librería seleccionada,así como su conexión con la red de alimentación.

### 10.2.4. Ejecución de las pruebas DRC, antena e implementación de rellenos de metal

Esta sección ya no presentó más cambios además de los mencionados en la sección 9.1.4. Según el Cuadro 10.22 primero se guarda el bloque y luego se inicia la inserción de rellenos BEOL. Se vuelve a guardar el bloque y se pasa a la inserción de rellenos FEOL.

Finalmente, se ejecutan las verificaciones DRC y LVS con los comandos `signoff_check_drc` y `check_lvs`, respectivamente. Estos comandos permiten la ejecución de las pruebas disponibles en la herramienta IC Validator, pero están integrados en IC Compiler II de modo que se facilita su uso [23]. También se emplea el comando `signoff_fix_drc` para corregir automáticamente las violaciones encontradas en el DRC, de modo que el comando vuelve a ejecutar la verificación DRC y despliega los resultados de las correcciones que efectuó, e indica si aún existen violaciones. A estos comandos no se les agregaron parámetros adicionales, pero sí se deben configurar las opciones de la aplicación para modificar la forma en que se van a ejecutar. En ocasiones, puede ser útil indicarle a la herramienta la cantidad máxima de repeticiones que se deben realizar, dado que puede tardar un tiempo considerable en hacer las correcciones.

En la parte final, se añadió un comando que permite tener mayor control sobre los mensajes del tipo *warning*, dado que se observó que la consola muestra la advertencia de que algunos nombres son demasiado extensos. Esto no es un error crítico para la aplicación, pero dada la gran cantidad de celdas que contiene el diseño esto genera incomodidad al revisar los resultados de las verificaciones físicas. Para observar más detalles sobre el comando, se puede consultar la documentación de IC Compiler II en el apartado de "mensajes de error"[23].

**Cuadro 10.22.** Comandos para la inserción de rellenos y verificaciones físicas

```
1
2
3  # Guarda el bloque antes de iniciar DRC
4  save_block $ndm_design_library:$CIRCUIT_NAME
5
6  # Insercion de metal fill (Back-End Of Line)
7  signoff_create_metal_fill -mode add -foundry_fill_type beol
8
9  # Guarda nuevamente tras la insercion BEOL
10 save_block $ndm_design_library:$CIRCUIT_NAME
11
12 # Insercion de metal fill (Front-End Of Line)
13 signoff_create_metal_fill -mode add -foundry_fill_type feol -foundry_for_feol_fill
       generic
14
15 # Corre verificacion DRC y correcciones automaticas
16 signoff_check_drc
17 signoff_fix_drc
18
19 # Guarda bloque actualizado tras DRC
20 save_block $ndm_design_library:$CIRCUIT_NAME
21
22 # Verifica LVS (Layout vs. Schematic)
23 check_lvs
24
25 # signoff_check_drc  ;# Ejecucion adicional de DRC si se requiere
26
27 # Desactiva mensajes especificos de GDS
28 set_msg GDS-028 -level off
```

Nota. Este cuadro presenta el proceso de insersión de rellenos FEOL y BEOL así como también la ejecución de las verificaciones físicas DRC y LVS.

Para finalizar el proceso de síntesis física, se deben exportar los resultados en un archivo con extensión GDS para que puedan ser utilizados en etapas posteriores del flujo de diseño. En el Cuadro 10.23 se observa que primero se genera un *netlist* que ya contiene las celdas de relleno y todos los componentes que se agregaron para el diseño físico, y luego se escribe el archivo GDS con el comando `write_gds`, que ya contiene información física sobre el *layout* del diseño, así como también los *frames* de las celdas.

Al final, se colocó un mensaje que ayuda a distinguir en la terminal cuando terminó el proceso, y se colocó el comando que abre la interfaz gráfica de la aplicación. Este comando no es necesario, pero se colocó para facilitar la visualización de los resultados; aunque, dependiendo de la necesidad, puede comentarse.

**Cuadro 10.23.** Exportación de archivos resultantes del diseño físico

```
1
2  # Genera el netlist Verilog del circuito
3  write_verilog -include all ./Outputs/IO/$CIRCUIT_NAME.v
4
5  # Exporta a formato GDS
6  write_gds \
7    -library $ndm_design_library \
8    -design $CIRCUIT_NAME \
9    -view design \
10   -hierarchy all \
11   -lib_cell_view frame \
12   ./Outputs/$CIRCUIT_NAME.gds
13
14 ##############################################################################
15 # ETAPA 7: FINALIZACION
16 ##############################################################################
17
18 echo "Finalizo la sintesis fisica!"
19 start_gui
```

Nota. El cuadro contiene las últimas fases del *script* principal en donde se exportan los archivos resultantes del diseño físico en formatos Verilog y GDS y se muestra un comando que finaliza la síntesis física.

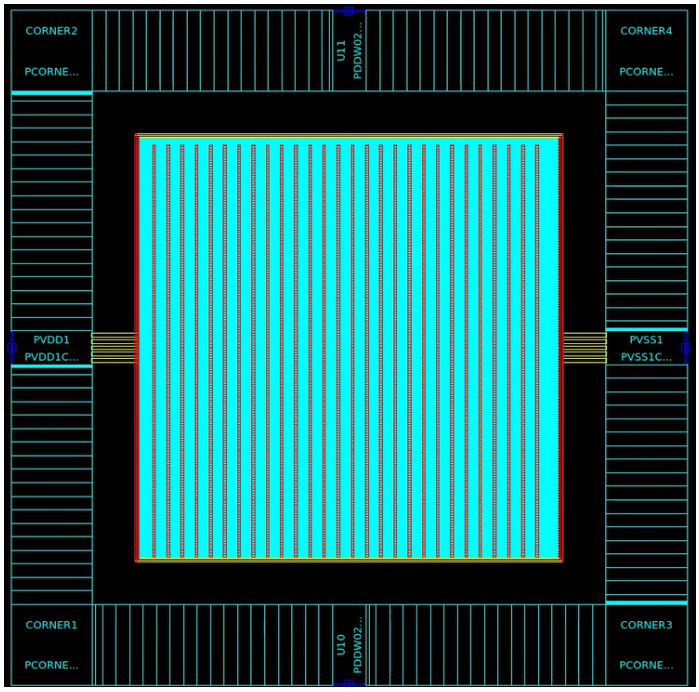
De esta manera se culmina con la explicación de los *scripts* y los cambios efectuados a partir de recibir los cursos de IC Compiler II. Aún no se obtienen resultados libres de errores, por lo cual en secciones posteriores se explican otros aspectos que deben ser tomados en cuenta para mejorar los resultados obtenidos.

## 10.3. Resultado final de las pruebas con la compuerta NOT

### 10.3.1. Resultados del *layout* de la compuerta NOT

Una vez realizados los cambios se obtuvieron los resultados que se muestran a continuación para cada uno de los casos. En la Figura 30 se observa el resultado del *layout* final que se obtuvo al realizar todos los cambios necesarios.

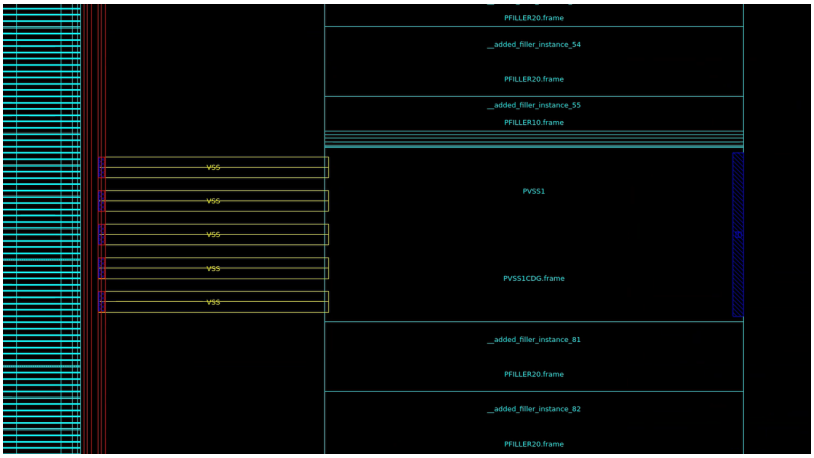
**Figura 30.** Resultado final del *layout* de la compuerta NOT




Nota. La figura muestra la vista completa del *layout* de la compuerta NOT luego de ejecutar el flujo de síntesis física con los cambios implementados.

En la Figura 31 se puede distinguir con mayor detalle el resultado de la síntesis. Cabe resaltar que para este punto se logró corregir la necesidad de dibujar manualmente las rutas que conectan el *pad* de alimentación con el anillo de poder que rodea el *core* del circuito, pero aún no se ha resulto el inconveniente de que se conecten con vías.

**Figura 31.** Detalle del resultado final del *layout* de la compuerta NOT



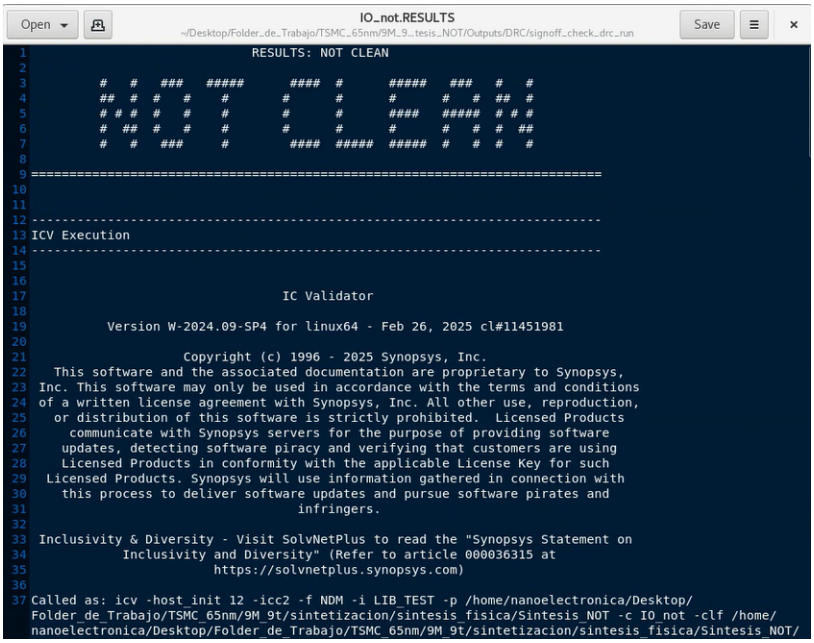

Nota. En esta imagen se observa un detalle del *layout* de la compuerta NOT en donde se observa un pin de alimentación y la conexión del pad con el anillo de poder.

### 10.3.2. Resultados de las pruebas DRC de la compuerta NOT

A pesar de que se efectuaron cambios en los *scripts* aún no se logró obtener resultados libres de errores. Es posible acceder a las carpetas con los resultados de las evaluaciones DRC accediendo a la ruta: `TSMC_65nm/9m_9t/sintetizacion/sintesis_fisica/Sintesis_NOT/Outputs/DRC/signoff_check_drc_run/IO_NOT.RESULTS`.

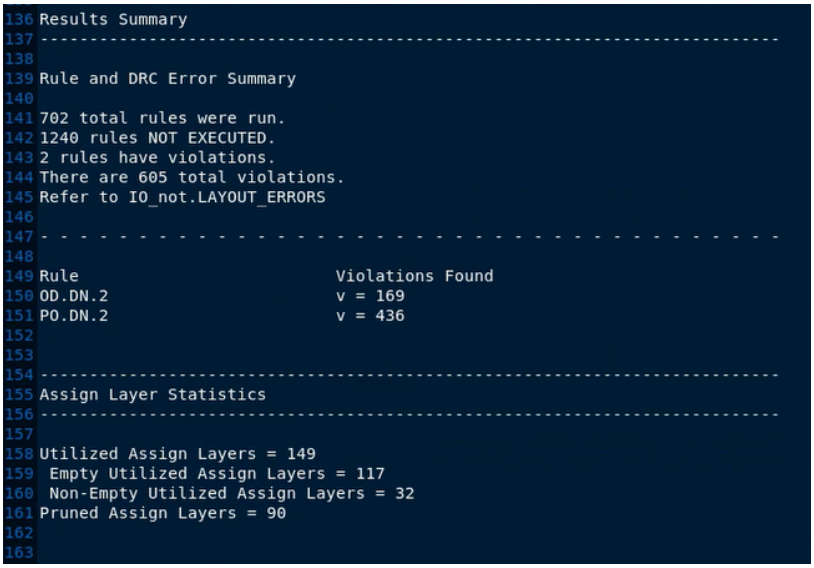
En la Figura 32 se encuentra el resultado general de la prueba DRC en donde se observa que no está libre de violaciones. En la siguiente figura (33) se observa el detalle de los resultados obtenidos, en donde se indica que existen 169 violaciones de la regla `OD.DN.2` y 436 en la regla `PO.DN.2`, las cuales corresponden a reglas de densidad relacionadas con los rellenos OD/PO. Al abrir el archivo con nombre `IO_not.LAYOUT_ERRORS` se puede observar el detalle en donde se indica lo mostrado en el Cuadro 10.24. Cabe resaltar que al iniciar las pruebas con los *scripts* originales se había iniciado con una cantidad total de 34894 errores y la cantidad se redujo en esta última prueba hasta 605 violaciones, lo cual implica una reducción del 98.26 % en las pruebas DRC.

**Figura 32.** Resultado del reporte de DRC tras la síntesis física de la compuerta NOT




Nota. La figura muestra el resultado del reporte de la verificación DRC el cual aún contiene errores.

**Figura 33.** Detalle del resultado del reporte de DRC de la compuerta NOT




Nota. Esta imagen muestra las violaciones que existieron en las pruebas DRC de la compuerta NOT

**Cuadro 10.24.** Detalle de errores DRC en la compuerta NOT final

```
1
2            ERROR SUMMARY
3
4  OD.DN.2 : Min. OD density over window 150 step 75
5  >= 20%
6    density ..................................... 169 violations found.
7
8  PO.DN.2 : {OD OR DOD OR PO OR DPO } local density
9  (minimun) over window 20um x 20um stepping 10um >=
10 0.1%
11   density ..................................... 436 violations found.
```

Nota. En el cuadro se observa el resultado que describe las violaciones DRC que se encontraron en la compuerta NOT final.

### 10.3.3. Resultados de verificación de antena de la compuerta NOT

Para ejecutar la verificación de antena se preparó otro breve *script* que se encuentra en la siguiente ruta:

```
1 /home/nanoelectronica/Desktop/Folder\_de\_Trabajo/TSMC\_65nm/9M\_9t/Verificaciones
      /Antena/IO\_not/antenna.tcl
```

Para ejecutarlo se debe acceder a la ubicación del archivo y abrir la consola. Dado que este archivo TCL ya contiene los comandos necesarios no es necesario abrir la aplicación de ICC2 ni IC Validator. En el Cuadro 10.25 se observa el contenido del *script* que permite realizar la verificación de antena del diseño de la compuerta NOT.

**Cuadro 10.25.** *Script* para verificación de antena

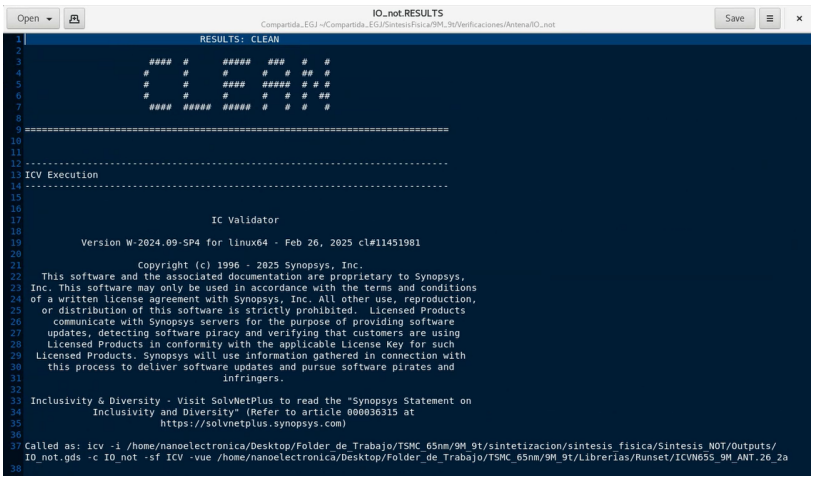
```
1
2 file delete *rdb
3 file delete *rep
4 file delete .remote*
5
6 icv -i /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t/
      sintetizacion/sintesis_fisica/Sintesis_NOT/Outputs/IO_not.gds -c IO_not -sf
      ICV -vue /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t/
      Librerias/Runset/ICVN65S_9M_ANT.26_2a
```

Nota. El *script* permite realizar la verificación de antena del diseño de la compuerta NOT.

Este mismo *script* se utilizó para las verificaciones del resto de circuitos, dentro de la carpeta llamada `Verificaciones/Antena/IO_<nombre del circuito>`.

Para el caso de la compuerta NOT se obtuvo el resultado de la Figura 34, en donde se observa el resultado "*Clean*"que indica que no se encontraron violaciones de las reglas de antena en el diseño final.

**Figura 34.** Resultado de la verificación de antena de la compuerta NOT




Nota. Esta imagen muestra el resultado libre de errores de la verificación de antena de la compuerta NOT.

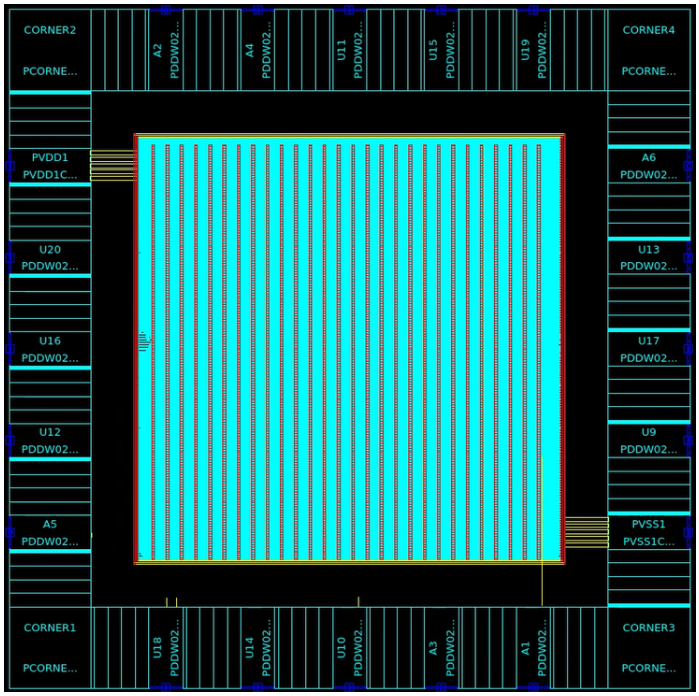
## 10.4. Resultados de las pruebas de síntesis física con un circuito de *ALU* (unidad aritmético lógica)

Este circuito no se sintetizó previo a ejecutar cambios en los *scripts*, por lo cual a continuación solo se muestran los resultados posteriores a la implementación de las mejoras.

### 10.4.1. Resultados del *layout* del circuito ALU

El circuito de la ALU es considerablemente más grande que la compuerta NOT, por lo cual en la Figura 35 se observa el resultado general del *layout* final. A simple vista se pueden observar diferencias en la cantidad de celdas que se encuentran en el anillo de entradas y salidas y la ubicación de los *pads* de alimentación. En esta imagen en la parte izquierda del *core* se distingue una pequeña sección difusa, la cual se puede ampliar dado que es en esta sección donde se encuentran las celdas estándar del diseño.

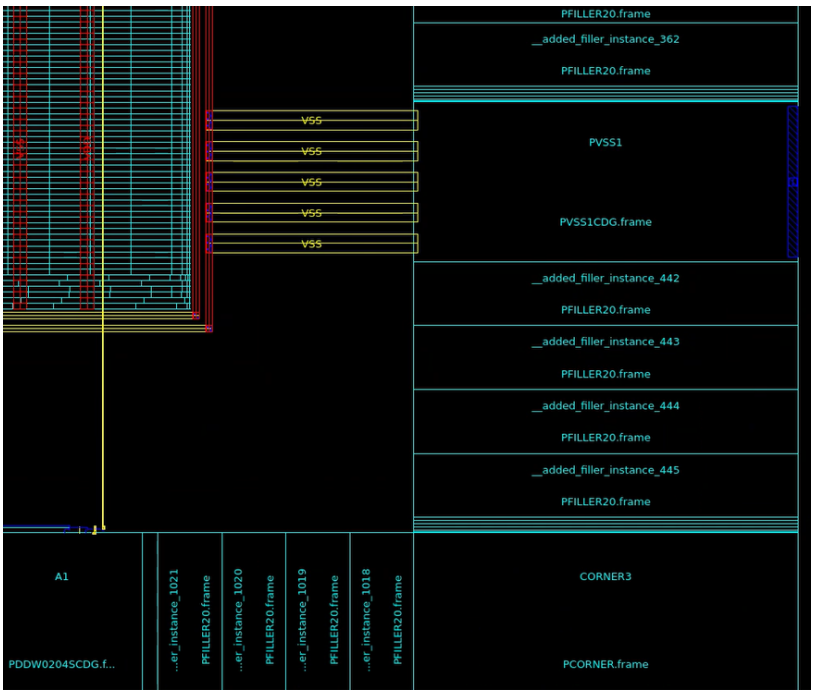
**Figura 35.** Resultado final del *layout* del circuito ALU




Nota. Esta figura contiene el resultado final del *layout* del circuito ALU luego de ejecutar el flujo de síntesis física con los cambios implementados.

La Figura 36 contiene el detalle de la sección que corresponde a la conexión del *pad* de alimentación con el anillo de poder. Del mismo modo que en la copuerta NOT se genera la ruta pero no se conecta al *pad*. En esta imagen se puede observar más detalle sobre celdas del anillo de entradas y salidas que se conectan a segmentos que se dirigen hacia el *core* del diseño, así como también en las orillas se distinguen los pines que permiten la conexión externa con las celdas del anillo de entradas y salidas.

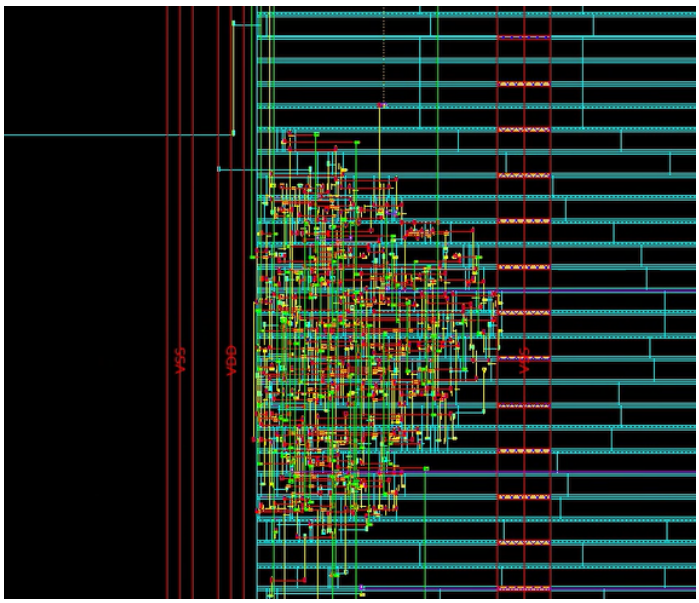
**Figura 36.** Detalle de pines y conexión de alimentación del circuito ALU




Nota. En la imagen se distingue un detalle del *layout* del circuito ALU en donde se observa un pin de alimentación y la conexión del pad con el anillo de poder.

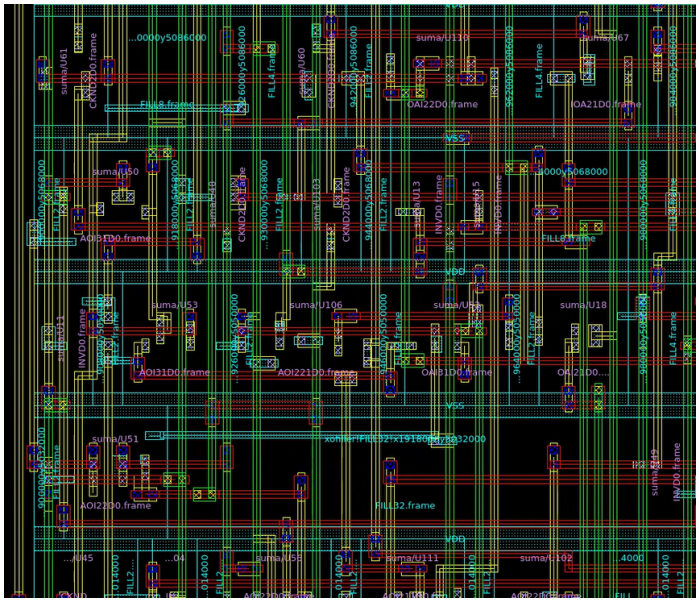
Por otro lado, la Figura 37 la ampliación de la sección mencionada previamente que contiene las celdas estándar del diseño. En esta imagen contiene la parte más importante del *core* del diseño, ya que en esta sección se encuentran las celdas que conforman el circuito de la ALU. Al igual que la compuerta NOT el resto del circuito solamente corresponde a celdas de relleno, lo cual se debe principalmente a que el tamaño del circuito es muy pequeño en comparación con el espacio disponible. Además, la Figura 38 contiene un detalle maximizado en donde se puede ver la complejidad de la colocación de las celdas estándar y las conexiones entre ellas.

**Figura 37.** Detalle de conexiones del circuito ALU




Nota. La imagen contiene un detalle de la vista ampliada del *layout* del circuito ALU en donde se observan las conexiones entre celdas estándar.

**Figura 38.** Detalle de celdas y conexiones del circuito ALU

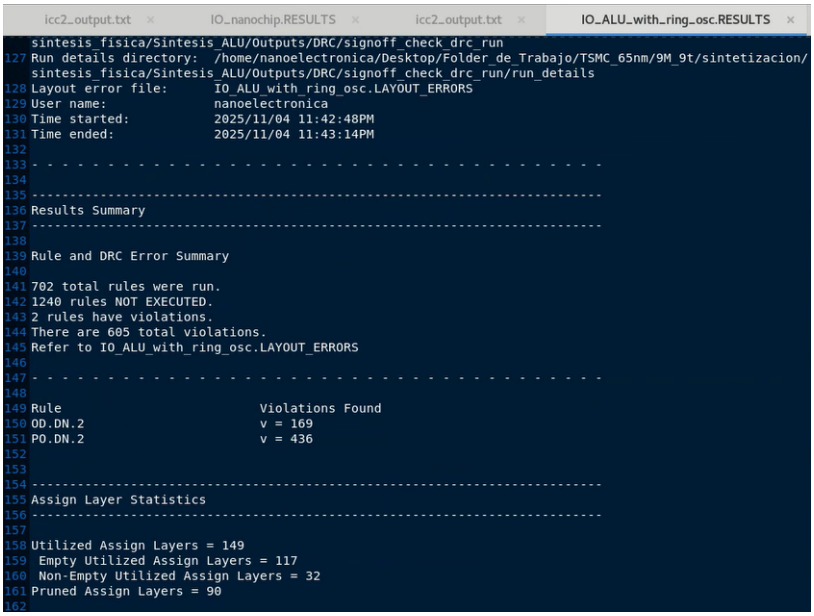



Nota. Esta figura contiene un detalle ampliado donde se observan las celdas estándar, rellenos, los nombres de las vistas de los *frames* y conexiones en el circuito de la ALU.

### 10.4.2. Resultados de las pruebas DRC del circuito ALU

Del mismo modo que en la compuerta NOT, el circuito de la ALU aún no se encuentra libre de errores. En la Figura 39 se observa el resultado general de la verificación DRC del diseño, en donde se observa que al igual que en la compuerta NOT existen dos errores relacionados con la densidad de celdas.

**Figura 39.** Resultado de pruebas DRC del circuito ALU




Nota. La imagen contiene una captura de pantalla del resultado de realizar la verificación DRC del circuito ALU luego de ejecutar el flujo de síntesis física.

Al abrir el archivo con nombre `IO_ALU.LAYOUT_ERRORS` se puede observar el detalle de los errores que se muestra en el Cuadro 10.26, en donde se indican las mismas violaciones que en la compuerta NOT.

**Cuadro 10.26.** Detalle de errores DRC en el circuito de ALU

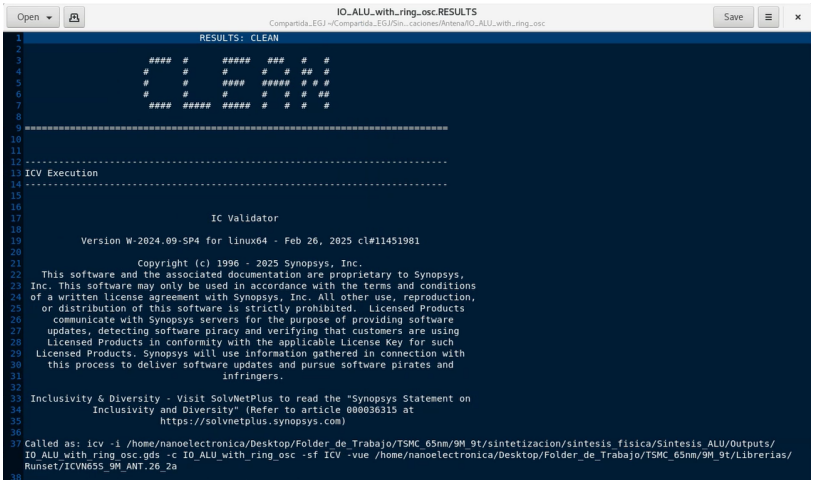
```
1
2              ERROR SUMMARY
3
4  OD.DN.2 : Min. OD density over window 150 step 75
5  >= 20%
6    density ..................................... 169 violations found.
7
8  PO.DN.2 : {OD OR DOD OR PO OR DPO } local density
9  (minimun) over window 20um x 20um stepping 10um >=
10 0.1%
11   density ..................................... 436 violations found.
```

Nota. En el cuadro se observa el resultado que describe las violaciones DRC que se encontraron en el circuito de ALU.

### 10.4.3. Resultados de verificación de antena del circuito ALU

Se realizó la verificación de antena del circuito de la ALU utilizando el mismo TCL que se describió en la sección 10.3.3, en el cual solamente se cambió la ruta de entrada del archivo GDS y el nombre del diseño. El resultado se muestra en la Figura 40, en donde se observa que no se encontraron violaciones de las reglas de antena en el diseño final.

**Figura 40.** Resultado de la verificación de antena de la ALU sintetizada




Nota. Esta imagen muestra el resultado libre de errores de la verificación de antena de la ALU sintetizada.

## 10.5. Resultados de las pruebas de síntesis física del circuito de "El Gran Jaguar"

### 10.5.1. Comparativa de los resultados previo a implementar cambios

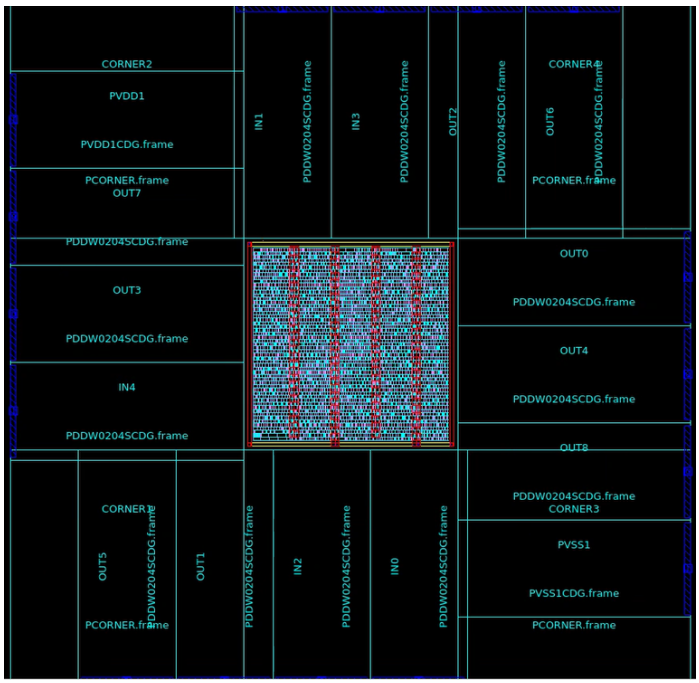
Para mostrar una mejor referencia sobre los cambios obtenidos en la ejecución de los *scripts*, a continuación se presentan los resultados obtenidos antes de implementar las mejoras.

El la imagen de la Figura 41 se observa el primer *layout* obtenido para el circuito de "El Gran Jaguar", donde se observa una gran área vacía ya que no se habían configurado las dimensiones correctas del *core* y el *offset* del anillo de entradas y salidas. Del mismo modo, se distingue que el área del circuito es bastante reducida y eso genera que las celdas estándar se encuentren muy juntas e incluso las celdas del anillo de entradas y salidas se encuentren sobrepuestas, como se observa en la Figura 42.

**Figura 41.** Resultado del *layout* del circuito "El Gran Jaguar" antes de implementar mejoras


Nota. La imagen contiene una vista general del *layout* del circuito El Gran Jaguar antes de implementar las mejoras en el flujo de síntesis física. en esta figura aún se observa el área libre debido a la incorrecta elección de las medidas del *core* y *offset*.

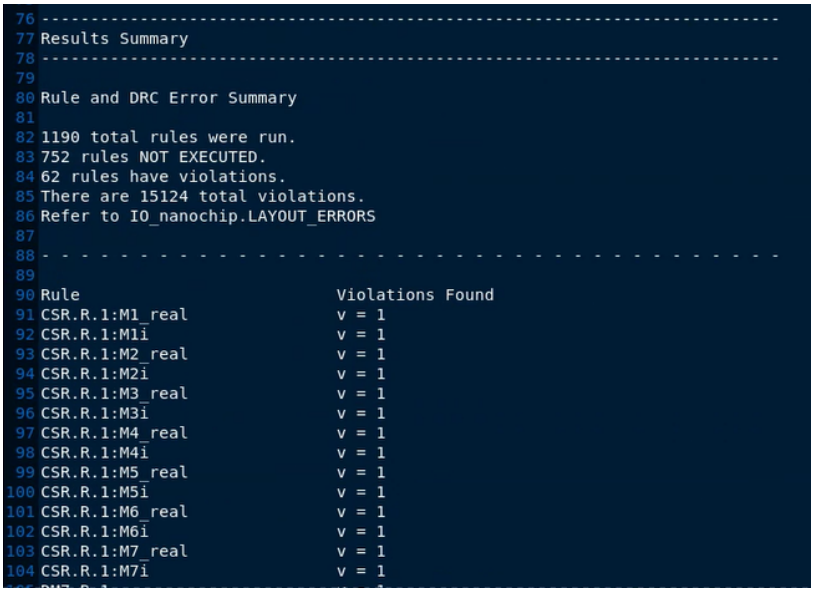
**Figura 42.** Vista ampliada del resultado del *layout* del circuito "El Gran Jaguar" antes de implementar mejoras




Nota. Esta figura contiene la vista ampliada del circuito de "El Gran Jaguar" antes de implementar mejoras donde se observan problemas con celdas sobrepuestas.

Además, en la Figura 43 se observa el resultado de las verificaciones físicas obtenidas antes de implementar las mejoras, donde se observa que existen un total de 15124 violaciones DRC para las cuales se muestra un listado de varias reglas que no cumplen con los parámetros establecidos en el *runset* utilizado.

**Figura 43.** Resultado de verificación DRC del circuito "El Gran Jaguar" antes de implementar mejoras

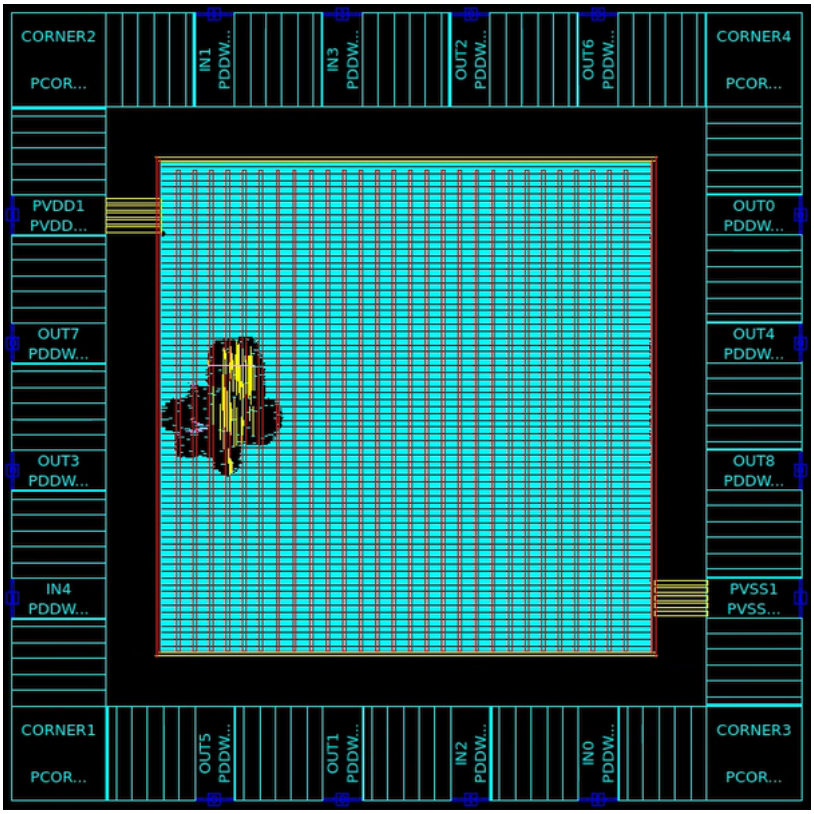



Nota. La imagen muestra una captura de pantalla del resultado de la verificación DRC del circuito El Gran Jaguar antes de implementar mejoras, en donde se observan múltiples violaciones.

### 10.5.2. Resultados finales del *layout* del circuito "El Gran Jaguar" después de implementar mejoras

Después de implementar las mejoras en los *scripts* se obtuvieron los resultados que se muestran a continuación para el circuito de "El Gran Jaguar". En la Figura 44 se observa el resultado del *layout*, donde ya es posible observar un circuito que cubre toda el área disponible para el diseño y donde también se puede notar que el área que ocupa el circuito principal es solamente un pequeño porcentaje del área total, lo cual implica que es posible utilizar un circuito más grande y más complejo en el mismo espacio.

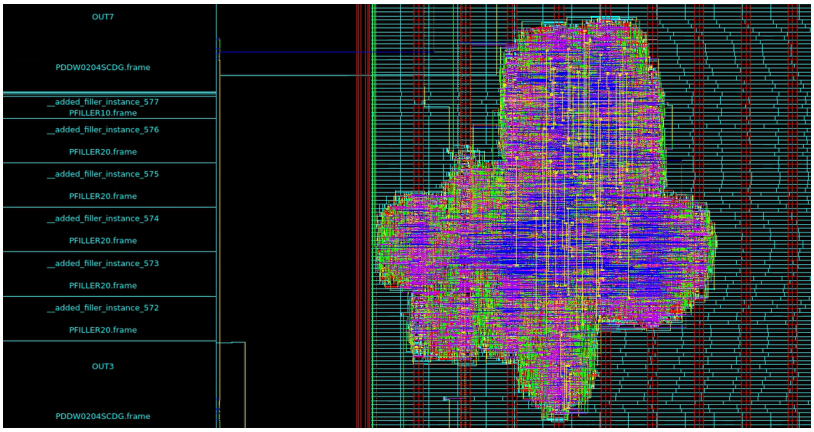
**Figura 44.** Resultado de verificación DRC del circuito "El Gran Jaguar" después de implementar mejoras




Nota. La imagen muestra el resultado final del *layout* del circuito "El Gran Jaguar".

En la Figura 45 se observa un acercamiento al circuito principal. Dada la cantidad de celdas y rutas no es posible distinguir con claridad cada uno de los componentes, pero sí es posible distinguir la complejidad del circuito y también la distribución de las celdas estándar en un solo punto, ya que el resto del *core* está ocupado por celdas de relleno.

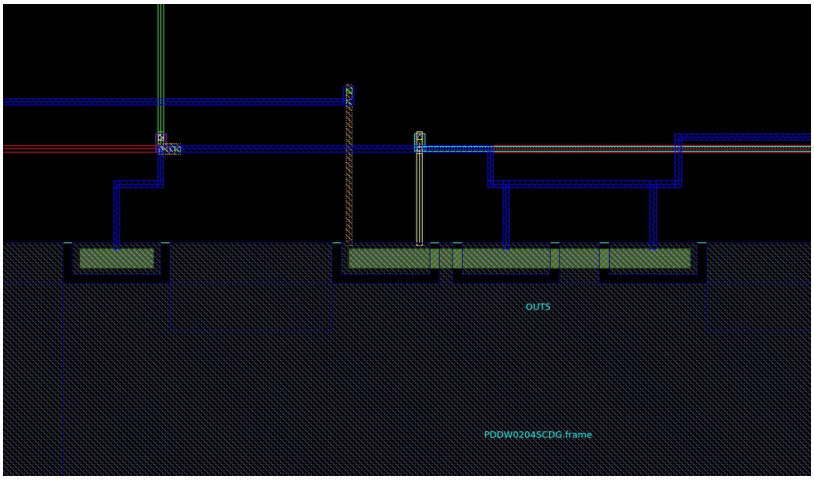
**Figura 45.** Detalle del *layout* del circuito "El Gran Jaguar" después de implementar mejoras




Nota. La figura contiene un acercamiento al *core* del diseño del circuito "El Gran Jaguar", en donde se observan las celdas estándar y las conexiones entre ellas.

Otro detalle importante que se puede observar en el *layout* del circuito es la interconexión de los *pads* con los metales que conectan con las celdas estándar del circuito. La Figura 46 muestra un acercamiento de esta conexión donde se configuró la vista para mostrar el *frame* que contiene mayor información sobre las conexiones.

**Figura 46.** Detalle de conexión de *pads* del circuito "El Gran Jaguar"




Nota. Esta imagen muestra cómo se realiza las conexiones de los *pads* con las celdas del *core* a través de metales y vías.

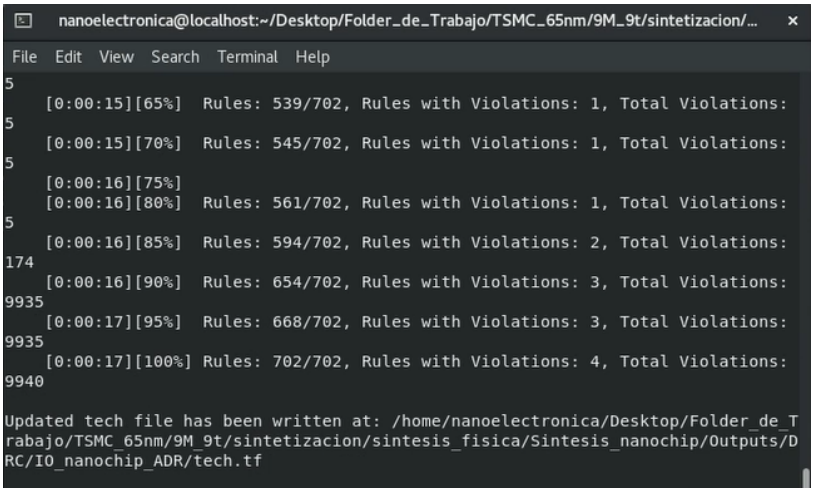
### 10.5.3. Resultados de las pruebas DRC del circuito "El Gran Jaguar"

Al realizar la prueba de las verificaciones DRC del circuito se obtuvo el resultado que se muestra en la Figura 47 en la terminal de ICC2. Se observó que la cantidad de errores mostrados sigue siendo elevada. Sin embargo, se pudo identificar un aspecto importante que aún requiere de investigación adicional, y es que este resultado corresponde a las violaciones DRC que se encuentran en el proceso de ADR (Automatic Design Rule Repair), resultado que se encuentra en la ruta de `./Outputs/DRC/IO_nanochip_ADR/IO_nanochip_ADR.RESULTS`.

Este resultado difiere al que se encuentra en la ruta `./Outputs/DRC/signoff_check_drc_run/IO_nanochip.RESULTS`, el cual contiene una cantidad menor de violaciones como se observa en el Cuadro 10.27.

Al revisar la documentación de ICC2, fue posible observar que los resultados mostrados en la carpeta ADR corresponden al comando ejecutado de `signoff_fix_drc`, de modo que según el manual de comandos cuando no se le indica la opción `"-select_rules"` el comando por defecto revisa todas las reglas, lo cual no genera los resultados esperados ya que el *runset* proporcionado por TSMC revisa reglas específicas según las secciones comentadas en el área de definiciones.

**Figura 47.** Resultado en la consola de la verificación DRC del circuito "El Gran Jaguar"




Nota. La captura de pantalla muestra la salida de la consola sobre las verificaciones DRC del circuito "El Gran Jaguar". Esto corresponde al proceso de ADR.

**Cuadro 10.27.** Detalle de errores DRC en el circuito de "El Gran Jaguar"

```
1
2
3              ERROR SUMMARY
4
5  M2.S.1 : Spacing >= 0.1 um
6    external1 ..................................... 5 violations found.
7
8  M2.S.5 : Space at Mx line-end (W<Q=0.120) in a
9  dense-line-end configuration: If Mx has parallel
10 run length with opposite Mx (measured with T=0.035
11 extension) along 2 adjacent edges of Mx [any one
12 edge <Q distance from the corner of the two edges],
13 then one of the space (S1 or S2) needs to be at least
14 this value (except for small jog with edge length
15 < 0.10 um (R)) >= 0.12 um.
16   external2 ..................................... 3 violations found.
17
18 M3.S.1 : Spacing >= 0.1 um
19   external1 ..................................... 1 violation found.
20
21 M3.S.5 : Space at Mx line-end (W<Q=0.120) in a
22 dense-line-end configuration: If Mx has parallel
23 run length with opposite Mx (measured with T=0.035
24 extension) along 2 adjacent edges of Mx [any one
25 edge <Q distance from the corner of the two edges],
26 then one of the space (S1 or S2) needs to be at least
27 this value (except for small jog with edge length
28 < 0.10 um (R)) >= 0.12 um.
29   external2 ..................................... 1 violation found.
30
31 OD.DN.2 : Min. OD density over window 150 step 75
32 >= 20%
33   density ....................................... 169 violations found.
34
35 PO.DN.2 : {OD OR DOD OR PO OR DPO } local density
36 (minimun) over window 20um x 20um stepping 10um >=
37 0.1%
38   density ....................................... 436 violations found.
```

Nota. En el cuadro se observa el resultado que describe las violaciones DRC que se encontraron en el circuito de "El Gran Jaguar".

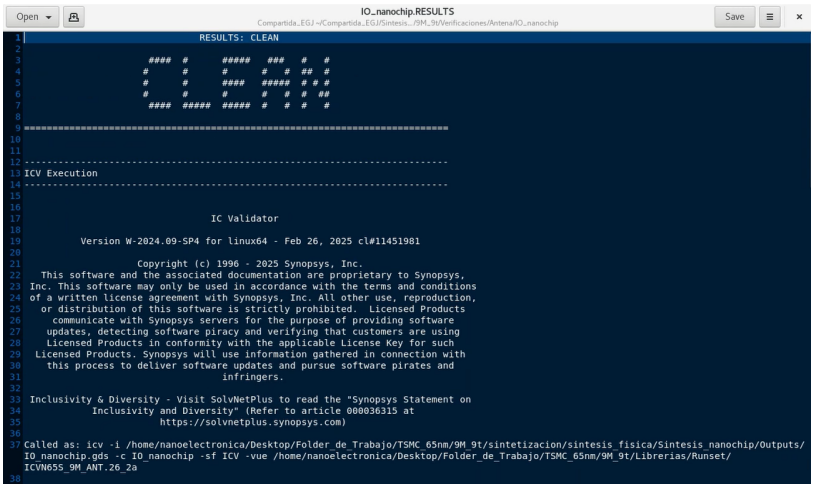
Finalmente, según lo descrito, el Cuadro 10.27 contiene el detalle de los errores DRC según el *runset* proporcionado por TSMC, por lo cual estas son las reglas reales que aún deben ser corregidas para obtener un diseño libre de violaciones. Los mensajes corresponden a los errores de densidad de las capas OD/PO al igual que en los casos anteriores, pero además se observan errores relacionados con el espaciado mínimo en las capas M2 y M3. Tomando este resultado como referencia permitió observar que la cantidad de errores se redujo de 15124 a 614 (hasta un 96.00 %), tomando en consideración que la primera cantidad

ya tenía mejoras implementadas y este circuito no se sintetizó con los *scripts* originales que generarían una cantidad más elevada de errores.

### 10.5.4. Resultados de verificación de antena del circuito "El Gran Jaguar"

Finalmente se realizó la prueba de antena con el circuito de "El Gran Jaguar", con el mismo método descrito en secciones anteriores. El resultado se muestra en la Figura 48, en donde se observa que para este diseño tampoco se encontraron violaciones de las reglas de antena.

**Figura 48.** Resultado de la verificación de antena del circuito "El Gran Jaguar"




Nota. Esta imagen muestra el resultado libre de errores de la verificación de antena del circuito "El Gran Jaguar".

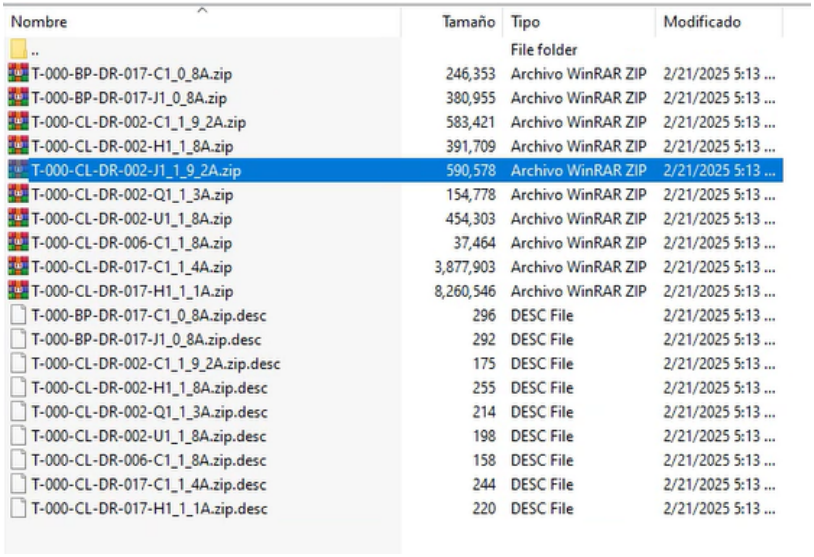
# CAPÍTULO 11

## Verificaciones BND

Las verificaciones BND son un conjunto de pruebas que se realizan para asegurar que el diseño físico cumple con los requisitos de conectividad desde el DIE al empaquetado del circuito. Estas verificaciones dependen de la tecnología de fabricación, ya que existen *runsets* que pueden corresponder a la tecnología *flip-chip* o a la tecnología *wire-bonding*. En este caso se utilizó la segunda opción dado que es la que IMEC brinda para proyectos como el presente. A partir de esta información fue posible encontrar un *deck* de verificaciones BND que se encuentra disponible en la carpeta principal de la tecnología en el directorio llamado "*utilsz* del cual se muestra su contenido en la Figura 49.

Para ejecutar el *runset* es necesario previamente definir la configuración correcta según las indicaciones del fabricante, del mismo modo que se configuraron los *runsets* de DRC y de rellenos de metal.

**Figura 49.** Ruta hacia el *deck* de verificaciones BND



| Nombre | Tamaño | Tipo | Modificado |
|---|---|---|---|
| .. | | File folder | |
| T-000-BP-DR-017-C1_0_8A.zip | 246,353 | Archivo WinRAR ZIP | 2/21/2025 5:13 ... |
| T-000-BP-DR-017-J1_0_8A.zip | 380,955 | Archivo WinRAR ZIP | 2/21/2025 5:13 ... |
| T-000-CL-DR-002-C1_1_9_2A.zip | 583,421 | Archivo WinRAR ZIP | 2/21/2025 5:13 ... |
| T-000-CL-DR-002-H1_1_8A.zip | 391,709 | Archivo WinRAR ZIP | 2/21/2025 5:13 ... |
| T-000-CL-DR-002-J1_1_9_2A.zip | 590,578 | Archivo WinRAR ZIP | 2/21/2025 5:13 ... |
| T-000-CL-DR-002-Q1_1_3A.zip | 154,778 | Archivo WinRAR ZIP | 2/21/2025 5:13 ... |
| T-000-CL-DR-002-U1_1_8A.zip | 454,303 | Archivo WinRAR ZIP | 2/21/2025 5:13 ... |
| T-000-CL-DR-006-C1_1_8A.zip | 37,464 | Archivo WinRAR ZIP | 2/21/2025 5:13 ... |
| T-000-CL-DR-017-C1_1_4A.zip | 3,877,903 | Archivo WinRAR ZIP | 2/21/2025 5:13 ... |
| T-000-CL-DR-017-H1_1_1A.zip | 8,260,546 | Archivo WinRAR ZIP | 2/21/2025 5:13 ... |
| T-000-BP-DR-017-C1_0_8A.zip.desc | 296 | DESC File | 2/21/2025 5:13 ... |
| T-000-BP-DR-017-J1_0_8A.zip.desc | 292 | DESC File | 2/21/2025 5:13 ... |
| T-000-CL-DR-002-C1_1_9_2A.zip.desc | 175 | DESC File | 2/21/2025 5:13 ... |
| T-000-CL-DR-002-H1_1_8A.zip.desc | 255 | DESC File | 2/21/2025 5:13 ... |
| T-000-CL-DR-002-Q1_1_3A.zip.desc | 214 | DESC File | 2/21/2025 5:13 ... |
| T-000-CL-DR-002-U1_1_8A.zip.desc | 198 | DESC File | 2/21/2025 5:13 ... |
| T-000-CL-DR-006-C1_1_8A.zip.desc | 158 | DESC File | 2/21/2025 5:13 ... |
| T-000-CL-DR-017-C1_1_4A.zip.desc | 244 | DESC File | 2/21/2025 5:13 ... |
| T-000-CL-DR-017-H1_1_1A.zip.desc | 220 | DESC File | 2/21/2025 5:13 ... |

Nota. Esta imagen muestra la ubicación del *deck* para las verificaciones BND en ICC2.

Tomando esta información como referencia y con el apoyo de Synopsys a través de la plataforma de SolvNetPlus fue posible identificar que había un comando que no se había utilizado de la mejor manera en el *script* de configuración del entorno de ICC2. Según la información trasladada por Synopsys, para efectuar las pruebas de BND se deben utilizar los comandos que se muestran en el Cuadro 11.1.

**Cuadro 11.1.** Comandos para ejecución de verificaciones BND

```
1
2 # Comando que se incluye en el setup.tcl para configurar la ruta del archivo
      runset de BND
3 set_app_options -name signoff.check_design.runset -value <path_to_your_runset_file
      >
4
5 # Comando que se incluye en el segmento de verificaciones para ejecutar las
      pruebas BND
6 signoff_check_design
```

Nota. Este cuadro contiene los comandos necesarios para ejecutar las verificaciones BND en ICC2. El primer comando se debe colocar en el *script* de `setup.tcl` y el segundo en `sintesis_fisica_top.tcl`.

En los flujos implementados previamente no se había configurado correctamente este *runset*, lo cual también es importante considerando que son aspectos cruciales en el proceso de fabricación del circuito y en caso no se cumplan el diseño no podrá ser fabricado correctamente.

El comando `signoff_check_design` permite invocar la herramienta de IC Validator para generar las verificaciones y validar si existen violaciones del diseño. Dado que fue uno de los aspectos que se iniciaron a explorar en el presente proyecto aún requieren de revisión y pruebas, así como también es necesario indagar en las opciones como las que se muestran en el Cuadro 11.2 que fueron brindadas como ejemplo por Synopsys para configurar las reglas de *wire-bonding*.

**Cuadro 11.2.** Ejemplo de Synopsys para la configuración de reglas de *wire-bonding*

```
1
2 place_io
3 create_routing_rule wirebond_rule -widths {M1 1.0} -spacings {M1 0.5}
4 set_routing_rule -rule wirebond_rule [get_nets -of_objects [get_pins -of_objects [
      get_cells * -filter "design_type==pad"]]]
5 route_eco -layers {M1}
```

Nota. Los comandos del cuadro fueron brindados por Synopsys como ejemplo para configurar las reglas de *wire-bonding* en ICC2 para posterior ejecutar las revisiones de BND.

# CAPÍTULO 12

## Metodología de referencia de Synopsys

A partir de la información proporcionada por Synopsys fue posible encontrar documentación relacionada con la metodología de referencia (RM, por sus siglas en inglés) que se utiliza para la herramienta IC Compiler II. Esta metodología es un conjunto de *scripts* prediseñados por Synopsys que permiten ejecutar el flujo de la síntesis física completo con ICC2.

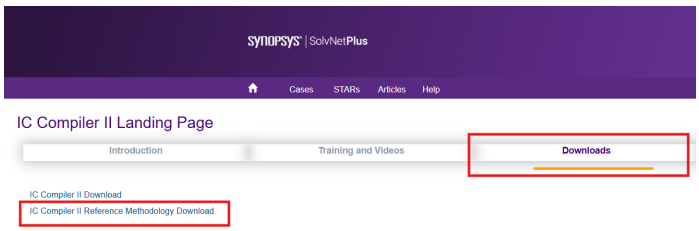
Estos *scripts* están diseñados para que se puedan configurar de acuerdo a las necesidades del usuario y dependiendo de cómo se elijan las opciones se pueden tener diferentes archivos de entrada y correr secciones específicas del flujo. La documentación de la RM se encuentra disponible en SolvNetPlus y contiene información detallada sobre cada uno de los pasos que se deben seguir para ejecutar el flujo completo, así como también la explicación de cada uno de los comandos que se utilizan en los *scripts*.

El trabajo escrito por Miguel Chacón en [33], contiene información más detallada sobre el uso de la RM de IC Compiler II así como su estructura y los flujos recomendados por Synopsys.

## 12.1. Descripción general de la RM para IC Compiler II

La metodología de referencia de IC Compiler II se puede descargar desde la página oficial de SolvNetPlus, para lo cual es necesario contar con una cuenta activa y los permisos necesarios para acceder a las descargas. En la Figura 50 se observa la ventana de descargas de SolvNetPlus, en donde se encuentra la sección de Metodologías de Referencia. Una vez en esta ventana se deben configurar las opciones que se enmarcan en la Figura 51 para descargar la versión correspondiente a IC Compiler II.

**Figura 50.** Descarga de la RM de IC Compiler II desde SolvNetPlus



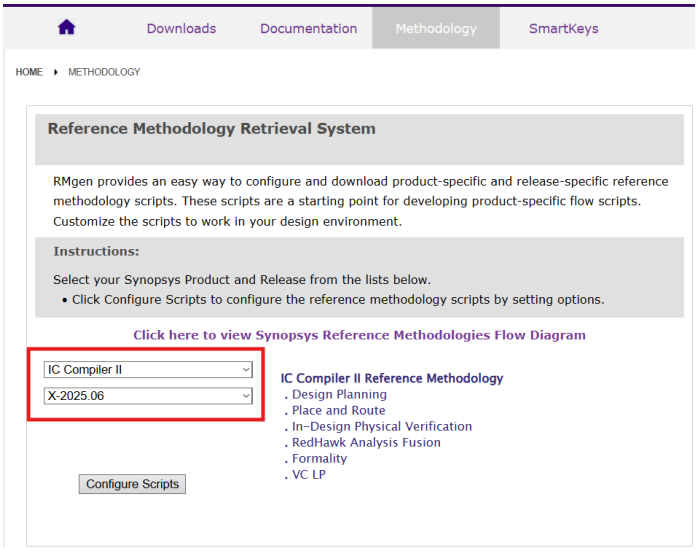

Nota. La imagen muestra cómo acceder a la ventana de descargas de SolvNetPlus para obtener la metodología de referencia de IC Compiler II.

Como se indica en la Figura 51, la metodología de referencia para ICC2 contiene las fases de:

- *Design Planning*
- *Place and Route*
- *In-Design Physical Verification*
- *RedHawk Analysis Fusion*
- *Formality*
- *VC LP*

Para fines del proyecto solo se enfocó en las primeras 3 fases dado que son las que corresponden a la síntesis física y verificación del diseño. Como se indicó anteriormente, para observar el trabajo realizado con la RM de IC Compiler II se recomienda revisar el documento escrito por Miguel Chacón en [33].

**Figura 51.** Configuración para la descarga de la RM de IC Compiler II

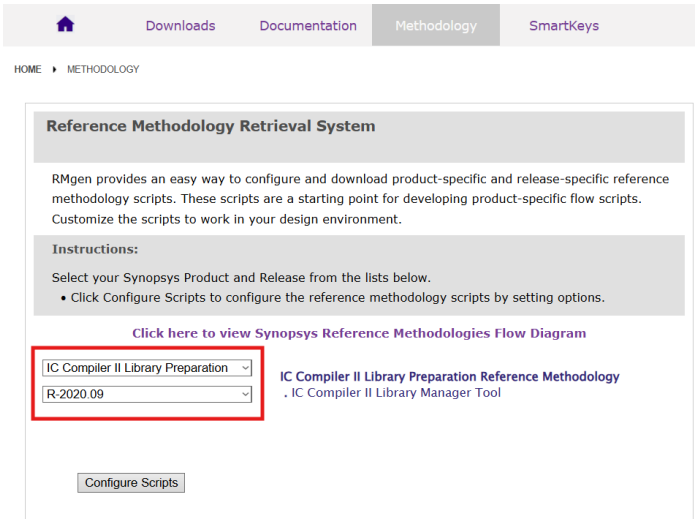



Nota. La imagen muestra la configuración en la ventana de descargas de RM para descargar la versión correspondiente a IC Compiler II.

## 12.2. Descripción de la RM para Library Manager Tool

Así como se encuentra disponible un flujo para IC Complier II, en la página oficial de SolvNetPlus se encuentran disponibles las metodologías de referencia de otras herramientas, como lo es el caso de Library Manager Tool. Esta metodología contiene el flujo recomendado para la generación de librerías NDM a partir de diferentes entradas como lo pueden ser librerías Milkyway, archivos de tecnología LEF, archivos DB, entre otros. La documentación de esta metodología también se encuentra disponible en SolvNetPlus. En el presente proyecto no se implementó en su totalidad pero se investigó la estructura y a continuación (en la Figura 52) se muestra cómo configurar la descarga de esta RM desde SolvNetPlus.

**Figura 52.** Configuración para la descarga de la RM de Library Manager Tool




Nota. La imagen muestra la configuración en la ventana de descargas de RM para descargar la versión correspondiente a Library Manager Tool.

# CAPÍTULO 13

## Generación de librerías NDM a partir de las librerías Milkyway

Al generar las librerías en la herramienta de Library Manager Tool fue posible observar una advertencia que generó confusión durante la lectura del archivo LEF y que se muestra en el Cuadro 13.1. Si bien esta advertencia no representó un error crítico como para detener el proceso de creación de librerías, esta advertencia sí puede tener influencia en el desarrollo de otras fases del diseño.

**Cuadro 13.1.** Mensajes de advertencia en la lectura de LEF en la creación de librerías

```
1
2 Warning: Line 51, Ignoring PROPERTYDEFINITION 'LAYER LEF57_MINSTEP'; LAYER
    property definition is not supported. (LEFR-050)
3 Warning: Line 2041, Ignoring unsupported syntax 'WIDTH' on VIARULE GENERATE '
    VIAGEN12'. (LEFR-017)
4 Warning: Line 2477, OBS (1.39, 1.11),(1.45, 1.29) on layer M1 does not have a "
    SPACING 0" attribute but overlaps or touches pin AN2D4/Z. (LEFR-067)
```

Nota. El cuadro contiene los mensajes de advertencia que se observaron al leer el archivo LEF durante la creación de las librerías NDM.

Los mensajes de advertencia que se observan en el cuadro indican que existen definiciones del archivo que no son compatibles con la sintaxis que espera la herramienta. Dado que no son errores críticos la herramienta continúa con la creación de las librerías, pero el hecho de que no se lean correctamente genera inconsistencias en la información física de las celdas que pueden tener impacto en etapas posteriores como lo puede ser el enrutamiento o la colocación de las celdas, ya que la herramienta no es capaz de colocar correctamente las rutas o las celdas porque lo interpreta como violaciones del diseño.

A partir de esta información se realizó una revisión del repositorio compartido por IMEC, de modo que al revisar la ruta donde se obtienen los archivos DB se encontró el archivo con nombre `VERSION_NUMBERING_SCHEME.txt`, en el cual se observaron los detalls que se observan en el Cuadro 13.2 en cuanto a los archivos con extensión .tar que se habían utilizado hasta el momento.

**Cuadro 13.2.** Explicación de la numeración de versiones de los archivos .tar

```
apf = Apollo FRAM (phantom) view
apt = Apollo FRAM and CEL (layout) view
sef = Silicon Ensemble LEF (phantom) view
syn = Synopsys
```

Nota. El cuadro contiene un segmento del texto del archivo que contiene la explicación de la numeración de versiones de los archivos .tar del repositorio de IMEC.

Al observar este cuadro fue posible determinar que la ruta a la cual se había estado haciendo referencia por el archivo LEF (con terminación "sef") corresponde a una herramienta ajena a la *suite* de Synopsys. Por otro lado, no se encontró ningún archivo con extensión .tar que corresponda directamente a Synopsys, pero por otro lado sí se encontraron archivos con la terminación .ªpf"que corresponde a una herramienta antigua de la *suite* de Synopsys llamada Apollo.

Una vez identificado esto, se procedió a verificar el archivo con nombre `tcbn65lp_200a_apf.tar` y se observó que dentro contiene una carpeta llamada "milkyway". El nombre Milkyway corresponde a un formato de librerías que se utilizaba en versiones anteriores de la *suite* de Synopsys, de modo que al abrir el archivo con la información del kit (llamado `DESIGNKIT.INFO`) fue posible comprender que la librería fue realizada con la herramienta de Milkyway y es utilizable con las herramientas de Apollo y Astro. El contenido relevante que se tomó como referencia del archivo se muestra en el Cuadro 13.3.

**Cuadro 13.3.** Contenido de la librería tcbn65lp de Apollo

```
1
2  Design Kit Information
3
4     Applicable EDA Tool: Apollo / Astro
5     Library Name       : tcbn65lp
6
7  Tools used for generation of this design-kit
8
9     Tool Name            Version
10    ---------            -------
11    Milkyway             2007.12-SP5-5
12
13 Design Kit Contents
14
15    Apollo library
16
17           tcbn65lp/
```

Nota. El cuadro contiene segmentos importantes del archivo DESIGNKIT.INFO que describen el contenido de la librería tcbn65lp de Apollo.

Una vez realizada esta revisión se llegó a la conclusión que algunos de los errores podrían deberse a que la librería no está diseñada originalmente para las herramientas de ICC2 ni Library Manager Tool, lo cual puede causar errores como los indicados en el Cuadro 13.1. A partir de esto se exploró otra forma de generar las librerías NDM a partir de las que se encuentran en formato Milkyway, de modo que en la siguiente sección se describe el procedimiento que se siguió para lograrlo.

## 13.1. Generación de librerías NDM a partir de librerías Milkyway

Para utilizar este proceso, primero fue necesario instalar el entorno de Milkyway desde la página oficial de Synopsys SolvNetPlus, donde esta herramienta aún se encuentra disponible. Cabe destacar que esto funciona siempre y cuando se realice el proceso de agregar las variables de entorno y posiblemente durante la instalación será necesario añadir librerías como las que se añadieron con los comandos del Cuadro 13.4.

**Cuadro 13.4.** Librerías agregadas para instalar la herramienta de Milkyway

```

# Instalar libcurses.so.5
sudo dnf install libcurses.so.5
# Instalar vnc server
sudo dnf install tigervnc-server
```

Nota. Los comandos listados en este cuadro permiten la instalación de librerías adicionales necesarias para ejecutar el entorno de Milkyway.

A partir de la documentación de Synopsys para la herramienta de IC Compiler II [34] fue posible observar el siguiente comando:

```

generate_frame_from_mw
```

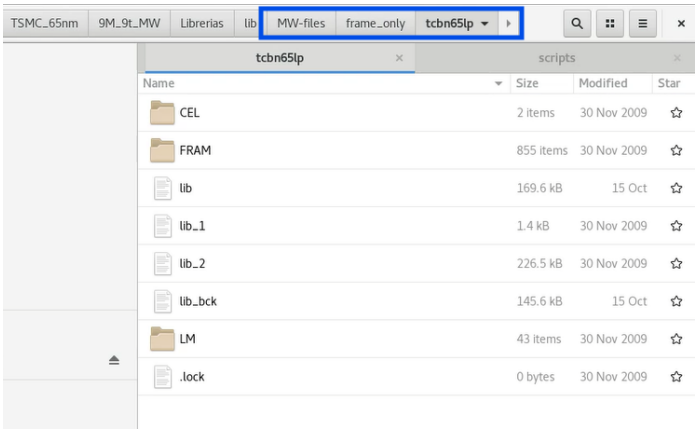
Este comando permite la creación de librerías NDM qu econtienen la información física de las celdas para que puedan ser utilizadas en el entorno de IC Compiler II a partir de librerías con formato Milkyway. Este comando recibe por defecto el nombre de la librería que se va a generar, y necesita como entrada la opción `-mw_lib` para recibir la carpeta en formato Milkyway que se va a utilizar como referencia. Para seleccionar las librerías se tomó como referencia las rutas descritas en la sección 7.1 para los archivos DB, de modo que se tomaron los siguientes dos archivos que contienen una carpeta con el nombre "milky-way"que corresponde a las librerías de las celdas estándar y las celdas de entradas y salidas respectivamente:

- `tcbn65lp_200a_sef.tar.gz`
- `tpan65lpnv2od3_200a_apfu9lm.tar.gz`

En la Figura *fig:dirmw* se observa el contenido de la carpeta de la librería Milkyway el cual fue extraído del primer archivo mencionado y trasladado al directorio del folder de trabajo correspondiente. En la imagen es posible observar que contiene 3 carpetas que se mencionan a continuación:

- CEL: este directorio contiene las definiciones de los sitios de las celdas estándar. En este caso existen dos sitios denominados "*unitz* "*gaunit*".
- FRAM: esta carpeta contiene toda la información de los *frames* de todas las celdas disponibles en la librería. Estos archivos tienen la extensión .FRAME.
- LM: este directorio contiene la información lógica de las celdas en los archivos DB. Estos son utilizados por las herramientas de Library Manager para generar las librerías completas con la información física y lógica de las celdas.

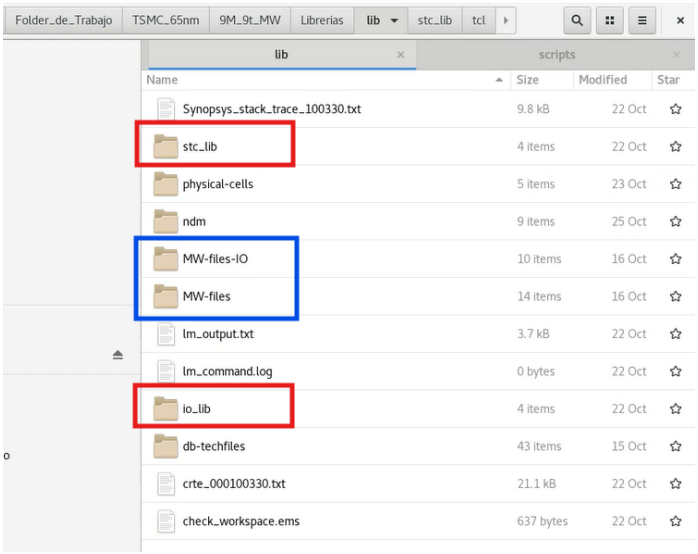
**Figura 53.** Contenido de la librería Milkyway de las celdas estándar




Nota. Esta imagen muestra el contenido de la carpeta de la librería Milkyway de las celdas estándar para 9 capas de metal con 9 *tracks*.

Se realizó el mismo procedimiento con la librería de las celdas del anillo de entradas y salidas, de modo que se generó una carpeta para cada uno dentro del directorio equivalente utilizado para las pruebas como se observa en la Figura 54.

**Figura 54.** Ubicación de librerías Milkyway en el folder de trabajo




Nota. La imagen muestra en azul la ubicación de las librerías Milkyway utilizadas para la generación de los archivos NDM con el nuevo proceso. En rojo se encuentran señaladas las salidas del comando de generación de *frames* a partir de Milkyway.

Una vez preparados los documentos se puede dar inicio al proceso de creación de librerías NDM desde la herramienta de Library Manager. El Cuadro 13.5 contiene la secuencia de comandos que hay que utilizar para la creación de las librerías de las celdas estándar. Este

mismo procedimiento se debe realizar con las librerías de entradas y salidas.

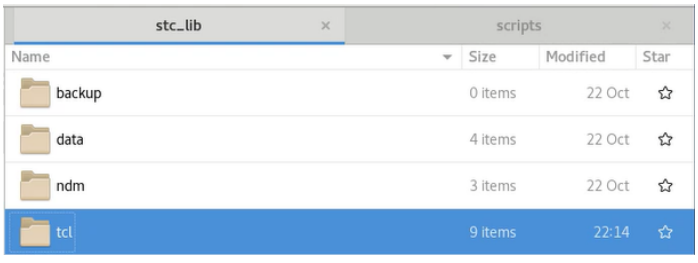
**Cuadro 13.5.** Comandos para generación de archivos a partir de librerías Milkyway

```
1
2  # Iniciar el entorno de Library Manager
3  lm_shell
4
5  # Definir la ruta hacia el ejecutable de la herramienta Milkyway
6  set_app_options -name lib.setting.milkyway_exec -value /usr/synopsys/mw/X-2025.06/
       bin/linux64/Milkyway
7
8  # Comando a utilzar en caso se emplee IC Compiler en lugar de Milkyway
9  # set_app_options -name lib.setting.icc_shell_exec -value <ruta a ejecutable de
       ICC>
10
11 # Generar archivos a partir de Milkyway
12 generate_frame_from_mw lib_prueba -mw_lib /home/nanoelectronica/Desktop/
       Folder_de_Trabajo/TSMC_65nm/9M_9t_MW/Librerias/lib/MW-files/frame_only/
       tcbn65lp -overwrite
```

Nota. Este cuadro contiene la secuencia de comandos que hay que ingresar a la consola para la generación de archivos a partir de librerías Milkyway de referencia en Library Manager.

El resultado de utilizar este comando es una carpeta como las que se muestran en color rojo en la Figura 54. Estas carpetas de salida contienen una estructura como la que se muestra en la Figura 55, en donde la carpeta más relevante para el flujo que se ha utilizado en el proyecto es la llamada "tcl". Dentro de esta carpeta se encuentra un *script* con nombre `run_icc2_lm.tcl` que se puede utilizar nuevamente en el entorno de Library Manager y es el que finalmente permitirá generar los archivos NDM que se utilizarán en el flujo descrito en secciones anteriores. El Cuadro 13.6 contiene el *script* que se genera automáticamente el cual solamente se modificó en las secciones que indica con el comentario "*Please complete the following...*".

**Figura 55.** Estructura de la carpeta de salida del comando para generación de librerías a partir de Milkyway




Nota. La imagen muestra la estructura interna de la carpeta generada como salida del comando `generate_frame_from_mw`.

**Cuadro 13.6.** *Script* para generación de archivos NDM a partir de librerías Milkyway

```

# ****** create tech only NDM ******
create_workspace tcbn65lp_technology_only -technology /home/nanoelectronica/
    Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t_MW/Librerias/lib/stc_lib/data/TF/
    tcbn65lp.tf -flow normal
source /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t_MW/
    Librerias/lib/stc_lib/data/TCL/tcbn65lp_update_technology.tcl
commit_workspace
catch {sh mv tcbn65lp_technology_only.ndm /home/nanoelectronica/Desktop/
    Folder_de_Trabajo/TSMC_65nm/9M_9t_MW/Librerias/lib/stc_lib/ndm}

# ****** create frame only NDM ******
create_workspace tcbn65lp_frame_only -technology /home/nanoelectronica/Desktop/
    Folder_de_Trabajo/TSMC_65nm/9M_9t_MW/Librerias/lib/stc_lib/data/TF/tcbn65lp.tf
     -flow frame
#import_icc_fram /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9
    M_9t_MW/Librerias/lib/stc_lib/data/LEF/tcbn65lp.tar.gz
read_lef /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t_MW/
    Librerias/lib/stc_lib/data/LEF/tcbn65lp/tcbn65lp.lef -preserve_lef_cell_site

check_workspace
commit_workspace
catch {sh mv tcbn65lp_frame_only.ndm /home/nanoelectronica/Desktop/
    Folder_de_Trabajo/TSMC_65nm/9M_9t_MW/Librerias/lib/stc_lib/ndm}

# ****** merge .db files with frame only NDM ******
set_app_options -name lib.logic_model.auto_remove_timing_only_designs -value true
create_workspace tcbn65lp_merge_db -flow exploration
read_ndm -view frame /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9
    M_9t_MW/Librerias/lib/stc_lib/ndm/tcbn65lp_frame_only.ndm

# Please complete the following set_pvt_configuration command
set_pvt_configuration -clear_filter all -process_labels {typ} -process_numbers {1
    .0} -voltages {0.9} -temperatures {25}

# Please complete the following read_db command
read_db [glob {/home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9M_9t_MW/
    Librerias/lib/db-techfiles/*.db}]

group_libs
write_workspace -f exploration.tcl
check_workspace
commit_workspace
catch {sh mv *.ndm /home/nanoelectronica/Desktop/Folder_de_Trabajo/TSMC_65nm/9
    M_9t_MW/Librerias/lib/stc_lib/ndm}
```

Nota. Este *script* contiene los comandos necesarios para la generación de archivos NDM a partir de librerías Milkyway el cual fue generado automáticamente por la herramienta de Library Manager.

Este mismo proceso se debe repetir para la librería de celdas de entradas y salidas, para generar un conjunto de archivos NDM que se estarán utilizando en el flujo descrito para la síntesis física en capítulos anteriores. Cabe resaltar que este procedimiento aún no se ha probado en su totalidad y aún se generan errores al utilizar las librerías en el flujo de ICC2. Por tal motivo en esta sección se documentó el proceso para la generación de NDM pero una siguiente fase de investigación puede estar enfocada en la implementación de las librerías en el flujo de diseño.

# CAPÍTULO 14

## Conclusiones

- En el trabajo realizado, fue posible replicar los trabajos de estudiantes de años anteriores y esto permitió una mejor comprensión del flujo de diseño de circuitos integrados.
- Fue posible definir una gran cantidad de cambios para adaptar el trabajo previo a la tecnología de 65 nm. Estos cambios se debieron tanto a que la tecnología es diferente, como a que en los años anteriores no habían tenido acceso a contenido detallado sobre el manejo de las herramientas de Synopsys.
- No fue posible obtener resultados libres de errores en las pruebas de DRC y BND, dado que para adaptar la tecnología de 65 nm fue necesario cambiar la estructura principal que se tenía de trabajos anteriores. Esto se realizó debido a que, al tener acceso a los cursos de Synopsys, se determinó que había fases que no se estaban desarrollando de manera adecuada o bien defectos en el uso de comandos en etapas como el *floorplan*, *placement* y el enrutamiento.
- Se logró realizar correcciones en el flujo de diseño, más aún es necesario realizar depuraciones para garantizar que el circuito sea fabricable. Esto se debe a que existen manuales extensos para el uso de cada una de las herramientas de Synopsys y comprenderlos es fundamental para lograr eliminar los errores y realizar un proceso más limpio en el diseño de circuitos integrados.
- Durante el trabajo fue posible conocer y explorar sobre los *runsets* BND. Aún es necesario realizar pruebas de implementación y realizar las modificaciones en el archivo para que corresponda a las indicaciones brindadas por IMEC y TSMC.
- Se ejecutó de forma exitosa el método que permite realizar la adaptación de las librerías Milkyway al entorno de ICC2 en formato NDM. Aún es necesario generar cambios en el *script* de generación de archivos para que puedan ser utilizados en el flujo establecido.
- Finalmente, el último de los objetivos planteados inicialmente se cumplió ya que se desarrollaron *scripts* con explicaciones detalladas del funcionamiento de cada comando para que futuros estudiantes puedan utilizarlos como guía para darle continuidad al

proyecto de El Gran Jaguar. Aún es necesario realizar mejoras para el manejo de las verificaciones BND y también para la generación del *layout* con cortes de 45°.

# CAPÍTULO 15

## Recomendaciones

- El desarrollo del proyecto requiere un gran nivel de implicación por parte del equipo de trabajo. Es necesario comprender a profundidad el flujo de diseño propuesto por las herramientas de Synopsys para tener éxito en la implementación del circuito. Es recomendable que previo a iniciar el desarrollo y pruebas con los trabajos realizados hasta el momento, se realicen los cursos de Synopsys y se verifiquen con detalle los documentos brindados por IMEC para garantizar que se cuenta con toda la información necesaria para el desarrollo del proyecto.
- Debe considerarse en el alcance del proyecto que el trabajo llevará una parte fuerte de investigación, dado que si no se tiene una buena comprensión sobre herramientas como Design Compilar, IC Compiler II y StarRC, pueden tenerse errores que no se logren solucionar fácilmente o bien realizar implementaciones incorrectas como sucedió en trabajos anteriores. Para saber cómo manejar los *runsets* y archivos brindados por IMEC también deben revisar documentos como el manual de diseño de 65 nm y el manual de fabricación, ya que en él se detallan modificaciones que se necesitan hacer para cumplir con los requisitos de manufactura del circuito.
- Es recomendable que el equipo de trabajo tenga retroalimentación continua y mantenenga comunicación con los contactos de IMEC y Synopsys, ya que esto permitirá resolver dudas y problemas que puedan surgir durante las distitas etapas del proyecto. Sin embargo, más allá de esperar respuestas es necesario que el equipo realice pruebas propias y busque las soluciones tomando como referencia la gran cantidad de manuales y documentos a los que tienen acceso en la Universidad del Valle de Guatemala, ya que esto permitirá tanto un mejor aprendizaje como evitar que el proyecto se estanque esperando respuestas externas.
- Para el desarrollo del proyecto es recomendable dar seguimiento a la investigación correspondiente a la metodología de referencia y a la verificación de las advertencias que se obtienen al generar las librerías. Es posible que verificar estos errores permitan obtener un mejor resultado en las verificaciones.

- Finalmente, parte importante del proyecto es tener pensamiento crítico y cuestionarse constantemente si las bases que están utilizando son las correctas, dado que al ser un área poco explorada en el país, es posible que existan errores y sea necesario realizar adaptaciones y cambios en los trabajos que existen en la actualidad.

# CAPÍTULO 16

## Referencias


[1] L. A. N. Vásquez, «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 doct., Universidad del Valle de Guatemala, 2019.

[2] S. H. R. Vasquez, «Definición del Flujo de Diseño para Fabricación de un Chip con Tecnología VLSI CMOS,» Tesis doct., Universidad del Valle de Guatemala, 2019.

[3] J. A. de los Santos Chonay, «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.,» 2014.

[4] K. D. Hernández, «Automatización de la etapa de síntesis lógica, optimización del proceso de instalación de las aplicaciones de Synopsys y documentación de los pasos de Front-End y Back-End para el diseño del Circuito Integrado "El Gran Jaguar",» 2024.

[5] M. S. Illescas, «Verificación de reglas de diseño (DRC) para el desarrollo de un flujo funcional de un circuito integrado con tecnología nanométrica.,» 2021.

[6] M. G. F. Espino, «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.,» 2021.

[7] L. E. A. López, «Posicionamiento e interconexión entre componentes de un circuito sintetizado para el flujo de diseño de un circuito a escala nanométrica utilizando la herramienta de IC Compiler.,» 2022.

[8] A. A. Hernández, «Diseño de un circuito integrado con tecnología de 180nm usando librerías de diseño de TSMC: ejecución de la síntesis física, verificación de reglas de diseño y corrección de errores obtenidos.,» 2022.

[9] J. A. A. Escobar, «Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: ejecución de la síntesis física, verificaciones de antena y corrección de errores obtenidos.,» 2022.

[10] J. E. S. Jo, «Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC : ejecución de la síntesis física, validación de reglas eléctricas y corrección de errores obtenidos.,» 2022.

[11] D. R. E. Pirir, «Diseño e implementación de interfaz gráfica de la automatización de las fases de diseño de un circuito integrado y uso avanzado de IC Compiler II.,» 2022.

[12] C. A. L. Torres, «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.,» 2023.

[13] N. F. P. Búcaro, «Diseño de un circuito integrado con tecnología de 180 nm, utilizando las librerías de diseño de TSMC y las librerías educativas de Synopsys: corrección de errores de densidad y polisilicio, verificación DRC y antena.,» 2024.

[14] D. S. A. Mota, «Diseño y fabricación de un circuito integrado con tecnología de 65 nm usando librerías de diseño de TSMC: pruebas de LVS, ERC y extracción de parásitos,» 2024.

[15] L. M. R. Vásquez, «Diseño y fabricación de un circuito integrado con tecnología de 65 nm usando librerías de diseño de TSMC: pruebas finales de funiconamiento en HSPICE, generación y documentación de la síntesis física, verificaciones de Antena y DRC,» 2024.

[16] IMEC, «Ic-link Tapeout Team - Mini@sic Manual TSMC 28nm, 40nm, and 65nm,» jun. de 2024.

[17] K. L. Kishore y V. Prabhakar, *VLSI DESIGN*. I.K. International Publishing House Pvt. Ltd., 2010, págs. 1-4.

[18] M. M. Vai, *Vlsi Design*. CRC Press, dic. de 2017, págs. 1-10, ISBN: 9781315274201. DOI: 10.1201/9781315274201.

[19] R. B. Singh, A. S. Baghel y A. Agarwal, «A review on VLSI floorplanning optimization using metaheuristic algorithms,» en *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, IEEE, mar. de 2016, págs. 4198-4202, ISBN: 978-1-4673-9939-5. DOI: 10.1109/ICEEOT.2016.7755508.

[20] W. Song y L. Glasser, «Power distribution techniques for VLSI circuits,» *IEEE Journal of Solid-State Circuits*, vol. 21, págs. 150-156, 1 feb. de 1986, ISSN: 0018-9200. DOI: 10.1109/JSSC.1986.1052491.

[21] A. B. Chong, «Hybrid Multisource Clock Tree Synthesis,» en *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, IEEE, nov. de 2021, págs. 1-6, ISBN: 978-1-7281-8281-0. DOI: 10.1109/ICECS53924.2021.9665516.

[22] X. Chen, G. Liu, N. Xiong, Y. Su y G. Chen, «A Survey of Swarm Intelligence Techniques in VLSI Routing Problems,» *IEEE Access*, vol. 8, págs. 26 266-26 292, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2971574.

[23] Synopsys, *IC Compiler Online Help*, 2025.

[24] B. H. Shraddha, N. Shanbhag, S. V. Siddamal y N. C. Iyer, «Implementation of Custom DRC in TSMC 0.18 µm for Optimized Layout,» en 2021, págs. 481-491. DOI: 10.1007/978-981-33-4866-0_59.

[25] Synopsys, *IC Validator Online Help*, 2025.

[26] R. Roberts y C. Fourie, «Layout-versus-schematic verification for superconductive integrated circuits,» *IEEE Transactions on Applied Superconductivity*, págs. 1-1, 2014, ISSN: 1051-8223. DOI: 10.1109/TASC.2014.2373035.

[27] K. Patel, «Boundary Scan Verification,» *Institute of Technology*, jun. de 2020.

[28] V. Shukla, V. Gupta, C. Guruprasad y G. Kadamati, «Automated antenna detection and correction methodology in VLSI designs,» en *2003 8th International Symposium Plasma- and Process-Induced Damage.*, IEEE, abr. de 2003, págs. 158-161, ISBN: 0-7803-7747-8. DOI: 10.1109/PPID.2003.1200947.

[29] Synopsys, *IC Compiler™ II Implementation User Guide*. 2022.

[30] Synopsys, *IC Validator Landing Page*, 2025.

[31] Synopsys, *IC Validator Physical Verification Datasheet*. Synopsys, 2019.

[32] Synopsys, *Using the IC Validator Tool in Your Design Flow*, 2025.

[33] M. Chacón, «Desarrollo y documentación del Proceso de Síntesis Física, DRC y BND para el nanochip El Gran Jaguar en tecnología de 65nm de TSMC, utilizando herramientas de Synopsys,» Tesis doct., Universidad del Valle de Guatemala, 2025.

[34] Synopsys, *IC Compiler II Tool Commands Version X-2025.06-SP1*, ago. de 2025.

[35] Synopsys, *IC Compiler II: Block Level Implementation*, 2025.

# CAPÍTULO 17

# Anexos

## 17.1. Replicación del proceso de síntesis física con tecnología de 180 nanometros

Inicialmente se realizó una revisión de la documentación de los trabajos anteriores para identificar los procedimientos de síntesis física, las herramientas utilizadas y los resultados obtenidos. Primero fue necesario comprender la estructura y la jerarquía de directorios diseñada para la ubicación de los archivos. Dado que este fue el primer paso del equipo de trabajo encargado del proyecto en el año 2025, no se hicieron cambios en los archivos originales ni en los comandos empleados para ejecutar la síntesis física.

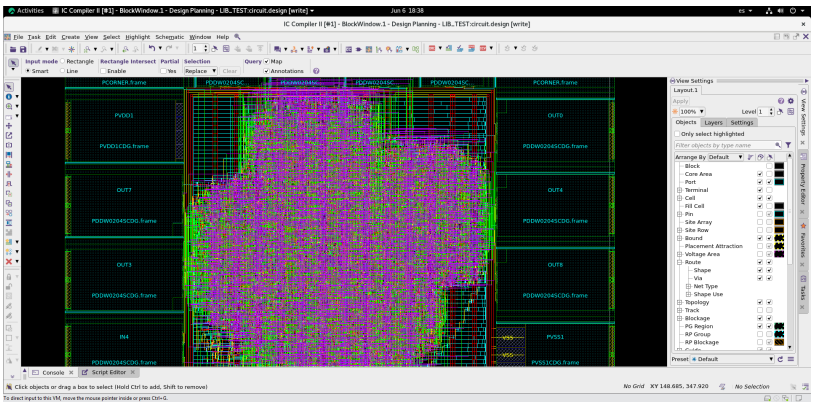
Se realizó el proceso de síntesis física con la tecnología de 180 nm para la compuerta NOT y para el circuito de "El Gran Jaguar". Para lograr la ráplica primero fue necesario elaborar las librerías tomando como referencia los videos y documentación realizada por Lourdes Ruiz [15] ya que realizó las últimas adecuaciones a los archivos con la tecnología anterior. Se utilizó la herramienta de Library Manager, con las rutas y archivos seleccionados según la documentación antes mencionada.

Posterior a la elaboración de librerías, se realizó un análisis de los *scripts* llamados **sintesis_fisica_top.tcl, setup.tcl y floorplan.tcl**. Estos archivos son los que realizan el proceso de síntesis física y en ellos también se ejecutan comandos de las herramientas de IC Compiler II y IC Validator que permiten la ejecución de pruebas como DRC y verificaciones de antena. En secciones posteriores se analiza con mayor detalle cada uno de los archivos y los cambios que fueron necesarios realizar en la transición a la tecnología de 65 nm.

Estas pruebas iniciales presentaron algunas diferencias dado que, a pesar de reutilizar los archivos, se trabajó con actualizaciones de la suite de Synopsys. Por tal motivo, previo a realizar este procedimiento fue necesaria la creación de nuevas máquinas virtuales y el traslado de la información generada en años anteriores.

El resultado de la figura 56 presenta la réplica realizada inicialmente para mejorar la

**Figura 56.** Réplica de la síntesis física del circuito integrado "El Gran Jaguar".



Nota. En esta imagen se muestra la réplica realizada el grupo de trabajo del año 2025 del circuito integrado "El Gran Jaguar" con tecnología de 180 nm.

comprensión de la estructura y del flujo de trabajo trazado por los estudiantes en años anteriores. Esta réplica se realizó con la tecnología de 180 nm, por lo cual en el presente documento no se presenta el detalle de los resultados ni los procedimientos realizados por estudiantes de años anteriores [12] [15] [13].

**Figura 57.** Resultado DRC de réplica de El Gran Jaguar

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

--------------------------------------------------------------------------------
Results Summary
--------------------------------------------------------------------------------

Rule and DRC Error Summary

182 total rules were run.
217 rules NOT EXECUTED.
1 rule has violations.
There is 1 total violation.
Refer to circuit.LAYOUT_ERRORS

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Rule                          Violations Found
M2.R.1                        v = 1
```

Nota. La imagen muestra que el resultado DRC de la réplica en 180 nanómetros aún tenía errores.

En esta primera etapa no se realizó ningún cambio en los *scripts*, pero fue posible lograr la réplica y obtener los resultados esperados según lo documentado en los trabajos de graduación previos. Sin embargo, cabe resaltar que durante la fase de ejecución se identificaron algunos errores que se tomaron también como base para la implementación de los cambios en la transición a la tecnología de 65 nm. En la figura 58 se pueden observar los resultados de las pruebas DRC realizadas con los documentos de la réplica realizada del proyecto El Gran Jaguar. En la imagen es posible observar que hay solamente un error de densidad en la capa de metal 2. Si bien se obtuvo un resultado de un solo error, al revisar la documentación fue posible validar que en años anteriores se había modificado el *runset* DRC, lo cual llevó

a una reducción de errores. Sin embargo, este procedimiento no era el adecuado, ya que son documentos que el fabricante brinda para cumplir con los requisitos y por lo tanto no deben ser manipulados. Esto permitió comprender que a pesar de haber obtenido un solo error aún necesitaban corregirse partes puntuales del flujo para obtener las salidas correctas para la fabricación del diseño.

**Figura 58.** Error en la prueba DRC de réplica de El Gran Jaguar

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

--------------------------------------------------------------------------------
Results Summary
--------------------------------------------------------------------------------

Rule and DRC Error Summary

182 total rules were run.
217 rules NOT EXECUTED.
1 rule has violations.
There is 1 total violation.
Refer to circuit.LAYOUT_ERRORS

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Rule                          Violations Found
M2.R.1                        v = 1
```

Nota. La imagen muestra el error que se obtuvo en la prueba DRC del proyecto El Gran Jaguar en 180 nm.

# CAPÍTULO 18

## Glosario

***ALU* (*arithmetic logic unit*)** Es una unidad dentro de un procesador encargada de realizar operaciones aritméticas y lógicas sobre los datos..

**ASCII** Es un estándar de codificación de caracteres que representa texto en computadoras y dispositivos digitales. ASCII utiliza códigos numéricos para representar letras, números y símbolos comunes.

***Back-end*** Es la etapa final del diseño de circuitos integrados que incluye la síntesis física, el diseño del layout, la verificación y la generación de archivos para la fabricación del chip.

**Celda** En el contexto de nanoelectrónica, una celda se refiere a un bloque o unidad básica que se emplea para el diseño de circuitos integrados. Puede ser por ejemplo, una compuerta o circuitos básicos que sirven para la conexión de entradas y salidas.

**CMOS** Abreviación en inglés de *Complementary Metal-Oxide-Semiconductor*. Es una tecnología de fabricación de circuitos integrados que utiliza transistores MOSFET complementarios, es decir NMOS y PMOS, para implementar funciones lógicas y de memoria.

***Core*** En el contexto de un nanocircuito, se refiere a la región central del chip donde se ubican las celdas lógicas principales..

***Die*** Es la porción individual de silicio cortada de una oblea que contiene a un circuito..

**DRC (*design rule check*)** Es una verificación que se realiza en el diseño de circuitos integrados para asegurar que el diseño cumple con las reglas de diseño establecidas para que se pueda cumplir con el proceso de manufactura del circuito.

**ERC (*electrical rule check*)** Es una verificación que se realiza en el diseño de circuitos integrados para asegurar que las conexiones eléctricas cumplen con las reglas y especificaciones eléctricas definidas por el fabricante.

***Flip Chip*** Es una técnica de empaquetado donde el chip se monta boca abajo directamente sobre el sustrato mediante esferas de soldadura para la conexión del DIE..

***Floorplan*** Es una etapa del flujo de diseño en la que se define la disposición general de los bloques funcionales y las áreas de los componentes en el chip.

***Front-end*** Es la etapa inicial del diseño de circuitos integrados que incluye la especificación del diseño, la creación del diagrama esquemático y la síntesis lógica del circuito.

**GDS (*graphic data system*)** Se refiere a un formato estándar que se utiliza en la industria de semiconductores para describir la geometría final del diseño de un circuito integrado.

**IMEC (*Interuniversity Microelectronics Centre*)** Es un instituto de investigación y desarrollo de nanotecnología con sede en Bélgica. IMEC colabora con la industria y la academia para avanzar en tecnologías de semiconductores y sistemas electrónicos.

***Interconnects*** Son los metales que forman las conexiones físicas que permiten la comunicación entre diferentes componentes de un circuito integrado .

***Layout*** Es la representación física de un circuito, que muestra la disposición y conexión de los componentes electrónicos en el chip así como las capas de metal y otros materiales utilizados en la fabricación del mismo.

**LVS (*layout versus schematic*)** Es una prueba que se realiza en el diseño de circuitos integrados para asegurar que el diseño físico coincida con el diagrama eléctrico del mismo.

**MOSFET** Es un tipo de transistor que se utiliza en circuitos integrados. Su nombre significa Transistor de Efecto de Campo de Óxido Metálico Semiconductor.

**Máquina de estados** Es un modelo matemático utilizado para diseñar sistemas digitales. Consta de una cantidad finita de estados y transiciones entre ellos, que se activan en respuesta a entradas específicas.

***Netlist*** Es una representación textual de un circuito electrónico que describe los componentes y sus conexiones. Se utiliza en el diseño y simulación de circuitos integrados.

**NMOS (*N channel metal-oxide-semiconductor*)** Transistor MOSFET que utiliza portadores de carga tipo N para conducir la corriente.

***Pads*** Son las áreas metálicas del chip destinadas a la conexión eléctrica con el exterior, ya sea mediante *wirebonding* o *flip chip*..

**PMOS (*P channel metal-oxide-semiconductor*)** Es un tipo de transistor MOSFET que utiliza portadores de carga tipo P para conducir la corriente.

***Power-grids*** Son las redes de distribución de energía eléctrica dentro de un circuito integrado.

***RM*** Es la abreviación en inglés de *Reference Methodology*, que se refiere a un conjunto de *scripts* y comandos brindados por Synopsys para guiar en el diseño y la implementación de circuitos integrados utilizando sus herramientas y tecnologías..

***Routing*** Es el proceso de trazar las conexiones eléctricas entre los diferentes componentes de un circuito integrado.

***Runset*** Es un archivo de configuración que define reglas y parámetros de diseño y opciones que se deben verificar para el proceso de manufactura.

***Script*** Es un archivo de texto que contiene una serie de comandos que se ejecutan en un entorno específico. Los scripts se utilizan para automatizar tareas repetitivas o complejas.

***Sealring*** Es una estructura metálica que rodea el borde del chip para proteger las capas internas contra daños mecánicos, humedad o contaminación durante el corte y empaquetado..

**Synopsys** Es una empresa que proporciona herramientas de software y servicios para el diseño y verificación de circuitos integrados y sistemas electrónicos.

**Síntesis física** Es el proceso de transformar un diseño lógico en un diseño físico que puede ser implementado en un chip.

**TSMC (*Taiwan Semiconductor Manufacturing Company*)** Es una empresa taiwanesa que se especializa en la fabricación de semiconductores. Es uno de los mayores fabricantes de chips del mundo y ofrece servicios de fabricación para diversas empresas de tecnología a nivel mundial.

**Verilog** Es un lenguaje de descripción de hardware utilizado para modelar y diseñar sistemas digitales. Permite describir el comportamiento y la estructura de circuitos electrónicos de manera similar a un lenguaje de programación.

**VLSI** Abreviación en inglés de *Very Large Scale Integration*. Es una metodología de diseño de circuitos integrados que permite la integración de una gran cantidad de transistores en un solo chip.

**Vías** Son pequeños conductores verticales que conectan distintas capas metálicas dentro del chip..

***Wirebond*** Es un método de conexión eléctrica en el que se unen los pads del DIE al encapsulado mediante finos hilos metálicos, comúnmente de oro o aluminio..