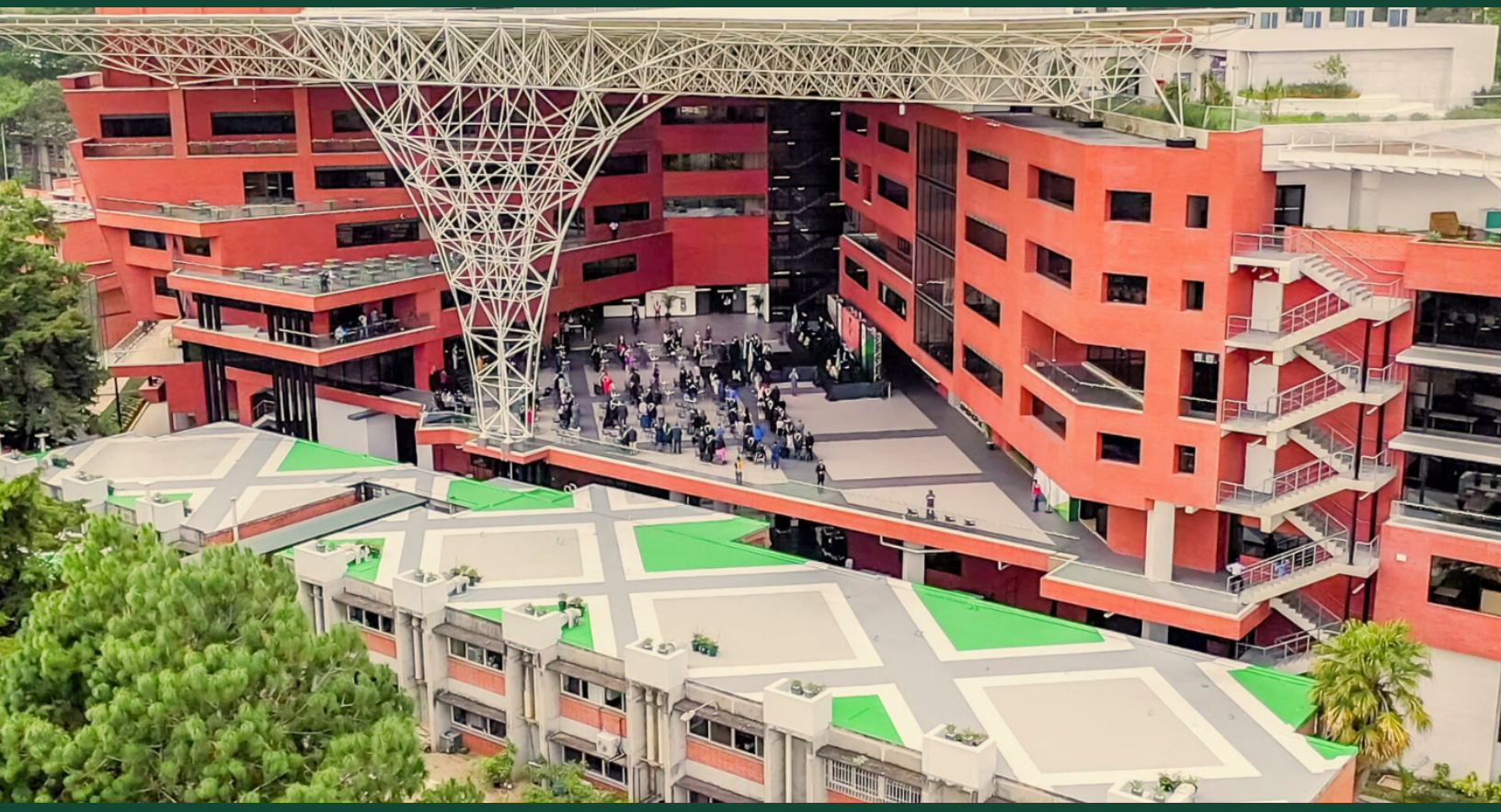
Nanochip El Gran Jaguar: síntesis física, LVS y pruebas finales

José Mario Méndez Rodríguez



UVG | UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Nanochip El Gran Jaguar: síntesis física, LVS y pruebas finales

Trabajo de graduación presentado por José Mario Méndez Rodríguez
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

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

Prefacio

Este trabajo representa la culminación de una etapa académica y personal que no habría sido posible sin el apoyo de muchas personas a quienes deseo expresar mi más profundo agradecimiento.

A mis padres, por su apoyo incondicional y por enseñarme siempre a buscar más; gracias por acompañarme en cada paso de mi vida y darme todos los recursos necesarios para alcanzar mis metas. Gracias a cada uno por su esfuerzo incansable y por priorizarnos siempre. Siempre van a ser un ejemplo para mí.

A María José Cordero, por estar junto a mí, no solo durante este trabajo de graduación, sino también en mi vida, recordándome con su amor y paciencia que nunca estoy solo en este camino, así como por inspirarme y motivarme cada día.

A mis hermanos y sus parejas, por enseñarme que la dedicación trae frutos y por ser un ejemplo constante que me inspira a dar siempre lo mejor de mí.

A la familia Cordero por apoyarme como una familia más.

A los ingenieros del departamento, por compartir generosamente sus conocimientos y permitirme aprender más cada día, enriqueciendo mi formación y haciendo posible la realización de este proyecto; de manera especial al MSc. Carlos Esquit, por transmitir su pasión por la fabricación de chips y el diseño VLSI, mostrándome nuevas oportunidades no solo para mi desarrollo personal, sino también para el futuro de Guatemala. Finalmente, al MSc. Jonathan de los Santos, por su guía y apoyo fundamentales en la elaboración de este trabajo de graduación, cuya orientación fue clave para alcanzar los objetivos propuestos.

Índice general

Índice de figuras

Índice de cuadros

Resumen

Este trabajo documenta la preparación para fabricar el nanochip El Gran Jaguar en tecnología de 65 nm mediante un flujo digital alineado con prácticas de Synopsys. El objetivo central es habilitar el *pre-floorplanning* de la síntesis física con Design Compiler NXT (DCNXT) y ejecutar la verificación *layout versus schematic* (LVS) con la *reference methodology* (RM) de IC Validator, generando paquetes reproducibles para su integración posterior en IC Compiler II (ICC2). La investigación se delimita al dominio digital del PDK de 65 nm, es decir, al uso de librerías de celdas estándar y reglas asociadas al núcleo lógico sin abarcar dispositivos analógicos o de alto voltaje, e integra la migración de artefactos desde 180 nm, la definición de celdas y *black boxes*, y la activación de un conjunto mínimo de reglas ERC para fortalecer la correspondencia electro-topológica.

En la metodología, se adoptó una síntesis topográfica en DCNXT para estimación de *die/core*, colocación preliminar de macros y pines, y preparación de restricciones físicas y temporales. Asimismo, se configuró el entorno RM de ICV para LVS/ERC con *decks*, equivalencias y plantillas `XML` trazables. El flujo se validó con casos base y jerárquicos antes de escalar al diseño principal. Por lo tanto, los resultados confirmaron la viabilidad del *pre-floorplanning* y la preparación de entregables reproducibles para ICC2, así como la obtención de bloques representativos con LVS = PASS y la identificación de violaciones `FLOATING.psub` en ERC, derivadas de celdas sin conexión a `VSS` durante la etapa física. Se concluyó que la replicación disciplinada del flujo en tecnología previa acelera la migración a 65 nm, disminuye el riesgo de integración y proporciona una base sólida para el perfeccionamiento del *sign-off* eléctrico en futuros proyectos.

Palabras clave: nanoelectrónica, síntesis física, LVS, Design Compiler NXT.

Abstract

This work documents the preparation for fabricating the El Gran Jaguar nanochip in 65 nm technology through a digital flow aligned with Synopsys practices. Its main objective is to enable *pre-floorplanning* for physical synthesis using Design Compiler NXT (DCNXT) and to perform *layout versus schematic* (LVS) verification with the *reference methodology* (RM) of IC Validator, generating reproducible packages for subsequent integration into IC Compiler II (ICC2). The research is limited to the digital domain of the 65 nm PDK, that is, to the use of standard cell libraries and rules associated with the logic core, without covering analog or high-voltage devices. It also includes the migration of artifacts from 180 nm, the definition of cells and *black boxes*, and the activation of a minimum set of ERC rules to strengthen electro-topological correspondence.

The methodology adopted topographical synthesis in DCNXT for *die/core* estimation, preliminary macro and pin placement, and the preparation of physical and timing constraints. Likewise, the ICV RM environment was configured for LVS/ERC using traceable *decks*, equivalences, and XML templates. The flow was validated with baseline and hierarchical cases before being scaled to the main design. The results confirmed the feasibility of *pre-floorplanning* and the preparation of reproducible deliverables for ICC2. They also demonstrated the successful generation of representative blocks with LVS = PASS and the identification of FLOATING.psub violations in ERC, caused by cells lacking connection to VSS during the physical stage. It was concluded that the disciplined replication of the flow in a previous technology accelerates migration to 65 nm, reduces integration risk, and provides a solid foundation for refining electrical *sign-off* in future projects.

Keywords: nanoelectronics, physical synthesis, LVS, Design Compiler NXT.

CAPÍTULO 1

Introducción

El diseño y la verificación física de circuitos integrados constituyen un pilar estratégico del desarrollo tecnológico contemporáneo. En Guatemala, el proyecto El Gran Jaguar articula un esfuerzo sostenido por establecer capacidades locales de diseño digital y verificación física con estándares industriales, siguiendo una ruta de aprendizaje que inicia en 180 nm y migra a 65 nm para alinear desempeño, densidad e integración con las prácticas vigentes del sector. Este trabajo se ubica en el campo de la nanoelectrónica digital y persigue consolidar un flujo reproducible que habilite la preparación para fabricación del *mini ASIC* El Gran Jaguar en 65 nm, integrando etapas de *pre-floorplanning* y verificación *layout versus schematic* (LVS), junto con reglas ERC para asegurar coherencia topológica y seguridad funcional.

La delimitación comprende la síntesis física en modo topográfico con DCNXT, la preparación de artefactos de entrada para IC Compiler II (ICC2) y la ejecución de LVS/ERC mediante la *reference methodology* de Synopsys en IC Validator, bajo las reglas del PDK de 65 nm, donde se analizaron las discrepancias eléctricas derivadas de conexiones incompletas hacia VSS detectadas en la etapa física. Se asume un alcance digital, centrado en la correspondencia entre *netlists* esquemáticos y extraídos, la definición de equivalencias, *black boxes* y parámetros eléctricos esenciales, y la organización del proyecto con trazabilidad de *runsets*, plantillas XML y registros de ejecución. Para la metodología se adoptó una estrategia incremental: validación con casos base y jerárquicos, integración de restricciones temporales y físicas, y consolidación de paquetes reproducibles para *floorplanning* y verificación.

Al final, se muestra que la replicación disciplinada del flujo en tecnología previa reduce la incertidumbre al migrar a 65 nm y acorta tiempos de integración; que el cierre de LVS depende de la preparación rigurosa de insumos y de la coordinación entre diseño, *layout* y verificación; y que la activación selectiva de reglas ERC permitió identificar violaciones FLOATING.psub asociadas a celdas sin conexión a VSS, evidenciando la necesidad de refinar la etapa de *place & route* para fortalecer la integridad eléctrica del *sign-off*.

CAPÍTULO 2

Antecedentes

El proyecto El Gran Jaguar se apoya en una sucesión de investigaciones que, desde 2014, han construido el flujo de diseño necesario para fabricar el primer nanochip universitario de Centroamérica. En primer lugar, [1] introdujo a la Universidad del Valle de Guatemala (UVG) en la nanoelectrónica al demostrar la viabilidad de un sumador/restador de 32 bits usando herramientas Synopsys y librerías educativas de dicha empresa.

Durante 2019 se abordó la fase inicial de síntesis lógica. Los trabajos de [2] y [3] documentaron paso a paso cómo transformar diseños HDL en *netlists* de compuertas lógicas con Design Vision, validando la equivalencia funcional con Formality y simulaciones en VCS. Sus guías de instalación y solución de errores sentaron la base para estudiantes posteriores.

En 2020 la atención se desplazó a la verificación física. Primero, [4] aplicó *design rule checking* para garantizar que el *layout* cumpliera las reglas del proceso. Luego, [5] implementó la verificación *layout versus schematic* (LVS), confirmando la correspondencia entre el *netlist* del *layout* y el del esquemático original. Complementariamente, [6] corrigió el anillo de E/S y ejecutó pruebas de antena y *electrical rule checking* (ERC), asegurando la fiabilidad eléctrica del chip.

Con el flujo de verificación consolidado, 2021 se centró en optimizar la síntesis lógica. [7] reconfiguró exhaustivamente los *scripts* de Design Vision y documentó la generación de *netlists* a nivel de transistores, garantizando que cada módulo cumpliera los requisitos de las etapas siguientes.

La fase 2022 validó la funcionalidad completa del diseño. [8] trasladó el *netlist* sintetizado a un FPGA Xilinx Genesys para comprobar su comportamiento en *hardware*, mientras que [9] procesó las señales de salida mediante un microcontrolador y una aplicación en Python que convertía los datos en audio, demostrando la operación integral del circuito.

Durante 2024 se inició la migración a la tecnología de 65 nm y se cerró el flujo de diseño con dos aportes complementarios. En primer lugar, [10] automatizó la síntesis física en IC Compiler II, IC Validator y VCS, afinó las reglas DRC y Antena, y validó el núcleo de varios

circuitos mediante simulaciones en HSPICE y WaveView. Su trabajo documenta paso a paso la generación de la síntesis física y deja *scripts* reproducibles para futuras iteraciones. Por su parte, [11] depuró y actualizó el flujo de verificación física: definió un procedimiento eficiente para las pruebas *layout versus schematic* (LVS) y *electrical rule checking* (ERC), además de incorporar la extracción de parásitos con StarRC. Validó la metodología en circuitos de complejidad creciente, desde una compuerta NOT hasta el macrodiseño El Gran Jaguar, y elaboró documentación que facilita la réplica y futura automatización del proceso.

En conjunto, ambas investigaciones consolidan la fase final del proyecto al garantizar, primero, la correcta implementación física con *scripts* automatizados y, segundo, la verificación y caracterización eléctrica del diseño en 180 nm, dejando un flujo robusto y completamente documentado para las siguientes generaciones de diseñadores.

CAPÍTULO 3

Justificación

En la actualidad, el diseño y la manufactura de *nanochips* constituyen un pilar estratégico para la transformación digital y la competitividad económica a escala global. La tendencia a la miniaturización, impulsada por la Ley de Moore [12] y por la demanda de dispositivos cada vez más potentes, exige integrar un número creciente de transistores en áreas cada vez más reducidas, lo que se traduce en circuitos más eficientes, rápidos y confiables.

Para Guatemala, invertir en investigación y desarrollo (I+D) de esta tecnología representa una oportunidad histórica. Proyecciones recientes del mercado mundial de semiconductores [13], [14] evidencian su peso económico y su carácter estratégico para sectores como la salud, las telecomunicaciones, la energía y la seguridad. Impulsar capacidades locales permitiría crear industrias de alto valor agregado, dinamizar la economía, generar empleos altamente calificados y promover la transferencia de conocimiento entre la academia y el sector productivo, además de posicionar al país como un actor emergente en la cadena global de suministro de semiconductores.

El proyecto El Gran Jaguar constituye el primer intento serio en Centroamérica de diseñar y prototipar un nanochip basado en tecnología de 65 nm. Este esfuerzo actúa como piedra angular para establecer capacidades locales de diseño físico, verificación *layout versus schematic* y caracterización; crea un documento replicable y sirve de referencia para futuros proyectos que busquen integrar a Guatemala en la cadena de valor mundial de semiconductores.

Adicionalmente, el desarrollo de este trabajo incorpora herramientas de última generación como Design Compiler NXT (DCNXT) y la *reference methodology* (RM) de Synopsys, que permiten aplicar estrategias de diseño avanzadas no exploradas en ejercicios académicos previos, tales como el *pre-floorplanning*, la agregación de *timing constraints* y la inclusión de *physical constraints*. La integración de estas etapas representa un avance significativo en la formación y práctica del diseño de circuitos integrados en Guatemala, ya que alinea el flujo metodológico del proyecto con los estándares industriales recomendados por Synopsys y utilizados en entornos de producción a nivel internacional.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Realizar el proceso de *pre-floorplanning* de síntesis física utilizando Design Compiler NXT y pruebas de LVS con la *reference methodology* de Synopsys en un circuito integrado diseñado con tecnología de 65 nm de TSMC, para asegurar su funcionamiento y dar inicio al proceso de fabricación.

4.2. Objetivos específicos

- Replicar el circuito integrado desarrollado en tecnología de 180 nm.
- Generar los archivos necesarios que servirán como entrada para el proceso de *pre-floorplanning* en ICC2.
- Generar las celdas estándar y las *black boxes* necesarias para la síntesis física y las pruebas en tecnología de 65 nm.
- Contribuir a la mejora de los resultados de la prueba física LVS mediante el uso de la *reference methodology* de Synopsys, con el fin de reducir al máximo los errores obtenidos en la salida de LVS.

CAPÍTULO 5

Alcance

El presente proyecto de trabajo de graduación abarca el diseño, validación y preparación para la fabricación física del circuito integrado El Gran Jaguar en tecnología de 65 nm, utilizando librerías proporcionadas por TSMC. El desarrollo se enmarca dentro del flujo completo de diseño digital con herramientas de Synopsys, y se extiende hasta la generación del archivo GDSII y la validación del diseño mediante simulaciones funcionales y verificaciones físicas.

5.1. Alcance técnico

El trabajo comprende las siguientes etapas técnicas:

- Configuración e instalación de las herramientas EDA requeridas, incluyendo Design Compiler NXT, IC Compiler II e IC Validator, así como la preparación de licencias, variables de entorno y estructura de proyecto conforme a las buenas prácticas de la *reference methodology* (RM) de Synopsys.
- Migración del diseño del circuito integrado El Gran Jaguar a tecnología de 65 nm, adaptando su descripción estructural y las restricciones físicas conforme a las reglas y límites del PDK seleccionado.
- Síntesis lógica y física con Design Compiler NXT en modo topográfico para generar el *pre-floorplan*: estimación de *die/core*, relación de aspecto, *targets* de utilización, precolocación de macros y pines, zonas de *keepout* y bloqueos preliminares de colocación.
- Preparación de artefactos para el *floorplan* en ICC2, incluyendo *netlist* a nivel de compuertas, restricciones físicas y el archivo inicial con *macro/pin placement* y directrices de bloqueo.
- Aplicación de verificaciones físicas, específicamente LVS (*layout versus schematic*) utilizando IC Validator con el flujo de la *reference methodology* (RM), para garantizar la correspondencia entre el esquemático y el *layout* del circuito.

5.2. Alcance en recursos

El desarrollo del proyecto contempla el uso exclusivo de herramientas provistas por la Universidad del Valle de Guatemala, dentro del entorno académico autorizado por la institución. El acceso a las librerías tecnológicas de TSMC en tecnología de 65 nm se realizará bajo convenio académico. Además, se utilizarán recursos computacionales locales (máquinas virtuales habilitadas con entorno Linux).

5.3. Alcance temporal

El proyecto se ejecutará en el período comprendido entre febrero y noviembre del año 2025. El cronograma incluye etapas de documentación preliminar, instalación del entorno de trabajo, replicación del diseño en la nueva tecnología, verificación funcional y física, validación de desempeño y generación del *layout* final. El proyecto concluye con la entrega de la documentación necesaria para el inicio del trámite de fabricación física del chip.

5.4. Limitaciones

El proyecto no contempla la caracterización eléctrica del circuito una vez fabricado, ni su integración en sistemas físicos o plataformas embebidas. Tampoco incluye pruebas *post-silicon* ni análisis de confiabilidad a largo plazo. El alcance concluye con la validación del diseño mediante la realización de las pruebas correspondientes.

CAPÍTULO 6

Marco teórico

6.1. Fundamentos del diseño de circuitos integrados

Un circuito integrado (IC) es un conjunto de dispositivos electrónicos, principalmente transistores, interconectados sobre un mismo chip de semiconductor, lo que permite implementar funciones complejas en un área reducida. La evolución de la tecnología ha permitido integrar millones e incluso miles de millones de transistores en un solo chip, aumentando enormemente la complejidad de los diseños. El diseño manual de circuitos a esta escala sería inviable; por ello, se emplean metodologías automatizadas y herramientas de automatización del diseño electrónico (EDA) para manejar la complejidad [15]. En lugar de diseñar cada transistor individualmente, se utilizan abstracciones más altas, como descripciones de hardware en VHDL o Verilog, y flujos de diseño sistemáticos. Los circuitos integrados pueden clasificarse según su grado de personalización. En un extremo están los circuitos estándar, por ejemplo, familias lógicas comerciales, memorias o microprocesadores genéricos, disponibles para cualquier diseñador; en el otro extremo se encuentran los circuitos específicos de aplicación, o ASIC, diseñados a medida para una aplicación particular [16]. El uso de ASIC permite integrar en un solo chip funciones que, de otro modo, requerirían múltiples circuitos estándar, logrando así sistemas más compactos, rápidos y de menor consumo, a costa de un esfuerzo de diseño mayor, pero justificado en aplicaciones de alto volumen o rendimiento crítico [17]. En la práctica moderna, el diseño de ICs, especialmente ASIC digitales, se basa en flujos de diseño bien establecidos y soportados por herramientas EDA, que aseguran que el diseño cumpla con los requerimientos de funcionalidad, rendimiento, potencia y área desde la concepción hasta la fabricación.

6.2. Flujo de diseño ASIC

El proceso de diseño de un ASIC típicamente se divide en etapas *front-end*, o diseño lógico, y *back-end*, o implementación física [18]. Cada etapa transforma la descripción del

circuito hacia niveles más concretos, verificando en cada paso que se mantenga la correctitud del diseño. En la fase de diseño lógico, se define la funcionalidad del circuito a nivel abstracto. Esto suele implicar la especificación del comportamiento mediante un lenguaje de descripción de hardware (HDL), como VHDL o Verilog, o mediante esquemáticos de alto nivel [19]. Una vez escrita la descripción RTL (*register transfer level*), se realiza verificación funcional por simulación para asegurarse de que el diseño cumple las especificaciones deseadas. Tras validar el comportamiento, se procede a la síntesis lógica. En esta etapa, un algoritmo de síntesis, por ejemplo, usando Design Compiler, traduce el código RTL en una *netlist* de puertas lógicas interconectadas, utilizando celdas lógicas de una librería estándar específica del proceso tecnológico [20]. Durante la síntesis, se aplican restricciones de diseño, por ejemplo, frecuencias de reloj objetivo, límites de área y requerimientos de temporización, para obtener un circuito que, idealmente, pueda operar a la velocidad deseada en el ASIC final. El resultado de la síntesis es una *gate-level netlist* optimizada según las métricas de tiempo, área y potencia. Esta *netlist* se valida con simulaciones lógicas, es decir, simulación *pre-layout*, para comprobar que la funcionalidad sigue siendo correcta después de la traducción a puertas [19]. Si la lógica es muy grande, antes de la implementación física puede ser necesario particionarla en bloques manejables, por módulo funcional o por región del chip, en un proceso conocido como partición de sistema, que facilita el diseño jerárquico [21].

En flujos modernos, la frontera entre síntesis lógica e implementación física se difumina mediante técnicas de síntesis físicamente consciente, o *physical-aware synthesis*. Una práctica destacada es el *pre-floorplanning*, que forma parte del proceso de síntesis física en herramientas como Design Compiler NXT. Durante esta fase, el diseñador define especificaciones físicas clave del chip, como el tamaño del área activa (*core*), el porcentaje de utilización, los márgenes de congestión y la ubicación preliminar de pines y bloques. Estas directrices se integran durante la síntesis para producir una *netlist* lógica ya optimizada bajo restricciones físicas realistas, junto con una guía física denominada *synthesis place guidance* (SPG), que será utilizada posteriormente por herramientas como IC Compiler II para mejorar la colocación inicial y reducir iteraciones [20], [22]. De este modo, el *pre-floorplanning* actúa como paso preparatorio para la implementación física.

La siguiente fase es la implementación física del circuito, comúnmente denominada *place & route*. Aquí, la *netlist* sintetizada se lleva al plano físico de silicio. Primero se realiza una planificación de piso (*floorplanning*), determinando la distribución general de bloques lógicos en el chip, las áreas de bloques IP, las *I/O pads* y la infraestructura de potencia. Luego, en colocación (*placement*), se asigna una posición específica en el *layout* a cada celda lógica de la *netlist*, buscando minimizar la congestión y las longitudes de interconexión. Seguidamente, se lleva a cabo la síntesis del árbol de reloj (CTS), insertando *buffers* y distribuidores de reloj para garantizar que la señal de reloj llegue con mínimas diferencias de fase, o *skew*, a todos los *flops*. Después, en la etapa de enrutamiento (*routing*), se traza el cableado físico que interconecta todas las celdas según la *netlist*, respetando las reglas de diseño del proceso. Completado el enrutamiento, se extraen las parasíticas, es decir, resistencias y capacitancias, de las interconexiones [23]. Con esta información, se realiza una verificación *post-layout*, que incluye simulación con retardos *back-annotated* y análisis de temporización estática, para comprobar que el circuito aún cumple con las especificaciones de tiempo y funcionalidad al considerar las cargas e interconexiones reales [24]. Si se detectan violaciones, por ejemplo, caminos críticos más lentos de lo permitido, se itera refinando el diseño: se pueden resintetizar ciertas partes con diferentes opciones, realizar optimizaciones de lugar y ruta o ajustar

las restricciones. De hecho, muchas de estas fases son iterativas y pueden requerir volver a etapas previas; por ejemplo, resultados de la implementación física, como tiempos de ruta, pueden obligar a refinar la síntesis lógica. Sin embargo, las herramientas modernas incorporan tecnologías para reducir la cantidad de iteraciones necesarias. Por ejemplo, la síntesis topográfica de Synopsys permite predecir con alta fidelidad el resultado físico durante la síntesis, de modo que la *netlist* sintetizada esté lo más cerca posible de cumplir los requisitos tras el *layout* [20]. En conjunto, el flujo ASIC abarca desde la concepción del circuito hasta la obtención de un diseño listo para fabricar, pasando por múltiples etapas de optimización y verificación que garantizan que parámetros críticos como el área, la potencia y el rendimiento cumplan con los objetivos establecidos [19].

6.3. Herramientas Synopsys

El proceso descrito se apoya en herramientas de EDA especializadas. Synopsys, como líder de la industria, provee un conjunto de herramientas integradas que abarcan las distintas fases del flujo de diseño ASIC. A continuación, se describen las principales herramientas de Synopsys empleadas en dicho flujo:

- VCS (Verilog Compiled Simulator): es la herramienta de verificación funcional, o simulación, de Synopsys. VCS compila el código HDL del diseño y del banco de pruebas, y simula su comportamiento, permitiendo detectar y depurar errores en la lógica antes de proceder a etapas de síntesis o implementación. Synopsys VCS se destaca por ser una solución de simulación de alto rendimiento y es la herramienta primaria de verificación utilizada por muchas compañías líderes en semiconductores [25]. VCS explota paralelismo a nivel fino en procesadores multinúcleo para acelerar simulaciones de larga duración o con alta actividad, con muy poca intervención del usuario [25]. También soporta características avanzadas como *testbench* nativo en C/C++, verificación orientada a restricciones, con generación aleatoria de casos de prueba, y coberturas funcionales, integrándose de forma nativa con el entorno de depuración Synopsys Verdi [25]. Con VCS, los diseñadores pueden verificar exhaustivamente el diseño en etapas tempranas, aumentando la probabilidad de éxito en silicio en el primer intento al detectar y corregir la mayoría de errores lógicos en el entorno de simulación.
- Design Compiler NXT: es la herramienta de síntesis lógica avanzada de Synopsys, sucesora directa de Design Compiler Graphical. Mantiene compatibilidad con *scripts* existentes y vistas físicas en formato MilkyWay, pero introduce mejoras significativas en rendimiento, calidad de resultados (*QoR*) y soporte para tecnologías avanzadas [20]. A través de su tecnología de síntesis topográfica, Design Compiler NXT realiza estimaciones precisas de temporización y parasíticos usando modelos físicos del *layout*, lo que permite predecir el comportamiento *post-route* con mayor fidelidad. Como parte de su enfoque de síntesis físicamente consciente, permite realizar el proceso de *pre-floorplanning* desde la misma herramienta de síntesis, definiendo parámetros físicos del diseño como tamaño del *core*, densidad objetivo y márgenes de colocación. Durante este proceso, se genera una guía de colocación denominada *synthesis place guidance* (SPG), que es consumida por herramientas de implementación física como IC Compiler II o Fusion Compiler [22]. Esta integración reduce iteraciones entre *front-end* y *back-*

end al alinear mejor la *netlist* con las restricciones físicas. Además, Design Compiler NXT incorpora optimizaciones como el mapeo orientado a potencia, la optimización concurrente de datos y reloj (*CCD*) y soporte nativo para nodos FinFET de 7 nm o menores, incluyendo reglas de accesibilidad de pines y litografía compleja [20]. En conjunto, ofrece una plataforma de síntesis física robusta y eficiente, clave para obtener *netlists* de alta calidad preparadas para implementación avanzada.

- IC Compiler II: es la herramienta de implementación física, o colocación y ruteo, de Synopsys, encargada de llevar la *netlist* lógica al diseño físico completo (GDSII). Se considera una solución líder en *place & route*, diseñada para afrontar las estrictas exigencias de los nodos tecnológicos avanzados en términos de rendimiento, potencia, área y tiempo de comercialización [23]. Incluye innovaciones para diseño plano y jerárquico, exploración temprana del diseño, colocación consciente de congestión, síntesis de reloj, enrutamiento avanzado y cierre de *signoff* dentro de la misma plataforma [23]. IC Compiler II está construida sobre una arquitectura altamente paralelizada, lo que le permite manejar diseños de gran tamaño de manera eficiente. Asimismo, emplea técnicas de optimización multiobjetivo durante la colocación global y optimizaciones concurrentes de señales de reloj y datos para mejorar el cumplimiento de temporización [23]. Además, soporta flujos específicos para las complejidades de tecnologías modernas, como FinFET y múltiples patrones de litografía, e incluso aplica aprendizaje automático para guiar optimizaciones y lograr un cierre de diseño más rápido y predecible [23]. En conjunto, IC Compiler II toma la *netlist* sintetizada e itera mediante colocación, ruteo y optimizaciones físicas hasta producir un *layout* libre de violaciones de diseño, lo más cercano posible a los objetivos de temporización. Es común que IC Compiler II se use en conjunto con IC Validator, a través del entorno Fusion Compiler, para realizar optimizaciones *in-design* basadas en análisis de temporización y verificaciones de reglas, acelerando la convergencia del diseño.

- IC Validator: Solución de verificación física de Synopsys, que abarca la verificación de reglas de diseño (DRC) y la comprobación *layout versus schematic* (LVS) en etapa de *signoff*. Es una herramienta de alto rendimiento y gran capacidad, escalable para correr en cientos o miles de núcleos de CPU en paralelo, lo cual le permite verificar los diseños más complejos, incluidos chips con miles de millones de transistores, en tiempos de ejecución razonables [26]. Esta herramienta se utiliza con los *runsets* oficiales de las fábricas (PDK) para validar que el *layout* final cumple todas las reglas geométricas y eléctricas requeridas para una fabricación correcta. Empresas líderes han adoptado IC Validator para *signoff* en nodos avanzados dada su capacidad y precisión. Una característica importante es su estrecha integración con el flujo de implementación de Synopsys, incluidas sus interacciones con IC Compiler II y Fusion Compiler [26]. Esto significa que problemas de verificación detectados, por ejemplo, violaciones DRC o errores de litografía, pueden corregirse automáticamente dentro del entorno de *place & route*, sin necesidad de largos ciclos de depuración manual [26]. Por ejemplo, IC Validator puede identificar *hotspots* de DRC durante la fase de *routing* y enviar guiado de corrección al enrutador para resolverlos antes de finalizar el diseño. Esta retroalimentación en tiempo real acelera el cierre de diseño y reduce iteraciones de *signoff* y verificación. En suma, IC Validator asegura que el diseño final esté libre de violaciones de reglas de fabricación y coincida con el circuito lógico previsto, sirviendo como paso final antes de la generación de máscaras.

6.4. Verificación final (DRC, LVS, ERC)

Antes de enviar el diseño a fabricación, es imprescindible realizar una verificación final, conocida como *physical verification*, que garantice que el *layout* cumple las reglas de fabricación y que representa el diseño lógico pretendido. Las principales comprobaciones en esta etapa son DRC, LVS y ERC, usualmente llevadas a cabo con herramientas de *signoff* como IC Validator de Synopsys u otras equivalentes. Estas verificaciones constituyen la última barrera de control antes del *tape-out*, ya que permiten detectar errores geométricos, eléctricos y de conectividad que podrían comprometer la manufactura o el funcionamiento correcto del circuito.

La verificación de reglas de diseño, o DRC (*design rule check*), consiste en revisar todas las geometrías del *layout* contra un conjunto de reglas proporcionadas por la fábrica, o *foundry*. Estas reglas definen restricciones como anchos mínimos de polisilicio y metal, espacios mínimos entre componentes, densidad de capas y reglas de enclavamiento entre diferentes capas, entre muchas otras. Un verificador DRC recorre todo el chip para detectar cualquier violación a estas reglas. El cumplimiento de dichas reglas asegura que variaciones normales del proceso de fabricación no resulten en defectos fatales en el chip [27]. Por ejemplo, una regla puede dictar que dos trazos metálicos paralelos tengan al menos cierta distancia para evitar cortocircuitos por problemas de fotolitografía. Si el diseño viola alguna regla, marcada como error DRC, debe corregirse editando el *layout* hasta que el DRC quede limpio. Un diseño *DRC clean* es condición necesaria para que la fábrica acepte producir las máscaras del chip.

La verificación *layout versus schematic* (LVS) es un proceso de comparación eléctrica entre el *layout* y el esquemático o *netlist* original. El objetivo de LVS es confirmar que el circuito implementado físicamente en el *layout* coincide exactamente con la intención de diseño capturada en la *netlist* generada tras síntesis. Para ello, se extrae del *layout* una *netlist*, identificando transistores, sus conexiones y parasíticas básicas, y se compara contra la *netlist* de referencia. La herramienta LVS empareja cada dispositivo en el esquemático con su correspondiente en el *layout*, verificando que cada nodo lógico conecte exactamente a los mismos dispositivos en ambas representaciones [28]. Si hay discrepancias, por ejemplo, un cable conectado a un pin equivocado, o una puerta faltante o sobrante en el *layout*, se reportan errores LVS. Un diseño que pase LVS, es decir, *LVS clean*, implica que no hay conexiones faltantes ni extras y que el *layout* es una implementación fiel del circuito diseñado.

La verificación de reglas eléctricas, o ERC (*electrical rule check*), complementa al DRC verificando condiciones eléctricas del circuito que no pueden expresarse únicamente mediante reglas geométricas. Mientras DRC y LVS operan principalmente sobre geometría y conectividad, ERC analiza aspectos como conexiones correctas de potencia, límites de carga y *fan-out*, y otras condiciones de diseño eléctrico. Por ejemplo, ERC verifica que todos los transistores tengan pozos y sustratos correctamente conectados a fuentes de potencia o tierra, evitando transistores flotantes; también revisa que no existan entradas desconectadas, o *floating nodes*, ni salidas lógicas unidas directamente entre sí, como dos *drivers* en corto [29]. Además, puede incluir verificaciones como tiempos máximos de transición, o *slew*, capacitancias de carga bajo ciertos umbrales y reglas orientadas a prevenir problemas de confiabilidad, como posibles exposiciones a ESD o corrientes excesivas en nodos sensibles [29]. Un diseño que pase ERC se considera libre de configuraciones potencialmente peligrosas desde el punto de

vista eléctrico.

En conjunto, DRC, LVS y ERC constituyen la verificación final obligatoria de cualquier ASIC antes del *tape-out*. Si el diseño es *DRC/LVS/ERC clean*, se asume que es fabricable y que implementa correctamente el circuito deseado. Aun después de estas verificaciones, pueden realizarse simulaciones *post-layout* con extracción de parasíticos (PEX) para validar temporización y funcionalidad con mayor precisión. Solo tras superar todas estas comprobaciones, el diseño entra en la fase de cierre final y preparación de datos para fabricación.

6.5. Celdas estándar

Las celdas estándar, o *standard cells*, son bloques lógicos predefinidos y caracterizados que conforman la base de los diseños digitales en VLSI. Estas celdas incluyen funciones lógicas comunes como compuertas NAND, NOR, inversores, *flip-flops*, *buffers*, entre otras, y se disponen en una grilla regular para facilitar su integración automática en etapas de síntesis y colocación y ruteo. Cada celda estándar posee distintas vistas:

- Lógica o funcional, por ejemplo, en formato Verilog.
- De temporización, por ejemplo, en formato Liberty `.lib`.
- Física, usualmente en archivos `.lef` o `.gds`.
- Eléctrica, por ejemplo, para extracción de parasíticos en formato `.spef`.

Estas vistas permiten que herramientas como Design Compiler e IC Compiler II de Synopsys automaticen el flujo RTL-to-GDSII, utilizando librerías que contienen estas celdas caracterizadas. Por ejemplo, Design Compiler utiliza la vista `.lib` para estimar retardo y potencia durante la síntesis, mientras que IC Compiler II se apoya en la vista `.lef` para la colocación y el ruteo [20], [22].

Las celdas estándar son caracterizadas a través de simulaciones SPICE que permiten generar modelos precisos bajo diversas condiciones de voltaje, temperatura y carga, resultando en librerías *multi-corner* que favorecen un cierre de diseño robusto [15], [16]. La disposición regular de estas celdas permite la automatización de tareas clave como *placement*, *clock tree synthesis* (CTS), inserción de *fillers* y verificaciones DRC. Las herramientas modernas de *signoff* extraen y validan las vistas físicas y eléctricas a partir de estas celdas con precisión, permitiendo asegurar el cumplimiento de especificaciones [23], [24].

6.6. Black boxes

Una caja negra, o *black box*, es una abstracción utilizada en el flujo de diseño de circuitos integrados para representar un bloque cuyo comportamiento interno no se detalla, pero cuya interfaz, es decir, sus pines de entrada y salida, es conocida. Las *black boxes* se usan comúnmente cuando no se dispone de la *netlist* interna de un módulo, pero es necesario referenciarlo para simulación funcional, verificación o integración [15], [21].

En verificación *layout versus schematic* (LVS), por ejemplo, se utilizan definiciones de tipo `.SUBCKT` en formato SPICE o CDL que incluyen solamente los pines, sin ninguna descripción de transistores. Esto permite a la herramienta de LVS verificar que las conexiones entre bloques sean correctas sin requerir la *netlist* interna de cada celda.

Las herramientas de síntesis y verificación de Synopsys permiten declarar celdas como *black box* mediante directivas específicas, de forma que no se intenten expandir durante el procesamiento. En flujos jerárquicos, las *black boxes* permiten modularidad y reducción de complejidad durante las verificaciones, y son esenciales cuando se trabaja con bloques de IP preexistentes, celdas estándar o macros analógicos caracterizados por terceros [10], [11], [28].

6.7. Cierre de diseño

El cierre de diseño, o *design closure*, es la etapa final del flujo en la cual se busca resolver cualquier discrepancia remanente y asegurar que el diseño cumple integralmente con todos los objetivos y restricciones antes de ser liberado a fabricación. Implica lograr simultáneamente el cumplimiento de las restricciones de temporización, sin *paths* que violen tiempos de *setup/hold*, las restricciones físicas, sin violaciones DRC, LVS o ERC, y las metas de consumo de potencia y área. El cierre de diseño requiere regularmente iteraciones finales de optimización y corrección de errores hasta que no queden violaciones abiertas.

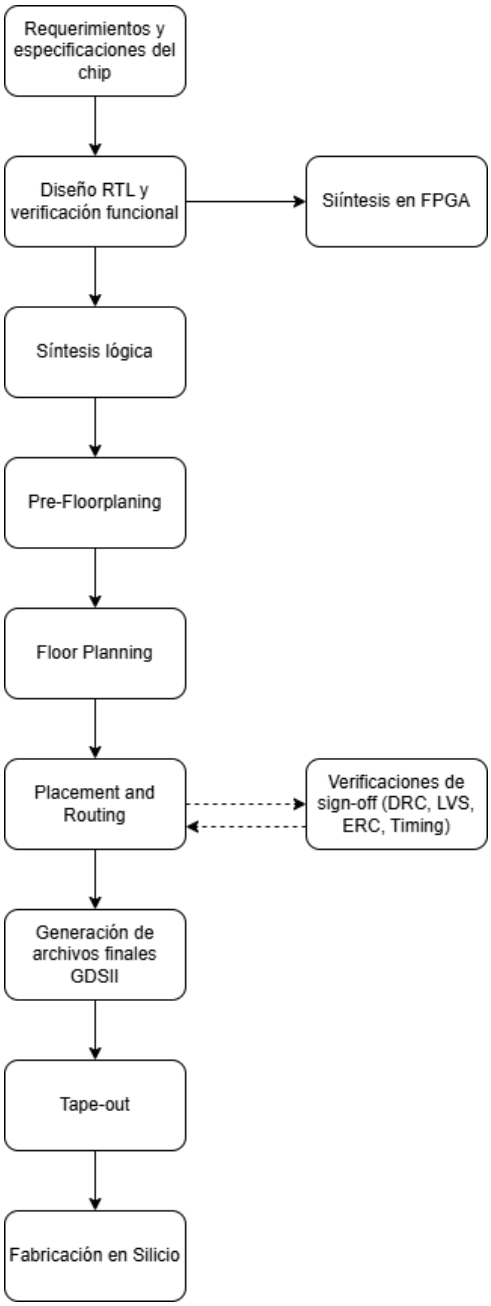
Un enfoque común en esta etapa consiste en iterar entre análisis de temporización y ajustes localizados del diseño, conocidos como *engineering change orders* (ECOs). Por ejemplo, si tras el enrutamiento se detectan unos pocos caminos críticos fuera de tiempo, se pueden realizar ECOs para reemplazar ciertas puertas por versiones más rápidas, agregar *buffers* para mejorar los *slews*, o modificar ligeramente el ruteo para reducir capacitancias, todo ello buscando eliminar las violaciones de temporización sin impactar negativamente otras partes del diseño [30]. Cada cambio debe validarse. Del mismo modo, si existe alguna violación DRC de último momento, por ejemplo, por ajustes manuales o efectos de composición de bloques, esta se corrige en la edición final y se vuelve a ejecutar la verificación física para asegurar que el diseño quede limpio.

En los flujos actuales, el cierre de temporización, señal y energía suele realizarse de forma holística. Las herramientas de *place & route*, como IC Compiler II, integran verificación física con apoyo de IC Validator durante la implementación, lo que permite ir solucionando violaciones en el proceso y llegar al cierre de diseño con mayor rapidez [23], [26]. No obstante, el ingeniero de diseño suele reservar un período de cierre en el que realiza comprobaciones independientes de *signoff* para tener absoluta confianza en que el diseño cumple las especificaciones en todas las condiciones.

Una vez obtenido un diseño libre de errores de verificación y con todas las señales dentro del presupuesto de tiempo y potencia, se considera alcanzado el cierre de diseño. En este punto se genera el *tape-out*, es decir, los datos finales, como GDSII u OASIS, *netlists* e información de *timing*, que se entregarán a la fundición para la fabricación de las máscaras. El éxito en el cierre de diseño asegura que el ASIC tendrá alta probabilidad de funcionar correctamente desde el primer silicio, cumpliendo con los requisitos para los cuales fue concebido

[19].

Figura 1. Flujo teórico completo de diseño ASIC. Desde RTL hasta verificación de *sign-off* y *tape-out*.



Nota. El diagrama sintetiza las etapas principales del flujo ASIC, desde la descripción RTL hasta *tape-out*.

CAPÍTULO 7

Implementación del *pre-floorplanning* en 65 nm

7.1. *Pre-floorplanning* utilizando DCNXT

La síntesis física constituye una de las etapas más críticas en el flujo de diseño de circuitos integrados (ASIC), pues traduce la descripción RTL en una representación estructural optimizada, considerando tanto los requisitos lógicos como las restricciones físicas y temporales del proceso tecnológico. En esta fase se empleó Synopsys Design Compiler NXT en modo *topographical*, el cual permite predecir con alta precisión el área, la potencia y el retardo del diseño al integrar información física proveniente de las bibliotecas tecnológicas y modelos de interconexión (*TLU+*).

El flujo aplicado comprende la lectura y vinculación de bibliotecas (`.db` y `.ndm`), la verificación estructural del diseño, la definición de restricciones temporales (*timing constraints*) y la configuración de los parámetros físicos del chip mediante el archivo `nanochip.pcon`. Posteriormente, se realizó la optimización global del diseño mediante el comando `compile_ultra`, habilitando opciones avanzadas como *retiming*, *scan insertion* y *power optimization*.

A través de esta etapa se buscó alcanzar un equilibrio entre desempeño, área y consumo energético, evaluando los resultados mediante reportes de área, potencia, temporización y restricciones físicas. Los resultados obtenidos permitieron analizar el comportamiento del flujo topográfico y validar la coherencia entre la síntesis lógica y las condiciones físicas del diseño, previo a su implementación en herramientas de *place & route*.

7.1.1. Archivos y recursos previos necesarios

Antes de ejecutar la síntesis física en Synopsys Design Compiler NXT, es indispensable disponer de los archivos tecnológicos, lógicos y de restricción que permiten configurar

correctamente el entorno de trabajo. Estos archivos establecen los parámetros eléctricos, geométricos y funcionales que el sintetizador utilizará durante el proceso.

Archivos tecnológicos

Los archivos tecnológicos proporcionan la información física y eléctrica de las celdas estándar del proceso. Para el diseño `nanochip`, se emplearon los archivos correspondientes a la tecnología de TSMC 65 nm, los cuales deben descargarse o solicitarse al proveedor académico:

- `.tf`. Define las capas tecnológicas, las reglas de diseño y los parámetros eléctricos básicos.
- `.tluplus`. Contiene los modelos de resistencia y capacitancia de interconexión, necesarios para la estimación de *delay* y *wire load*.
- `.lef`. Proporciona la información geométrica de las celdas estándar, como altura, anchura y pines de conexión.
- `.ndm`. Define la biblioteca física utilizada por Design Compiler NXT en modo topográfico.

Archivos lógicos

Los archivos lógicos describen el comportamiento y las características funcionales de las celdas a nivel lógico:

- `.db`. Biblioteca de celdas estándar con información de área, potencia y retardo, utilizada para la síntesis lógica.
- `.v`. Archivo RTL del diseño (`nanochip.v`), que describe el comportamiento funcional en Verilog.

Archivos de configuración y restricción

El flujo requiere además un conjunto de archivos TCL y SDC que definen las condiciones de síntesis y las restricciones físicas:

- `setup.tcl`. Archivo principal de configuración; define las rutas de búsqueda, las bibliotecas y la librería física activa.
- `nanochip.con`. Contiene las restricciones temporales del diseño, como el período de reloj, la incertidumbre y los retardos de entrada y salida.
- `nanochip.pcon`. Define las restricciones físicas, incluyendo el tamaño del *die*, el área de núcleo (*core*), el aspecto y la densidad de colocación.
- `nanochip.svf`. Archivo de registro para los cambios incrementales entre compilaciones.

7.1.2. Explicación del archivo `setup.tcl`

El archivo `setup.tcl` constituye el punto de partida del flujo de síntesis física, pues configura las variables globales del entorno de Design Compiler NXT y vincula las bibliotecas necesarias para el análisis y la compilación del diseño. Además, permite verificar de forma inmediata las rutas de búsqueda y las librerías activas antes de proceder con la lectura del código RTL.

A continuación, se muestra el contenido del *script* utilizado en el proyecto:

Cuadro 1. Archivo de configuración del entorno (`setup.tcl`).

```
source rm_setup/dc_setup.tcl

set_app_var alib_library_analysis_path ./   ;# Common ALIB library location
define_design_lib WORK -path ./work         ;# Location of "analyze"d files

# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#  Verify Settings
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

echo "\n=========================================================================="
echo "\nLibrary Settings:"
echo "search_path:              $search_path"
echo "link_library:             $link_library"
echo "target_library:           $target_library"
echo "physical libraries:       $NDM_REFERENCE_LIBS"
echo "physical design library:  $NDM_DESIGN_LIB"
echo "Universidad del Valle de Guatemala || $designer || $company"
echo "\n=========================================================================="
```

Nota. El cuadro presenta el script `setup.tcl` empleado para configurar el entorno de trabajo y verificar las bibliotecas necesarias antes de iniciar la síntesis.

El *script* inicia con la instrucción `source`, que importa el archivo `dc_setup.tcl` del entorno. En dicho archivo se definen las variables fundamentales del flujo, tales como `search_path`, `link_library`, `target_library` y los parámetros de las bibliotecas físicas `NDM_REFERENCE_LIBS`.

Posteriormente, el comando `set_app_var alib_library_analysis_path` establece la ruta local donde se almacenan los archivos intermedios generados durante el análisis de librerías, o *ALIBs*, optimizando los tiempos de compilación en ejecuciones posteriores.

La instrucción `define_design_lib WORK -path ./work` crea la librería de trabajo `WORK`, donde se guardan los archivos analizados del diseño. Este directorio actúa como repositorio temporal de las representaciones intermedias utilizadas en la etapa de análisis y vinculación.

El bloque de comentarios introduce la sección de verificación de configuración. Las múltiples sentencias `echo` imprimen en la consola el resumen de las variables principales del entorno, confirmando la correcta asignación de rutas y librerías antes de iniciar la síntesis. Entre ellas destacan:

- `search_path`: lista de directorios donde el compilador buscará los archivos HDL y las bibliotecas asociadas.

- `link_library`: librerías utilizadas para resolver referencias cruzadas durante el proceso de vinculación.
- `target_library`: conjunto de celdas estándar elegidas como destino de mapeo lógico.
- `NDM_REFERENCE_LIBS` y `NDM_DESIGN_LIB`: bibliotecas físicas (`.ndm`) que habilitan el modo topográfico y la estimación física del diseño.

Finalmente, la línea `echo "Universidad del Valle de Guatemala || $designer || $company"` agrega una marca de identificación al registro de ejecución, útil para la trazabilidad en reportes y bitácoras.

Este archivo garantiza que el entorno de Design Compiler NXT se encuentre completamente definido antes de la lectura del RTL, constituyendo una práctica estándar en flujos profesionales de síntesis topográfica.

7.1.3. Explicación del archivo `nanochip.con`

El archivo `nanochip.con` define las restricciones temporales del diseño, las cuales permiten que el proceso de síntesis física optimice la lógica cumpliendo los tiempos de propagación y las condiciones de operación establecidas. Este archivo, basado en el formato Synopsys Design Constraints (SDC), especifica el comportamiento temporal del reloj, los márgenes de incertidumbre, los retardos de entrada y salida, y las cargas de salida asociadas a los puertos del circuito.

A continuación, se muestra el contenido del archivo de restricciones temporales utilizado en el proyecto:

Cuadro 2. Archivo de restricciones temporales (`nanochip.con`).

```
# Constraints file for design nanochip
remove_sdc
create_clock -period 1.2 [get_ports clk]
set_clock_uncertainty -setup 0.1 [get_clocks clk]
set_clock_transition -max 0.05 [get_clocks clk]

# Using default "Operating Conditions"
# from the "slow corner" library: ss0p75v125c
# Process :     0.99
# Temperature : 125.00
# Voltage :     0.75

set_input_delay -clock clk -max 0.2 [get_ports "select reset_ring_osc reset_nanochip"]
remove_input_delay [get_ports clk]

set_driving_cell -max -no_design_rule -lib_cell BUFFD0 [all_inputs]
remove_driving_cell [get_ports clk]

set_output_delay -clock clk -max 0.2 [all_outputs]
set_load -max 5 [all_outputs]
```

Nota. El cuadro presenta el archivo `nanochip.con`, en el cual se definen las restricciones temporales del diseño.

El archivo inicia con el comando `remove_sdc`, que limpia las restricciones previas almacenadas en el entorno del compilador para evitar conflictos con ejecuciones anteriores.

La instrucción `create_clock -period 1.2 [get_ports clk]` define el reloj principal del diseño con un período de 1.2 ns, lo que corresponde a una frecuencia aproximada de 833 MHz. Este reloj actúa como referencia para la temporización de todos los elementos secuenciales del circuito.

Posteriormente, `set_clock_uncertainty -setup 0.1 [get_clocks clk]` establece una incertidumbre de 0.1 ns en los análisis de *setup*, reflejando variaciones posibles en la distribución del reloj o en el retardo del enrutamiento. La instrucción `set_clock_transition -max 0.05 [get_clocks clk]` limita la transición máxima del reloj a 0.05 ns, lo que favorece una estimación realista de carga y consumo dinámico.

Las líneas comentadas describen las condiciones de operación del análisis, denominadas *slow corner*, o esquina lenta, de la biblioteca utilizada. Esta corresponde al punto de peor desempeño, o *worst case*, bajo las siguientes condiciones: proceso 0.99, temperatura de 125 °C y voltaje de 0.75 V. Estas condiciones aseguran que el diseño cumpla los tiempos incluso bajo variaciones adversas de fabricación o temperatura.

El bloque `set_input_delay` especifica el retardo máximo de entrada de 0.2 ns relativo al reloj para los puertos `select`, `reset_ring_osc` y `reset_nanochip`, mientras que `remove_input_delay [get_ports clk]` excluye al puerto del reloj de dichas restricciones.

A continuación, el comando `set_driving_cell -max -no_design_rule -lib_cell BUFFD0 [all_inputs]` define la celda `BUFFD0` como la fuente que impulsa las señales de entrada, asignando características eléctricas de manejo realistas para el cálculo de cargas. La celda asociada al reloj se elimina mediante `remove_driving_cell [get_ports clk]`, ya que su estímulo es generado internamente por la herramienta.

Finalmente, las instrucciones `set_output_delay -clock clk -max 0.2 [all_outputs]` y `set_load -max 5 [all_outputs]` definen, respectivamente, el retardo máximo de salida y la carga equivalente aplicada a los puertos de salida. Estas restricciones permiten al compilador dimensionar las celdas de salida y ajustar la lógica para cumplir los tiempos de propagación esperados en las interfaces externas.

En conjunto, el archivo `nanochip.con` garantiza que el proceso de síntesis física considere los márgenes de seguridad necesarios para un diseño robusto, estable y coherente con las condiciones reales de operación del circuito integrado.

7.1.4. Explicación del archivo `nanochip.pcon`

El archivo `nanochip.pcon` define las restricciones físicas del diseño, determinando la geometría del chip, el área destinada al núcleo y las condiciones de colocación utilizadas por Design Compiler NXT durante la síntesis topográfica. Estas restricciones permiten estimar la distribución espacial del circuito antes de pasar a la etapa de colocación y ruteo en una herramienta de diseño físico, o *place & route*.

A continuación, se muestra el contenido del archivo utilizado en el proyecto:

Cuadro 3. Archivo de restricciones físicas (`nanochip.pcon`).

```
# Define el die y core
create_die_area   -coordinate {0 0 1000 1000}
create_core_area  -coordinate {100 100 900 900}

# Ajusta la densidad de colocacion
set_utilization 0.40

# Controla la forma del core
set_aspect_ratio 1.0

# Forzar menor densidad local
set_app_var placer_max_cell_density_threshold 0.55

set_ideal_network [get_ports clk]
set_false_path -from [get_ports {reset_nanochip reset_ring_osc}]
set_ideal_network [get_ports {reset_nanochip reset_ring_osc}]
```

Nota. El cuadro presenta el archivo `nanochip.pcon`, en el cual se definen el tamaño del *die*, el área del núcleo, la densidad de colocación y las restricciones físicas y temporales iniciales del diseño.

El archivo inicia con la definición del área total del chip mediante el comando `create_die_area -coordinate {0 0 1000 1000}`, que establece los límites del *die* en coordenadas micrométricas. Esto genera un área cuadrada de 1000×1000 µm, lo que representa los bordes físicos del circuito integrado dentro del sustrato.

La instrucción `create_core_area -coordinate {100 100 900 900}` define el área del *core*, o núcleo activo, delimitando la zona donde se colocarán las celdas estándar y la lógica sintetizada. En este caso, el núcleo se ubica dentro del *die* con un margen de 100 µm en cada lado, esto genera un espacio perimetral reservado para *pads*, pines de entrada y salida, y celdas de alimentación.

El comando `set_utilization 0.40` establece una utilización global del 40 % dentro del núcleo, lo que significa que el área ocupada por las celdas no excederá ese porcentaje. Este parámetro controla la densidad general del diseño y permite dejar espacio libre para el enrutamiento posterior.

La instrucción `set_aspect_ratio 1.0` fuerza una relación de aspecto cuadrada entre el ancho y el alto del núcleo, lo que garantiza una distribución equilibrada de las celdas en ambas direcciones. Para evitar concentraciones excesivas de lógica en regiones específicas, el parámetro `placer_max_cell_density_threshold` se fija en 0.55 mediante la instrucción `set_app_var`, limitando la densidad local máxima durante la fase de estimación topográfica.

Las últimas líneas del *script* corresponden a restricciones de conectividad y excepciones de temporización. El comando `set_ideal_network [get_ports clk]` designa la red del reloj principal como ideal, lo que indica que no se considerará su retardo dentro del análisis físico, ya que la etapa de *clock tree synthesis* (CTS) se ejecuta posteriormente dentro del flujo físico. La sentencia `set_false_path -from [get_ports {reset_nanochip reset_ring_osc}]` excluye del análisis de temporización los caminos asociados a las señales de reinicio, mientras que `set_ideal_network [get_ports {reset_nanochip reset_ring_osc}]` marca también dichas redes como ideales.

7.1.5. Explicación del archivo `nanochip.svf`

El archivo `nanochip.svf`, o *Synopsys verification file*, cumple la función de registrar los cambios realizados durante el proceso de síntesis. A diferencia de los demás archivos del flujo, este no contiene comandos ejecutables ni código legible, sino que almacena información interna generada automáticamente por Design Compiler NXT en formato binario o comprimido. Durante la ejecución del comando:

Cuadro 4. Comando para activar el registro incremental de cambios (`set_svf`).

```
1 set_svf nanochip.svf
```

Nota. El cuadro presenta el comando utilizado para activar la generación del archivo `nanochip.svf`, en el cual se registra la secuencia de cambios aplicada durante la síntesis.

el compilador crea el archivo `nanochip.svf` en el directorio de trabajo y comienza a registrar la secuencia de operaciones aplicadas sobre el diseño, tales como optimizaciones lógicas, modificaciones de jerarquía, reemplazo de celdas, inserción de compuertas de *scan* o ajustes de temporización.

Este registro incremental permite que el flujo de síntesis sea reproducible y facilita la depuración de errores. Además, el archivo puede reutilizarse en etapas posteriores del flujo de implementación, especialmente en procesos de verificación formal o equivalencia lógica mediante herramientas como Formality, que comparan la *netlist* sintetizada con el código RTL original para asegurar que ambas representaciones sean funcionalmente idénticas.

En el contexto del proyecto, el archivo `nanochip.svf` garantiza la trazabilidad completa de la síntesis del diseño `nanochip`, permitiendo validar los resultados obtenidos, replicar ejecuciones previas y mantener consistencia entre las versiones del flujo topográfico.

7.2. Corrida del flujo de síntesis

Para ejecutar el flujo de síntesis, se abre una terminal en la carpeta raíz del proyecto, donde se encuentran los archivos `setup.tcl`, `nanochip.con`, `nanochip.pcon` y el código RTL del diseño.

Luego, se inicia la interfaz de Design Compiler NXT en modo topográfico con el siguiente comando:

Cuadro 5. Invocación del entorno de síntesis topográfica.

```
1 dcnxt_shell -topo -gui
```

Nota. El cuadro presenta el comando utilizado para abrir Design Compiler NXT en modo topográfico y con interfaz gráfica.

La opción `-topo` habilita el modo topográfico, que permite considerar estimaciones físicas de área y capacitancia durante la síntesis. El parámetro `-gui` abre la interfaz gráfica, facilitando la observación de jerarquías, reportes y mensajes de consola en tiempo real.

Dentro de la interfaz gráfica se ejecuta el *script* principal `dc.tcl`, mostrado a continuación, el cual contiene la secuencia completa del flujo de síntesis topográfica:

Cuadro 6. Script principal de ejecución del flujo (`dc.tcl`).

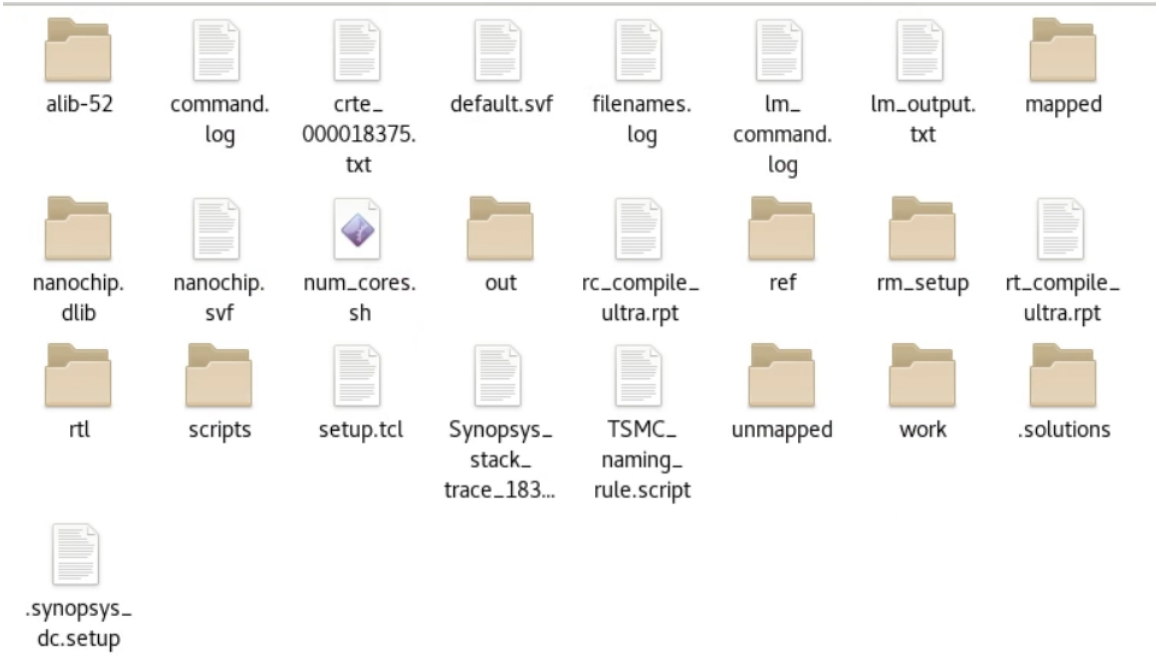
```
source setup.tcl
set_svf nanochip.svf

analyze   -format verilog nanochip.v
elaborate nanochip
link
check_design

source -echo nanochip.con
check_timing

source -echo nanochip.pcon

set_clock_gating_style -positive_edge_logic {integrated} -negative_edge_logic
    {integrated} \
                        -control_point before -max_fanout 64
set_leakage_optimization true

report_clock
group_path -name clk -critical 0.21 -weight 5
group_path -name INPUTS  -from [all_inputs]
group_path -name OUTPUTS -to [all_outputs]
group_path -name COMBO   -from [all_inputs] -to [all_outputs]

set_ungroup [get_designs "nanochip"] false
set_optimize_registers true -design nanochip
set_cost_priority -delay

report_physical_constraints
report_path_group
get_attribute [get_designs "nanochip"] ungroup
get_attribute [get_designs "nanochip"] optimize_registers
get_attribute [get_designs "nanochip"] cost_priority

write_file -f ddc -hier -out unmapped/nanochip.ddc
set_host_options -max_cores 16
report_host_options

compile_ultra -spg -retime -scan

list_licenses
report_hierarchy -noleaf

redirect -tee -file rc_compile_ultra.rpt {report_constraint -all}
redirect -tee -file rt_compile_ultra.rpt {report_timing}

write_file -f ddc -hier -out mapped/nanochip.ddc
set_svf -off

get_cells -hier *r_REG*_S*
report_cell -nosplit I_MIDDLE/I_PIPELINE
get_cells -hier *z2_reg*
get_cells -hier R_*
report_cell -nosplit I_IN
```

```
get_cells I_IN/*_reg*
report_physical_constraints

exit
```

Nota. El cuadro presenta el *script* principal del flujo de síntesis topográfica, incluyendo lectura de archivos, aplicación de restricciones, optimización y generación de reportes y salidas del diseño.

Una vez ejecutado el *script*, Design Compiler NXT procesa el diseño, aplica las restricciones y genera los reportes de síntesis dentro del directorio de trabajo. La Figura 2 muestra la estructura final de la carpeta resultante, la cual incluye archivos de salida, reportes y librerías del proyecto.

Figura 2. Estructura de la carpeta de trabajo luego de ejecutar el flujo de síntesis topográfica.



Nota. La figura muestra la organización de archivos generados por Design Compiler NXT durante la síntesis topográfica. Esta estructura facilita el rastreo de reportes, *netlists*, restricciones, archivos intermedios y *logs* necesarios para depuración, verificación y reproducibilidad del flujo.

7.3. Análisis de resultados de la síntesis topográfica

7.3.1. Ejecución del archivo `setup.tcl`

La primera instrucción del flujo ejecutado corresponde a la carga del entorno de trabajo mediante el comando:

Cuadro 7. Ejecución del archivo `setup.tcl`.

```
dcnxt_shell-topo> source setup.tcl
RM: Opening existing NDM design library: nanochip.dlib
Information: Loading library file '/home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/nanochip.dlib' (FILE-007)
Information: Loading library file '/home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/CLIBs/stclib_tcbn65gpluswc.ndm' (FILE-007)
Information: Loading library file '/home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/CLIBs/io_tpdn65gpgv2od3_sdwc.ndm' (FILE-007)
Information: Using the 'open_lib' command has enabled NDM mode for the current Design
    Compiler NXT session. (DCT-294)

================================================================

Library Settings:
search_path:              . /usr/synopsys/syn/W-2024.09-SP4/libraries/syn
    /usr/synopsys/syn/W-2024.09-SP4/dw/syn_ver
    /usr/synopsys/syn/W-2024.09-SP4/dw/sim_ver ./ref/DBs ./ref/CLIBs ./ref/tech ./rtl
    ./scripts
link_library:             * stclib_tcbn65gpluswc.db io_tpdn65gpgv2od3_sdwc.db
target_library:           stclib_tcbn65gpluswc.db io_tpdn65gpgv2od3_sdwc.db
physical libraries:       stclib_tcbn65gpluswc.ndm io_tpdn65gpgv2od3_sdwc.ndm
physical design library:  nanochip.dlib
Universidad del Valle de Guatemala || UVG_EGJ_Jose_Mendez || UVG_TSMC

================================================================
```

Nota. El cuadro presenta la salida en consola de la ejecución del archivo `setup.tcl`, en la cual se verifica la carga correcta de las bibliotecas lógicas y físicas del entorno.

El resultado confirma la correcta carga del entorno de Design Compiler NXT en modo topográfico (`NDM mode`), el cual permite estimar parámetros físicos desde la fase de síntesis. Durante esta ejecución, se verifica que las bibliotecas lógicas (`.db`) y físicas (`.ndm`) se enlazaron correctamente, estableciendo la base tecnológica del diseño sobre el proceso TSMC de 65 nm.

La línea `RM: Opening existing NDM design library: nanochip.dlib` indica que la biblioteca física del proyecto fue abierta exitosamente, lo que confirma la compatibilidad entre el entorno de trabajo y los archivos de referencia.

Asimismo, el bloque `Library Settings` muestra las rutas de búsqueda (`search_path`) y las librerías activas de vinculación y destino (`link_library` y `target_library`), verificando que el flujo reconocerá las celdas estándar y los módulos de entrada y salida definidos en las bibliotecas `stclib_tcbn65gpluswc` y `io_tpdn65gpgv2od3_sdwc`.

Con esta verificación inicial se confirma que el entorno se encuentra completamente configurado y listo para proceder con la lectura y vinculación del diseño RTL.

7.3.2. Ejecución del comando `set_svf`

El primer paso después de configurar el entorno consiste en habilitar el registro incremental de cambios mediante el comando:

Cuadro 8. Activación del registro incremental (`set_svf`).

```
dcnxt_shell-topo> set_svf nanochip.svf
1
```

Nota. El cuadro presenta la ejecución del comando `set_svf`, mediante el cual se activa el registro incremental de cambios del flujo de síntesis.

Este comando crea el archivo `nanochip.svf`, que registra todas las operaciones realizadas durante la sesión. El valor de retorno `1` confirma que la grabación se inició correctamente. El archivo resultante servirá posteriormente para verificación formal con Formality, garantizando que la *netlist* sintetizada sea funcionalmente equivalente al código RTL original.

7.3.3. Ejecución del comando `analyze`

La siguiente instrucción analiza el código fuente en Verilog, traduciendo el diseño RTL a una representación interna reconocida por Design Compiler NXT.

Cuadro 9. Análisis del código fuente en Verilog (`analyze`).

```
dcnxt_shell-topo> analyze -format verilog nanochip.v
Running PRESTO HDLC
Compiling source file ./rtl/nanochip.v
Presto compilation completed successfully.
Loading db file '/home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/stclib_tcbn65gpluswc.db'
Loading db file '/home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/io_tpdn65gpgv2od3_sdwc.db'
1
```

Nota. El cuadro presenta la salida de la ejecución del comando `analyze`, en la cual se verifica la compilación exitosa del archivo RTL y la carga de las bibliotecas tecnológicas asociadas.

El compilador PRESTO HDL Compiler interpreta el archivo `nanochip.v` sin errores de sintaxis. La salida confirma la carga exitosa de las bibliotecas tecnológicas de celdas estándar (`stclib_tcbn65gpluswc`) y de entrada y salida (`io_tpdn65gpgv2od3_sdwc`), las cuales se usarán para mapear la lógica durante la síntesis.

7.3.4. Ejecución del comando `elaborate`

Una vez analizado el código RTL, se elabora el diseño para generar la estructura jerárquica interna que define sus componentes, registros y conexiones.

Cuadro 10. Elaboración del diseño (`elaborate`).

```
dcnxt_shell-topo> elaborate nanochip
Loading db file '/usr/synopsys/syn/W-2024.09-SP4/libraries/syn/gtech.db'
Loading db file '/usr/synopsys/syn/W-2024.09-SP4/libraries/syn/standard.sldb'
  Loading link library 'tcbn65gpluswc'
  Loading link library 'tpdn65gpgv2od3_sdwc'
  Loading link library 'gtech'
Running PRESTO HDLC

```

```
Inferred memory devices in process in routine 'nanochip' in file './rtl/nanochip.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB |  Set  | Reset | ST | Line |
===============================================================================
|     counter_reg     | Flip-flop |  12   |  Y  | N  | None  | Async | N  |  20  |
|        q_reg        | Flip-flop |   8   |  Y  | N  | None  | None  | N  |  35  |
===============================================================================
Presto compilation completed successfully. (nanochip)
Module: nanochip, Ports: 14, Input: 5, Output: 9, Inout: 0
Module: nanochip, Registers: 20, Async set/reset: 12, Sync set/reset: 0
Information: Module nanochip report end. (ELAB-965)
Elaborated 1 design.
Current design is now 'nanochip'.
1
Current design is 'nanochip'.
```

Nota. El cuadro presenta la salida del comando `elaborate`, mediante el cual se genera la estructura jerárquica interna del diseño y se identifican sus puertos y registros.

La elaboración identifica correctamente los módulos, puertos y elementos secuenciales del circuito. El informe interno muestra un total de 14 puertos, 5 entradas y 9 salidas, así como 20 registros, entre los que destacan `counter_reg` y `q_reg`. El compilador reconoce también los mecanismos de reinicio asincrónico asociados a los registros del contador. Al final, la herramienta confirma que el diseño activo es `nanochip`.

7.3.5. Ejecución del comando `link`

Posteriormente, el diseño se vincula con las bibliotecas lógicas y físicas definidas en el entorno.

Cuadro 11. Vinculación del diseño (`link`).

```
dcnxt_shell-topo> link
  Linking design 'nanochip'
  Using the following designs and libraries:
  --------------------------------------------------------------------------
  nanochip                    /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
  Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/nanochip.db
  tcbn65gpluswc (library)     /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
  Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/stclib_tcbn65gpluswc.db
  tpdn65gpgv2od3_sdwc (library) /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
  Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/io_tpdn65gpgv2od3_sdwc.db
1
```

Nota. El cuadro presenta la salida del comando `link`, en la cual se verifica la vinculación del diseño con las bibliotecas lógicas necesarias para la síntesis.

La vinculación garantiza que todas las celdas lógicas referenciadas estén disponibles dentro de las bibliotecas cargadas. El archivo `nanochip.db`, generado durante el análisis, queda reconocido como diseño principal dentro del flujo.

7.3.6. Ejecución del comando `check_design`

Finalmente, se verifica la integridad estructural y la conectividad de los puertos mediante `check_design`.

Cuadro 12. Verificación estructural del diseño (`check_design`).

```
dcnxt_shell-topo> check_design

****************************************
check_design summary:
Version:     W-2024.09-SP4
Date:        Sat Nov  8 16:39:03 2025
****************************************

                   Name                                                    Total
--------------------------------------------------------------------------------
Inputs/Outputs                                                              1
    Feedthrough (LINT-29)                                                   1
--------------------------------------------------------------------------------

Warning: In design 'nanochip', input port 'reset_ring_osc' is connected directly to
    output port 'out_ring_osc'. (LINT-29)
1
```

Nota. El cuadro presenta el resultado de `check_design`, en el cual se resume la verificación estructural del diseño y se reportan advertencias de conectividad.

El informe confirma que no hay errores críticos de jerarquía ni de conectividad. La advertencia `LINT-29` señala un *feedthrough* entre `reset_ring_osc` y `out_ring_osc`, situación común en rutas de control sin lógica intermedia.

7.3.7. Ejecución del comando `source -echo nanochip.con`

Este paso aplica las restricciones de temporización y entorno definidas en `nanochip.con`. Se fijan parámetros del reloj, retardos de entrada y salida, y condiciones de operación.

Cuadro 13. Carga de restricciones temporales (`nanochip.con`).

```
dcnxt_shell-topo> source -echo nanochip.con
####################################
# Constraints file for design nanochip
remove_sdc
create_clock -period 1.2 [get_ports clk]
set_clock_uncertainty -setup 0.1 [get_clocks clk]
set_clock_transition -max 0.05 [get_clocks clk]
# Using default Operating Conditions from the slow corner library: ss0p75v125c
# Process :     0.99
# Temperature : 125.00
# Voltage :     0.75
set_input_delay -clock clk -max 0.2 [get_ports "select reset_ring_osc reset_nanochip"]
remove_input_delay [get_ports clk]
set_driving_cell -max -no_design_rule -lib_cell BUFFD0 [all_inputs]
remove_driving_cell [get_ports clk]
set_output_delay -clock clk -max 0.2 [all_outputs]
```

```
set_load -max 5 [all_outputs]
1
```

Nota. El cuadro presenta la ejecución del archivo `nanochip.con`, mediante el cual se cargan las restricciones temporales del diseño.

El archivo establece un período de reloj de 1.2 ns, una incertidumbre de 0.1 ns y una transición máxima de 0.05 ns. También fija retardos de entrada y salida, así como una celda de manejo para las entradas. Estas restricciones guían el análisis estático de tiempo en Design Compiler NXT.

7.3.8. Ejecución del comando `check_timing`

El comando `check_timing` valida la coherencia de las restricciones e informa riesgos como redes de alto *fanout* o puertos con *delay* parcial.

Cuadro 14. Verificación de coherencia de restricciones (`check_timing`).

```
dcnxt_shell-topo> check_timing
Warning: The variable hdlin_source_to_gates_mode is obsolete with v3.3a. (HDL-361)
Module: DW01_cmp2_width12, Ports: 28, Input: 26, Output: 2, Inout: 0
Module: DW01_cmp2_width12, Registers: 0, Async set/reset: 0, Sync set/reset: 0
Module: DW01_inc_width12, Ports: 24, Input: 12, Output: 12, Inout: 0
Module: DW01_inc_width12, Registers: 0, Async set/reset: 0, Sync set/reset: 0
Information: Updating design information... (UID-85)
Warning: Design 'nanochip' contains 25 high-fanout nets. A fanout number of 1000 will
    be used for delay calculations involving these nets. (TIM-134)

Information: Checking generated_clocks...
Information: Checking loops...
Information: Checking no_input_delay...
Information: Checking unconstrained_endpoints...
Information: Checking pulse_clock_cell_type...
Information: Checking no_driving_cell...
Information: Checking partial_input_delay...
Warning: there are 4 input ports that only have partial input delay specified.
    (TIM-212)
1
Current design is 'nanochip'.
```

Nota. El cuadro presenta la salida de `check_timing`, en la cual se reporta la verificación de las restricciones temporales y las advertencias asociadas al diseño.

La salida indica 25 redes de alto *fanout* y 4 puertos con *delay* parcial. Se recomienda completar `set_input_delay` en dichos puertos. No se reportan *endpoints* sin restricción.

7.3.9. Ejecución del comando `source -echo nanochip.pcon`

Ahora se cargan las restricciones físicas de `nanochip.pcon`. Se define el área del *die*, el *core*, la utilización objetivo y las redes ideales para evitar impacto en el análisis funcional de *timing*.

Cuadro 15. Carga de restricciones físicas (`nanochip.pcon`).

```
dcnxt_shell-topo> source -echo nanochip.pcon
# Define die and core
create_die_area -coordinate {0 0 1000 1000}
Information: linking reference library :
    /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/CLIBs/stclib_tcbn65gpluswc.ndm. (PSYN-878)
Information: linking reference library :
    /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/CLIBs/io_tpdn65gpgv2od3_sdwc.ndm. (PSYN-878)

  Linking design 'nanochip'
  Using the following designs and libraries:
  --------------------------------------------------------------------------
  nanochip                    /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
  Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/nanochip.db
  tcbn65gpluswc (library)     /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
  Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/stclib_tcbn65gpluswc.db
  tpdn65gpgv2od3_sdwc (library)
                              /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/

    Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/io_tpdn65gpgv2od3_sdwc.db

create_core_area -coordinate {100 100 900 900}
# Placement density
set_utilization 0.40
# Core aspect ratio
set_aspect_ratio 1.0
# Lower local density threshold
set_app_var placer_max_cell_density_threshold 0.55
set_ideal_network [get_ports clk]
set_false_path -from [get_ports {reset_nanochip reset_ring_osc}]
set_ideal_network [get_ports {reset_nanochip reset_ring_osc}]
1
Current design is 'nanochip'.
```

Nota. El cuadro presenta la carga del archivo `nanochip.pcon`, en el cual se definen las restricciones físicas del diseño y las redes tratadas como ideales.

Este archivo define un *die* de 1000×1000 y un *core* de 800×800 unidades, con una utilización global del 40 %. Además, marca `clk`, `reset_nanochip` y `reset_ring_osc` como redes ideales y declara rutas falsas para las señales de reinicio, evitando su impacto en el análisis funcional de *timing*.

7.3.10. Ejecución del comando `set_clock_gating_style`

Este comando define el estilo de *clock gating* para reducir el consumo dinámico del circuito.

Cuadro 16. Configuración del estilo de *clock gating*.

```
dcnxt_shell-topo> set_clock_gating_style -positive_edge_logic {integrated}
    -negative_edge_logic {integrated} -control_point before -max_fanout 64
Current clock gating style....
Sequential cell: latch
```

```
Minimum bank bitwidth: 3
Minimum bank bitwidth for enhanced clock gating: 6
Maximum fanout: 64
Setup time for clock gate: -
Hold time for clock gate: -
Clock gating circuitry (positive edge): integrated
Clock gating circuitry (negative edge): integrated
Note: inverter between clock gating circuitry and (negative edge) register clock pin.
Control point insertion: before
Control signal for control point: scan_enable
Observation point insertion: false
Observation logic depth: 5
1
```

Nota. El cuadro presenta la configuración del estilo de *clock gating* aplicada durante la síntesis para reducir el consumo dinámico del diseño.

El estilo de *clock gating* se establece como `integrated` para ambos flancos del reloj. Esto permite insertar lógica de habilitación antes del punto de control, reduciendo el consumo dinámico sin alterar la sincronización funcional.

7.3.11. Ejecución del comando `set_leakage_optimization`

Este paso activa la optimización de consumo estático en el proceso de síntesis.

Cuadro 17. Habilitación de la optimización de fuga (`set_leakage_optimization`).

```
dcnxt_shell-topo> set_leakage_optimization true
1
```

Nota. El cuadro presenta la activación de la optimización de fuga, utilizada para reducir el consumo estático durante la síntesis.

El parámetro `true` habilita la optimización de celdas basada en la minimización de corriente de fuga, o *leakage*, reduciendo el consumo estático en nodos inactivos del circuito.

7.3.12. Ejecución del comando `report_clock`

Finalmente, se genera un reporte de los relojes definidos en el diseño para verificar su configuración.

Cuadro 18. Reporte de los relojes del diseño.

```
dcnxt_shell-topo> report_clock
Information: Updating graph... (UID-83)
Warning: Design 'nanochip' contains 25 high-fanout nets. A fanout number of 1000 will
    be used for delay calculations involving these nets. (TIM-134)

****************************************
Report : clocks
Design : nanochip
Version: W-2024.09-SP4
Date   : Sat Nov  8 17:25:30 2025
```

```
10 ****************************************
11
12 Attributes:
13     d - dont_touch_network
14     f - fix_hold
15     p - propagated_clock
16     G - generated_clock
17     g - lib_generated_clock
18
19 Clock          Period   Waveform            Attrs     Sources
20 -------------------------------------------------------------------------------
21 clk              1.20   {0 0.6}                       {clk}
22 -------------------------------------------------------------------------------
23 1
24 Current design is 'nanochip'.
```

Nota. El cuadro presenta la salida del comando `report_clock`, en la cual se muestra la configuración del reloj principal del diseño.

El comando `report_clock` muestra el reloj principal `clk` con un período de 1.2 ns y una forma de onda {0 0.6}, lo que confirma la correcta definición del dominio de reloj para la etapa posterior de optimización.

7.3.13. Ejecución del comando `group_path` para el grupo `clk`

Este comando agrupa las rutas relacionadas con el reloj principal con el fin de priorizar su optimización durante la compilación.

Cuadro 19. Agrupación de rutas del reloj principal.

```
1 dcnxt_shell-topo> group_path -name clk -critical 0.21 -weight 5
2 1
```

Nota. El cuadro presenta la definición del grupo de rutas asociado al reloj principal del diseño.

El grupo `clk` se define con un rango crítico de 0.21 ns y un peso de 5, lo que da mayor prioridad a las rutas de registro a registro durante la optimización de temporización.

7.3.14. Ejecución del comando `group_path` para entradas

Cuadro 20. Agrupación de rutas de entrada.

```
1 dcnxt_shell-topo> group_path -name INPUTS -from [all_inputs]
2 1
```

Nota. El cuadro presenta la definición del grupo de rutas asociado a las entradas del diseño.

Este grupo abarca todas las rutas que parten de las entradas del diseño. Su definición permite controlar los retardos de propagación desde los pines externos hasta los registros internos.

7.3.15. Ejecución del comando `group_path` para salidas

Cuadro 21. Agrupación de rutas de salida.

```
dcnxt_shell-topo> group_path -name OUTPUTS -to [all_outputs]
1
```

Nota. El cuadro presenta la definición del grupo de rutas asociado a las salidas del diseño.

El grupo `OUTPUTS` abarca las rutas que van desde los registros internos hacia los puertos de salida. Con ello, el compilador puede balancear la optimización de las rutas externas junto con las internas.

7.3.16. Ejecución del comando `group_path` para rutas combinacionales

Cuadro 22. Agrupación de rutas combinacionales.

```
dcnxt_shell-topo> group_path -name COMBO -from [all_inputs] -to [all_outputs]
1
```

Nota. El cuadro presenta la definición del grupo de rutas combinacionales entre entradas y salidas del diseño.

El grupo `COMBO` reúne las rutas puramente combinacionales entre entradas y salidas, lo que permite un análisis separado de los caminos que no involucran registros.

7.3.17. Ejecución del comando `set_ungroup`

Cuadro 23. Conservación de la jerarquía del diseño principal.

```
dcnxt_shell-topo> set_ungroup [get_designs "nanochip"] false
1
```

Nota. El cuadro presenta la configuración utilizada para conservar la jerarquía del diseño durante la síntesis.

Con `set_ungroup` se conserva la jerarquía del bloque principal `nanochip`, lo que evita su desagrupación durante la síntesis. Esta configuración facilita la verificación jerárquica y el mapeo estructural en las etapas posteriores del flujo de diseño.

7.3.18. Ejecución del comando `set_optimize_registers`

Este ajuste habilita la optimización de registros, incluido el *retiming* elegible, dentro del diseño indicado.

Cuadro 24. Habilitación de la optimización de registros.

```
dcnxt_shell-topo> set_optimize_registers true -design nanochip
1
```

Nota. El cuadro presenta la activación de la optimización de registros para el diseño `nanochip`.

La opción `true` permite a la herramienta mover o reequilibrar registros donde la legalidad del diseño lo permita, con el objetivo de mejorar el *slack* y cumplir las restricciones, siempre que no existan directivas que impidan el *retiming*.

7.3.19. Ejecución del comando `set_cost_priority`

Este comando define la prioridad de costo durante la optimización.

Cuadro 25. Prioridad de costo enfocada en *delay*.

```
dcnxt_shell-topo> set_cost_priority -delay
1
```

Nota. El cuadro presenta la configuración de prioridad de costo utilizada durante la optimización del diseño.

Con `-delay` se prioriza la corrección de violaciones de *setup* por encima de las violaciones DRC, lo que orienta las decisiones de dimensionamiento y de inserción de *buffers* hacia la mejora de la temporización.

7.3.20. Ejecución del comando `report_physical_constraints`

Se verifica el estado de las restricciones físicas aplicadas al *floorplan*.

Cuadro 26. Reporte de restricciones físicas.

```
dcnxt_shell-topo> report_physical_constraints
****************************************
report_physical_constraints
Version: W-2024.09-SP4
Date: Sat Nov  8 17:31:15 2025
****************************************

 Design nanochip
 Unit is micron

 Die Area                      { { 0.000 0.000 } { 1000.000 0.000 } { 1000.000
    1000.000 } { 0.000 1000.000 } { 0.000 0.000 } }
 Placement area                { 100.000 100.000 900.000 900.000 }
1
```

Nota. El cuadro presenta el reporte de restricciones físicas activas antes de la compilación topográfica.

El reporte confirma que el área del *die* es de 1000×1000 unidades y que el área de colocación corresponde a $\{100, 100, 900, 900\}$. Esto valida la activación de las directivas definidas en `nanochip.pcon` antes de ejecutar `compile_ultra`.

7.3.21. Ejecución del comando `report_path_group`

Este comando muestra los grupos de caminos definidos y sus parámetros de prioridad y rango crítico.

Cuadro 27. Reporte de grupos de caminos.

```
dcnxt_shell-topo> report_path_group
Information: Updating design information... (UID-85)
Warning: Design 'nanochip' contains 25 high-fanout nets. A fanout number of 1000 will
    be used for delay calculations involving these nets. (TIM-134)
****************************************
Report : path_group
Design : nanochip
Version: W-2024.09-SP4
Date   : Sat Nov  8 17:34:07 2025
****************************************
                          Critical
Group Name    Weight    Range
--------------------------------------------------------------------------------
**default**   1.00      0.00
COMBO         1.00      0.00
INPUTS        1.00      0.00
OUTPUTS       1.00      0.00
clk           5.00      0.21
Path Group COMBO:
        -from    { clk\
                   reset_nanochip\
                   select[1]\
                   select[0]\
                   reset_ring_osc }\
        -to      { q_out[7]\
                   q_out[6]\
                   q_out[5]\
                   q_out[4]\
                   q_out[3]\
                   q_out[2]\
                   q_out[1]\
                   q_out[0]\
                   out_ring_osc }
Path Group INPUTS:
        -from    { reset_nanochip\
                   reset_ring_osc }
        -from    { clk\
                   select[1]\
                   select[0] }
Path Group OUTPUTS:
        -to      { q_out[7]\
                   q_out[6]\
                   q_out[5]\
                   q_out[4]\
                   q_out[3]\
                   q_out[2]\
                   q_out[1]\
                   q_out[0]\
                   out_ring_osc }
Path Group clk:
        -to      clk
1
```

Nota. El cuadro presenta el reporte de grupos de caminos definidos para la optimización del diseño.

El reporte confirma que los grupos `clk`, `INPUTS`, `OUTPUTS` y `COMBO` están activos con los pesos y rangos definidos. El grupo `clk` conserva un peso de 5 y un rango crítico de 0.21 ns.

7.3.22. Ejecución del comando `get_attribute` para `ungroup`

Cuadro 28. Verificación del atributo `ungroup`.

```
dcnxt_shell-topo> get_attribute [get_designs "nanochip"] ungroup
false
```

Nota. El cuadro presenta la consulta del atributo `ungroup` del diseño `nanochip`.

El valor `false` indica que la jerarquía del diseño se mantiene agrupada, tal como se esperaba.

7.3.23. Ejecución del comando `get_attribute` para `optimize_registers`

Cuadro 29. Verificación del atributo `optimize_registers`.

```
dcnxt_shell-topo> get_attribute [get_designs "nanochip"] optimize_registers
true
```

Nota. El cuadro presenta la consulta del atributo `optimize_registers` del diseño `nanochip`.

El valor `true` confirma que la optimización de registros está habilitada en el diseño.

7.3.24. Ejecución del comando `get_attribute` para `cost_priority`

Cuadro 30. Verificación del atributo `cost_priority`.

```
dcnxt_shell-topo> get_attribute [get_designs "nanochip"] cost_priority
max_delay
```

Nota. El cuadro presenta la consulta del atributo `cost_priority` del diseño `nanochip`.

El valor `max_delay` valida que la prioridad de costo favorece la resolución de violaciones de *setup*.

7.3.25. Ejecución del comando `write_file`

Cuadro 31. Guardado del diseño sin compilar.

```
dcnxt_shell-topo> write_file -f ddc -hier -out unmapped/nanochip.ddc
Writing ddc file 'unmapped/nanochip.ddc'.
1
Current design is 'nanochip'.
```

Nota. El cuadro presenta la escritura de la vista jerárquica no mapeada del diseño en formato `ddc`.

La herramienta guarda la vista jerárquica sin compilar en `unmapped/nanochip.ddc`.

7.3.26. Ejecución del comando `set_host_options`

Cuadro 32. Definición de opciones de procesamiento.

```
dcnxt_shell-topo> set_host_options -max_cores 16
1
```

Nota. El cuadro presenta la configuración del número máximo de núcleos utilizados durante la compilación.

La herramienta se configura para utilizar hasta 16 núcleos con el fin de acelerar la compilación.

7.3.27. Ejecución del comando `report_host_options`

Cuadro 33. Reporte de configuración de procesamiento.

```
dcnxt_shell-topo> report_host_options
****************************************
Report : host_options
Version: W-2024.09-SP4
Date   : Sat Nov  8 17:34:08 2025
****************************************

Max_cores: 16

1
```

Nota. El cuadro presenta el reporte de configuración de procesamiento paralelo del entorno de síntesis.

El reporte confirma que la configuración de ejecución paralela utiliza 16 núcleos.

7.3.28. Ejecución del comando `compile_ultra`

Se ejecuta la compilación topográfica con `compile_ultra`, con las opciones `-spg`, `-retime` y `-scan` habilitadas para optimización temporal y soporte de prueba.

Cuadro 34. Compilación topográfica con `retime` y `scan` (parte 1).

```
dcnxt_shell-topo> compile_ultra -spg -retime -scan
Warning: Congestion optimization is enabled as part of the physical guidance flow.
    (OPT-1445)
Loading db file '/usr/synopsys/syn/W-2024.09-SP4/libraries/syn/dw_foundation.sldb'
Warning: DesignWare synthetic library dw_foundation.sldb is added to the
    synthetic_library in the current command. (UISN-40)
Information: Performing leakage power optimization. (PWR-850)
CPU Load: 0%, Ram Free: 0 GB, Swap Free: 6 GB, Work Disk Free: 255 GB, Tmp Disk Free:
    1094 GB
Alib files are up-to-date.

TLU+ File = /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/tech/cln65g+_1p09m+alrdl_rcworst_top2.tluplus

--------- Sanity Check on TLUPlus Files -------------
1. Checking the conducting layer names in ITF and mapping file ...
```

```
[ Passed! ]
2. Checking the via layer names in ITF and mapping file ...
[ Passed! ]
3. Checking the consistency of Min Width and Min Spacing between MW-tech and ITF ...
[ Passed! ]
----------------- Check Ends ------------------
Warning: Tool will use native floorplan capabilities to generate the floorplan.
    (DCT-398)
Using operating conditions 'WCCOM' found in library 'tcbn65gpluswc'.
Information: There is no user-defined operating conditions in the design. The tool is
    inferring operating conditions 'WCCOM' from the library 'tcbn65gpluswc'. (DCT-399)
Warning: '14' ports are missing location, compile_ultra will automatically assign a
    location for these ports. For more details run compile_ultra -check_only -spg.
    (SPG-013)
Warning: IO pad 'PVSS2DGZ_SD' is unusable: unknown logic function.  (OPT-1022)
Warning: IO pad 'PVDD2POC_SD' is unusable: unknown logic function.  (OPT-1022)
Warning: IO pad 'PVDD2DGZ_SD' is unusable: unknown logic function.  (OPT-1022)
Information: Running optimization using a maximum of 16 cores. (OPT-1500)
Information: Choosing a test methodology will restrict the optimization of sequential
    cells. (UIO-12)
Warning: Setting attribute 'fix_multiple_port_nets' on design 'nanochip'. (UIO-59)
Warning: DesignWare synthetic library dw_foundation.sldb is added to the
    synthetic_library in the current command. (UISN-40)
Warning: The variable hdlin_source_to_gates_mode is obsolete with v3.3a. (HDL-361)
Information: Evaluating DesignWare library utilization. (UISN-27)

============================================================================
| DesignWare Building Block Library  |         Version         | Available |
============================================================================
| Basic DW Building Blocks           | W-2024.09-DWBB_202409.4 |     *     |
| Licensed DW Building Blocks        | W-2024.09-DWBB_202409.4 |     *     |
============================================================================

================================================================================
| Flow Information                                                             |
--------------------------------------------------------------------------------
| Flow         | Design Compiler NXT Graphical                                 |
| Command Line | compile_ultra -retime -spg -scan                              |
================================================================================
| Design Information                                 | Value                   |
================================================================================
| Number of Scenarios                                | 0                       |
| Leaf Cell Count                                    | 61275                   |
| Number of User Hierarchies                         | 0                       |
| Sequential Cell Count                              | 20                      |
| Macro Count                                        | 0                       |
| Pad Count                                          | 0                       |
| Number of Power Domains                            | 0                       |
| Number of Path Groups                              | 5                       |
| Number of VT Class                                 | 0                       |
| Number of Voltage Areas                            | 0                       |
| Number of Clocks                                   | 1                       |
| Number of Dont Touch Cells                         | 4718                    |
| Number of Dont Touch Nets                          | 5                       |
| Number of Size Only Cells                          | 0                       |
| Design with Target Library Subset (with clock_path)| false  (false )         |
| Design with UPF Data                               | false                   |
| NDM Mode                                           | true                    |
| IC Compiler II Link                                | false                   |
```

```
| Advanced RC Correlation                                | false                 |
===============================================================================
Information: Sequential output inversion is enabled.  SVF file must be used for formal
    verification. (OPT-1208)
Information: Retiming is enabled. SVF file must be used for formal verification.
    (OPT-1210)

  Simplifying Design 'nanochip'

Warning: The trip points for the library named tpdn65gpgv2od3_sdwc differ from those
    in the library named tcbn65gpluswc. (TIM-164)
```

Nota. El cuadro presenta la primera parte de la ejecución de `compile_ultra`, en la cual se inicializan las bibliotecas, los modelos RC y las condiciones de operación del diseño.

Cuadro 35. Compilación topográfica con `retime` y `scan` (parte 2).

```
****************************************************************
Information: TLUPlus based RC computation is enabled. (RCEX-141)
****************************************************************
Information: The distance unit in Capacitance and Resistance is 1 micron. (RCEX-007)
Information: The RC model used is TLU+. (RCEX-015)
Information: Library Derived Cap for layer M1 : 0.00022 0.00022 (RCEX-011)
Information: Library Derived Res for layer M1 : 0.0026 0.0026 (RCEX-011)
Information: Library Derived Cap for layer M2 : 0.00022 0.00022 (RCEX-011)
Information: Library Derived Res for layer M2 : 0.0023 0.0023 (RCEX-011)
Information: Library Derived Cap for layer M3 : 0.00022 0.00022 (RCEX-011)
Information: Library Derived Res for layer M3 : 0.0023 0.0023 (RCEX-011)
Information: Library Derived Cap for layer M4 : 0.00022 0.00022 (RCEX-011)
Information: Library Derived Res for layer M4 : 0.0023 0.0023 (RCEX-011)
Information: Library Derived Cap for layer M5 : 0.00022 0.00022 (RCEX-011)
Information: Library Derived Res for layer M5 : 0.0023 0.0023 (RCEX-011)
Information: Library Derived Cap for layer M6 : 0.00022 0.00022 (RCEX-011)
Information: Library Derived Res for layer M6 : 0.0023 0.0023 (RCEX-011)
Information: Library Derived Cap for layer M7 : 0.00021 0.00021 (RCEX-011)
Information: Library Derived Res for layer M7 : 0.0023 0.0023 (RCEX-011)
Information: Library Derived Cap for layer M8 : 0.00033 0.00033 (RCEX-011)
Information: Library Derived Res for layer M8 : 8.8e-05 8.8e-05 (RCEX-011)
Information: Library Derived Cap for layer M9 : 0.00033 0.00033 (RCEX-011)
Information: Library Derived Res for layer M9 : 8.8e-05 8.8e-05 (RCEX-011)
Information: Library Derived Cap for layer AP : 0.00032 0.00032 (RCEX-011)
Information: Library Derived Res for layer AP : 1.1e-05 1.1e-05 (RCEX-011)
Information: Library Derived Horizontal Cap : 0.00024 0.00024 (RCEX-011)
Information: Library Derived Horizontal Res : 0.0019 0.0019 (RCEX-011)
Information: Library Derived Vertical Cap : 0.00026 0.00026 (RCEX-011)
Information: Library Derived Vertical Res : 0.0014 0.0014 (RCEX-011)
Information: Using derived R and C coefficients. (RCEX-008)
Information: Using region-based R and C coefficients. (RCEX-013)
Information: Library Derived Via Res : 0.0022 0.0022 (RCEX-011)
  Loading target library 'tpdn65gpgv2od3_sdwc'
Warning: IO pad 'PVSS2DGZ_SD' is unusable: unknown logic function.  (OPT-1022)
Warning: IO pad 'PVDD2POC_SD' is unusable: unknown logic function.  (OPT-1022)
Warning: IO pad 'PVDD2DGZ_SD' is unusable: unknown logic function.  (OPT-1022)
Loaded alib file './alib-52/stclib_tcbn65gpluswc.db.alib'
Loaded alib file './alib-52/io_tpdn65gpgv2od3_sdwc.db.alib' (placeholder)
Warning: Operating condition WCCOM set on design nanochip has different process,
voltage and temperatures parameters than the parameters at which target library
tpdn65gpgv2od3_sdwc is characterized. Delays may be inaccurate as a result. (OPT-998)
Module: DW01_NAND2, Ports: 3, Input: 2, Output: 1, Inout: 0
```

```
Module: DW01_NAND2, Registers: 0, Async set/reset: 0, Sync set/reset: 0
  Building model 'DW01_NAND2'
Information: Ungrouping 0 of 1 hierarchies before Pass 1 (OPT-775)
CPU Load: 0%, Ram Free: 0 GB, Swap Free: 6 GB, Work Disk Free: 255 GB, Tmp Disk Free: 1094 GB
Information: State dependent leakage is now switched from on to off.

  Beginning Pass 1 Mapping
  ------------------------
  Processing 'nanochip'
Information: The register 'q_reg[7]' is a constant and will be removed. (OPT-1206)
 Implement Synthetic for 'nanochip'.
Information: Added key list 'DesignWare' to design 'nanochip'. (DDB-72)
CPU Load: 1%, Ram Free: 0 GB, Swap Free: 6 GB, Work Disk Free: 255 GB, Tmp Disk Free: 1094 GB

  Updating timing information
Information: Updating design information... (UID-85)
Information: The target library(s) contains cell(s), other than black boxes, that are not characterized for internal power. (PWR-24)
Information: Automatic shift-register identification is enabled for scan. Not all registers will be scan-replaced. (OPT-467)

  Beginning Mapping Optimizations  (Ultra High effort)
  -------------------------------
Information: Checking pipeline property of design nanochip. (RTDC-137)
Information: design nanochip is not a pipeline. (RTDC-139)
Information: Checking pipeline property of design nanochip. (RTDC-137)
Information: Pipeline detection aborted. Reason: cell counter_reg[6] is on a feedback loop. (RTDC-138)
Information: Aborted pipeline detection on design nanochip. (RTDC-140)
  Mapping Optimization (Phase 1)
Information: Checkpoint guidance is generated in the .svf file. (INFO-122)
  Retiming nanochip (top)
  Preferred flip-flop is DFD2 with setup = 0.02

  Retiming base-clock clk, rising edge.
  Beginning minimum period retiming ...
  ... minimum period retiming done.
  Beginning minimum area retiming step 1 ...
  ... minimum area retiming step 1 done.
  Beginning minimum area retiming step 2 ...

  ... minimum area retiming step 2 done.
  Beginning minimum area retiming step 3 ...

  ... minimum area retiming step 3 done.
  Beginning final register move ...
  ... final register move done.
  Lower bound estimate = 0.95
  Critical path length = 0.95
  Clock correction = 0.25 (clock-to-Q delay = 0.14, setup = 0.02, uncertainty = 0.10)
Information: Automatic shift-register identification is enabled for scan. Not all registers will be scan-replaced. (OPT-467)

  Beginning Global Optimizations
  ------------------------------
  Numerical Synthesis (Phase 1)
  Numerical Synthesis (Phase 2)
```

```
170 |
171 |                                  TOTAL
172 |    ELAPSED            WORST NEG   SETUP    DESIGN                                LEAKAGE
173 |      TIME      AREA      SLACK     COST    RULE COST         ENDPOINT             POWER
174 |    --------- --------- --------- --------- --------- ------------------------- ---------
175 |      0:00:34   11825.3     11.03     152.4   66853.6
        89990.3438
176 |      0:00:34   11741.4     11.03     156.5   66853.6
        89332.2500
177 |
178 |    Beginning Delay Optimization
179 |    ----------------------------
180 |      0:00:34   11741.4     11.03     156.5   66853.6
        89332.2500
181 |      0:00:35   13154.4      1.41      17.9   58281.1
        107716.9062
182 |      0:00:36   13154.4      1.41      17.9   58281.1
        107716.9062
183 |      0:00:36   13157.6      1.41      21.6   58281.1
        107985.8984
184 |      0:00:36   13157.6      1.41      21.6   58281.1
        107985.8984
185 |      0:00:36   13138.9      1.41      21.4   58281.1
        107775.9141
186 | CPU Load: 29%, Ram Free: 0 GB, Swap Free: 6 GB, Work Disk Free: 255 GB, Tmp Disk Free:
        1094 GB
187 | Loading db file '/home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
188 | Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/stclib_tcbn65gpluswc.db'
189 | Loading db file '/home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
190 | Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/io_tpdn65gpgv2od3_sdwc.db'
191 | CPU Load: 29%, Ram Free: 0 GB, Swap Free: 6 GB, Work Disk Free: 255 GB, Tmp Disk Free:
        1094 GB
192 |   Loading design 'nanochip'
193 | Information: The target library(s) contains cell(s), other than black boxes, that are
        not characterized for internal power. (PWR-24)
194 | ************************************************************
195 | Information: TLUPlus based RC computation is enabled. (RCEX-141)
196 | ************************************************************
197 | Information: The distance unit in Capacitance and Resistance is 1 micron. (RCEX-007)
198 | Information: The RC model used is TLU+. (RCEX-015)
199 | Information: Library Derived Cap for layer M1 : 0.00022 0.00022 (RCEX-011)
200 | Information: Library Derived Res for layer M1 : 0.0026 0.0026 (RCEX-011)
201 | Information: Library Derived Cap for layer M2 : 0.00022 0.00022 (RCEX-011)
202 | Information: Library Derived Res for layer M2 : 0.0023 0.0023 (RCEX-011)
203 | Information: Library Derived Cap for layer M3 : 0.00022 0.00022 (RCEX-011)
204 | Information: Library Derived Res for layer M3 : 0.0023 0.0023 (RCEX-011)
205 | Information: Library Derived Cap for layer M4 : 0.00022 0.00022 (RCEX-011)
206 | Information: Library Derived Res for layer M4 : 0.0023 0.0023 (RCEX-011)
207 | Information: Library Derived Cap for layer M5 : 0.00022 0.00022 (RCEX-011)
208 | Information: Library Derived Res for layer M5 : 0.0023 0.0023 (RCEX-011)
209 | Information: Library Derived Cap for layer M6 : 0.00022 0.00022 (RCEX-011)
210 | Information: Library Derived Res for layer M6 : 0.0023 0.0023 (RCEX-011)
211 | Information: Library Derived Cap for layer M7 : 0.00021 0.00021 (RCEX-011)
212 | Information: Library Derived Res for layer M7 : 0.0023 0.0023 (RCEX-011)
213 | Information: Library Derived Cap for layer M8 : 0.00033 0.00033 (RCEX-011)
214 | Information: Library Derived Res for layer M8 : 8.8e-05 8.8e-05 (RCEX-011)
215 | Information: Library Derived Cap for layer M9 : 0.00033 0.00033 (RCEX-011)
216 | Information: Library Derived Res for layer M9 : 8.8e-05 8.8e-05 (RCEX-011)
217 | Information: Library Derived Cap for layer AP : 0.00032 0.00032 (RCEX-011)
```

```
Information: Library Derived Res for layer AP : 1.1e-05 1.1e-05 (RCEX-011)
Information: Library Derived Horizontal Cap : 0.00024 0.00024 (RCEX-011)
Information: Library Derived Horizontal Res : 0.0019 0.0019 (RCEX-011)
Information: Library Derived Vertical Cap : 0.00026 0.00026 (RCEX-011)
Information: Library Derived Vertical Res : 0.0014 0.0014 (RCEX-011)
Information: Using derived R and C coefficients. (RCEX-008)
Information: Using region-based R and C coefficients. (RCEX-013)
Information: Library Derived Via Res : 0.0022 0.0022 (RCEX-011)

Information: The design utilization is 0.02
Running Main Compile placer.
Placing standard cells from scratch.
Running precluster optimization.
...13%...25%...38%...50%...63%...75%...88%...100% done.
Running Main Compile placer - done.

                                  TOTAL
   ELAPSED            WORST NEG   SETUP    DESIGN                              LEAKAGE
    TIME      AREA      SLACK     COST    RULE COST         ENDPOINT            POWER
  --------- --------- --------- --------- --------- ------------------------- ---------
    0:00:41   13138.9      1.74     279.1   58350.5
    107775.9141
    0:00:42   11959.9      1.73     257.1   58325.4
    94786.7656

 Collecting Buffer Trees ... Found 107

 Processing Buffer Trees ...

    [11]  10% ...
    [22]  20% ...
    [33]  30% ...
    [44]  40% ...
    [55]  50% ...
    [66]  60% ...
    [77]  70% ...
    [88]  80% ...
    [99]  90% ...
    [107] 100% Done ...

Information: Automatic high-fanout synthesis deletes 542 cells. (HFS-802)
Information: Automatic high-fanout synthesis adds 311 new cells. (PSYN-864)

  Beginning Timing Optimizations
  ------------------------------

  Beginning Design Rule Fixing  (max_transition)  (max_capacitance)
  -----------------------------------

                                  TOTAL
   ELAPSED            WORST NEG   SETUP    DESIGN                              LEAKAGE
    TIME      AREA      SLACK     COST    RULE COST         ENDPOINT            POWER
  --------- --------- --------- --------- --------- ------------------------- ---------
    0:00:43   11754.0      1.48     148.5   58269.1
    92350.3750

Fixing max_capacitance rule(DRC-100)

    0:00:43   11754.0      1.48     148.5   58269.1
```

```
    92350.3750
    0:00:43   11754.0      1.48     148.5   58269.1 q_out[1]
    92350.3750

Fixing max_transition rule(DRC-101)

    0:00:43   11754.0      1.48     148.5   58269.1 q_out[1]
    92350.3750
    0:00:43   11754.0      1.48     148.5   58269.1 q_out[1]
    92350.3750

Fixing max_capacitance rule(DRC-100)

    0:00:43   11754.0      1.48     148.5   58269.1 q_out[1]
    92350.3750
    0:00:43   11754.0      1.48     148.5   58269.1 q_out[1]
    92350.3750

Fixing max_transition rule(DRC-101)

    0:00:43   11754.0      1.48     148.5   58269.1 q_out[1]
    92350.3750
    0:00:43   11754.0      1.48     148.5   58269.1 q_out[1]
    92350.3750

Running Main Compile placer.
Running congestion-aware direct-timing-driven placement.
Info: Enhanced timing-DTDP is turned ON.  TNS scheme is 6.  RAT subtractor is
    0.000000.  Crit is 0.000001.
Info: Enhanced timing-DTDP is turned ON.  TNS scheme is 6.  RAT subtractor is
    0.000000.  Crit is 0.000001.
...14%...29%...43%...57%...71%...86%...100% done.
Running Main Compile placer - done.

  Beginning Timing Optimizations
  ------------------------------

                                 TOTAL
   ELAPSED            WORST NEG  SETUP    DESIGN                              LEAKAGE
    TIME      AREA      SLACK     COST    RULE COST         ENDPOINT           POWER
  --------- --------- --------- --------- --------- ------------------------ ---------
    0:00:51   11985.1      2.25     130.1   58455.1
    92512.4453

  Beginning Design Rule Fixing  (max_transition)  (max_capacitance)
  ----------------------------------

                                 TOTAL
   ELAPSED            WORST NEG  SETUP    DESIGN                              LEAKAGE
    TIME      AREA      SLACK     COST    RULE COST         ENDPOINT           POWER
  --------- --------- --------- --------- --------- ------------------------ ---------
    0:00:51   11985.1      2.25     130.1   58455.1
    92512.4453

Fixing max_transition rule(DRC-101)

    0:00:51   11985.1      2.25     130.1   58455.1
    92512.4453
    0:00:51   11985.1      2.25     130.1   58455.1 q_out[1]
```

```
    92512.4453
```

Nota. El cuadro presenta la segunda parte de la ejecución de `compile_ultra`, en la cual se observan el mapeo, el *retiming*, la colocación inicial y las optimizaciones de temporización y reglas de diseño.

Cuadro 36. Compilación topográfica con `retime` y `scan` (parte 3).

```
320   Fixing max_transition rule(DRC-101)
321
322     0:00:51   11985.1      2.25     130.1   58455.1 q_out[1]
        92512.4453
323     0:00:51   12047.0      0.99      44.3   54422.6 q_out[1]
        94054.3047
324     0:00:54   12376.8      0.99      28.2   54422.6
        98914.7734
325     0:00:55   12400.6      0.99      17.9   54422.6
        99566.2422
326     0:00:55   12400.6      0.99      17.8   54422.6
        99566.2422
327     0:00:55   12400.6      0.99      17.8   54422.6
        99566.2422
328     0:00:55   12400.6      0.99      17.8   54422.6
        99566.2422
329     0:00:55   12400.6      0.99      17.8   54422.6
        99566.2422
330     0:00:55   12400.6      0.99      17.8   54422.6
        99566.2422
331     0:00:55   12400.6      0.99      17.8   54422.6
        99566.2422
332
333   Beginning High Effort Delay Optimization
334   ----------------------------------------
335
336                                 TOTAL
337    ELAPSED            WORST NEG   SETUP    DESIGN                              LEAKAGE
338      TIME      AREA      SLACK     COST    RULE COST         ENDPOINT           POWER
339   --------- --------- --------- --------- --------- ------------------------- ---------
340     0:00:55   12400.6      0.99      17.8   54422.6
        99566.2422
341
342 CPU Load: 29%, Ram Free: 0 GB, Swap Free: 6 GB, Work Disk Free: 255 GB, Tmp Disk Free:
        1094 GB
343 Info-lpmode: computing default table with library = lib11
344     0:00:55   12400.6      0.99      17.8   54422.6
        99566.2422
345     0:00:55   12400.6      0.99      17.8   54422.6
        99566.2422
346     0:00:55   12430.1      0.99      15.9   54422.6
        100369.2500
347     0:00:55   12453.8      0.99      15.5   54422.4
        100547.7578
348     0:00:55   12489.1      0.98      14.2   54422.2
        101004.9219
349     0:00:55   12519.0      0.97      13.1   54422.1
        101452.2109
350     0:00:55   12578.4      0.97      11.8   54422.1
        102448.6875
351     0:00:55   12612.2      0.96      10.5   54422.1
        102732.9922
```

```
352     0:00:55   12667.3      0.96       9.3   54422.1
        103376.6328
353     0:00:55   12722.4      0.96       8.8   54422.1
        104293.2266
354     0:00:56   12787.9      0.96       8.3   54422.2
        105074.4531
355     0:00:56   12837.2      0.95       7.7   54422.2
        105733.9375
356     0:00:56   12857.4      0.95       7.5   54422.2
        106191.3750
357     0:00:56   12920.4      0.94       7.1   54422.2
        107035.3750
358     0:00:56   12970.1      0.94       6.9   54422.2
        107576.3047
359     0:00:56   13001.0      0.94       6.6   54422.2
        108035.3750
360     0:00:56   13068.0      0.94       6.5   54422.2
        108768.3594
361     0:00:56   13114.8      0.94       6.5   54422.2
        109511.0859
362     0:00:56   13127.0      0.94       6.5   54422.3
        109668.9688
363     0:00:56   13136.0      0.93       6.5   54422.3
        109774.3828
364     0:00:56   13158.7      0.93       6.5   54422.3
        110162.6719
365     0:00:56   13178.5      0.93       6.5   54422.3
        110482.8125
366     0:00:56   13218.1      0.93       6.5   54422.3
        110976.1250
367     0:00:56   13231.8      0.92       6.4   54422.3
        111131.4766
368     0:00:56   13230.4      0.92       6.4   54422.3
        111151.1094
369     0:00:56   13239.0      0.92       6.4   54422.3
        111338.5156
370     0:00:56   13245.1      0.92       6.4   54422.3
        111426.7578
371     0:00:56   13248.0      0.92       6.4   54422.3
        111443.9922
372     0:00:56   13246.2      0.92       6.4   54422.2
        111476.3359
373     0:00:56   13258.4      0.92       6.4   54422.2
        111627.7188
374     0:00:56   13268.9      0.91       6.4   54422.2
        111769.3047
375     0:00:56   13283.3      0.91       6.4   54422.2
        111932.3203
376     0:00:56   13298.4      0.91       6.4   54422.2
        112205.9297
377     0:00:56   13304.5      0.91       6.4   54422.2
        112392.9844
378     0:00:56   13306.3      0.91       6.3   54422.2
        112419.2500
379     0:00:56   13319.3      0.91       6.3   54422.2
        112584.5469
380     0:00:56   13347.0      0.91       6.3   54422.2
        113009.8828
381     0:00:56   13347.0      0.91       6.3   54422.2
```

```
    113009.8828
382     0:00:57   13120.9      0.91       6.3   54422.2
    109651.5391
383     0:00:57   13120.9      0.91       6.3   54422.2
    109651.5391
384     0:00:57   13091.0      0.90       6.3   54422.2
    109304.7578
385     0:00:57   13091.0      0.90       6.3   54422.2
    109304.7578
386     0:00:57   13091.0      0.90       6.3   54422.2
    109304.7578
387     0:00:57   13069.4      0.90       6.3   54422.2
    108965.3672
388     0:00:57   13069.4      0.90       6.3   54422.2
    108965.3672
389     0:00:57   13044.2      0.90       6.3   54422.2
    108008.2188
390     0:00:57   13044.2      0.90       6.3   54422.2
    108008.2188
391 Information: "Ignoring all path group weights for Area Optimization. To disable this,
    please use "set optimize_area_ignore_path_group_weights FALSE". (OPT-1725)
392     0:00:59   12529.4      0.90       6.3   54422.2
    98058.4062
393
394   Set up Remaining Resource Array
395
396 Note: Symbol # after min delay cost means estimated hold TNS across all active
    scenarios
397
398   Optimization Complete
399   ---------------------
400
401                                  TOTAL
402    ELAPSED            WORST NEG  SETUP    DESIGN                              LEAKAGE
403      TIME      AREA      SLACK    COST    RULE COST         ENDPOINT           POWER
404   --------- --------- --------- --------- --------- ------------------------- ---------
405     0:01:00   12529.4      0.90       6.3   54422.2
    98058.4062
406 CPU Load: 36%, Ram Free: 0 GB, Swap Free: 6 GB, Work Disk Free: 255 GB, Tmp Disk Free:
    1094 GB
407 Loading db file '/home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
408 Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/stclib_tcbn65gpluswc.db'
409 Loading db file '/home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
410 Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/io_tpdn65gpgv2od3_sdwc.db'
411   Loading target library 'tpdn65gpgv2od3_sdwc'
412 Warning: IO pad 'PVSS2DGZ_SD' is unusable: unknown logic function.  (OPT-1022)
413 Warning: IO pad 'PVDD2POC_SD' is unusable: unknown logic function.  (OPT-1022)
414 Warning: IO pad 'PVDD2DGZ_SD' is unusable: unknown logic function.  (OPT-1022)
415 Information: State dependent leakage is now switched from off to on.
416 Information: Propagating switching activity (low effort zero delay simulation). (PWR-6)
417 CPU Load: 33%, Ram Free: 0 GB, Swap Free: 6 GB, Work Disk Free: 255 GB, Tmp Disk Free:
    1094 GB
418 Information: Total number of MV cells in the design.
419 --------------------------------------------------------------------------
420  MV Cells                                          Total Number
421 --------------------------------------------------------------------------
422  Level Shifter:                                    0
423  Enable Level Shifter:                             0
424  Isolation Cell:                                   0
```

```
 Retention Cell:                                 0
 Retention Clamp Cell:                           0
 Switch Cell:                                    0
 Always-On Cell:                                 0
 Repeater Cell:                                  0

---------------------------------------------------------------------------
Unmapped MV Cells
---------------------------------------------------------------------------
0 Isolation Cells are unmapped
0 Retention Clamp Cells are unmapped
---------------------------------------------------------------------------
1
Current design is 'nanochip'.
```

Nota. El cuadro presenta la tercera parte de la ejecución de `compile_ultra`, en la cual se observa el cierre de la optimización y el estado final del diseño sintetizado.

La ejecución confirma una compilación exitosa con `-spg`, `-retime` y `-scan`. El reporte muestra *retiming* efectivo, corrección del período mínimo y optimizaciones de reglas y retardo. Entre las advertencias relevantes figuran el uso de `WCCOM` inferido, los puertos sin ubicación asignada y los *pads* de entrada y salida no utilizables, condiciones que no impiden la compilación. El diseño queda con un área aproximada de 12529.4 unidades y un *worst negative slack* reportado como 0.90 positivo, lo que indica la existencia de holgura de *setup* en el escenario considerado.

7.4. Resultados y reportes

7.4.1. Ejecución del comando `report_timing` (`-delay_type max`)

Cuadro 37. Reporte de temporización del diseño sintetizado.

```
dcnxt_shell-topo> report_timing -delay_type max -max_paths 3 -path_type full_clock
    -significant_digits 4 -sort_by slack

****************************************
Report : timing
        -path full_clock
        -delay max
        -max_paths 3
        -sort_by slack
Design : nanochip
Version: W-2024.09-SP4
Date   : Sat Nov  8 18:40:47 2025
****************************************

 * Some/all delay information is back-annotated.

Operating Conditions: WCCOM   Library: tcbn65gpluswc
Wire Load Model Mode: Inactive.

  Startpoint: clk_r_REG57_S2
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: q_out[3] (output port clocked by clk)
```

```
  Path Group: OUTPUTS
  Path Type: max

  Point                                   Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                  0.0000     0.0000
  clock network delay (ideal)            0.0000     0.0000
  clk_r_REG57_S2/CP (SDFQD4)             0.0000     0.0000 r
  clk_r_REG57_S2/Q (SDFQD4)              0.1405     0.1405 r
  U4756/ZN (OAI221XD4)                   0.0651 *   0.2055 f
  U4757/ZN (OAI211D2)                    0.0354 *   0.2409 r
  U4760/ZN (NR2D2)                       0.0187 *   0.2596 f
  U4761/ZN (NR2XD1)                      0.0244 *   0.2840 r
  U4544/ZN (AOI21D2)                     0.0264 *   0.3105 f
  U4768/ZN (CKND2)                       0.0219 *   0.3324 r
  U3077/ZN (ND2D4)                       0.0259 *   0.3582 f
  U3843/ZN (INVD6)                       0.0244 *   0.3826 r
  U4679/Z (BUFFD16)                      1.3980 *   1.7806 r
  q_out[3] (out)                         0.0221 *   1.8027 r
  data arrival time                                 1.8027

  clock clk (rise edge)                  1.2000     1.2000
  clock network delay (ideal)            0.0000     1.2000
  clock uncertainty                     -0.1000     1.1000
  output external delay                 -0.2000     0.9000
  data required time                                0.9000
  -----------------------------------------------------------
  data required time                                0.9000
  data arrival time                                -1.8027
  -----------------------------------------------------------
  slack (VIOLATED)                                 -0.9027

  Startpoint: clk_r_REG187_S3
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: q_out[6] (output port clocked by clk)
  Path Group: OUTPUTS
  Path Type: max

  Point                                   Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                  0.0000     0.0000
  clock network delay (ideal)            0.0000     0.0000
  clk_r_REG187_S3/CP (SDFQD4)            0.0000     0.0000 r
  clk_r_REG187_S3/Q (SDFQD4)             0.1387     0.1387 r
  U3072/ZN (ND2D3)                       0.0260 *   0.1647 f
  U3167/ZN (CKND2)                       0.0163 *   0.1810 r
  U3168/ZN (AOI211D2)                    0.0275 *   0.2084 f
  U3243/ZN (ND2D2)                       0.0217 *   0.2301 r
  U3226/ZN (ND2D2)                       0.0216 *   0.2517 f
  U4361/ZN (AOI211XD2)                   0.0550 *   0.3067 r
  U3999/ZN (IND3D4)                      0.0632 *   0.3699 r
  U4682/Z (BUFFD16)                      1.4055 *   1.7754 r
  q_out[6] (out)                         0.0239 *   1.7993 r
  data arrival time                                 1.7993

  clock clk (rise edge)                  1.2000     1.2000
  clock network delay (ideal)            0.0000     1.2000
  clock uncertainty                     -0.1000     1.1000
  output external delay                 -0.2000     0.9000
```

```
  data required time                                    0.9000
  ------------------------------------------------------------
  data required time                                    0.9000
  data arrival time                                    -1.7993
  ------------------------------------------------------------
  slack (VIOLATED)                                     -0.8993

  Startpoint: clk_r_REG57_S2
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: q_out[1] (output port clocked by clk)
  Path Group: OUTPUTS
  Path Type: max

  Point                                     Incr       Path
  ------------------------------------------------------------
  clock clk (rise edge)                   0.0000     0.0000
  clock network delay (ideal)             0.0000     0.0000
  clk_r_REG57_S2/CP (SDFQD4)              0.0000     0.0000 r
  clk_r_REG57_S2/Q (SDFQD4)               0.1405     0.1405 r
  U3112/ZN (AOI32D2)                      0.0497 *   0.1902 f
  U3115/ZN (AOI21D2)                      0.0411 *   0.2313 r
  U3116/ZN (AOI22D1)                      0.0420 *   0.2733 f
  U3131/ZN (IOA21D2)                      0.0630 *   0.3363 f
  U4547/ZN (INVD4)                        0.0335 *   0.3698 r
  U4678/Z (BUFFD16)                       1.4022 *   1.7719 r
  q_out[1] (out)                          0.0258 *   1.7978 r
  data arrival time                                  1.7978

  clock clk (rise edge)                   1.2000     1.2000
  clock network delay (ideal)             0.0000     1.2000
  clock uncertainty                      -0.1000     1.1000
  output external delay                  -0.2000     0.9000
  data required time                                 0.9000
  ------------------------------------------------------------
  data required time                                 0.9000
  data arrival time                                 -1.7978
  ------------------------------------------------------------
  slack (VIOLATED)                                  -0.8978

1
dcnxt_shell-topo>
```

Nota. El cuadro presenta el reporte de temporización máxima, en el cual se identifican las rutas de salida más críticas del diseño.

El reporte muestra las tres rutas de salida más críticas del diseño `nanochip`, todas con violaciones negativas de tiempo de *setup*. Los valores de `slack` indican que la señal llega tarde con respecto al reloj de referencia, y la peor violación alcanza $-0.9027\,\text{ns}$. El análisis utiliza condiciones de peor caso (`WCCOM`) con la librería `tcbn65gpluswc`, y la opción `-delay_type max` calcula los retardos máximos de propagación a lo largo de todo el recorrido del reloj (`full_clock`). Este resultado evidencia la necesidad de optimizar el camino de salida mediante inserción de *buffers* o ajustes de *constraints* para alcanzar el cierre temporal en la síntesis topográfica.

7.4.2. Ejecución del comando `report_timing` (`-delay_type min`)

Cuadro 38. Reporte de verificación de tiempos de retención (*hold*).

```
dcnxt_shell-topo> report_timing -delay_type min -max_paths 3 -path_type full_clock
    -significant_digits 4 -sort_by slack

****************************************
Report : timing
        -path full_clock
        -delay min
        -max_paths 3
        -sort_by slack
Design : nanochip
Version: W-2024.09-SP4
Date   : Sat Nov  8 18:48:07 2025
****************************************

 * Some/all delay information is back-annotated.

Operating Conditions: WCCOM   Library: tcbn65gpluswc
Wire Load Model Mode: Inactive.

  Startpoint: clk_r_REG43_S2
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: clk_r_REG43_S2
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: min

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                  0.0000     0.0000
  clock network delay (ideal)            0.0000     0.0000
  clk_r_REG43_S2/CP (SDFQD4)             0.0000     0.0000 r
  clk_r_REG43_S2/Q (SDFQD4)              0.1108     0.1108 f
  U4622/ZN (IOA21D1)                     0.0569 *   0.1677 f
  clk_r_REG43_S2/D (SDFQD4)              0.0000 *   0.1678 f
  data arrival time                                 0.1678

  clock clk (rise edge)                  0.0000     0.0000
  clock network delay (ideal)            0.0000     0.0000
  clk_r_REG43_S2/CP (SDFQD4)             0.0000     0.0000 r
  library hold time                     -0.0220    -0.0220
  data required time                               -0.0220
  -----------------------------------------------------------
  data required time                               -0.0220
  data arrival time                                -0.1678
  -----------------------------------------------------------
  slack (MET)                                       0.1897

  Startpoint: clk_r_REG62_S2
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: clk_r_REG62_S2
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: min

  Point                                    Incr       Path
```

```
  -----------------------------------------------------------
  clock clk (rise edge)                   0.0000     0.0000
  clock network delay (ideal)             0.0000     0.0000
  clk_r_REG62_S2/CP (SDFQD1)              0.0000     0.0000 r
  clk_r_REG62_S2/Q (SDFQD1)               0.1230     0.1230 f
  U721/ZN (IOA21D1)                       0.0573 *   0.1803 f
  clk_r_REG62_S2/D (SDFQD1)               0.0000 *   0.1804 f
  data arrival time                                  0.1804

  clock clk (rise edge)                   0.0000     0.0000
  clock network delay (ideal)             0.0000     0.0000
  clk_r_REG62_S2/CP (SDFQD1)              0.0000     0.0000 r
  library hold time                      -0.0163    -0.0163
  data required time                                -0.0163
  -----------------------------------------------------------
  data required time                                -0.0163
  data arrival time                                 -0.1804
  -----------------------------------------------------------
  slack (MET)                                        0.1967

  Startpoint: clk_r_REG13_S2
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: clk_r_REG13_S2
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: min

  Point                                     Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                   0.0000     0.0000
  clock network delay (ideal)             0.0000     0.0000
  clk_r_REG13_S2/CP (SDFQD1)              0.0000     0.0000 r
  clk_r_REG13_S2/Q (SDFQD1)               0.1231     0.1231 f
  U2944/ZN (IOA21D1)                      0.0586 *   0.1817 f
  clk_r_REG13_S2/D (SDFQD1)               0.0000 *   0.1817 f
  data arrival time                                  0.1817

  clock clk (rise edge)                   0.0000     0.0000
  clock network delay (ideal)             0.0000     0.0000
  clk_r_REG13_S2/CP (SDFQD1)              0.0000     0.0000 r
  library hold time                      -0.0161    -0.0161
  data required time                                -0.0161
  -----------------------------------------------------------
  data required time                                -0.0161
  data arrival time                                 -0.1817
  -----------------------------------------------------------
  slack (MET)                                        0.1978

1
dcnxt_shell-topo>
```

Nota. El cuadro presenta el reporte de temporización mínima, en el cual se verifica la condición de retención del diseño.

El reporte evalúa las tres rutas más rápidas del diseño para verificar la condición de retención (*hold*). A diferencia del análisis de *setup*, aquí se usa `-delay_type min` para calcular el retardo mínimo y garantizar que los datos no lleguen demasiado pronto a los *flip-flops*

de destino. Todas las rutas analizadas presentan `slack (MET)` positivo, lo que confirma la ausencia de violaciones de retención bajo las condiciones de peor caso (`WCCOM`) y la librería `tcbn65gpluswc`. Este resultado valida la estabilidad temporal interna del reloj tras la síntesis topográfica.

7.4.3. Ejecución del comando `report_qor`

Cuadro 39. Reporte de calidad de resultados (QoR).

```
dcnxt_shell-topo> report_qor

****************************************
Report : qor
Design : nanochip
Version: W-2024.09-SP4
Date   : Sat Nov  8 18:50:40 2025
****************************************

  Timing Path Group 'clk'
  -----------------------------------
  Levels of Logic:              12.00
  Critical Path Length:          1.06
  Critical Path Slack:           0.00
  Critical Path Clk Period:      1.20
  Total Negative Slack:          0.00
  No. of Violating Paths:        0.00
  Worst Hold Violation:          0.00
  Total Hold Violation:          0.00
  No. of Hold Violations:        0.00
  -----------------------------------

  Timing Path Group 'INPUTS'
  -----------------------------------
  Levels of Logic:              15.00
  Critical Path Length:          0.84
  Critical Path Slack:           0.00
  Critical Path Clk Period:      1.20
  Total Negative Slack:          0.00
  No. of Violating Paths:        0.00
  Worst Hold Violation:          0.00
  Total Hold Violation:          0.00
  No. of Hold Violations:        0.00
  -----------------------------------

  Timing Path Group 'OUTPUTS'
  -----------------------------------
  Levels of Logic:               9.00
  Critical Path Length:          1.80
  Critical Path Slack:          -0.90
  Critical Path Clk Period:      1.20
  Total Negative Slack:         -6.26
  No. of Violating Paths:        7.00
  Worst Hold Violation:          0.00
  Total Hold Violation:          0.00
  No. of Hold Violations:        0.00
  -----------------------------------

```

```
  Cell Count
  ----------------------------------
  Hierarchical Cell Count:          0
  Hierarchical Port Count:          0
  Leaf Cell Count:               4970
  Buf/Inv Cell Count:             963
  Buf Cell Count:                 145
  Inv Cell Count:                 818
  CT Buf/Inv Cell Count:            0
  Combinational Cell Count:      4744
  Sequential Cell Count:          226
  Macro Count:                      0
  ----------------------------------

  Area
  ----------------------------------
  Combinational Area:    10343.160156
  Noncombinational Area:  2186.279993
  Buf/Inv Area:           1535.760040
  Total Buffer Area:           485.28
  Total Inverter Area:        1050.48
  Macro/Black Box Area:      0.000000
  Net Area:                  0.000000
  Net XLength        :       66512.47
  Net YLength        :       65983.62
  ----------------------------------
  Cell Area:             12529.440149
  Design Area:           12529.440149
  Net Length        :       132496.09

  Design Rules
  ----------------------------------
  Total Number of Nets:          4975
  Nets With Violations:             7
  Max Trans Violations:             7
  Max Cap Violations:               7
  ----------------------------------

  Hostname: localhost.localdomain

  Compile CPU Statistics
  ---------------------------------------
  Resource Sharing:                  25.75
  Logic Optimization:               309.82
  Mapping Optimization:              81.80
  ---------------------------------------
  Overall Compile Time:             430.57
  Overall Compile Wall Clock Time:   62.76
  --------------------------------------------------------------------
  Design  WNS: 0.90  TNS: 6.26  Number of Violating Paths: 7

  Design (Hold)  WNS: 0.00  TNS: 0.00  Number of Violating Paths: 0
  --------------------------------------------------------------------
1
dcnxt_shell-topo>
```

Nota. El cuadro presenta el reporte global de calidad de resultados del diseño sintetizado, incluyendo métricas de temporización, área, potencia y reglas de diseño.

El reporte de calidad de resultados, o *quality of results* (QoR), resume los indicadores globales de rendimiento, área, reglas de diseño y tiempos del circuito sintetizado. El diseño `nanochip` presenta un total de 4970 celdas, con un área total de $12\,529.44\,\mu m^2$, en la que predomina la lógica combinacional sobre la secuencial. El grupo de rutas `OUTPUTS` muestra una violación de temporización con un *worst negative slack* (WNS) de -0.90 ns y un *total negative slack* (TNS) acumulado de -6.26 ns, distribuidos en siete rutas.

Además, se reportan siete violaciones eléctricas por transición y capacitancia máximas, asociadas principalmente a las redes de salida. Estas desviaciones son esperables en la etapa de síntesis topográfica, ya que la optimización de interconexiones y el cumplimiento estricto de reglas físicas ocurren en la siguiente fase del flujo de implementación, con uso de IC Compiler II.

En IC Compiler II, las etapas de *placement*, *clock tree synthesis* y *route optimization* ajustan las longitudes de red, la inserción de *buffers* y la carga de salida, lo que permite cerrar la temporización y eliminar las violaciones restantes. Por lo tanto, este reporte confirma que el diseño sintetizado cumple parcialmente los objetivos de área y retención, mientras que las violaciones de salida se corregirán en la fase física del flujo.

7.4.4. Ejecución del comando `report_area`

Cuadro 40. Reporte de distribución de área del diseño.

```
dcnxt_shell-topo> report_area -hierarchy

****************************************
Report : area
Design : nanochip
Version: W-2024.09-SP4
Date   : Sat Nov  8 18:55:14 2025
****************************************

Library(s) Used:

    tcbn65gpluswc (File: /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
    Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/stclib_tcbn65gpluswc.db)

Number of ports:                           14
Number of nets:                          4975
Number of cells:                         4970
Number of combinational cells:           4744
Number of sequential cells:               226
Number of macros/black boxes:               0
Number of buf/inv:                        963
Number of references:                     148

Combinational area:              10343.160156
Buf/Inv area:                     1535.760040
Noncombinational area:            2186.279993
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 12529.440149
Total area:                 undefined
```

```
32
33 Hierarchical area distribution
34 ------------------------------
35
36                                 Global cell area           Local cell area
37                                 ------------------  ------------------------------
38 Hierarchical cell                Absolute    Percent  Combi-      Noncombi-  Black-
39                                 Total       Total    national    national   boxes
       Design
40 -------------------------------  ----------  -------  ----------  ---------  ------
       ---------
41 nanochip                        12529.4401    100.0  10343.1602  2186.2800  0.0000
       nanochip
42 -------------------------------  ----------  -------  ----------  ---------  ------
       ---------
43 Total                                                10343.1602  2186.2800  0.0000
44
45 1
46 dcnxt_shell-topo>
```

Nota. El cuadro presenta el reporte jerárquico de distribución de área del diseño sintetizado.

El reporte de área jerárquica muestra la distribución física resultante de la síntesis topográfica del diseño `nanochip`. El total de 4970 celdas ocupa un área combinacional de $10\,343.16\,\mu m^2$ y un área secuencial de $2\,186.28\,\mu m^2$, con lo cual se alcanza un área total sintetizada de $12\,529.44\,\mu m^2$. La librería empleada es `tcbn65gpluswc`, correspondiente al nodo de 65 nm en condiciones de peor caso (*worst corner*).

No se reportan macros ni bloques jerárquicos, por lo que la distribución de área es completamente plana. El área de interconexión (*net area*) aparece como indefinida, dado que el modelo de carga (*wire load*) no asigna una longitud efectiva a las redes en esta etapa.

Este resultado confirma la correcta integración de las celdas lógicas y secuenciales, y sirve como base para la etapa de colocación física en IC Compiler II, en la cual se calculará el área real de interconexiones y metalización. Adicionalmente, el alto porcentaje de área combinacional refleja que el diseño mantiene una arquitectura optimizada para velocidad y lógica combinada, lo que resulta coherente con el flujo de síntesis topográfica aplicado.

7.4.5. Ejecución del comando `report_power`

Cuadro 41. Reporte de consumo de potencia del diseño sintetizado.

```
1  dcnxt_shell-topo> report_power -analysis_effort medium
2  Information: Propagating switching activity (medium effort zero delay simulation).
       (PWR-6)
3  Warning: Design has unannotated primary inputs. (PWR-414)
4  Warning: Design has unannotated sequential cell outputs. (PWR-415)
5
6  ****************************************
7  Report : power
8          -analysis_effort medium
9  Design : nanochip
10 Version: W-2024.09-SP4
11 Date   : Sat Nov  8 18:56:51 2025
```

```
****************************************
Library(s) Used:

    tcbn65gpluswc (File: /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/
    Lab_DCNXT/DCNXT_2021.06/EGJ-Jose/ref/DBs/stclib_tcbn65gpluswc.db)

Operating Conditions: WCCOM   Library: tcbn65gpluswc
Wire Load Model Mode: Inactive.

Global Operating Voltage = 0.9
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW

Attributes
----------
i - Including register clock pin internal power

Warning: Cannot report correlated power unless power prediction mode is set. (PWR-727)
Power Breakdown
---------------
                            Cell        Driven Net  Tot Dynamic      Cell
                            Internal    Switching   Power (mW)       Leakage
Cell                        Power (mW)  Power (mW)  (% Cell/Tot)     Power (nW)
-------------------------------------------------------------------------------
Netlist Power                  2.3136      3.4591   5.773e+00 (40%)  1.045e+05
Estimated Clock Tree Power  N/A         N/A         N/A (N/A)        N/A
-------------------------------------------------------------------------------

            Internal         Switching           Leakage            Total
Power Group   Power            Power               Power              Power   (   %
    )  Attrs
-----------------------------------------------------------------------------------------
io_pad          0.0000            0.0000            0.0000            0.0000  (
    0.00%)
memory          0.0000            0.0000            0.0000            0.0000  (
    0.00%)
black_box       0.0000            0.0000            0.0000            0.0000  (
    0.00%)
clock_network   1.5001            0.0000            0.0000            1.5001  (
    25.52%)  i
register        0.1364        4.5171e-02        1.5820e+04            0.1974  (
    3.36%)
sequential      0.0000            0.0000            0.0000            0.0000  (
    0.00%)
combinational   0.6771            3.4139        8.8547e+04            4.1796  (
    71.12%)
-----------------------------------------------------------------------------------------
Total              2.3136 mW         3.4591 mW     1.0437e+05 nW         5.8770 mW
1
dcnxt_shell-topo>
```

Nota. El cuadro presenta el reporte de potencia estimada del diseño sintetizado bajo condiciones de peor caso.

El reporte de potencia cuantifica el consumo total del diseño `nanochip` bajo condiciones de peor caso (`WCCOM`) y con una tensión de operación de 0.9 V. La simulación de actividad se ejecutó con esfuerzo medio (`medium effort`) y sin vectores de entrada, o *vectorless mode*, lo que produce advertencias por entradas y salidas no anotadas, situación propia de un análisis predictivo previo al modelado funcional completo.

El consumo total se estima en 5.877 mW, compuesto por 2.3136 mW de potencia interna, 3.4591 mW de potencia de conmutación y 1.0437×10^5 nW de fuga. La red de reloj (*clock network*) representa el 25.5 % del consumo total, mientras que la lógica combinacional concentra el 71.1 %, lo que refleja la alta densidad de celdas lógicas frente al número reducido de registros.

La potencia reportada es estimativa y se refinará durante la etapa de implementación física en IC Compiler II, donde la red de reloj será sintetizada y la capacitancia de interconexión se calculará con mayor precisión. En esa fase, las herramientas de análisis estático o dinámico permitirán ajustar el modelo de consumo con base en actividad real y en la distribución espacial de las celdas.

7.4.6. Ejecución del comando `report_constraints`

Cuadro 42. Reporte de violaciones a restricciones temporales y eléctricas.

```
dcnxt_shell-topo> report_constraints -all_violators -verbose

****************************************
Report : constraint
        -all_violators
        -verbose
Design : nanochip
Version: W-2024.09-SP4
Date   : Sat Nov  8 18:58:47 2025
****************************************

  Startpoint: clk_r_REG57_S2
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: q_out[3] (output port clocked by clk)
  Path Group: OUTPUTS
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  clk_r_REG57_S2/CP (SDFQD4)               0.00       0.00 r
  clk_r_REG57_S2/Q (SDFQD4)                0.14       0.14 r
  U4756/ZN (OAI221XD4)                     0.07 *     0.21 f
  U4757/ZN (OAI211D2)                      0.04 *     0.24 r
  U4760/ZN (NR2D2)                         0.02 *     0.26 f
  U4761/ZN (NR2XD1)                        0.02 *     0.28 r
  U4544/ZN (AOI21D2)                       0.03 *     0.31 f
  U4768/ZN (CKND2)                         0.02 *     0.33 r
  U3077/ZN (ND2D4)                         0.03 *     0.36 f
  U3843/ZN (INVD6)                         0.02 *     0.38 r
  U4679/Z (BUFFD16)                        1.40 *     1.78 r
```

```
  q_out[3] (out)                              0.02 *     1.80 r
  data arrival time                                      1.80

  clock clk (rise edge)                       1.20       1.20
  clock network delay (ideal)                 0.00       1.20
  clock uncertainty                          -0.10       1.10
  output external delay                      -0.20       0.90
  data required time                                     0.90
  -----------------------------------------------------------
  data required time                                     0.90
  data arrival time                                     -1.80
  -----------------------------------------------------------
  slack (VIOLATED)                                      -0.90

  Startpoint: clk_r_REG187_S3
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: q_out[6] (output port clocked by clk)
  Path Group: OUTPUTS
  Path Type: max

  Point                                       Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                       0.00       0.00
  clock network delay (ideal)                 0.00       0.00
  clk_r_REG187_S3/CP (SDFQD4)                 0.00       0.00 r
  clk_r_REG187_S3/Q (SDFQD4)                  0.14       0.14 r
  U3072/ZN (ND2D3)                            0.03 *     0.16 f
  U3167/ZN (CKND2)                            0.02 *     0.18 r
  U3168/ZN (AOI211D2)                         0.03 *     0.21 f
  U3243/ZN (ND2D2)                            0.02 *     0.23 r
  U3226/ZN (ND2D2)                            0.02 *     0.25 f
  U4361/ZN (AOI211XD2)                        0.05 *     0.31 r
  U3999/ZN (IND3D4)                           0.06 *     0.37 r
  U4682/Z (BUFFD16)                           1.41 *     1.78 r
  q_out[6] (out)                              0.02 *     1.80 r
  data arrival time                                      1.80

  clock clk (rise edge)                       1.20       1.20
  clock network delay (ideal)                 0.00       1.20
  clock uncertainty                          -0.10       1.10
  output external delay                      -0.20       0.90
  data required time                                     0.90
  -----------------------------------------------------------
  data required time                                     0.90
  data arrival time                                     -1.80
  -----------------------------------------------------------
  slack (VIOLATED)                                      -0.90

  Startpoint: clk_r_REG57_S2
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: q_out[1] (output port clocked by clk)
  Path Group: OUTPUTS
  Path Type: max

  Point                                       Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                       0.00       0.00
  clock network delay (ideal)                 0.00       0.00
  clk_r_REG57_S2/CP (SDFQD4)                  0.00       0.00 r
```

```
  clk_r_REG57_S2/Q (SDFQD4)                0.14       0.14 r
  U3112/ZN (AOI32D2)                       0.05 *     0.19 f
  U3115/ZN (AOI21D2)                       0.04 *     0.23 r
  U3116/ZN (AOI22D1)                       0.04 *     0.27 f
  U3131/ZN (IOA21D2)                       0.06 *     0.34 f
  U4547/ZN (INVD4)                         0.03 *     0.37 r
  U4678/Z (BUFFD16)                        1.40 *     1.77 r
  q_out[1] (out)                           0.03 *     1.80 r
  data arrival time                                   1.80

  clock clk (rise edge)                    1.20       1.20
  clock network delay (ideal)              0.00       1.20
  clock uncertainty                       -0.10       1.10
  output external delay                   -0.20       0.90
  data required time                                  0.90
  ----------------------------------------------------------
  data required time                                  0.90
  data arrival time                                  -1.80
  ----------------------------------------------------------
  slack (VIOLATED)                                   -0.90

  Startpoint: clk_r_REG57_S2
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: q_out[4] (output port clocked by clk)
  Path Group: OUTPUTS
  Path Type: max

  Point                                    Incr       Path
  ----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  clk_r_REG57_S2/CP (SDFQD4)               0.00       0.00 r
  clk_r_REG57_S2/Q (SDFQD4)                0.13       0.13 f
  U3737/Z (BUFFD4)                         0.05 *     0.18 f
  U4542/ZN (ND2D2)                         0.02 *     0.20 r
  U4540/ZN (IND3D4)                        0.04 *     0.24 f
  U4539/ZN (CKND2)                         0.02 *     0.26 r
  U3177/ZN (NR2D2)                         0.02 *     0.27 f
  U4645/ZN (OAI21D4)                       0.03 *     0.31 r
  U4644/ZN (OAI21D4)                       0.04 *     0.34 f
  U835/ZN (INVD6)                          0.03 *     0.37 r
  U4680/Z (BUFFD16)                        1.40 *     1.77 r
  q_out[4] (out)                           0.02 *     1.80 r
  data arrival time                                   1.80

  clock clk (rise edge)                    1.20       1.20
  clock network delay (ideal)              0.00       1.20
  clock uncertainty                       -0.10       1.10
  output external delay                   -0.20       0.90
  data required time                                  0.90
  ----------------------------------------------------------
  data required time                                  0.90
  data arrival time                                  -1.80
  ----------------------------------------------------------
  slack (VIOLATED)                                   -0.90

  Startpoint: clk_r_REG187_S3
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: q_out[5] (output port clocked by clk)
```

```
  Path Group: OUTPUTS
  Path Type: max

  Point                                     Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                     0.00       0.00
  clock network delay (ideal)               0.00       0.00
  clk_r_REG187_S3/CP (SDFQD4)               0.00       0.00 r
  clk_r_REG187_S3/Q (SDFQD4)                0.13       0.13 f
  U3160/ZN (INVD1)                          0.03 *     0.16 r
  U3159/ZN (ND2D2)                          0.03 *     0.19 f
  U3158/ZN (ND3D3)                          0.03 *     0.22 r
  U4562/ZN (AOI22D2)                        0.04 *     0.25 f
  U3341/ZN (CKND2)                          0.02 *     0.27 r
  U3788/ZN (NR2D2)                          0.02 *     0.29 f
  U3936/ZN (IND2D4)                         0.05 *     0.34 f
  U4625/ZN (ND2D4)                          0.03 *     0.37 r
  U4681/Z (BUFFD16)                         1.40 *     1.77 r
  q_out[5] (out)                            0.02 *     1.79 r
  data arrival time                                    1.79

  clock clk (rise edge)                     1.20       1.20
  clock network delay (ideal)               0.00       1.20
  clock uncertainty                        -0.10       1.10
  output external delay                    -0.20       0.90
  data required time                                   0.90
  -----------------------------------------------------------
  data required time                                   0.90
  data arrival time                                   -1.79
  -----------------------------------------------------------
  slack (VIOLATED)                                    -0.89

  Startpoint: clk_r_REG57_S2
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: q_out[0] (output port clocked by clk)
  Path Group: OUTPUTS
  Path Type: max

  Point                                     Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                     0.00       0.00
  clock network delay (ideal)               0.00       0.00
  clk_r_REG57_S2/CP (SDFQD4)                0.00       0.00 r
  clk_r_REG57_S2/Q (SDFQD4)                 0.13       0.13 f
  U3610/Z (BUFFD3)                          0.06 *     0.19 f
  U3208/ZN (OAI21D1)                        0.04 *     0.23 r
  U3210/ZN (ND3D1)                          0.05 *     0.28 f
  U3738/ZN (NR2XD1)                         0.03 *     0.31 r
  U3069/ZN (NR2D2)                          0.02 *     0.33 f
  U294/ZN (INVD6)                           0.02 *     0.36 r
  U4684/Z (BUFFD16)                         1.40 *     1.76 r
  q_out[0] (out)                            0.03 *     1.79 r
  data arrival time                                    1.79

  clock clk (rise edge)                     1.20       1.20
  clock network delay (ideal)               0.00       1.20
  clock uncertainty                        -0.10       1.10
  output external delay                    -0.20       0.90
  data required time                                   0.90
```

```
  ------------------------------------------------------------
  data required time                                   0.90
  data arrival time                                   -1.79
  ------------------------------------------------------------
  slack (VIOLATED)                                    -0.89

  Startpoint: clk_r_REG7_S2
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: q_out[2] (output port clocked by clk)
  Path Group: OUTPUTS
  Path Type: max

  Point                                    Incr       Path
  ------------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  clk_r_REG7_S2/CP (SDFQD4)                0.00       0.00 r
  clk_r_REG7_S2/Q (SDFQD4)                 0.13       0.13 f
  U3082/ZN (AOI21D1)                       0.05 *     0.18 r
  U3154/ZN (ND2D1)                         0.03 *     0.21 f
  U3134/ZN (INVD1)                         0.02 *     0.23 r
  U3133/ZN (NR2D1)                         0.02 *     0.25 f
  U3132/ZN (OAI21D2)                       0.03 *     0.28 r
  U3130/ZN (ND3D3)                         0.05 *     0.33 f
  U3207/ZN (INVD6)                         0.03 *     0.36 r
  U4683/Z (BUFFD16)                        1.40 *     1.76 r
  q_out[2] (out)                           0.02 *     1.78 r
  data arrival time                                   1.78

  clock clk (rise edge)                    1.20       1.20
  clock network delay (ideal)              0.00       1.20
  clock uncertainty                       -0.10       1.10
  output external delay                   -0.20       0.90
  data required time                                  0.90
  ------------------------------------------------------------
  data required time                                  0.90
  data arrival time                                  -1.78
  ------------------------------------------------------------
  slack (VIOLATED)                                   -0.88

    Net: q_out[3]

    max_transition         0.56
  - Transition Time        2.61
  -----------------------------
    Slack                -2.05  (VIOLATED)

    List of pins on net "q_out[3]" with transition violations :
   ------------------------------------------------------------------
                             Required        Actual
                            Transition     Transition       Slack
   ------------------------------------------------------------------
       PIN :   U4679/Z          0.56           2.61          -2.05  (VIOLATED)

    Net: q_out[0]

    max_transition         0.56
  - Transition Time        2.61
  -----------------------------
```

```
  Slack                 -2.05  (VIOLATED)

  List of pins on net "q_out[0]" with transition violations :
 ---------------------------------------------------------------
                          Required        Actual
                         Transition     Transition       Slack
 ---------------------------------------------------------------
     PIN :   U4684/Z          0.56           2.61          -2.05  (VIOLATED)

  Net: q_out[4]

  max_transition         0.56
- Transition Time        2.61
 ------------------------------
  Slack                 -2.05  (VIOLATED)

  List of pins on net "q_out[4]" with transition violations :
 ---------------------------------------------------------------
                          Required        Actual
                         Transition     Transition       Slack
 ---------------------------------------------------------------
     PIN :   U4680/Z          0.56           2.61          -2.05  (VIOLATED)

  Net: q_out[2]

  max_transition         0.56
- Transition Time        2.61
 ------------------------------
  Slack                 -2.05  (VIOLATED)

  List of pins on net "q_out[2]" with transition violations :
 ---------------------------------------------------------------
                          Required        Actual
                         Transition     Transition       Slack
 ---------------------------------------------------------------
     PIN :   U4683/Z          0.56           2.61          -2.05  (VIOLATED)

  Net: q_out[1]

  max_transition         0.56
- Transition Time        2.61
 ------------------------------
  Slack                 -2.05  (VIOLATED)

  List of pins on net "q_out[1]" with transition violations :
 ---------------------------------------------------------------
                          Required        Actual
                         Transition     Transition       Slack
 ---------------------------------------------------------------
     PIN :   U4678/Z          0.56           2.61          -2.05  (VIOLATED)

  Net: q_out[5]

  max_transition         0.56
- Transition Time        2.61
 ------------------------------
  Slack                 -2.05  (VIOLATED)

  List of pins on net "q_out[5]" with transition violations :
```

```
  ------------------------------------------------------------------
                           Required        Actual
                          Transition     Transition       Slack
  ------------------------------------------------------------------
      PIN :   U4681/Z          0.56           2.61          -2.05  (VIOLATED)

   Net: q_out[6]

   max_transition         0.56
 - Transition Time        2.61
 -----------------------------
   Slack                -2.05  (VIOLATED)

   List of pins on net "q_out[6]" with transition violations :
  ------------------------------------------------------------------
                           Required        Actual
                          Transition     Transition       Slack
  ------------------------------------------------------------------
      PIN :   U4682/Z          0.56           2.61          -2.05  (VIOLATED)

   Net: q_out[0]

   max_capacitance        1.05
 - Capacitance            5.00
 -----------------------------
   Slack                -3.95  (VIOLATED)

   Net: q_out[1]

   max_capacitance        1.05
 - Capacitance            5.00
 -----------------------------
   Slack                -3.95  (VIOLATED)

   Net: q_out[2]

   max_capacitance        1.05
 - Capacitance            5.00
 -----------------------------
   Slack                -3.95  (VIOLATED)

   Net: q_out[4]

   max_capacitance        1.05
 - Capacitance            5.00
 -----------------------------
   Slack                -3.95  (VIOLATED)

   Net: q_out[5]

   max_capacitance        1.05
 - Capacitance            5.00
 -----------------------------
   Slack                -3.95  (VIOLATED)

   Net: q_out[6]

   max_capacitance        1.05
 - Capacitance            5.00
```

```
  ------------------------------
    Slack                 -3.95  (VIOLATED)

    Net: q_out[3]

    max_capacitance        1.05
  - Capacitance            5.00
  ------------------------------
    Slack                 -3.95  (VIOLATED)

    Net: out_ring_osc

    multiport_net          0.00
  - actual cost            1.00
  ------------------------------
    Violation             -1.00  (VIOLATED)

1
dcnxt_shell-topo>
```

Nota. El cuadro presenta el reporte detallado de violaciones de restricciones temporales y eléctricas detectadas tras la síntesis topográfica.

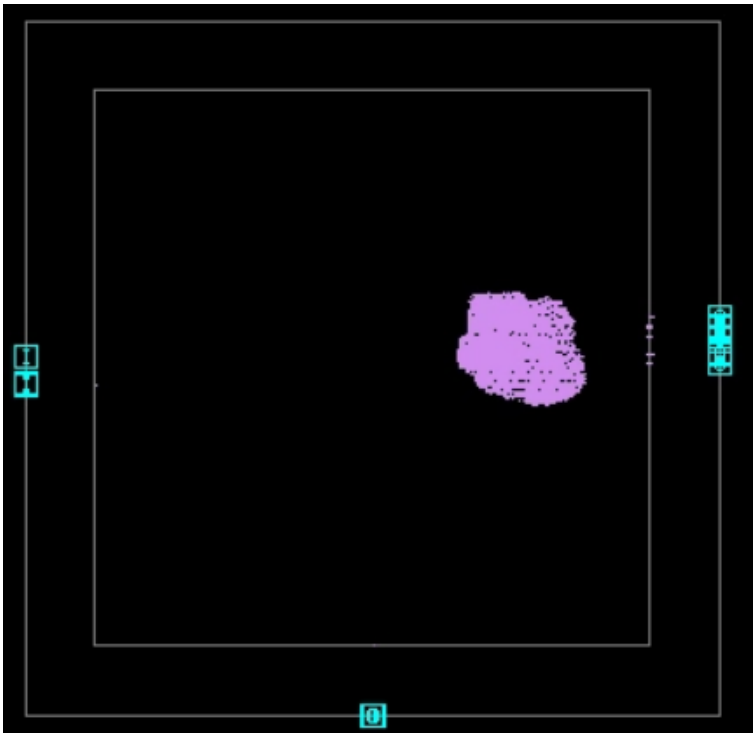
Los resultados del comando `report_constraints` muestran que, tras la síntesis topográfica en Design Compiler NXT, el diseño `nanochip` aún presenta violaciones de temporización y de tipo eléctrico en las salidas principales. Las violaciones identificadas, correspondientes a `max_transition` y `max_capacitance`, reflejan el comportamiento esperado en esta etapa del flujo, ya que el modelo utilizado en DCNXT no incorpora información física completa ni parásitos extraídos.

Debido a las limitaciones de licencia y el alcance del flujo implementado, el proceso de optimización física no puede ejecutarse dentro de DCNXT. Por consiguiente, las correcciones finales de transición, capacitancia y temporización se realizarán en IC Compiler II, durante las fases de *floorplanning*, *placement*, *clock tree synthesis* y *routing*.

En dichas etapas, el diseño se someterá a un análisis con parásitos reales, lo que permitirá ajustar *buffers*, redimensionar celdas y optimizar el enrutamiento, con lo cual se asegurará el cierre total de tiempo y la eliminación de violaciones eléctricas. De esta forma, los resultados de DCNXT se consideran satisfactorios dentro de su alcance, ya que entregan una *netlist* completamente sintetizada y funcionalmente equivalente, que servirá como base para la implementación física posterior en IC Compiler II.

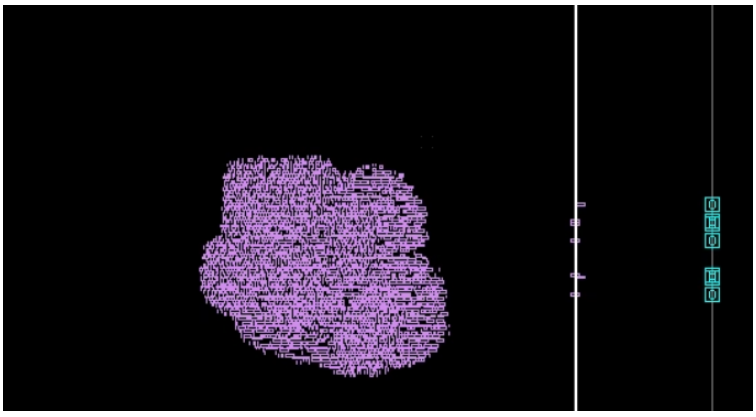
7.5. Resultados finales de la síntesis topográfica

Figura 3. Visualización del contorno del dado generado tras la síntesis topográfica en Design Compiler NXT.



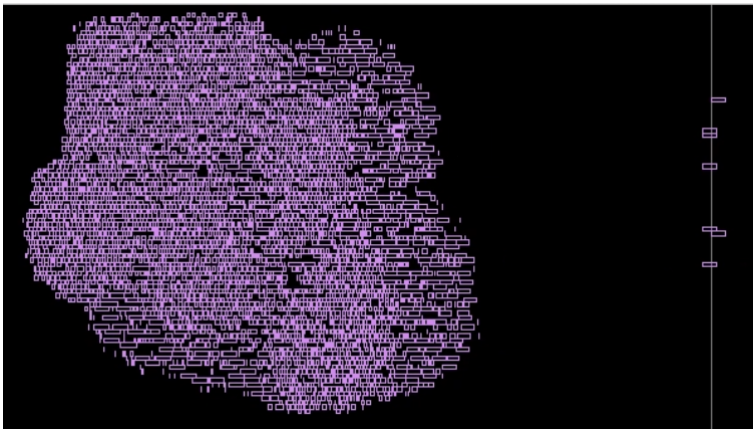
Nota. La figura presenta el contorno del dado estimado por Design Compiler NXT durante la síntesis topográfica. Esta visualización permite validar dimensiones preliminares, distribución del área y consistencia del diseño físico antes de pasar a etapas posteriores como *floorplanning* y *place & route*.

Figura 4. Distribución inicial de celdas y puertos de entrada y salida del diseño `nanochip`.



Nota. La figura muestra la distribución preliminar de celdas estándar y la ubicación inicial de los puertos de entrada y salida generadas por Design Compiler NXT durante la síntesis topográfica. Esta información permite evaluar la coherencia del posicionamiento lógico-físico antes de pasar a un *floorplanning* definitivo.

Figura 5. Detalle de la ubicación de celdas lógicas sintetizadas dentro del área activa.



Nota. La figura muestra la colocación preliminar de las celdas lógicas generadas durante la síntesis topográfica. Esta visualización permite inspeccionar la densidad, distribución y consistencia espacial del diseño antes de las etapas formales de *place & route*.

Los resultados obtenidos a partir de la síntesis topográfica muestran la correcta generación del área del dado (*die area*) y la ubicación preliminar de las celdas lógicas del diseño `nanochip`. En la Figura 3 se observa la definición del contorno del chip junto con los puertos perimetrales, mientras que las Figuras 4 y 5 ilustran la distribución interna de las celdas y la concentración lógica dentro de la región activa.

Esta distribución corresponde a la estimación que realiza automáticamente el motor de síntesis topográfica de Design Compiler NXT, el cual aproxima la ocupación de área y la posición de las celdas a partir de los modelos de la librería estándar `tcbn65gpluswc`, sin contar aún con información física completa ni parásitos extraídos.

Debido a las limitaciones del entorno de licencia y al alcance del flujo utilizado, el proceso concluye en esta etapa dentro de DCNXT. Las fases posteriores de implementación física, incluidas *floorplanning*, síntesis del árbol de reloj, enrutamiento y cierre definitivo de tiempo, se llevarán a cabo en IC Compiler II.

En este contexto, la disposición de celdas representada constituye una aproximación preliminar del *floorplan* final, la cual servirá como referencia para la optimización física y la validación temporal durante la etapa de implementación.

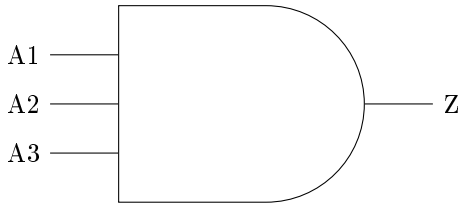
CAPÍTULO 8

Pruebas finales LVS e ERC

8.1. Celdas

8.1.1. AND de tres entradas

Figura 6. Esquemático de la celda AN3D0.



Nota. La figura muestra el esquema lógico de la celda AN3D0 como una compuerta AND de tres entradas. Esta representación permite verificar la correspondencia entre las señales de entrada y la salida lógica de la celda.

Cuadro 43. Tabla de verdad de la celda AN3D0.

INPUT			OUTPUT
A1	A2	A3	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Nota. La tabla presenta la relación lógica entre las tres entradas de la celda AN3D0 y su salida correspondiente.

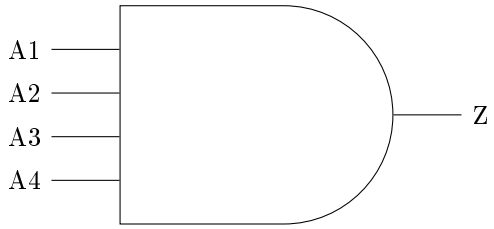
Cuadro 44. Código Verilog de la celda AN3D0.

```
1 module AN3D0 (input A1, input A2, input A3, output Z);
2     assign Z = A1 & A2 & A3;
3 endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AN3D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.2. AND de cuatro entradas

Figura 7. Esquemático de la celda AN4D0.



Nota. La figura muestra el esquema lógico de la celda AN4D0 como una compuerta AND de cuatro entradas. Esta representación permite verificar la correspondencia entre las señales de entrada y la salida lógica de la celda.

Cuadro 45. Tabla de verdad de la celda AN4D0.

INPUT				OUTPUT
A1	A2	A3	A4	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Nota. La tabla presenta la relación lógica entre las cuatro entradas de la celda AN4D0 y su salida correspondiente.

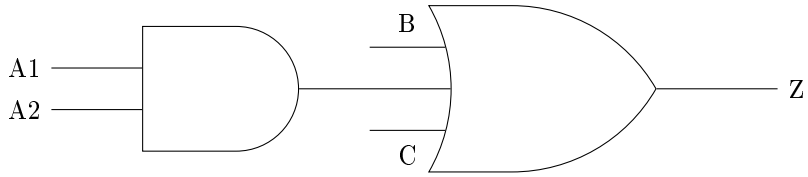
Cuadro 46. Código Verilog de la celda AN4D0.

```
1 module AN4D0 (input A1, input A2, input A3, input A4, output Z);
2     assign Z = A1 & A2 & A3 & A4;
3 endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AN4D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.3. AND de dos entradas y OR de tres entradas

Figura 8. Esquemático de la celda AO211D0.



Nota. La figura muestra el esquema lógico de la celda AO211D0. La salida corresponde a la operación lógica entre una compuerta AND de dos entradas y una compuerta OR de tres entradas.

Cuadro 47. Tabla de verdad de la celda AO211D0.

INPUT				OUTPUT
A1	A2	B	C	Z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Nota. La tabla presenta la relación lógica entre las entradas A1, A2, B y C de la celda AO211D0 y su salida correspondiente.

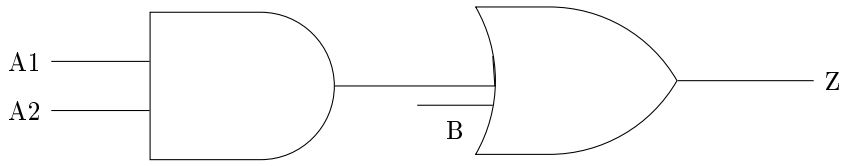
Cuadro 48. Código Verilog de la celda AO211D0.

```
module AO211D0 (
    input  A1,
    input  A2,
    input  B,
    input  C,
    output Z
);
    assign Z = (A1 & A2) | B | C;
endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AO211D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.4. AND de dos entradas y OR de dos entradas

Figura 9. Esquemático de la celda AO21D0.



Nota. La figura muestra el esquema lógico de la celda AO21D0. La salida corresponde a la operación lógica entre una compuerta AND de dos entradas y una compuerta OR de dos entradas.

Cuadro 49. Tabla de verdad de la celda AO21D0.

INPUT			OUTPUT
A1	A2	B	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Nota. La tabla presenta la relación lógica entre las entradas A1, A2 y B de la celda AO21D0 y su salida correspondiente.

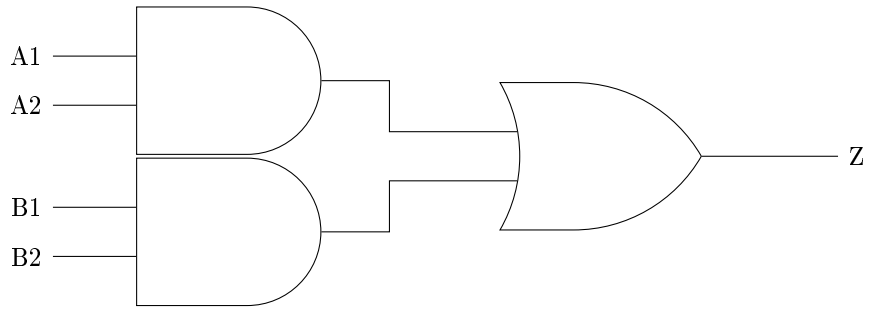
Cuadro 50. Código Verilog de la celda AO21D0.

```
1 module AO21D0 (
2     input  A1,
3     input  A2,
4     input  B,
5     output Z
6 );
7     assign Z = (A1 & A2) | B;
8 endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AO21D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.5. Dos AND de dos entradas seguidas de un OR de dos entradas

Figura 10. Esquemático de la celda AO22D0.



Nota. La figura muestra el esquema lógico de la celda AO22D0. La salida corresponde a la operación OR entre los resultados de dos compuertas AND de dos entradas.

Cuadro 51. Tabla de verdad de la celda AO22D0.

INPUT				OUTPUT
A1	A2	B1	B2	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Nota. La tabla presenta la relación lógica entre las entradas A1, A2, B1 y B2 de la celda AO22D0 y su salida correspondiente.

Cuadro 52. Código Verilog de la celda AO22D0.

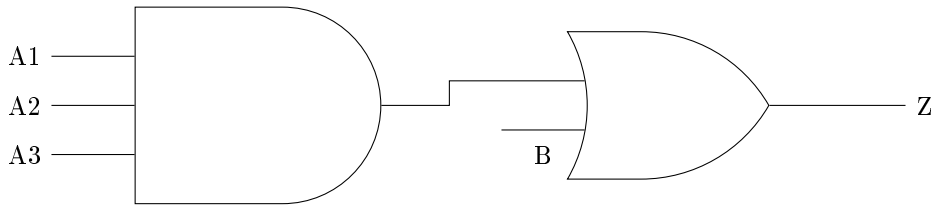
```
1 module AO22D0 (
2     input  A1,
3     input  A2,
4     input  B1,
5     input  B2,
```

```
6     output Z
7 );
8     assign Z = (A1 & A2) | (B1 & B2);
9 endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AO22D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.6. AND de tres entradas y OR de dos entradas

Figura 11. Esquemático de la celda AO31D0.



Nota. La figura muestra el esquema lógico de la celda AO31D0. La salida corresponde a la operación OR entre el resultado de una compuerta AND de tres entradas y la señal B.

Cuadro 53. Tabla de verdad de la celda AO31D0.

INPUT				OUTPUT
A1	A2	A3	B	Z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Nota. La tabla presenta la relación lógica entre las entradas A1, A2, A3 y B de la celda AO31D0 y su salida correspondiente.

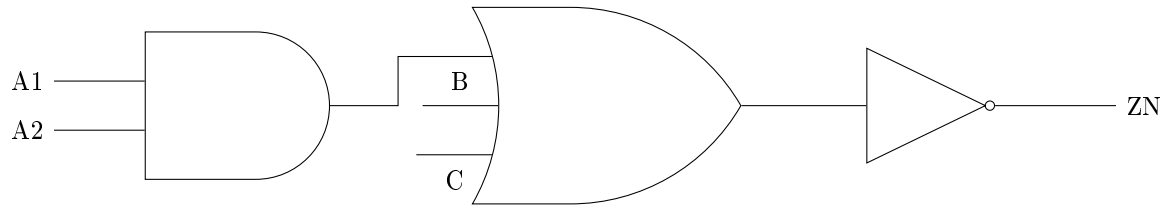
Cuadro 54. Código Verilog de la celda AO31D0.

```
module AO31D0 (
    input  A1,
    input  A2,
    input  A3,
    input  B,
    output Z
);
    assign Z = (A1 & A2 & A3) | B;
endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AO31D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.7. AND de dos entradas y OR de tres entradas con salida negada

Figura 12. Esquemático de la celda AOI211D0.



Nota. La figura muestra el esquema lógico de la celda AOI211D0. La salida negada corresponde a la inversión de la operación OR entre el resultado de una compuerta AND de dos entradas y las señales B y C.

Cuadro 55. Tabla de verdad de la celda AOI211D0.

INPUT				OUTPUT
A1	A2	B	C	ZN
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Nota. La tabla presenta la relación lógica entre las entradas A1, A2, B y C de la celda AOI211D0 y su salida negada correspondiente.

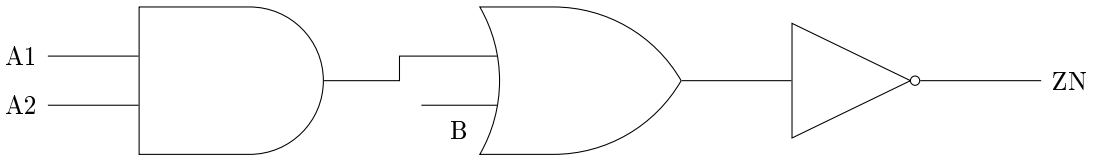
Cuadro 56. Código Verilog de la celda AOI211D0.

```
module AOI211D0 (
    input  A1,
    input  A2,
    input  B,
    input  C,
    output ZN
);
    assign ZN = ~((A1 & A2) | B | C);
endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AOI211D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.8. AND de dos entradas y OR de dos entradas con salida negada

Figura 13. Esquemático de la celda AOI21D0.



Nota. La figura muestra el esquema lógico de la celda AOI21D0. La salida negada corresponde a la inversión de la operación OR entre el resultado de una compuerta AND de dos entradas y la señal B.

Cuadro 57. Tabla de verdad de la celda AOI21D0.

INPUT			OUTPUT
A1	A2	B	ZN
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Nota. La tabla presenta la relación lógica entre las entradas A1, A2 y B de la celda AOI21D0 y su salida negada correspondiente.

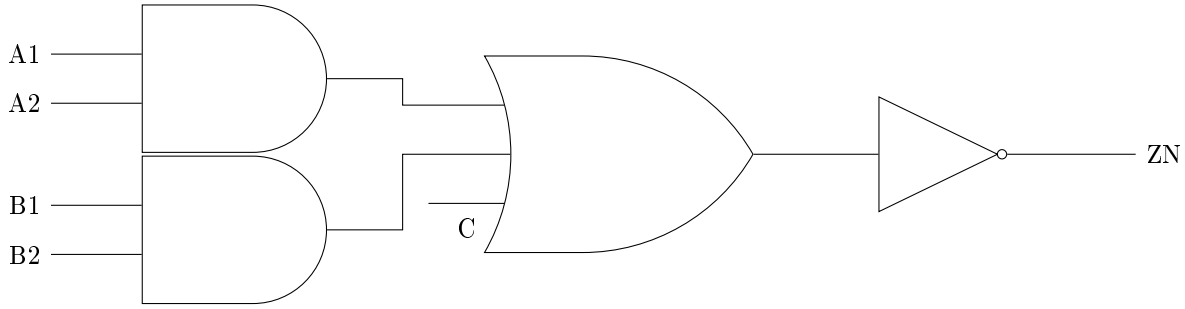
Cuadro 58. Código Verilog de la celda AOI21D0.

```
1 module AOI21D0 (
2     input  A1,
3     input  A2,
4     input  B,
5     output ZN
6 );
7     assign ZN = ~((A1 & A2) | B);
8 endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AOI21D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.9. Dos AND de dos entradas y OR de tres entradas con salida negada

Figura 14. Esquemático de la celda AOI221D0.



Nota. La figura muestra el esquema lógico de la celda AOI221D0. La salida negada corresponde a la inversión de la operación OR entre los resultados de dos compuertas AND de dos entradas y la señal C.

Cuadro 59. Tabla de verdad de la celda AOI221D0.

INPUT					OUTPUT
A1	A2	B1	B2	C	ZN
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	0	1	0
0	1	1	1	0	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	0	1	0
1	1	0	1	0	0
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	0	1	0
1	1	1	1	0	0
1	1	1	1	1	0

Nota. La tabla presenta la relación lógica entre las entradas A1, A2, B1, B2 y C de la celda AOI221D0 y su salida negada correspondiente.

Cuadro 60. Código Verilog de la celda AOI221D0.

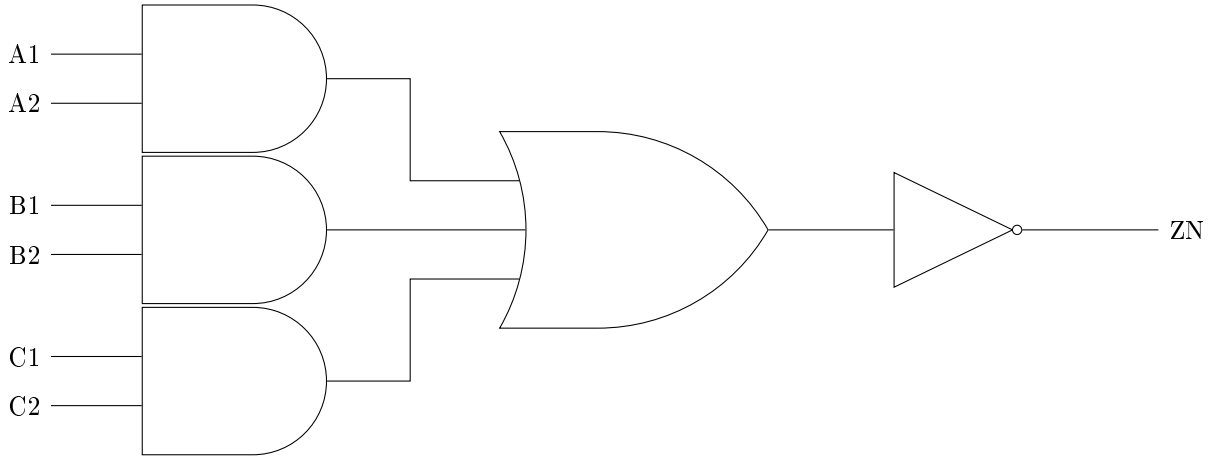
```
1 module AOI221D0 (
2     input  A1,
3     input  A2,
```

```
4       input  B1,
5       input  B2,
6       input  C,
7       output ZN
8   );
9       assign ZN = ~((A1 & A2) | (B1 & B2) | C);
10  endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AOI221D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.10. Tres AND de dos entradas y OR de tres entradas con salida negada

Figura 15. Esquemático de la celda AOI222D0.



Nota. La figura muestra el esquema lógico de la celda AOI222D0. La salida negada corresponde a la inversión de la operación OR entre los resultados de tres compuertas AND de dos entradas.

Cuadro 61. Tabla de verdad de la celda AOI222D0.

INPUT						OUTPUT
A1	A2	B1	B2	C1	C2	ZN
0	0	0	0	0	0	1
0	0	0	0	0	1	1
0	0	0	0	1	0	1
0	0	0	0	1	1	0
0	0	0	1	0	0	1
0	0	0	1	0	1	1
0	0	0	1	1	0	1
0	0	0	1	1	1	0
0	0	1	0	0	0	1
0	0	1	0	0	1	1

INPUT						OUTPUT
A1	A2	B1	B2	C1	C2	ZN
0	0	1	0	1	0	1
0	0	1	0	1	1	0
0	0	1	1	0	0	0
0	0	1	1	0	1	0
0	0	1	1	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	0	1
0	1	0	0	0	1	1
0	1	0	0	1	0	1
0	1	0	0	1	1	0
0	1	0	1	0	0	1
0	1	0	1	0	1	1
0	1	0	1	1	0	1
0	1	0	1	1	1	0
0	1	1	0	0	0	1
0	1	1	0	0	1	1
0	1	1	0	1	0	1
0	1	1	0	1	1	0
0	1	1	1	0	0	0
0	1	1	1	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	0	1
1	0	0	0	0	1	1
1	0	0	0	1	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	0	1	0	1	1
1	0	0	1	1	0	1
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	0	1	0	0	1	1
1	0	1	0	1	0	1
1	0	1	0	1	1	0
1	0	1	1	0	0	0
1	0	1	1	0	1	0
1	0	1	1	1	0	0
1	0	1	1	1	1	0
1	1	0	0	0	0	0
1	1	0	0	0	1	0
1	1	0	0	1	0	0

INPUT						OUTPUT
A1	A2	B1	B2	C1	C2	ZN
1	1	0	0	1	1	0
1	1	0	1	0	0	0
1	1	0	1	0	1	0
1	1	0	1	1	0	0
1	1	0	1	1	1	0
1	1	1	0	0	0	0
1	1	1	0	0	1	0
1	1	1	0	1	0	0
1	1	1	0	1	1	0
1	1	1	1	0	0	0
1	1	1	1	0	1	0
1	1	1	1	1	0	0
1	1	1	1	1	1	0

Nota. La tabla presenta la relación lógica entre las entradas A1, A2, B1, B2, C1 y C2 de la celda AOI222D0 y su salida negada correspondiente.

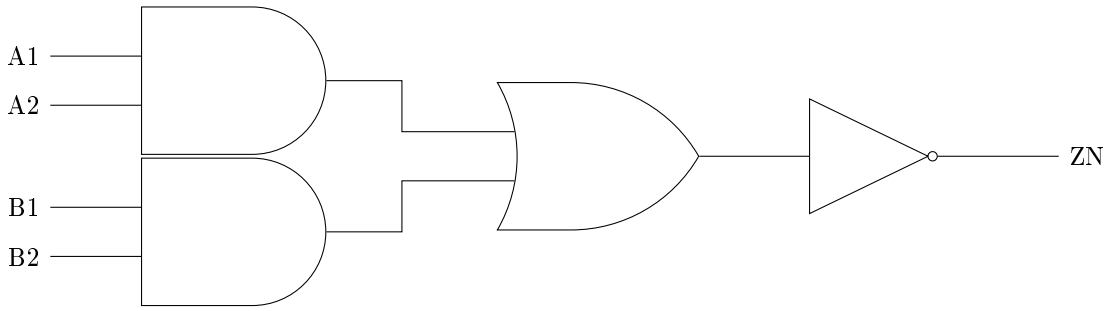
Cuadro 62. Código Verilog de la celda AOI222D0.

```
module AOI222D0 (
    input  A1,
    input  A2,
    input  B1,
    input  B2,
    input  C1,
    input  C2,
    output ZN
);
    assign ZN = ~((A1 & A2) | (B1 & B2) | (C1 & C2));
endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AOI222D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.11. Dos AND de dos entradas y OR de dos entradas con salida negada

Figura 16. Esquemático de la celda AOI22D0.



Nota. La figura muestra el esquema lógico de la celda AOI22D0. La salida negada corresponde a la inversión de la operación OR entre los resultados de dos compuertas AND de dos entradas.

Cuadro 63. Tabla de verdad de la celda AOI22D0.

INPUT				OUTPUT
A1	A2	B1	B2	ZN
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Nota. La tabla presenta la relación lógica entre las entradas A1, A2, B1 y B2 de la celda AOI22D0 y su salida negada correspondiente.

Cuadro 64. Código Verilog de la celda AOI22D0.

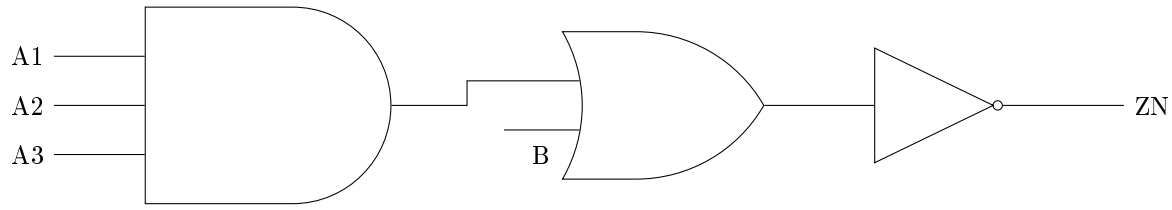
```
1 module AOI22D0 (
2     input  A1,
3     input  A2,
4     input  B1,
5     input  B2,
```

```
6     output ZN
7 );
8     assign ZN = ~((A1 & A2) | (B1 & B2));
9 endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AOI22D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.12. AND de tres entradas y OR de dos entradas con salida negada

Figura 17. Esquemático de la celda AOI31D0.



Nota. La figura muestra el esquema lógico de la celda AOI31D0. La salida negada corresponde a la inversión de la operación OR entre el resultado de una compuerta AND de tres entradas y la señal B.

Cuadro 65. Tabla de verdad de la celda AOI31D0.

INPUT				OUTPUT
A1	A2	A3	B	ZN
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Nota. La tabla presenta la relación lógica entre las entradas A1, A2, A3 y B de la celda AOI31D0 y su salida negada correspondiente.

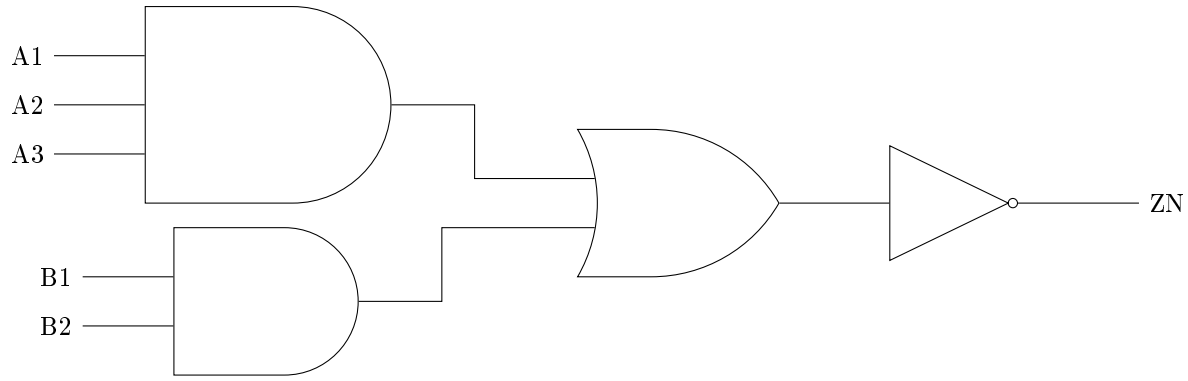
Cuadro 66. Código Verilog de la celda AOI31D0.

```
module AOI31D0 (
    input  A1,
    input  A2,
    input  A3,
    input  B,
    output ZN
);
    assign ZN = ~((A1 & A2 & A3) | B);
endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AOI31D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.13. Una AND de tres entradas y una AND de dos entradas seguidas de un OR de dos entradas con salida negada

Figura 18. Esquemático de la celda AOI32D0.



Nota. La figura muestra el esquema lógico de la celda AOI32D0. La salida negada corresponde a la inversión de la operación OR entre el resultado de una compuerta AND de tres entradas y el resultado de una compuerta AND de dos entradas.

Cuadro 67. Tabla de verdad de la celda AOI32D0.

INPUT					OUTPUT
B1	B2	A1	A2	A3	ZN
0	0	0	0	0	1
0	0	0	0	1	1
0	0	0	1	0	1
0	0	0	1	1	1
0	0	1	0	0	1
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	0	1	1
0	1	0	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	0	1	1
1	0	0	1	0	1
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	0	1	0
1	1	0	1	0	0
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	0	1	0
1	1	1	1	0	0
1	1	1	1	1	0

Nota. La tabla presenta la relación lógica entre las entradas A1, A2, A3, B1 y B2 de la celda AOI32D0 y su salida negada correspondiente.

Cuadro 68. Código Verilog de la celda AOI32D0.

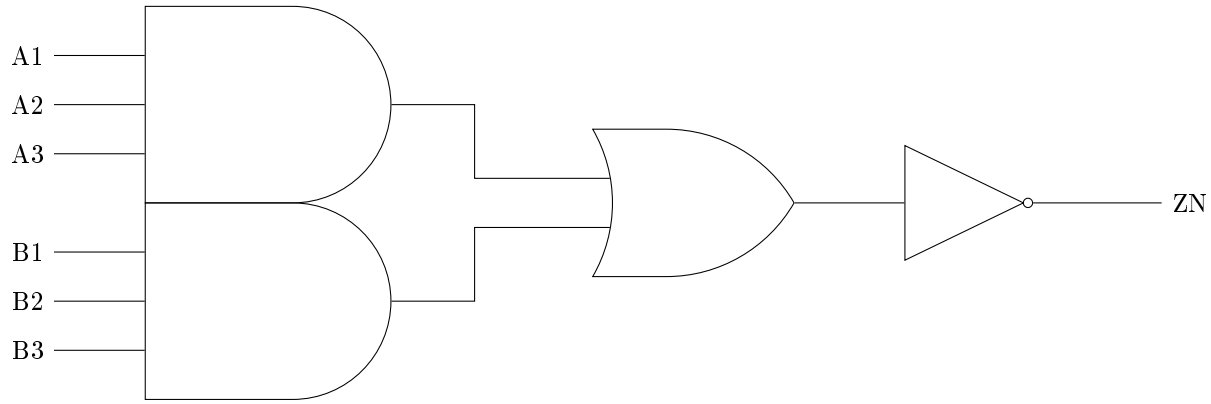
```
1 module AOI32D0 (
2     input  A1,
3     input  A2,
4     input  A3,
5     input  B1,
```

```
6       input  B2,
7       output ZN
8   );
9       assign ZN = ~((A1 & A2 & A3) | (B1 & B2));
10  endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AOI32D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.14. Dos AND de tres entradas y OR de dos entradas con salida negada

Figura 19. Esquemático de la celda AOI33D0.



Nota. La figura muestra el esquema lógico de la celda AOI33D0. La salida negada corresponde a la inversión de la operación OR entre los resultados de dos compuertas AND de tres entradas.

Cuadro 69. Tabla de verdad de la celda AOI33D0.

INPUT						OUTPUT
B1	B2	B3	A1	A2	A3	ZN
0	0	0	0	0	0	1
0	0	0	0	0	1	1
0	0	0	0	1	0	1
0	0	0	0	1	1	1
0	0	0	1	0	0	1
0	0	0	1	0	1	1
0	0	0	1	1	0	1
0	0	0	1	1	1	0
0	0	1	0	0	0	1
0	0	1	0	0	1	1
0	0	1	0	1	0	1
0	0	1	0	1	1	1
0	0	1	1	0	0	1
0	0	1	1	0	1	1

INPUT						OUTPUT
B1	B2	B3	A1	A2	A3	ZN
0	0	1	1	1	0	1
0	0	1	1	1	1	0
0	1	0	0	0	0	1
0	1	0	0	0	1	1
0	1	0	0	1	0	1
0	1	0	0	1	1	1
0	1	0	1	0	0	1
0	1	0	1	0	1	1
0	1	0	1	1	0	1
0	1	0	1	1	1	0
0	1	1	0	0	0	1
0	1	1	0	0	1	1
0	1	1	0	1	0	1
0	1	1	0	1	1	1
0	1	1	1	0	0	1
0	1	1	1	0	1	1
0	1	1	1	1	0	1
0	1	1	1	1	1	0
1	0	0	0	0	0	1
1	0	0	0	0	1	1
1	0	0	0	1	0	1
1	0	0	0	1	1	1
1	0	0	1	0	0	1
1	0	0	1	0	1	1
1	0	0	1	1	0	1
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	0	1	0	0	1	1
1	0	1	0	1	0	1
1	0	1	0	1	1	1
1	0	1	1	0	0	1
1	0	1	1	0	1	1
1	0	1	1	1	0	1
1	0	1	1	1	1	0
1	1	0	0	0	0	1
1	1	0	0	0	1	1
1	1	0	0	1	0	1
1	1	0	0	1	1	1
1	1	0	1	0	0	1
1	1	0	1	0	1	1
1	1	0	1	1	0	1

INPUT						OUTPUT
B1	B2	B3	A1	A2	A3	ZN
1	1	0	1	1	1	0
1	1	1	0	0	0	0
1	1	1	0	0	1	0
1	1	1	0	1	0	0
1	1	1	0	1	1	0
1	1	1	1	0	0	0
1	1	1	1	0	1	0
1	1	1	1	1	0	0
1	1	1	1	1	1	0

Nota. La tabla presenta la relación lógica entre las entradas A1, A2, A3, B1, B2 y B3 de la celda AOI33D0 y su salida negada correspondiente.

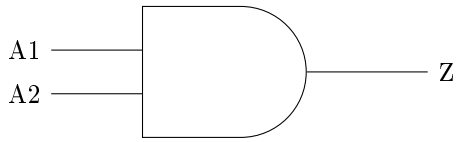
Cuadro 70. Código Verilog de la celda AOI33D0.

```
module AOI33D0 (
    input  A1,
    input  A2,
    input  A3,
    input  B1,
    input  B2,
    input  B3,
    output ZN
);
    assign ZN = ~((A1 & A2 & A3) | (B1 & B2 & B3));
endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda AOI33D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.15. AND de dos entradas

Figura 20. Esquemático de la celda CKAN2D0.



Nota. La figura muestra el esquema lógico de la celda CKAN2D0 como una compuerta AND de dos entradas. Esta representación permite visualizar la relación entre las señales de entrada y la salida lógica de la celda.

Cuadro 71. Tabla de verdad de la celda CKAN2D0.

INPUT		OUTPUT
A1	A2	Z
0	0	0
0	1	0
1	0	0
1	1	1

Nota. La tabla presenta la relación lógica entre las entradas A1 y A2 de la celda CKAN2D0 y su salida correspondiente.

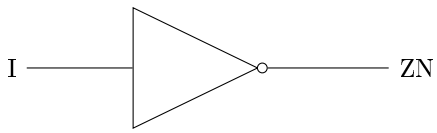
Cuadro 72. Código Verilog de la celda CKAN2D0.

```
module CKAN2D0 (
    input  A1,
    input  A2,
    output Z
);
    assign Z = A1 & A2;
endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda CKAN2D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.16. Inversor

Figura 21. Esquemático de la celda CKND1.



Nota. La figura muestra el esquema lógico de la celda CKND1 como un inversor. Esta celda permite cambiar la polaridad de una señal dentro del diseño digital.

Cuadro 73. Tabla de verdad de la celda CKND1.

INPUT	OUTPUT
I	ZN
0	1
1	0

Nota. La tabla presenta la relación lógica entre la entrada I de la celda CKND1 y su salida negada correspondiente.

Cuadro 74. Código Verilog de la celda CKND1.

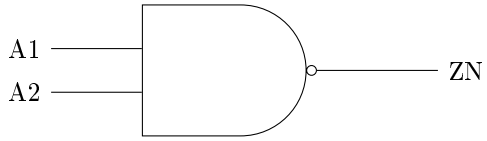
```
module CKND1 (
    input  I,
```

```
3     output ZN
4 );
5     assign ZN = ~I;
6 endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda CKND1, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.17. NAND de dos entradas

Figura 22. Esquemático de la celda CKND2D0.



Nota. La figura muestra el esquema lógico de la celda CKND2D0 como una compuerta NAND de dos entradas. Esta representación permite verificar la relación entre las entradas y la salida negada de la celda.

Cuadro 75. Tabla de verdad de la celda CKND2D0.

INPUT		OUTPUT
A1	A2	ZN
0	0	1
0	1	1
1	0	1
1	1	0

Nota. La tabla presenta la relación lógica entre las entradas A1 y A2 de la celda CKND2D0 y su salida negada correspondiente.

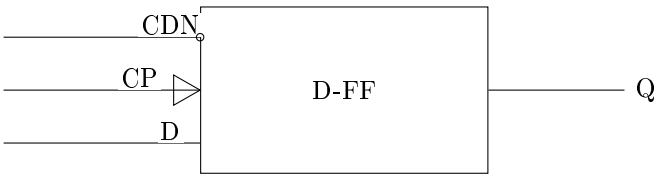
Cuadro 76. Código Verilog de la celda CKND2D0.

```
1 module CKND2D0 (
2     input  A1,
3     input  A2,
4     output ZN
5 );
6     assign ZN = ~(A1 & A2);
7 endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda CKND2D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.1.18. Flip-flop D con reset asíncrono activo en bajo

Figura 23. Esquemático de la celda DFCNQD1.



Nota. La figura muestra el esquema lógico de la celda DFCNQD1. Esta celda corresponde a un flip-flop tipo D con reset asíncrono activo en bajo y salida Q.

Cuadro 77. Tabla de verdad de la celda DFCNQD1.

INPUT			OUTPUT
CDN	CP	D	Q^+
0	–	–	0
1	↑	0	0
1	↑	1	1
1	*sin flanco*	–	Q

Nota. La tabla presenta la relación funcional entre las entradas CDN, CP y D de la celda DFCNQD1 y el siguiente estado de la salida Q.

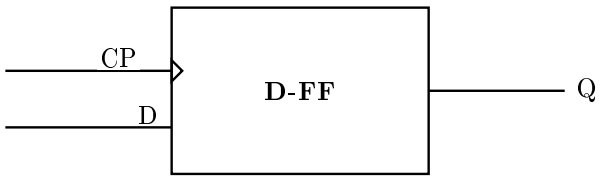
Cuadro 78. Código Verilog de la celda DFCNQD1.

```
1  module DFCNQD1 (
2      input  wire D,
3      input  wire CP,
4      input  wire CDN,
5      output reg  Q
6  );
7    always @(posedge CP or negedge CDN) begin
8      if (!CDN)
9        Q <= 1'b0;
10     else
11       Q <= D;
12   end
13 endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda DFCNQD1, utilizada para definir el comportamiento secuencial del flip-flop dentro del diseño.

8.1.19. Flip-flop D sin reset con flanco positivo

Figura 24. Esquemático de la celda DFQD1.



Nota. La figura muestra el esquema lógico de la celda DFQD1. Esta celda corresponde a un flip-flop tipo D que captura el dato en el flanco ascendente del reloj.

Cuadro 79. Tabla de verdad de la celda DFQD1.

INPUT		OUTPUT
CP	D	Q^+
↑	0	0
↑	1	1
sin flanco	–	Q

Nota. La tabla presenta la relación funcional entre las entradas CP y D de la celda DFQD1 y el siguiente estado de la salida Q.

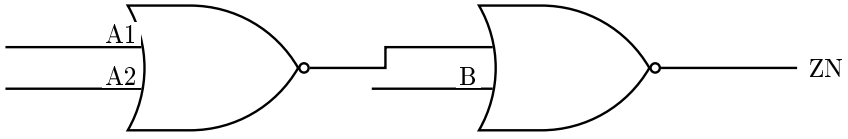
Cuadro 80. Código Verilog de la celda DFQD1.

```
module DFQD1 (
    input  wire D,
    input  wire CP,
    output reg  Q
);
  always @(posedge CP) begin
    Q <= D;
  end
endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda DFQD1, utilizada para definir el comportamiento secuencial del flip-flop dentro del diseño.

8.1.20. IAO21D0

Figura 25. Esquemático de la celda IAO21D0.



Nota. La figura muestra el esquema lógico de la celda IAO21D0, implementada mediante dos compuertas NOR en cascada. Esta estructura es equivalente a una función lógica compuesta con salida negada.

La expresión lógica de la celda es la siguiente:

$$ZN = \neg\big((\neg A1 \wedge \neg A2) \vee B\big)$$

Una forma equivalente de escribirla es:

$$ZN = (\neg B) \wedge (A1 \vee A2)$$

Cuadro 81. Tabla de verdad de la celda IAO21D0.

INPUT			OUTPUT
A1	A2	B	ZN
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Nota. La tabla presenta la relación lógica entre las entradas A1, A2 y B de la celda IAO21D0 y su salida negada correspondiente.

Cuadro 82. Código Verilog de la celda IAO21D0.

```
module IAO21D0 (
    input  A1,
    input  A2,
    input  B,
    output ZN
);
    assign ZN = (!B) & (A1 | A2);
endmodule
```

Nota. El cuadro presenta la descripción en Verilog de la celda IAO21D0, utilizada para definir su comportamiento lógico dentro del diseño.

8.2. Black boxes de celdas lógicas

Cuadro 83. Definiciones *gate-level* de las celdas utilizadas.

```
module AN3D0 (input A1, input A2, input A3, output Z);
and (Z, A1, A2, A3);
endmodule

module AN4D0 (input A1, input A2, input A3, input A4, output Z);
and (Z, A1, A2, A3, A4);
```

```
endmodule

module AO211D0 (input A1, input A2, input B, input C, output Z);
wire n0;
and (n0, A1, A2);
or  (Z, n0, B, C);
endmodule

module AO21D0 (input A1, input A2, input B, output Z);
wire n0;
and (n0, A1, A2);
or  (Z, n0, B);
endmodule

module AO22D0 (input A1, input A2, input B1, input B2, output Z);
wire n0, n1;
and (n0, A1, A2);
and (n1, B1, B2);
or  (Z, n0, n1);
endmodule

module AO31D0 (input A1, input A2, input A3, input B, output Z);
wire n0;
and (n0, A1, A2, A3);
or  (Z, n0, B);
endmodule

module AOI211D0 (input A1, input A2, input B, input C, output ZN);
wire n0;
and (n0, A1, A2);
nor (ZN, n0, B, C);
endmodule

module AOI21D0 (input A1, input A2, input B, output ZN);
wire n0;
and (n0, A1, A2);
nor (ZN, n0, B);
endmodule

module AOI221D0 (input A1, input A2, input B1, input B2, input C, output ZN);
wire n0, n1;
and (n0, A1, A2);
and (n1, B1, B2);
nor (ZN, n0, n1, C);
endmodule

module AOI222D0 (input A1, input A2, input B1, input B2, input C1, input C2,
    output ZN);
wire n0, n1, n2;
and (n0, A1, A2);
and (n1, B1, B2);
and (n2, C1, C2);
nor (ZN, n0, n1, n2);
endmodule
```

```
60
61 module AOI22D0 (input A1, input A2, input B1, input B2, output ZN);
62 wire n0, n1;
63 and (n0, A1, A2);
64 and (n1, B1, B2);
65 nor (ZN, n0, n1);
66 endmodule
67
68 module AOI31D0 (input A1, input A2, input A3, input B, output ZN);
69 wire n0;
70 and (n0, A1, A2, A3);
71 nor (ZN, n0, B);
72 endmodule
73
74 module AOI32D0 (input A1, input A2, input A3, input B1, input B2, output ZN);
75 wire n0, n1;
76 and (n0, A1, A2, A3);
77 and (n1, B1, B2);
78 nor (ZN, n0, n1);
79 endmodule
80
81 module AOI33D0 (input A1, input A2, input A3, input B1, input B2, input B3,
       output ZN);
82 wire n0, n1;
83 and (n0, A1, A2, A3);
84 and (n1, B1, B2, B3);
85 nor (ZN, n0, n1);
86 endmodule
87
88 module CKAN2D0 (input A1, input A2, output Z);
89 and (Z, A1, A2);
90 endmodule
91
92 module CKND1 (input I, output ZN);
93 not (ZN, I);
94 endmodule
95
96 module CKND2D0 (input A1, input A2, output ZN);
97 nand (ZN, A1, A2);
98 endmodule
99
100 module DFCNQD1 (input D, input CP, input CDN, output reg Q);
101 always @(posedge CP or negedge CDN) begin
102   if (!CDN)
103     Q <= 1'b0;
104   else
105     Q <= D;
106 end
107 endmodule
108
109 module DFQD1 (input D, input CP, output reg Q);
110 always @(posedge CP) begin
111   Q <= D;
112 end
```

```
endmodule

module IAO21D0 (input A1, input A2, input B, output ZN);
wire nB, y0;
not (nB, B);
or  (y0, A1, A2);
and (ZN, nB, y0);
endmodule

module OA211D0 (input A1, input A2, input B, input C, output Z);
wire n0;
or  (n0, A1, A2);
and (Z, n0, B, C);
endmodule

module OA21D0 (input A1, input A2, input B, output Z);
wire n0;
or  (n0, A1, A2);
and (Z, n0, B);
endmodule

module OA221D0 (input A1, input A2, input B1, input B2, input C, output Z);
wire n0, n1;
or  (n0, A1, A2);
or  (n1, B1, B2);
and (Z, n0, n1, C);
endmodule

module OA22D0 (input A1, input A2, input B1, input B2, output Z);
wire n0, n1;
or  (n0, A1, A2);
or  (n1, B1, B2);
and (Z, n0, n1);
endmodule

module OA31D0 (input A1, input A2, input A3, input B, output Z);
wire n0;
or  (n0, A1, A2, A3);
and (Z, n0, B);
endmodule

module OA32D0 (input A1, input A2, input A3, input B1, input B2, output Z);
wire n0, n1;
or  (n0, A1, A2, A3);
or  (n1, B1, B2);
and (Z, n0, n1);
endmodule

module OAI211D0 (input A1, input A2, input B, input C, output ZN);
wire n0;
or   (n0, A1, A2);
nand (ZN, n0, B, C);
endmodule

```

```
module OAI21D0 (input A1, input A2, input B, output ZN);
wire n0;
or   (n0, A1, A2);
nand (ZN, n0, B);
endmodule

module OAI221D0 (input A1, input A2, input B1, input B2, input C, output ZN);
wire n0, n1;
or   (n0, A1, A2);
or   (n1, B1, B2);
nand (ZN, n0, n1, C);
endmodule

module OAI222D0 (input A1, input A2, input B1, input B2, input C1, input C2,
    output ZN);
wire n0, n1, n2;
or   (n0, A1, A2);
or   (n1, B1, B2);
or   (n2, C1, C2);
nand (ZN, n0, n1, n2);
endmodule
```

Nota. El cuadro presenta las definiciones *gate-level* de las celdas lógicas utilizadas como *black boxes* dentro del flujo de verificación.

8.3. LVS con tecnología de 65 nm utilizando la *reference methodology* de ICV

La verificación *Layout Versus Schematic* (LVS) constituye una de las etapas fundamentales dentro del flujo de diseño físico en tecnología de 65 nm, pues garantiza la equivalencia eléctrica entre el diseño esquemático y su implementación geométrica en el *layout*. En este proceso, el verificador analiza la estructura topológica extraída del circuito integrado, identifica dispositivos, nodos y conectividades, y compara dicha información con el netlist de referencia para detectar discrepancias que puedan comprometer la funcionalidad del chip.

En esta sección se describe el procedimiento empleado para ejecutar LVS utilizando IC Validator bajo el marco de trabajo proporcionado por la *reference methodology* para nodos de 65 nm. Esta metodología permite estandarizar la preparación del entorno, organizar los insumos del diseño y automatizar las ejecuciones mediante *runsets* certificados para la tecnología empleada. A continuación se detalla la configuración adoptada, la estructura del proyecto y los resultados principales derivados del proceso de verificación.

8.3.1. Archivos previos necesarios

Para ejecutar la verificación *Layout Versus Schematic* (LVS) en tecnología de 65 nm es necesario preparar un conjunto de archivos coherentes entre el netlist generado en síntesis lógica y el *layout* final en formato `GDSII`. Este proceso garantiza que IC Validator pueda

analizar ambas representaciones bajo un mismo modelo eléctrico compatible con los *runsets* proporcionados por Synopsys.

Los archivos necesarios son los siguientes:

- Netlist lógico en Verilog, generado durante la síntesis lógica.
- Biblioteca SPICE depurada (`headers.sp`), empleada para mapear celdas estándar.
- Archivo `.icv` o `CDL.icv`, obtenido mediante traducción con `icv_nettran`.
- Archivo de *layout* `.gds`, generado en la síntesis física.
- *Runsets* LVS/RC proporcionados por Synopsys.

Traducción inicial a SPICE (`icv_nettran`)

IC Validator requiere que la biblioteca lógica esté disponible en formato SPICE. Esto se genera aplicando el siguiente comando:

Cuadro 84. Traducción de Verilog a SPICE.

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

Este archivo SPICE sirve como puente entre el nivel lógico y el nivel de dispositivos, permitiendo a IC Validator extraer las conexiones de cada celda estándar.

Adecuación de la biblioteca SPICE

Como indica la documentación técnica utilizada durante el proceso, es necesario modificar el encabezado de la biblioteca para que sea compatible con la tecnología de 65 nm. El encabezado final debe incluir referencias globales, parámetros y la inclusión del archivo `source.added` provisto por el fabricante:

Cuadro 85. Encabezado requerido en la biblioteca SPICE.

```
1 .GLOBAL VDD VSS VDDPST VSSPST
2 *.EQUATION
3 *.SCALE METER
4 *.MEGA
5 .PARAM
6 .INCLUDE /ruta/personal/source.added
```

Depuración del netlist SPICE

Durante la síntesis física (*place and route*), el compilador introduce celdas adicionales que no deben ser consideradas en el LVS. Específicamente, deben eliminarse:

- `PCORNER`, correspondiente a celdas de esquina del arreglo estándar.
- `PFILLER` o `FILLx`, correspondientes a celdas de relleno para continuidad de *well* y rieles de potencia.

Estas celdas no tienen comportamiento lógico ni dispositivos eléctricos, por lo que su presencia genera discrepancias durante la comparación.

La Figura 26 muestra un fragmento típico de estas instancias:

Figura 26. Instancias `PCORNER` y `PFILLER` generadas por P&R.

```
29 TIEL U9 ( .ZN ( \*Logic0*1 ) , .VDD ( VDD ) , .VSS ( V
30 PCORNER CORNER1 () ;
31 PCORNER CORNER2 () ;
32 PCORNER CORNER3 () ;
33 PCORNER CORNER4 () ;
34 PVDD1CDG PVDD1 ( .VDD ( VDD ) ) ;
35 PVSS1CDG PVSS1 ( .VSS ( VSS ) ) ;
36 PFILLER20 __added_filler_instance_0 () ;
37 PFILLER20 __added_filler_instance_1 () ;
38 PFILLER20 __added_filler_instance_2 () ;
39 PFILLER20 __added_filler_instance_3 () ;
40 PFILLER20 __added_filler_instance_4 () ;
41 PFILLER20 __added_filler_instance_5 () ;
42 PFILLER20 __added_filler_instance_6 () ;
43 PFILLER20 __added_filler_instance_7 () ;
44 PFILLER20 __added_filler_instance_8 () ;
45 PFILLER20 __added_filler_instance_9 () ;
46 PFILLER20 __added_filler_instance_10 () ;
47 PFILLER20 __added_filler_instance_11 () ;
48 PFILLER20 __added_filler_instance_12 () ;
49 PFILLER20 __added_filler_instance_13 () ;
50 PFILLER20 __added_filler_instance_14 () ;
51 PFILLER20 __added_filler_instance_15 () ;
52 PFILLER20 __added_filler_instance_16 () ;
```

Nota. Estas celdas no representan dispositivos eléctricos ni aportan conectividad lógica; su presencia en el netlist físico puede inducir discrepancias durante LVS. En el flujo se eliminan para que el netlist eléctrico refleje únicamente dispositivos relevantes.

Asimismo, se deben eliminar las conexiones redundantes a `VDD` y `VSS` generadas automáticamente por el *place and route*. Estas se observan usualmente como rieles adicionales:

Figura 27. Rieles y conexiones adicionales a `VDD`/`VSS` en el netlist físico.

```
284 FILL1 \xofiller!FILL1!x1900000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
285 FILL1 \xofiller!FILL1!x1900000y2134000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
286 FILL1 \xofiller!FILL1!x1900000y2152000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
287 FILL8 \xofiller!FILL8!x8084000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
288 FILL16 \xofiller!FILL16!x8052000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
289 FILL64 \xofiller!FILL64!x7924000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
290 FILL64 \xofiller!FILL64!x7796000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
291 FILL64 \xofiller!FILL64!x7668000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
292 FILL64 \xofiller!FILL64!x7540000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
293 FILL64 \xofiller!FILL64!x7412000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
294 FILL64 \xofiller!FILL64!x7284000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
295 FILL64 \xofiller!FILL64!x7156000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
296 FILL64 \xofiller!FILL64!x7028000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
297 FILL64 \xofiller!FILL64!x6900000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
298 FILL64 \xofiller!FILL64!x6772000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
299 FILL64 \xofiller!FILL64!x6644000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
300 FILL64 \xofiller!FILL64!x6516000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
301 FILL64 \xofiller!FILL64!x6388000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
302 FILL64 \xofiller!FILL64!x6260000y1900000 ( .VSS ( VSS ) , .VDD ( VDD ) ) ;
```

Nota. Las conexiones de potencia globales (`VDD`/`VSS`) se tratan como nodos globales en ICV; conexiones duplicadas generadas por P&R pueden provocar advertencias o desajustes de conectividad. Se depuran antes de la traducción final.

Generación del archivo `CDL.icv`

Una vez depurados los archivos, se genera el netlist final en formato ICV/CDL compatible con el *runset* de LVS:

Cuadro 86. Generación del netlist ICV/CDL final.

```
icv_nettran -verilog circuito.v \
            -sp headers.sp \
            -outType ICV \
            -outName CDL.icv
```

Este proceso es equivalente al utilizado en trabajos previos de verificación física en tecnologías CMOS, donde se realiza la convergencia entre los modelos lógicos y físicos para su análisis en IC Validator.

Una vez generados estos archivos, no requieren nuevas ediciones. Únicamente deben ser consumidos por el entorno RM para la ejecución de LVS.

8.3.2. Preparación del entorno con la *reference methodology* de ICV

La *reference methodology* de Synopsys provee una estructura estandarizada de *scripts*, plantillas y *runsets* que normaliza la ejecución de verificaciones físicas, como LVS y ERC, en IC Validator. Su objetivo es asegurar reproducibilidad, trazabilidad y consistencia entre diseños y equipos, minimizando la edición manual de reglas y centralizando la configuración en archivos versionados.

En primer lugar, se debe obtener el paquete oficial de RM desde el portal de Synopsys. Para este trabajo se empleó la versión más reciente disponible al momento del experimento, ICV-RM V-2023.12. El paquete se distribuye como un archivo comprimido que debe

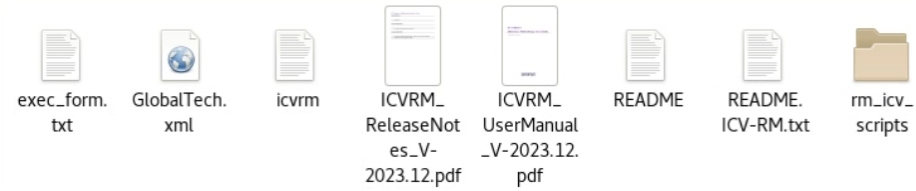
extraerse en el directorio de trabajo elegido.

Figura 28. Paquete RM descargado: `ICV-RM_V-2023.12.tar`.

Nota. El paquete contiene el esqueleto del entorno RM para ICV: scripts TCL, *runsets*, manual de usuario y notas de versión.

Tras la extracción, la carpeta resultante expone los archivos base del entorno RM, por ejemplo `icvrm`, `exec_form.txt`, `GlobalTech.xml`, `rm_icv_scripts/` y la documentación oficial, que serán referenciados por los flujos de ejecución.

Figura 29. Contenido principal del paquete RM tras la extracción.



Nota. Los elementos clave son `icvrm`, controlador TCL del flujo; `exec_form.txt`, formulario de ejecución; `GlobalTech.xml`, configuración global; `rm_icv_scripts/`, scripts internos; y la documentación oficial.

Organización inicial del directorio de trabajo y archivos de entrada

Para mantener un flujo de verificación ordenado y reproducible, cada diseño debe ubicarse dentro de su propia carpeta en el entorno RM. Esta estructura permite separar los archivos de entrada (`.gds` y `.icv`) de cada bloque y facilita la ejecución independiente de LVS y ERC para múltiples circuitos.

Así, dentro del directorio principal `ICV-RM_V-2023.12`, se crea una carpeta por diseño. Por ejemplo:

- `NOT/Inputs/`.

- `ALU/Inputs/`.
- `Gran_Jaguar/Inputs/`.

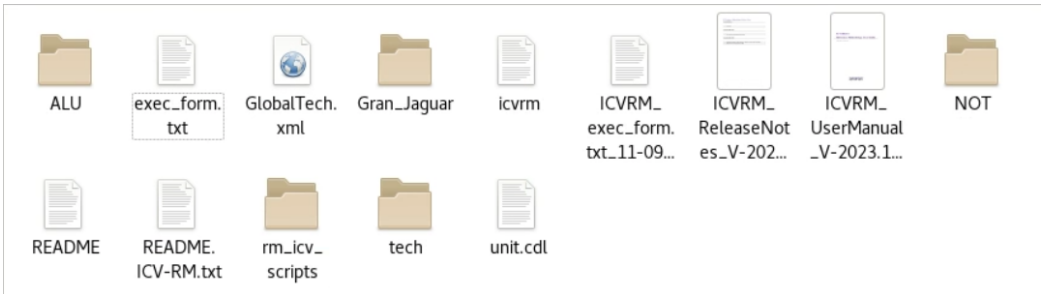
En cada una de estas carpetas `Inputs/` se almacenan únicamente los archivos necesarios para la verificación:

- El archivo `GDSII` generado durante la síntesis física (`circuito.gds`).
- El netlist traducido al formato de IC Validator (`circuito.icv`).

También se crea una carpeta `tech/` para almacenar el *runset* específico de la tecnología. Con el fin de contar con un único punto de referencia, se copia en esta carpeta el *runset* oficial de Synopsys para 65 nm.

Con esta organización, el entorno RM puede apuntar fácilmente a los archivos específicos de cada diseño sin modificar la estructura base del método de referencia.

Figura 30. Carpeta de trabajo organizada para múltiples diseños bajo RM.



Nota. En esta carpeta se pueden organizar múltiples diseños, cada uno con su propio subdirectorio `Inputs/` que contiene los archivos `.gds` y `.icv` necesarios para la verificación. La carpeta `tech/` almacena el runset de tecnología común.

Configuración del ambiente y variables necesarias

Antes de ejecutar RM, es indispensable configurar las variables de entorno que permiten a ICV encontrar su instalación, la carpeta de encabezados y la licencia flotante. Esta configuración evita errores de ejecución, especialmente aquellos relacionados con rutas o versiones.

Las variables que deben declararse son las siguientes:

- `ICV_HOME_DIR`, directorio de instalación de IC Validator.
- `ICV_INCLUDES`, carpeta de encabezados de ICV.
- `PATH`, ajuste para que el binario `icv` pueda invocarse desde cualquier directorio.

- `SNPSLMD_LICENSE_FILE`, servidor o archivo de licencia de Synopsys.

Estas variables deben declararse al inicio de cada sesión de trabajo para garantizar que RM pueda invocar correctamente todos los componentes de IC Validator.

Los comandos exactos utilizados fueron los siguientes:

Cuadro 87. Variables de entorno necesarias para ejecutar IC Validator.

```
# Directorio de instalacion de IC Validator
export ICV_HOME_DIR=/usr/synopsys/icvalidator/W-2024.09-SP4

# Carpeta de includes
export ICV_INCLUDES="$ICV_HOME_DIR/include"

# Extender el PATH para acceder al binario icv
export PATH="$ICV_HOME_DIR/bin/LINUX.64:$PATH"

# Servidor/licencia de Synopsys
export SNPSLMD_LICENSE_FILE=27020@192.168.74.1

# Verificacion de instalacion
which icv
icv -V
```

Estas declaraciones deben realizarse en cada sesión nueva, ya que el entorno no persiste entre terminales.

Ajuste de la versión de ICV dentro del archivo `icvrm`

El archivo principal del flujo RM, `icvrm`, incluye una sección donde se valida la versión de IC Validator. Para evitar errores como:

```
can't read ïcvver": no such variable
```

fue necesario editar manualmente el archivo `icvrm` e incluir explícitamente la versión instalada. El ajuste aplicado fue el siguiente al inicio del archivo:

Cuadro 88. Edición manual del archivo `icvrm` para definir la versión de ICV.

```
#!/usr/bin/env tclsh
if {![info exists icvver]} {
    if {[info exists env(ICV_HOME_DIR)]} {
        set icvver [file tail $env(ICV_HOME_DIR)]
    } else {
        set icvver "W-2024.09-SP4"
    }
}
```

Con esta modificación, el flujo RM reconoce consistentemente la versión correcta del ejecutable IC Validator y se evitan fallos al iniciar la ejecución de LVS y ERC.

Este ajuste solo debe realizarse una vez, ya que aplica para todos los diseños del proyecto.

Creación de BUILDS y registro de diseños con RM

Una vez organizado el directorio y configurado el ambiente, se procede a que RM genere la estructura de trabajo (BUILDS) y el esqueleto de cada diseño. Para ello, se utiliza el instalador del flujo:

Cuadro 89. Generación de BUILDS y registro de diseños en RM.

```
cd ICV-RM_V-2023.12

# Asegurar permisos de ejecucion del controlador RM
chmod +x ./icvrm

# Registrar varios disenos en un solo comando
./icvrm -install "NOT ALU Gran_Jaguar"

# Estructura esperada
# BUILDS/
# -----NOT/
# -----------NOT.xml
# -----ALU/
# -----------ALU.xml
# -----Gran_Jaguar/
# -----------Gran_Jaguar.xml
```

El directorio BUILDS/ será el destino donde se guardarán las ejecuciones de LVS, ERC y RCX, así como los *logs* por fecha. Para cada diseño, RM crea un archivo <diseño>.xml que actúa como formulario de ejecución del bloque.

Edición del XML de cada diseño (BUILDS/*diseño*/*diseño*.xml)

El archivo <diseño>.xml define, entre otros elementos, el nombre del diseño, la celda superior y las rutas a los archivos de entrada (.gds y .icv). A continuación, se muestra un ejemplo mínimo utilizado en los experimentos. Las rutas y nombres deben ajustarse en cada caso:

Cuadro 90. Plantilla mínima para BUILDS/design/design.xml.

```
<?xml version="1.0"?>
<ICV>
  <Design>Nombre_Topcell</Design>
  <TopCell>Nombre_Topcell</TopCell>

  <!-- Rutas a los archivos de entrada -->
  <LayoutFile>/ruta/ICV-RM_V-2023.12/design/Inputs/Design.gds</LayoutFile>
  <SchematicICV>/ruta/ICV-RM_V-2023.12/design/Inputs/Design.icv</SchematicICV>

  <!-- Nodo tecnologico -->
  <Node>N65</Node>
</ICV>
```

Los puntos que deben verificarse al editar cada <diseño>.xml son los siguientes:

- <Design> y <TopCell> deben coincidir con el bloque y la *top cell* del archivo .gds.

- `<LayoutFile>` debe apuntar al archivo `.gds` dentro de *`diseño`*`/Inputs/`.
- `<SchematicICV>` debe apuntar al netlist traducido a ICV (`.icv`) del mismo diseño.
- `<Node>` debe reflejar el nodo tecnológico utilizado, N65 en este trabajo.

Edición del archivo global `GlobalTech.xml`

El archivo `GlobalTech.xml` centraliza la selección del *runset* y la configuración de ejecución, como CPU y multinodo. Para LVS en 65 nm, se utilizó la siguiente configuración. La ruta del *runset* debe ajustarse a la carpeta `tech/` correspondiente:

Cuadro 91. Configuración de `GlobalTech.xml` para LVS en N65.

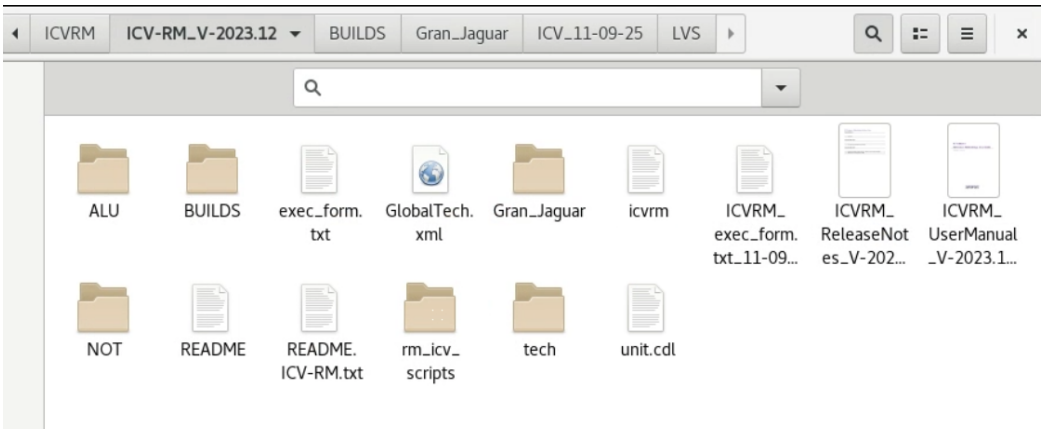
```
<?xml version="1.0"?>
<ICV>
  <FoundryTech>
    <LVSRunset> ./tech/archivo_runset_a_utilizar</LVSRunset>
    <Node> N65 </Node>
  </FoundryTech>

  <Execution>
    <NumCPUs> 4 </NumCPUs>
    <EnableMultiHost> 0 </EnableMultiHost>
    <GridType> None </GridType>
    <GridExecutionCommand></GridExecutionCommand>
  </Execution>
</ICV>
```

Algunas notas prácticas son las siguientes:

- `<LVSRunset>` puede apuntar a cualquier ruta válida, ya sea relativa o absoluta. Si se planea usar ERC, puede mantenerse un único *runset* y conmutar reglas dentro del archivo `.9m`, o preparar un `ERCRunset` en una copia de `GlobalTech.xml`.
- `<NumCPUs>` controla el paralelismo de ICV. En las pruebas realizadas se utilizó el valor `4`.

Con `BUILDS` creado, los archivos `<diseño>.xml` apuntando a sus respectivos `Inputs` y `GlobalTech.xml` referenciando el *runset* correcto, el entorno queda listo para validar (`-v`) y ejecutar (`-e`) el flujo RM en cada bloque.

Figura 31. Carpeta final de trabajo organizada para múltiples diseños bajo RM.



Nota. En esta carpeta se pueden organizar múltiples diseños, cada uno con su propio subdirectorio `Inputs/` que contiene los archivos `.gds` y `.icv` necesarios para la verificación. La carpeta `tech/` almacena el runset de tecnología común.

8.3.3. Edición y ejecución del archivo `exec_form.txt`

El archivo `exec_form.txt` actúa como el punto de control principal del flujo RM. En él se especifica qué tarea debe ejecutar el entorno de referencia, por ejemplo LVS, ERC o RCX, y sobre qué diseño previamente registrado mediante el comando `icvrm -install`.

En este proyecto se empleó una configuración mínima, suficiente para ejecutar la verificación LVS del diseño en curso. El archivo contiene únicamente una instrucción:

Cuadro 92. Contenido mínimo de `exec_form.txt`.

```
1 EXECUTE_LVS design
```

Aquí, el argumento `design` corresponde al nombre del bloque registrado en la fase de instalación del entorno RM.

Validación del archivo con `-v`

Antes de ejecutar cualquier verificación, RM permite validar la consistencia del entorno mediante la opción `-v`. Este paso revisa que:

- las rutas en el XML del diseño sean correctas;
- el archivo `.gds` exista y sea legible;
- el netlist `.icv` esté correctamente formateado;
- el *runset*, correspondiente a las reglas LVS de 65 nm, esté presente;

- la versión de ICV coincida con la definida en `icvrm`.

La validación se ejecuta con el siguiente comando:

Cuadro 93. Validación del flujo RM con la opción `-v`.

```
cd ICV-RM_V-2023.12
./icvrm -v exec_form.txt
```

Si la configuración es correcta, el sistema reporta el estado de cada archivo sin ejecutar todavía el motor LVS.

Ejecución del flujo LVS con `-e`

Una vez validado el entorno, la verificación LVS se inicia con:

Cuadro 94. Ejecución completa del flujo LVS mediante RM.

```
./icvrm -e exec_form.txt
```

Este comando activa el motor de IC Validator, aplica el *runset* seleccionado y genera automáticamente:

- la carpeta `BUILDS/<diseño>/ICV_<fecha>/LVS/`;
- el archivo `<diseño>.LVS_ERRORS`;
- el archivo `<diseño>.LAYOUT_ERRORS`;
- el archivo `<diseño>.RESULTS`;
- los reportes detallados de conectividad;
- el árbol completo de *logs* y archivos temporales.

Esta estructura permite llevar un control histórico por fecha y facilita el análisis posterior de errores de conectividad o discrepancias entre el *layout* y el netlist.

8.3.4. Resultados de la verificación eléctrica y topológica (LVS y ERC)

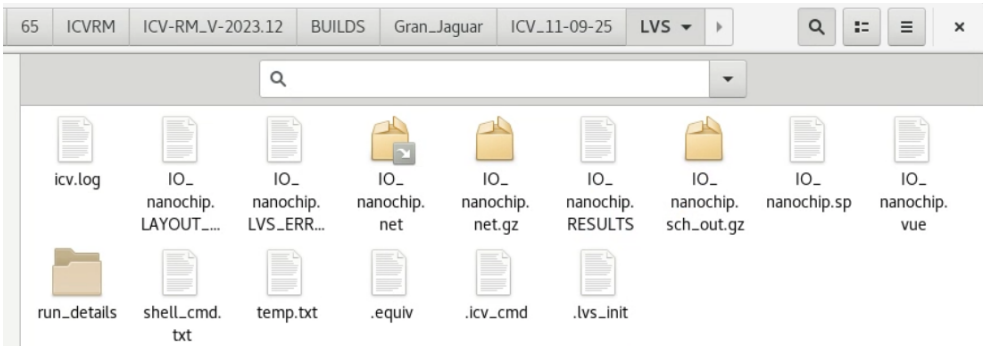
El flujo de verificación implementado en este proyecto permitió comprobar la integridad eléctrica y topológica de los diseños sintetizados en tecnología de 65 nm, siguiendo el método de referencia de Synopsys. La ejecución conjunta de *Layout Versus Schematic* (LVS) y *Electrical Rule Check* (ERC) proporcionó una validación exhaustiva de la equivalencia entre el diseño lógico y su implementación física, así como la detección de posibles riesgos eléctricos.

En las subsecciones siguientes se presentan los resultados obtenidos para cada categoría de verificación, acompañados de fragmentos reales generados por IC Validator y del análisis correspondiente.

Estructura de resultados generada por RM

Para cada diseño verificado, el método RM crea una carpeta con todos los artefactos necesarios para auditoría, depuración y trazabilidad del proceso. La estructura típica incluye archivos con el netlist extraído, equivalencias, reportes de errores y comandos ejecutados.

Figura 32. Contenido típico de la carpeta `LVS` generada para un diseño verificado.



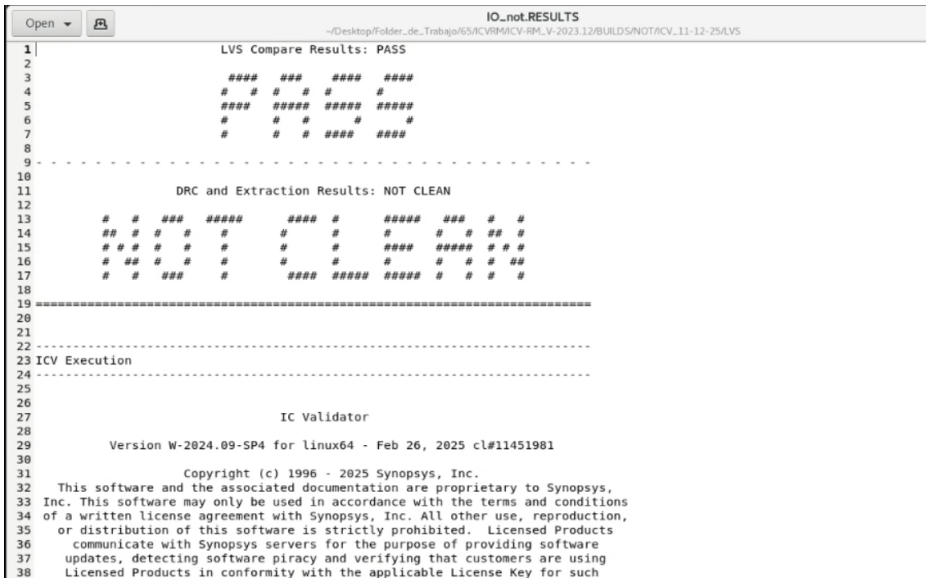
Nota. Entre los archivos más relevantes se encuentran `.net`, que contiene el netlist extraído; `.sp`, que incluye la vista SPICE reconstituida desde el *layout*; `RESULTS`, con el resumen global; `.vue`, compatible con herramientas gráficas; y `run_details`, que registra cada comando ejecutado. Estos elementos ofrecen una evidencia completa y verificable del proceso efectuado.

Resultados LVS por diseño

En esta subsección se presentan los resultados obtenidos para los diseños `NOT`, `ALU` y `Gran_Jaguar`, junto con los reportes de errores de *layout* generados por IC Validator.

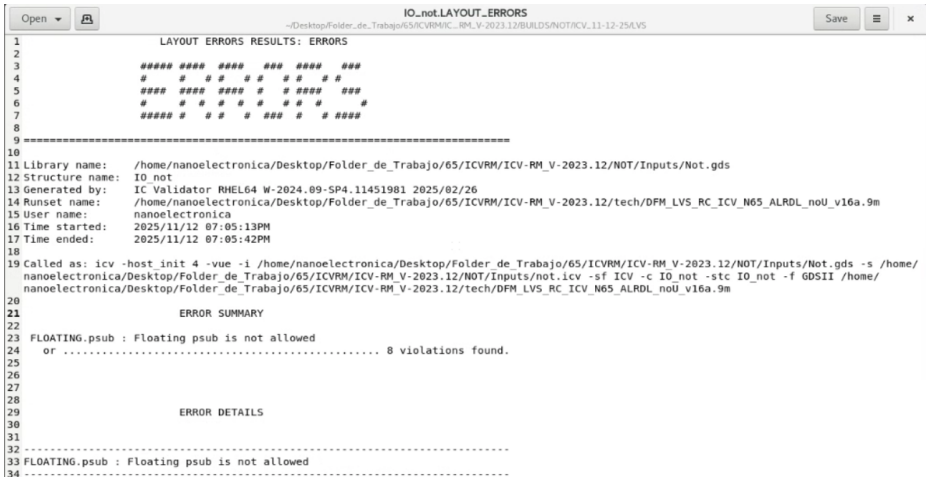
En los tres diseños, `NOT`, `ALU` y `Gran_Jaguar`, la verificación reportó `LVS = PASS`, lo que confirma la equivalencia topológica entre el netlist de referencia y el *layout* extraído. Estos resultados validan la preparación del entorno y el uso del *runset* bajo ICVRM para 65 nm, así como la consistencia del flujo de traducción y lectura de capas.

Figura 33. Resultado LVS para el inversor `NOT`.



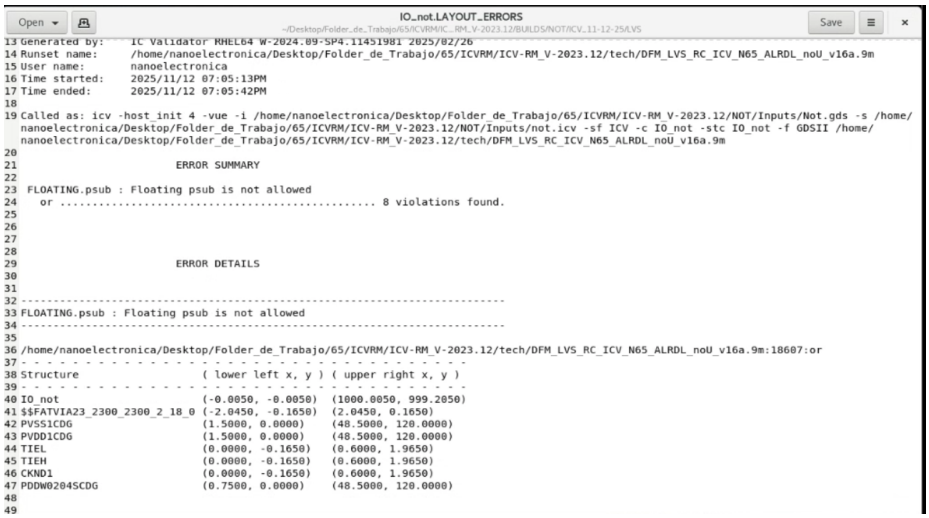
Nota. El inversor `NOT` obtuvo un resultado `LVS = PASS`, lo que confirma que el *layout* y el netlist coinciden topológicamente. No obstante, el resultado general se reporta como `NOT CLEAN` debido a la presencia de ocho errores del tipo `FLOATING.psub`, evidenciados en los reportes de *layout errors*. Durante la etapa física, algunas celdas no lograron conectarse al nodo de referencia `VSS`, lo que provocó regiones de substrato flotante.

Figura 34. Reporte de errores de *layout* para el diseño `NOT` (parte 1).



Nota. El archivo `IO_not.LAYOUT_ERRORS` identifica ocho violaciones relacionadas con substratos tipo P no conectados, clasificadas como `FLOATING.psub`.

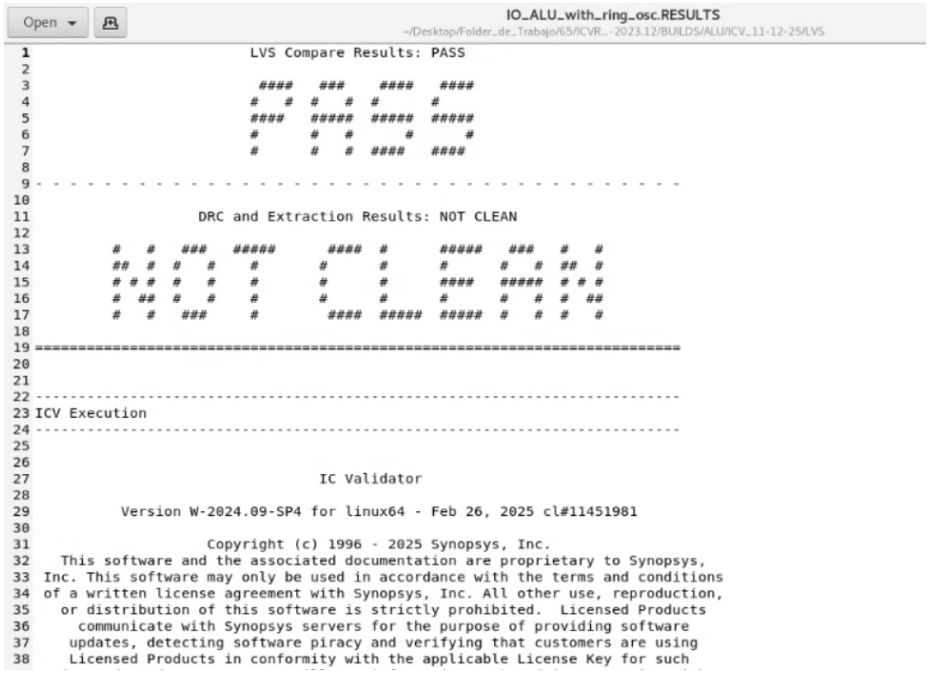
Figura 35. Reporte de errores de *layout* para el diseño `NOT` (parte 2).



```
IO_not.LAYOUT_ERRORS
Open  Save
13 Generated by:     IC Validator RHEL64 W-2024.09-SP4.11451981 2025/02/26
14 Runset name:      /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/ICVRM/ICV-RM_V-2023.12/tech/DFM_LVS_RC_ICV_N65_ALRDL_noU_v16a.9m
15 User name:        nanoelectronica
16 Time started:     2025/11/12 07:05:13PM
17 Time ended:       2025/11/12 07:05:42PM
18
19 Called as: icv -host_init 4 -vue -i /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/ICVRM/ICV-RM_V-2023.12/NOT/Inputs/Not.gds -s /home/
   nanoelectronica/Desktop/Folder_de_Trabajo/65/ICVRM/ICV-RM_V-2023.12/NOT/Inputs/not.icv -sf ICV -c IO_not -stc IO_not -f GDSII /home/
   nanoelectronica/Desktop/Folder_de_Trabajo/65/ICVRM/ICV-RM_V-2023.12/tech/DFM_LVS_RC_ICV_N65_ALRDL_noU_v16a.9m
20
21                        ERROR SUMMARY
22
23 FLOATING.psub : Floating psub is not allowed
24   or ................................................ 8 violations found.
25
26
27
28
29                        ERROR DETAILS
30
31
32 ------------------------------------------------------------------------
33 FLOATING.psub : Floating psub is not allowed
34 ------------------------------------------------------------------------
35
36 /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/ICVRM/ICV-RM_V-2023.12/tech/DFM_LVS_RC_ICV_N65_ALRDL_noU_v16a.9m:18607:or
37 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
38 Structure                 ( lower left x, y ) ( upper right x, y )
39 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
40 IO_not                   (-0.0050, -0.0050)  (1000.0050, 999.2850)
41 $$FATVIA23_2300_2300_2_18_0 (-2.0450, -0.1650)  (2.0450, 0.1650)
42 PVSS1CDG                 (1.5000, 0.0000)    (48.5000, 120.0000)
43 PVDD1CDG                 (1.5000, 0.0000)    (48.5000, 120.0000)
44 TIEL                     (0.0000, -0.1650)   (0.6000, 1.9650)
45 TIEH                     (0.0000, -0.1650)   (0.6000, 1.9650)
46 CKND1                    (0.0000, -0.1650)   (0.6000, 1.9650)
47 PDDW0204SCDG             (0.7500, 0.0000)    (48.5000, 120.0000)
48
49
```

Nota. En esta sección se muestran las coordenadas y estructuras implicadas en las violaciones, lo que permite rastrear su ubicación dentro del *layout*.

Figura 36. Resultado LVS para el diseño `ALU`.



```
IO_ALU_with_ring_osc.RESULTS
Open
~/Desktop/Folder_de_Trabajo/65/ICVR...-2023.12/BUILDS/ALU/ICV_11-12-25/LVS
 1                        LVS Compare Results: PASS
 2
 3                         ####   ###    ####   ####
 4                        #   #  #   #  #      #
 5                        ####   #####  #####  #####
 6                        #      #   #      #      #
 7                        #      #   #  ####   ####
 8
 9 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
10
11                    DRC and Extraction Results: NOT CLEAN
12
13          #   #   ###   #####      ####  #      #####   ###   #   #
14          ##  #  #   #    #       #      #      #      #   #  ##  #
15          # # #  #   #    #       #      #      ####   #####  # # #
16          #  ##  #   #    #       #      #      #      #   #  #  ##
17          #   #   ###     #        ####  #####  #####  #   #  #   #
18
19 ============================================================================
20
21
22 ----------------------------------------------------------------------------
23 ICV Execution
24 ----------------------------------------------------------------------------
25
26
27                                IC Validator
28
29           Version W-2024.09-SP4 for linux64 - Feb 26, 2025 cl#11451981
30
31                     Copyright (c) 1996 - 2025 Synopsys, Inc.
32    This software and the associated documentation are proprietary to Synopsys,
33  Inc. This software may only be used in accordance with the terms and conditions
34  of a written license agreement with Synopsys, Inc. All other use, reproduction,
35    or distribution of this software is strictly prohibited.  Licensed Products
36      communicate with Synopsys servers for the purpose of providing software
37     updates, detecting software piracy and verifying that customers are using
38     Licensed Products in conformity with the applicable License Key for such
```

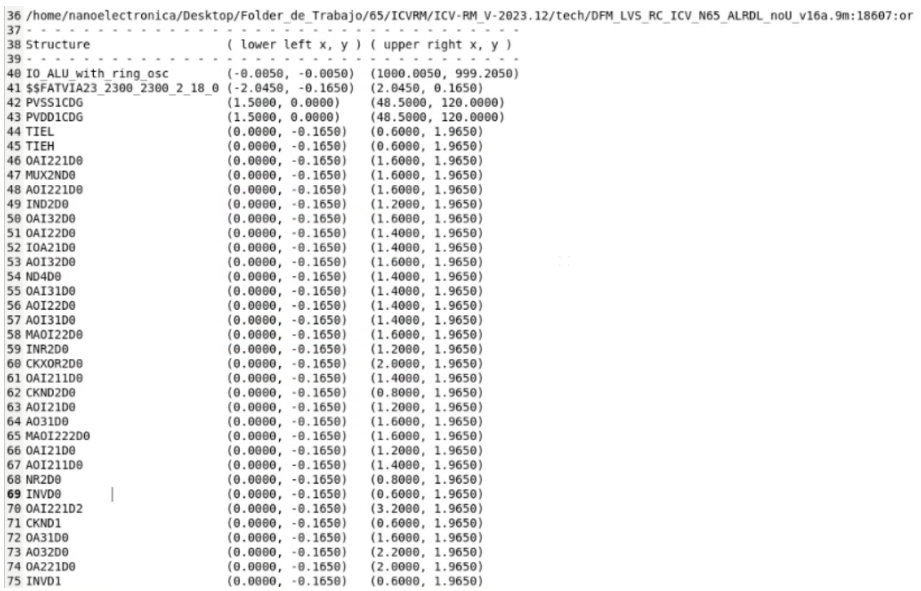
Nota. El diseño `ALU` obtuvo `LVS = PASS`. No obstante, el resultado general se reporta como `NOT CLEAN` en *DRC/Extraction* debido a 39 violaciones del tipo `FLOATING.psub`. Durante la etapa física, algunas celdas no se conectaron al nodo `VSS`, lo que generó regiones de substrato en flotación.

Figura 37. Reporte de errores de *layout* para `ALU` (parte 1).

```
Open   IO_ALU_with_ring_osc.LAYOUT_ERRORS   Save
~/Desktop/Folder_de_Trabajo/65/I...23.12/BUILDS/ALU/ICV_11-12-25/LVS

                    LAYOUT ERRORS RESULTS: ERRORS

                 ##### ####  ####   ###  ####   ###
                 #     #   # #   # #   # #   # #
                 ####  ####  ####  #   # ####   ###
                 #     #  #  #  #  #   # #  #      #
                 ##### #   # #   #  ###  #   # ####

================================================================================

Library name:     /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/ICVRM/ICV-RM_V-2023.12/ALU/Inputs/IO_ALU_with_ring_osc.gds
Structure name:   IO_ALU_with_ring_osc
Generated by:     IC Validator RHEL64 W-2024.09-SP4.11451981 2025/02/26
Runset name:      /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/ICVRM/ICV-RM_V-2023.12/tech/DFM_LVS_RC_ICV_N65_ALRDL_noU_v16a.9m
User name:        nanoelectronica
Time started:     2025/11/12 07:02:23PM
Time ended:       2025/11/12 07:02:44PM

Called as: icv -host_init 4 -vue -i /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/ICVRM/ICV-RM_V-2023.12/ALU/Inputs/
  IO_ALU_with_ring_osc.gds -s /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/ICVRM/ICV-RM_V-2023.12/ALU/Inputs/IO_ALU_with_ring_osc.icv -
  sf ICV -c IO_ALU_with_ring_osc -stc IO_ALU_with_ring_osc -f GDSII /home/nanoelectronica/Desktop/Folder_de_Trabajo/65/ICVRM/ICV-
  RM_V-2023.12/tech/DFM_LVS_RC_ICV_N65_ALRDL_noU_v16a.9m

                        ERROR SUMMARY

 FLOATING.psub : Floating psub is not allowed
   or ................................................ 39 violations found.



                        ERROR DETAILS


--------------------------------------------------------------------------------
FLOATING.psub : Floating psub is not allowed
--------------------------------------------------------------------------------
```

Nota. El archivo `IO_ALU_with_ring_osc.LAYOUT_ERRORS` resume 39 violaciones clasificadas como `FLOATING.psub`.

Figura 38. Reporte de errores de *layout* para `ALU` (parte 2).



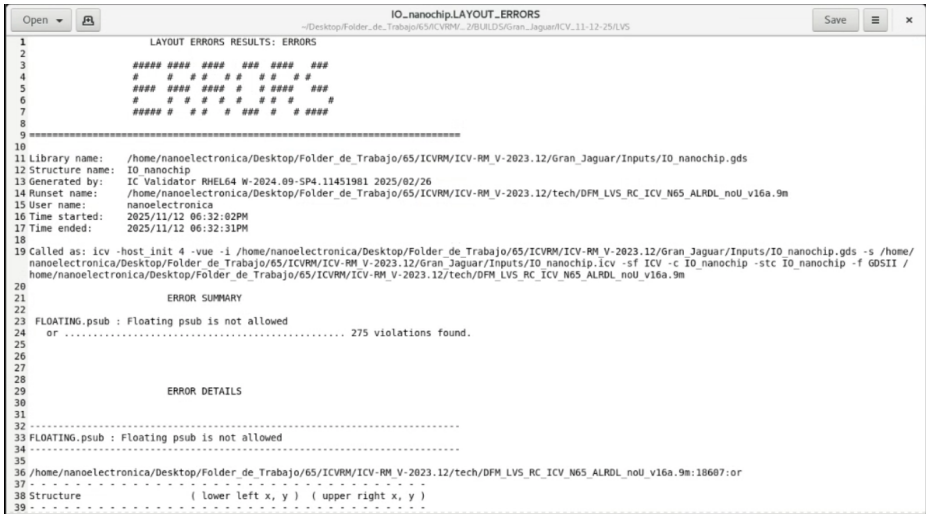
```
/home/nanoelectronica/Desktop/Folder_de_Trabajo/65/ICVRM/ICV-RM_V-2023.12/tech/DFM_LVS_RC_ICV_N65_ALRDL_noU_v16a.9m:18607:or
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Structure                    ( lower left x, y ) ( upper right x, y )
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
IO_ALU_with_ring_osc         (-0.0050, -0.0050)  (1000.0050, 999.2050)
$$FATVIA23_2300_2300_2_18_0 (-2.0450, -0.1650)  (2.0450, 0.1650)
PVSS1CDG                     (1.5000, 0.0000)    (48.5000, 120.0000)
PVDD1CDG                     (1.5000, 0.0000)    (48.5000, 120.0000)
TIEL                         (0.0000, -0.1650)   (0.6000, 1.9650)
TIEH                         (0.0000, -0.1650)   (0.6000, 1.9650)
OAI221D0                     (0.0000, -0.1650)   (1.6000, 1.9650)
MUX2ND0                      (0.0000, -0.1650)   (1.6000, 1.9650)
AOI221D0                     (0.0000, -0.1650)   (1.6000, 1.9650)
IND2D0                       (0.0000, -0.1650)   (1.2000, 1.9650)
OAI32D0                      (0.0000, -0.1650)   (1.6000, 1.9650)
OAI22D0                      (0.0000, -0.1650)   (1.4000, 1.9650)
IOA21D0                      (0.0000, -0.1650)   (1.4000, 1.9650)
AOI32D0                      (0.0000, -0.1650)   (1.6000, 1.9650)
ND4D0                        (0.0000, -0.1650)   (1.4000, 1.9650)
OAI31D0                      (0.0000, -0.1650)   (1.4000, 1.9650)
AOI22D0                      (0.0000, -0.1650)   (1.4000, 1.9650)
AOI31D0                      (0.0000, -0.1650)   (1.4000, 1.9650)
MAOI22D0                     (0.0000, -0.1650)   (1.6000, 1.9650)
INR2D0                       (0.0000, -0.1650)   (1.2000, 1.9650)
CKXOR2D0                     (0.0000, -0.1650)   (2.0000, 1.9650)
OAI211D0                     (0.0000, -0.1650)   (1.4000, 1.9650)
CKND2D0                      (0.0000, -0.1650)   (0.8000, 1.9650)
AOI21D0                      (0.0000, -0.1650)   (1.2000, 1.9650)
AO31D0                       (0.0000, -0.1650)   (1.6000, 1.9650)
MAOI222D0                    (0.0000, -0.1650)   (1.6000, 1.9650)
OAI21D0                      (0.0000, -0.1650)   (1.2000, 1.9650)
AOI211D0                     (0.0000, -0.1650)   (1.4000, 1.9650)
NR2D0                        (0.0000, -0.1650)   (0.8000, 1.9650)
INVD0                        (0.0000, -0.1650)   (0.6000, 1.9650)
OAI221D2                     (0.0000, -0.1650)   (3.2000, 1.9650)
CKND1                        (0.0000, -0.1650)   (0.6000, 1.9650)
OA31D0                       (0.0000, -0.1650)   (1.6000, 1.9650)
AO32D0                       (0.0000, -0.1650)   (2.2000, 1.9650)
OA221D0                      (0.0000, -0.1650)   (2.0000, 1.9650)
INVD1                        (0.0000, -0.1650)   (0.6000, 1.9650)
```

Nota. Se listan las estructuras y coordenadas implicadas, lo que evidencia las áreas en las que no se materializaron conexiones a `VSS` durante la etapa de *place & route*.

Figura 39. Resultado LVS para el diseño principal `Gran_Jaguar`.

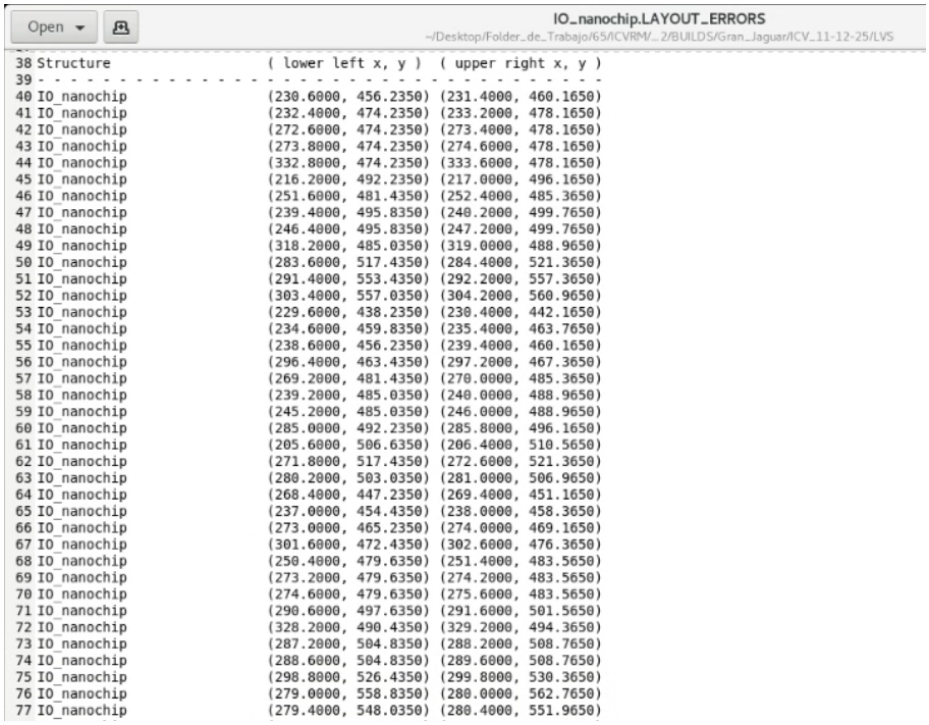
Nota. El diseño principal `Gran_Jaguar` obtuvo un resultado `LVS = PASS`, lo que valida la consistencia entre el netlist esquemático y el *layout* final. Sin embargo, el reporte global presenta un estado `NOT CLEAN` en *DRC/Extraction* debido a 275 violaciones del tipo `FLOATING.psub`. Durante la etapa física, múltiples celdas del bloque no lograron conectarse correctamente al nodo `VSS`, lo que generó substratos flotantes distribuidos a lo largo del chip.

Figura 40. Reporte de errores de *layout* para `Gran_Jaguar` (parte 1).



Nota. El archivo `IO_nanochip.LAYOUT_ERRORS` evidencia un total de 275 violaciones clasificadas como `FLOATING.psub`. Estas se relacionan con celdas no enlazadas a tierra durante la etapa de *place & route*.

Figura 41. Reporte de errores de *layout* para `Gran_Jaguar` (parte 2).



Open

IO_nanochip.LAYOUT_ERRORS

~/Desktop/Folder_de_Trabajo/65/ICVRM/... 2/BUILDS/Gran_Jaguar/ICV_11-12-25/LVS

38	Structure	(lower left x, y)	(upper right x, y)
39	- - - - - - - - - - - - - - - - - -	- -	- -
40	IO_nanochip	(230.6000, 456.2350)	(231.4000, 460.1650)
41	IO_nanochip	(232.4000, 474.2350)	(233.2000, 478.1650)
42	IO_nanochip	(272.6000, 474.2350)	(273.4000, 478.1650)
43	IO_nanochip	(273.8000, 474.2350)	(274.6000, 478.1650)
44	IO_nanochip	(332.8000, 474.2350)	(333.6000, 478.1650)
45	IO_nanochip	(216.2000, 492.2350)	(217.0000, 496.1650)
46	IO_nanochip	(251.6000, 481.4350)	(252.4000, 485.3650)
47	IO_nanochip	(239.4000, 495.8350)	(240.2000, 499.7650)
48	IO_nanochip	(246.4000, 495.8350)	(247.2000, 499.7650)
49	IO_nanochip	(318.2000, 485.0350)	(319.0000, 488.9650)
50	IO_nanochip	(283.6000, 517.4350)	(284.4000, 521.3650)
51	IO_nanochip	(291.4000, 553.4350)	(292.2000, 557.3650)
52	IO_nanochip	(303.4000, 557.0350)	(304.2000, 560.9650)
53	IO_nanochip	(229.6000, 438.2350)	(230.4000, 442.1650)
54	IO_nanochip	(234.6000, 459.8350)	(235.4000, 463.7650)
55	IO_nanochip	(238.6000, 456.2350)	(239.4000, 460.1650)
56	IO_nanochip	(296.4000, 463.4350)	(297.2000, 467.3650)
57	IO_nanochip	(269.2000, 481.4350)	(270.0000, 485.3650)
58	IO_nanochip	(239.2000, 485.0350)	(240.0000, 488.9650)
59	IO_nanochip	(245.2000, 485.0350)	(246.0000, 488.9650)
60	IO_nanochip	(285.0000, 492.2350)	(285.8000, 496.1650)
61	IO_nanochip	(205.6000, 506.6350)	(206.4000, 510.5650)
62	IO_nanochip	(271.8000, 517.4350)	(272.6000, 521.3650)
63	IO_nanochip	(280.2000, 503.0350)	(281.0000, 506.9650)
64	IO_nanochip	(268.4000, 447.2350)	(269.4000, 451.1650)
65	IO_nanochip	(237.0000, 454.4350)	(238.0000, 458.3650)
66	IO_nanochip	(273.0000, 465.2350)	(274.0000, 469.1650)
67	IO_nanochip	(301.6000, 472.4350)	(302.6000, 476.3650)
68	IO_nanochip	(250.4000, 479.6350)	(251.4000, 483.5650)
69	IO_nanochip	(273.2000, 479.6350)	(274.2000, 483.5650)
70	IO_nanochip	(274.6000, 479.6350)	(275.6000, 483.5650)
71	IO_nanochip	(290.6000, 497.6350)	(291.6000, 501.5650)
72	IO_nanochip	(328.2000, 490.4350)	(329.2000, 494.3650)
73	IO_nanochip	(287.2000, 504.8350)	(288.2000, 508.7650)
74	IO_nanochip	(288.6000, 504.8350)	(289.6000, 508.7650)
75	IO_nanochip	(298.8000, 526.4350)	(299.8000, 530.3650)
76	IO_nanochip	(279.0000, 558.8350)	(280.0000, 562.7650)
77	IO_nanochip	(279.4000, 548.0350)	(280.4000, 551.9650)

Nota. En esta sección se listan las coordenadas de las violaciones, correspondientes a distintas instancias dentro del *layout* del chip. Los errores se concentran en celdas estándar distribuidas en las regiones centrales del diseño.

8.4. ERC con tecnología 65 nm utilizando *reference methodology* (ICVRM)

El *Electrical Rule Check* (ERC) se ejecutó con el mismo *runset* tecnológico utilizado en LVS, ajustando únicamente los comandos del *environment setup*. Los resultados obtenidos reflejan violaciones de conectividad hacia `VSS` observadas durante la etapa física del diseño.

Cuadro 95. Parámetros ERC activados en el runset de 65 nm.

```
1 #define GATE_TO_PG_CHECK      // Verifica que toda puerta MOS tenga camino a potencia
2 #define PATH_CHECK            // Verifica la existencia de caminos validos hacia VDD/VSS
```

Nota. El cuadro muestra los parámetros habilitados para la ejecución de ERC dentro del *runset* de 65 nm. Estas opciones permiten verificar la conectividad eléctrica básica de compuertas y nodos de alimentación.

Figura 42. Fragmento del `runset` con reglas ERC activadas.

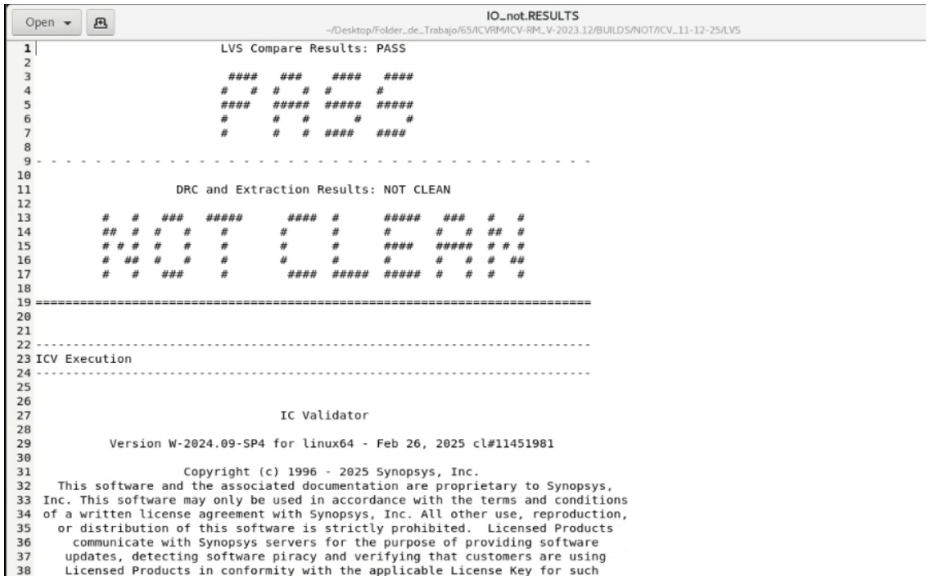
```
679 //#define USER_EQUIV_FILE                   // Turn on for user-specified equivalent point file flow
680 #ifdef USER_EQUIV_FILE
681 #include "./user.equiv"                     // Set equivalent point file here
682 #endif
683
684 /* EDIT: The following section contains all of the runset variables for RC extraction tools. */
685 //#define DFM_RULE                          // Turn on when this deck would like to include DFM rules
686 //#define RC_DECK                           // Turn on for LPE/RC extraction
687 //#define CROSS_REFERENCE                         // Turn on for source cross reference in LPE extraction
688 //#define ZERO_NRS_NRD                      // Turn off when this deck would calculate NRS and NRD
689 //#define FILTER_DGS_TIED_MOS               // Turn on to filter MOS with D, G and S tied together (default filter MOS with all pins tied)
690
691 #define WELL_TO_PG_CHECK                    // Default is on. Turn on to highlight if nwell connects to ground or psub connects to power.
692 #define GATE_TO_PG_CHECK                    // Default is off. Turn on to highlight if a mos gate directly connects to power or ground.
693 #define PATH_CHECK                          // Default is off. Turn on to highlight if
694                                             // (1) nodes have a path to power but no path to ground
695                                             // (2) nodes have a path to ground but no path to power
696                                             // (3) nodes have no path to power or ground
697                                             // (4) nodes have no path to any label net
698 #define DS_TO_PG_CHECK                      // Default is on. Turn on to highlight if drain connects to power and source connects to ground.
699 #define FLOATING_WELL_CHECK                 // Default is on. Turn on to highlight if well does not connect to power or ground.
700                                             // The nwell of moscaps and nwell-resistor are excluded
701 //#define NW_RING                           // Turn on to enable NW ring to separate the node from BULK
702
703 //#define extract_dnwdio                    // Turn on to extract parasitic pw-dnw, dnwpsub diodes.
704 //#define extract_pnwdio                    // Turn on to extract parasitic nw-psub diode.
705 //#define CELLIMP
706
707 //#define STD_LIB_9_TRACK                   // Turn on to estimate 9 track WPE on STD cell, do not set STD_LIB_11_TRACK = 1 at the same time.
708 //#define STD_LIB_11_TRACK                  // Turn on to estimate 11 track WPE on STD cell, do not set STD_LIB_9_TRACK = 1 at the same time.
709 //#define AP_UT                             // Turn on for thick AP RDL
710 #define MIMCAP                              // Turn on if no MIMCAP devices, or no ctm/cbm layer in RC techfiles
711 #define edram                               // Turn on if no eDram devices, or no p3/crown layer in RC techfiles
712
```

Nota. La figura muestra el fragmento del *runset* en el que se habilitan las verificaciones ERC empleadas en este trabajo. Estas reglas permiten identificar problemas de conectividad eléctrica hacia nodos de potencia y tierra.

8.4.1. Resultados ERC por diseño

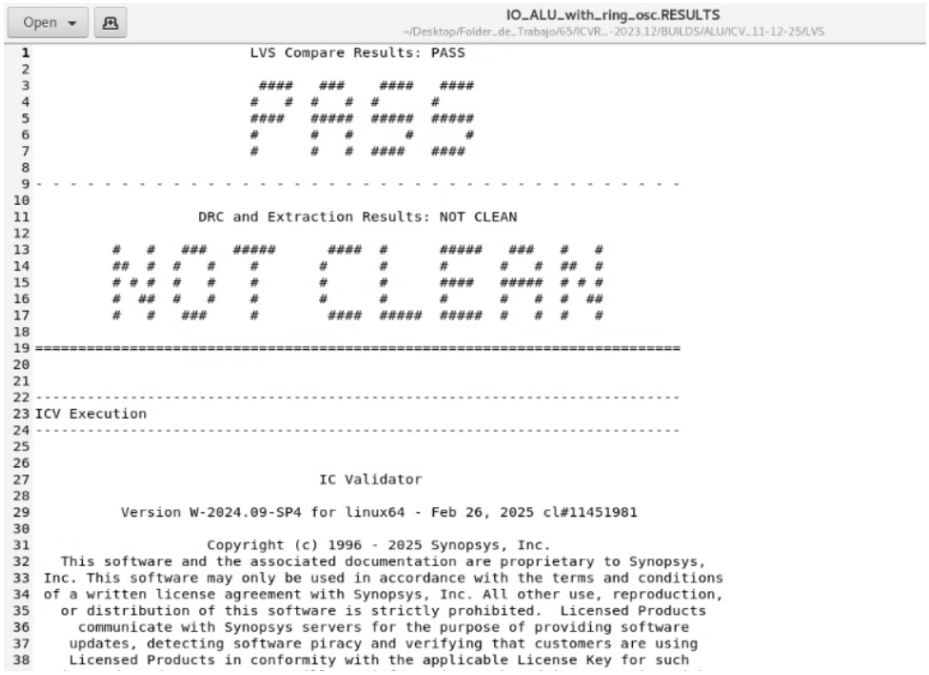
En esta subsección se presentan los resultados de ERC obtenidos para los diseños `NOT`, `ALU` y `Gran_Jaguar`. En los tres casos, la verificación reportó un estado `NOT CLEAN`, asociado a violaciones del tipo `FLOATING.psub` detectadas en la etapa física.

Figura 43. Resultado ERC para el inversor `NOT`.



Nota. El diseño `NOT` reportó un estado `NOT CLEAN` debido a violaciones `FLOATING.psub`. Durante la etapa física, algunas celdas no se conectaron correctamente al nodo `VSS`. Para el detalle de coordenadas y estructuras implicadas, véanse las Figuras 34 y 35.

Figura 44. Resultado ERC para el diseño `ALU`.



Nota. El diseño `ALU` reportó un estado `NOT CLEAN` con 39 violaciones `FLOATING.psub`. Algunas celdas no materializaron la conexión a `VSS` durante la etapa de *place & route*. El detalle puede consultarse en las Figuras 37 y 38.

Figura 45. Resultado ERC para el diseño principal `Gran_Jaguar`.

Nota. El diseño principal `Gran_Jaguar` reportó un estado `NOT CLEAN` con 275 violaciones `FLOATING.psub`. Varias celdas no se conectaron adecuadamente a `VSS`, lo que generó substratos en flotación en distintas regiones del chip. El detalle se presenta en las Figuras 40 y 41.

Cuadro 96. Resumen del estado ERC por diseño.

Diseño	Estado ERC	Número de violaciones	Descripción
NOT	NOT CLEAN	8	Violaciones FLOATING.psub; celdas sin conexión a VSS.
ALU	NOT CLEAN	39	Violaciones FLOATING.psub; conexiones a VSS no materializadas durante *place & route*.
Gran_Jaguar	NOT CLEAN	275	Violaciones FLOATING.psub; presencia de substratos en flotación en distintas regiones del chip.

Nota. La tabla resume el estado general de ERC para cada diseño verificado, así como la cantidad de violaciones detectadas y su causa principal.

La verificación ERC = NOT CLEAN en los tres diseños refleja la presencia de violaciones FLOATING.psub, derivadas de conexiones a VSS que no se materializaron durante la etapa de *place & route*. Este resultado complementa la verificación LVS, ya que evidencia condiciones eléctricas que no afectan la equivalencia lógica del circuito, pero sí comprometen su integridad eléctrica.

8.4.2. Síntesis del uso de ICVRM

La implementación del flujo ICVRM en tecnología de 65 nm permitió integrar síntesis física y verificación mediante un proceso reproducible y escalable. Mientras que la verificación LVS = PASS demostró la correspondencia entre el esquemático y el *layout*, la verificación ERC = NOT CLEAN puso en evidencia incidencias de conectividad hacia VSS originadas durante la etapa física. En conjunto, este capítulo establece una base técnica sólida para el perfeccionamiento del *sign-off* eléctrico en trabajos futuros.

CAPÍTULO 9

Conclusiones

- La réplica disciplinada del flujo en una tecnología previa permitió adquirir el conocimiento crítico para migrar a 65 nm. Aunque cambian las reglas de diseño, las bibliotecas, los *corners* y las restricciones, la arquitectura del flujo y sus etapas se mantienen, lo que reduce el riesgo y los tiempos de integración.
- Se alcanzó un *pre-floorplanning* viable en DCNXT y se generaron paquetes reproducibles para ICC2. En todos los bloques verificados se obtuvo `LVS = PASS`, lo que confirma la equivalencia topológica entre el *netlist* esquemático y el *layout* extraído.
- La generación y validación de celdas es esencial para la verificación funcional y temporal. Vincular correctamente las celdas del PDK con sus modelos funcionales (Verilog), temporales (`.lib`) y eléctricos (SPICE), e integrarlas con plataformas de emulación en FPGA, habilita una verificación temprana y consistente a lo largo del flujo.
- El cierre de LVS requiere una preparación rigurosa y coordinada de los archivos *netlist* esquemático y extraído, los *decks* o *runsets*, los archivos de equivalencias, la definición de *black boxes* y las reglas ERC, así como la articulación entre diseño, *layout* y verificación para resolver discrepancias de forma sistemática.
- La estandarización de *runsets*, plantillas `XML` y bitácoras facilita la trazabilidad, la reproducibilidad y la transferencia del flujo a cohortes futuras y a proyectos afines.

CAPÍTULO 10

Recomendaciones

- Establecer un cronograma con hitos y dependencias explícitas para cada etapa del flujo, y sostener reuniones breves de seguimiento, diarias o semanales según la fase, centradas en bloqueos y acuerdos. En un proceso lineal, cada entrega debe completarse antes del siguiente *handoff*, para prevenir cuellos de botella y mantener la continuidad operativa.
- Incrementar la validación temprana mediante circuitos sencillos para verificar herramientas, modelos y *scripts* antes de escalar a módulos complejos. Asimismo, automatizar regresiones de simulación y *sign-off* (LVS/DRC/ERC) mediante *scripts* o integración continua, a fin de detectar desviaciones de forma temprana, repetible y costo-eficiente.
- Evaluar la adopción de flujos y herramientas recientes de Synopsys que simplifican la configuración y aceleran las iteraciones, tales como Design Compiler NXT y la *reference methodology* (RM), priorizando la compatibilidad con nodos tecnológicos y las buenas prácticas de *timing closure*.
- Implementar una estrategia de *benchmarking* interno con métricas de área, potencia y *timing* para comparar librerías, versiones del flujo y configuraciones. Esto permite cuantificar mejoras, justificar migraciones y consolidar estándares de diseño.
- Profundizar en el uso de herramientas avanzadas como Fusion Compiler, que integran síntesis, *place & route* y verificación física en un solo flujo, con el fin de optimizar los resultados de desempeño y área. Asimismo, se recomienda investigar estrategias jerárquicas y de bajo consumo para futuros proyectos, mediante la aplicación de técnicas de *clock gating* y análisis de potencia dinámica, con el propósito de fortalecer la eficiencia del diseño y la escalabilidad del flujo ASIC.
- Ejecutar verificación *power-aware* basada en UPF, incluyendo aislamiento, *level shifters* y *power switches*, junto con ERC específicos por dominio de potencia, a fin de mitigar escenarios de substratos flotantes o caminos de potencia incompletos, como los observados en este proyecto.

CAPÍTULO 11

Referencias

[1] 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,» Trabajo de graduación, Universidad del Valle de Guatemala, Guatemala, 2014.

[2] L. A. N. Nájera 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,» Trabajo de graduación, Universidad del Valle de Guatemala, Guatemala, 2019.

[3] S. H. Rubio Vásquez, «Definición del flujo de diseño para fabricación de un chip con tecnología VLSI CMOS,» Trabajo de graduación, Universidad del Valle de Guatemala, Guatemala, 2019.

[4] M. S. Sibrián Illescas, «Verificación de reglas de diseño (DRC) para el desarrollo de un flujo funcional de un circuito integrado con tecnología nanométrica,» Trabajo de graduación, Universidad del Valle de Guatemala, Guatemala, 2020.

[5] J. R. Rubio García, «Etapa de verificación física de Diseño en Silicio vs. Esquemático (LVS) en el flujo de diseño para un chip a nanoescala,» Trabajo de graduación, Universidad del Valle de Guatemala, Guatemala, 2020.

[6] M. G. Flores Espino, «Corrección de anillo de entradas/salidas y pruebas de antena y ERC para la definición del flujo de diseño del primer chip con tecnología nanométrica desarrollado en Guatemala,» Trabajo de graduación, Universidad del Valle de Guatemala, Guatemala, 2020.

[7] K. S. Cardona Polanco, «Mejoramiento del proceso de síntesis lógica llevada a cabo para la elaboración de un circuito integrado a escala nanométrica,» Trabajo de graduación, Universidad del Valle de Guatemala, Guatemala, 2021.

[8] E. G. Ruballos, «Implementación y verificación de la funcionalidad de código obtenido durante la síntesis lógica de la arquitectura del nanochip Gran Jaguar utilizando una plataforma de desarrollo con un FPGA Xilinx Virtex-5 Genesys,» Trabajo de graduación, Universidad del Valle de Guatemala, Guatemala, 2022.

[9] L. R. Velásquez, «Procesamiento de las señales generadas por un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC montado en un FPGA Digilent Genesys Board, demostrando el funcionamiento mediante una aplicación y sintetizador digital,» Trabajo de graduación, Universidad del Valle de Guatemala, Guatemala, 2022.

[10] L. M. Ruiz Vásquez, «Diseño y fabricación de un circuito integrado con tecnología de 65 nm usando librerías de diseño de TSMC: pruebas finales de funcionamiento en HSPICE, generación y documentación de la síntesis física, verificaciones de Antena y DRC,» Trabajo de graduación, Universidad del Valle de Guatemala, Guatemala, 2024.

[11] D. S. Alvarado Mota, «Diseño de un circuito integrado con tecnología de 65 nm utilizando librerías de diseño de TSMC: pruebas de LVS, ERC y extracción de parásitos,» Trabajo de graduación, Universidad del Valle de Guatemala, Guatemala, 2024.

[12] R. R. Schaller, «Moore's Law: Past, Present and Future,» *IEEE Spectrum*, vol. 34, n.º 6, págs. 52-59, 1997.

[13] World Semiconductor Trade Statistics, «WSTS Semiconductor Market Forecast Spring 2024,» WSTS, Report, 2024.

[14] M.
bibinitperiod Company, «The $1 Trillion Semiconductor Industry,» McKinsey Global Institute, Industry Report, 2022.

[15] N. H. E. Weste y D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4.ª ed. Addison–Wesley, 2011.

[16] M. J. S. Smith, *Application–Specific Integrated Circuits*. Addison–Wesley, 1997.

[17] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*, 3.ª ed. Springer, 1999.

[18] P. Radhakrishnan, *ASIC Design Flow*, Technical tutorial, 2000. dirección: `http://gcedocs.tripod.com/docs/asic_flow.pdf`.

[19] N. Deshpande y K. B. Sowmya, «A Review on ASIC Flow Employing EDA Tools by Synopsys,» *SSRG International Journal of VLSI & Signal Processing*, vol. 7, n.º 1, págs. 15-19, 2020.

[20] *Design Compiler NXT Datasheet*, Synopsys, Inc., 2022. dirección: `https://www.synopsys.com/content/dam/synopsys/implementation%26signoff/datasheets/design-compiler-nxt-ds.pdf`.

[21] A. B. Kahng, J. Lienig, I. L. Markov y J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, 2.ª ed. Springer, 2022.

[22] *Fusion Compiler RTL-to-GDSII Design Solution*, Synopsys, Inc., 2025. dirección: `https://www.synopsys.com/implementation-and-signoff/physical-implementation/fusion-compiler.html`.

[23] *IC Compiler II Place-and-Route Solution*, Synopsys, Inc., 2025. dirección: `https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html`.

[24] *PrimeTime Static Timing Analysis*, Synopsys, Inc., 2025. dirección: `https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html`.

[25] *VCS Functional Verification Solution*, Synopsys, Inc., 2021. dirección: https://www.synopsys.com/verification/simulation/vcs.html.

[26] *IC Validator High-Performance Signoff Physical Verification*, Synopsys, Inc., 2025. dirección: https://www.synopsys.com/implementation-and-signoff/physical-verification.html.

[27] SemiEngineering. «Design Rule Checking for Advanced Nodes. »dirección: https://semiengineering.com/design-rule-checking-for-advanced-nodes.

[28] SemiEngineering. «Layout Versus Schematic Challenges. »dirección: https://semiengineering.com/layout-versus-schematic-challenges.

[29] Wikipedia contributors. «Electrical Rule Check. »dirección: https://en.wikipedia.org/wiki/Electrical_rule_check.

[30] Wikipedia contributors. «Static Timing Analysis. »dirección: https://en.wikipedia.org/wiki/Static_timing_analysis.

CAPÍTULO 12

Anexos

12.1. Replicación en tecnología de 180 nm

12.1.1. Archivos generados

Generación del archivo ICV para preparación de librerías

Para generar el archivo *ICV* se utilizaron las librerías `CORE.v` e `IO.v`, correspondientes a la tecnología de 180 nm. En primer lugar, ambos archivos se concatenaron para obtener un único archivo con todas las celdas.

Cuadro 97. Comando para concatenar las librerías *verilog*.

```
cat CORE.v IO.v > headers2025.v
```

Nota. Elaboración propia.

Con el archivo resultante `headers2025.v`, se generó la cabecera *SPICE* mediante NetTran de IC Validator, con el fin de disponer de subcircuitos compatibles para la extracción.

Cuadro 98. Comando para generar la cabecera *SPICE* a partir de las librerías.

```
icv_nettran -verilog headers2025.v -outType SPICE -outName headers2025.sp
```

Nota. Elaboración propia.

Finalmente, se editaron las primeras líneas de `headers2025.sp` para declarar las fuentes globales e incluir los subcircuitos del PDK de 180 nm.

Cuadro 99. Líneas agregadas a la cabecera *SPICE* para la réplica en 180 nm.

```
.GLOBAL VDD VSS VDDPST VSSPST
*.EQUATION
```

```
*.SCALE METER
*.MEGA
.PARAM
.INCLUDE /home/nanoelectronica/Desktop/Folder_de_Trabajo/180/LVS2025/source.added
```

Nota. Elaboración propia.

12.1.2. Adecuación de archivos para LVS

El siguiente paso consistió en depurar el *netlist* generado durante la síntesis lógica (`VERILOG2025.v`). Para ello, se eliminaron las instancias `PCORNER` y `FILLER`/`FILL`, así como las conexiones explícitas a potencia (`.VDD(VDD)` y `.VSS(VSS)`), debido a que corresponden a elementos puramente físicos y no aportan conectividad lógica para la verificación LVS. Como medida de seguridad, esta limpieza se realizó sobre una copia denominada `Verilog_Clean_2025.v`.

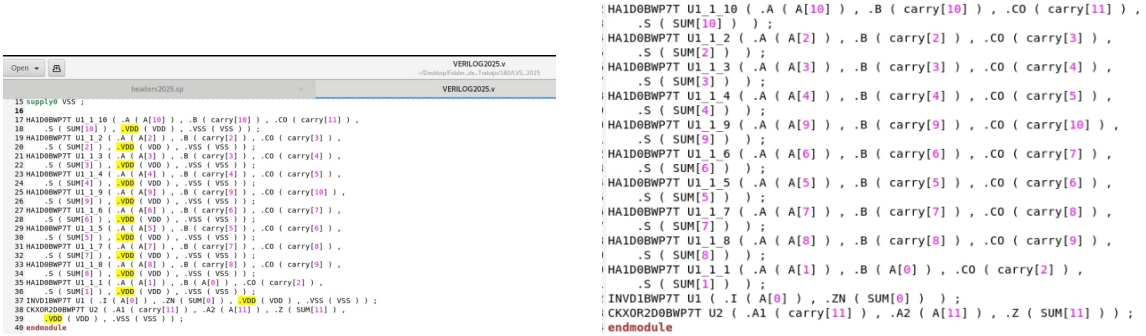
Figura 46. Depuración de celdas de esquina `PCORNER` en el *netlist gate-level*.

Nota. Elaboración propia. La figura ilustra la eliminación de celdas `PCORNER` del *netlist gate-level* para asegurar consistencia estructural y evitar discrepancias durante la comparación entre esquemático y *layout*.

Figura 47. Depuración de celdas de relleno `PFILLER`/`FILL` en el *netlist gate-level*.

Nota. Elaboración propia. La figura muestra la eliminación de celdas de relleno heredadas del flujo físico, con el propósito de conservar únicamente la información relevante para la verificación de conectividad.

Figura 48. Remoción de mapeos explícitos de potencia en coherencia con los nodos `.GLOBAL`.



```
supply0 VSS ;

HA1D0BWP7T U1_1_10 ( .A ( A[10] ) , .B ( carry[10] ) , .CO ( carry[11] ) ,
    .S ( SUM[10] ) ) ;
HA1D0BWP7T U1_1_2 ( .A ( A[2] ) , .B ( carry[2] ) , .CO ( carry[3] ) ,
    .S ( SUM[2] ) ) ;
HA1D0BWP7T U1_1_3 ( .A ( A[3] ) , .B ( carry[3] ) , .CO ( carry[4] ) ,
    .S ( SUM[3] ) ) ;
HA1D0BWP7T U1_1_4 ( .A ( A[4] ) , .B ( carry[4] ) , .CO ( carry[5] ) ,
    .S ( SUM[4] ) ) ;
HA1D0BWP7T U1_1_9 ( .A ( A[9] ) , .B ( carry[9] ) , .CO ( carry[10] ) ,
    .S ( SUM[9] ) ) ;
HA1D0BWP7T U1_1_6 ( .A ( A[6] ) , .B ( carry[6] ) , .CO ( carry[7] ) ,
    .S ( SUM[6] ) ) ;
HA1D0BWP7T U1_1_5 ( .A ( A[5] ) , .B ( carry[5] ) , .CO ( carry[6] ) ,
    .S ( SUM[5] ) ) ;
HA1D0BWP7T U1_1_7 ( .A ( A[7] ) , .B ( carry[7] ) , .CO ( carry[8] ) ,
    .S ( SUM[7] ) ) ;
HA1D0BWP7T U1_1_8 ( .A ( A[8] ) , .B ( carry[8] ) , .CO ( carry[9] ) ,
    .S ( SUM[8] ) ) ;
HA1D0BWP7T U1_1_1 ( .A ( A[1] ) , .B ( A[0] ) , .CO ( carry[2] ) ,
    .S ( SUM[1] ) ) ;
INVD1BWP7T U1 ( .I ( A[0] ) , .ZN ( SUM[0] ) ) ;
CKXOR2D0BWP7T U2 ( .A1 ( carry[11] ) , .A2 ( A[11] ) , .Z ( SUM[11] ) ) ;
endmodule
```

Nota. Elaboración propia. La figura compara el *netlist* original, en el que cada instancia conectaba explícitamente `VDD` y `VSS`, con la versión depurada en la que dichos mapeos fueron eliminados para mantener consistencia con los nodos `.GLOBAL` definidos en la biblioteca.

12.1.3. Ejecución de LVS y ERC en tecnología de 180 nm

Cuadro 100. Generación del *netlist* ICV/CDL a partir de `Verilog_Clean_2025.v`.

```
icv_nettran -verilog Verilog_Clean_2025.v -sp headers2025.sp -outType ICV
    -outName Gran_Jaguar_180_2025
```

Nota. Elaboración propia.

Configuración del *runset* y ejecución de LVS

Para ejecutar LVS, se ajustó el bloque *Environment setup* del *runset* con las rutas reales del proyecto: `library_name`, apuntando a `chip.gds`; `cell`, con el nombre exacto de la *top cell* del GDS; `SCHEMATIC_TOPCELL`; y `sch_db`, con las rutas al esquemático de referencia en formato ICV/CDL (`CDL_GranJaguar_180_2025.icv`) y, cuando correspondió, al archivo de librería *SPICE*/CDL (`unit.cdl`). Además, se declararon las *black boxes* requeridas de acuerdo con el *netlist gate-level*, respetando el mismo orden de pines definido en `CORE.v` y `IO.v`. Con ello, el flujo quedó preparado para ejecutar LVS.

Figura 49. Configuración del bloque *Environment setup* en el *runset* para LVS en 180 nm.

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

library(
        library_name = "/home/nanoelectronica/Desktop/Folder_de_Trabajo/180/LVS_2025/chip.gds",
        cell = "circuit",
        format = GDSII
);

SCHEMATIC_TOPCELL : string = "circuit";         // Set schematic top cell name here
sch_db = schematic(
        schematic_file = {{"/home/nanoelectronica/Desktop/Folder_de_Trabajo/180/LVS_2025/Gran_Jaguar_180_2025.icv", ICV}},
        schematic_library_file = {{"/home/nanoelectronica/Desktop/Folder_de_Trabajo/180/LVS_2025/unit.cdl", SPICE}},
        expand_multiple_devices = true,
        spice_settings = {slash_is_space = false}
);
//#define USER_EQUIV_FILE                       // Turn on for user-specified equivalent point file flow
#ifdef USER_EQUIV_FILE
#include "./user.equiv"                         // Set equivalent point file here
#endif



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

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

Nota. Elaboración propia. La figura muestra la configuración de las rutas hacia `chip.gds`, `CDL_GranJaguar_180_2025.icv`, `unit.cdl` y los nombres de *top cell* requeridos para la ejecución correcta de LVS.

Cuadro 101. Comando de ejecución de LVS con IC Validator.

```
icv -i chip.gds -c circuit -s Gran_Jaguar_180_2025.icv -sf ICV -vue
    /ruta/al/LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a
```

Nota. Elaboración propia.

Figura 50. Declaración de *black boxes* en el *runset* mediante `lvs_black_box_options`.

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

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

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

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

lvs_black_box_options(
        equiv_cells ={{schematic_cell = "PDDW0204SCDG", layout_cell = "PDDW0204SCDG"}},
        remove_schematic_ports = {"I", "DS", "OEN", "PAD", "C", "PE", "IE"}
);


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

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

Nota. Elaboración propia. La figura ejemplifica la definición de *black boxes* mediante la asociación entre celda de esquemático, celda de *layout* y puertos omitidos durante la comparación.

Lectura de resultados de LVS y ERC

Al finalizar la corrida, IC Validator generó dos archivos principales dentro de la carpeta del proyecto: `circuit.RESULTS` y `circuit.LVS_ERRORS`. En `circuit.RESULTS` puede observarse el estado `CLEAN` y, cuando corresponde, `PASS`. Por su parte, `circuit.LVS_ERRORS` presenta el resumen de la comparación y cualquier discrepancia detectada. Los detalles adicionales también pueden consultarse en `run_details/`*`compare`*`/<runset>.log`.

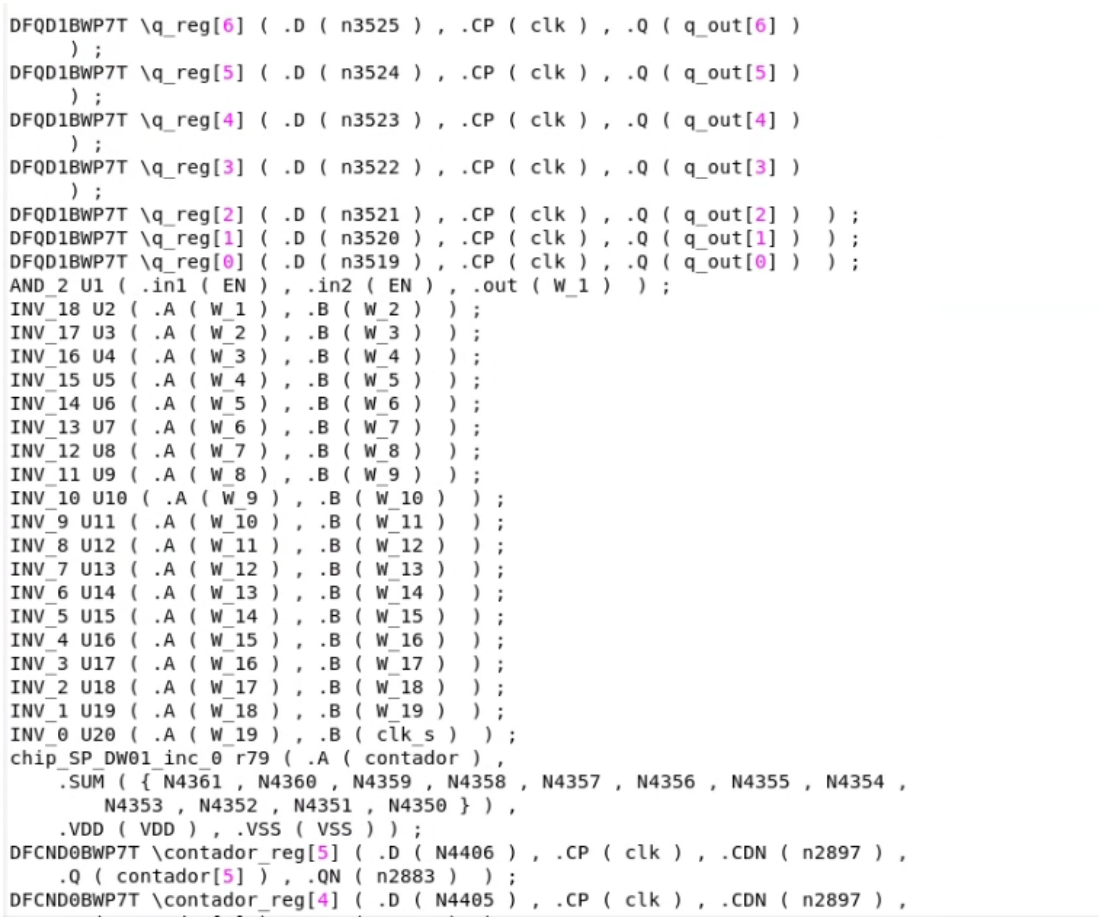
Configuración y ejecución de ERC

Para ejecutar ERC, se utilizó el mismo *runset*, pero con modificaciones puntuales en las banderas: se comentó `RC_DECK` y se habilitaron `#define GATE_TO_PG_CHECK` y `#define PATH_CHECK`. El comando de ejecución se mantuvo sin cambios.

12.1.4. Comparativa entre LVS y ERC

Ambas pruebas son fundamentales dentro de la verificación física de un diseño. LVS asegura que la conectividad lógica del *layout* coincida con la del esquemático, mientras que ERC verifica el cumplimiento de las reglas eléctricas del proceso tecnológico, incluyendo aspectos como integridad de señal, distribución de potencia y protección contra descargas electrostáticas. Aunque ambas corridas se ejecutan de forma similar, cada una utiliza una

Figura 51. Referencia de celdas de la librería de 180 nm utilizada para verificar nombres y orden de pines.



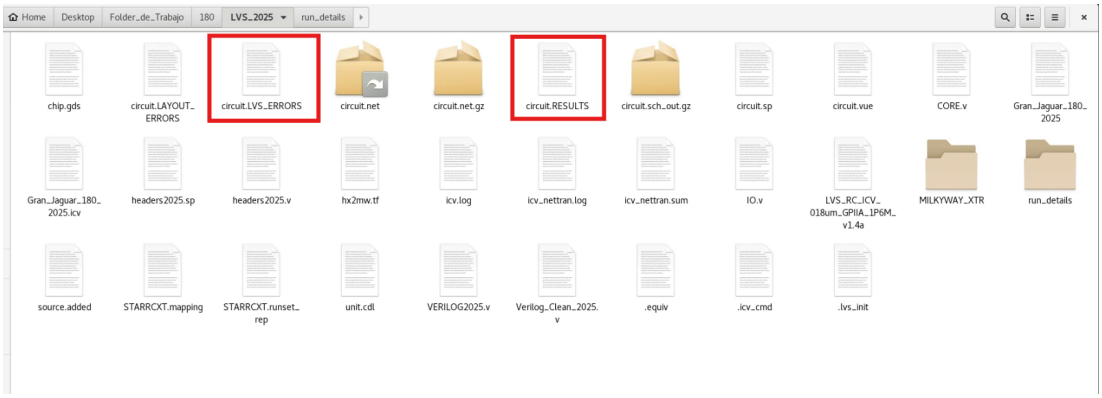
```
DFQD1BWP7T \q_reg[6] ( .D ( n3525 ) , .CP ( clk ) , .Q ( q_out[6] )
     ) ;
DFQD1BWP7T \q_reg[5] ( .D ( n3524 ) , .CP ( clk ) , .Q ( q_out[5] )
     ) ;
DFQD1BWP7T \q_reg[4] ( .D ( n3523 ) , .CP ( clk ) , .Q ( q_out[4] )
     ) ;
DFQD1BWP7T \q_reg[3] ( .D ( n3522 ) , .CP ( clk ) , .Q ( q_out[3] )
     ) ;
DFQD1BWP7T \q_reg[2] ( .D ( n3521 ) , .CP ( clk ) , .Q ( q_out[2] )  ) ;
DFQD1BWP7T \q_reg[1] ( .D ( n3520 ) , .CP ( clk ) , .Q ( q_out[1] )  ) ;
DFQD1BWP7T \q_reg[0] ( .D ( n3519 ) , .CP ( clk ) , .Q ( q_out[0] )  ) ;
AND_2 U1 ( .in1 ( EN ) , .in2 ( EN ) , .out ( W_1 )  ) ;
INV_18 U2 ( .A ( W_1 ) , .B ( W_2 )  ) ;
INV_17 U3 ( .A ( W_2 ) , .B ( W_3 )  ) ;
INV_16 U4 ( .A ( W_3 ) , .B ( W_4 )  ) ;
INV_15 U5 ( .A ( W_4 ) , .B ( W_5 )  ) ;
INV_14 U6 ( .A ( W_5 ) , .B ( W_6 )  ) ;
INV_13 U7 ( .A ( W_6 ) , .B ( W_7 )  ) ;
INV_12 U8 ( .A ( W_7 ) , .B ( W_8 )  ) ;
INV_11 U9 ( .A ( W_8 ) , .B ( W_9 )  ) ;
INV_10 U10 ( .A ( W_9 ) , .B ( W_10 )  ) ;
INV_9 U11 ( .A ( W_10 ) , .B ( W_11 )  ) ;
INV_8 U12 ( .A ( W_11 ) , .B ( W_12 )  ) ;
INV_7 U13 ( .A ( W_12 ) , .B ( W_13 )  ) ;
INV_6 U14 ( .A ( W_13 ) , .B ( W_14 )  ) ;
INV_5 U15 ( .A ( W_14 ) , .B ( W_15 )  ) ;
INV_4 U16 ( .A ( W_15 ) , .B ( W_16 )  ) ;
INV_3 U17 ( .A ( W_16 ) , .B ( W_17 )  ) ;
INV_2 U18 ( .A ( W_17 ) , .B ( W_18 )  ) ;
INV_1 U19 ( .A ( W_18 ) , .B ( W_19 )  ) ;
INV_0 U20 ( .A ( W_19 ) , .B ( clk_s )  ) ;
chip_SP_DW01_inc_0 r79 ( .A ( contador ) ,
    .SUM ( { N4361 , N4360 , N4359 , N4358 , N4357 , N4356 , N4355 , N4354 ,
        N4353 , N4352 , N4351 , N4350 } ) ,
    .VDD ( VDD ) , .VSS ( VSS ) ) ;
DFCND0BWP7T \contador_reg[5] ( .D ( N4406 ) , .CP ( clk ) , .CDN ( n2897 ) ,
    .Q ( contador[5] ) , .QN ( n2883 )  ) ;
DFCND0BWP7T \contador_reg[4] ( .D ( N4405 ) , .CP ( clk ) , .CDN ( n2897 ) ,
```

Nota. Elaboración propia. La figura presenta una referencia visual de las celdas de la librería de 180 nm, útil para verificar consistencia en los nombres de pines y en el orden de conexión al definir *black boxes*.

Figura 52. Contenido del archivo `circuit.RESULTS` tras la ejecución de LVS en 180 nm.

Nota. Elaboración propia. La imagen muestra un ejemplo de los estados `LVS Compare Results: PASS` y `DRC and Extraction Results: CLEAN`, los cuales indican equivalencia entre *netlist* y *layout*, así como ausencia de inconsistencias en la extracción.

Figura 53. Archivos principales generados en la carpeta de trabajo después de la ejecución de LVS.



Nota. Elaboración propia. La figura destaca los archivos `circuit.LVS_ERRORS` y `circuit.RESULTS`, utilizados para verificar el estado final de la comparación lógica-física.

configuración distinta del *runset*. Como puede observarse, LVS verifica un conjunto reducido de reglas centradas en conectividad, mientras que ERC realiza una validación eléctrica considerablemente más amplia.

Figura 54. Resumen del archivo `circuit.LVS_ERRORS` con el resultado final de comparación.

Nota. Elaboración propia. La figura presenta el resultado final de la comparación, junto con el conteo de *black boxes* y equivalencias reconocidas por la herramienta.

Figura 55. Ajustes del *runset* para la ejecución de ERC en 180 nm.

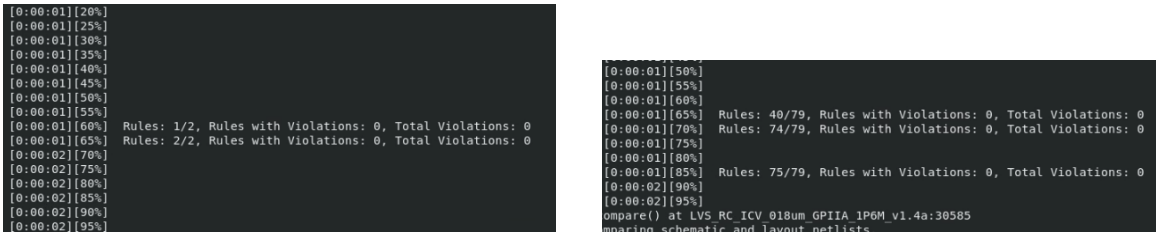
```
//////////////////////
// ENVIRONMENT SETUP //
//////////////////////

library(
        library_name = "/home/nanoelectronica/Desktop/Folder_de_Trabajo/180/LVS_2025/chip.gds",
        cell = "circuit",
        format = GDSII
);

SCHEMATIC_TOPCELL : string = "circuit";         // Set schematic top cell name here
sch_db = schematic(
        schematic_file = {{"/home/nanoelectronica/Desktop/Folder_de_Trabajo/180/LVS_2025/Gran_Jaguar_180_2025.icv", ICV}},
        schematic_library_file = {{"/home/nanoelectronica/Desktop/Folder_de_Trabajo/180/LVS_2025/unit.cdl", SPICE}},
        expand_multiple_devices = true,
        spice_settings = {slash_is_space = false}
);
//#define USER_EQUIV_FILE                       // Turn on for user-specified equivalent point file flow
#ifdef USER_EQUIV_FILE
#include "./user.equiv"                         // Set equivalent point file here
#endif



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

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

Nota. Elaboración propia. La figura muestra la desactivación de `RC_DECK` y la activación de `GATE_TO_PG_CHECK` y `PATH_CHECK`, necesarias para evaluar condiciones eléctricas como compuertas flotantes o caminos no válidos.

Figura 56. Archivos de salida generados tras la ejecución de ERC en 180 nm.

Nota. Elaboración propia. Los archivos generados y su lectura son análogos a los obtenidos en LVS. Su estructura permite identificar de forma directa las condiciones eléctricas anómalas detectadas durante la verificación.

Figura 57. Comparación del alcance de reglas evaluadas por LVS y ERC.



Nota. Elaboración propia. La figura compara el alcance de las verificaciones. Mientras que LVS se concentra en la conectividad y la equivalencia entre *netlist* y *layout*, ERC amplía el análisis a múltiples reglas eléctricas adicionales, lo que permite detectar nodos flotantes, cortocircuitos o condiciones eléctricas no válidas.

CAPÍTULO 13

Glosario

Design Compiler: herramienta de Synopsys utilizada para realizar la síntesis lógica, que convierte el diseño RTL en una *netlist* lista para la implementación física

Design Flow: proceso estructurado que se sigue para crear un chip, desde la especificación de alto nivel hasta la implementación física

Design Rule Check (DRC): proceso de verificación que asegura que el diseño cumpla con las reglas de fabricación específicas de la tecnología, con base en la revisión de dimensiones, espaciamiento y otros parámetros físicos

Design Vision: interfaz gráfica de Synopsys que permite visualizar, analizar y optimizar el diseño tras la síntesis lógica realizada con Design Compiler

Discovery Visual Environment (DVE): interfaz gráfica de usuario (GUI) interactiva utilizada para analizar y depurar diseños en SystemVerilog, VHDL, Verilog y SystemC

Front-End: etapa del diseño de circuitos integrados enfocada en la descripción de alto nivel del sistema, generalmente en RTL, antes de la síntesis física

Hardware Description Language (HDL): lenguaje de descripción de hardware utilizado para modelar el comportamiento y la estructura de circuitos electrónicos, como VHDL o Verilog

Netlist de una síntesis lógica: descripción de un circuito después de la síntesis, que detalla las puertas lógicas y los *flip-flops* utilizados, junto con sus conexiones, lista para la implementación física en un chip

Synopsys: empresa líder en el desarrollo de software para diseño electrónico automatizado (EDA), utilizado para diseñar y verificar chips de silicio en tecnología VLSI

Síntesis Física: proceso que transforma la *netlist* lógica en un diseño físico, mediante el posicionamiento y enrutamiento de los componentes para que el chip sea fabricable en silicio

Síntesis Lógica: proceso de convertir un diseño RTL (*register transfer level*) escrito en HDL en una lista de puertas lógicas y *flip-flops*, generando una *netlist* que representa el diseño en términos de componentes lógicos

Taiwan Semiconductor Manufacturing Company (TSMC): empresa líder en la fabricación de semiconductores, que produce circuitos integrados para diversas tecnologías avanzadas, como CMOS

Verilog: lenguaje de descripción de hardware utilizado para modelar el comportamiento de sistemas digitales, ampliamente empleado en el diseño de circuitos integrados

Verilog Compiler Simulator (VCS): herramienta de Synopsys para simular diseños en Verilog, utilizada para verificar el comportamiento funcional del diseño antes de la síntesis física

Very Large Scale Integration (VLSI): proceso de diseño de circuitos integrados que permite integrar miles o millones de transistores en un solo chip