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

Lourdes María Ruiz Vásquez

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

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 presentado por Lourdes María Ruiz Vásquez
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2025

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

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 presentado por Lourdes María Ruiz Vásquez
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2025

Vo.Bo.:

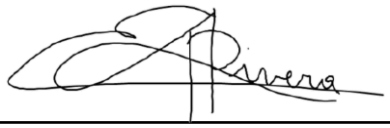
(f) _____

M. Sc. Carlos Esquit

Tribunal Examinador:

(f) _____

M.Sc. Carlos Esquit



(f) _____

Dr. Luis Alberto Rivera Estrada

(f) _____

Ing. Kurt Emmanuel Kellner

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

Prefacio

La tecnología ha evolucionado de forma impresionante en las últimas décadas, y la nanoelectrónica es prueba de ello. Lo que antes parecía ciencia ficción ahora es una realidad, con dispositivos cada vez más pequeños y rápidos, capaces de procesar grandes cantidades de información en fracciones de segundo. Contribuir con el desarrollo del primer circuito integrado a escala nanométrica de la región no solo representa un logro personal, sino que también marca un hito que abre las puertas a nuevas oportunidades, posicionándonos a la vanguardia de la innovación tecnológica global.

Este proyecto no habría sido posible sin el apoyo de varias personas a quienes deseo expresar mi agradecimiento. A mi familia, por su amor incondicional y por brindarme la oportunidad de recibir una educación de calidad. A mis amigos, cuyo ánimo y compañía fueron fundamentales para superar los desafíos que se presentaron. A MSc. Carlos Esquit, quien ha sido un guía en mi formación y ha desempeñado un papel crucial en la difusión de este campo revolucionario en la universidad. Finalmente, a mi asesor, MSc. Jonathan de los Santos, cuyo apoyo constante, orientación y retroalimentación fueron cruciales para la culminación de este trabajo.

Índice

Lista de figuras

Lista de cuadros

Resumen

Este trabajo se centra en la mejora del flujo de síntesis física de circuitos integrados, construido sobre las bases establecidas en años anteriores. El objetivo fue no solo automatizar aún más el flujo existente, sino también abordar los errores que emergen al ejecutar los scripts de síntesis.

Dado que se trata de un proyecto intergeneracional, se buscó replicar y comprender los avances previos, familiarizándose con herramientas como *IC Compiler II*, *IC Validator, VCS* y *WaveView*. A partir de esto, se identificaron áreas de mejora, tales como la generación de un archivo Verilog con instancias de componentes lógicos y físicos, la definición de variables para valores repetidos y la eliminación de cálculos de tamaño previamente codificados de forma fija. Tras implementar estos cambios, se realizaron síntesis físicas de varios circuitos, comenzando con circuitos combinacionales simples y avanzando hacia circuitos secuenciales más complejos, incluyendo la síntesis del circuito "El Gran Jaguar".

Después de ajustar el *runset* de inserción de metal, las verificaciones *DRC* confirmaron que la mayoría de los circuitos cumplían con los requisitos mínimos de fabricación en todas las capas de metal. La única excepción fue "El Gran Jaguar", que presentó un error en la capa Metal 2, aunque estuvo muy cerca de cumplir con las especificaciones. Este inconveniente se espera resolver con la implementación de la tecnología de 65 nm, ya que *TSMC* decidió descontinuar la tecnología de 180 nm. Por otro lado, las verificaciones de antena mostraron resultados satisfactorios en todos los circuitos.

También se analizaron los archivos Verilog generados tras la síntesis y se identificó que no representaban completamente el *layout* físico del diseño. Asimismo, se documentaron detalladamente todas las etapas del flujo, incluyendo videos explicativos sobre el uso de las herramientas y la creación de librerías *NDM*. Este esfuerzo busca facilitar el aprendizaje y la adopción de nuevas tecnologías por parte de futuras generaciones. Finalmente, se realizaron pruebas funcionales del núcleo de algunos circuitos utilizando *HSPICE* y *WaveView*, validando así su correcto funcionamiento.

De cara al futuro, se recomienda trabajar en un flujo de diseño más automatizado que sea capaz de ajustar las dimensiones del circuito en función de características específicas como el número de puertos, el tamaño, la frecuencia y el uso del reloj. Además, es crucial

automatizar el mapeo de pines tras la extracción de parásitos, así como revisar el archivo *CMD* que controla este proceso. Esto no solo aceleraría las pruebas finales, sino que también garantizaría que el flujo sea más eficiente y confiable.

Abstract

This work focuses on improving the physical synthesis flow of integrated circuits, building on foundations established in previous years. The aim was not only to further automate the existing flow but also to address errors that arise when running synthesis scripts.

As this is an intergenerational project, the goal was to replicate and understand previous advancements, becoming familiar with tools such as IC Compiler II, IC Validator, VCS, and WaveView. From this, areas for improvement were identified, such as generating a Verilog file with instances of logical and physical components, defining variables for repeated values, and eliminating fixed-size calculations previously hardcoded. After implementing these changes, physical syntheses of various circuits were performed, starting with simple combinational circuits and progressing to more complex sequential circuits, including the synthesis of the "El Gran Jaguar" circuit.

After adjusting the metal insertion runset, DRC checks confirmed that most circuits met minimum manufacturing requirements on all metal layers. The only exception was "El Gran Jaguar," which presented an error in the Metal 2 layer, although it was very close to meeting the specifications. This issue is expected to be resolved with the implementation of 65 nm technology, as TSMC decided to discontinue 180 nm technology. On the other hand, antenna checks showed satisfactory results for all circuits.

The Verilog files generated after synthesis were also analyzed and found not to fully represent the physical layout of the design. Additionally, all stages of the flow were thoroughly documented, including explanatory videos on the use of tools and the creation of NDM libraries. This effort is intended to simplify the learning process and support future generations in adopting new technologies. Finally, functional tests of the core of some circuits were performed using HSPICE and WaveView, thus validating their correct operation.

Looking ahead, it is recommended to work on a more automated design flow capable of adjusting circuit dimensions based on specific features such as the number of ports, size, frequency, and clock usage. Additionally, automating pin mapping after parasitic extraction and reviewing the CMD file that controls this process are essential steps. These improvements would not only accelerate final testing but also enhance the efficiency and reliability of the design flow.

CAPÍTULO 1

Introducción

La nanoelectrónica ha surgido como una solución que permite lograr altos niveles de rendimiento en espacios cada vez más reducidos, y su impacto es tan significativo que es difícil imaginar el mundo moderno sin ella. Este campo se ha convertido en una de las industrias más influyentes en los países más desarrollados. Sin embargo, en Guatemala, los estudios y desarrollos en esta área aún son limitados. La Universidad del Valle de Guatemala ha realizado esfuerzos significativos para impulsar esta tecnología en el país, y el presente trabajo se alinea con esta iniciativa.

Este proyecto profundiza en la investigación para optimizar y automatizar el flujo de síntesis física y explora opciones para verificaciones finales utilizando librerías de 180 nm de TSMC. Además, se ofrece una documentación detallada de cada etapa del flujo de síntesis física, facilitando así su replicación en futuras generaciones, sin importar la tecnología o el circuito en cuestión. Para asegurar una mejor comprensión, el documento se ha estructurado en capítulos que explican de manera detallada cada etapa del proceso.

El Capítulo 7, titulado *Proceso de síntesis física*, describe detalladamente los tres *scripts* desarrollados para la síntesis física en iteraciones pasadas, así como las mejoras implementadas en cada uno de ellos. Además, se presenta la jerarquía de directorios utilizada para estructurar y optimizar el flujo de diseño.

En el Capítulo 8, *Síntesis física y verificaciones físicas*, se detalla la síntesis física aplicada tanto a circuitos combinacionales como secuenciales, destacando los resultados obtenidos tras las verificaciones *Design Rule Check* (*DRC*). Estas verificaciones se realizaron luego de modificar el *runset* de relleno de metal, abordando los desafíos relacionados con la densidad de las capas metálicas.

El Capítulo 9, *Verificación de funcionamiento por medio de Verilog*, documenta el uso de *testbenches* para evaluar los archivos Verilog generados al final del proceso de síntesis física. Este capítulo detalla cómo se validó el comportamiento lógico de los circuito.

En el Capítulo 10, titulado *Creación de librerías con IC Compiler II Library Manager*

Tool, se describe a profundidad el proceso de generación de las librerías *NDM*, esenciales para el diseño físico. Finalmente, el Capítulo 11, *Pruebas finales de funcionamiento en HSPICE*, detalla el mapeo de pines de un *deck* con elementos parásitos y la simulación del núcleo de varios circuitos.

CAPÍTULO 2

Antecedentes

La Universidad del Valle de Guatemala se ha destacado por estar a la vanguardia en tecnología y fomentar la investigación y desarrollo de soluciones innovadoras. Un claro ejemplo radica en los distintos proyectos en los que la universidad se ha desempeñado, específicamente en los departamentos de Ingeniería Electrónica, Mecatrónica y Biomédica.

El MSc. Carlos Esquit, actual director de carrera, ha modificado el mapa curricular desde el año en el que inició su labor (2009), con el propósito de modernizar y aplicar todos los conocimientos que adquirió en el extranjero, incluyendo la tecnología nanométrica. En el año 2013, se inició impartiendo el curso de "Introducción a Diseño de Sistemas VLSI" (*Very Large Scale Integration*), donde se empezaron a desarrollar diseños básicos de circuitos a micro y nano escala. Un año después, se forma un acuerdo académico con la empresa Synopsys, líder en el desarrollo de software para el mercado de semiconductores. Esta alianza permitió que los estudiantes hicieran uso de las herramientas de simulación que proveen, lo cual abrió el paso para la creación de diseños más complejos. Ese mismo año se llevó a cabo el primer trabajo de graduación basado en el área de VLSI, en el cual el Ing. Jonathan de los Santos realizó el primer diseño en silicio de un sumador/restador de 32 bits con tecnología de 28 nm [1].

En 2015 hubo una readecuación en el pénsum y se introdujeron los cursos de Nanoelectrónica 1 y 2, de manera que se pudiera profundizar en este campo. El proyecto "El Gran Jaguar" inició en 2019, en el que dos estudiantes desarrollaron trabajos de graduación, con el objetivo de definir el flujo de diseño para la fabricación de un circuito integrado (CI) con tecnología de 180 nm. El trabajo de graduación de Steven Rubio propone un esquema detallado para el diseño de un CI [2]. Asimismo, el estudiante Luis Nájera expone el diseño de un CI de una máquina de estados finitos que imprima un mensaje en codificación ASCII [3]. En conjunto se llevó a cabo el flujo implementando librerías de TSMC (*Taiwan Semiconductor Manufacturing Company*), empresa que domina el mercado de fabricación de chips a nivel mundial. Cabe destacar que se llegó a un acuerdo con Interuniversity Microelectronics Centre (IMEC) para la fabricación del chip, organización que tiene un convenio de fabricación con TSMC.

El grupo de trabajo para el proyecto aumentó en el 2020, etapa en la que documentaron el proceso de la realización de verificaciones físicas; Design Rule Check (DRC) [4], Electrical Rule Check (ERC), Antenna Rule Check [5] y Layout Versus Schematic (LVS) [6]. Las simulaciones se llevaron a cabo por medio de los programas IC Validator y IC Compiler I. Asimismo, se inició la automatización del proceso de comprobación del funcionamiento de archivos HDL por medio de la herramienta VCS, en el trabajo de Jefferson Ruano [7]. En el año 2021 se tomaron en cuenta los hallazgos encontrados en relación con la existencia de la herramienta IC Compiler 2 y se planteó la migración del flujo de síntesis física a esta. Los ingenieros José Ayala [8], José Ruiz [9], Antonio Altuna [10] y Julio Shin [11] se encargaron de llevar a cabo las verificaciones mencionadas anteriormente. Por medio de estas simulaciones se logró que el *layout* no presentara errores en la prueba ERC y de Antena. Sin embargo, la verificación DRC presentó 6 errores de densidad. En el trabajo de Luis Abadía se empleó IC Compiler para realizar el posicionamiento e interconexión de componentes por medio de una serie de comandos [12]. Asimismo, el Ing. Elmer Torres llevó a cabo la etapa de síntesis lógica por medio de comandos proporcionados por la Ing. Sophia Cardona [13] [14].

En el 2022, un grupo amplio de estudiantes trabajaron en replicar las etapas avanzadas en los trabajos de graduación en años anteriores. Esto incluyó la automatización de la etapa de síntesis lógica [15] y síntesis física [16], donde también se redujeron los errores de densidad de 6 a 2; la documentación de alternativas para el flujo de diseño [17]; así como la automatización de la fase de verificaciones físicas, que incluye DRC, ERC, LVS y Verificación de Antena [18]. Cabe destacar que en el trabajo de graduación de Paola Mendizábal, se automatizó el proceso de descarga e instalación del software de Synopsys empleado para trabajar en el diseño del circuito integrado, lo cual permitió agilizar dicho proceso para generaciones posteriores [19].

En 2023, Noel Prado realizó mejoras al flujo de diseño de circuitos integrados, al simplificar la jerarquía de directorios que se utiliza para ambos la síntesis lógica y física. Asimismo, se replicó el flujo de diseño con las mejoras implementadas, empleando la librerías educativas de diseño de *Synopsys*. Además, se trabajó en la corrección de los errores de densidad restantes, en los que la densidad de metal en distintas capas del circuito no llegan al requisito mínimo de fabricación. Estos errores no se resolvieron por completo, pero sí se redujo el porcentaje de error por medio de un *runset* para realizar el relleno de metal proveído por TSMC [20].

CAPÍTULO 3

Justificación

La electrónica, desde sus inicios, ha sido el motor que impulsa la revolución tecnológica. Desde calculadoras hasta sistemas de comunicación complejos, esta tiene un impacto significativo en cada aspecto de nuestra vida diaria. En el núcleo de la electrónica se encuentra una invención que cambió el curso de la historia; el transistor. Inventado en 1947, no solo reemplaza los frágiles tubos al vacío, sino también sienta las bases para una era de electrónica más compacta y eficiente. Desde entonces se ha llevado a cabo una búsqueda de alternativas para mejorar su rendimiento, en donde hacerlo cada vez más pequeño ha sido la respuesta. De esta forma, se reduce el costo y el consumo de energía y se mejora el rendimiento de un *chip*, al aumentar la densidad de componentes dentro de este en espacios reducidos.

Avances en la tecnología de fabricación y diseño de circuitos integrados han permitido que el tamaño de los transistores llegue a escala nanométrica. Los circuitos integrados a escala nanométrica, también conocidos como *nanochips*, se componen de millones de transistores. Desde su invención, estos también se han vuelto cada vez más pequeños y eficientes y son componentes clave en una amplia gama de dispositivos electrónicos como computadoras y teléfonos inteligentes, implantes médicos y sistemas de control en aviones y automóviles. Sin duda, la capacidad de diseñar y fabricar *chips* es y seguirá siendo fundamental para el avance continuo de la humanidad.

En la actualidad, varias universidades del primer mundo han priorizado la implementación de cursos de nanoelectrónica en sus programas académicos. La Universidad del Valle de Guatemala es la única universidad en Centroamérica que ofrece estos cursos, estableciéndose así como pionera en la región en este campo de estudio. A lo largo de cinco años se ha trabajado en el proyecto "El Gran Jaguar", el cual tiene como objetivo desarrollar el primer *nanochip* diseñado en la historia de Guatemala y, por ende, realizar lo más complejo que el ser humano puede hacer. Este proyecto abre la puerta a que crezca el mercado laboral en Guatemala e incluso a la posibilidad de traer la industria de los semiconductores acá y aumentar nuestra autonomía tecnológica.

Por esto, se busca replicar cada paso del flujo de diseño llevado a cabo por estudiantes de años anteriores y, de esta manera, comprender a fondo las etapas en las que consiste

el flujo: síntesis lógica, síntesis física y verificaciones físicas. Esto permite adquirir conocimientos avanzados y dominar el flujo completo, con el objetivo de posteriormente generar documentación que transmita todo el aprendizaje obtenido a través de los años. La documentación proporciona un registro detallado y estructurado de todas las etapas del proceso, desde la concepción inicial hasta la implementación final del diseño. Este registro sirve como una guía para el equipo actual y futuras iteraciones del proyecto, facilitando la comprensión y la replicabilidad del flujo de diseño y garantizando la continuidad del conocimiento en nanoelectrónica y la posibilidad de mejorar y perfeccionar el flujo en el futuro.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Generar la documentación detallada y verificaciones finales del proyecto de diseño y fabricación de un circuito integrado con tecnología de 180 nm, para que sea replicable por futuras generaciones de estudiantes de la UVG y aplique para cualquier proyecto.

4.2. Objetivos específicos

- Replicar los trabajos de graduación de generaciones anteriores para aprender y dominar el proceso de diseño y pruebas de funcionamiento del *nanochip*, incluyendo la síntesis lógica, síntesis física, verificaciones físicas y extracción de parásitos.
- Realizar la síntesis física y las verificaciones físicas de Antena y DRC del *nanochip* "El Gran Jaguar"
- Recopilar, ordenar y sintetizar todo el conocimiento adquirido hasta el momento y documentar la fase de síntesis física para que pueda servir de referencia en futuros trabajos.
- Resolver errores de densidad restantes por medio de investigación y comunicación con TSMC, por medio de la organización IMEC.
- Realizar verificaciones de cumplimiento de los requerimientos de TSMC y establecer comunicación constante con la organización IMEC para darle seguimiento al proceso de envío y fabricación.
- Realizar las verificaciones finales de funcionamiento del proyecto "El Gran Jaguar", posterior a la extracción de parásitos.

CAPÍTULO 5

Alcance

El alcance de este trabajo consiste en llevar a cabo la síntesis física y verificaciones físicas (DRC y Antena) de cualquier *nanochip*, con base especial en el proyecto "El Gran Jaguar", así como la adaptación del flujo de diseño existente, que inicialmente utilizaba tecnología de 180 nm, a la nueva tecnología de 65 nm de TSMC. Este trabajo busca la resolución de errores que aparezcan luego de generar las verificaciones pertinentes y, de esta manera, realizar las pruebas finales de funcionamiento del circuito, por medio de las herramientas *HSPICE* y *WaveView*, de un *deck* simulable generado en la etapa de extracción de parásitos. Estas pruebas representan uno de los pasos finales del proceso, condicionadas por la verificación exitosa del cumplimiento de todos los requisitos establecidos por TSMC, antes de proceder con el envío del diseño para su fabricación.

Finalmente, se elaborarán manuales detallados y videos instructivos en donde se documentará la fase de síntesis física, verificaciones físicas y pruebas de funcionamiento. Estos recursos estarán diseñados para ser utilizados por futuras generaciones de estudiantes de la universidad.

El proyecto está programado para desarrollarse desde enero hasta noviembre de 2024. Es importante considerar que su avance podría verse afectado por posibles demoras en la recepción de información y recursos por parte de IMEC. Además, el proceso de recepción del *nanochip* fabricado no está incluido dentro de los alcances de este proyecto, ya que se espera que ocurra fuera del tiempo disponible para este trabajo. Por lo tanto, el enfoque se centrará en la entrega del diseño listo para fabricación y en la documentación asociada.

CAPÍTULO 6

Marco teórico

6.1. Diseño VLSI

El proceso *very-large-scale integration* (VLSI) consiste en la creación de un circuito integrado (IC) por medio de la combinación de millones o incluso cientos de millones de transistores en un solo chip. VLSI permite añadir componentes. El flujo de diseño VLSI se compone de pasos secuenciales en los que se diseña, se llevan a cabo verificaciones y se procede con la fabricación de ICs que llegan a tener una amplia variedad de aplicaciones en los equipos electrónicos más avanzados de la actualidad [21].

En la Figura 1, se puede observar el ciclo de diseño VLSI, en el que se inicia definiendo las especificaciones del sistema; los factores y requerimientos que se deben considerar antes de empezar la construcción de un diseño, y así entender la funcionalidad, rendimiento, límites de potencia, etc. El diseño de esquemático determina la interconexión del IC, mientras que en el diseño funcional y lógico se esquematiza el IC en un nivel de bloques funcionales por medio de algún lenguaje descriptor de hardware (HDL) como Verilog o VHDL (RTL o *Register-transfer level*). Antes de proceder con el resto del proceso se llevan a cabo verificaciones funcionales para asegurarse que el diseño RTL se alinea con las especificaciones establecidas, lo cual se hace empleando software de simulación que realiza pruebas del circuito bajo distintas circunstancias [22].

Figura 1: Flujo de diseño VLSI para un circuito integrado

Luego de realizar estas pruebas ya se procede con el diseño del circuito, en el que se plasma la lógica empleando componentes físicos. En el diseño físico, la *netlist* realizada anteriormente se transforma a una representación geométrica, es decir, el *layout*. Posteriormente se llevan a cabo las verificaciones físicas pertinentes para garantizar la funcionalidad y el cumplimiento de las especificaciones. Finalmente, la representación gráfica del diseño se envía a la fundición y al regresar pasa por un proceso riguroso de pruebas de funcionamiento para verificar que su funcionamiento sea el esperado [22].

6.2. *Frontend*

El *frontend* es la etapa de diseño en la que se inicia definiendo los objetivos principales y requerimientos del sistema a elaborar. Esto abarca funcionalidad, rendimiento, dimensiones físicas y tecnología de producción. Algunos de los pasos más importantes incluye:

1. Diseño funcional y lógico: en el diseño funcional se define la conectividad y funcionalidad de cada módulo, enfocándose en el comportamiento a alto nivel, es decir, su conjunto de entradas, salidas y funcionamiento en términos de tiempo. Ahora bien, el diseño lógico se lleva a cabo en RTL (*register-transfer level*) por medio de un lenguaje HDL como Verilog o VHDL, ya sea de manera *behavioral* o *structural*.
2. Síntesis lógica: por medio de herramientas como *Design Vision* se traduce el circuito elaborado en verilog a una *netlist*, una representación textual que describe la conectividad entre los diferentes componentes, a nivel de compuertas lógicas.
3. *Netlist* / diseño del circuito: luego de finalizar la síntesis, el *netlist* obtenido genera una descripción más detallada del sistema ya que no solo describe su comportamiento si no que también su estructura. Este se obtiene de forma *structural*, empleando componentes de las librerías de la tecnología y fabricante con la que se trabajará.
4. Verificaciones: en la fase final del *frontend* se debe verificar que el sistema descrito en el *netlist* sea equivalente al que se describe por el RTL, lo cual se puede realizar por medio de las herramientas *VCS* o *Formality*. [7]

6.3. *Backend*

Esta etapa se enfoca en transformar la descripción funcional de alto nivel del diseño en una implementación física del circuito integrado. Este incluye varios pasos que contribuyen a la creación del *layout* final del chip, el cual luego puede ser enviado para su fabricación. En el proceso de diseño físico de un chip, se lleva a cabo la instanciación de todos los componentes del diseño con sus representaciones geométricas. Esto implica asignar ubicaciones espaciales a celdas, compuertas, transistores, etc., con formas y tamaños fijos por capa de fabricación. Este proceso, conocido como colocación o *placement*, se complementa con la conexión de rutas apropiadas, denominado enrutamiento o *routing*, realizado en capas metálicas en la superficie del chip [23]. El resultado es un conjunto de especificaciones de fabricación que deben ser posteriormente verificadas. El diseño físico se ejecuta siguiendo reglas de diseño que reflejan las limitaciones físicas del proceso de fabricación de una tecnología específica. Por ejemplo, se establecen distancias y anchos mínimos para los cables. Es importante destacar que el diseño físico debe ser adaptado a cada nueva tecnología de fabricación [10].

En la síntesis física se inicia con la creación del *layout* por medio del *netlist* obtenido en la etapa anterior. Debido a la complejidad del diseño físico, este se puede dividir en una serie de pasos (véase en la Figura 2):

1. *Partitioning*: descompone un circuito en subcircuitos o módulos más pequeños, que pueden diseñarse o analizarse individualmente.

2. *Floorplanning*: define las formas y la disposición espacial de los módulos. Además, determina la ubicación estratégica de puertos externos y macrobloques.

3. *Power and ground routing*: distribuye los nodos de alimentación (VDD) y tierra (GND) a lo largo de todo el chip.

4. *Placement*: encuentra ubicaciones espaciales óptimas para todas las celdas dentro de cada bloque y las coloca.

5. *Clock network synthesis*: determina el enrutamiento y gestión de energía de la señal de reloj y asegura que se cumplan los requisitos de desviación y retardo.

6. *Routing*: inicia asignando recursos para conexiones a nivel global mediante el uso de celdas globales, estableciendo rutas generales de conexión. Finaliza asignando rutas específicas a capas metálicas, detallando la conexión entre celdas por medio de *nets*, y asegurando una asignación eficiente de recursos. Cabe mencionar que la herramienta empleada para llevar a cabo el *floorplan*, *placement* y *routing* es IC Compiler 2 [23].

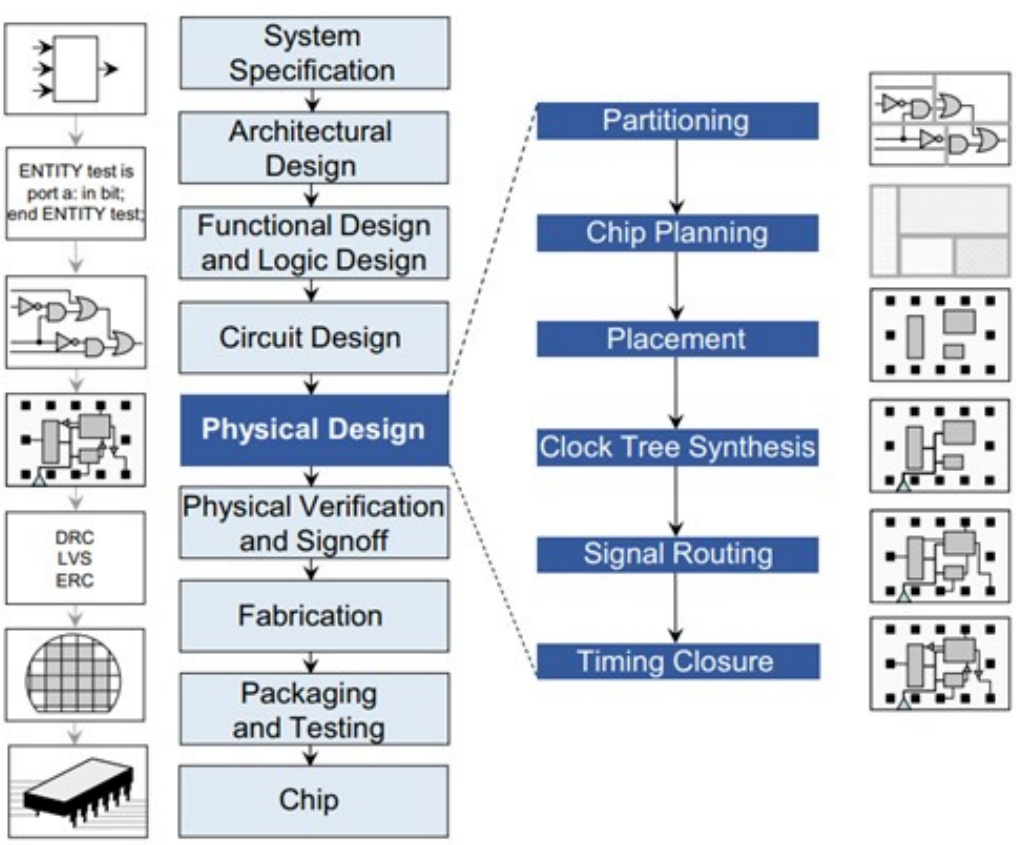


Figura 2: Pasos en el flujo de diseño de circuitos VLSI [23]

6.3.1. *Floorplanning*

La etapa de *floorplanning* es el proceso de organizar y definir la distribución física de los componentes de un chip durante la etapa de diseño. Este paso ocurre luego de que el

diseño ha sido importado a la herramienta de implementación de diseño físico y consiste en determinar la colocación óptima de elementos como celdas estándar, macros y pads de I/O dentro del *die*. Su objetivo principal es establecer el tamaño del *die*, asignar espacios para los diferentes bloques, planificar la red de poder y asegurarse que la disposición física cumple con las restricciones de área máxima, temporización y potencia [24].

Este diseño puede llevarse a cabo de forma manual, pero generalmente se realiza de forma automática por medio del software. En la Figura 3 se puede ver ejemplificado este proceso. Los algoritmos utilizados minimizan la longitud de los cables necesarios para las interconexiones, lo cual es crucial para optimizar el rendimiento del chip. Antes de iniciar con este proceso, se recomienda separar las celdas en áreas según su necesidad de comunicarse entre sí, lo cual permite reducir la longitud de los cables [24]. Luego que todas las celdas están organizadas, estas permanecen fijas en sus posiciones dentro del *layout*.

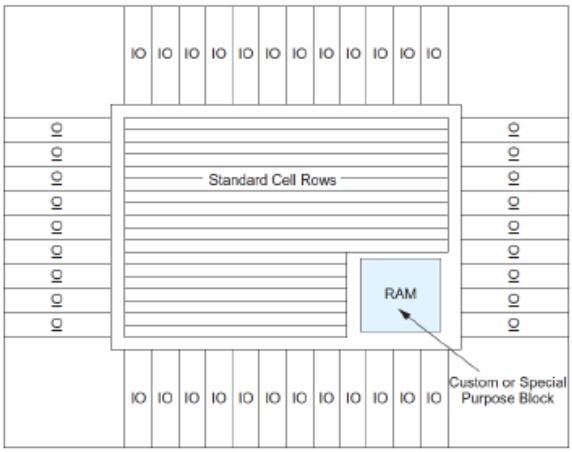


Figura 3: *Floorpaning* en la etapa de síntesis física [24]

6.3.2. *Routing*

El enrutamiento es el proceso de interconectar las celdas que ya han sido posicionadas dentro del *layout*, siguiendo las rutas descritas dentro del *netlist* generado en la etapa de síntesis lógica. Este proceso se lleva a cabo de manera automática mediante el software, el cual utiliza una serie de criterios como la geometría del *layout* y los costos de cada conexión. Cabe destacar que el enrutamiento se realiza en varias etapas, cada una con un propósito específico. El enrutamiento de poder y tierra (*PG Route*) se realiza durante la etapa de *floorplanning*, mientras que el enrutamiento secundario de poder y tierra ocurre después de la colocación de las celdas estándar. El enrutamiento del reloj (*Clock Route*) se lleva a cabo durante la síntesis del árbol de reloj, después de la inserción de los búferes de reloj. Finalmente, el enrutamiento de señales (*Signal Route*) se realiza después de que todas las celdas han sido colocadas, en la etapa de enrutamiento [25].

El objetivo principal del enrutamiento es completar todas las interconexiones de señales con un mínimo de violaciones de reglas de diseño (DRC) y optimizar la lógica del camino de datos para mejorar la temporización, el cumplimiento de DRC y el consumo de potencia [23]. La Figura 4 muestra un ejemplo de un chip luego de la etapa de enrutamiento, en donde

cada línea representa un cable que conecta dos nodos dentro del circuito. Las diferentes capas de metal están representadas por distintos colores, lo que facilita la visualización de cómo se organizan las rutas. Realizar estas conexiones manualmente sería extremadamente laborioso y consumiría una gran cantidad de tiempo y recursos humanos, lo que justifica la necesidad de automatización.

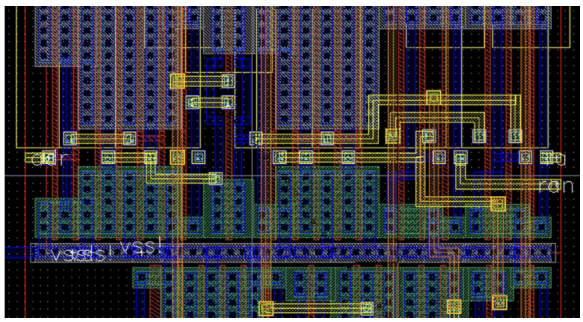


Figura 4: *Routing* en la etapa de síntesis física [26]

6.4. Verificaciones físicas

Luego de haber concluído con el diseño físico del circuito, se debe de verificar y garantizar el correcto funcionamiento de su parte eléctrica y lógica. Algunos problemas encontrados durante la verificación física pueden ser tolerados si su impacto en el rendimiento del chip será insignificante durante la fabricación. En otros casos, el *layout* debe ser modificado, pero estos cambios deben ser mínimos y se debe tener cuidado de no introducir nuevos errores. Para llevar a cabo esta etapa, se llevan a cabo las siguientes verificaciones (por medio de la herramienta IC Validator):

1. *Design Rule Check* (DRC): verifica que el diseño cumpla con todas las restricciones y reglas de diseño impuestas por la tecnología. También verifica la densidad de capa para el pulido químico-mecánico (CMP) [4].

2. *Layout vs. Schematic* (LVS): verifica la funcionalidad del diseño. Para ello, se utiliza el diseño para derivar un *netlist*, que se compara con el *netlist* original producido a partir de la síntesis lógica [6].

3. *Parasitic extraction*: deriva los parámetros eléctricos de los elementos del diseño a partir de sus representaciones geométricas. El *netlist* resultante se utiliza para verificar las características eléctricas del circuito.

4. *Antenna rule checking*: busca prevenir los efectos de antena ya que pueden dañar los gates de los transistores durante la fabricación, a través de la acumulación de carga excesiva en cables metálicos que no están conectados a nodos de unión PN [5].

5. *Electrical rule checking* (ERC): verifica las conexiones de alimentación y tierra, y que los tiempos de transición de señal y las cargas capacitivas estén adecuadamente limitados [5].

6.4.1. DRC

Es una etapa del flujo de diseño en donde se verifica si el diseño físico del chip cumple con las reglas y restricciones impuestas por el *foundry* (fabricante de semiconductores, como TSMC), por medio de un *runset*. El DRC asegura que el diseño físico sea manufacturable y que se cumplan los estándares de calidad [27]. Una de las reglas incluye ancho mínimo, la cual establece el grosor mínimo para los elementos de una capa del diseño para evitar su colapso o que el circuito se abra debido a falta de material. Además, el espaciado mínimo intracapa es otra regla crucial que determina la distancia mínima entre los elementos de una misma capa. Esta separación es necesaria para prevenir cortocircuitos no deseados y asegurar que las distintas partes del circuito estén correctamente aisladas. De manera similar, se aplica un espaciado mínimo intercapa entre elementos de diferentes capas. Este espaciado es vital para evitar problemas como la formación de capacitancias parásitas o corrientes de fuga, las cuales podrían degradar el rendimiento del circuito [28].

Otra regla importante es la del cercado mínimo, que se refiere a la restricción aplicada a estructuras que atraviesan varias capas, como las vías o contactos. Para estas estructuras, se define un tamaño mínimo que garantiza un buen contacto entre las capas, incluso en caso de imperfecciones durante la fabricación. Finalmente, las reglas de densidad verifican que la distribución de los elementos en el *layout* cumpla con criterios específicos de ocupación [28]. En caso de que el área designada no esté suficientemente ocupada, puede ser necesario añadir elementos *dummy*, los cuales no tienen una función en el diseño.

6.4.2. Verificación de antena

La verificación de antena es un proceso que busca prevenir problemas relacionados con la acumulación de carga electromagnética durante la fabricación. Estos problemas, conocidos como efectos de antena, pueden surgir cuando los metales conectados a las compuertas de los transistores acumulan suficiente carga durante procesos como el grabado por plasma o la implantación de iones. La acumulación de esta carga puede generar descargas que dañan la capa de óxido de las compuertas, lo cual deteriora el rendimiento y la vida útil de los transistores al incrementar la corriente de fuga y alterar el voltaje umbral [24].

Para evitar estos efectos, se emplean reglas específicas que limitan la cantidad de metal que puede estar conectada a una compuerta sin una fuente o drenaje que actúe como elemento de descarga. En la práctica, se adoptan soluciones como interrumpir la continuidad del metal mediante conexiones a otras capas, reduciendo así el área acumulada, o la incorporación de diodos que descargan la acumulación de carga a tierra [24]. El proceso de esta verificación es similar al DRC en cuanto a su implementación por medio de un *runset* proveído por el *foundry*.

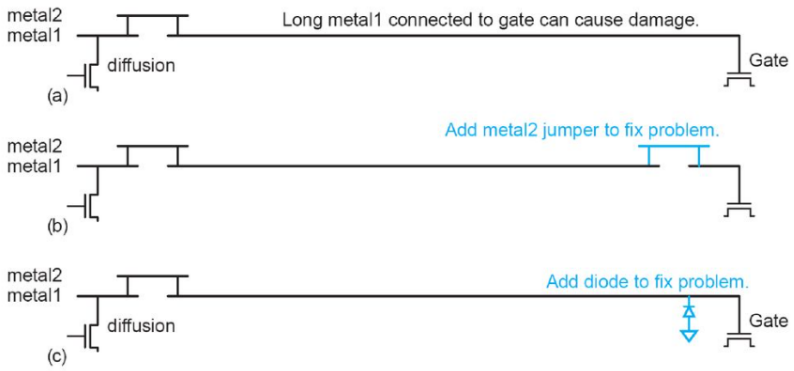


Figura 5: Soluciones para problemas relacionados con el efecto de antena [24]

6.5. *Synopsys*

Synopsys Inc. es una compañía líder en el desarrollo de software para el diseño de semiconductores. Sus productos permiten que se desarrollen y prueben diseños de IC antes de el envío a producción, lo cual acelera el proceso de innovación en la electrónica. A continuación, se detallan las herramientas de Synopsys pertinentes para el desarrollo del proyecto "El Gran Jaguar".

6.6. VCS

Esta herramienta es un simulador de código compilado, es decir, evalúa cada elemento lógico en cada "paso de tiempo", independientemente de si existe un cambio en la entrada o no. La esencia de un simulador de código compilado es que la estructura del sistema que se está simulando se refleja en la memoria de la computadora y en que cada elemento lógico tiene su propio código. VCS permite estudiar, compilar y simular lenguajes descriptores de hardware como Verilog, VHDL, SystemVerilog, entre otros [29]. El flujo de simulación empleado consta de dos pasos:

1. Compilación: se compila el archivo verilog, con su respectivo *testbench*, y se genera un archivo binario ejecutable.
2. Simulación del diseño: el archivo generado en el paso anterior (*simv*) se ejecuta y se procede a correr la simulación [29].

6.6.1. Compilación

En este primer paso, se puede compilar el diseño ya sea en modo optimizado o en modo de depuración (interactivo), aunque Synopsys recomienda que se emplee el segundo modo hasta que se obtengan comportamientos esperados.

6.6.1.1. Comandos

En la terminal se debe de cambiar al directorio en donde los archivos y sus respectivos testbench se encuentran:

```
cd <path>
```

Luego se compilan los archivos escribiendo:

```
vcs -v file_name.v file_name_tb.v -o name_simulation -full64 -debug_access+all
```

6.6.2. Simulación

El archivo binario ejecutable generado por VCS (simv) se usa para correr la simulación.

6.6.2.1. Comandos

Para correr la simulación se escribe en la terminal:

```
./simv
```

Luego de que termine este proceso, un archivo llamado *fileName.vcd* aparece en el mismo folder donde se encuentran los demás archivos, en el cual se podrán desplegar los resultados de manera gráfica, empleando el comando:

```
dve -full64
```

Este comando abre una interfaz, como la que se observa en en la Figura 6 que se podrá seleccionar el archivo .vcd y una por una ir agregando las señales que se desean observar.

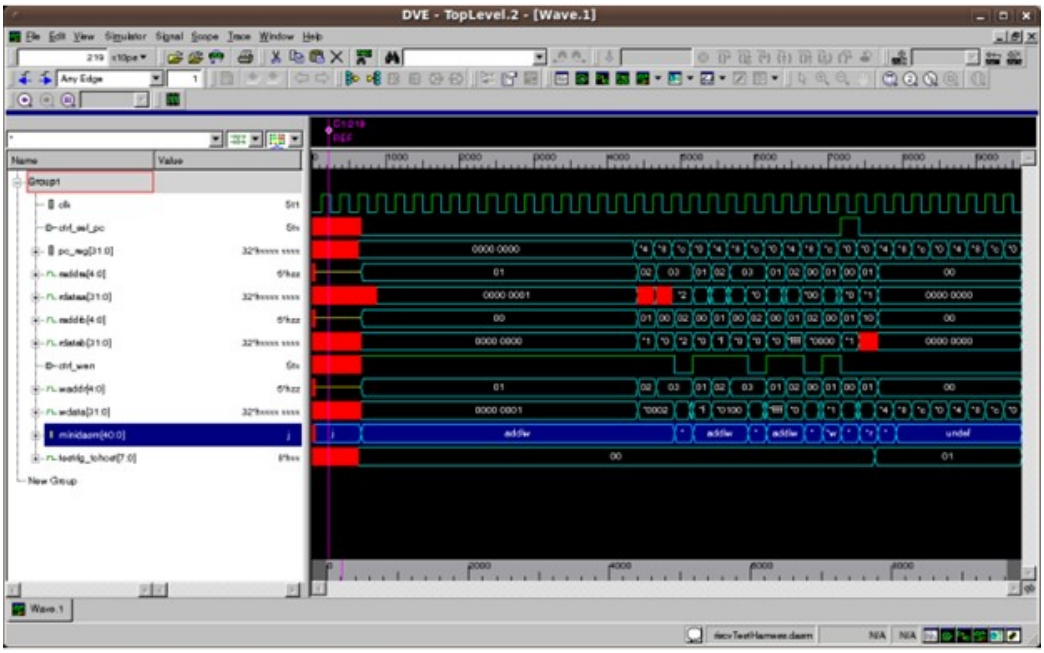


Figura 6: Interfaz de la herramienta VCS [7]

6.7. *IC Compiler II*

IC Compiler II es un sistema completo de implementación de *netlist* a GDSII que abarca desde la exploración y prototipado tempranos del diseño, la planificación detallada del diseño, la implementación de bloques, hasta el ensamblaje del chip. La herramienta toma como entrada una *netlist* de Verilog, restricciones de tiempo y bibliotecas lógicas y físicas, y como resultado genera un archivo en formato GDS con el *layout* del diseño [30].

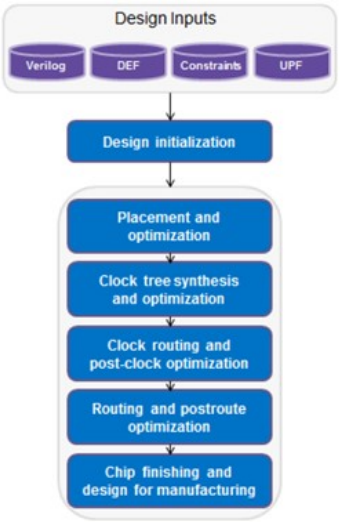


Figura 7: Flujo *Place and Route* de IC Compiler II [30]

En la Figura 7 se puede observar el flujo que se lleva a cabo por medio de la herramienta para la realización de la síntesis física de un circuito integrado.

En el primer paso, *Design Initialization*, algunas de las tareas incluye:

1. Crear o abrir la librería asociada con el diseño.
2. Leer los archivos *netlist* usando el comando *read_ verilog*.
3. Importar la información de *floorplan*, con las restricciones físicas.
4. Configurar restricciones de tiempo.
5. Especificar restricciones físicas adicionales.
6. Controlar la optimización de celdas, *nets*, pines y puertos.
7. Generar el *power grid* [30]

Para *placement and optimization*, usando el comando *place_ opt*, se puede llevar a cabo el posicionamiento, optimización física y legalización y tiene las siguientes etapas:

1. Posicionamiento inicial
2. Corrección inicial de violaciones DRC
3. Optimización inicial relacionada con *timing*, área física, congestión y potencia de fuga.
4. Posicionamiento final para mejorar *timing* y congestión y legaliza el diseño, es decir, asegurarse que cumpla con todas las reglas de diseño establecidas por la herramienta.
5. Optimización final [30]

En *Clock tree synthesis and optimization*, se usa el comando *clock_ opt* para llevar a cabo la síntesis y optimización del árbol de relojes. Esto permite realizar de manera conjunta la síntesis de árboles de reloj, el enrutamiento de las redes de reloj y optimizaciones adicionales en el diseño. Este paso garantiza que la señal de reloj se distribuirá eficientemente y que el diseño cumplirá con los requisitos de temporización y restricciones de diseño [30].

El siguiente paso, *Routing and post-route optimization*, se puede iniciar llevando a cabo el comando *check_ routability*, para chequear si el bloque se encuentra listo para la generación del ruteo. El objetivo del ruteo es conectar los pines de las celdas en el diseño de acuerdo con las reglas de diseño de enrutamiento y las restricciones de temporización, congestión y multi-voltaje. El objetivo de la fase de optimización posruteo es afinar el diseño enrutado de acuerdo con las reglas de diseño y restricciones mencionadas anteriormente [30]. La herramienta puede llevar a cabo dos tipos de optimización:

1. Se mejora la temporización, uso de área y consumo de energía, y corrige violaciones de DRC lógicas. Este se lleva a cabo empleando el comando *route_ opt*.

2. De ruteabilidad: Aumenta el espacio entre las celdas para corregir violaciones de DRC de enrutamiento causadas por problemas de acceso a pines. El comando utilizado para este tipo de optimización es *optimize_routability*.

Cabe mencionar que si se llevaran a cabo ambas, la herramienta recomienda que se ejecute primero la optimización lógica y luego la de ruteabilidad.

De esta manera, IC Compiler II genera las siguientes entradas:

- Datos de la librería: información sobre las características y comportamientos de los elementos de diseño.
- Base de datos del diseño: contiene la representación digital detallada del diseño en proceso.
- *Scripts* de flujo: definen y controlan el flujo de trabajo de la herramienta durante el proceso de compilación [30].

Así como los siguientes resultados esperados:

- Datos de la librería actualizados
- Base de datos del diseño actualizada
- Archivos log: estos registran eventos, mensajes y resultados del proceso de compilación.
- Archivos de reporte: proporcionan información detallada sobre diversos aspectos del diseño [30].

En la última fase, *Chip finishing*, el objetivo es incrementar la fiabilidad del chip mediante la implementación de distintas técnicas:

- Inserción de capacitores de desacoplamiento: mejora la estabilidad del suministro de energía y reduce el ruido en el sistema.
- Inserción de celdas de relleno: optimiza la distribución de celdas en áreas del diseño donde pueda haber exceso de espacio o irregularidades.
- Inserción de relleno metálico: contribuye a una distribución uniforme de material durante la fabricación.
- Análisis de electromigración: evalúa la migración de átomos de metal bajo el efecto de la corriente, lo cual puede llegar a afectar la integridad de las conexiones metálicas [30].

6.8. *IC Validator*

IC Validator es una herramienta de Synopsys empleada para la etapa de verificaciones físicas y *Sign-off*. Sus funciones básicas incluyen realizar DRC, ERC, LVS, *Antenna rule checking* y *Metal fill*; en el que se incluye formas en las capas para lograr la densidad deseada de polígonos, asegurando la uniformidad de las capas [31].

Esta herramientas se puede correr dentro de IC Compiler II, por medio de un flujo llamado *Design physical verification*, en el que se lleva a cabo la verificación de *layouts* generados por la herramienta.

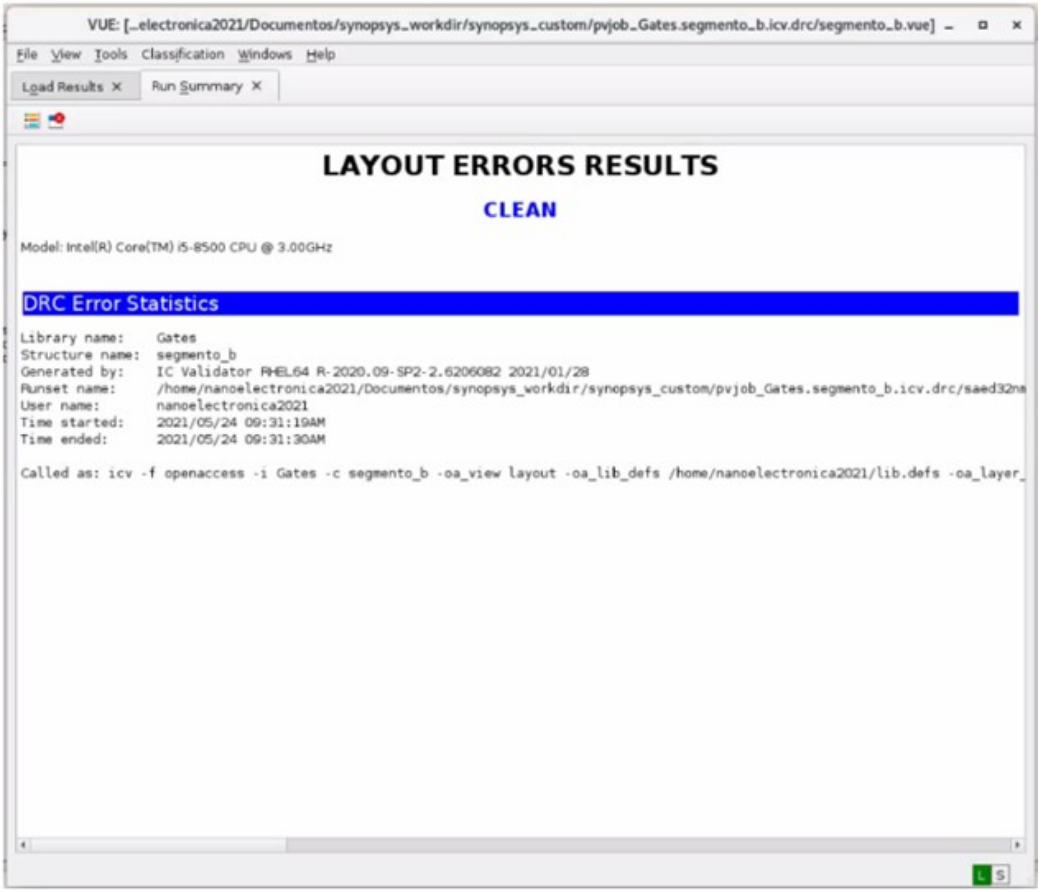


Figura 8: Resultados obtenidos al correr ICV

En la Figura 9 se puede observar que se debe generar una base de datos para *IC Validator* por medio del *netlist* del esquemático, un *Runset script* y una base de datos física. El resultado que se obtiene es una base de datos o el *report file* se puede usar para crear un *netlist* parásito o la extracción del diseño por medio de un archivo GDS. Cabe mencionar que la base de datos del diseño generada por ICC2 se tiene que importar a IC Validator y así poder llevar a cabo las verificaciones. ICV luego genera un reporte detallado en el que resalta cualquier violación o error encontrado durante el proceso, y proporciona información de la ubicación de estos junto con detalles específiciso de la regla de diseño que fue violada. Luego que los errores fueron identificados, se procede a resolverlos por medio de ajustes al

layout. Las verificaciones físicas son comúnmente un proceso iterativo, en el que se llevan a cabo múltiples repeticiones en las que se corre la herramienta y se corrigen los errores hasta que el diseño cumpla con todas las reglas de diseño requeridas y pase la verificación [31].

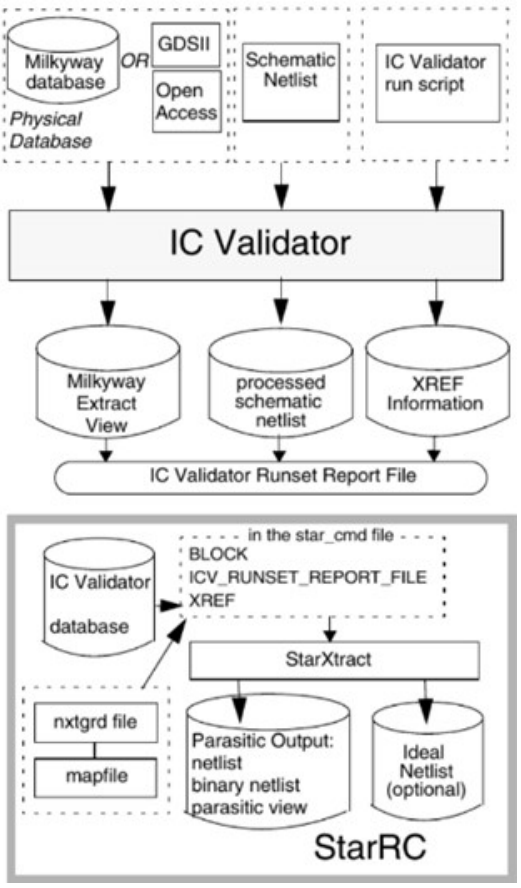


Figura 9: Flujo de diseño de ICV [31]

6.9. *StarRC*

StarRC se trata de una herramienta extractora de parásitos, la cual provee con métodos para la exploración de los contenidos de un GPD; una base de datos que contiene diseños de elementos parásitos que la herramienta extrae, tales como resistencia, capacitores e inductores [32].

Para ambos flujos a nivel compuerta y nivel transistor, a través de la interfaz de línea de comandos, se puede llevar a cabo lo siguiente:

- Crear una colección de resistencias parásitas, capacitores de tierra y de acoplamiento a partir de una o más *nets*.
- Consultar atributos de elementos parásitos como resistencia, capacitancia, nombre de subnodo, nombre de capa, número de capa y ubicación física.

- Reportar propiedades de los datos en el GPD como completitud, la versión de StarRC utilizada para realizar la extracción, la presencia o ausencia de tipos específicos de datos y el número de *nets*, celdas y puertos.
- Informar los nombres de las esquinas y las capas definidas en el GPD [32].

CAPÍTULO 7

Proceso de síntesis física

IC Compiler II es una herramienta que permite automatizar múltiples etapas del flujo de diseño, incluyendo la creación de *floorplans*, la colocación de celdas, la generación de redes de alimentación y la optimización del enrutamiento. Para la automatización del flujo y el manejo de tareas repetitivas o complejas, es común el uso de *scripts* basados en Tcl (*Tool Command Language*). La ventaja de utilizar *scripts* Tcl radica en la capacidad de ejecutar comandos secuencialmente sin intervención manual, permitiendo definir un flujo de trabajo claro que puede replicarse fácilmente.

Existen diferentes formas de ejecutar un script Tcl en IC Compiler II:

1. Mediante el uso del comando `source` desde la línea de comandos de `icc2_shell`:

   ```
   source <nombre_del_script>
   ```

2. A través de la interfaz gráfica de usuario (GUI), seleccionando la opción:

   ```
   File > Execute Script
   ```

Para la ejecución del flujo de síntesis física, se emplean tres *scripts*:

- `sintesis_fisica_top.tcl`: Este archivo inicia el flujo de la síntesis física y llama a otros *scripts* que se encuentran en la carpeta `scripts`: `setup.tcl` y `floorplan.tcl`.
- `setup.tcl`: En este *script*, se configuran las variables de entorno asociadas a todos los archivos necesarios para la síntesis física. Esta configuración simplifica las fases posteriores, ya que permite hacer referencia a los archivos de manera más clara.

- `floorplan.tcl`: En este archivo, se lleva a cabo la inicialización del *floorplan*, que incluye la creación de la red de alimentación y la colocación preliminar de las celdas estándar. Este paso establece la estructura básica del diseño sobre la que se construirá el resto del circuito.

7.1. *Script* setup.tcl

1. Se elimina la librería `LIB_TEST` si existe, garantizando que siempre se comience con una versión limpia de este archivo. Al eliminarlo, se evitan conflictos con versiones anteriores o datos corruptos que puedan interferir en los pasos siguientes del proceso de diseño.

```
    file delete -force LIB_TEST.ndm
```

Cuadro 1: Bloque 1 `setup.tcl`

2. Se guarda un archivo de *log* para tener un registro detallado de los comandos ejecutados durante la sesión dentro del directorio `Logs`.

```
    set command_log_file "./Logs/command.log"
```

Cuadro 2: Bloque 2 `setup.tcl`

3. Se define la ubicación de las librerías de la tecnología (180 nm) y de referencia del diseño.

```
    set DESIGN_REF_PATH "../../Librerias"
    set NDM_REFERENCE_LIB_DIRS  " \
        ${DESIGN_REF_PATH}/lib/ndm/TSMCWorkspace.ndm
        "
```

Cuadro 3: Bloque 3 `setup.tcl`

4. Define el archivo de tecnología, que contiene las reglas de diseño, capas metálicas, espaciado mínimo y otros parámetros.

```
    set TECH_FILE        "${DESIGN_REF_PATH}/tech/tsmc018_6lm.tf
       "
```

Cuadro 4: Bloque 4 `setup.tcl`

5. Se define el archivo de mapa, el cual contiene información sobre las capas de metal y su uso en el ruteo.

```
    set MAP_FILE         "${DESIGN_REF_PATH}/tlu_and_map/
       star.map_6M"
```

Cuadro 5: Bloque 5 `setup.tcl`

6. Define el archivo que contiene modelos de extracción de parásitos.

```
set TLUPLUS_FILE "${DESIGN_REF_PATH}/tlu_and_map/
    t018lo_1p6m_typical.tluplus"
```

Cuadro 6: Bloque 6 `setup.tcl`

7. Define las redes y puertos de voltaje y tierra como `VDD` y `VSS`, respectivamente.

```
set MW_POWER_NET                "VDD" ;#
set MW_POWER_PORT               "VDD" ;#
set MW_GROUND_NET               "VSS" ;#
set MW_GROUND_PORT              "VSS" ;#
```

Cuadro 7: Bloque 7 `setup.tcl`

8. Establece las capas mínimas (`METAL1`) y máximas (`METAL6`) que se utilizarán para el ruteo del diseño.

```
set MIN_ROUTING_LAYER            "METAL1"    ;# Min routing
    layer
set MAX_ROUTING_LAYER            "METAL6"    ;# Max routing
    layer
```

Cuadro 8: Bloque 8 `setup.tcl`

9. Se asigna el valor de la variable `$NDM_REFERENCE_LIB_DIRS`, la ruta de la librería de referencia, a la variable `LIBRARY_FILES`.

```
set LIBRARY_FILES "${NDM_REFERENCE_LIB_DIRS}"
```

Cuadro 9: Bloque 9 `setup.tcl`

10. Se añade una ruta adicional al listado de lugares (`search_path`) donde la herramienta buscará las librerías NLDM.

```
lappend search_path "${DESIGN_REF_PATH}/lib/db_nldm"
```

Cuadro 10: Bloque 10 `setup.tcl`

11. Se le indica a la herramienta que use hasta 12 núcleos de procesamiento. Esto maximiza la eficiencia del proceso de síntesis, permitiendo que se paralelicen las tareas y, por tanto, se complete el proceso más rápidamente.

```
set_host_options -max_cores 12
```

Cuadro 11: Bloque 11 `setup.tcl`

12. Se define la librería de referencia (`TSMCWorkspace.ndm`), y el nombre de la librería de diseño como `LIB_TEST`. Este es el espacio donde se almacenarán todos los archivos resultados del flujo de síntesis física.

```
set ndm_reference_library ${NDM_REFERENCE_LIB_DIRS}
set ndm_design_library LIB_TEST
```

Cuadro 12: Bloque 12 `setup.tcl`

13. Este bloque verifica si la librería de diseño ya existe. Si existe, se elimina para evitar conflictos con versiones anteriores del diseño.

```
if { [file exists $ndm_design_library] } {
    puts "Libreria existente: $ndm_design_library,
       eliminando..."
    sh rm -rf $ndm_design_library
    if { [file exists $ndm_design_library] } {
        puts "Error: No se pudo eliminar la libreria:
           $ndm_design_library"
        exit 1
    }
}
```

Cuadro 13: Bloque 13 `setup.tcl`

14. Se crea una nueva librería de diseño y se le asocia la librería de referencia.

```
puts "Creando libreria: $ndm_design_library"
create_lib -technology $TECH_FILE -ref_libs $LIBRARY_FILES $
   {ndm_design_library}
```

Cuadro 14: Bloque 14 `setup.tcl`

7.2. *Script* floorplan.tcl

El script `setup.tcl` es estándar para cualquier tipo de circuito, mientras que los *scripts* `floorplan.tcl` y `sintesis_fisica_top.tcl` presentan variaciones en ciertos aspectos según el diseño específico. En esta sección se detalla y explica el flujo completo aplicado al diseño de un circuito NOT. Además, se realizó la síntesis física de una ALU de 4 *bits*, un sumador de 32 *bits*, así como del circuito correspondiente al proyecto "El Gran Jaguar".

1. Se crean celdas de esquinas usando la celda `PCORNER`, las cuales ayudan a establecer los límites y proporcionar puntos de referencia para el diseño del `layout`.

```
create_cell {CORNER1 CORNER2 CORNER3 CORNER4} PCORNER
```

Cuadro 15: Bloque 1 `floorplan.tcl`

2. Se crean celdas de pads de alimentación (`PVDD1` para `VDD` y `PVSS1` para `VSS`) usando las celdas `PVDD1CDG` y `PVSS1CDG`. Cabe mencionar que los pads son puntos de conexión para las señales de alimentación y tierra del chip, y se colocan para permitir la conexión de las señales desde el exterior del chip.

```
    create_cell {PVDD1} PVDD1CDG
    create_cell {PVSS1} PVSS1CDG
```

Cuadro 16: Bloque 2 `floorplan.tcl`

3. Se resuelve y asegura que las redes de alimentación estén definidas correctamente, y se crean *nets* para `VDD` y `VSS`. Se conectan las *nets* a los pines correspondientes. Seguidamente, se asegura que todas las conexiones de la red de energía y tierra se completen automáticamente, abarcando áreas o componentes no conectados previamente o ajustando conexiones incompletas. Los comandos `connect_pg_net -net VDD [get_pins -physical_context *VDD]` y `connect_pg_net -net VSS [get_pins -physical_context *VSS]` especifican la red a conectar y luego indican que se deben conectar todos los pines cuyo nombre termine en `VDD` o `VSS`. El carácter `*` actúa como un comodín, permitiendo que cualquier pin cuyo nombre termine con `VDD` o `VSS` sea identificado y conectado automáticamente a las redes correspondientes. Estas conexiones son lógicas, por lo que no se ven reflejadas en la interfaz, como se puede observar en la Figura 10.

```
    resolve_pg_nets
    create_net -power VDD
    create_net -ground VSS
    connect_pg_net -net VDD [get_pins -physical_context *VDD]
    connect_pg_net -net VSS [get_pins -physical_context *VSS]
    connect_pg_net -automatic
    report_cells -power
```

Cuadro 17: Bloque 3 `floorplan.tcl`

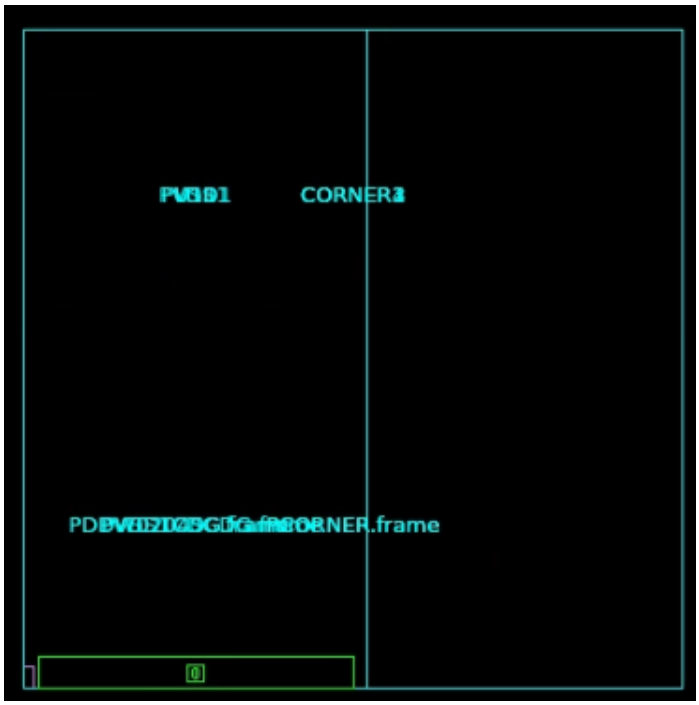


Figura 10: Resultados de los primeros tres pasos de la creación de `floorplan`

4. Se inicializa el *floorplan*.

```
initialize_floorplan -site_def unit -use_site_row -keep_all
   -side_length {100 100} -core_offset {125}
```

Cuadro 18: Bloque 4 `floorplan.tcl`

- El argumento `-site_def unit` define el tamaño de una celda estándar, que en el archivo techfile está especificado como 3.92 µm bajo el nombre `unit`. Esto asegura que el tamaño del núcleo sea un múltiplo de esta unidad para generar filas alineadas. Por ejemplo, en este caso, aunque se especifique un núcleo de 100 x 100 μm^2, el programa ajusta a 98 x 98 μm^2 para formar 25 filas exactas de 3.92 µm.
- El argumento `-use_site_row` indica que se debe usar la dimensión `unit` para definir las filas dentro del core.
- El argumento `-keep_all` preserva todas las celdas instanciadas previamente.
- El argumento `-side_length {100 100}` define un core con dimensiones de 100 x 100 μm^2.
- El argumento `-core_offset {125}` especifica la distancia entre el borde del core y el borde del margen externo. Este valor de 125 micrómetros se eligió porque, en trabajos de graduación previos, demostró generar los mejores resultados, asegurando un diseño más eficiente y evitando interrupciones en el enrutamiento de señales. El *core_ offset* es fundamental para asegurar que haya suficiente espacio entre el *core* y el borde del chip. Este espacio es clave para que las señales puedan enrutarse correctamente hacia las áreas periféricas, como los pines de *I/O*. Además, ayuda a cumplir con las reglas de diseño, evitando que las rutas de señales se congestionen cerca del borde del *core*. Asimismo, mejora la organización general del diseño, lo que contribuye a un enrutamiento más limpio y reduce problemas como interferencias electromagnéticas.

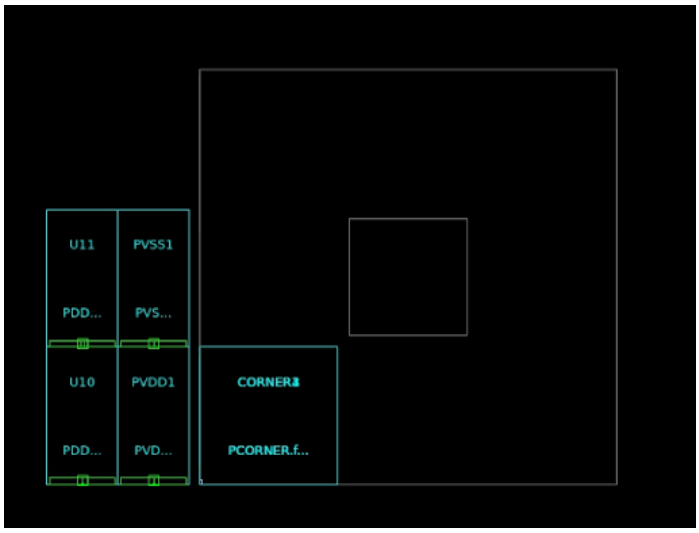


Figura 11: Inicialización del `floorplan`

5. Se crea un anillo de entrada/salida (IO) alrededor del *core* del diseño con el nombre `anillo_IO_NOT`. El `corner_height` de 115 µm define la altura de las esquinas del anillo. Esto proporciona un anillo de *pads* de IO alrededor del core, facilitando la conexión con el exterior.

```
create_io_ring -name anillo_IO_NOT -corner_height 115
```

Cuadro 19: Bloque 5 `floorplan.tcl`

6. Se agregan los *pads* `PVDD*` al lado izquierdo del anillo IO y los *pads* `PVSS*` al lado derecho. Finalmente, se colocan los *pads* de IO en el diseño.

```
add_to_io_guide [get_io_guides anillo_IO_NOT.left] PVDD*
add_to_io_guide [get_io_guides anillo_IO_NOT.right] PVSS*
place_io
```

Cuadro 20: Bloque 6 `floorplan.tcl`

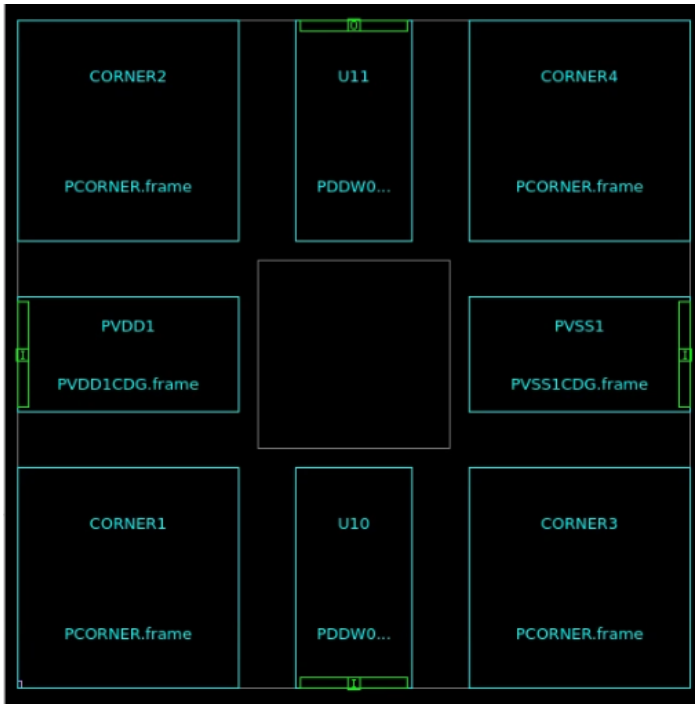


Figura 12: Colocación de los *pads* en el anillo de IO

7. Se crea un patrón para el anillo de PG (`power grid`) utilizando capas metálicas `METAL2` para las conexiones horizontales y `METAL3` para las verticales.

   - **Paso 1:** Los *tracks* horizontales del anillo tendrán un ancho de 2, un espaciado de 2 y se colocan en la segunda capa de metal. Para el caso de los *tracks* verticales, se tienen las mismas características para el ancho y el espaciado pero se colocan en la capa 3 de metal. El *spacing* de este comando se refiere a la distancia que se tendrá entre *tracks* paralelos que pertenezcan a este patrón.

- **Paso 2:** Luego de configurar el patrón, se establece una estrategia. Esta estrategia se encarga de asociar un patrón a las *nets* que se quieran conectar. Asimismo, se define el desplazamiento del anillo respecto al borde del núcleo. En este caso, el anillo está separado 1 μm horizontalmente y verticalmente del *core*.
- **Paso 3:** Se compila la estrategia y se conecta la red VDD Y VSS al núcleo del diseño usando el patrón especificado para garantizar que las señales se distribuyan de manera adecuada en todo el núcleo.

```
#Paso 1
create_pg_ring_pattern ring_pattern -horizontal_layer METAL2
    -horizontal_width {2} -horizontal_spacing {2}
   -vertical_layer METAL3 -vertical_width {2}
   -vertical_spacing {2}

#Paso 2
set_pg_strategy core_ring -pattern {{name: ring_pattern} {
   nets: {VDD VSS}} {offset: {1 1}}} -core

#Paso 3
compile_pg -strategies core_ring
```

Cuadro 21: Bloque 8 `floorplan.tcl`

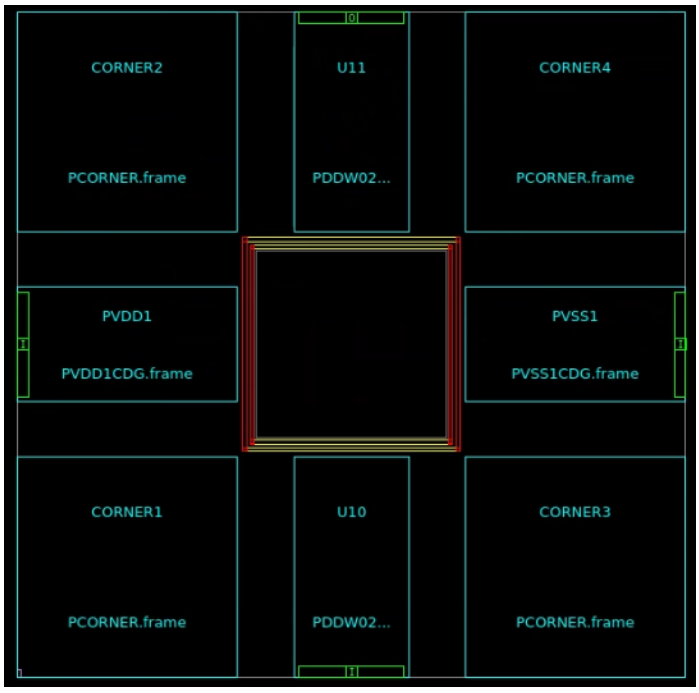


Figura 13: Colocación del anillo de PG

8. Se genera un nuevo patrón relacionado a las conexiones de celdas macro de PG. Las celdas macro son celdas que tienen un mayor tamaño que las celdas estándar.

```
#Paso 1
```

```
create_pg_macro_conn_pattern hm_pattern -pin_conn_type
   scattered_pin -layers {METAL2 METAL3} -nets {VDD VSS}
   -pin_layers {METAL2}

#Paso 2
set_app_options -name plan.pgroute.treat_pad_as_macro -value
    true

#Paso 3
set_pg_strategy macro_conn -macros [get_cells {PVDD* PVSS*}]
    -pattern {{name: hm_pattern} {nets: {VDD VSS}}}

#Paso 4
set_pg_strategy_via_rule macro_conn_via_rule -via_rule { { {
    {strategies: macro_conn}} { {existing: all} {layers:
   METAL3} } {via_master: default} }{{intersection:
   undefined}{via_master: NIL}}}

#Paso 5
compile_pg -strategies macro_conn -via_rule
   macro_conn_via_rule -tag test
```

Cuadro 22: Bloque 9 `floorplan.tcl`

- **Paso 1:** Se le indica el tipo de pin que tiene los pads como primer parámetro. El patrón indica la conexión de los pines de alimentación usando las capas de metal 2 y 3 para las redes de VDD y VSS. Finalmente, la opción de `-pin_layers` es utilizada para indicar las capas de metal en donde se deben de encontrar los pines para generar las conexiones.
- **Paso 2:** Indica que los pads deben tratarse como macros, lo que influye en cómo se enrutan las conexiones de alimentación hacia ellos.
- **Paso 3:** Establece la estrategia de conexión de energía para las macros, en este caso los pads de alimentación identificados como PVDD* y PVSS*. Usa el patrón definido anteriormente (`hm_pattern`) para conectar las redes de VDD y VSS a estos pads.
- **Paso 4:**Define una regla de uso de vías para conectar las diferentes capas de metal en las conexiones de alimentación. Asimismo, se especifica que se usarán vías en la capa 3, con la estrategia definida previamente.
- **Paso 5:** Compila las estrategias definidas para aplicar la conexión de alimentación a los pads de VDD y VSS, utilizando el patrón de macros y las reglas de vías definidas.

9. Se crea y compila la malla de alimentación para las redes de alimentación.

```
#Paso 1
connect_pg_net -automatic

#Paso 2
create_pg_mesh_pattern mesh_pattern -layers {{{
   vertical_layer: METAL3} {width: 4.2}{ pitch: 42} {
   spacing: interleaving }}}
```

```
#Paso 3
set_pg_strategy mesh_strategy -polygon {{125.000 118.000}
    {224.670 230.000 }} -pattern {{ pattern: mesh_pattern }{
    nets: {VDD VSS }}} -blockage {macros: all}

#Paso 4
create_pg_std_cell_conn_pattern std_cell_pattern

#Paso 5
set_pg_strategy std_cell_strategy -core -pattern {{pattern:
    std_cell_pattern}{nets: {VDD VSS}}}

#Paso 6
compile_pg
```

Cuadro 23: Bloque 10 `floorplan.tcl`

- **Paso 1:** Conecta automáticamente las redes de VDD y VSS a los pines y macros que necesitan estar alimentados. Se vuelve a ejecutar para asegurar que las nuevas redes y celdas que se hayan añadido queden conectadas correctamente.
- **Paso 2:** Se crea un patrón de malla para la alimentación. Se define que los `tracks` verticales se colocan en la capa de metal 3. Establece el ancho de las líneas en 4.2 µm y el espacio entre estas en 42 µm. Finalmente, define un espaciado intercalado entre las líneas para maximizar la cobertura.
- **Paso 3:**Se define una estrategia para la malla de alimentación. Se delimita un polígono por medio de coordenadas (en µm) para la colocación de la malla y se aplica el patrón previamente definido a las redes de VDD y VSS. Además, el comando asegura que las áreas ocupadas por macros sean evitadas por la malla de alimentación.
- **Paso 4:** Crea un patrón de conexión de alimentación para celdas estándar (`std_cell`), que serán conectadas a las redes de VDD y VSS.
- **Paso 5:** Se define una estrategia de alimentación para las celdas estándar y se aplica el patrón creado previamente a las redes de alimentación.
- **Paso 6:** Compila todas las estrategias de alimentación que se definieron y realiza las conexiones físicas.

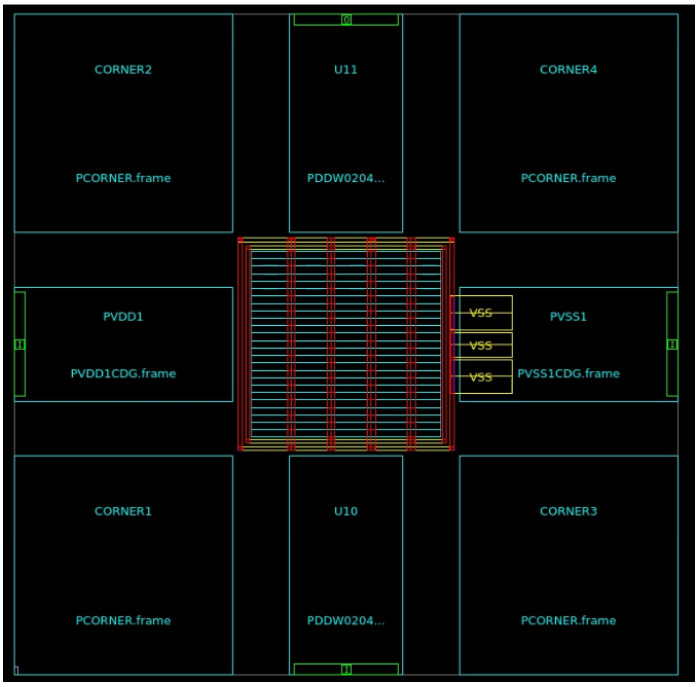


Figura 14: Conexión de las celdas estándar al anillo por medio de un *mesh*

10. Se combina la malla de alimentación (*PG mesh*) con el anillo de distribución de energía (*PG ring*) para las redes VDD y VSS. Esto asegura que tanto el anillo como los *tracks* de alimentación en las capas 2 y 3 estén conectados y trabajen en conjunto para distribuir las señales por todo el diseño.

```
merge_pg_mesh -nets {VDD VSS} -types {ring stripe} -layers {
    METAL2 METAL3}
```

Cuadro 24: Bloque 11 `floorplan.tcl`

11. *Placement* y legalización.

```
#Paso 1
set_app_options -name place.coarse.fix_hard_macros -value
    false

#Paso 2
set_app_options -name plan.place.auto_create_blockages
    -value auto

#Paso 3
create_placement -floorplan -timing_driven -congestion
    -effort high -congestion_effort high

#Paso 4
legalize_placement
```

Cuadro 25: Bloque 12 `floorplan.tcl`

- **Paso 1:** Configura que las macros no se fijen en una posición específica durante la colocación. Esto permite que se puedan mover si es necesario. Para esto la opción se pone en *false*.
- **Paso 2:** Se establecen bloqueos automáticos, lo cual asegura que las áreas donde no se deben colocar celdas, como sobre macros o bloques reservados, se definan automáticamente, evitando que el diseñador tenga que especificar manualmente estas restricciones.
- **Paso 3:** Crea el *placement* de las celdas en el diseño, optimizado para el *floorplan* y ajustado para mejorar la temporización y minimizar la congestión de las rutas.
- **Paso 4:** Este último comando ajusta y corrige la colocación de las celdas para cumplir con las reglas de diseño, asegurando que las celdas estén en ubicaciones válidas y que no hayan violaciones.

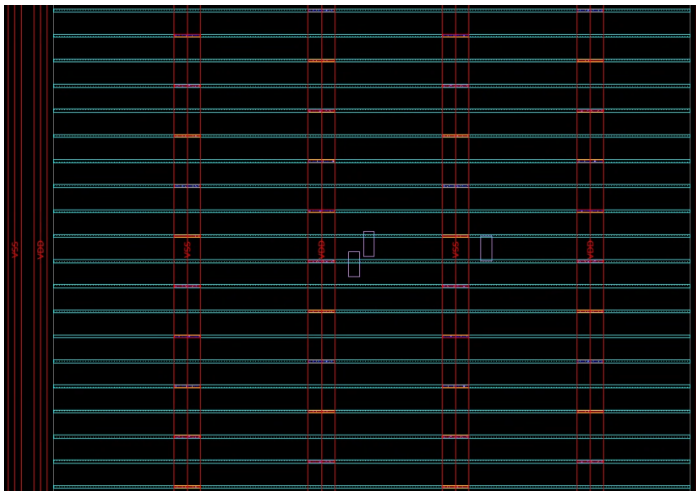


Figura 15: *Cell placement* inicial

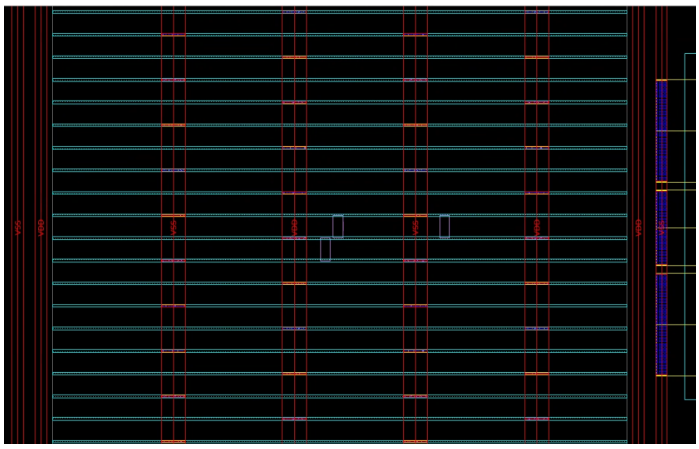


Figura 16: *Cell placement* luego de legalizarlo

7.3. *Script* sintesis_fisica_top.tcl

1. Comienza ejecutando el script `setup.tcl` y crea los directorios donde se almacenan los archivos resultantes de la síntesis física.

```
source scripts/setup.tcl

file mkdir -parents ./Outputs/IO/

file mkdir -parents ./Outputs/DRC/

file mkdir -parents ./Outputs/Fill/
```

Cuadro 26: Bloque 1 `sintesis_fisica_top.tcl`

2. Se lee el archivo Verilog sintetizado y el archivo SDC (*Synopsys Design Constraints*) que contiene las restricciones de diseño. Asimismo, lee los archivos que contienen los modelos de extracción de parásitos y el mapa que especifica las características de las capas de metal.

```
read_verilog ./Inputs/not_s.v

read_sdc -echo ./Inputs/not_s.sdc

read_parasitic_tech -tlup $TLUPLUS_FILE -layermap
   $MAP_FILE
```

Cuadro 27: Bloque 2 `sintesis_fisica_top.tcl`

3. Se eliminan todas las estrategias de red de alimentación (PG) previamente definidas en el diseño. Esto se hace para limpiar cualquier configuración anterior y evitar conflictos con nuevas configuraciones que se implementen.

```
remove_pg_strategies -all
```

Cuadro 28: Bloque 3 `sintesis_fisica_top.tcl`

4. Se ejecuta el *script* que aplica las reglas de antena y se ejecuta el *script* `floorplan.tcl`

```
source -echo -verbose "../../Librerias/Runset/
   antennaRule_018_6lm.tcl"

source scripts/floorplan.tcl
```

Cuadro 29: Bloque 4 `sintesis_fisica_top.tcl`

5. Se realiza el enrutamiento automático del circuito y se asegura que todos los pines se conecten con sus respectivas redes.

```
route_auto
```

Cuadro 30: Bloque 5 `sintesis_fisica_top.tcl`

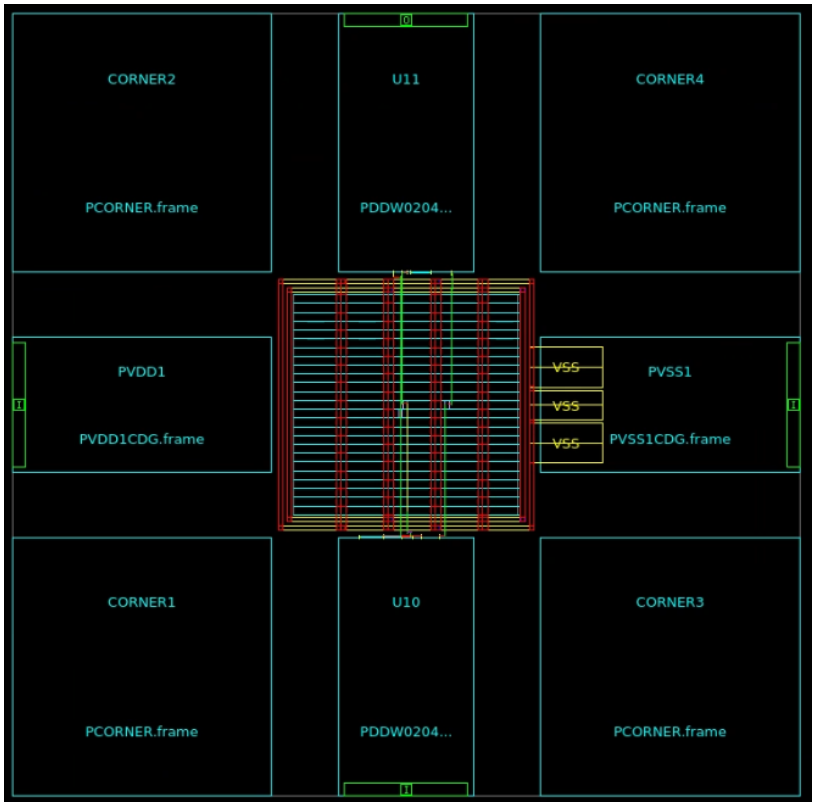


Figura 17: Circuito NOT luego de llevar a cabo el ruteo

6. Creación de *Filler Cells* y conexiones de *Power Grid*

```
#Paso 1
create_io_filler_cells -io_guides [get_io_guides {
   anillo_IO_NOT.top anillo_IO_NOT.right
   anillo_IO_NOT.left anillo_IO_NOT.bottom}]
   -reference_cells {PFILLER0005 PFILLER1 PFILLER5
   PFILLER05PFILLER10 PFILLER20}

#Paso 2
create_stdcell_fillers -lib_cells [get_lib_cells {
   TSMCWorkspace|FillersWorkspace/FILL64BWP7T
   TSMCWorkspace|FillersWorkspace/FILL32BWP7T
   TSMCWorkspace|FillersWorkspace/FILL16BWP7T
   TSMCWorkspace|FillersWorkspace/FILL8BWP7T
   TSMCWorkspace|FillersWorkspace/FILL4BWP7T
   TSMCWorkspace|FillersWorkspace/FILL2BWP7T
   TSMCWorkspace|FillersWorkspace/FILL1BWP7T}]

#Paso 3
connect_pg_net -automatic

#Paso 4
remove_stdcell_fillers_with_violation

#Paso 5
```

```
check_legality
```

Cuadro 31: Bloque 6 `sintesis_fisica_top.tcl`

- **Paso 1:** Se crean los *fillers* de los puertos de entrada y salida.
- **Paso 2:** Se crean los *fillers* de las celdas en el *core*, especificando varias opciones de tamaños para adaptarse a los diferentes espacios vacíos en el núcleo.
- **Paso 3:** Después de insertar los *fillers*, es necesario reconectar todas las redes de alimentación automáticamente, ya que los nuevos *fillers* pueden requerir conexiones adicionales. Cabe mencionar que si esto no se hace, se podrían presentar errores en la verificación de LVS.
- **Paso 4:** Se eliminan automáticamente los *fillers cells* que causen violaciones de diseño, como cortocircuitos o sobreposiciones con otras celdas.
- **Paso 5:** Se verifica que todas las celdas cumplan con todas las reglas de diseño

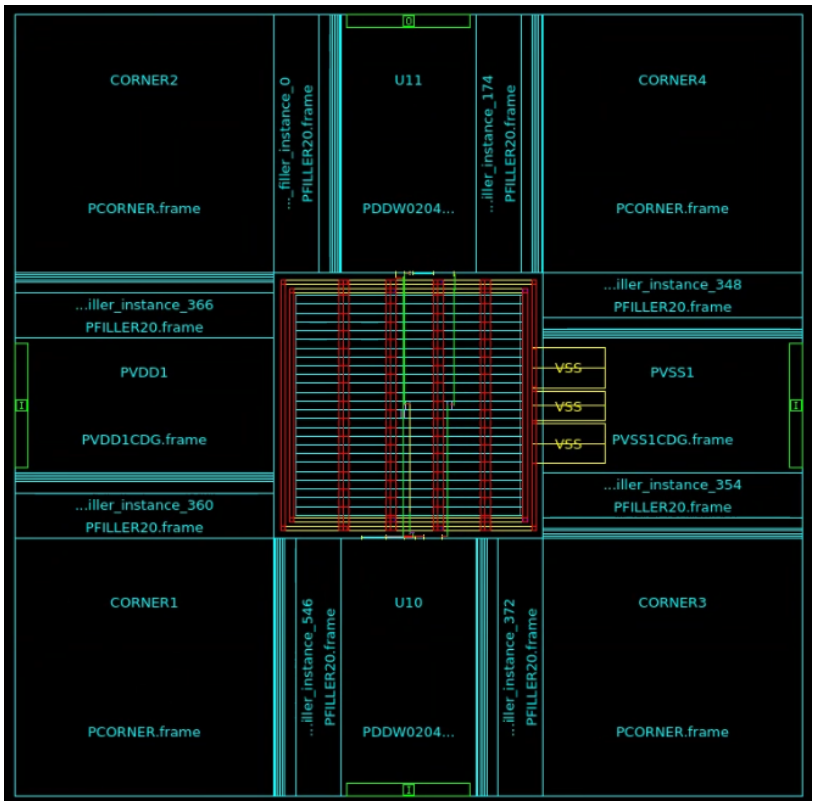


Figura 18: Circuito NOT luego de insertar los *fillers* de los puertos de IO.

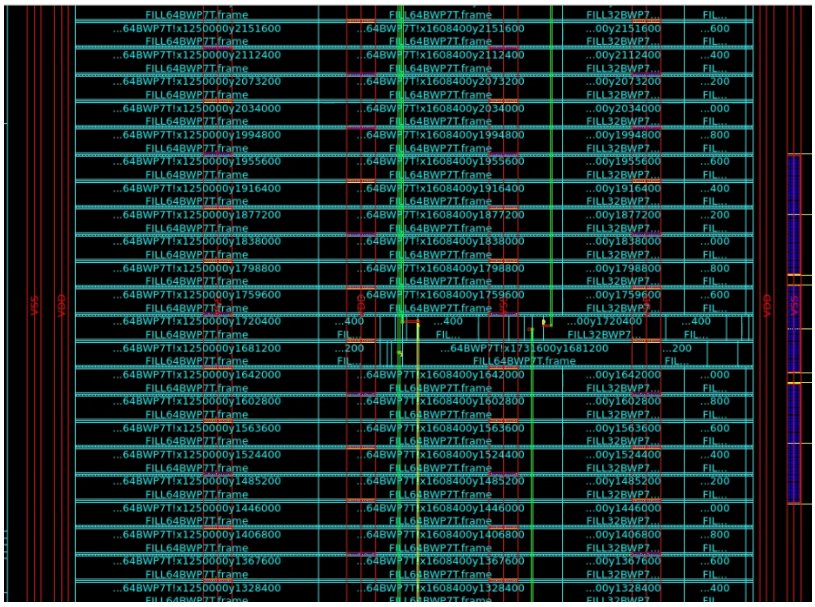


Figura 19: Circuito NOT luego de insertar los *fillers* en el *core*.

7. Se definen los directorios y archivos de *runset* para la verificación DRC, la corrección DRC y el relleno de metal. También habilita la corrección de errores de densidad luego de agregar los *fillers* de metal.

```
        set_app_options -list {signoff.check_design.run_dir {./
            Outputs/DRC/}}

        set_app_options -list {signoff.check_drc.run_dir {./
            Outputs/DRC/}}

        set_app_options -list {signoff.fix_drc.run_dir {./
            Outputs/DRC/}}

        set_app_options -list {signoff.create_metal_fill.run_dir
             {./Outputs/Fill/}}

        set_app_options -list {signoff.check_design.runset {../
            ../Librerias/Runset/
            ICVLM18_LM16_LM152_6M.215a_pre041518}}

        set_app_options -list {signoff.check_drc.runset {../../
            Librerias/Runset/ICVLM18_LM16_LM152_6M.215a_pre041518
            }}

        set_app_options -list {signoff.create_metal_fill.runset
            {../../Librerias/Runset/Dummy_Metal_ICV_0.18um.215a}}

        set_app_options -list {
            signoff.create_metal_fill.fix_density_errors true}
```

Cuadro 32: Bloque 7 `sintesis_fisica_top.tcl`

8. Guardado del diseño y verificación DRC

```
#Paso 1
signoff_create_metal_fill

#Paso 2
signoff_fix_drc

#Paso 3
signoff_check_drc

#Paso 4
save_block LIB_TEST:Not_IO
```

Cuadro 33: Bloque 8 `sintesis_fisica_top.tcl`

- **Paso 1:** Se inserta el relleno de metal.
- **Paso 2:** Se corrigen automáticamente las violaciones de DRC.
- **Paso 3:** Realiza la verificación DRC.
- **Paso 4:** Se guarda el bloque.

9. Exportación de archivos finales

```
write_verilog -include all ./Outputs/IO/VERILOG.v

write_gds -library LIB_TEST -design Not_IO -view design
   -hierarchy all -lib_cell_view frame ./Outputs/IO/
   chip.gds
```

Cuadro 34: Bloque 9 `sintesis_fisica_top.tcl`

- Se genera un archivo Verilog (VERILOG.v) que representa el diseño físico sintetizado.
- Se exporta el diseño a un archivo *GDS II* (chip.gds) en el directorio `/Outputs/IO/`.

7.4. Mejoras en la síntesis física

7.4.1. *Script* `setup.tcl`

1. Se define una variable llamada `CLOCK_SYNTHESIS` y al establecerla en `TRUE`, indica que la síntesis de reloj está habilitada. Si se cambia a `FALSE`, la síntesis de reloj estará deshabilitada.

```
set CLOCK_SYNTHESIS TRUE;
```

Cuadro 35: Bloque 1 `setup.tcl`

2. Establece `SIDE_LENGTH` como un array de dos valores {1750 1750}, que representa las dimensiones de los lados del *core* (en µm). El primer valor es el ancho y el segundo es la altura del núcleo.

```
set SIDE_LENGTH {1750 1750};
```

Cuadro 36: Bloque 2 `setup.tcl`

3. Define un *offset* de 125 µm que se añadirá a ambos lados del núcleo. Este *offset* permite la creación de redes de suministro de energía (anillo de PG).

```
set CORE_OFFSET 125;
```

Cuadro 37: Bloque 3 `setup.tcl`

4. Se define el nombre de la *top cell* del circuito, el módulo principal en el *netlist* en Verilog. Este módulo agrupa todas las celdas y conexiones del diseño en un solo módulo principal. Este módulo contiene los *pads* de entrada y salida que se conectan al mundo exterior. Durante la síntesis física, la herramienta ICC2 toma este *netlist* y lo convierte en un *layout* físico. Por esta razón, este módulo se identifica como el único que aparece una sola vez en el archivo Verilog.

```
set CIRCUIT_NAME "multiplier_16bit_IO";
```

Cuadro 38: Bloque 4 `setup.tcl`

5. Se define el nombre del anillo de PG como una variable de entorno en el *script* para que se pueda utilizar fácilmente en todos los comandos sin necesidad de repetir el nombre del anillo en múltiples lugares.

```
set ANILLO_NAME "ANILLO_IO";
```

Cuadro 39: Bloque 5 `setup.tcl`

6. Se calculan las dimensiones del núcleo, sumando el *CORE_ OFFSET* a cada lado del núcleo (`SIDE_LENGTH`). El resultado se almacena en `POLYGON_DIMENSIONS`, una variable que contiene las dimensiones ajustadas en formato `<ancho alto>`.

```
set adjusted_polygon_x [expr {[lindex $SIDE_LENGTH 0] +
    $CORE_OFFSET}]
set adjusted_polygon_y [expr {[lindex $SIDE_LENGTH 1] +
    $CORE_OFFSET}]
set POLYGON_DIMENSIONS "$adjusted_polygon_x
    $adjusted_polygon_y"
```

Cuadro 40: Bloque 6 `setup.tcl`

7. Se imprimen los parámetros actuales del diseño en la consola para verificar las configuraciones y cálculos realizados. En la Figura 20 se puede observar la impresión de los parámetros durante el desarrollo de la síntesis física de un multiplicador de 32 *bits*.

```
puts "Adjusted polygon dimensions: $POLYGON_DIMENSIONS"
puts "Clock synthesis enabled: $CLOCK_SYNTHESIS"
puts "Circuit name: $CIRCUIT_NAME"
```

Cuadro 41: Bloque 7 `setup.tcl`

8. Se copian dos archivos generados a partir de la síntesis lógica que serán usados como entradas para la síntesis física. Uno es el archivo Verilog con el *netlist* del circuito, y el otro es un archivo SDC con restricciones de diseño.

```
file copy -force "/home/nanoelectronica/Desktop/
    Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
    sintesis_logica/Outputs/MULT32b/Outputs_finales/
    MULT32b_sintesis_final.v" "./Inputs/"
file copy -force "/home/nanoelectronica/Desktop/
    Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
    sintesis_logica/Outputs/MULT32b/Outputs_finales/
    MULT32b_sintesis_final.sdc" "./Inputs/"
```

Cuadro 42: Bloque 8 `setup.tcl`

- Un ejemplo generalizado de la lógica empleada se puede ver a continuación:

```
file copy -force ``<ruta_al_archivo_verilog>''
    ``<ruta_de_destino/>''
file copy -force ``<ruta_al_archivo_sdc>'' ``<
    ruta_de_destino/>''
```

Cuadro 43: Bloque 8 `setup.tcl`

9. Se crean cuatro directorios (IO, DRC, Fill y Antena) dentro de la carpeta `./Outputs/` para organizar los diferentes tipos de archivos de salida.

```
file mkdir ./Outputs/IO/
file mkdir ./Outputs/DRC/
file mkdir ./Outputs/Fill/
file mkdir ./Outputs/Antena/
```

Cuadro 44: Bloque 9 `setup.tcl`

10. Se copia el archivo `ICVLM18_LM16_LM152_6M.ANT.215a_pre041518` al directorio `./Outputs/Antena/`. Este archivo es el *runset* empleado para llevar a cabo la verificación física de Antena.

```
file copy -force "<ruta_al_archivo_ANT>" "./Outputs/
    Antena/"
```

Cuadro 45: Bloque 10 `setup.tcl`

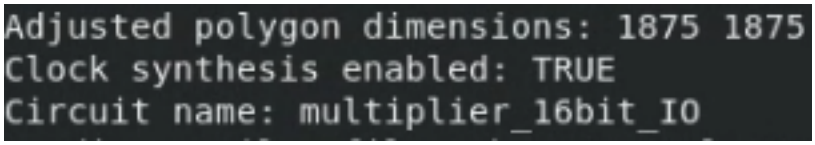


Figura 20: Parámetros de un circuito multiplicador de 32 *bits* durante su síntesis)

```
file delete -force  LIB_TEST.ndm
set command_log_file "./Logs/command.log"

set DESIGN_REF_PATH "../../Librerias"

set NDM_REFERENCE_LIB_DIRS  " \
    ${DESIGN_REF_PATH}/lib/ndm/TSMCWorkspace.ndm
    "
set TECH_FILE        "${DESIGN_REF_PATH}/tech/tsmc018_6lm.tf"
set MAP_FILE         "${DESIGN_REF_PATH}/tlu_and_map/star.map_6M
   "
set TLUPLUS_FILE "${DESIGN_REF_PATH}/tlu_and_map/
   t018lo_1p6m_typical.tluplus"

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

set MIN_ROUTING_LAYER            "METAL1"   ;# Min routing layer
set MAX_ROUTING_LAYER            "METAL6"   ;# Max routing layer

set LIBRARY_FILES "${NDM_REFERENCE_LIB_DIRS}"
lappend search_path "${DESIGN_REF_PATH}/lib/db_nldm"

set_host_options -max_cores 12
set ndm_reference_library ${NDM_REFERENCE_LIB_DIRS}
set ndm_design_library LIB_TEST

if { [file exists $ndm_design_library] } {
    puts "Libreria existente: $ndm_design_library, eliminando...
       "
    sh rm -rf $ndm_design_library
    if { [file exists $ndm_design_library] } {
        puts "Error: No se pudo eliminar la libreria:
           $ndm_design_library"
        exit 1
    }
}
puts "Creando libreria: $ndm_design_library"
create_lib -technology $TECH_FILE -ref_libs $LIBRARY_FILES ${
   ndm_design_library}

# Variables for automation
set CLOCK_SYNTHESIS TRUE  ;# Set to TRUE or FALSE to enable/
   disable clock synthesis
```

```
    set SIDE_LENGTH {1750 1750} ;# Define the side length of the
       core
    set CORE_OFFSET 125       ;# Define the core offset to be added
       to the side length
    set CIRCUIT_NAME "multiplier_16bit_IO" ;# Define the circuit
       name for save_block
    set ANILLO_NAME "ANILLO_IO" ;# Define the circuit name for
       save_block

    # Adjust the polygon dimensions by adding core offset to the
       side length
    set adjusted_polygon_x [expr {[lindex $SIDE_LENGTH 0] +
       $CORE_OFFSET}]
    set adjusted_polygon_y [expr {[lindex $SIDE_LENGTH 1] +
       $CORE_OFFSET}]
    set POLYGON_DIMENSIONS "$adjusted_polygon_x $adjusted_polygon_y"

    # Output the values for debugging
    puts "Adjusted polygon dimensions: $POLYGON_DIMENSIONS"
    puts "Clock synthesis enabled: $CLOCK_SYNTHESIS"
    puts "Circuit name: $CIRCUIT_NAME"

    file copy -force "/home/nanoelectronica/Desktop/
       Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_logica/
       Outputs/MULT32b/Outputs_finales/MULT32b_sintesis_final.v" "./
       Inputs/"
    file copy -force "/home/nanoelectronica/Desktop/
       Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_logica/
       Outputs/MULT32b/Outputs_finales/MULT32b_sintesis_final.sdc" "
       ./Inputs/"

    file mkdir ./Outputs/IO/
    file mkdir ./Outputs/DRC/
    file mkdir ./Outputs/Fill/
    file mkdir ./Outputs/Antena/

    file copy -force "/home/nanoelectronica/Desktop/
       Folder_de_Trabajo/Rui/TSMC/Librerias/Runset/
       ICVLM18_LM16_LM152_6M.ANT.215a_pre041518" "./Outputs/Antena/"
```

Cuadro 46: Script `setup.tcl` para un multiplicador de 32 *bits*

7.4.2. *Scripts* `floorplan.tcl` y `sintesis_fisica_top.tcl`

En el script `floorplan.tcl` se realizaron cambios significativos al reemplazar los valores fijos y repetitivos por las variables definidas en `setup.tcl`. Además, después de probar cada comando en la interfaz gráfica y confirmar que ofrecían los mejores resultados, se procedió a optimizar los *scripts*, conservando únicamente los comandos esenciales.

```
initialize_floorplan -site_def unit -use_site_row -keep_all
   -side_length $SIDE_LENGTH -core_offset $CORE_OFFSET
```

```
create_io_ring -name $ANILLO_NAME -corner_height 115

add_to_io_guide [get_io_guides $ANILLO_NAME.left] PVDD*
add_to_io_guide [get_io_guides $ANILLO_NAME.right] PVSS*
```

Cuadro 47: Comandos iniciales de `floorplan.tcl`

A continuación, en el bloque que configura las conexiones de `VDD` y `VSS` y establece un patrón de malla para ambas redes utilizando las coordenadas definidas en `$adjusted_polygon_x` y `$adjusted_polygon_y` para establecer la ubicación y las dimensiones del patrón de malla.

```
connect_pg_net -automatic

create_pg_mesh_pattern mesh_pattern -layers {{{ vertical_layer:
    METAL3} {width: 4.2}{ pitch: 42} {spacing: interleaving }}}

set_pg_strategy mesh_strategy -polygon "{125.000 118.000} {
    $adjusted_polygon_x $adjusted_polygon_y}" -pattern {{ pattern:
    mesh_pattern }{ nets: {VDD VSS }}} -blockage {macros: all}

create_pg_std_cell_conn_pattern std_cell_pattern

set_pg_strategy std_cell_strategy -core -pattern {{pattern:
    std_cell_pattern}{nets: {VDD VSS}}}

compile_pg
```

Cuadro 48: Conexiones de VDD y VSS y patrón de malla

```
#Creacion de corners
create_cell {CORNER1 CORNER2 CORNER3 CORNER4} PCORNER

#Creacion de pads para  VDD y VSS
create_cell {PVDD1} PVDD1CDG
create_cell {PVSS1} PVSS1CDG

#Creacion de nets de VDD y VSS
resolve_pg_nets
create_net -power VDD
create_net -ground VSS
connect_pg_net -net VDD [get_pins -physical_context *VDD]
connect_pg_net -net VSS [get_pins -physical_context *VSS]
connect_pg_net -automatic
report_cells -power

#Floorplan inicial
initialize_floorplan -site_def unit -use_site_row -keep_all
   -side_length $SIDE_LENGTH -core_offset $CORE_OFFSET

#Creacion del anillo IO
create_io_ring -name $ANILLO_NAME -corner_height 115

#Coloca los pines de entradas y salidas (Pads) en un lugar
   arbitrario de no ser especificado en el floorplan
add_to_io_guide [get_io_guides $ANILLO_NAME.left] PVDD*
add_to_io_guide [get_io_guides $ANILLO_NAME.right] PVSS*
place_io

#Creacion del anillo de PG
create_pg_ring_pattern ring_pattern -horizontal_layer METAL2
   -horizontal_width {2} -horizontal_spacing {2}
   -vertical_layer METAL3 -vertical_width {2} -vertical_spacing
   {2}
set_pg_strategy core_ring    -pattern {{name: ring_pattern}    {
   nets: {VDD VSS}} {offset: {1 1}}} -core
compile_pg -strategies core_ring

# Conexion del anillo IO con los pads de alimentacion
create_pg_macro_conn_pattern hm_pattern -pin_conn_type
   scattered_pin -layers {METAL2 METAL3} -nets {VDD VSS}
   -pin_layers {METAL2}
set_app_options -name plan.pgroute.treat_pad_as_macro -value
   true
set_pg_strategy macro_conn -macros [get_cells {PVDD* PVSS*}]
   -pattern {{name: hm_pattern} {nets: {VDD VSS}}}
set_pg_strategy_via_rule macro_conn_via_rule -via_rule { { { {
   strategies: macro_conn}} { {existing: all} {layers: METAL3} }
    {via_master: default} }{{intersection: undefined}{
   via_master: NIL}}}
compile_pg -strategies macro_conn -via_rule macro_conn_via_rule
   -tag test
```

```
    connect_pg_net -automatic
    create_pg_mesh_pattern mesh_pattern -layers {{{ vertical_layer:
       METAL3} {width: 4.2}{ pitch: 42} {spacing: interleaving }}}
    set_pg_strategy mesh_strategy -polygon "{125.000 118.000} {
       $adjusted_polygon_x $adjusted_polygon_y}" -pattern {{
       pattern: mesh_pattern }{ nets: {VDD VSS }}} -blockage {
       macros: all}
    create_pg_std_cell_conn_pattern std_cell_pattern
    set_pg_strategy std_cell_strategy -core -pattern {{pattern:
       std_cell_pattern}{nets: {VDD VSS}}}
    compile_pg

    #Merge del mesh con el pg ring
    merge_pg_mesh -nets {VDD VSS} -types {ring stripe} -layers {
       METAL2 METAL3}

    #Creamos el placement
    set_app_options -name place.coarse.fix_hard_macros -value false
    set_app_options -name plan.place.auto_create_blockages -value
       auto
    create_placement -floorplan -timing_driven -congestion -effort
       high -congestion_effort high
    legalize_placement
```

Cuadro 49: Script `floorplan.tcl` para un multiplicador de 32 *bits*

En el *script* `sintesis_fisica_top.tcl` se emplean las variables `CLOCK_SYNTHESIS` y `CIRCUIT_NAME`. `CIRCUIT_NAME` se usa en comandos de guardado de bloque y exportación de archivos en formatos Verilog y GDS.

```
save_block DESIGN_LIB_NAME:$CIRCUIT_NAME
write_verilog -include all ./Outputs/IO/$CIRCUIT_NAME.v
write_gds -library LIB_TEST -design $CIRCUIT_NAME -view design
   -hierarchy all -lib_cell_view frame ./Outputs/IO/chip.gds
```

Cuadro 50: Comandos de exportación de bloque

`CLOCK_SYNTHESIS` se emplea para habilitar o deshabilitar la síntesis de reloj, activándose solo en circuitos secuenciales. Al establecerlo en `TRUE`, el script ejecuta una serie de pasos para generar y optimizar el árbol de reloj del circuito. En pruebas iniciales del flujo de síntesis física, utilizando *scripts* de iteraciones anteriores, se presentaba un error (ver Figura 21) al descomentar el bloque de síntesis de reloj. Para abordar este problema, se consultó la herramienta ICC2, donde se identificó el comando necesario para iniciar el proceso de síntesis de reloj. A través del comando `create_clock`, se define una señal de reloj, especificando su periodo y asociándola a la red correspondiente en el netlist.

```
if { $CLOCK_SYNTHESIS == "TRUE" } {
    puts "Sintetizando relojes..."

    create_clock -period 2 -name clk [get_nets -design [
       current_block] {clk}]

    check_clock_trees -clocks clk
```

```
    check_design -checks pre_clock_tree_stage

    synthesize_clock_trees -clocks clk -postroute
       -routed_clock_stage detail_with_signal_routes

    clock_opt -list_only

    check_design -checks cts_qor
} else {
    puts "Clock synthesis deshabilitada."
}
```

Cuadro 51: Sintesis de reloj

- Se evalúa si la variable `CLOCK_SYNTHESIS` es igual a `"TRUE"`. Si es así, se ejecuta el bloque de comandos asociado a la síntesis de reloj.
- `create_clock` define un reloj en el diseño.
  - `-period 2` define el período del reloj como 0.5 unidades de tiempo (en este caso, nanosegundos).
  - `-name clk` asigna el nombre `clk` al reloj.
  - `[get_nets -design [current_block] {clk}]` obtiene la red de señal asociada al nombre `clk` dentro del bloque de diseño actual.
- `check_clock_trees -clocks clk` asegura las conexiones necesarias y verifica el netlist, restricciones de señal, timing o ruteo.
- `check_design -checks pre_clock_tree_stage` verifica el diseño antes de la creación y optimización del árbol de reloj.
- `synthesize_clock_trees -clocks clk -postroute -routed_clock_stage detail_-with_signal_routes` sintetiza la señal de reloj después del enrutamiento de señal.
- `clock_opt -list_only` optimiza la señal del reloj.
- `check_design -checks cts_qor` realiza una última verificación luego del enrutamiento.

```
Sintetizando relojes...
Warning: Nothing implicitly matched 'clk' (SEL-003)
Error: Nothing matched for -clocks (SEL-005)
Information: The command 'check_clock_trees' cleared the undo history. (UNDO-016)
Error: Failed to initialize CTS session.
       Use error_info for more info. (CMD-013)
Information: script '/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_mult32_vclock/sintesis_fisica_top.tcl'
             stopped at line 25 due to error. (CMD-081)
Extended error info:
Failed to initialize CTS session.
    while executing
"check_clock_trees -clocks clk"
    (file "/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_mult32_vclock/sintesis_fisica_top.tcl" line 25)
 -- End Extended Error Info
Error  when sourcing file /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_mult32_vclock/sintesis_fisica_top.tcl
       Tcl stack follows:
Failed to initialize CTS session.
    while executing
"check_clock_trees -clocks clk"
    (file "/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_mult32_vclock/sintesis_fisica_top.tcl" line 25)
    invoked from within
"source -verbose -echo /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_mult32_vclock/sintesis_fisica_top.tcl"
    (in namespace eval "::" script line 1)
    invoked from within
"namespace eval :: {source -verbose -echo /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_mult32_vclock/sint..."
    (in namespace eval "::" script line 1)
    invoked from within
"namespace eval :: $source_cmd "
```

Figura 21: Error obtenido en la síntesis física inicial (sin creación de reloj)

```
source scripts/setup.tcl

#Abrimos el verilog file sintetizado

read_verilog ./Inputs/MULT32b_sintesis_final.v
read_sdc -echo ./Inputs/MULT32b_sintesis_final.sdc

# read_verilog ./Inputs/granjaguar.v
# read_sdc -echo ./Inputs/granjaguar.sdc

read_parasitic_tech -tlup $TLUPLUS_FILE -layermap $MAP_FILE

remove_pg_strategies -all

#antenna
source -echo -verbose "../../Librerias/Runset/
    antennaRule_018_6lm.tcl"

source scripts/floorplan.tcl

#Sintetizacion de relojes

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

```
#Ruteamos
check_routability -check_pg_blocked_ports true
check_design -checks pre_route_stage
route_auto

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

# Configuramos el DRC runset file
set_app_options -list {signoff.check_design.run_dir {./Outputs/
   DRC/}}
set_app_options -list {signoff.check_drc.run_dir {./Outputs/DRC
   /}}
set_app_options -list {signoff.fix_drc.run_dir {./Outputs/DRC/}}
set_app_options -list {signoff.create_metal_fill.run_dir {./
   Outputs/Fill/}}

set_app_options -list {signoff.check_design.runset {../../
   Librerias/Runset/ICVLM18_LM16_LM152_6M.215a_pre041518}}
set_app_options -list {signoff.check_drc.runset {../../Librerias
   /Runset/ICVLM18_LM16_LM152_6M.215a_pre041518}}

set_app_options -list {signoff.create_metal_fill.runset {../../
   Librerias/Runset/Dummy_Metal_ICV_0.18um.215a}}

set_app_options -list {
   signoff.create_metal_fill.fix_density_errors true}


#Guardamos el bloque y corremos DRC y su Fix
save_block LIB_TEST:$CIRCUIT_NAME

signoff_create_metal_fill -mode add
#signoff_create_metal_fill
signoff_fix_drc
signoff_check_drc

save_block LIB_TEST:$CIRCUIT_NAME

check_lvs
```

```
#Creamos el verilog equivalente de la sintesis fisica
write_verilog -include all ./Outputs/IO/$CIRCUIT_NAME.v

#ver errores gui show error data
#gds
write_gds -library LIB_TEST -design $CIRCUIT_NAME -view design
   -hierarchy all -lib_cell_view frame ./Outputs/IO/chip.gds
#exit
```

Cuadro 52: Script `sintesis_fisica_top.tcl` de un multiplicador de 32 *bits*

7.5. Jerarquía de directorios

7.5.1. Jerarquía de directorios general

El siguiente diagrama muestra la jerarquía de directorios diseñada para organizar el flujo completo de trabajo en el proyecto. Esta estructura fue acordada con los integrantes del equipo para facilitar la entrega de archivos necesarios en cada etapa del flujo y mantener un espacio de trabajo ordenado.

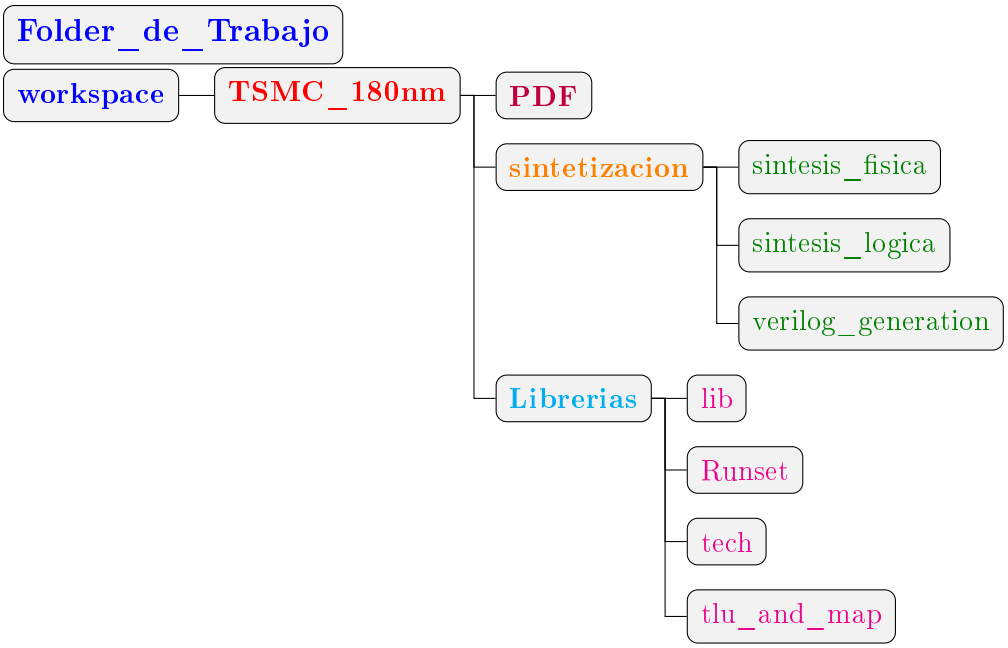


Figura 22: Jerarquía principal de directorios del proyecto

La distribución de cada directorio es la siguiente:

- **Folder_de_Trabajo**: Es el directorio principal que contiene todos los subdirectorios necesarios para el proyecto.

- **workspace**: Dentro de este directorio se crean carpetas específicas para la tecnología utilizada. En este caso, se emplea `TSMC_180nm`. Al recibir bibliotecas de una nueva tecnología, se agregaría un nuevo directorio con el nombre correspondiente a esa tecnología.
    - **TSMC_180nm**: Este directorio contiene tres subcarpetas principales:
        - **PDF**: Aquí se almacenan archivos PDF con información detallada sobre las celdas de la tecnología de 180 nm de TSMC (compuertas y pines de I/O). Incluye especificaciones técnicas, tablas de verdad y esquemáticos.
        - **sintetizacion**: Directorio designado para el trabajo de síntesis completo de los circuitos.
            - **sintesis_logica**: Contiene todos los archivos necesarios para realizar la síntesis lógica de cualquier circuito. Incluye subdirectorios para organizar archivos de entrada y salida. Esta sección no se explicará en detalle, ya que el enfoque principal de este proyecto es la síntesis física.
            - **sintesis_fisica**: Contiene todos los archivos necesarios para la síntesis física de los circuitos, organizados en subdirectorios de entrada y salida. Esta estructura se describirá en mayor detalle en la siguiente sección.
        - **Librerías**: Almacena los archivos de bibliotecas específicos para la tecnología de 180 nm.
            - **lib**: Incluye las bibliotecas utilizadas para la síntesis física y lógica. Específicamente, dentro del subdirectorio `ndm` se encuentran las librerías NDM que contienen la información lógica y física de las celdas.
            - **Runset**: Contiene todos los *scripts* necesarios para las verificaciones físicas, tales como el *runset* de DRC y el archivo de inserción de metal.
            - **tech**: Almacena el archivo con especificaciones de la tecnología de 180 nm y 6 capas de metal.
            - **tlu_and_map**: Contiene los archivos de mapeo con información sobre las capas de metal y su uso durante la etapa de ruteo.

Esta estructura de directorios garantiza un espacio de trabajo bien organizado, lo cual facilita el desarrollo del proyecto y asegura la correcta gestión de dependencias en cada etapa del flujo de diseño.

7.5.2. Jerarquía de directorios para síntesis física

Esta jerarquía de directorios es una extensión de la propuesta en el trabajo [20], en la que se ha añadido un subdirectorio adicional dentro de *Outputs*, denominado Antena. Se recomienda que, para cada circuito al cual se le realice la síntesis física, se cree un subdirectorio en *sintesis_fisica* con el nombre del circuito.

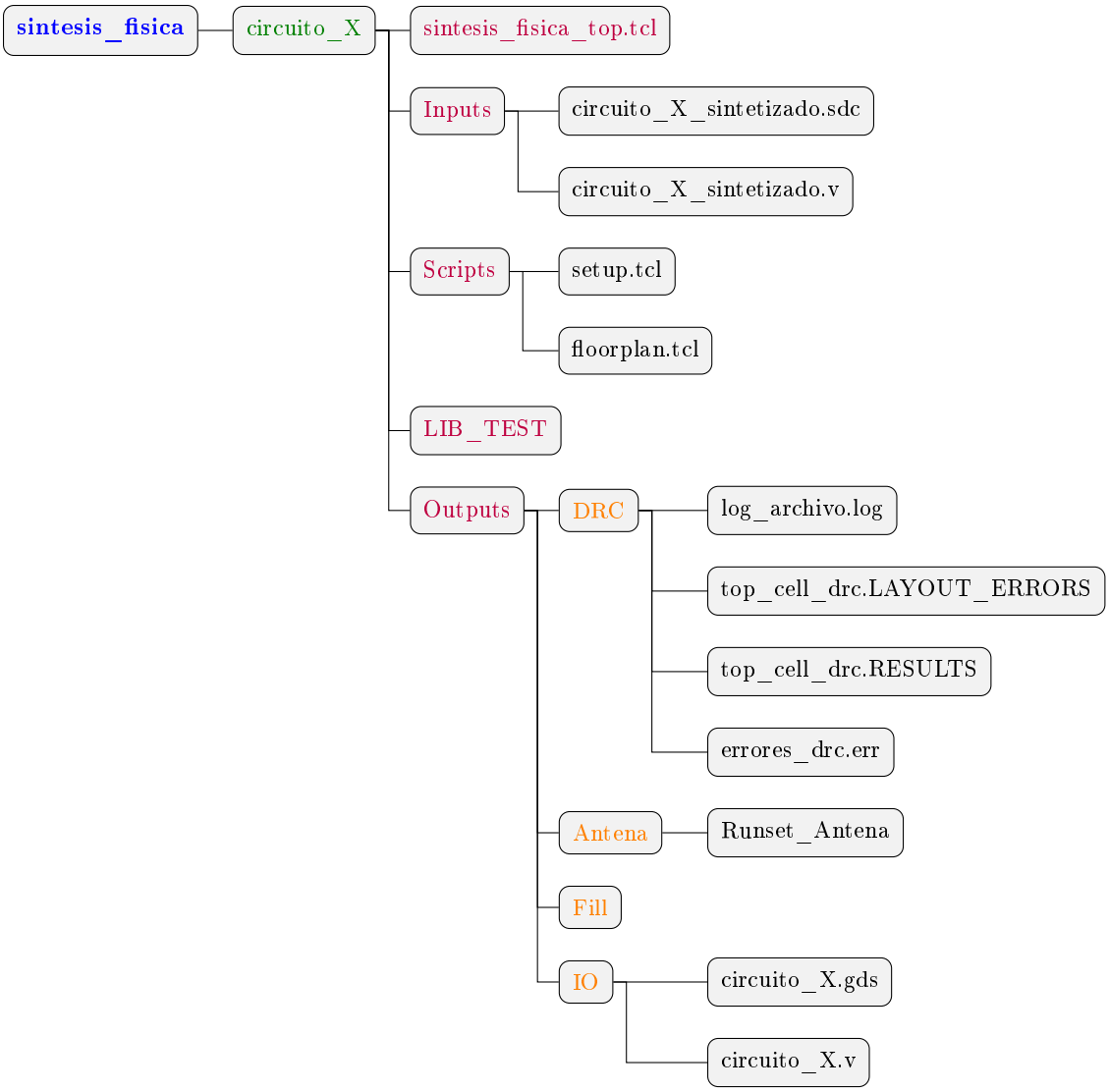


Figura 23: Jerarquía de directorios para síntesis física

A continuación, se describen los directorios en detalle:

- **circuito_X**: Este es el directorio específico para cada circuito en el que se va a realizar la síntesis física. Contiene todos los archivos necesarios para el proceso de síntesis y verificación.
  - **Inputs**: Este subdirectorio incluye los archivos requeridos para la síntesis física, los cuales se generan después de la síntesis lógica. Entre estos archivos se encuentran:
    - El archivo Verilog sintetizado del diseño.
    - El archivo `.sdc` con las restricciones de diseño necesarias para la síntesis física.
  - **LIB_TEST**: Es la librería de diseño *NDM*, en la cual se almacenará toda la información relevante del bloque que se está creando.

- **Outputs**: Este subdirectorio contiene todos los archivos generados después de la síntesis física y las verificaciones, incluyendo:
  - Los resultados de la verificación *DRC*.
  - Los resultados de la verificación de *antena*.
  - El archivo *GDS* que representa el diseño final para la fabricación.
  - El archivo Verilog actualizado tras la síntesis física.

CAPÍTULO 8

Síntesis física y verificaciones físicas

8.1. Síntesis física

8.1.1. Circuitos combinacionales

8.1.1.1. Compuerta NOT

El primer circuito pasó por la síntesis física en ICC2 y las verificaciones física DRC y de Antena. Se inició con este diseño sencillo para facilitar la exploración del entorno de ICC2, permitiendo llevar a cabo la síntesis física de manera detallada, paso por paso. Este enfoque permitió identificar áreas de mejora en los *scripts* utilizados en trabajos previos, asegurando el entendimiento de las herramientas y la metodología, lo cual es esencial en la síntesis de circuitos más complejos en futuras etapas. El resultado de síntesis física de esta compuerta se puede encontrar en la Figura 24.

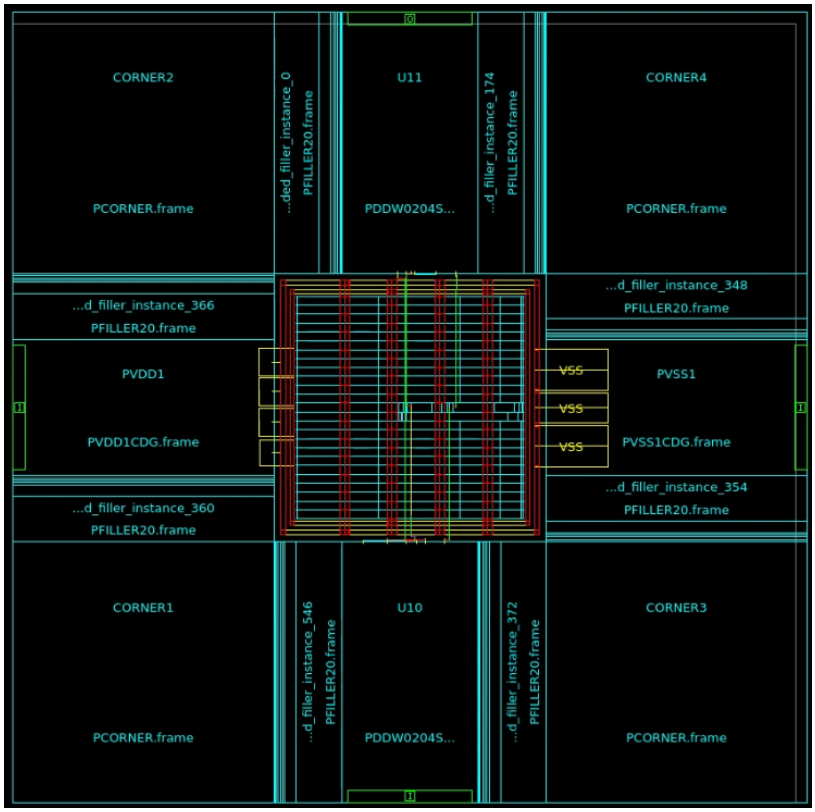


Figura 24: Síntesis física completa de un circuito NOT

Aunque el proceso de síntesis está mayormente automatizado, es necesario realizar ciertas modificaciones específicas en el script `setup.tcl` para adaptar el entorno a los requisitos del diseño. Entre estas modificaciones se incluyen:

- La definición de la variable que habilita la síntesis de reloj.
- Las dimensiones del núcleo del circuito.
- La especificación del valor de *offset*.
- La designación del nombre de la *top cell* del circuito.
- La configuración del nombre del anillo de PG.
- La definición de las rutas para los archivos generados en la síntesis lógica, incluyendo los archivos `.v` y `.sdc`.

8.1.1.2. ALU de 4 *bits*

El segundo circuito sintetizado fue una ALU de 4 *bits*, diseñada para realizar operaciones aritméticas y lógicas, incluyendo suma, resta y desplazamientos. Este diseño utiliza compuertas lógicas como OR, AND y XOR, además de componentes de selección como multiplexores, permitiendo manejar múltiples operaciones con un único conjunto de entradas y

salidas. Al incrementar tanto el número de entradas y salidas como la cantidad de celdas en el núcleo del circuito, se aumentó significativamente la complejidad y, consecuentemente, el área total ocupada en el diseño.

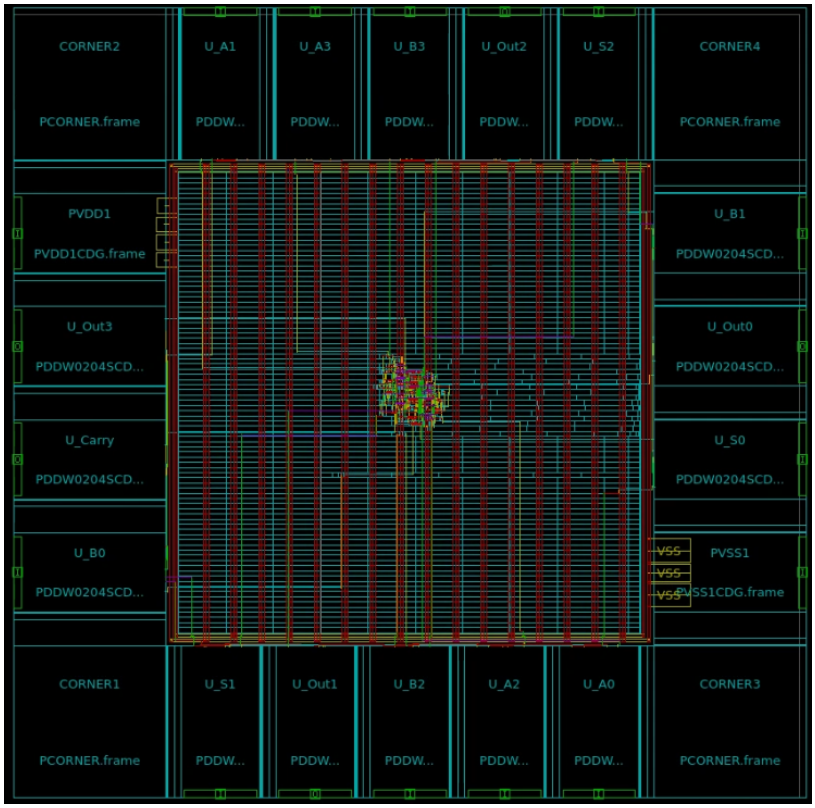


Figura 25: Síntesis física completa de un circuito ALU

8.1.1.3. Circuito sumador de 32 *bits*

El tercer circuito combinacional al cual se le llevó a cabo síntesis física fue un sumador *carry look-ahead* de 32 *bits*. Este tipo de sumador minimiza el tiempo necesario para calcular los acarreos de una operación de suma, ya que permite anticipar el valor del acarreo. El sumador de 32 *bits* se compone de varios bloques CLA de 4 *bits*, diseñados para funcionar en paralelo y calcular los acarreos internamente dentro de cada bloque.

Durante el proceso de síntesis física, se observó que los componentes responsables de la lógica de suma ocupan relativamente poco espacio dentro del diseño general. Sin embargo, el tamaño total del *core* se ve considerablemente afectado por la gran cantidad de pines de I/O necesarios para operar el circuito. Para cumplir con los requisitos de manufactura, el resto del área del *core* se cubrió mediante el uso de *metal fillers*. Estos *fillers* no contribuyen a la funcionalidad lógica del circuito, pero son esenciales para mantener la integridad estructural y funcional del diseño físico durante el proceso de fabricación.

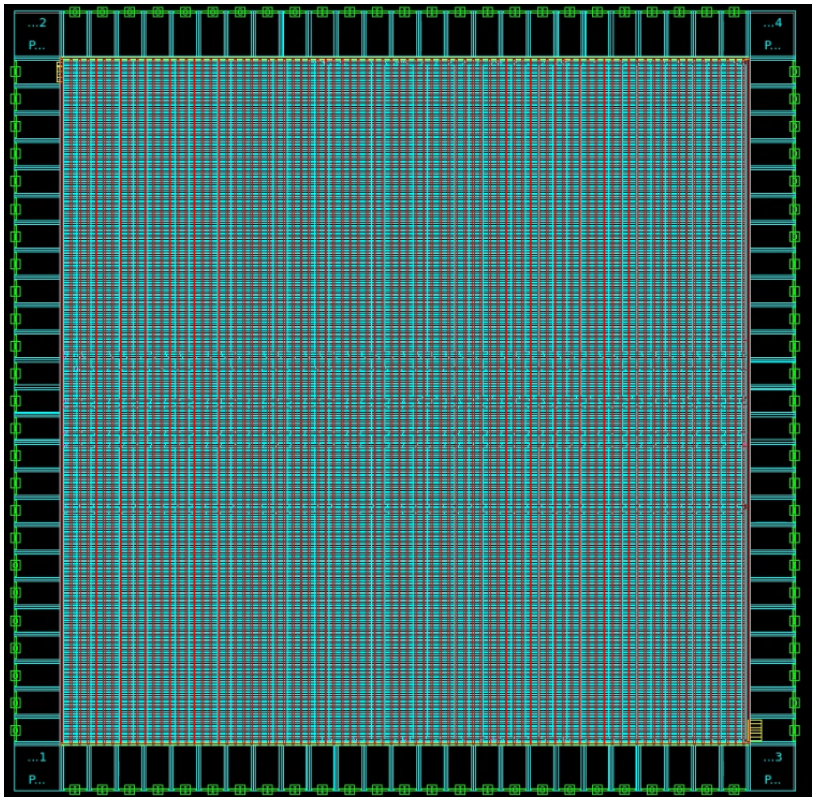


Figura 26: Síntesis física completa de un circuito sumador de 32 *bits*

8.1.2. Circuitos secuenciales

A diferencia de los primeros tres circuitos, los siguientes diseños incorporan lógica secuencial, lo cual implica un cambio significativo en la arquitectura. En lugar de depender únicamente de lógica combinacional, estos circuitos incluyen una señal de reloj y elementos de almacenamiento, como *flip-flops*, dentro del núcleo del diseño. La presencia de la señal de reloj permite sincronizar las operaciones a través de varios ciclos de reloj, manteniendo y actualizando los valores de salida en función del estado anterior

8.1.2.1. Circuito multiplicador de 32 *bits*

El multiplicador de 32 *bits* diseñado con entradas de 16 *bits* se implementa como un multiplicador con dos entradas de 16 *bits* y una salida de 32. A diferencia del sumador, un multiplicador requiere calcular múltiples sumas parciales y realizar desplazamientos, lo que implica un circuito más complejo y una cantidad significativamente mayor de componentes lógicos, como se puede observar en la Figura 27. En un sumador, la operación es instantánea, mientras que el multiplicador necesita múltiples ciclos de reloj para completar una sola multiplicación, lo que agrega complejidad de control en el diseño. Además, el multiplicador utiliza un registro de desplazamiento para almacenar el resultado parcial de la multiplicación y realizar desplazamientos en cada ciclo. Este registro debe gestionar la entrada de datos, realizar los desplazamientos y controlar la inserción de acarreos, lo cual agrega complejidad

adicional.

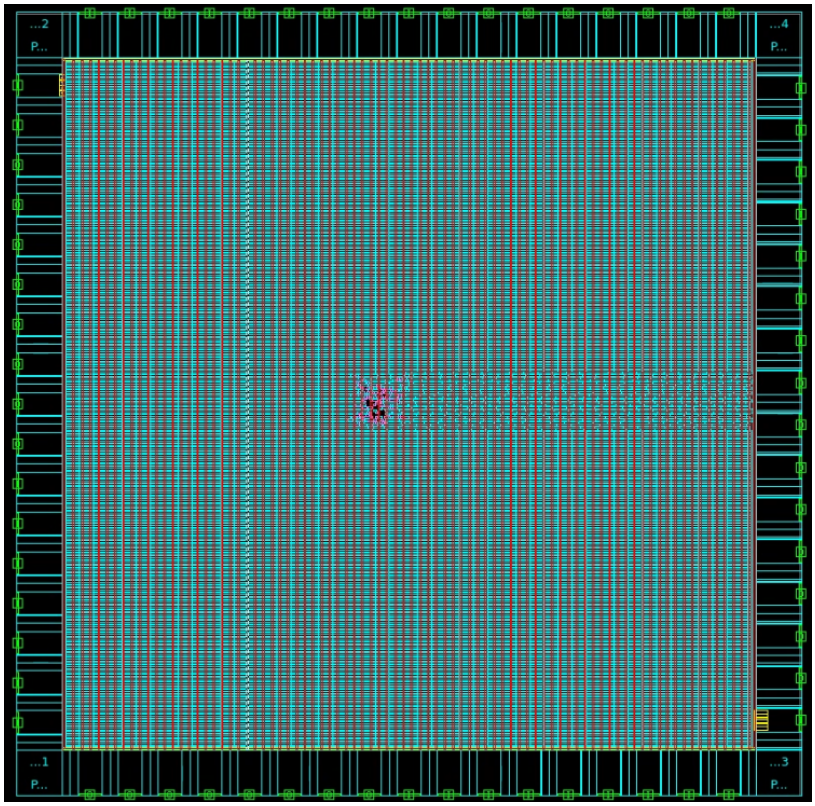


Figura 27: Síntesis física completa de un circuito sumador de 32 *bits*

8.1.2.2. Circuito divisor de 32 *bits*

El funcionamiento de este circuito se basa en un registro de 32 *bits* que inicialmente almacena el dividendo en sus *bits* menos significativos, dejando los *bits* superiores en cero para recibir los resultados parciales de las operaciones. En cada ciclo, el registro se desplaza, lo cual permite al circuito "probar"la posibilidad de restar el divisor de este valor parcial acumulado. Si la resta resulta exitosa, el cociente se ajusta registrando un "1.$^{\text{en}}$ la posición correspondiente, mientras que el valor resultante de la resta se acumula en el registro, quedando disponible para el siguiente ciclo de prueba. Al finalizar el número de ciclos necesarios, el cociente completo queda disponible en la salida. La complejidad del proceso de división se puede ver reflejada en el *layout* (véase Figura 28), donde se observa una mayor densidad de componentes en comparación con el sumador y multiplicador.

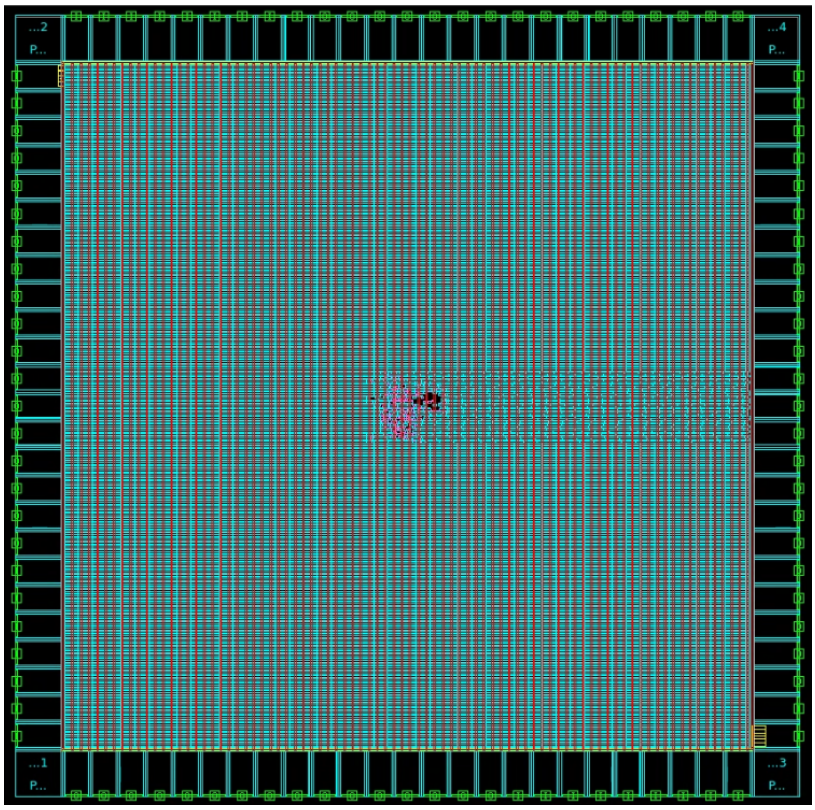


Figura 28: Síntesis física completa de un circuito sumador de 32 *bits*

8.1.2.3. Circuito "El Gran Jaguar"

Por último, se sintetizó el circuito "El Gran Jaguar", el cual consiste de una máquina de estados finitos que genera una secuencia de caracteres, representados en código ASCII, en una salida de 8 *bits*. En cada ciclo del contador, el valor de la salida cambia para representar un carácter distinto en la secuencia, que forma un mensaje completo. Asimismo, cuenta con la capacidad de controlar la velocidad de reproducción del mensaje.

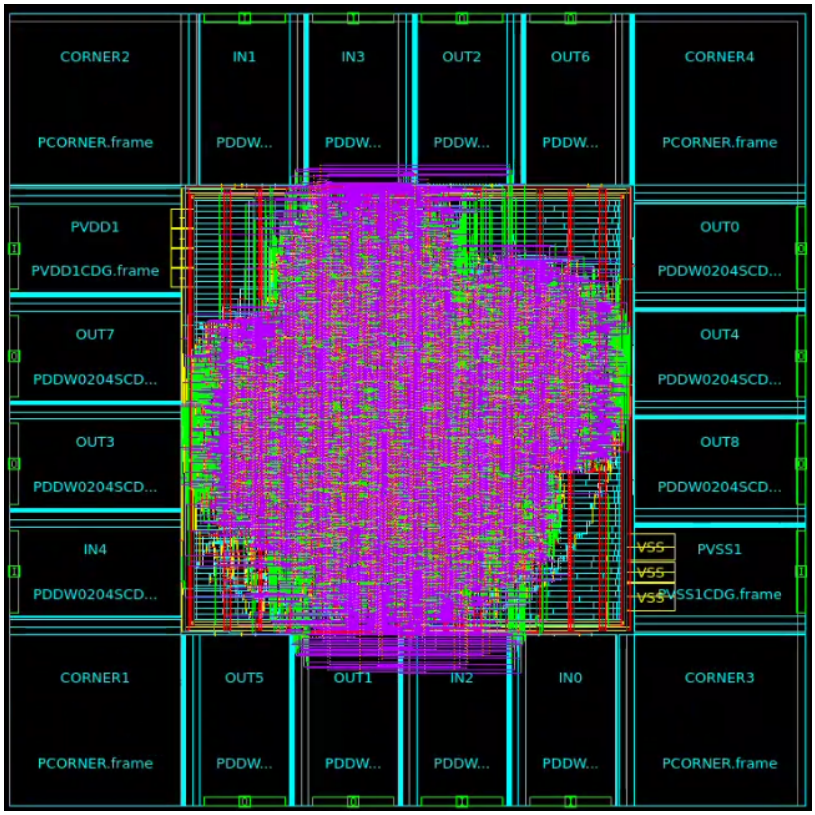


Figura 29: Síntesis física completa de un circuito "El Gran Jaguar"

8.2. DRC

Para realizar la verificación de reglas de diseño (DRC), es fundamental contar con una herramienta de validación adecuada y los archivos necesarios. *IC Validator* de *Synopsys* es la herramienta que permite llevar a cabo esta verificación, en la cual los procesos de síntesis física y DRC se ejecutan de forma consecutiva. Este flujo continuo facilita una verificación y corrección constante de errores a medida que el diseño avanza.

Un aspecto esencial de este proceso es el intercambio continuo de resultados, que permite una retroalimentación inmediata y una corrección de errores más eficiente. La herramienta *ICC II* permite configurar e invocar a *ICV* directamente desde su interfaz, proporcionando una integración que simplifica el flujo de trabajo. Los reportes de errores generados por *ICV* pueden visualizarse y corregirse en *ICC II*, acelerando el proceso de diseño y logrando una síntesis física más sólida y robusta.

8.2.1. *DRC Runset*

El objetivo principal del proyecto "El Gran Jaguar" es enviar a fabricación el circuito integrado. Para ello, se proporcionaron ciertos archivos necesarios para realizar las verificaciones físicas y asegurar que el diseño sea manufacturable. Estas verificaciones son

indispensables, ya que, sin ellas, aunque el circuito podría enviarse a fabricación, no se garantizaría el correcto funcionamiento del diseño. Para esta verificación, se utilizó el *runset* `ICVLM18_LM16_LM152_6M.215a_pre041518`, el cual contiene las reglas de diseño basadas en los estándares de fabricación de *TSMC*. Como se mencionó en secciones anteriores, el comando `signoff_check_drc` se emplea para ejecutar el *runset* de DRC, especificado en el script `setup.tcl`.

Este *runset* permite comentar o descomentar líneas en función de los requisitos específicos del circuito. En el proyecto [10], la línea `#define RECOMMEND` fue comentada, dado que esta contiene opciones recomendadas pero no obligatorias. Al dejarla descomentada, se generaban más de 400 errores. Con la configuración final del *runset*, presentada a continuación, se logró reducir estos errores a 6.

```
// OPTION SETUP
//===============

//#define CHECK_SRAM_EXCL                    /* turn on only when
    you want check M2 and upward layers in SRAM region covered
   by EXCL. otherwise please turn off it */
#define CHECK_ALRDL                         /* turn on only when
   you want check MD&VIAD rules. otherwise please turn off it */
//#define CHECK_DUMMY_ODPO                  /* turn on only when
   you want check Dummy OD&PO. otherwise please turn off it */
#define GUIDELINE_LUP                       /* turn on only when
   you want check latchup guideline rules. otherwise please turn
    off it */
//#define DISCONNECT_ALL_RESISTOR           /* turn on only when
   you want to disconnect all resistors to check latchup rules,
   otherwise please turn off it */
//#define CONNECT_ALL_RESISTOR              /* turn on only when
   you want to connect all resistors to check latchup rules,
   otherwise please turn off it */
#define GUIDELINE_ESD                       /* turn on only when
   you want check ESD guideline rules. otherwise please turn off
    it */
//#define RECOMMEND                         /* turn on only when
   you want to check Recommended rules. otherwise please turn
   off it */
//#define MIX_MODE                          /* turn on only when
   you use Mixed-Singal/RF process. otherwise please turn off it
    */
#define THICK_40K                          /* turn on only when 40
   KA Thick Top Metal is used. otherwise please turn off it */
//#define THICK_20K                         /* turn on only when
   20KA Thick Top Metal is used. otherwise please turn off it */
//#define LP                                /* turn on only when
   you use Low Power process. otherwise please turn off it  */
//#define C016                              /* turn on only when
   you use 0.16um process. otherwise please turn off it */
//#define C0152                             /* turn on only when
   you use 0.152um process. otherwise please turn off it */
#define _3_3V                               /* turn on only when
   HIGH_VOLTAGE is 3.3V. otherwise please turn off it */
```

```
//#define _5V                                   /* turn on only when
   HIGH_VOLTAGE is 5V. otherwise please turn off it */
//#define _2_5V                                 /* turn on only when
   HIGH_VOLTAGE is 2.5V. otherwise please turn off it */
#define _1_8V                                   /* turn on only when
   CORE_VOLTAGE is 1.8V. otherwise please turn off it */
//#define _1_5V                                 /* turn on only when
   CORE_VOLTAGE is 1.5V. otherwise please turn off it */
//#define CHECK_LATCHUP_BY_TEXT                /* Turn on to
   recognize IO PAD by following text   */
```

Cuadro 53: Configuración del *runset* de DRC con especificación de voltaje para IO y *core*, con opciones recomendadas desactivadas

8.2.2. *Metal Fill Runset*

En el proyecto [20], se realizó la verificación DRC de varios circuitos tras la adquisición del *runset* de relleno de metal de *TSMC*, compatible con *ICV*. Este *runset*, denominado `Dummy_Metal_ICV_0.18um.215a`, permite la inserción de patrones de relleno de metal (*dummy metal*) en las distintas capas del diseño, cumpliendo con las reglas de densidad de metal necesarias para evitar problemas en el proceso de fabricación. Al ejecutar el comando `signoff_create_metal_fill`, se generaron estos patrones de relleno, logrando, en este caso, una reducción de los errores de densidad de 6 a 2, específicamente en las capas de Metal 1 y Metal 2.

Para la inserción de relleno en capas de metal específicas, el *runset* incluye opciones como `FILL_Mx`, que habilitan el relleno en capas de metal seleccionadas desde `M1` hasta `M6`. Por ejemplo, activando `#define FILL_M1`, se incluye relleno en la capa de metal `M1`. Además, el *runset* permite ajustar el grosor de las capas de metal mediante opciones como `THICK_Mx`, `THICK_30K_Mx` y `THICK_40K_Mx`, las cuales especifican espesores específicos para cada capa, siendo especialmente útiles en capas superiores que requieren un mayor grosor para cumplir con los requisitos del diseño. Asimismo, se configuran las dimensiones, *offsets* y restricciones de densidad para los patrones de metal en cada capa del diseño. A continuación, se muestra parte del *runset* original en el que se configura lo mencionado anteriormente.

```
//******* Fill dimension & offset & density constraints
   *****************
DMW_M1 : double = 0.6;                  /* small DM1 width    */
DML_M1 : double = 0.6;                  /* small DM1 length   */
DMS_M1 : double = 0.6;                  /* small DM1 spacing    */
DM2M1 : double = 1.2;                   /* small DM1 to real metal
   spacing   */
MDMW_M1 : double = 1.0;                 /* middle DM1 width    */
MDML_M1 : double = 1.0;                 /* middle DM1 length   */
MDMS_M1 : double = 1.0;                 /* middle DM1 spacing    */
MDM2M1 : double = 1.2;                  /* middle DM1 to real metal
    spacing  */
WDMW_M1 : double = 3.0;                 /* large DM1 width    */
WDML_M1 : double = 3.0;                 /* large DM1 length   */
WDMS_M1 : double = 2.0;                 /* large DM1 spacing  */
```

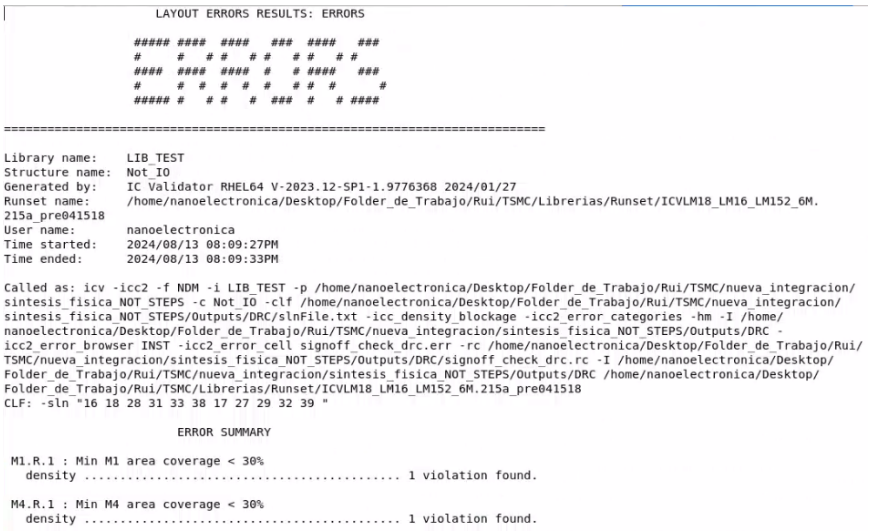
```
    WDM2M1 : double = 3.0;                  /* large DM1 to real metal
       spacing   */
    DM0_M1 : double = 0.2;                  /* small DM1 offset = space
       *(1/3) */
    MDM0_M1 : double = 0.33;                /* middle DM1 offset =
       space*(1/3) */
    WDM0_M1 : double = 0.66;                /* large DM1 offset = space
       *(1/3) */
    MIND_M1 : double = 0.3;                 /* M1 density for insertion
        */
    MAXD_M1 : double = 0.9;                 /* M1 density for remove
       inserted dummy */
    EMPTYD_M1 : double = 0.001;             /* M1 density for blank
       region selection */
    WDM2DM_M1 : double = 1.0;               /* large DM1 to DM1 spacing
        */
    DM2DM1 : double = 0.6;                  /* DM1 to existed dummy
       metal spacing */
    DMW_M2 : double = 0.6;                  /* small DM2 width    */
    DML_M2 : double = 0.6;                  /* small DM2 length   */
    DMS_M2 : double = 0.6;                  /* small DM2 spacing    */
    DM2M2 : double = 1.2;                   /* small DM2 to real metal
       spacing   */
    MDMW_M2 : double = 1.0;                 /* middle DM2 width    */
    MDML_M2 : double = 1.0;                 /* middle DM2 length   */
    MDMS_M2 : double = 1.0;                 /* middle DM2 spacing    */
    MDM2M2 : double = 1.2;                  /* middle DM2 to real metal
        spacing  */
    WDMW_M2 : double = 3.0;                 /* large DM2 width    */
    WDML_M2 : double = 3.0;                 /* large DM2 length   */
    WDMS_M2 : double = 2.0;                 /* large DM2 spacing  */
    WDM2M2 : double = 3.0;                  /* large DM2 to real metal
       spacing   */
    DM0_M2 : double = 0.2;                  /* small DM2 offset = space
       *(1/3) */
    MDM0_M2 : double = 0.33;                /* middle DM2 offset =
       space*(1/3) */
    WDM0_M2 : double = 0.66;                /* large DM2 offset = space
       *(1/3) */
    MIND_M2 : double = 0.3;                 /* M2 density for insertion
        */
    MAXD_M2 : double = 0.9;                 /* M2 density for remove
       inserted dummy */
    EMPTYD_M2 : double = 0.001;             /* M2 density for blank
       region selection */
    WDM2DM_M2 : double = 1.0;               /* large DM2 to DM2 spacing
        */
    DM2DM2 : double = 0.6;                  /* DM2 to existed dummy
       metal spacing */
```

Cuadro 54: Configuración de dimensiones de patrones de Metal 1 y 2 en el*runset* de relleno de metal

8.2.3. Compuerta NOT

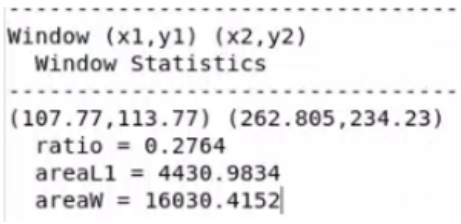
Este circuito fue el único que presentó un error de densidad en la capa de Metal 4 en lugar de en la capa de Metal 2, manteniendo el error en la capa de Metal 1.

8.2.3.1. Resultados iniciales



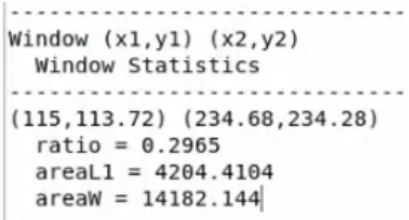
```
                LAYOUT ERRORS RESULTS: ERRORS

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

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

Library name:    LIB_TEST
Structure name:  Not_IO
Generated by:    IC Validator RHEL64 V-2023.12-SP1-1.9776368 2024/01/27
Runset name:     /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/Librerias/Runset/ICVLM18_LM16_LM152_6M.
215a_pre041518
User name:       nanoelectronica
Time started:    2024/08/13 08:09:27PM
Time ended:      2024/08/13 08:09:33PM

Called as: icv -icc2 -f NDM -i LIB_TEST -p /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
sintesis_fisica_NOT_STEPS -c Not_IO -clf /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
sintesis_fisica_NOT_STEPS/Outputs/DRC/slnFile.txt -icc_density_blockage -icc2_error_categories -hm -I /home/
nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_NOT_STEPS/Outputs/DRC -
icc2_error_browser INST -icc2_error_cell signoff_check_drc.err -rc /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
TSMC/nueva_integracion/sintesis_fisica_NOT_STEPS/Outputs/DRC/signoff_check_drc.rc -I /home/nanoelectronica/Desktop/
Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_NOT_STEPS/Outputs/DRC /home/nanoelectronica/Desktop/
Folder_de_Trabajo/Rui/TSMC/Librerias/Runset/ICVLM18_LM16_LM152_6M.215a_pre041518
CLF: -sln "16 18 28 31 33 38 17 27 29 32 39 "

                     ERROR SUMMARY

 M1.R.1 : Min M1 area coverage < 30%
   density ..................................... 1 violation found.

 M4.R.1 : Min M4 area coverage < 30%
   density ..................................... 1 violation found.
```

Figura 30: Resultado inicial de la verificación DRC para un circuito NOT



```
-----------------------------------
Window (x1,y1) (x2,y2)
  Window Statistics
-----------------------------------
(107.77,113.77) (262.805,234.23)
  ratio = 0.2764
  areaL1 = 4430.9834
  areaW = 16030.4152
```

Figura 31: Porcentaje de metal 1 para el circuito NOT



```
-------------------------------
Window (x1,y1) (x2,y2)
  Window Statistics
-------------------------------
(115,113.72) (234.68,234.28)
  ratio = 0.2965
  areaL1 = 4204.4104
  areaW = 14182.144
```

Figura 32: Porcentaje de metal 4 para el circuito NOT

8.2.3.2. Modificaciones en el *Metal Fill Runset*

```
    DMW_M1  :  double  =  0.8;                  /*  small  DM1  width     */
```

```
    DML_M1 : double = 0.8;                  /* small DM1 length    */
    DMS_M1 : double = 0.5;                  /* small DM1 spacing     */
    DM2M1 : double = 1.2;                   /* small DM1 to real metal
       spacing    */
    MDMW_M1 : double = 1.1;                 /* middle DM1 width     */
    MDML_M1 : double = 1.1;                 /* middle DM1 length    */
    MDMS_M1 : double = 1.0;                 /* middle DM1 spacing     */
```

Cuadro 55: Modificaciones en el *runset* de relleno de metal para el circuito NOT en M1

```
    #else
          DMW_M4 : double = 0.9;        /* small DM4 width     */
          DML_M4 : double = 0.9;        /* small DM4 length    */
```

Cuadro 56: Modificaciones en el *runset* de relleno de metal para el circuito NOT en M4

8.2.3.3. Resultados finales

En la Figura 33 se puede observar el resultado de la verificación DRC luego de aplicar el relleno de metal empleando el *runset* modificado, en la cual se cumple con todas las reglas de diseño y, por ende, el resultado es `CLEAN`.

Figura 33: Resultado de DRC sin errores para el circuito NOT

8.2.4. ALU de 4 *bits*

Este circuito y todos los demás presentaron errores de densidad en el Metal 1 y 2.

8.2.4.1. Resultados iniciales

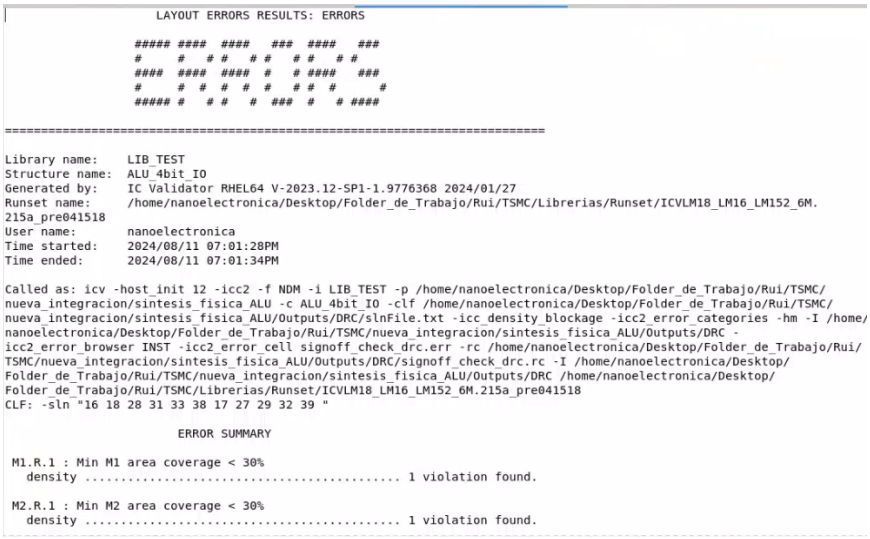


Figura 34: Resultado inicial de la verificación DRC para un circuito ALU de 4 *bits*

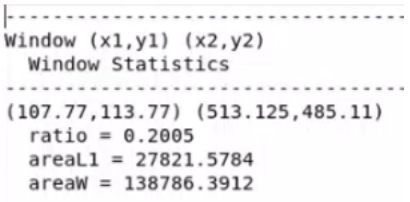


Figura 35: Porcentaje de metal 1 para el circuito ALU de 4 *bits*

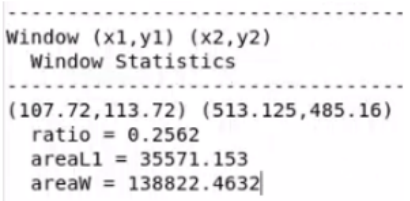


Figura 36: Porcentaje de metal 2 para el circuito ALU de 4 *bits*

8.2.4.2. Modificaciones en el *Metal Fill Runset*

```
    DMW_M1 : double = 1.1;                  /* small DM1 width     */
    DML_M1 : double = 1.1;                  /* small DM1 length    */
    DMS_M1 : double = 0.4;                  /* small DM1 spacing     */
    DM2M1 : double = 1.0;                   /* small DM1 to real metal
       spacing    */
    MDMW_M1 : double = 1.2;                 /* middle DM1 width     */
    MDML_M1 : double = 1.2;                 /* middle DM1 length    */
    MDMS_M1 : double = 0.8;                 /* middle DM1 spacing     */
```

```
MIND_M1 : double = 0.32;                /* M1 density for
    insertion */
```

Cuadro 57: Modificaciones en el *runset* de relleno de metal para el circuito ALU en M1

```
DMW_M2 : double = 1.1;                  /* small DM2 width     */
DML_M2 : double = 1.1;                  /* small DM2 length    */
DMS_M2 : double = 0.4;                  /* small DM2 spacing     */
DM2M2 : double = 1.2;                   /* small DM2 to real metal
    spacing    */
MDMW_M2 : double = 1.2;                 /* middle DM2 width     */
MDML_M2 : double = 1.2;                 /* middle DM2 length    */
MDMS_M2 : double = 0.8;                 /* middle DM2 spacing     */
```

Cuadro 58: Modificaciones en el *runset* de relleno de metal para el circuito ALU en M2

8.2.4.3. Resultados finales

En la Figura 37 se puede observar el resultado de la verificación DRC luego de aplicar el relleno de metal empleando el *runset* modificado, en la cual se cumple con todas las reglas de diseño y, por ende, el resultado es `CLEAN`.

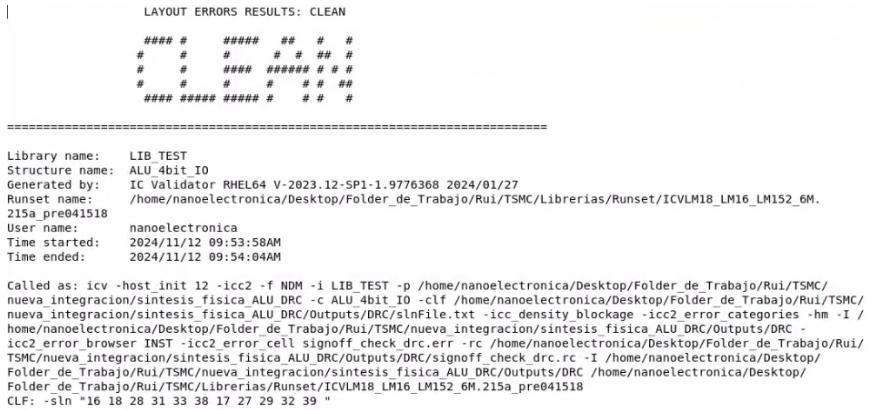
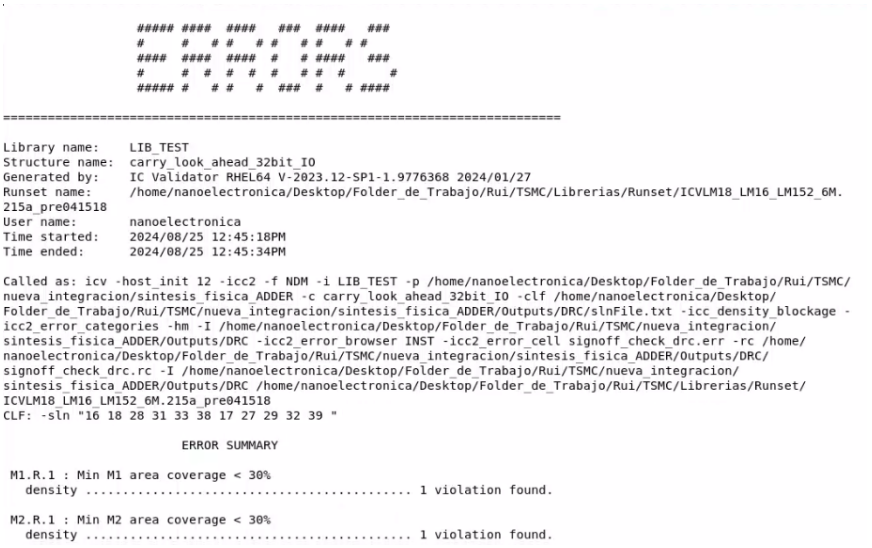


Figura 37: Resultado de DRC sin errores para el circuito ALU

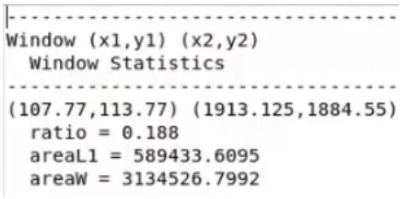
8.2.5. Circuito sumador de 32 *bits*

8.2.5.1. Resultados iniciales



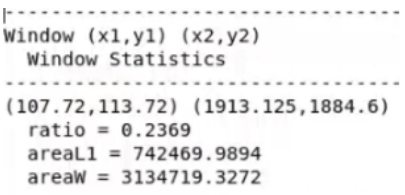
```
 ##### ####  ####   ###  ####   ###
 #    #   # #   # #   # #   # #
 ####  ####  ####  #   # ####   ###
 #    #  #  #  #  #   # #  #      #
 ##### #   # #   #  ### #   # ####

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

Library name:    LIB_TEST
Structure name:  carry_look_ahead_32bit_IO
Generated by:    IC Validator RHEL64 V-2023.12-SP1-1.9776368 2024/01/27
Runset name:     /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/Librerias/Runset/ICVLM18_LM16_LM152_6M.
215a_pre041518
User name:       nanoelectronica
Time started:    2024/08/25 12:45:18PM
Time ended:      2024/08/25 12:45:34PM

Called as: icv -host_init 12 -icc2 -f NDM -i LIB_TEST -p /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_ADDER -c carry_look_ahead_32bit_IO -clf /home/nanoelectronica/Desktop/
Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_ADDER/Outputs/DRC/slnFile.txt -icc_density_blockage -
icc2_error_categories -hm -I /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
sintesis_fisica_ADDER/Outputs/DRC -icc2_error_browser INST -icc2_error_cell signoff_check_drc.err -rc /home/
nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_ADDER/Outputs/DRC/
signoff_check_drc.rc -I /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
sintesis_fisica_ADDER/Outputs/DRC /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/Librerias/Runset/
ICVLM18_LM16_LM152_6M.215a_pre041518
CLF: -sln "16 18 28 31 33 38 17 27 29 32 39 "

                      ERROR SUMMARY

 M1.R.1 : Min M1 area coverage < 30%
   density ..................................... 1 violation found.

 M2.R.1 : Min M2 area coverage < 30%
   density ..................................... 1 violation found.
```

Figura 38: Resultado inicial de la verificación DRC para un circuito sumador



```
-----------------------------------
Window (x1,y1) (x2,y2)
  Window Statistics
-----------------------------------
(107.77,113.77) (1913.125,1884.55)
  ratio = 0.188
  areaL1 = 589433.6095
  areaW = 3134526.7992
```

Figura 39: Porcentaje de metal 1 para el circuito sumador



```
-----------------------------------
Window (x1,y1) (x2,y2)
  Window Statistics
-----------------------------------
(107.72,113.72) (1913.125,1884.6)
  ratio = 0.2369
  areaL1 = 742469.9894
  areaW = 3134719.3272
```

Figura 40: Porcentaje de metal 2 para el circuito sumador

8.2.5.2. Modificaciones en el *Metal Fill Runset*

Dado que el Porcentaje de metal en estas dos capas era inferior al de otros circuitos, las modificaciones necesarias fueron más significativas.

```
    DMW_M1  :  double  =  1.15;                /* small DM1 width     */
    DML_M1  :  double  =  1.15;                /* small DM1 length    */
```

```
DMS_M1 : double = 0.4;                  /* small DM1 spacing     */
DM2M1 : double = 1.0;                   /* small DM1 to real metal
   spacing    */
MDMW_M1 : double = 1.2;                 /* middle DM1 width      */
MDML_M1 : double = 1.2;                 /* middle DM1 length     */
```

Cuadro 59: Modificaciones en el *runset* de relleno de metal para el circuito sumador en M1

```
MDMS_M1 : double = 0.8;                 /* middle DM1 spacing     */
DMW_M2 : double = 1.1;                  /* small DM2 width     */
DML_M2 : double = 1.1;                  /* small DM2 length    */
DMS_M2 : double = 0.4;                  /* small DM2 spacing     */
DM2M2 : double = 1.2;                   /* small DM2 to real metal
   spacing    */
MDMW_M2 : double = 1.2;                 /* middle DM2 width      */
MDML_M2 : double = 1.2;                 /* middle DM2 length     */
```

Cuadro 60: Modificaciones en el *runset* de relleno de metal para el circuito sumador en M2

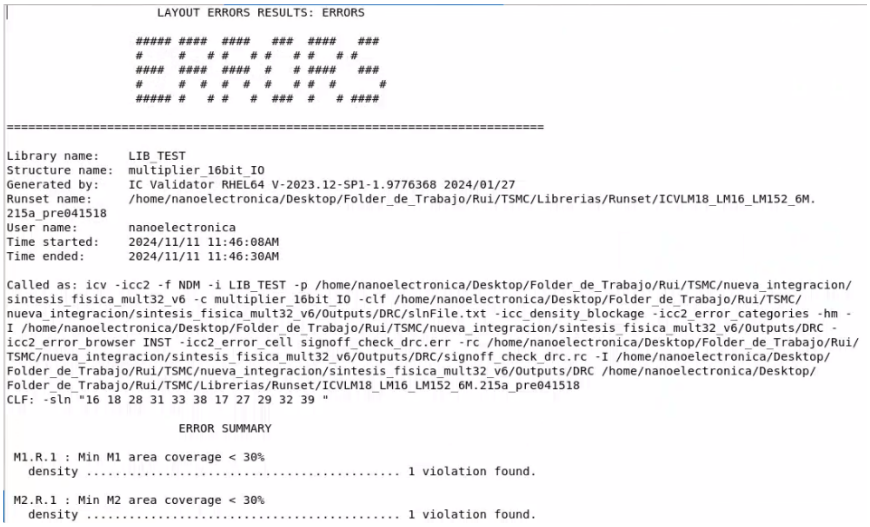
8.2.5.3. Resultados finales

En la Figura 41 se puede observar el resultado de la verificación DRC luego de aplicar el relleno de metal empleando el *runset* modificado, en la cual se cumple con todas las reglas de diseño y, por ende, el resultado es `CLEAN`.

Figura 41: Resultado de DRC sin errores para el circuito sumador

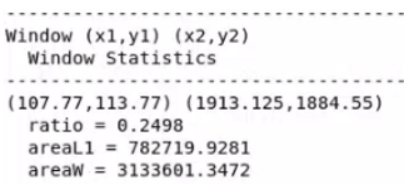
8.2.6. Circuito multiplicador de 32 *bits*

8.2.6.1. Resultados iniciales



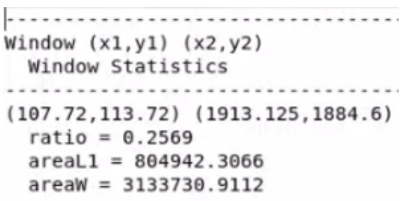
```
                    LAYOUT ERRORS RESULTS: ERRORS

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

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

Library name:    LIB_TEST
Structure name:  multiplier_16bit_IO
Generated by:    IC Validator RHEL64 V-2023.12-SP1-1.9776368 2024/01/27
Runset name:     /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/Librerias/Runset/ICVLM18_LM16_LM152_6M.
215a_pre041518
User name:       nanoelectronica
Time started:    2024/11/11 11:46:08AM
Time ended:      2024/11/11 11:46:30AM

Called as: icv -icc2 -f NDM -i LIB_TEST -p /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
sintesis_fisica_mult32_v6 -c multiplier_16bit_IO -clf /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_mult32_v6/Outputs/DRC/slnFile.txt -icc_density_blockage -icc2_error_categories -hm -
I /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_mult32_v6/Outputs/DRC -
icc2_error_browser INST -icc2_error_cell signoff_check_drc.err -rc /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
TSMC/nueva_integracion/sintesis_fisica_mult32_v6/Outputs/DRC/signoff_check_drc.rc -I /home/nanoelectronica/Desktop/
Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_mult32_v6/Outputs/DRC /home/nanoelectronica/Desktop/
Folder_de_Trabajo/Rui/TSMC/Librerias/Runset/ICVLM18_LM16_LM152_6M.215a_pre041518
CLF: -sln "16 18 28 31 33 38 17 27 29 32 39 "

                        ERROR SUMMARY

 M1.R.1 : Min M1 area coverage < 30%
   density ..................................... 1 violation found.

 M2.R.1 : Min M2 area coverage < 30%
   density ..................................... 1 violation found.
```

Figura 42: Resultado inicial de la verificación DRC para un circuito multiplicador



```
-------------------------------------
Window (x1,y1) (x2,y2)
  Window Statistics
-------------------------------------
(107.77,113.77) (1913.125,1884.55)
  ratio = 0.2498
  areaL1 = 782719.9281
  areaW = 3133601.3472
```

Figura 43: Porcentaje de metal 1 para el circuito multiplicador



```
-------------------------------------
Window (x1,y1) (x2,y2)
  Window Statistics
-------------------------------------
(107.72,113.72) (1913.125,1884.6)
  ratio = 0.2569
  areaL1 = 804942.3066
  areaW = 3133730.9112
```

Figura 44: Porcentaje de metal 2 para el circuito multiplicador

8.2.6.2. Modificaciones en el *Metal Fill Runset*

En este caso, las mismas modificaciones realizadas para el circuito ALU generaron un *layout* libre de errores de diseño.

8.2.6.3. Resultados finales

En la Figura 45 se puede observar el resultado de la verificación DRC luego de aplicar el relleno de metal empleando el *runset* modificado, en la cual se cumple con todas las reglas de diseño y, por ende, el resultado es `CLEAN`.

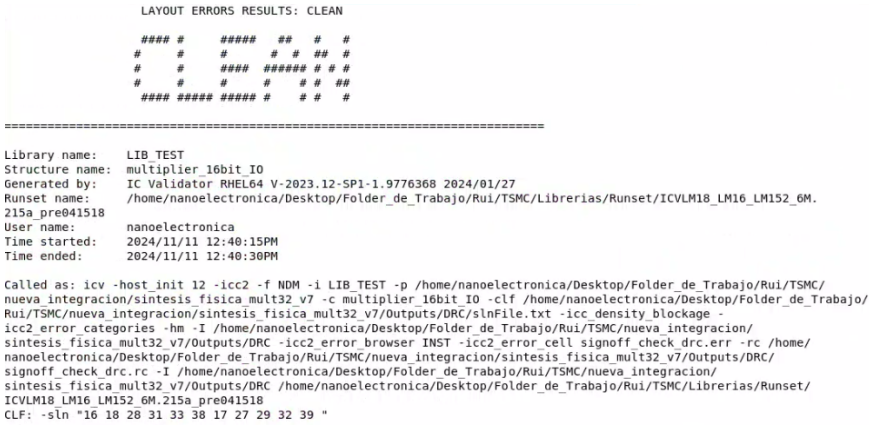


Figura 45: Resultado de DRC sin errores para el circuito multiplicador

8.2.7. Circuito divisor de 32 *bits*

8.2.7.1. Resultados iniciales

Figura 46: Resultado inicial de la verificación DRC para un circuito divisor

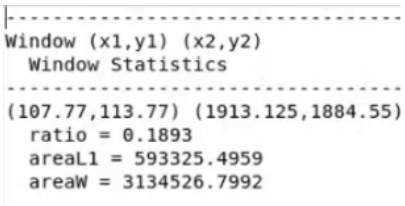


Figura 47: Porcentaje de metal 1 para el circuito divisor

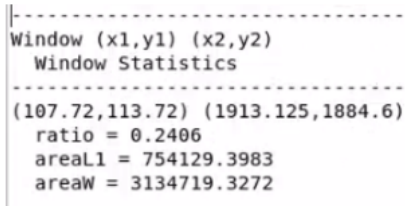


Figura 48: Porcentaje de metal 2 para el circuito divisor

8.2.7.2. Modificaciones en el *Metal Fill Runset*

En este caso, las mismas modificaciones realizadas para el circuito sumador generaron un *layout* libre de errores de diseño.

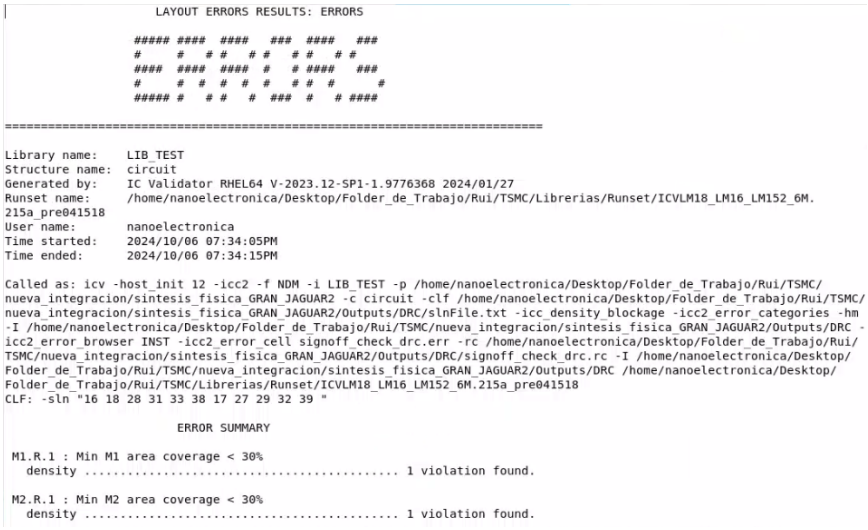
8.2.7.3. Resultados finales

En la Figura 49 se puede observar el resultado de la verificación DRC luego de aplicar el relleno de metal empleando el *runset* modificado, en la cual se cumple con todas las reglas de diseño y, por ende, el resultado es `CLEAN`.

Figura 49: Resultado de DRC sin errores para el circuito multiplicador

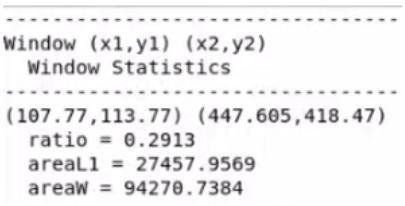
8.2.8. Circuito "El Gran Jaguar"

8.2.8.1. Resultados iniciales



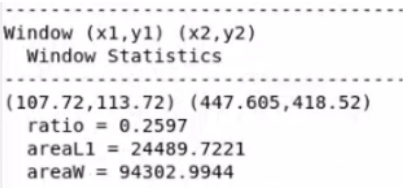
```
                LAYOUT ERRORS RESULTS: ERRORS

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

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

Library name:    LIB_TEST
Structure name:  circuit
Generated by:    IC Validator RHEL64 V-2023.12-SP1-1.9776368 2024/01/27
Runset name:     /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/Librerias/Runset/ICVLM18_LM16_LM152_6M.
215a_pre041518
User name:       nanoelectronica
Time started:    2024/10/06 07:34:05PM
Time ended:      2024/10/06 07:34:15PM

Called as: icv -host_init 12 -icc2 -f NDM -i LIB_TEST -p /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_GRAN_JAGUAR2 -c circuit -clf /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_GRAN_JAGUAR2/Outputs/DRC/slnFile.txt -icc_density_blockage -icc2_error_categories -hm
-I /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_GRAN_JAGUAR2/Outputs/DRC -
icc2_error_browser INST -icc2_error_cell signoff_check_drc.err -rc /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
TSMC/nueva_integracion/sintesis_fisica_GRAN_JAGUAR2/Outputs/DRC/signoff_check_drc.rc -I /home/nanoelectronica/Desktop/
Folder_de_Trabajo/Rui/TSMC/nueva_integracion/sintesis_fisica_GRAN_JAGUAR2/Outputs/DRC /home/nanoelectronica/Desktop/
Folder_de_Trabajo/Rui/TSMC/Librerias/Runset/ICVLM18_LM16_LM152_6M.215a_pre041518
CLF: -sln "16 18 28 31 33 38 17 27 29 32 39 "

                      ERROR SUMMARY

 M1.R.1 : Min M1 area coverage < 30%
   density ..................................... 1 violation found.

 M2.R.1 : Min M2 area coverage < 30%
   density ..................................... 1 violation found.
```

Figura 50: Resultado inicial de la verificación DRC para el circuito "El Gran Jaguar"



```
-----------------------------------
Window (x1,y1) (x2,y2)
  Window Statistics
-----------------------------------
(107.77,113.77) (447.605,418.47)
  ratio = 0.2913
  areaL1 = 27457.9569
  areaW = 94270.7384
```

Figura 51: Porcentaje de metal 1 para el circuito "El Gran Jaguar"



```
-----------------------------------
Window (x1,y1) (x2,y2)
  Window Statistics
-----------------------------------
(107.72,113.72) (447.605,418.52)
  ratio = 0.2597
  areaL1 = 24489.7221
  areaW = 94302.9944
```

Figura 52: Porcentaje de metal 2 para el circuito "El Gran Jaguar"

8.2.8.2. Modificaciones en el *Metal Fill Runset*

Dado que el Porcentaje de metal 1 se aproxima más a 0.3, se decidió aplicar las mismas modificaciones que se usaron en el circuito ALU para esa capa de metal.

Ahora bien, las modificaciones para M2 se muestran a continuación.

```
    DMW_M2 : double = 2.7;                  /* small DM2 width     */
    DML_M2 : double = 2.7;                  /* small DM2 length    */
```

```
    DMS_M2 : double = 0.25;                  /* small DM2 spacing     */
    DM2M2 : double = 1.0;                   /* small DM2 to real metal
       spacing    */
    MDMW_M2 : double = 2.9;                 /* middle DM2 width      */
    MDML_M2 : double = 2.9;                 /* middle DM2 length     */
    MDMS_M2 : double = 0.65;                 /* middle DM2 spacing
       */
    MDM2M2 : double = 1.0;                  /* middle DM2 to real metal
        spacing  */

    MIND_M2 : double = 0.38;                 /* M2 density for
       insertion */
```

Cuadro 61: Modificaciones en el *runset* de relleno de metal para el circuito "El Gran Jaguar" en M2

8.2.8.3. Resultados finales

Figura 53: Resultado final de la verificación DRC para un circuito "El Gran Jaguar"

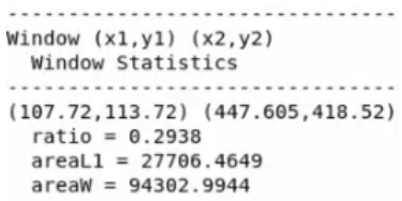


Figura 54: Porcentaje de metal 2 para el circuito "El Gran Jaguar" luego de las modificaciones

Como se puede observar, se logró corregir el error en la capa de Metal 1; sin embargo, el error en la capa de Metal 2 no se resolvió por completo, aunque se incrementó el porcentaje de densidad de 25.97 % a 29.38 %, quedando un 0.62 % por llenar. No se continuó modificando el *runset*, ya que al seguir aumentando las dimensiones de los patrones, los resultados para

la capa de Metal 2 empeoraban. Es importante señalar que sólo se ajustaron las dimensiones de los patrones de metal pequeños y medianos, ya que al modificar los patrones grandes, se observó un efecto adverso, disminuyendo aún más los porcentajes de densidad e incluso generando otros tipos de errores de diseño en algunos casos.

El error restante podría deberse a que la tecnología de 180 nm es relativamente antigua, habiéndose comercializado a finales de los años 90. Se espera que, con librerías de una tecnología más moderna, como la de 65 nm, estos *runsets* permitan que los circuitos cumplan con los requisitos de densidad en cada capa de metal.

8.3. Verificación de antena

La verificación de antenas se realiza sobre el archivo GDS generado al finalizar la síntesis física, ya que este contiene el diseño final que se enviará al *foundry* para su fabricación. Este proceso se lleva a cabo utilizando la herramienta *ICV*, mediante un *runset* que define todas las reglas a validar.

Dado que el diseño se ha realizado empleando las bibliotecas de *TSMC* para tecnología de 180 nm con 6 capas de metal, se debe utilizar el *runset* proporcionado por *TSMC*. En este caso, el archivo correspondiente es `ICVLM18_LM16_LM152_6M.ANT.215A_pre041518`.

La verificación no se realiza directamente con *ICC2*, sino que requiere abrir una nueva terminal e ingresar el siguiente comando:

```
icv -i <ruta-completa-archivo.gds> -c <nombre-top-cell> -sf ICV
    -vue <ruta-completa-runset>
```

Cuadro 62: Verificación de antenas con ICV

Este comando invoca a *ICV*, especificando que debe realizar la verificación según las reglas definidas en el *runset* sobre la *top-cell* del diseño contenido en el archivo `GDS`.

Para confirmar que el diseño ha pasado la verificación, es necesario revisar los archivos generados:

- `<nombre-top-cell>.RESULTS`: Este archivo detalla las reglas evaluadas. Si hay errores, también se enumeran en este archivo.
- `<nombre-top-cell>.LAYOUT_ERRORS`: Si el diseño cumple con todas las reglas de antenas, este archivo mostrará el estado `CLEAN` al inicio.

La verificación de antena no modifica ni genera un nuevo archivo *GDS*, ya que su propósito es únicamente evaluar si el diseño cumple con las reglas definidas en el *runset*. Sin embargo, si se detectan errores durante esta prueba, será necesario realizar modificaciones en el diseño. Para solucionar estos problemas, se pueden emplear técnicas como la conexión de los *tracks* a otras capas de metal o la inserción de diodos. Por lo tanto, sería recomendable investigar las opciones que proporciona la herramienta *ICC2* para implementar estas soluciones, ya sea ajustando el enrutamiento o automatizando la inserción de diodos.

A continuación se muestran los resultados de las verificaciones de antena realizadas a múltiples circuitos, donde se obtuvo un resultado exitoso en todos los casos.

8.3.1. Compuerta NOT

```
                LAYOUT ERRORS RESULTS: CLEAN

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

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

Library name:    /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/IO/chip.gds
Structure name:  Not_IO
Generated by:    IC Validator RHEL64 V-2023.12-SP1-1.9776368 2024/01/27
Runset name:     /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_NOT_MODIFIED/ICVLM18_LM16_LM152_6M.ANT.
215a_pre041518
User name:       nanoelectronica
Time started:    2024/10/16 12:18:00PM
Time ended:      2024/10/16 12:18:33PM

Called as: icv -i /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/IO/chip.gds -c Not_IO -sf
ICV -vue /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
sintesis_fisica_NOT_MODIFIED/ICVLM18_LM16_LM152_6M.ANT.215a_pre041518
```

Figura 55: Verificación de antena sobre el diseño de una compuerta NOT

8.3.2. ALU de 4 *bits*

```
                LAYOUT ERRORS RESULTS: CLEAN

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

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

Library name:    /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_ALU/Outputs/IO/chip.gds
Structure name:  ALU_4bit_IO
Generated by:    IC Validator RHEL64 V-2023.12-SP1-1.9776368 2024/01/27
Runset name:     /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_ALU/Outputs/Antena/ICVLM18_LM16_LM152_6M.ANT.
215a_pre041518
User name:       nanoelectronica
Time started:    2024/10/16 12:37:42PM
Time ended:      2024/10/16 12:37:46PM

Called as: icv -i /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_ALU/Outputs/IO/chip.gds -c ALU_4bit_IO -sf ICV -
vue /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
sintesis_fisica_ALU/Outputs/Antena/ICVLM18_LM16_LM152_6M.ANT.215a_pre041518
```

Figura 56: Verificación de antena sobre el diseño de un circuito ALU de 4 *bits*

8.3.3. Circuito sumador de 32 *bits*

Figura 57: Verificación de antena sobre el diseño de un sumador de 32 *bits*

8.3.4. Circuito multiplicador de 32 *bits*

Figura 58: Verificación de antena sobre el diseño de un multiplicador de 32 *bits*

8.3.5. Circuito divisor de 32 *bits*

```
                  LAYOUT ERRORS RESULTS: CLEAN

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

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

Library name:    /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_div32/Outputs/IO/chip.gds
Structure name:  divisor_16bit_IO
Generated by:    IC Validator RHEL64 V-2023.12-SP1-1.9776368 2024/01/27
Runset name:     /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_div32/Outputs/Antena/ICVLM18_LM16_LM152_6M.ANT.
215a_pre041518
User name:       nanoelectronica
Time started:    2024/10/21 01:08:27PM
Time ended:      2024/10/21 01:08:38PM

Called as: icv -i /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_div32/Outputs/IO/chip.gds -c divisor_16bit_IO -sf
ICV -vue /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
sintesis_fisica_div32/Outputs/Antena/ICVLM18_LM16_LM152_6M.ANT.215a_pre041518
```

Figura 59: Verificación de antena sobre el diseño de un divisor de 32 *bits*

8.3.6. Circuito "El Gran Jaguar"

```
                  LAYOUT ERRORS RESULTS: CLEAN

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

============•===========================================================

Library name:    /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_GRAN_JAGUAR/Outputs/IO/chip.gds
Structure name:  circuit
Generated by:    IC Validator RHEL64 V-2023.12-SP1-1.9776368 2024/01/27
Runset name:     /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_GRAN_JAGUAR/Outputs/Antena/
ICVLM18_LM16_LM152_6M.ANT.215a_pre041518
User name:       nanoelectronica
Time started:    2024/10/21 12:57:25PM
Time ended:      2024/10/21 12:57:28PM

Called as: icv -i /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
nueva_integracion/sintesis_fisica_GRAN_JAGUAR/Outputs/IO/chip.gds -c circuit -sf
ICV -vue /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
sintesis_fisica_GRAN_JAGUAR/Outputs/Antena/ICVLM18_LM16_LM152_6M.ANT.215a_pre041518
```

Figura 60: Verificación de antena sobre el diseño del circuito "El Gran Jaguar"

CAPÍTULO 9

Verificación de funcionamiento por medio de Verilog

El comando permite exportar una descripción jerárquica completa del diseño en formato Verilog, representando todas las instancias de celdas y las conexiones entre módulos. El comando `write_verilog` se ejecuta en el contexto del diseño físico, asegurándose de que este esté completamente vinculado dentro de *ICC2*. En caso de que el diseño no esté vinculado, la herramienta realiza un enlace automático para garantizar que todos los módulos y celdas necesarios se incluyan correctamente en el archivo generado. Esto garantiza que el Verilog del diseño sintetizado esté completamente alineado con el estado físico del diseño después de la etapa de *placement* y ruteo.

Este archivo es el que se utiliza como una entrada para la elaboración de la verificación *LVS* en *IC Validator*, proceso que se explica a detalle en el trabajo [33]. El Verilog obtenido contiene las instancias de componentes puramente lógicos, lógicos y físicos, y puramente físicos. En el archivo Verilog resultante de la síntesis lógica, se describe únicamente la lógica funcional del circuito. Este archivo incluye instancias de celdas estándar como la compuerta `NOT (CKND0BWP7T)`, en este caso, y las celdas de I/O (`PDDW0204SCDG`). Sin embargo, no incluye conexiones a las redes de voltaje y tierra, ni celdas adicionales necesarias para garantizar la manufacturabilidad.

Por otro lado, el archivo Verilog generado tras la síntesis física contiene conexiones explícitas a las redes `VDD` y `VSS`, que son definidas como `supply1` y `supply0`, respectivamente. Además, incluye celdas de soporte, como `TIELBWP7T` y `TIEHBWP7T`, utilizadas para establecer valores lógicos constantes en diferentes partes del diseño (1 o 0 lógico), así como celdas de relleno (`PFILLER`) y celdas de esquinas (`PCORNER`), las cuales definen los límites físicos del diseño.

```
////////////////////////////////////////////////////////////
// Created by: Synopsys DC Expert(TM) in wire load mode
// Version   : V-2023.12-SP1
// Date      : Fri Aug  9 21:51:16 2024
```

```
////////////////////////////////////////////////////////////////


module Not ( A, Y );
  input A;
  output Y;


  CKND0BWP7T U1 ( .I(A), .ZN(Y) );
endmodule


module Not_IO ( A, Y );
  input A;
  output Y;
  wire   A_w, Y_w, n1, n2;
  tri   A;
  tri   Y;

  Not Compuerta ( .A(A_w), .Y(Y_w) );
  PDDW0204SCDG U10 ( .I(n1), .OEN(n2), .IE(n2), .PAD(A), .DS(n1)
      , .PE(n1), .C(
        A_w) );
  PDDW0204SCDG U11 ( .I(Y_w), .OEN(n1), .IE(n2), .PAD(Y), .DS(n1
     ), .PE(n1) );
  TIELBWP7T U6 ( .ZN(n1) );
  TIEHBWP7T U7 ( .Z(n2) );
endmodule
```

Cuadro 63: Verilog obtenido luego de síntesis lógica para una compuerta NOT

```
// IC Compiler II Version V-2023.12 Verilog Writer
// Generated on 11/14/2024 at 1:0:43
// Library Name: LIB_TEST
// Block Name: Not_IO
// User Label:
// Write Command: write_verilog -include { all } ./Outputs/IO/
   VERILOG.v
module Not ( A , Y , VDD , VSS ) ;
input  A ;
output Y ;
input  VDD ;
input  VSS ;

supply1 VDD ;
supply0 VSS ;

CKND0BWP7T U1 ( .I ( A ) , .ZN ( Y ) , .VDD ( VDD ) , .VSS ( VSS
    ) ) ;
endmodule


module Not_IO ( A , Y , VDD , VSS ) ;
input  A ;
```

```
    output Y ;
    input  VDD ;
    input  VSS ;

    wire A_w ;
    wire Y_w ;
    wire n1 ;
    wire n2 ;
    supply1 VDD ;
    supply0 VSS ;
    wire SYNOPSYS_UNCONNECTED_1 ;

    Not Compuerta ( .A ( A_w ) , .Y ( Y_w ) , .VDD ( VDD ) , .VSS (
       VSS ) ) ;
    PDDW0204SCDG U10 ( .I ( n1 ) , .OEN ( n2 ) , .IE ( n2 ) , .PAD (
        A ) ,
        .DS ( n1 ) , .PE ( n1 ) , .C ( A_w ) ) ;
    PDDW0204SCDG U11 ( .I ( Y_w ) , .OEN ( n1 ) , .IE ( n2 ) , .PAD
       ( Y ) ,
        .DS ( n1 ) , .PE ( n1 ) , .C ( SYNOPSYS_UNCONNECTED_1 ) ) ;
    TIELBWP7T U6 ( .ZN ( n1 ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
    TIEHBWP7T U7 ( .Z ( n2 ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
    PCORNER CORNER1 () ;
    PCORNER CORNER2 () ;
    PCORNER CORNER3 () ;
    PCORNER CORNER4 () ;
    PVDD1CDG PVDD1 ( .VDD ( VDD ) ) ;
    PVSS1CDG PVSS1 ( .VSS ( VSS ) ) ;
    PFILLER20 __added_filler_instance_0 () ;
    PFILLER5 __added_filler_instance_1 () ;
    PFILLER1 __added_filler_instance_2 () ;
    PFILLER1 __added_filler_instance_3 () ;
    PFILLER1 __added_filler_instance_4 () ;
    PFILLER1 __added_filler_instance_5 () ;
    PFILLER0005 __added_filler_instance_6 () ;
    PFILLER0005 __added_filler_instance_7 () ;
    PFILLER0005 __added_filler_instance_8 () ;
    PFILLER0005 __added_filler_instance_9 () ;
    .
    .
    .
    endmodule
```

Cuadro 64: Verilog obtenido luego de síntesis física para una compuerta NOT

Al observar cambios significativos en los resultados obtenidos tras la síntesis física, se evaluó la posibilidad de generar *test benches* para los archivos Verilog resultantes con el objetivo de realizar simulaciones funcionales. Estas simulaciones permitirían validar el comportamiento del diseño en su forma física antes de proceder con las verificaciones finales en *HSPICE*. Este paso se planteó como una validación intermedia para confirmar que los cambios implementados en el *layout* se reflejan correctamente en el comportamiento funcional del diseño.

Para llevar a cabo estas simulaciones, se importaron los siguientes archivos clave:

- `tcb018gbwp7t.v`: Contiene las celdas estándar de la tecnología *TSMC* de 180 nm, utilizadas para construir la lógica principal del diseño.
- `tpd018nv.v`: Contiene las celdas estándar correspondientes a los pines de entrada y salida, también de la tecnología *TSMC* de 180 nm.

A continuación, se presentan los *test benches* generados y los resultados obtenidos para algunos de los circuitos diseñados.

9.1. Compuerta NOT

```
//‘timescale 1ns/1ps
‘include "noFill.v"
//‘include "tpd018nv.v"
//‘include "tcb018gbwp7t.v"

module Not_IO_tb;

    reg A;
    wire Y;
    reg VDD = 1’b1;
    reg VSS = 1’b0;

    Not_IO uut (
        .A(A),
        .Y(Y),
        .VDD(VDD),
        .VSS(VSS)
    );


    initial begin

    $fsdbDumpfile("Not_IO_tb.fsdb");
    $fsdbDumpvars(0, Not_IO_tb);
        $dumpfile("Not_IO_tb.vcd");
        $dumpvars(0, Not_IO_tb);


        A = 1’b0;

        #1;
        A = 1’b1;


        #1;
        A = 1’b0;

    #1;
```

```
            A = 1'b1;


        end


        initial begin
            $monitor("At time %tns: A = %b, Y = %b", $time, A, Y);
        end

    endmodule
```

Cuadro 65: *Testbench* realizado para simular el comportamiento de una compuerta NOT

9.1.1. Resultados

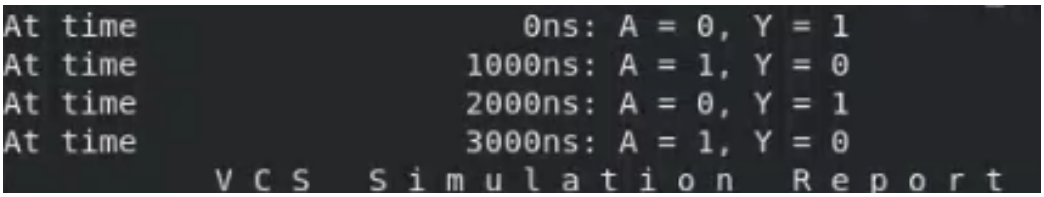


Figura 61: Resultados del *testbench* para una compuerta NOT en la terminal

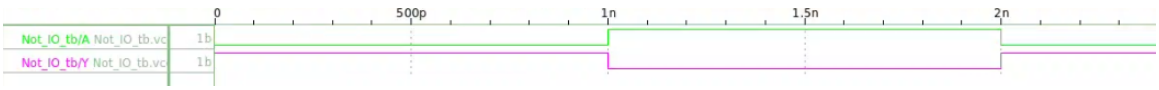


Figura 62: Resultados del *testbench* para una compuerta NOT en *WaveView*

9.2. ALU de 4 *bits*

```
    `timescale 1ns / 1ps
    `include "VERILOG.v"

    module tb_ALU_4bit_IO;

    // Inputs
    reg [3:0] A;
    reg [3:0] B;
    reg [2:0] ALU_Sel;
    reg VDD;
    reg VSS;

    wire [3:0] ALU_Out;
    wire Carry_Out;

    ALU_4bit_IO uut (
        .A(A),
```

```
        .B(B),
        .ALU_Sel(ALU_Sel),
        .ALU_Out(ALU_Out),
        .Carry_Out(Carry_Out),
        .VDD(VDD),
        .VSS(VSS)
    );

    initial begin
        $fsdbDumpfile("alu.fsdb");
            $fsdbDumpvars(0, tb_ALU_4bit_IO);
            $dumpfile("alu.vcd");
            $dumpvars(0, tb_ALU_4bit_IO);

        A = 0;
        B = 0;
        ALU_Sel = 0;
        VDD = 1;
        VSS = 0;


        #100;


        // Test Case 1: Addition
        A = 4'b0001;  // 1
        B = 4'b0010;  // 2
        ALU_Sel = 3'b000;  //000 selects ADD operation
        #10;
        $display("Time: %0t | A: %b | B: %b | ALU_Sel: %b | ALU_Out:
            %b | Carry_Out: %b",
                $time, A, B, ALU_Sel, ALU_Out, Carry_Out);

        // Test Case 2: Subtraction
        A = 4'b0110;  // 6
        B = 4'b0011;  // 3
        ALU_Sel = 3'b001;  // 001 selects SUB operation
        #10;
        $display("Time: %0t | A: %b | B: %b | ALU_Sel: %b | ALU_Out:
            %b | Carry_Out: %b",
                $time, A, B, ALU_Sel, ALU_Out, Carry_Out);

        // Test Case 3: AND
        A = 4'b1010;  // 10
        B = 4'b1100;  // 12
        ALU_Sel = 3'b010;  // 010 selects AND operation
        #10;
        $display("Time: %0t | A: %b | B: %b | ALU_Sel: %b | ALU_Out:
            %b | Carry_Out: %b",
                $time, A, B, ALU_Sel, ALU_Out, Carry_Out);

        // Test Case 4: OR
        A = 4'b0101;  // 5
        B = 4'b0011;  // 3
```

```
        ALU_Sel = 3'b011;  //011 selects OR operation
        #10;
        $display("Time: %0t | A: %b | B: %b | ALU_Sel: %b | ALU_Out:
            %b | Carry_Out: %b",
                $time, A, B, ALU_Sel, ALU_Out, Carry_Out);

        // Test Case 5: XOR
        A = 4'b1100;  // 12
        B = 4'b1010;  // 10
        ALU_Sel = 3'b100;  //100 selects XOR operation
        #10;
        $display("Time: %0t | A: %b | B: %b | ALU_Sel: %b | ALU_Out:
            %b | Carry_Out: %b",
                $time, A, B, ALU_Sel, ALU_Out, Carry_Out);


        $finish;
    end

    endmodule
```

Cuadro 66: *Testbench* realizado para simular el comportamiento de una ALU

Los casos evaluados para la ALU corresponden a las operaciones seleccionadas por *ALU_SEL*. Las configuraciones y sus respectivas funciones son las siguientes:

- `000`: Realiza una suma.
- `001`: Realiza una resta.
- `010`: Ejecuta una operación `AND`.
- `011`: Ejecuta una operación `OR`.
- `100`: Ejecuta una operación `XOR`.

9.2.1. Resultados

```
Time: 110000 | A: 0001 | B: 0010 | ALU_Sel: 000 | ALU_Out: 0011
Time: 120000 | A: 0110 | B: 0011 | ALU_Sel: 001 | ALU_Out: 0011
Time: 130000 | A: 1010 | B: 1100 | ALU_Sel: 010 | ALU_Out: 1000
Time: 140000 | A: 0101 | B: 0011 | ALU_Sel: 011 | ALU_Out: 0111
Time: 150000 | A: 1100 | B: 1010 | ALU_Sel: 100 | ALU_Out: 0110
```

Figura 63: Resultados del *testbench* para una ALU en la terminal

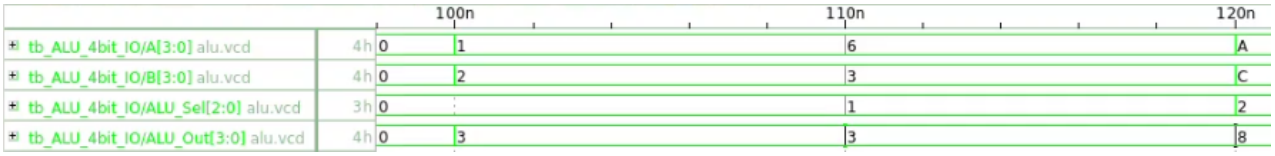


Figura 64: Resultados del *testbench* para una ALU en *WaveView*

9.3. Circuito sumador de 32 *bits*

```
`timescale 1ns/1ps
`include "noFill.v"

module adder32bit;

  // Inputs
  reg [31:0] a;
  reg [31:0] b;
  reg cin;
  reg VDD;
  reg VSS;

  // Outputs
  wire [31:0] S;
  wire cout;

  // Instantiate the Unit Under Test (UUT)
  carry_look_ahead_32bit_IO uut (
    .a(a),
    .b(b),
    .cin(cin),
    .S(S),
    .cout(cout),
    .VDD(VDD),
    .VSS(VSS)
  );

  initial begin
    // Initialize Inputs
    $fsdbDumpfile("adder32b.fsdb");
    $fsdbDumpvars(0, adder32bit);
    $dumpfile("adder32b.vcd");
    $dumpvars(0, adder32bit);

    a = 0;
    b = 0;
    cin = 0;
    VDD = 1;
    VSS = 0;

    /// Test Case 1: 5 + 10 + 0
    a = 32'd5;
    b = 32'd10;
```

```
        cin = 0;
        #100; // Wait 10 ns
        $display("Test Case 1: a=%d, b=%d, cin=%b, S=%d, a, b, cin,
            S);

        // Test Case 2: 12345 + 54321 + 1
        a = 32'd12345;
        b = 32'd54321;
        cin = 1;
        #100;
        $display("Test Case 2: a=%d, b=%d, cin=%b, S=%d, a, b, cin,
            S);

        // Test Case 3: 100000 + 100000 + 0
        a = 32'd100000;
        b = 32'd100000;
        cin = 0;
        #100;
        $display("Test Case 3: a=%d, b=%d, cin=%b, S=%d, a, b, cin,
            S);

        // Test Case 4: 1 + 2 + 1
        a = 32'd1;
        b = 32'd2;
        cin = 1;
        #100;
        $display("Test Case 4: a=%d, b=%d, cin=%b, S=%d, a, b, cin,
            S);

        // Test Case 5: 2147483647 (max 32-bit signed int) + 1 + 0
        a = 32'd2147483647;
        b = 32'd1;
        cin = 0;
        #100;
        $display("Test Case 5: a=%d, b=%d, cin=%b, S=%d, a, b, cin,
            S);

        // Test Case 6: 4294967295 + 0 + 1
        a = 32'd4294967295;
        b = 32'd0;
        cin = 1;
        #100;
        $display("Test Case 6: a=%d, b=%d, cin=%b, S=%d, a, b, cin,
            S);

        // Test Case 7: 123456789 + 987654321 + 0
        a = 32'd123456789;
        b = 32'd987654321;
        cin = 0;
        #100;
        $display("Test Case 7: a=%d, b=%d, cin=%b, S=%d, a, b, cin,
            S);

        // Test Case 8: 3000000000 + 1000000000 + 1
```

```
        a = 32'd3000000000;
        b = 32'd1000000000;
        cin = 1;
        #100;
        $display("Test Case 8: a=%d, b=%d, cin=%b, S=%d, a, b, cin,
            S);
        $finish;
    end

endmodule
```

Cuadro 67: *Testbench* realizado para simular el comportamiento de un sumador

9.3.1. Resultados

```
Test Case 1: a=          5, b=         10, cin=0, S=         15,
Test Case 2: a=      12345, b=      54321, cin=1, S=      66666,
Test Case 3: a=     100000, b=     100000, cin=0, S=     200000,
Test Case 4: a=          1, b=          2, cin=1, S=          3,
Test Case 5: a=2147483647, b=          1, cin=0, S=2147483648,
Test Case 6: a=4294967295, b=          0, cin=1, S=4294967295,
Test Case 7: a= 123456789, b= 987654321, cin=0, S=1111111110,
Test Case 8: a=3000000000, b=1000000000, cin=1, S=4000000000,
```

Figura 65: Resultados del *testbench* para un sumador de 32 *bits* en la terminal

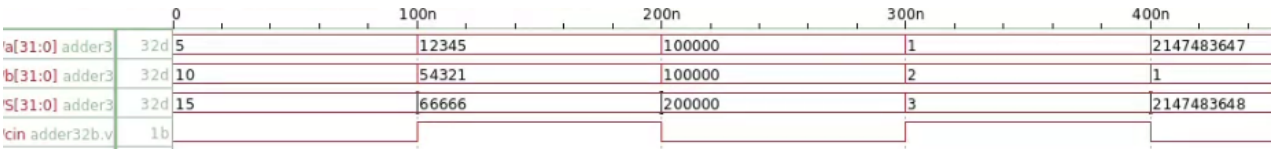


Figura 66: Resultados del *testbench* para un sumador de 32 *bits* en *WaveView*

9.4. Circuito "El Gran Jaguar"

```
`timescale 1ns/1ps
`include "noFill.v"

module nanochip_tb;

reg reset;
reg clk;
reg [1:0] select;
reg EN;
wire [7:0] q_out;
wire clk_s;
reg VDD;
reg VSS;
```

```
    chip_SP uut (
        .reset(reset),
        .clk(clk),
        .select(select),
        .EN(EN),
        .q_out(q_out),
        .clk_s(clk_s),
        .VDD(VDD),
        .VSS(VSS)
    );

    initial
    clk = 1'b0;
    always #5 clk = ~clk;

    initial begin

        clk = 0;
        VDD = 1;
        VSS = 0;

        $fsdbDumpfile("nanochip.fsdb");
        $fsdbDumpvars(0, nanochip_tb);
        $dumpfile("nanochip.vcd");
        $dumpvars(0, nanochip_tb);
        EN = 1'b0;
        reset = 1'b1;
        select = 2'b10;
        #15 reset =1'b0;
        #15 reset =1'b1;
        #15 reset =1'b0;

        #16000 $finish;

    end

    endmodule
```

Cuadro 68: *Testbench* realizado para simular el comportamiento de el circuito "El Gran Jaguar"

9.4.1. Resultados

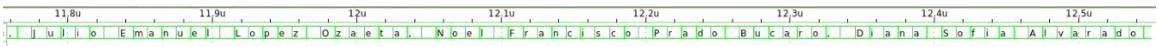


Figura 67: Resultados del *testbench* para circuito "El Gran Jaguar" en la terminal

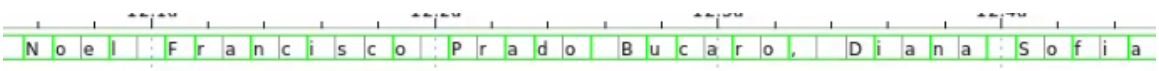


Figura 68: Acercamiento de los resultados del *testbench* para circuito "El Gran Jaguar" en la terminal

9.5. Resultados generales

Como se puede observar, todos los circuitos generaron los resultados esperados durante las simulaciones funcionales. Es importante destacar que, para estas pruebas, fue necesario remover las instancias de celdas puramente físicas, como las `PCORNERS` y `FILLERS`, ya que estas no tienen una representación en Verilog y no contribuyen a la funcionalidad lógica del circuito.

Adicionalmente, se realizó un segundo conjunto de pruebas en el que se introdujeron modificaciones intencionales en el *layout*, como la desconexión de ciertos componentes. Posteriormente, se generó nuevamente el archivo Verilog del diseño modificado y se observó que este no reflejaba los cambios realizados en el *layout*. Esto evidenció que la generación del Verilog a partir del diseño físico solo añade las instancias relacionadas con elementos físicos, como las celdas de `VDD` y `VSS`, así como algunas celdas auxiliares, pero no captura de manera completa el estado real del *layout*.

El archivo Verilog generado tras la síntesis física no es una representación fiel del *layout* físico, ya que no refleja modificaciones realizadas directamente en la disposición física de los componentes. Este archivo sirve principalmente como un *netlist* con elementos físicos para garantizar la simulación funcional, pero no proporciona una descripción exhaustiva del diseño a nivel físico. Por lo tanto, aunque estas simulaciones funcionales son útiles, no sustituyen las verificaciones más detalladas que se realizan directamente sobre el *layout* en herramientas específicas como *HSPICE*.

9.6. Comandos

```
vcs -v <design_file>.v <testbench_file>.v -o simulation_name -
   full64 -debug_access+all
```

Cuadro 69: Comando para compilar y simular un diseño descrito en Verilog utilizando VCS

Luego de ejecutar el comando mencionado anteriormente, se genera un archivo simulador con el nombre que se le haya asignado. Este se debe de ejecutar para que se genere el archivo `.vcd`, ya que se utilizará para verificar el comportamiento del circuito.

```
./simulation_name
```

Cuadro 70: Comando para ejecutar la simulación

```
wv -i <vcd_file>.vcd
```

Cuadro 71: Comando para visualizar el comportamiento de un circuito en *WaveView*

CAPÍTULO 10

Creación de librerías con *IC Compiler II Library Manager Tool*

En el siguiente diagrama se muestra la jerarquía de directorios desarrollada para la creación de librerías con el *IC Compiler II Library Manager Tool*.

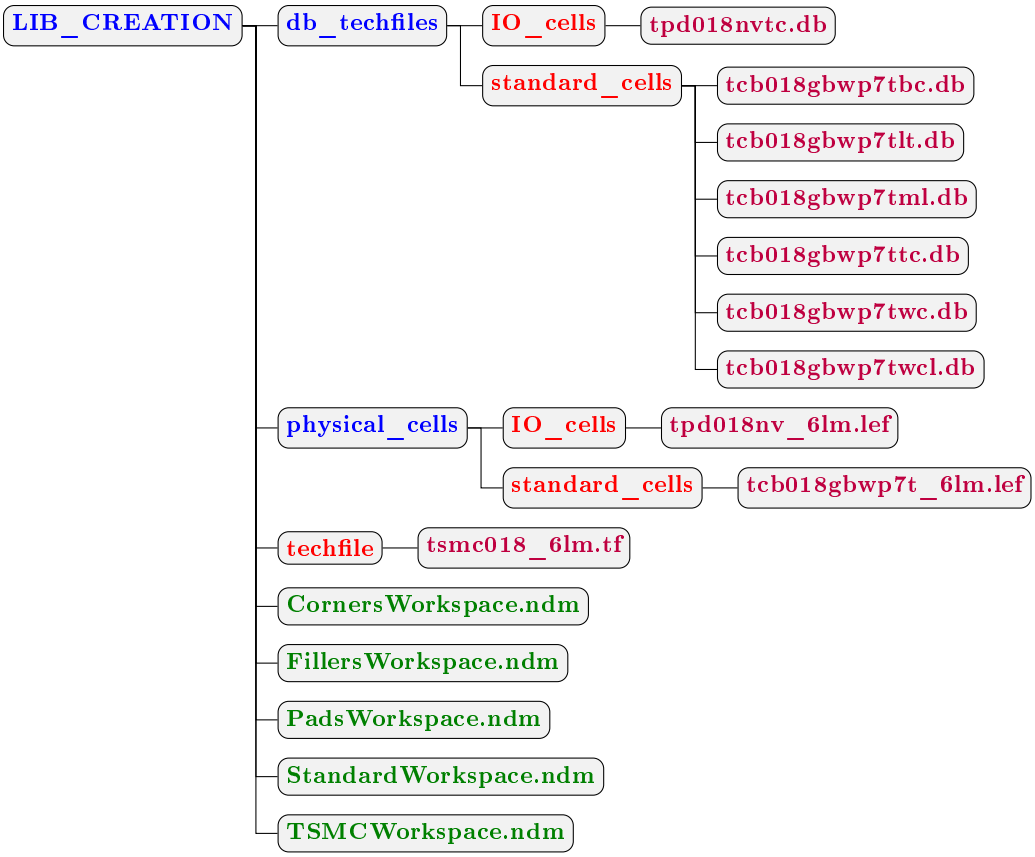


Figura 69: Jerarquia de directorios desarrollada para la creación de librerías

10.1. Archivos esenciales proporcionados por TSMC

La herramienta *IC Compiler II* (ICCII) utiliza librerías denominadas *New Data Model* (NDM), cuya extensión es `.ndm`. Estas librerías NDM son un conjunto que contiene toda la información necesaria sobre las celdas utilizadas en un diseño. Estas librerías pueden organizarse tanto como colecciones de referencia para el diseño lógico, como librerías completas que incorporan toda la información relacionada con la síntesis física.

En este proyecto, TSMC proporciona tres tipos principales de archivos esenciales para las librerías de referencia. Estos archivos son los siguientes:

Technology File

Contiene las reglas y restricciones específicas de diseño para la tecnología utilizada, incluyendo detalles sobre las capas de metal, dimensiones y reglas de espaciado.

- **Nombre del archivo:** `tsmc018_6lm.tf`
- **Ubicación del archivo:**

```
TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/
tcb018gbwp7t_270a_apf/TSMCHOME/digital/Back_End/milkyway/
tcb018gbwp7t_270a/techfiles
```

DB-Technology File

Este archivo contiene información lógica sobre las celdas estándar y los pads de I/O utilizados en la síntesis física. Entre las características descritas se incluyen parámetros de *timing*, área, consumo de potencia, y voltajes operativos.

Celdas estándar

- **Nombres de los archivos:**
  - `tcb018gbwp7tbc.db`
  - `tcb018gbwp7tlt.db`
  - `tcb018gbwp7tml.db`
  - `tcb018gbwp7ttc.db`
  - `tcb018gbwp7twc.db`
  - `tcb018gbwp7twcl.db`
- **Ubicación de los archivos:**

```
TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/
tcb018gbwp7t_270a_nldm/TSMCHOME/digital/Front_End/
timing_power_noise/NLDM/tcb018gbwp7t_270a
```

Pads de entrada y salida

- **Nombre del archivo:** `tpd018nvtc.db`
- **Ubicación del archivo:**

```
TSMC/180/CMOS/G/IO3.3V/iolib/LINEAR/tpd018nv_280a_FE/
tpd018nv_280a_nldm/TSMCHOME/digital/Front_End/
timing_power_noise/NLDM/tpd018nv_280a
```

Library Exchange Format File (LEF)

Este archivo contiene información relacionada con la descripción física de las celdas estándar y macros, como dimensiones, pines de entrada y salida, y reglas de espaciado. Contienen todas las representaciones de las celdas y pads en silicio.

Celdas estándar

- **Nombre del archivo:** `tcb018gbwp7t_6lm.lef`
- **Ubicación del archivo:**

  `TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/tcb018gbwp7t_270a_sef/TSMCHOME/digital/Back_End/lef/tcb018gbwp7t_270a/lef`

Pads de entrada y salida

- **Nombre del archivo:** `tpd018nv_6lm.lef`
- **Ubicación del archivo:**

  `TSMC/180/CMOS/G/IO3.3V/iolib/LINEAR/tpd018nv_280a_FE/tpd018nv_280a_sefu6lm/TSMCHOME/digital/Back_End/lef/tpd018nv_280a/mt_2/6lm/lef`

10.2. Flujos en el *Library Manager*

La herramienta *Library Manager* (LM) permite trabajar con diferentes *flows*, que definen cómo procesar las entradas para generar las librerías necesarias en la síntesis física. En este trabajo, se emplearon tres *flows* principales:

Normal Flow

Este *flow* estándar toma como entrada los archivos que describen las características físicas y lógicas de las celdas para generar una librería NDM. Es utilizado para celdas que tienen tanto una definición lógica como física, como compuertas.

Physical-only Flow

Este *flow* genera una librería NDM que contiene exclusivamente las características físicas de las celdas, omitiendo las definiciones lógicas. Esto es útil para celdas como los *fillers*, que no poseen comportamiento lógico pero son esenciales para el diseño físico.

Aggregate Flow

Este *flow* permite combinar múltiples librerías NDM en una sola.

Aplicación de los *Flows*

Para las celdas estándar, fue necesario generar dos librerías separadas:

- Una con el *Normal Flow* para las compuertas lógicas.
- Otra con el *Physical-only Flow* para los *fillers*.

De manera similar, en el caso de las librerías de los pads de entrada/salida, se emplearon ambos *flows* debido a la presencia de elementos como *corners* y *fillers* necesarios para el anillo de IO.

Finalmente, el *Aggregate Flow* se utilizó para consolidar todas las librerías generadas en un único archivo NDM, que contenía tanto las definiciones lógicas como físicas de las 4 librerías generadas por separado.

10.3. Creación de librerías NDM en *Library Manager*

El *Library Manager* permite la creación de librerías mediante una interfaz gráfica, la cual se puede invocar utilizando el siguiente comando:

```
icc2_lm_shell -gui
```

Cuadro 72: Comando para invocar el *Library Manager*

Al ejecutar este comando, se abre una ventana como la que se muestra en la Figura 70. Para comenzar, es necesario hacer clic en *"Create a new library"*, lo que desplegará una nueva ventana similar a la que se observa en la Figura 71.

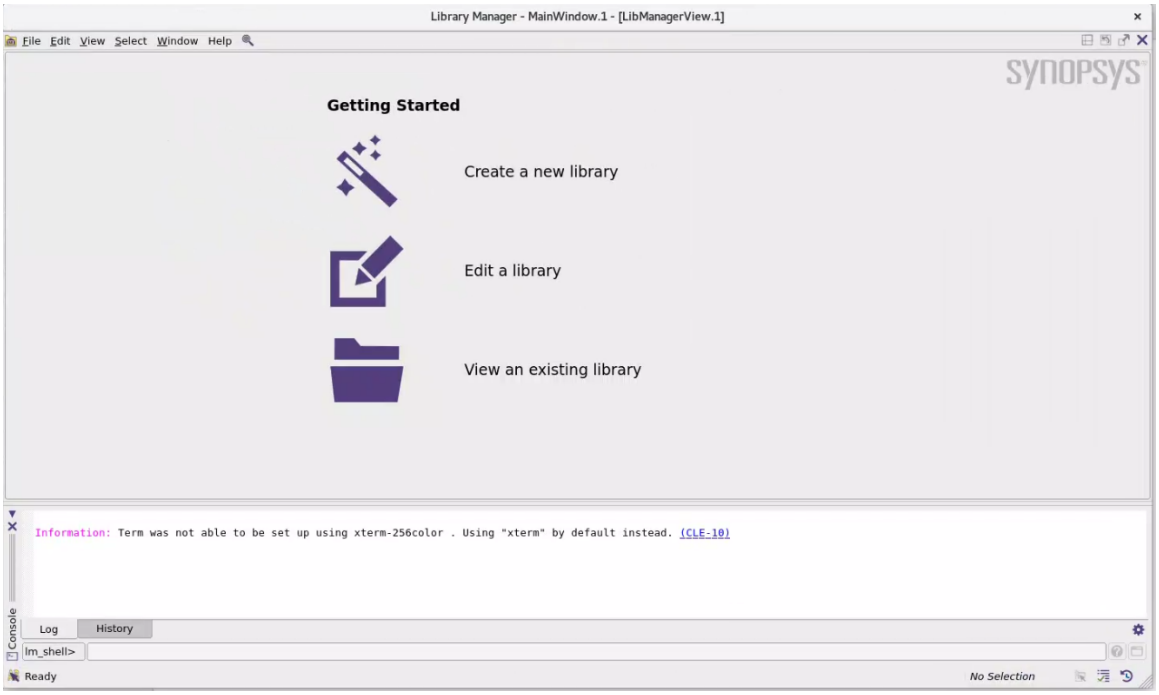


Figura 70: Interfaz gráfica del *Library Manager*

En esta ventana, se debe seleccionar el tipo de *flow* que se llevará a cabo, ya sea *Normal Flow*, *Physical-only Flow*, o *Aggregate Flow*. Además, es necesario definir el nombre del *workspace* donde se almacenará la librería y especificar el archivo `.tf` (*Technology File*) que se utilizará.

10.3.1. *Normal Flow*

El *Normal Flow* permite crear una librería NDM que incluye únicamente celdas que contienen tanto descripciones lógicas como físicas asociadas. Si alguna celda carece de una de estas descripciones, se excluye automáticamente de la librería. Este *flow* se utilizó para generar las librerías correspondientes a las celdas estándar y a los pads de I/O.

Proceso de creación de librerías *Normal Flow*

Como se observa en las Figuras 71 y 72, el proceso comienza con la importación de los archivos `.db` y `.lef`, los cuales contienen la información lógica y física de las celdas estándar.

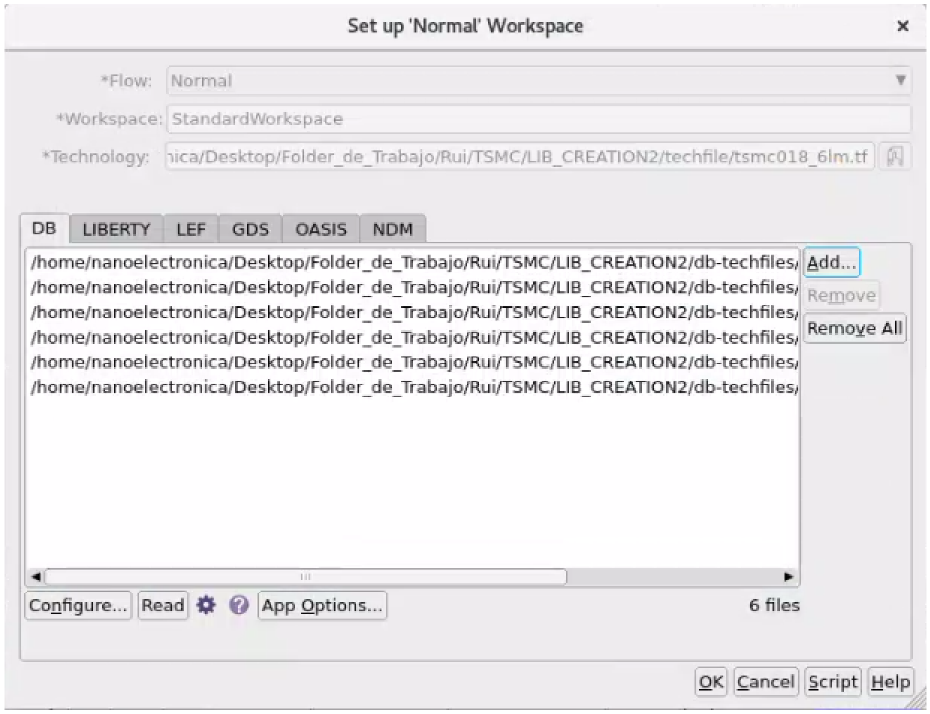


Figura 71: Ventana de creación de librería con *Normal Flow* (Agregar archivos .db)

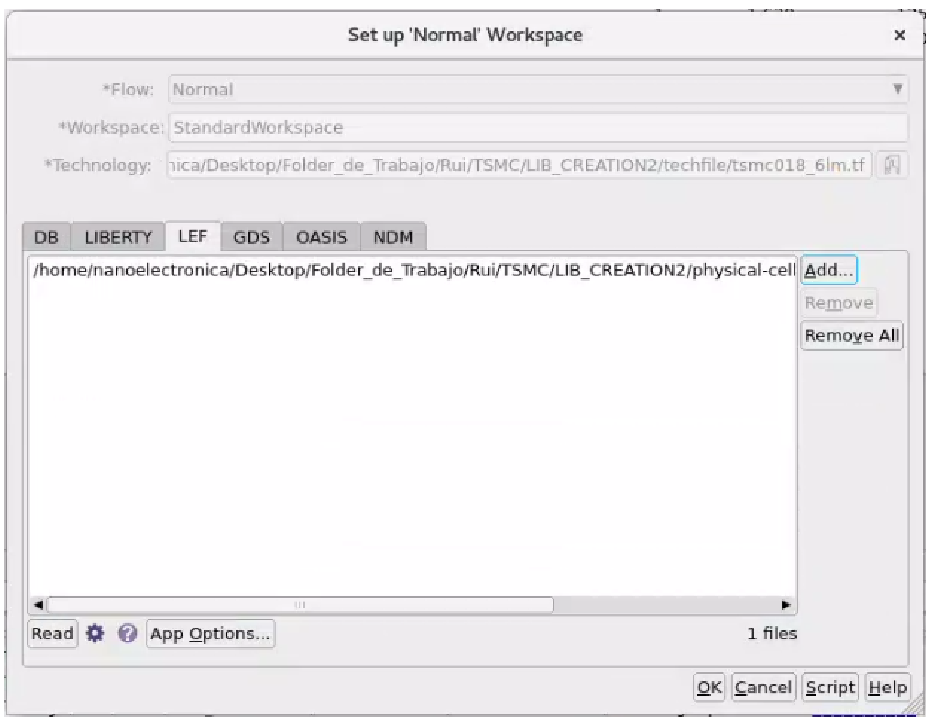


Figura 72: Ventana de creación de librería con *Normal Flow* (Agregar archivos .lef)

A continuación, se detallan los pasos:

1. **Importación de archivos:**
   - Seleccionar los archivos `.db` y `.lef` desde sus respectivas ubicaciones.
   - Hacer clic en el botón *Read* para cargar los contenidos de estos archivos en el *workspace* de la librería.
2. **Verificación y confirmación del *workspace*:**
   - Ejecutar el comando *Check Workspace*, que revisa las celdas cargadas y genera un reporte de errores, si los hubiera. (Figura 73)
   - Si la verificación es exitosa, proceder con *Commit Workspace*, que crea la librería NDM en el directorio donde se abrió la interfaz gráfica. (Figura 74)

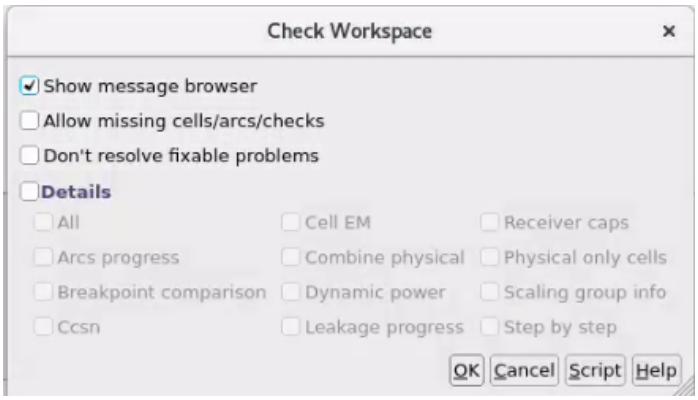


Figura 73: Comando *Check Workspace*

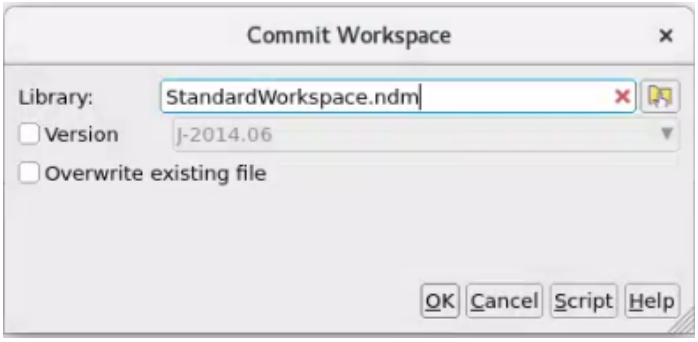


Figura 74: Comando *Commit Workspace*

A continuación, se presentan los comandos utilizados para generar las librerías NDM para las celdas estándar y los pads de I/O:

```
create_workspace -flow normal -technology /home/nanoelectronica/
   Desktop/Folder_de_Trabajo/Rui/TSMC/LIB_CREATION/techfile/
   tsmc018_6lm.tf StandardWorkspace
```

```
read_db { /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7tbc.
   db /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7tlt.db /
   home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7tml.db /
   home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7ttc.db /
   home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7twc.db /
   home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7twcl.db }

read_lef /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/LIB_CREATION/physical-cells/standard-cells/
   tcb018gbwp7t_6lm.lef

current_workspace; check_workspace

current_workspace StandardWorkspace; commit_workspace  -output
   StandardWorkspace.ndm
```

Cuadro 73: Comandos empleados para la realización de la librería NDM de celdas estándar

```
create_workspace -flow normal -technology /home/nanoelectronica/
   Desktop/Folder_de_Trabajo/Rui/TSMC/LIB_CREATION2/techfile/
   tsmc018_6lm.tf PadsWorkspace

read_db { /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/LIB_CREATION2/db-techfiles/IO-cells/tpd018nvtc.db }

read_lef /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/LIB_CREATION2/physical-cells/IO-cells/tpd018nv_6lm.lef

current_workspace; check_workspace

current_workspace PadsWorkspace; commit_workspace  -output
   PadsWorkspace.ndm
```

Cuadro 74: Comandos empleados para la realización de la librería NDM de pines de I/O

10.3.2. *Physical-only Flow*

El *Physical-only Flow* genera librerías NDM que incluyen únicamente celdas cuya información se encuentra en el archivo `.lef`, es decir, celdas que contienen exclusivamente datos físicos. Estas celdas no tienen información lógica asociada, por lo que su funcionalidad está relacionada únicamente con el proceso de fabricación, pero no tienen impacto en las simulaciones lógicas. Ejemplos de este tipo de celdas incluyen *non-metal fillers* y *corners*.

El proceso de creación de librerías con este flujo es similar al del *Normal Flow*, con la única diferencia de que se selecciona el tipo de flujo *Physical-only Flow* en la configuración

inicial (ver Figura 75).

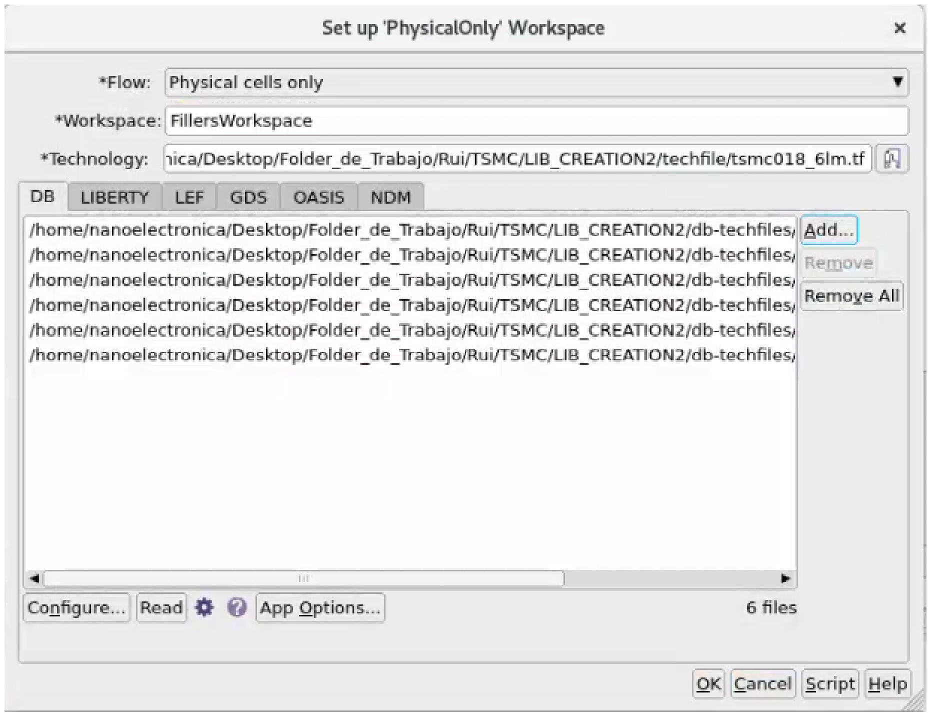


Figura 75: Ventana de creación de librería con *Physical-only Flow*

A continuación, se detallan los comandos utilizados para generar librerías NDM de tipo *Physical-only Flow*, tanto para las celdas estándar (*non-metal fillers*) como para los *pads* de I/O (*non-metal fillers* y *corners*):

```
create_workspace -flow physical_only -technology /home/
   nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION2/techfile/tsmc018_6lm.tf FillersWorkspace

read_db { /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7tbc.
   db /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7tlt.db /
   home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7tml.db /
   home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7ttc.db /
   home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7twc.db /
   home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/db-techfiles/standard-cells/tcb018gbwp7twcl.db }

read_lef /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/LIB_CREATION/physical-cells/standard-cells/
   tcb018gbwp7t_6lm.lef

current_workspace; check_workspace
```

```
current_workspace FillersWorkspace; commit_workspace  -output
   FillersWorkspace.ndm
```

Cuadro 75: Comandos empleados para la realización de la librería NDM física de celdas estándar

```
create_workspace -flow physical_only -technology /home/
   nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/techfile/tsmc018_6lm.tf CornersWorkspace

read_db { /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/LIB_CREATION/db-techfiles/IO-cells/tpd018nvtc.db }

read_lef /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/LIB_CREATION/physical-cells/IO-cells/tpd018nv_6lm.lef

current_workspace; check_workspace

current_workspace CornersWorkspace; commit_workspace  -output
   CornersWorkspace.ndm
```

Cuadro 76: Comandos empleados para la realización de la librería NDM física de pines de I/O

10.3.3. *Aggregate Flow*

Una vez generadas las cuatro librerías NDM, es necesario consolidarlas en una sola librería de referencia que se utilizará en el proceso de síntesis física. Esta librería unificada permite realizar el mapeo completo de las celdas instanciadas durante la síntesis lógica.

El *Aggregate Flow* es el flujo utilizado para combinar múltiples librerías NDM en un solo archivo. En la Figura 76 se ilustra el proceso de selección de las cuatro librerías previamente creadas y su inclusión en el *workspace*.

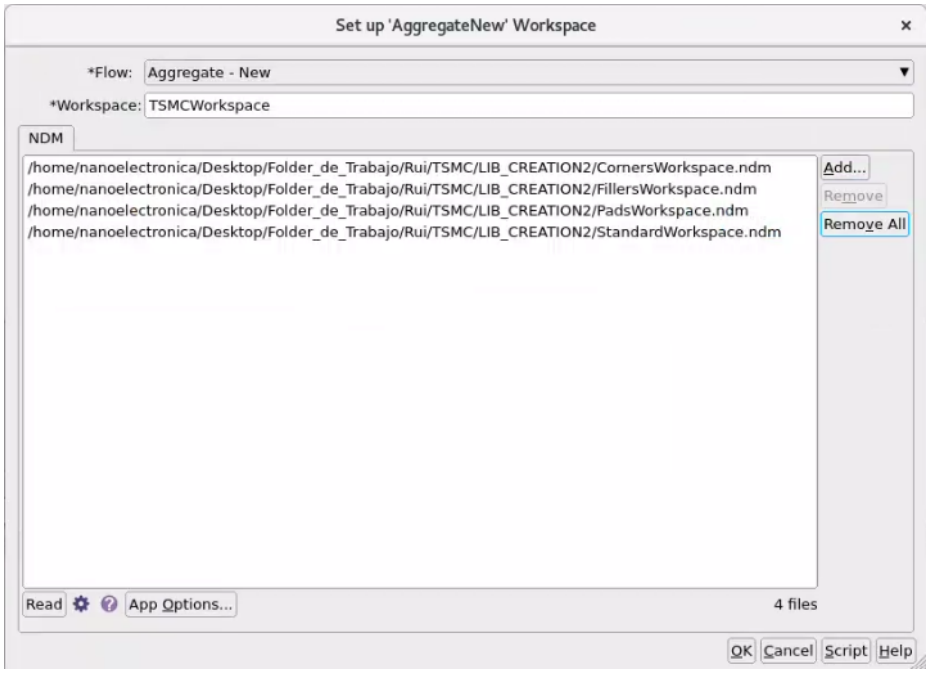


Figura 76: Ventana de creación de librería con *Aggregate Flow*

A continuación, se presentan los comandos utilizados para generar la librería NDM de referencia:

```
create_workspace -flow aggregate TSMCWorkspace

read_ndm /home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/LIB_CREATION/CornersWorkspace.ndm; read_ndm /home/
   nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/FillersWorkspace.ndm; read_ndm /home/
   nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/PadsWorkspace.ndm; read_ndm /home/
   nanoelectronica/Desktop/Folder_de_Trabajo/Rui/TSMC/
   LIB_CREATION/StandardWorkspace.ndm;

current_workspace; check_workspace

current_workspace TSMCWorkspace; commit_workspace  -output
   TSMCWorkspace.ndm
```

Cuadro 77: Comandos empleados para la realización de la librería de referencia

En la Figura 77 se pueden observar las cuatro librerías NDM creadas y la librería NDM de referencia.

Figura 77: Librerías NDM creadas

CAPÍTULO 11

Pruebas finales de funcionamiento en HSPICE

La última verificación que se realiza es la extracción parásita (*LPE* o *Layout Parasitic Extraction*), cuyo resultado genera un *deck* de simulación compatible con *HSPICE*. Este *deck* incluye información sobre los elementos parásitos presentes en el diseño físico, como capacitancias, resistencias e inductancias asociadas a las interconexiones y componentes del circuito. Estos elementos parásitos son calculados a partir del *layout* y son cruciales para analizar el impacto que tienen en el rendimiento y comportamiento del circuito. La simulación con este *deck* permite realizar un análisis más preciso del diseño, considerando los efectos reales del entorno físico en el que operará el circuito.

En iteraciones previas y en la actual, se observó que el archivo `.spf` generado por la extracción parásita no incluye los pines de entrada y salida, lo que imposibilita la simulación directa del *deck* resultante. Ante esta limitación, en esta ocasión se optó por realizar la extracción parásita exclusivamente del núcleo (*core*) de algunos circuitos, excluyendo las celdas de entrada y salida. Esta decisión tuvo como objetivo evaluar la funcionalidad interna del circuito mediante simulaciones más enfocadas.

Adicionalmente, para superar esta restricción, se utilizó el archivo Verilog generado al finalizar la síntesis física junto con los documentos PDF que contienen la información técnica de las celdas estándar proporcionadas por *TSMC*. Con estos recursos, se procedió a elaborar un mapeo manual de los pines de entrada y salida. Este mapeo permitió generar una representación completa del circuito que incluye tanto el núcleo como las interconexiones necesarias para su simulación. Finalmente, empleando librerías de transistores de tecnología de 180 nm, se simuló el comportamiento del circuito, logrando visualizar su funcionalidad.

11.1. Código de *decks* realizados para la simulación de circuitos en *HSPICE*

En los trabajos [34] y [35], se elaboraron *decks* de celdas estándar con el propósito de simular los archivos generados tras la extracción parásita. En este trabajo, se reutilizaron los *decks* previamente disponibles y, además, se diseñaron las definiciones para las celdas que no estaban presentes pero que tenían instancias mencionadas en los *decks* de los circuitos. Estas definiciones se pueden encontrar en el Cuadro 78.

```
*****************************************************
* Subcircuit for TIEHBWP7T (Tied High)
*****************************************************
.SUBCKT TIEHBWP7T Z
    V1 Z 0 DC 1.0
.ENDS TIEHBWP7T

*****************************************************
* Subcircuit for TIELBWP7T (Tied Low)
*****************************************************
.SUBCKT TIELBWP7T ZN
    V1 ZN 0 DC 0
.ENDS TIELBWP7T

*****************************************************
* Subcircuit for PVSS1CDG (Ground Connection)
*****************************************************
.SUBCKT PVSS1CDG VSS VSS1 VSS2 VSS3
    R1 VSS 0 1m * Low resistance to represent ideal connection
.ENDS PVSS1CDG

*****************************************************
* Subcircuit for PVDD1CDG (Power Connection)
*****************************************************
.SUBCKT PVDD1CDG VDD
    R1 VDD 0 1m * Low resistance to represent ideal connection
.ENDS PVDD1CDG

*****************************************************
* Subcircuit for CKND0BWP7T (NOT Gate - Simple)
*****************************************************
.SUBCKT CKND0BWP7T in1 out1 vsource size=1
    M1 out1 in1 0 0 nmos W='size*UNIT_W' L=UNIT_L
    M2 out1 in1 vsource vsource pmos W='2*size*UNIT_W' L=UNIT_L
.ENDS CKND0BWP7T

*****************************************************
* Subcircuit for CKND0BWP7T2 (NOT Gate - Extended)
*****************************************************
.SUBCKT CKND0BWP7T2 in1 out1 vdd vss vsource size=1
    M1 out1 in1 0 0 nmos W='size*UNIT_W' L=UNIT_L
    M2 out1 in1 vsource vsource pmos W='2*size*UNIT_W' L=UNIT_L
.ENDS CKND0BWP7T2
```

```
*****************************************************
* Subcircuit for DFCNQD1BWP7T (D Flip-Flop with Asynchronous
   Clear)
*****************************************************
.SUBCKT DFCNQD1BWP7T D CP CDN Q vsource
    X1 S2 S S1 vsource NandGate size=1
    X2 S1 CP A1 vsource AndGate size=1
    X3 A1 CDN S vsource NandGate size=1
    X4 S CP A2 vsource AndGate size=1
    X5 A2 S2 R vsource NandGate size=1
    X6 R D A3 vsource AndGate size=1
    X7 A3 CDN S2 vsource NandGate size=1
    X8 S Q Q vsource NandGate size=1
.ENDS DFCNQD1BWP7T

*****************************************************
* Subcircuit for AN2D0BWP7T (2-Input AND Gate)
*****************************************************
.SUBCKT AN2D0BWP7T in1 in2 out1 vdd vss vsource size=1
    X1 in1 in2 sal1 vsource NandGate size=1
    X2 sal1 sal1 out1 vsource NandGate size=1
.ENDS AN2D0BWP7T

*****************************************************
* Subcircuit for OR2D0BWP7T (2-Input OR Gate)
*****************************************************
.SUBCKT OR2D0BWP7T in1 in2 out1 vsource size=1
    X1 in1 in2 out_nor vsource NorGate size=1
    X2 out_nor out_nor out1 vsource NorGate size=1
.ENDS OR2D0BWP7T

*****************************************************
* Subcircuit for INVD1BWP7T (NOT Gate)
*****************************************************
.SUBCKT INVD1BWP7T in1 out1 vsource size=1
    M1 out1 in1 0 0 nmos W='size*UNIT_W' L=UNIT_L
    M2 out1 in1 vsource vsource pmos W='2*size*UNIT_W' L=UNIT_L
.ENDS INVD1BWP7T

*****************************************************
* Subcircuit for CKXOR2D1BWP7T (2-Input XOR Gate)
*****************************************************
.SUBCKT CKXOR2D1BWP7T A1 A2 out vsource
    X1 A1 A2 out vsource XorGate size=1
.ENDS CKXOR2D1BWP7T

*****************************************************
* Subcircuit for CKXOR2D0BWP7T (2-Input XOR Gate - Extended)
*****************************************************
.SUBCKT CKXOR2D0BWP7T A1 A2 out vdd vss vsource
    X1 A1 A2 out vsource XorGate size=1
.ENDS CKXOR2D0BWP7T

*****************************************************
```

```
* Subcircuit for XNR2D1BWP7T (2-Input XNOR Gate)
*****************************************************
.SUBCKT XNR2D1BWP7T A1 A2 ZN vdd vss vsource
    X1 A1 A2 nor1 vsource NorGate size=1
    X2 A1 A2 and1 vsource AndGate size=1
    X3 nor1 and1 ZN vsource OrGate size=1
.ENDS XNR2D1BWP7T
```

Cuadro 78: Subcircuitos agregados al *deck* realizado en el trabajo [34]

Para simular el comportamiento de los transistores, se utilizaron las librerías para tecnología *CMOS* de 180 nm desarrolladas por la Universidad Estatal de Arizona (*ASU*) (Cuadro 81). Además, se emplearon otros *decks* auxiliares para configurar parámetros específicos y señales necesarias para la simulación. A continuación, se describen los principales *decks* utilizados:

- `params.sp`: Este *deck* contiene la definición de los parámetros clave para la simulación, como los voltajes de operación y las dimensiones de los transistores a emplear en el circuito (Cuadro 79).
- `params_clocks.sp`: En este archivo se definen las variables que configuran las señales de reloj necesarias para las entradas de los *decks* analizados. Este *deck* permite la configuración precisa de las frecuencias (Cuadro 80).
- `main.sp`: Este archivo actúa como el núcleo central de la simulación. En él se invocan todos los *decks* requeridos para la simulación del circuito en cuestión, incluyendo parámetros, señales y definiciones de celdas. Debido a su rol central, este archivo es el que debe ejecutarse al realizar la simulación en *HSPICE*, asegurando que todas las dependencias y configuraciones sean correctamente integradas.

```
* Parameters definition deck

.PARAM Vdd1=1.5                    * Suggested Voltage for
   Technology to be used
.PARAM Tech = 180e-9
.PARAM UNIT_W = '2*Tech'    * Assuming contacted Difussions
.PARAM UNIT_L = Tech
```

Cuadro 79: Deck `params.sp`

```
* Parameters clk definition deck

.param tr=0.05n    ** rise time
.param tf=0.05n    ** fall time
.param td= 0n     ** initial delay
.param per128=128u    ** period
.param per64=64u    ** period
.param per32=32u    ** period
.param per16=16u    ** period
.param per8=8u    ** period
```

```
.param per4=4u     ** period
.param per2=2u     ** period
.param per1=1u     ** period
.param pwA128=' (per128-tr-tf)/2'  ** high pulse width
.param pwA64=' (per64-tr-tf)/2'  ** high pulse width
.param pwA32=' (per32-tr-tf)/2'  ** high pulse width
.param pwA16=' (per16-tr-tf)/2'  ** high pulse width
.param pwA8=' (per8-tr-tf)/2'  ** high pulse width
.param pwA4=' (per4-tr-tf)/2'  ** high pulse width
.param pwA2=' (per2-tr-tf)/2'  ** high pulse width
.param pwA1=' (per1-tr-tf)/2'  ** high pulse width
```

Cuadro 80: Deck params_clocks.sp

```
*
* Predictive Technology Model Beta Version
* 180nm NMOS SPICE Parametersv (normal one)
*

.model NMOS NMOS
+Level = 49

+Lint = 4.e-08 Tox = 4.e-09
+Vth0 = 0.3999 Rdsw = 250

+lmin=1.8e-7 lmax=1.8e-7 wmin=1.8e-7 wmax=1.0e-4 Tref=27.0
   version =3.1
+Xj= 6.0000000E-08          Nch= 5.9500000E+17
+lln= 1.0000000            lwn= 1.0000000              wln= 0.00
+wwn= 0.00                 ll= 0.00
+lw= 0.00                  lwl= 0.00                   wint=
   0.00
+wl= 0.00                  ww= 0.00                    wwl= 0.00
+Mobmod=  1                binunit= 2                  xl=  0
+xw=  0                    binflag=  0
+Dwg= 0.00                 Dwb= 0.00

+K1= 0.5613000               K2= 1.0000000E-02
+K3= 0.00                  Dvt0= 8.0000000             Dvt1=
   0.7500000
+Dvt2= 8.0000000E-03       Dvt0w= 0.00                 Dvt1w=
   0.00
+Dvt2w= 0.00               Nlx= 1.6500000E-07          W0= 0.00
+K3b= 0.00                 Ngate= 5.0000000E+20

+Vsat= 1.3800000E+05       Ua= -7.0000000E-10          Ub=
   3.5000000E-18
+Uc= -5.2500000E-11        Prwb= 0.00
+Prwg= 0.00                Wr= 1.0000000               U0=
   3.5000000E-02
+A0= 1.1000000             Keta= 4.0000000E-02         A1= 0.00
+A2= 1.0000000             Ags= -1.0000000E-02         B0= 0.00
+B1= 0.00
```

```
+Voff= -0.12350000          NFactor= 0.9000000          Cit=
   0.00
+Cdsc= 0.00                 Cdscb= 0.00                 Cdscd=
   0.00
+Eta0= 0.2200000            Etab= 0.00                  Dsub=
   0.8000000

+Pclm= 5.0000000E-02        Pdiblc1= 1.2000000E-02      Pdiblc2=
   7.5000000E-03
+Pdiblcb= -1.3500000E-02    Drout= 1.7999999E-02        Pscbe1=
   8.6600000E+08
+Pscbe2= 1.0000000E-20      Pvag= -0.2800000            Delta=
   1.0000000E-02
+Alpha0= 0.00               Beta0= 30.0000000

+kt1= -0.3700000            kt2= -4.0000000E-02         At=
   5.5000000E+04
+Ute= -1.4800000            Ua1= 9.5829000E-10          Ub1=
   -3.3473000E-19
+Uc1= 0.00                  Kt1l= 4.0000000E-09         Prt= 0.00

+Cj= 0.00365                Mj= 0.54                    Pb= 0.982
+Cjsw= 7.9E-10              Mjsw= 0.31                  Php=
   0.841
+Cta= 0                     Ctp= 0                      Pta= 0
+Ptp= 0                     JS=1.50E-08                 JSW=2.50E
   -13
+N=1.0                      Xti=3.0                     Cgdo
   =2.786E-10
+Cgso=2.786E-10             Cgbo=0.0E+00                Capmod= 2
+NQSMOD= 0                  Elm= 5                      Xpart= 1
+Cgsl= 1.6E-10              Cgdl= 1.6E-10               Ckappa=
   2.886
+Cf= 1.069e-10              Clc= 0.0000001              Cle= 0.6
+Dlc= 4E-08                 Dwc= 0                      Vfbcv= -1


*
* Predictive Technology Model Beta Version
* 180nm PMOS SPICE Parametersv (normal one)
*

.model PMOS PMOS
+Level = 49

+Lint = 3.e-08 Tox = 4.2e-09
+Vth0 = -0.42 Rdsw = 450

+lmin=1.8e-7 lmax=1.8e-7 wmin=1.8e-7 wmax=1.0e-4 Tref=27.0
   version =3.1
+Xj= 7.0000000E-08          Nch= 5.9200000E+17
+lln= 1.0000000            lwn= 1.0000000              wln= 0.00
+wwn= 0.00                  ll= 0.00
```

```
+lw= 0.00                  lwl= 0.00                  wint=
   0.00
+wl= 0.00                  ww= 0.00                   wwl= 0.00
+Mobmod=  1                binunit= 2                 xl= 0.00
+xw= 0.00
+binflag=  0               Dwg= 0.00                  Dwb= 0.00

+ACM= 0                    ldif=0.00                  hdif=0.00
+rsh= 0                    rd= 0                      rs= 0
+rsc= 0                    rdc= 0

+K1= 0.5560000             K2= 0.00
+K3= 0.00                  Dvt0= 11.2000000           Dvt1=
   0.7200000
+Dvt2= -1.0000000E-02      Dvt0w= 0.00                Dvt1w=
   0.00
+Dvt2w= 0.00               Nlx= 9.5000000E-08         W0= 0.00
+K3b= 0.00                 Ngate= 5.0000000E+20

+Vsat= 1.0500000E+05       Ua= -1.2000000E-10         Ub=
   1.0000000E-18
+Uc= -2.9999999E-11        Prwb= 0.00
+Prwg= 0.00                Wr= 1.0000000              U0=
   8.0000000E-03
+A0= 2.1199999             Keta= 2.9999999E-02        A1= 0.00
+A2= 0.4000000             Ags= -0.1000000            B0= 0.00
+B1= 0.00

+Voff= -6.40000000E-02      NFactor= 1.4000000         Cit=
   0.00
+Cdsc= 0.00                Cdscb= 0.00                Cdscd=
   0.00
+Eta0= 8.5000000           Etab= 0.00                 Dsub=
   2.8000000

+Pclm= 2.0000000           Pdiblc1= 0.1200000         Pdiblc2=
   8.0000000E-05
+Pdiblcb= 0.1450000        Drout= 5.0000000E-02       Pscbe1=
   1.0000000E-20
+Pscbe2= 1.0000000E-20     Pvag= -6.0000000E-02       Delta=
   1.0000000E-02
+Alpha0= 0.00              Beta0= 30.0000000

+kt1= -0.3700000           kt2= -4.0000000E-02        At=
   5.5000000E+04
+Ute= -1.4800000           Ua1= 9.5829000E-10         Ub1=
   -3.3473000E-19
+Uc1= 0.00                 Kt1l= 4.0000000E-09        Prt= 0.00

+Cj= 0.00138               Mj= 1.05                   Pb= 1.24
+Cjsw= 1.44E-09            Mjsw= 0.43                 Php=
   0.841
+Cta= 0.00093              Ctp= 0                     Pta=
   0.00153
```

```
+Ptp= 0                    JS=1.50E-08                JSW=2.50E
   -13
+N=1.0                     Xti=3.0                    Cgdo
   =2.786E-10
+Cgso=2.786E-10            Cgbo=0.0E+00               Capmod= 2
+NQSMOD= 0                 Elm= 5                     Xpart= 1
+Cgsl= 1.6E-10             Cgdl= 1.6E-10              Ckappa=
   2.886
+Cf= 1.058e-10             Clc= 0.0000001             Cle= 0.6
+Dlc= 3E-08                Dwc= 0                     Vfbcv= -1
```

Cuadro 81: Deck con la definición de la tecnología de MOSFET de 180nm

11.2. Compuerta NOT

El primer circuito al que se realizó el mapeo de pines y las pruebas en *HSPICE* fue una compuerta *NOT*. El archivo original generado a partir de la extracción parásita se presenta en el Cuadro 82. Como se puede observar, la definición del subcircuito carece de pines definidos, lo que dificulta la interpretación y simulación directa. Además, los nombres de los nodos, como `I_589437A31:N_3`, son poco intuitivos y complejos de identificar a simple vista, lo que añade un desafío adicional para comprender la estructura y funcionalidad del circuito.

Para superar estas limitaciones, fue necesario realizar un proceso de mapeo que asociara los pines funcionales del circuito con los nodos presentes en el archivo. Este mapeo permitió establecer una correspondencia clara entre los nombres de los nodos en el *deck* extraído y las conexiones funcionales esperadas en el diseño original.

```
*
*|DSPF 1.3
*|DESIGN Not_IO
*|DATE "Sat Nov  2 15:36:21 2024"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "V-2023.12-SP1"
*|DIVIDER /
*|DELIMITER :
**FORMAT SPF
*

** COMMENTS

** OPERATING_TEMPERATURE 25
** GLOBAL_TEMPERATURE 25
**
** TCAD_GRD_FILE /home/nanoelectronica/Desktop/Folder_de_Trabajo
   /Rui/TSMC/nueva_integracion/sintesis_fisica_NOT_MODIFIED/
   Outputs/LPE/cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
**   TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
```

```
**   TCADGRD_VERSION 62

.SUBCKT Not_IO

*|GROUND_NET 0

*|NET N_10 0PF
*|I (I_589437A33:N_3 I_589437A33 N_3 B 0 171.4800 168.1200)
*|I (I_589437A34:N_3 I_589437A34 N_3 B 0 232.1800 157.0000)
*|I (I_589437A34:N_4 I_589437A34 N_4 B 0 234.6800 173.8000)
*|I (I_589437A34:N_5 I_589437A34 N_5 B 0 232.1800 190.6000)
*|I (I_589437A35:N_2 I_589437A35 N_2 B 0 192.2000 175.9600)
*|I (I_589437A36:N_3 I_589437A36 N_3 B 0 173.7200 175.9600)
R2_1 I_589437A33:N_3 I_589437A34:N_5 16.3101
R2_2 I_589437A33:N_3 I_589437A34:N_3 15.18
R2_3 I_589437A33:N_3 I_589437A34:N_4 432.779
R2_4 I_589437A33:N_3 I_589437A35:N_2 10.5725
R2_5 I_589437A33:N_3 I_589437A36:N_3 13.6011
R2_6 I_589437A34:N_3 I_589437A36:N_3 24.0633
R2_7 I_589437A34:N_3 I_589437A35:N_2 17.638
R2_8 I_589437A34:N_3 I_589437A34:N_5 6.25994
R2_9 I_589437A34:N_3 I_589437A34:N_4 0.441559
R2_10 I_589437A34:N_4 I_589437A36:N_3 651.377
R2_11 I_589437A34:N_4 I_589437A35:N_2 476.655
R2_12 I_589437A34:N_4 I_589437A34:N_5 0.441555
R2_13 I_589437A34:N_5 I_589437A36:N_3 23.2844
R2_14 I_589437A34:N_5 I_589437A35:N_2 17.0112
R2_15 I_589437A35:N_2 I_589437A36:N_3 3.26711
R2_16 I_589437A35:N_2 N_10:7 0.278809

*|NET N_11 0PF
*|I (I_589437A33:N_4 I_589437A33 N_4 B 0 171.4800 172.0400)
*|I (I_589437A35:N_3 I_589437A35 N_3 B 0 192.2000 172.0400)
*|I (I_589437A36:N_4 I_589437A36 N_4 B 0 173.7200 172.0400)
*|I (I_589437A37:N_3 I_589437A37 N_3 B 0 111.5000 194.4400)
*|I (I_589437A37:N_4 I_589437A37 N_4 B 0 111.5000 181.0000)
*|I (I_589437A37:N_5 I_589437A37 N_5 B 0 111.5000 168.1200)
*|I (I_589437A37:N_6 I_589437A37 N_6 B 0 111.5000 153.8400)
R3_1 I_589437A33:N_4 I_589437A37:N_5 33.2013
R3_2 I_589437A33:N_4 I_589437A37:N_3 30.8269
R3_3 I_589437A33:N_4 I_589437A37:N_4 33.211
R3_4 I_589437A33:N_4 I_589437A35:N_3 10.9164
R3_5 I_589437A33:N_4 I_589437A37:N_6 30.6617
R3_6 I_589437A33:N_4 I_589437A36:N_4 0.371745
R3_7 I_589437A35:N_3 I_589437A37:N_5 63.3604
R3_8 I_589437A35:N_3 I_589437A37:N_3 58.7625
R3_9 I_589437A35:N_3 I_589437A37:N_4 63.3424
R3_10 I_589437A35:N_3 I_589437A37:N_6 58.5485
R3_11 I_589437A35:N_3 I_589437A36:N_4 3.0669
R3_12 I_589437A37:N_3 I_589437A37:N_5 45.9763
R3_13 I_589437A37:N_3 I_589437A37:N_6 52.5342
R3_14 I_589437A37:N_3 I_589437A37:N_4 38.7333
R3_15 I_589437A37:N_4 I_589437A37:N_5 40.0611
R3_16 I_589437A37:N_4 I_589437A37:N_6 46.3492
```

```
R3_17 I_589437A37:N_5 I_589437A37:N_6 39.0396

*|NET N_2 0PF
*|I (I_589437A31:N_4 I_589437A31 N_4 B 0 164.7600 114.5000)
*|I (I_589437A31:N_7 I_589437A31 N_7 B 0 181.5600 114.5000)
*|I (I_589437A32:N_7 I_589437A32 N_7 B 0 169.2400 233.5000)
*|I (I_589437A33:N_2 I_589437A33 N_2 B 0 172.8800 170.3600)
R4_1 I_589437A31:N_4 N_2:5 51.2485
R4_2 I_589437A31:N_4 I_589437A31:N_7 21.9558
R4_3 I_589437A31:N_7 N_2:5 53.0425
R4_4 I_589437A32:N_7 N_2:5 31.2247
R4_5 I_589437A33:N_2 N_2:5 19.278

*|NET N_3 0PF
*|I (I_589437A31:N_3 I_589437A31 N_3 B 0 154.1200 114.5000)
*|I (I_589437A31:N_5 I_589437A31 N_5 B 0 173.1600 114.5000)
*|I (I_589437A31:N_6 I_589437A31 N_6 B 0 177.6400 114.5000)
*|I (I_589437A32:N_4 I_589437A32 N_4 B 0 186.0400 233.5000)
*|I (I_589437A32:N_5 I_589437A32 N_5 B 0 175.4000 233.5000)
*|I (I_589437A32:N_6 I_589437A32 N_6 B 0 173.1600 233.5000)
*|I (I_589437A36:N_2 I_589437A36 N_2 B 0 175.1200 174.2800)
R5_1 I_589437A31:N_3 I_589437A31:N_6 1796.49
R5_2 I_589437A31:N_3 N_3:8 9459.93
R5_3 I_589437A31:N_3 I_589437A31:N_5 19.8344
R5_4 I_589437A31:N_5 I_589437A31:N_6 15.9882
R5_5 I_589437A31:N_5 N_3:8 84.1906
R5_6 I_589437A31:N_6 N_3:8 73.272
R5_7 I_589437A32:N_4 I_589437A32:N_5 16.2921
R5_8 I_589437A32:N_5 I_589437A36:N_2 135.577
R5_9 I_589437A32:N_5 N_3:8 283.644
R5_10 I_589437A32:N_5 I_589437A32:N_6 14.9417
R5_11 I_589437A32:N_6 I_589437A36:N_2 123.822
R5_12 I_589437A32:N_6 N_3:8 259.05
R5_13 I_589437A36:N_2 N_3:8 21.9614

*|NET N_4 0PF
*|I (I_589437A31:N_8 I_589437A31 N_8 B 0 189.9600 114.5000)
*|I (I_589437A35:N_4 I_589437A35 N_4 B 0 192.4800 173.7200)
R6_1 I_589437A31:N_8 I_589437A35:N_4 49.0987

*|NET N_5 0PF
*|I (I_589437A32:N_3 I_589437A32 N_3 B 0 195.0000 233.5000)
*|I (I_589437A35:N_5 I_589437A35 N_5 B 0 193.5775 174.8400)
R7_1 I_589437A32:N_3 I_589437A35:N_5 49.1767

*
* Instance Section
*
XI_589437A31 N_6 I_589437A31:N_3 I_589437A31:N_4 I_589437A31:N_5
    I_589437A31:N_6 I_589437A31:N_7 I_589437A31:N_8 PDDW0204SCDG
XI_589437A32 N_7 I_589437A32:N_3 I_589437A32:N_4 I_589437A32:N_5
    I_589437A32:N_6 I_589437A32:N_7 N__generated_12 PDDW0204SCDG
XI_589437A33 I_589437A33:N_2 I_589437A33:N_3 I_589437A33:N_4
   TIEHBWP7T
```

```
    XI_589437A34 N_8 I_589437A34:N_3 I_589437A34:N_4 I_589437A34:N_5
        PVSS1CDG
    XI_589437A35 I_589437A35:N_2 I_589437A35:N_3 I_589437A35:N_4
       I_589437A35:N_5 CKND0BWP7T
    XI_589437A36 I_589437A36:N_2 I_589437A36:N_3 I_589437A36:N_4
       TIELBWP7T
    XI_589437A37 N_9 I_589437A37:N_3 I_589437A37:N_4 I_589437A37:N_5
        I_589437A37:N_6 PVDD1CDG

    .ENDS
```

Cuadro 82: Deck .spf obtenido luego de LPE de una compuerta NOT

Por esta razón, se utilizó el archivo Verilog presentado en el Cuadro 83 para llevar a cabo el mapeo de los pines. Este archivo sirvió como referencia principal debido a que contiene información detallada sobre las instancias y el orden de los pines en el diseño lógico. A partir de esta información, se verificó y ajustó el mapeo de los nodos presentes en el archivo `.spf`, asegurando que las conexiones coincidieran correctamente con las definiciones del circuito extraído.

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

supply1 VDD ;
supply0 VSS ;

CKND0BWP7T U2 ( .I ( A ) , .ZN ( Y ) , .VDD ( VDD ) , .VSS ( VSS ) )
    ;

PVDD1CDG PVDD1 ( .VDD ( VDD ) ) ;
PVSS1CDG PVSS1 ( .VSS ( VSS ) ) ;

endmodule
```

Cuadro 83: Verilog del *core* de una *NOT* obtenido luego de la síntesis física

Luego del mapeo de pines del *core* de la NOT se obtuvo el siguiente deck:

```
        *
        *|DSPF 1.3
        *|DESIGN Not
        *|DATE "Sat Nov  9 18:46:14 2024"
        *|VENDOR "Synopsys"
```

```
*|PROGRAM "StarRC"
*|VERSION "V-2023.12-SP1"
*|DIVIDER /
*|DELIMITER :
**FORMAT SPF
*

** COMMENTS

** OPERATING_TEMPERATURE 25
** GLOBAL_TEMPERATURE 25
**
** TCAD_GRD_FILE /home/nanoelectronica/Desktop/
   Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
   sintesis_fisica_not_noIO/Outputs/LPE/
   cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
**   TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
**   TCADGRD_VERSION 62

.SUBCKT Not A Y vsource

*|GROUND_NET 0

*|NET N_2 0.354649PF
*|I (I_C8A989271:N_2 I_C8A989271 N_2 B 0 232.1800 157.0000)
*|I (I_C8A989271:Y I_C8A989271 Y B 0 234.6800 173.8000)
*|I (I_C8A989271:N_4 I_C8A989271 N_4 B 0 232.1800 190.6000)
*|I (A I_C8A989273 N_2 B 0 173.7200 175.9600)
Cg2_1 I_C8A989271:N_2 0 2.5463e-14
Cg2_2 I_C8A989271:N_4 0 7.21363e-14
Cg2_3 N_2:5 0 2.5705e-13
R2_1 I_C8A989271:N_2 A 25.215
R2_2 I_C8A989271:N_2 N_2:5 5.26195
R2_3 I_C8A989271:N_2 I_C8A989271:N_4 8.08502
R2_4 I_C8A989271:N_2 I_C8A989271:Y 0.441787
R2_5 I_C8A989271:Y A 1386.28
R2_6 I_C8A989271:Y N_2:5 202.746
R2_7 I_C8A989271:Y I_C8A989271:N_4 0.441272
R2_8 I_C8A989271:N_4 A 24586.8
R2_9 I_C8A989271:N_4 N_2:5 12.286
R2_10 A N_2:5 3.72771

*
* Instance Section
*
XI_C8A989271 I_C8A989271:N_2 I_C8A989271:Y I_C8A989271:N_4
   N_4 PVSS1CDG
XI_C8A989273 A Y vsource CKND0BWP7T

.ENDS
```

Cuadro 84: Archivo .spf con pines mapeados para una compuerta NOT

Como se puede observar, este *deck* ya es completamente simulable. A continuación, se

presenta la configuración del archivo `main.sp`, el cual sirve como archivo principal para llevar a cabo la simulación en *HSPICE*.

```
*Comentario inicial
* Se incluyen los archivos .sp seccionados

.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/
   LPE/DECKS/misDecks.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/
   LPE/DECKS/lib_180nm.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/
   LPE/DECKS/Params.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/
   LPE/DECKS/Params_clks.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/nueva_integracion/sintesis_fisica_NOT_MODIFIED/Outputs/
   LPE/Not_RC.spf'

* Fuente de alimentacion
v1 vdd 0 1.5

* Senal de reloj que sera inyectada a cada entrada del circuito.
V_A32 clk32 gnd pulse (0 vdd1 td tr tf pwA32 per32 )
V_A16 clk16 gnd pulse (0 vdd1 td tr tf pwA16 per16 )
V_A8 A gnd pulse (0 vdd1 td tr tf pwA8 per8 )
V_A4 clk4 gnd pulse (0 vdd1 td tr tf pwA4 per4 )
V_A2 clk2 gnd pulse (0 vdd1 td tr tf pwA2 per2 )
V_A1 clk1 gnd pulse (0 vdd1 td tr tf pwA1 per1 )

XNOT A Y vdd Not

*Configuracion del analisis transitorio
.TRAN 1e-9 32.1u
.Option post
.PROBE v(Y) v(A)
.END

.END
```

Cuadro 85: Configuración del archivo main.sp para una compuerta NOT

11.2.1. Resultados

```
hspice -i main.sp
```

Cuadro 86: Comando empleado para ejecutar main.sp en HSPICE

```
wv -i main.tr0 &
```

Cuadro 87: Comando empleado para visualizar el comportamiento de un circuito en *WaveView*

Se ejecutó la simulación del archivo `main.sp` con *HSPICE* y se visualizaron los resultados a través de la herramienta *WaveView*. Empleando los comandos de los Cuadros 86 y 87, se obtuvieron los siguientes resultados.

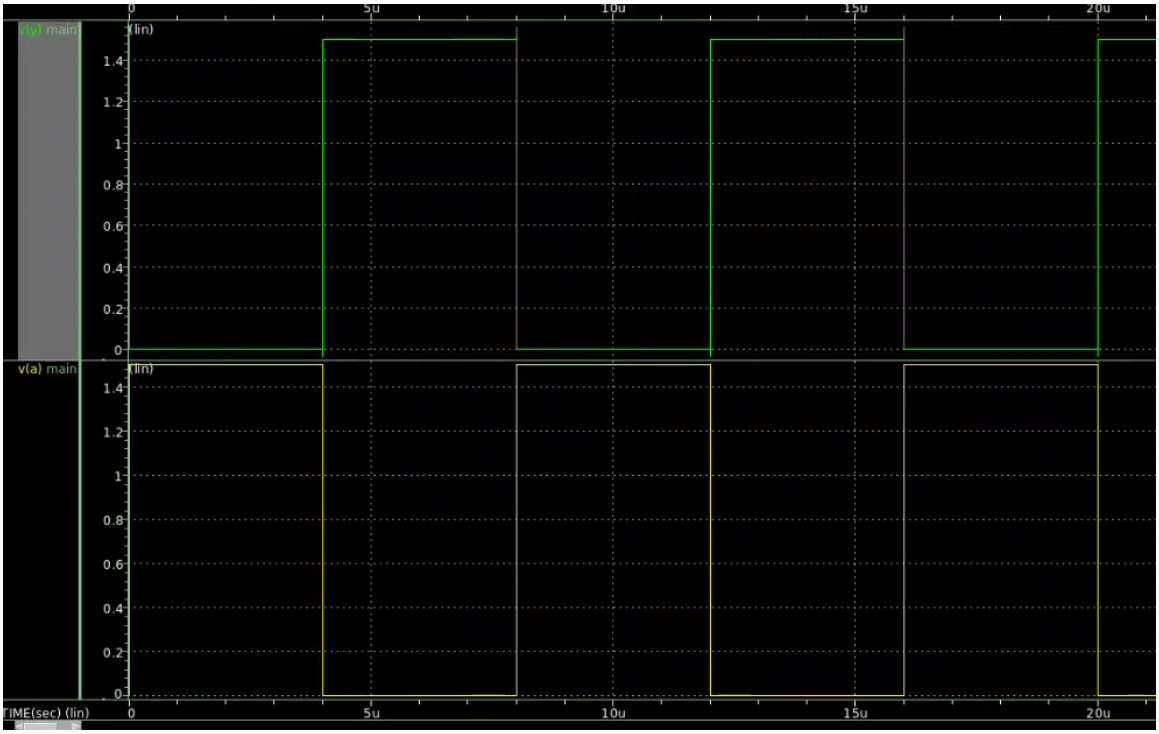


Figura 78: Resultados de la simulación de *HSPICE* en *WaveView* de una compuerta NOT

Como se puede observar, el comportamiento del circuito es el esperado, mostrando que la salida de la compuerta *NOT* responde correctamente a las transiciones de la señal de entrada. Es importante destacar que el comportamiento observado parece ser prácticamente ideal, con una transición limpia y rápida entre los estados lógicos.

Esto podría explicarse debido a la configuración utilizada durante la extracción parásita, la cual fue ajustada para minimizar la inserción de elementos parásitos en el diseño. Este enfoque buscó reducir al máximo el impacto de las capacitancias, resistencias e inductancias parásitas en el circuito, lo que podría resultar en simulaciones que reflejen un comportamiento más cercano al ideal en comparación con un entorno físico real.

Sin embargo, aunque este enfoque facilita el análisis funcional, en etapas posteriores sería recomendable realizar simulaciones más detalladas con una extracción parásita más completa para evaluar el impacto real de los parásitos en el rendimiento del diseño.

Tras la obtención de los *decks* con elementos parásitos de los demás circuitos trabajados en secciones anteriores, se observó que, incluso utilizando la opción para reducir la cantidad de parásitos, estos archivos llegaban a superar las 20,000 líneas. Esta complejidad dificultaba

tanto el análisis como la simulación de los circuitos en *HSPICE*.

Debido a esta situación, se optó por trabajar con circuitos más simples, lo que permitió implementar y validar el proceso de mapeo de pines de manera más manejable. Esta decisión también facilitó la documentación detallada del procedimiento, asegurando que los pasos seguidos para realizar el mapeo de pines quedaran claramente establecidos y pudieran ser replicados en futuros trabajos con circuitos de mayor complejidad.

11.3. Compuerta XOR

El siguiente circuito en ser sometido a pruebas de funcionamiento mediante *HSPICE* fue una compuerta *XOR*. A diferencia de la compuerta *NOT*, que únicamente contenía una instancia de una celda (el inversor), se decidió implementar la *XOR* utilizando módulos básicos de compuertas *AND*, *NOT* y *OR*. Este enfoque tenía como propósito practicar el proceso de mapeo de pines en un circuito que involucrara múltiples instancias de celdas.

Tras realizar el mapeo de pines en el *deck* con parásitos generado para este circuito, se observaron comportamientos extraños durante la simulación, como se ilustra en la Figura 79. A pesar de realizar múltiples revisiones, no se encontraron errores en el mapeo de los pines.

Debido a estos resultados inesperados, se tomó la decisión de eliminar todas las resistencias y capacitancias parásitas del *deck* para validar el funcionamiento exclusivamente del *core* del circuito, sin interferencias de los efectos parásitos.

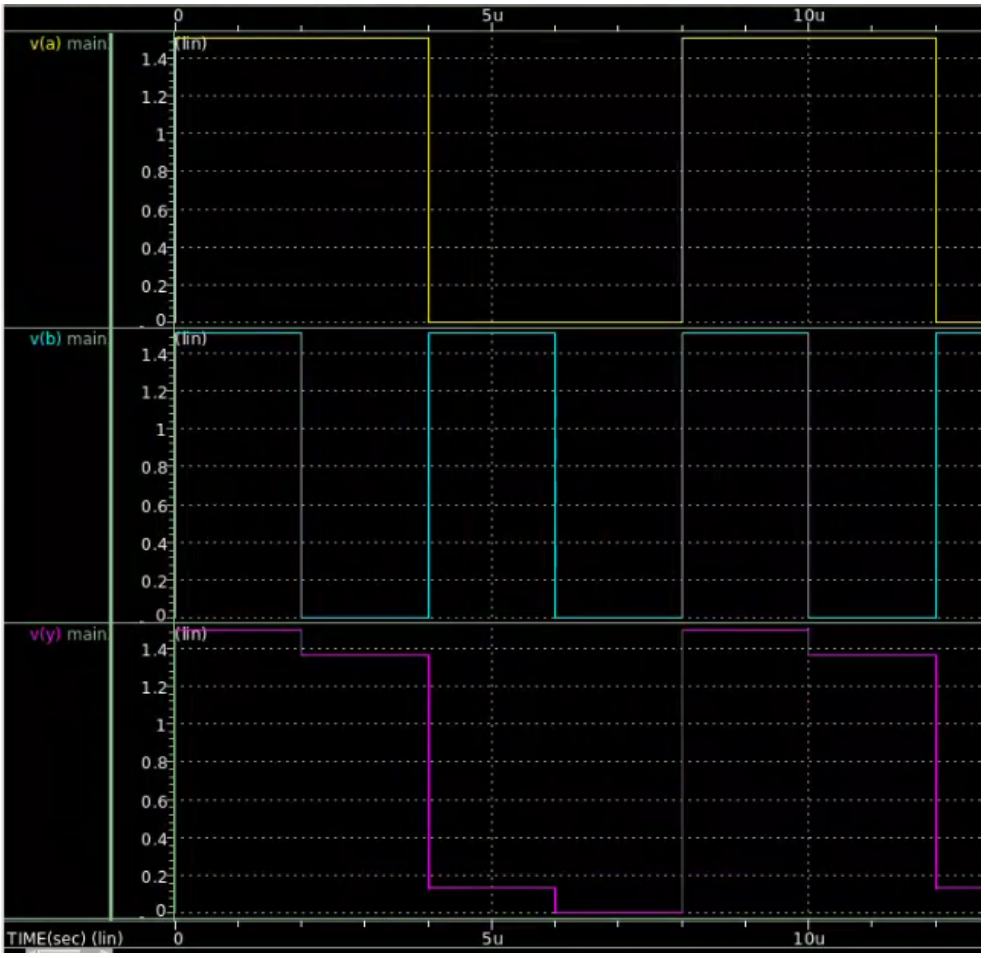


Figura 79: Simulación inicial de una compuerta XOR empleando su *deck* con parásitos

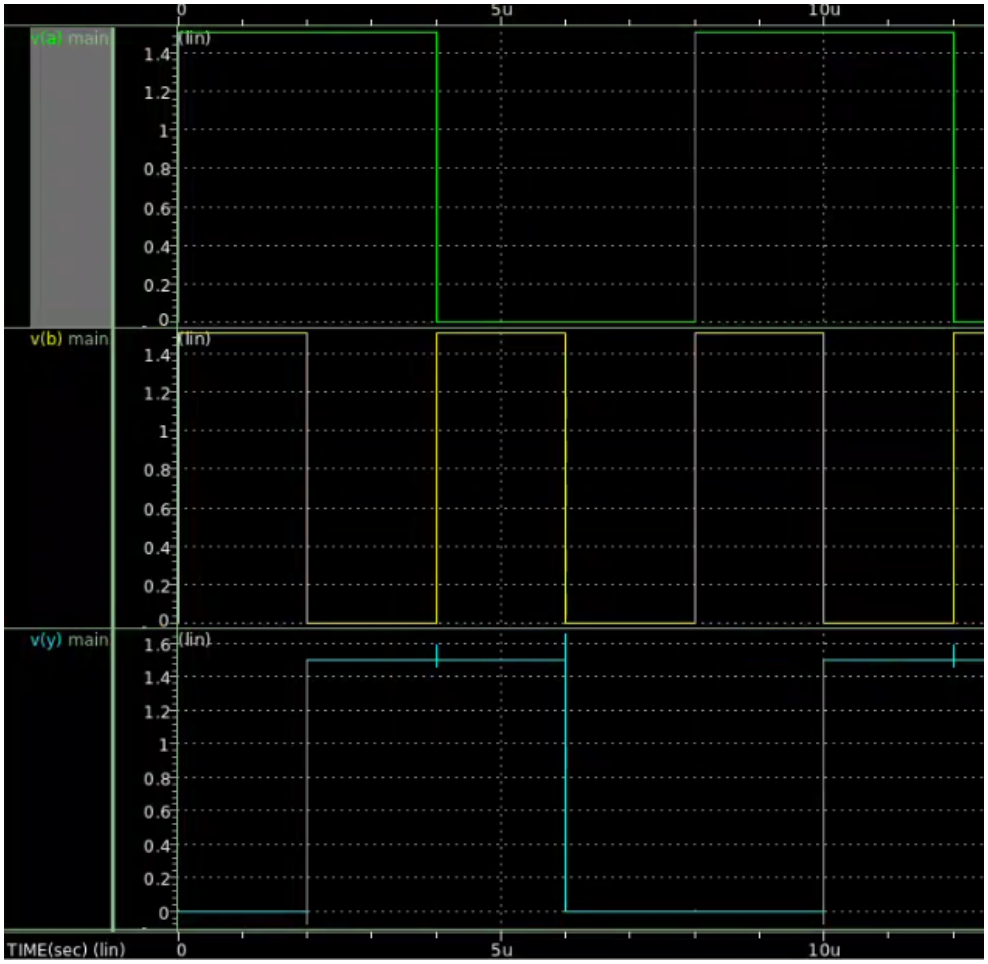


Figura 80: Simulación final de una compuerta XOR utilizando un *deck* simplificado que incluye únicamente el *core* del circuito

En la Figura 80 se muestra que, utilizando el *deck* presentado en el Cuadro 88, el funcionamiento de la compuerta *XOR* es el esperado. Los pines fueron mapeados empleando las instancias de los componentes definidas en el código Verilog de esta compuerta, lo cual permitió establecer correctamente las conexiones necesarias para la simulación.

Es posible que los problemas encontrados se debieran a varios factores, incluyendo posibles errores en el archivo `CMD`, el cual fue generado en años anteriores. Dicho archivo, que contiene los comandos necesarios para realizar la extracción de parásitos y las restricciones a considerar durante el proceso, no fue sometido a pruebas en su momento. Esto se debió principalmente a la falta de pines en los circuitos, lo que impidió llevar a cabo simulaciones funcionales en *HSPICE* durante aquellas iteraciones.

```
*
*|DSPF 1.3
*|DESIGN xor_using_basic_gates
*|DATE "Tue Nov 12 12:59:40 2024"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "V-2023.12-SP1"
```

```
*|DIVIDER /
*|DELIMITER :
**FORMAT SPF
*

** COMMENTS

** OPERATING_TEMPERATURE 25
** GLOBAL_TEMPERATURE 25
**
** TCAD_GRD_FILE //home/nanoelectronica/Desktop/
   Folder_de_Trabajo/Rui/TSMC/nueva_integracion/
   sintesis_fisica_xor_noIO/Outputs/LPE/
   cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
**   TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
**   TCADGRD_VERSION 62

.SUBCKT xor_using_basic_gates A B Y vsource

*|GROUND_NET 0

*
* Instance Section
*
XT1 NOTA B AND2 N4 N5 vsource AN2D0BWP7T
XT2 NOTB A AND1 N9 N10 vsource AN2D0BWP7T
XT3 A NOTA N13 N14 vsource CKND0BWP7T2
XT4 B NOTB N17 N18 vsource CKND0BWP7T2
XT5 AND1 AND2 Y N22 vsource OR2D0BWP7T
XT6 N23 N24 N25 N26 PVSS1CDG

.ENDS
```

Cuadro 88: Archivo `.spf` con pines mapeados para el *core* de una compuerta *XOR*

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

supply1 VDD ;
supply0 VSS ;

OR2D0BWP7T U1 ( .A1 ( A ) , .A2 ( B ) , .Z ( Y ) , .VDD ( VDD )
   ,
```

```
    .VSS ( VSS ) ) ;
endmodule


module and_gate_0 ( A , B , Y , VDD , VSS ) ;
input  A ;
input  B ;
output Y ;
input  VDD ;
input  VSS ;

supply1 VDD ;
supply0 VSS ;

AN2D0BWP7T U1 ( .A1 ( B ) , .A2 ( A ) , .Z ( Y ) , .VDD ( VDD )
   ,
    .VSS ( VSS ) ) ;
endmodule


module and_gate_1 ( A , B , Y , VDD , VSS ) ;
input  A ;
input  B ;
output Y ;
input  VDD ;
input  VSS ;

supply1 VDD ;
supply0 VSS ;

AN2D0BWP7T U1 ( .A1 ( B ) , .A2 ( A ) , .Z ( Y ) , .VDD ( VDD )
   ,
    .VSS ( VSS ) ) ;
endmodule


module not_gate_0 ( A , Y , VDD , VSS ) ;
input  A ;
output Y ;
input  VDD ;
input  VSS ;

supply1 VDD ;
supply0 VSS ;

CKND0BWP7T U1 ( .I ( A ) , .ZN ( Y ) , .VDD ( VDD ) , .VSS ( VSS
    ) ) ;
endmodule


module not_gate_1 ( A , Y , VDD , VSS ) ;
input  A ;
output Y ;
input  VDD ;
```

```
    input  VSS ;

    supply1 VDD ;
    supply0 VSS ;

    CKND0BWP7T U1 ( .I ( A ) , .ZN ( Y ) , .VDD ( VDD ) , .VSS ( VSS
        ) ) ;
    endmodule


    module xor_using_basic_gates ( A , B , Y , VDD , VSS ) ;
    input  A ;
    input  B ;
    output Y ;
    input  VDD ;
    input  VSS ;

    wire not_A ;
    wire not_B ;
    wire and1 ;
    wire and2 ;
    supply1 VDD ;
    supply0 VSS ;

    not_gate_1 not_inst_A ( .A ( A ) , .Y ( not_A ) , .VDD ( VDD ) ,
        .VSS ( VSS ) ) ;
    not_gate_0 not_inst_B ( .A ( B ) , .Y ( not_B ) , .VDD ( VDD ) ,
        .VSS ( VSS ) ) ;
    and_gate_1 and_inst1 ( .A ( A ) , .B ( not_B ) , .Y ( and1 ) , .
       VDD ( VDD ) ,
        .VSS ( VSS ) ) ;
    and_gate_0 and_inst2 ( .A ( not_A ) , .B ( B ) , .Y ( and2 ) , .
       VDD ( VDD ) ,
        .VSS ( VSS ) ) ;
    or_gate or_inst ( .A ( and1 ) , .B ( and2 ) , .Y ( Y ) , .VDD (
       VDD ) ,
        .VSS ( VSS ) ) ;
    PCORNER CORNER1 () ;
    PCORNER CORNER2 () ;
    PCORNER CORNER3 () ;
    PCORNER CORNER4 () ;
    PVDD1CDG PVDD1 ( .VDD ( VDD ) ) ;
    PVSS1CDG PVSS1 ( .VSS ( VSS ) ) ;
    PFILLER20 __added_filler_instance_0 () ;
    PFILLER20 __added_filler_instance_1 () ;
    PFILLER20 __added_filler_instance_2 () ;
    .
    .
    .
    endmodule
```

Cuadro 89: Verilog del *core* de un *XOR* obtenido luego de la síntesis física

El archivo principal `main.sp` empleado para la simulación se encuentra en el Cuadro 90.

```
*Comentario inicial
* Se incluyen los archivos .sp seccionados

.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/nueva_integracion/sintesis_fisica_xor_noIO/Outputs/LPE/
   DECKS/misDecks.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/nueva_integracion/sintesis_fisica_xor_noIO/Outputs/LPE/
   DECKS/lib_180nm.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/nueva_integracion/sintesis_fisica_xor_noIO/Outputs/LPE/
   DECKS/Params.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/nueva_integracion/sintesis_fisica_xor_noIO/Outputs/LPE/
   DECKS/Params_clks.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/Rui/
   TSMC/nueva_integracion/sintesis_fisica_xor_noIO/Outputs/LPE/
   xor3.spf'

* Fuente de alimentacion
v1 vdd 0 1.5

* Senal de reloj que sera inyectada a cada entrada del circuito.
V_A32 clk32 gnd pulse (0 vdd1 td tr tf pwA32 per32 )
V_A16 clk16 gnd pulse (0 vdd1 td tr tf pwA16 per16 )
V_A8 A gnd pulse (0 vdd1 td tr tf pwA8 per8 )
V_A4 B gnd pulse (0 vdd1 td tr tf pwA4 per4 )
V_A2 clk2 gnd pulse (0 vdd1 td tr tf pwA2 per2 )
V_A1 clk1 gnd pulse (0 vdd1 td tr tf pwA1 per1 )

* XNOT A Y vdd Not
XOR A B Y vdd xor_using_basic_gates



*Configuracion del analisis transitorio
.TRAN 1e-9 32.1u
.Option post
.PROBE v(B) v(A) v(Y)
.END
```

Cuadro 90: Configuración del archivo `main.sp` para una compuerta XOR

11.4. Sumador de 4 *bits*

El último circuito sometido a pruebas funcionales en *HSPICE* fue un sumador de 4 *bits carry-look-ahead*. Este diseño se eligió con el propósito de incrementar la complejidad en el proceso de mapeo de pines y evaluar el procedimiento en un circuito más avanzado.

Es importante mencionar que, al igual que en el caso de la compuerta *XOR*, los resultados

iniciales obtenidos utilizando el *deck* con elementos parásitos mostraron comportamientos inconsistentes, lo que dificultó la validación del funcionamiento del circuito. Para solucionar este problema y centrarse exclusivamente en el *core* del circuito, se optó por eliminar todas las resistencias y capacitancias parásitas del *deck*, lo cual permitió realizar simulaciones más confiables. El resultado final, tras esta depuración del *deck*, se presenta en el Cuadro 91, demostrando un comportamiento alineado con lo esperado para un sumador *carry-look-ahead* de 4 *bits*.

```
*
*|DSPF 1.3
*|DESIGN carry_look_ahead_4bit
*|DATE "Wed Nov 13 11:57:05 2024"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "V-2023.12-SP1"
*|DIVIDER /
*|DELIMITER :
**FORMAT SPF
*

** COMMENTS

** OPERATING_TEMPERATURE 25
** GLOBAL_TEMPERATURE 25
**
** TCAD_GRD_FILE /home/nanoelectronica/Desktop/Folder_de_Trabajo
   /Rui/TSMC/nueva_integracion/sintesis_fisica_adder4b_noIO/
   Outputs/LPE/cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
**   TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
**   TCADGRD_VERSION 62

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

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

```
    CKXOR2D0BWP7T
XI_D4A6F1246 b0 a0 n14 I_D4A6F1246:N_5 I_D4A6F1246:N_6 vsource
    CKXOR2D0BWP7T
XI_D4A6F1247 b2 a2 n10 n11 n8 I_D4A6F1247:N_7 I_D4A6F1247:N_8
    vsource AOI22D0BWP7T
XI_D4A6F1248 n14 cin a0 b0 n12 I_D4A6F1248:N_7 I_D4A6F1248:N_8
    vsource AOI22D0BWP7T
XI_D4A6F1249 N_4 I_D4A6F1249:N_3 I_D4A6F1249:N_4 I_D4A6F1249:N_5
     PVSS1CDG

.ENDS carry_look_ahead_4bit
```

Cuadro 91: Archivo `.spf` con pines mapeados para el *core* de un sumador de 4 *bits*

```
// IC Compiler II Version V-2023.12 Verilog Writer
// Generated on 11/13/2024 at 10:40:49
// Library Name: LIB_TEST
// Block Name: carry_look_ahead_4bit
// User Label:
// Write Command: write_verilog -include { all } ./Outputs/
   IO/carry_look_ahead_4bit.v
module carry_look_ahead_4bit ( a , b , cin , S , cout , VDD
    , VSS ) ;
input  [3:0] a ;
input  [3:0] b ;
input  cin ;
output [3:0] S ;
output cout ;
input  VDD ;
input  VSS ;

wire n8 ;
wire n9 ;
wire n10 ;
wire n11 ;
wire n12 ;
wire n13 ;
wire n14 ;
supply1 VDD ;
supply0 VSS ;

MOAI22D0BWP7T U13 ( .A1 ( n8 ) , .A2 ( n9 ) , .B1 ( b[3] ) ,
    .B2 ( a[3] ) ,
    .ZN ( cout ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
CKXOR2D0BWP7T U14 ( .A1 ( n8 ) , .A2 ( n9 ) , .Z ( S[3] ) ,
   .VDD ( VDD ) ,
    .VSS ( VSS ) ) ;
XNR2D1BWP7T U15 ( .A1 ( b[3] ) , .A2 ( a[3] ) , .ZN ( n9 ) ,
    .VDD ( VDD ) ,
    .VSS ( VSS ) ) ;
AOI22D0BWP7T U16 ( .A1 ( b[2] ) , .A2 ( a[2] ) , .B1 ( n10 )
    , .B2 ( n11 ) ,
    .ZN ( n8 ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
```

```
        CKXOR2D0BWP7T U17 ( .A1 ( n11 ) , .A2 ( n10 ) , .Z ( S[2] )
            , .VDD ( VDD ) ,
             .VSS ( VSS ) ) ;
        MOAI22D0BWP7T U18 ( .A1 ( n12 ) , .A2 ( n13 ) , .B1 ( b[1] )
             , .B2 ( a[1] ) ,
             .ZN ( n10 ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
        CKXOR2D0BWP7T U19 ( .A1 ( b[2] ) , .A2 ( a[2] ) , .Z ( n11 )
             , .VDD ( VDD ) ,
             .VSS ( VSS ) ) ;
        CKXOR2D0BWP7T U20 ( .A1 ( n12 ) , .A2 ( n13 ) , .Z ( S[1] )
            , .VDD ( VDD ) ,
             .VSS ( VSS ) ) ;
        XNR2D1BWP7T U21 ( .A1 ( b[1] ) , .A2 ( a[1] ) , .ZN ( n13 )
            , .VDD ( VDD ) ,
             .VSS ( VSS ) ) ;
        AOI22D0BWP7T U22 ( .A1 ( n14 ) , .A2 ( cin ) , .B1 ( a[0] )
            , .B2 ( b[0] ) ,
             .ZN ( n12 ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
        CKXOR2D0BWP7T U23 ( .A1 ( cin ) , .A2 ( n14 ) , .Z ( S[0] )
            , .VDD ( VDD ) ,
             .VSS ( VSS ) ) ;
        CKXOR2D0BWP7T U24 ( .A1 ( b[0] ) , .A2 ( a[0] ) , .Z ( n14 )
             , .VDD ( VDD ) ,
             .VSS ( VSS ) ) ;
        PCORNER CORNER1 () ;
        PCORNER CORNER2 () ;
        PCORNER CORNER3 () ;
        PCORNER CORNER4 () ;
        PVDD1CDG PVDD1 ( .VDD ( VDD ) ) ;
        PVSS1CDG PVSS1 ( .VSS ( VSS ) ) ;
        PFILLER20 __added_filler_instance_0 () ;
        PFILLER20 __added_filler_instance_1 () ;
        PFILLER20 __added_filler_instance_2 () ;
        PFILLER20 __added_filler_instance_3 () ;
        PFILLER20 __added_filler_instance_4 () ;
        .
        .
        .
        endmodule
```

Cuadro 92: Verilog del *core* de un sumador de 4 *bits* obtenido luego de la síntesis física

Luego de realizar la simulación utilizando el archivo principal presentado en el Cuadro 93, se obtuvieron los resultados mostrados en la Figura 81. Este sumador *carry-look-ahead* toma como entradas dos números de 4 *bits*, denominados `a[3:0]` y `b[3:0]`, y genera una salida de 4 *bits*, `s[3:0]`, que representa la suma de ambas entradas. Para simplificar las pruebas realizadas, los *bits* más significativos de cada entrada (`a3` y `b3`) fueron configurados en 0, permitiendo enfocar el análisis en los *bits* menos significativos del circuito.

```
        *Comentario inicial
        * Se incluyen los archivos .sp seccionados

        .include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/
```

```
        Rui/TSMC/nueva_integracion/sintesis_fisica_adder4b_noIO/
        Outputs/LPE/DECKS/misDecks.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/
        Rui/TSMC/nueva_integracion/sintesis_fisica_adder4b_noIO/
        Outputs/LPE/DECKS/lib_180nm.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/
        Rui/TSMC/nueva_integracion/sintesis_fisica_adder4b_noIO/
        Outputs/LPE/DECKS/Params.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/
        Rui/TSMC/nueva_integracion/sintesis_fisica_adder4b_noIO/
        Outputs/LPE/DECKS/Params_clks.sp'
.include '/home/nanoelectronica/Desktop/Folder_de_Trabajo/
        Rui/TSMC/nueva_integracion/sintesis_fisica_adder4b_noIO/
        Outputs/LPE/Add4b.spf'

* Power supplies
VDD VDD 0 DC 1.5
VSS VSS 0 DC 0


* Senal de reloj que sera inyectada a cada entrada del
        circuito.
V_A32 a0 gnd pulse (0 vdd1 td tr tf pwA32 per32 )
V_A16 b2 gnd pulse (0 vdd1 td tr tf pwA16 per16 )
V_A8 b1 gnd pulse (0 vdd1 td tr tf pwA8 per8 )
V_A4 a1 gnd pulse (0 vdd1 td tr tf pwA4 per4 )
V_A2 b0 gnd pulse (0 vdd1 td tr tf pwA2 per2 )
V_A1 a2 gnd pulse (0 vdd1 td tr tf pwA1 per1 )


* Instancia del subcircuito carry_look_ahead_4bit
Xadder a0 a1 a2 0 b0 b1 b2 0 0 VDD s0 s1 s2 s3 cout
        carry_look_ahead_4bit


*Configuracion del analisis transitorio
.TRAN 1e-9 32.1u
.Option post

.END
```

Cuadro 93: Configuración del archivo `main.sp` para un sumador de 4 *bits*

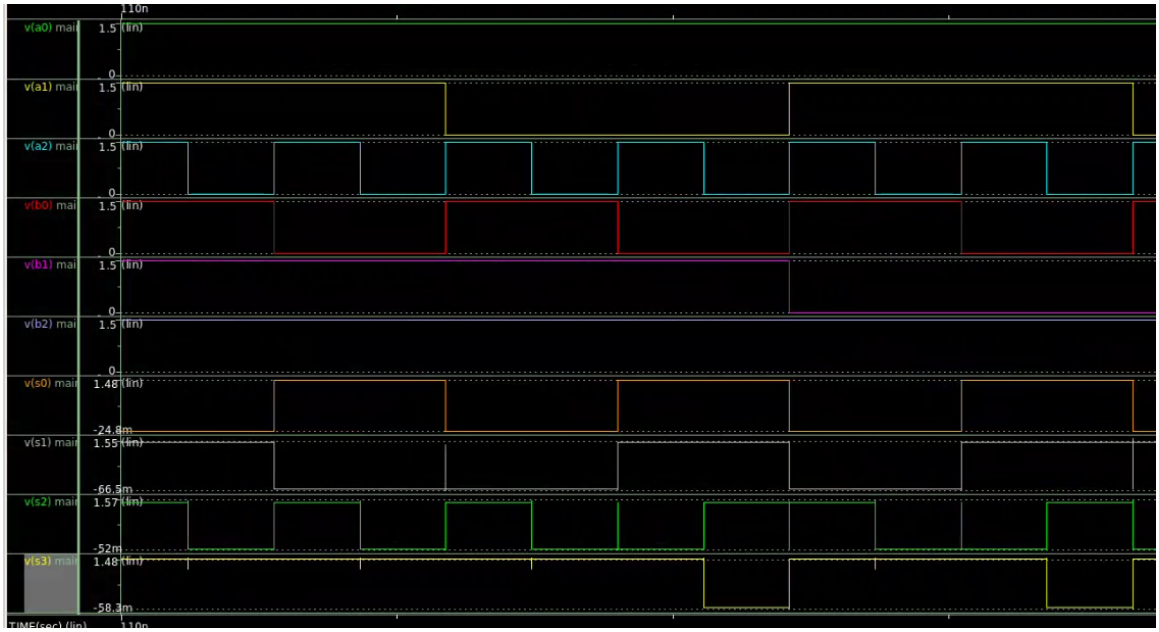


Figura 81: Simulación final de un sumador de 4 *bits* utilizando un *deck* simplificado que incluye únicamente el *core* del circuito

Un ejemplo concreto del funcionamiento se observa en la gráfica para el caso en que las entradas son `a = 0111` y `b = 0111`. Esto corresponde a los valores binarios:

- `a = 0111`: `a0 = 1`, `a1 = 1`, `a2 = 1`, `a3 = 0`
- `b = 0111`: `b0 = 1`, `b1 = 1`, `b2 = 1`, `b3 = 0`

Esto significa que `s0 = 0`, `s1 = 1`, `s2 = 1`, y `s3 = 1`. En la simulación, las señales observadas en las salidas `s[3:0]` coinciden con este resultado esperado, lo que valida el correcto funcionamiento del circuito.

CAPÍTULO 12

Conclusiones

- Se implementaron mejoras en el flujo de síntesis física en las librerías de TSMC, lo que permitió la automatización adicional de funciones como la síntesis de reloj y la definición de dimensiones.
- Se realizó la síntesis física de circuitos tanto de naturaleza combinacional y secuencial, así como del circuito "El Gran Jaguar" en las librerías de TSMC.
- Se llevó a cabo la verificación física *DRC* utilizando el *runset* de relleno de metal proporcionado por *TSMC*, obteniendo resultados libres de errores de densidad en la mayoría de los diseños. Sin embargo, el circuito "El Gran Jaguar" redujo sus errores a un único caso en Metal 2, donde la densidad aumentó de 25.97 % a 29.38 %.
- Se llevó a cabo la verificación de antena para múltiples circuitos, en los cuales se obtuvo resultados satisfactorios.
- Mediante el uso de la herramienta *Library Manager* de IC Compiler II, se logró crear las bibliotecas de diseño de TSMC utilizadas en el flujo de síntesis física. Todo el proceso se realizó con el objetivo de documentar los resultados y asegurar la calidad de las bibliotecas creadas con archivos de tecnologías futuras.
- Se realizó la generación del archivo Verilog de múltiples circuitos con IC Compiler II y, tras ejecutar el *testbench*, se concluyó que el *verilog* no es una representación exacta del *layout* del circuito, ya que aunque las conexiones lógicas se mantuvieron consistentes en las pruebas, la disposición y conexiones físicas de los componentes no influye en los resultados funcionales del diseño.
- Se llevó a cabo el mapeo de pines en el *deck* generado tras la extracción parásita. Posteriormente, se validó el funcionamiento del *core* de circuitos como compuertas *NOT* y *XOR*, así como un sumador de 4 bits, utilizando *HSPICE* y visualizando su comportamiento por medio de *WaveView*.
- Se mantuvo comunicación constante con el equipo de síntesis lógica para elaborar un manual detallado que describe cada paso del flujo de síntesis lógica, física y las

verificaciones físicas en las librerías de TSMC, complementado con videos explicativos sobre el uso de las herramientas.

CAPÍTULO 13

Recomendaciones

- Replicar el proceso actual y familiarizarse con la estructura y funcionamiento para minimizar la curva de aprendizaje frente a las herramientas y conceptos que se presenten. Se sugiere iniciar con el diseño de circuitos simples, tanto combinacionales como secuenciales, para luego incrementar gradualmente la complejidad, lo que permitirá un dominio más fluido de las técnicas y herramientas utilizadas en el proyecto.
- Emplear *SolvNet* para obtener documentación como manuales, guías de usuario, y mejores prácticas de las herramientas a utilizar y leerlas antes de iniciar a trabajar, asegurando estar al tanto de cualquier actualización o cambio importante en los flujos de trabajo.
- Investigar sobre la optimización del proceso de diseño para desarrollar un flujo de síntesis física que se auto-gestione y sea capaz de calcular automáticamente las dimensiones del diseño en función de características específicas, tales como tamaño, número y tipo de puertos de entrada y salida, frecuencia y uso del reloj, entre otras, por medio de IC Compiler II u otras herramientas de Synopsys.
- Iniciar comunicación con los equipos de IMEC y TSMC desde principios de año permitirá agilizar la obtención de cualquier documentación necesaria durante el desarrollo del trabajo de graduación.
- Establecer un flujo de trabajo colaborativo que integre de manera efectiva a los equipos de síntesis lógica y física, con reuniones periódicas para revisión de avances y asegurar que ambos trabajen en sintonía.
- Automatizar el proceso de mapeo de pines tras la extracción parásita mediante un algoritmo, y complementar este flujo con la creación de *decks* para las celdas que aún no han sido definidas.
- Llevar a cabo la documentación en paralelo a la elaboración del trabajo de graduación. Esto facilita la organización de ideas, asegura la inclusión de detalles relevantes que pueden ser fácilmente olvidados si se dejan para el final y optimiza el tiempo de redacción.

CAPÍTULO 14

Bibliografía

[1] J. D. los Santos, "Diseño de un sumador/restador completo de 32 bits con tecnología CMOS en un proceso de 28 nanómetros usando aplicaciones de diseño de la empresa Synopsys," *UVG*, 2014.

[2] S. Rubio, "Definición del Flujo de Diseño para Fabricación de un Chip con Tecnologia VLSI CMOS," *UVG*, 2019.

[3] L. Nájera, "Implementación de circuitos sintetizados a nivel netlist a partir de un diseño en lenguaje descriptivo de hardware como primer paso en el flujo de diseño de un circuito integrado.," *UVG*, 2019.

[4] M. Illescas, "Verificación de reglas de diseño (DRC) para el desarrollo de un flujo funcional de un circuito integrado con tecnología nanométrica," *UVG*, 2020.

[5] M. Flores, "Corrección de anillo de entradas/salidas y pruebas de antenna y ERC para la definición del flujo de diseño del primer chip con tecnología nanométrica desarrollado en Guatemal," *UVG*, 2020.

[6] J. Girón, "Etapa de verificación física de Diseño en Silicio vs. Esquemático (LVS) en el flujo de diseño para un chip a nanoescala," *UVG*, 2020.

[7] J. Ruano, "Definición del flujo en la herramienta VCS para la simulación de HDLs en la Fabricación de un Chip con Tecnología Nanométrica CMOS," *UVG*, 2020.

[8] J. Ayala, "Diseño de un Circuito Integrado con Tecnología de 180 nm Usando Librerías de Diseño de TSMC: Ejecución de la Síntesis Física, Verificaciones de Antena y Corrección de Errores Obtenido," *UVG*, 2021.

[9] J. Ruiz, "Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: ejecución de la fase de verificación física Layout vs Schematic (LVS)," *UVG*, 2021.

[10] A. Altuna, "Diseño de un Circuito Integrado con Tecnología de 180nm usando Librerías de Diseño de TSMC: Ejecución de la Síntesis Física, Verificación de Reglas de Diseño y Corrección de Errores Obtenidos.," *UVG*, 2021.

[11] J. Shin, "Diseño de un Circuito Integrado con Tecnología de 180nm usando Librerías de Diseño de TSMC: Ejecución de la Síntesis Física, Verificación de Reglas de Diseño y Corrección de Errores Obtenidos.," *UVG*, 2021.

[12] L. Abadía, "Posicionamiento e interconexión entre componentes de un circuito sintetizado para el flujo de diseño de un circuito a escala nanométrica utilizando la herramienta de IC Compiler," *UVG*, 2021.

[13] E. Torres, "Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: Ejecución y simulación para la etapa de síntesis lógica," *UVG*, 2021.

[14] S. Cardona, "Mejoramiento del proceso de síntesis lógica llevada a cabo para la elaboración de un circuito integrado a escala nanométrica," *UVG*, 2021.

[15] A. Aguilar, "Mejoramiento del proceso de síntesis lógica llevada a cabo para la elaboración de un circuito integrado a escala nanométrica," *UVG*, 2022.

[16] D. Equité, "Diseño e implementación de interfaz gráfica de la automatización de las fases de diseño de un circuito integrado y uso avanzado de IC Compiler II.," *UVG*, 2022.

[17] J. Ponce, "Diseño de un circuito integrado con tecnología de 180 nm utilizando librerías de diseño de TSMC: Implementación de un alternativo flujo de diseño proporcionado por Synopsys con las herramientas PrimeTime y TetraMAX.," *UVG*, 2022.

[18] S. Schwendener, "Automatización de las verificaciones físicas de un circuito integrado con tecnología de 180 nm utilizando librerías de diseño de TSMC," *UVG*, 2022.

[19] P. Mendizábal, "Automatización del proceso de instalación de software de Synopsys e implementación de mejoras utilizando contenedores," *UVG*, 2022.

[20] N. Prado, "Diseño de un circuito integrado con tecnología de 180 nm, utilizando las librerías de diseño de TSMC y las librerías educativas de Synopsys: corrección de errores de densidad y polisilicio, verificación DRC y antena.," *UVG*, 2023.

[21] Y. T. Tak H. Ning, *Fundamentals of Modern VLSI Devices*. Cambridge University Press, 2022, 3rd Edition.

[22] H. R. Wim Dehaene, *Efficient Design of Variation-Resilient Ultra-Low Energy Digital Processors*. Springer, 2019.

[23] J. L. Andrew B. Kahng, *VLSI Physical Design - From Graph Partitioning to Timing Closure*. Springer, 2011.

[24] D. M. H. Neil H. E. Weste, *CMOS VLSI Design: a circuits and systems perspective*. Pearson, 2010.

[25] R. S. S. Prashant Saxena, *Routing Congestion in VLSI Circuits: Estimation and Optimization*. Springer, 2007.

[26] K. Priyadarshi, *What are the 5 Steps Involved in Physical Design of VLSI Chips?* 2023. dirección: https://techovedas.com/what-are-the-5-steps-involved-in-physical-design-of-vlsi-chips/.

[27] Synopsys, *What is Design Rule Checking (DRC)?* Dirección: https://www.synopsys.com/glossary/what-is-design-rule-checking.html.

[28] H. Kaeslin, *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication*. Cambridge University Press, 2008.

[29] Synopsys, *Verification Continuum VCS UserGuide*, 2023.

[30] Synopsys, *IC Compiler II Implementation UserGuide*, 2023.

[31] Synopsys, *IC Validator Functional Safety for Implementation User Guide*, 2023.

[32] Synopsys, *StarRC Parasitic Explorer User Guide*, 2023.

[33] D. Alvarado, "Diseño de un circuito integrado con tecnología de 65 nm utilizando librerías de diseño de TSMC: Pruebas de LVS, ERC y extracción de parásitos," *UVG*, 2024.

[34] I. Arévalo, "Diseño de un circuito integrado con tecnología de 180 nm utilizando librerías de diseño de TSMC: Simulación en PrimeSim y análisis de errores.," *UVG*, 2022.

[35] C. Letona, "Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: uso avanzado de StarRC para la generación de un archivo HSPICE con componentes parásitos para su correcta simulación," *UVG*, 2022.

CAPÍTULO 15

Anexos

15.1. Verilog obtenido desde IC Compiler II

```
IC Compiler II Version V-2023.12 Verilog Writer
Generated on 8/24/2024 at 17:36:34
Library Name: LIB_TEST
Block Name: Not_IO
User Label:
Write Command: write_verilog -include { all } ./Outputs/IO/
   VERILOG.v
module Not ( A , Y , VDD , VSS ) ;
input  A ;
output Y ;
input  VDD ;
input  VSS ;

supply1 VDD ;
supply0 VSS ;

CKND0BWP7T U1 ( .I ( A ) , .ZN ( Y ) , .VDD ( VDD ) , .VSS ( VSS
    ) ) ;
endmodule


module Not_IO ( A , Y , VDD , VSS ) ;
input  A ;
output Y ;
input  VDD ;
input  VSS ;

wire A_w ;
wire Y_w ;
wire n1 ;
wire n2 ;
```

```
    supply1 VDD ;
    supply0 VSS ;
    wire SYNOPSYS_UNCONNECTED_1 ;

    Not Compuerta ( .A ( A_w ) , .Y ( Y_w ) , .VDD ( VDD ) , .VSS (
       VSS ) ) ;
    PDDW0204SCDG U10 ( .I ( n1 ) , .OEN ( n2 ) , .IE ( n2 ) , .PAD (
        A ) ,
        .DS ( n1 ) , .PE ( n1 ) , .C ( A_w ) ) ;
    PDDW0204SCDG U11 ( .I ( Y_w ) , .OEN ( n1 ) , .IE ( n2 ) , .PAD
       ( Y ) ,
        .DS ( n1 ) , .PE ( n1 ) , .C ( SYNOPSYS_UNCONNECTED_1 ) ) ;
    TIELBWP7T U6 ( .ZN ( n1 ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
    TIEHBWP7T U7 ( .Z ( n2 ) , .VDD ( VDD ) , .VSS ( VSS ) ) ;
    PCORNER CORNER1 () ;
    PCORNER CORNER2 () ;
    PCORNER CORNER3 () ;
    PCORNER CORNER4 () ;
    PVDD1CDG PVDD1 ( .VDD ( VDD ) ) ;
    PVSS1CDG PVSS1 ( .VSS ( VSS ) ) ;
    PFILLER20 __added_filler_instance_0 () ;
    PFILLER5 __added_filler_instance_1 () ;
    PFILLER1 __added_filler_instance_2 () ;
    PFILLER1 __added_filler_instance_3 () ;
    PFILLER1 __added_filler_instance_4 () ;
    PFILLER1 __added_filler_instance_5 () ;
    PFILLER0005 __added_filler_instance_6 () ;
    PFILLER0005 __added_filler_instance_7 () ;
    PFILLER0005 __added_filler_instance_8 () ;
    PFILLER0005 __added_filler_instance_9 () ;
    PFILLER0005 __added_filler_instance_10 () ;
    PFILLER0005 __added_filler_instance_11 () ;
    PFILLER0005 __added_filler_instance_12 () ;
    PFILLER0005 __added_filler_instance_13 () ;
    PFILLER0005 __added_filler_instance_14 () ;
    PFILLER0005 __added_filler_instance_15 () ;
    PFILLER0005 __added_filler_instance_16 () ;
    PFILLER0005 __added_filler_instance_17 () ;
    PFILLER0005 __added_filler_instance_18 () ;
    PFILLER0005 __added_filler_instance_19 () ;
    PFILLER0005 __added_filler_instance_20 () ;
    PFILLER0005 __added_filler_instance_21 () ;
    PFILLER0005 __added_filler_instance_22 () ;
    PFILLER0005 __added_filler_instance_23 () ;
    PFILLER0005 __added_filler_instance_24 () ;
    PFILLER0005 __added_filler_instance_25 () ;
    PFILLER0005 __added_filler_instance_26 () ;
    PFILLER0005 __added_filler_instance_27 () ;
    PFILLER0005 __added_filler_instance_28 () ;
    PFILLER0005 __added_filler_instance_29 () ;
    PFILLER0005 __added_filler_instance_30 () ;
    PFILLER0005 __added_filler_instance_31 () ;
    PFILLER0005 __added_filler_instance_32 () ;
    PFILLER0005 __added_filler_instance_33 () ;
```

```
    PFILLER0005 __added_filler_instance_34 () ;
    PFILLER0005 __added_filler_instance_35 () ;
    PFILLER0005 __added_filler_instance_36 () ;
    PFILLER0005 __added_filler_instance_37 () ;
    PFILLER0005 __added_filler_instance_38 () ;
    PFILLER0005 __added_filler_instance_39 () ;
    PFILLER0005 __added_filler_instance_40 () ;
    PFILLER0005 __added_filler_instance_41 () ;
    PFILLER0005 __added_filler_instance_42 () ;
    PFILLER0005 __added_filler_instance_43 () ;
    PFILLER0005 __added_filler_instance_44 () ;
    PFILLER0005 __added_filler_instance_45 () ;
    PFILLER0005 __added_filler_instance_46 () ;
    PFILLER0005 __added_filler_instance_47 () ;
    PFILLER0005 __added_filler_instance_48 () ;
    PFILLER0005 __added_filler_instance_49 () ;
    PFILLER0005 __added_filler_instance_50 () ;
    PFILLER0005 __added_filler_instance_51 () ;
    PFILLER0005 __added_filler_instance_52 () ;
    PFILLER0005 __added_filler_instance_53 () ;
    PFILLER0005 __added_filler_instance_54 () ;
    PFILLER0005 __added_filler_instance_55 () ;
    PFILLER0005 __added_filler_instance_56 () ;
    PFILLER0005 __added_filler_instance_57 () ;
    PFILLER0005 __added_filler_instance_58 () ;
    PFILLER0005 __added_filler_instance_59 () ;
    PFILLER0005 __added_filler_instance_60 () ;
    PFILLER0005 __added_filler_instance_61 () ;
    PFILLER0005 __added_filler_instance_62 () ;
    PFILLER0005 __added_filler_instance_63 () ;
    PFILLER0005 __added_filler_instance_64 () ;
    PFILLER0005 __added_filler_instance_65 () ;
    PFILLER0005 __added_filler_instance_66 () ;
    PFILLER0005 __added_filler_instance_67 () ;
    PFILLER0005 __added_filler_instance_68 () ;
    PFILLER0005 __added_filler_instance_69 () ;
    PFILLER0005 __added_filler_instance_70 () ;
    PFILLER0005 __added_filler_instance_71 () ;
    PFILLER0005 __added_filler_instance_72 () ;
    PFILLER0005 __added_filler_instance_73 () ;
    PFILLER0005 __added_filler_instance_74 () ;
    PFILLER0005 __added_filler_instance_75 () ;
    PFILLER0005 __added_filler_instance_76 () ;
    PFILLER0005 __added_filler_instance_77 () ;
    PFILLER0005 __added_filler_instance_78 () ;
    PFILLER0005 __added_filler_instance_79 () ;
    PFILLER0005 __added_filler_instance_80 () ;
    PFILLER0005 __added_filler_instance_81 () ;
    PFILLER0005 __added_filler_instance_82 () ;
    PFILLER0005 __added_filler_instance_83 () ;
    PFILLER0005 __added_filler_instance_84 () ;
    PFILLER0005 __added_filler_instance_85 () ;
    PFILLER0005 __added_filler_instance_86 () ;
    PFILLER0005 __added_filler_instance_87 () ;
```

```
    PFILLER0005 __added_filler_instance_88 () ;
    PFILLER0005 __added_filler_instance_89 () ;
    PFILLER0005 __added_filler_instance_90 () ;
    PFILLER0005 __added_filler_instance_91 () ;
    PFILLER0005 __added_filler_instance_92 () ;
    PFILLER0005 __added_filler_instance_93 () ;
    PFILLER0005 __added_filler_instance_94 () ;
    PFILLER0005 __added_filler_instance_95 () ;
    PFILLER0005 __added_filler_instance_96 () ;
    PFILLER0005 __added_filler_instance_97 () ;
    PFILLER0005 __added_filler_instance_98 () ;
    PFILLER0005 __added_filler_instance_99 () ;
    PFILLER0005 __added_filler_instance_100 () ;
    PFILLER0005 __added_filler_instance_101 () ;
    PFILLER0005 __added_filler_instance_102 () ;
    PFILLER0005 __added_filler_instance_103 () ;
    PFILLER0005 __added_filler_instance_104 () ;
    PFILLER0005 __added_filler_instance_105 () ;
    PFILLER0005 __added_filler_instance_106 () ;
    PFILLER0005 __added_filler_instance_107 () ;
    PFILLER0005 __added_filler_instance_108 () ;
    PFILLER0005 __added_filler_instance_109 () ;
    PFILLER0005 __added_filler_instance_110 () ;
    PFILLER0005 __added_filler_instance_111 () ;
    PFILLER0005 __added_filler_instance_112 () ;
    PFILLER0005 __added_filler_instance_113 () ;
    PFILLER0005 __added_filler_instance_114 () ;
    PFILLER0005 __added_filler_instance_115 () ;
    PFILLER0005 __added_filler_instance_116 () ;
    PFILLER0005 __added_filler_instance_117 () ;
    PFILLER0005 __added_filler_instance_118 () ;
    PFILLER0005 __added_filler_instance_119 () ;
    PFILLER0005 __added_filler_instance_120 () ;
    PFILLER0005 __added_filler_instance_121 () ;
    PFILLER0005 __added_filler_instance_122 () ;
    PFILLER0005 __added_filler_instance_123 () ;
    PFILLER0005 __added_filler_instance_124 () ;
    PFILLER0005 __added_filler_instance_125 () ;
    PFILLER0005 __added_filler_instance_126 () ;
    PFILLER0005 __added_filler_instance_127 () ;
    PFILLER0005 __added_filler_instance_128 () ;
    PFILLER0005 __added_filler_instance_129 () ;
    PFILLER0005 __added_filler_instance_130 () ;
    PFILLER0005 __added_filler_instance_131 () ;
    PFILLER0005 __added_filler_instance_132 () ;
    PFILLER0005 __added_filler_instance_133 () ;
    PFILLER0005 __added_filler_instance_134 () ;
    PFILLER0005 __added_filler_instance_135 () ;
    PFILLER0005 __added_filler_instance_136 () ;
    PFILLER0005 __added_filler_instance_137 () ;
    PFILLER0005 __added_filler_instance_138 () ;
    PFILLER0005 __added_filler_instance_139 () ;
    PFILLER0005 __added_filler_instance_140 () ;
    PFILLER0005 __added_filler_instance_141 () ;
```

```
    PFILLER0005 __added_filler_instance_142 () ;
    PFILLER0005 __added_filler_instance_143 () ;
    PFILLER0005 __added_filler_instance_144 () ;
    PFILLER0005 __added_filler_instance_145 () ;
    PFILLER0005 __added_filler_instance_146 () ;
    PFILLER0005 __added_filler_instance_147 () ;
    PFILLER0005 __added_filler_instance_148 () ;
    PFILLER0005 __added_filler_instance_149 () ;
    PFILLER0005 __added_filler_instance_150 () ;
    PFILLER0005 __added_filler_instance_151 () ;
    PFILLER0005 __added_filler_instance_152 () ;
    PFILLER0005 __added_filler_instance_153 () ;
    PFILLER0005 __added_filler_instance_154 () ;
    PFILLER0005 __added_filler_instance_155 () ;
    PFILLER0005 __added_filler_instance_156 () ;
    PFILLER0005 __added_filler_instance_157 () ;
    PFILLER0005 __added_filler_instance_158 () ;
    PFILLER0005 __added_filler_instance_159 () ;
    PFILLER0005 __added_filler_instance_160 () ;
    PFILLER0005 __added_filler_instance_161 () ;
    PFILLER0005 __added_filler_instance_162 () ;
    PFILLER0005 __added_filler_instance_163 () ;
    PFILLER0005 __added_filler_instance_164 () ;
    PFILLER0005 __added_filler_instance_165 () ;
    PFILLER0005 __added_filler_instance_166 () ;
    PFILLER0005 __added_filler_instance_167 () ;
    PFILLER0005 __added_filler_instance_168 () ;
    PFILLER0005 __added_filler_instance_169 () ;
    PFILLER0005 __added_filler_instance_170 () ;
    PFILLER0005 __added_filler_instance_171 () ;
    PFILLER0005 __added_filler_instance_172 () ;
    PFILLER0005 __added_filler_instance_173 () ;
    PFILLER20 __added_filler_instance_174 () ;
    PFILLER5 __added_filler_instance_175 () ;
    PFILLER1 __added_filler_instance_176 () ;
    PFILLER1 __added_filler_instance_177 () ;
    PFILLER1 __added_filler_instance_178 () ;
    PFILLER1 __added_filler_instance_179 () ;
    PFILLER0005 __added_filler_instance_180 () ;
    PFILLER0005 __added_filler_instance_181 () ;
    PFILLER0005 __added_filler_instance_182 () ;
    PFILLER0005 __added_filler_instance_183 () ;
    PFILLER0005 __added_filler_instance_184 () ;
    PFILLER0005 __added_filler_instance_185 () ;
    PFILLER0005 __added_filler_instance_186 () ;
    PFILLER0005 __added_filler_instance_187 () ;
    PFILLER0005 __added_filler_instance_188 () ;
    PFILLER0005 __added_filler_instance_189 () ;
    PFILLER0005 __added_filler_instance_190 () ;
    PFILLER0005 __added_filler_instance_191 () ;
    PFILLER0005 __added_filler_instance_192 () ;
    PFILLER0005 __added_filler_instance_193 () ;
    PFILLER0005 __added_filler_instance_194 () ;
    PFILLER0005 __added_filler_instance_195 () ;
```

```
    PFILLER0005 __added_filler_instance_196 () ;
    PFILLER0005 __added_filler_instance_197 () ;
    PFILLER0005 __added_filler_instance_198 () ;
    PFILLER0005 __added_filler_instance_199 () ;
    PFILLER0005 __added_filler_instance_200 () ;
    PFILLER0005 __added_filler_instance_201 () ;
    PFILLER0005 __added_filler_instance_202 () ;
    PFILLER0005 __added_filler_instance_203 () ;
    PFILLER0005 __added_filler_instance_204 () ;
    PFILLER0005 __added_filler_instance_205 () ;
    PFILLER0005 __added_filler_instance_206 () ;
    PFILLER0005 __added_filler_instance_207 () ;
    PFILLER0005 __added_filler_instance_208 () ;
    PFILLER0005 __added_filler_instance_209 () ;
    PFILLER0005 __added_filler_instance_210 () ;
    PFILLER0005 __added_filler_instance_211 () ;
    PFILLER0005 __added_filler_instance_212 () ;
    PFILLER0005 __added_filler_instance_213 () ;
    PFILLER0005 __added_filler_instance_214 () ;
    PFILLER0005 __added_filler_instance_215 () ;
    PFILLER0005 __added_filler_instance_216 () ;
    PFILLER0005 __added_filler_instance_217 () ;
    PFILLER0005 __added_filler_instance_218 () ;
    PFILLER0005 __added_filler_instance_219 () ;
    PFILLER0005 __added_filler_instance_220 () ;
    PFILLER0005 __added_filler_instance_221 () ;
    PFILLER0005 __added_filler_instance_222 () ;
    PFILLER0005 __added_filler_instance_223 () ;
    PFILLER0005 __added_filler_instance_224 () ;
    PFILLER0005 __added_filler_instance_225 () ;
    PFILLER0005 __added_filler_instance_226 () ;
    PFILLER0005 __added_filler_instance_227 () ;
    PFILLER0005 __added_filler_instance_228 () ;
    PFILLER0005 __added_filler_instance_229 () ;
    PFILLER0005 __added_filler_instance_230 () ;
    PFILLER0005 __added_filler_instance_231 () ;
    PFILLER0005 __added_filler_instance_232 () ;
    PFILLER0005 __added_filler_instance_233 () ;
    PFILLER0005 __added_filler_instance_234 () ;
    PFILLER0005 __added_filler_instance_235 () ;
    PFILLER0005 __added_filler_instance_236 () ;
    PFILLER0005 __added_filler_instance_237 () ;
    PFILLER0005 __added_filler_instance_238 () ;
    PFILLER0005 __added_filler_instance_239 () ;
    PFILLER0005 __added_filler_instance_240 () ;
    PFILLER0005 __added_filler_instance_241 () ;
    PFILLER0005 __added_filler_instance_242 () ;
    PFILLER0005 __added_filler_instance_243 () ;
    PFILLER0005 __added_filler_instance_244 () ;
    PFILLER0005 __added_filler_instance_245 () ;
    PFILLER0005 __added_filler_instance_246 () ;
    PFILLER0005 __added_filler_instance_247 () ;
    PFILLER0005 __added_filler_instance_248 () ;
    PFILLER0005 __added_filler_instance_249 () ;
```

```
    PFILLER0005 __added_filler_instance_250 () ;
    PFILLER0005 __added_filler_instance_251 () ;
    PFILLER0005 __added_filler_instance_252 () ;
    PFILLER0005 __added_filler_instance_253 () ;
    PFILLER0005 __added_filler_instance_254 () ;
    PFILLER0005 __added_filler_instance_255 () ;
    PFILLER0005 __added_filler_instance_256 () ;
    PFILLER0005 __added_filler_instance_257 () ;
    PFILLER0005 __added_filler_instance_258 () ;
    PFILLER0005 __added_filler_instance_259 () ;
    PFILLER0005 __added_filler_instance_260 () ;
    PFILLER0005 __added_filler_instance_261 () ;
    PFILLER0005 __added_filler_instance_262 () ;
    PFILLER0005 __added_filler_instance_263 () ;
    PFILLER0005 __added_filler_instance_264 () ;
    PFILLER0005 __added_filler_instance_265 () ;
    PFILLER0005 __added_filler_instance_266 () ;
    PFILLER0005 __added_filler_instance_267 () ;
    PFILLER0005 __added_filler_instance_268 () ;
    PFILLER0005 __added_filler_instance_269 () ;
    PFILLER0005 __added_filler_instance_270 () ;
    PFILLER0005 __added_filler_instance_271 () ;
    PFILLER0005 __added_filler_instance_272 () ;
    PFILLER0005 __added_filler_instance_273 () ;
    PFILLER0005 __added_filler_instance_274 () ;
    PFILLER0005 __added_filler_instance_275 () ;
    PFILLER0005 __added_filler_instance_276 () ;
    PFILLER0005 __added_filler_instance_277 () ;
    PFILLER0005 __added_filler_instance_278 () ;
    PFILLER0005 __added_filler_instance_279 () ;
    PFILLER0005 __added_filler_instance_280 () ;
    PFILLER0005 __added_filler_instance_281 () ;
    PFILLER0005 __added_filler_instance_282 () ;
    PFILLER0005 __added_filler_instance_283 () ;
    PFILLER0005 __added_filler_instance_284 () ;
    PFILLER0005 __added_filler_instance_285 () ;
    PFILLER0005 __added_filler_instance_286 () ;
    PFILLER0005 __added_filler_instance_287 () ;
    PFILLER0005 __added_filler_instance_288 () ;
    PFILLER0005 __added_filler_instance_289 () ;
    PFILLER0005 __added_filler_instance_290 () ;
    PFILLER0005 __added_filler_instance_291 () ;
    PFILLER0005 __added_filler_instance_292 () ;
    PFILLER0005 __added_filler_instance_293 () ;
    PFILLER0005 __added_filler_instance_294 () ;
    PFILLER0005 __added_filler_instance_295 () ;
    PFILLER0005 __added_filler_instance_296 () ;
    PFILLER0005 __added_filler_instance_297 () ;
    PFILLER0005 __added_filler_instance_298 () ;
    PFILLER0005 __added_filler_instance_299 () ;
    PFILLER0005 __added_filler_instance_300 () ;
    PFILLER0005 __added_filler_instance_301 () ;
    PFILLER0005 __added_filler_instance_302 () ;
    PFILLER0005 __added_filler_instance_303 () ;
```

```
    PFILLER0005 __added_filler_instance_304 () ;
    PFILLER0005 __added_filler_instance_305 () ;
    PFILLER0005 __added_filler_instance_306 () ;
    PFILLER0005 __added_filler_instance_307 () ;
    PFILLER0005 __added_filler_instance_308 () ;
    PFILLER0005 __added_filler_instance_309 () ;
    PFILLER0005 __added_filler_instance_310 () ;
    PFILLER0005 __added_filler_instance_311 () ;
    PFILLER0005 __added_filler_instance_312 () ;
    PFILLER0005 __added_filler_instance_313 () ;
    PFILLER0005 __added_filler_instance_314 () ;
    PFILLER0005 __added_filler_instance_315 () ;
    PFILLER0005 __added_filler_instance_316 () ;
    PFILLER0005 __added_filler_instance_317 () ;
    PFILLER0005 __added_filler_instance_318 () ;
    PFILLER0005 __added_filler_instance_319 () ;
    PFILLER0005 __added_filler_instance_320 () ;
    PFILLER0005 __added_filler_instance_321 () ;
    PFILLER0005 __added_filler_instance_322 () ;
    PFILLER0005 __added_filler_instance_323 () ;
    PFILLER0005 __added_filler_instance_324 () ;
    PFILLER0005 __added_filler_instance_325 () ;
    PFILLER0005 __added_filler_instance_326 () ;
    PFILLER0005 __added_filler_instance_327 () ;
    PFILLER0005 __added_filler_instance_328 () ;
    PFILLER0005 __added_filler_instance_329 () ;
    PFILLER0005 __added_filler_instance_330 () ;
    PFILLER0005 __added_filler_instance_331 () ;
    PFILLER0005 __added_filler_instance_332 () ;
    PFILLER0005 __added_filler_instance_333 () ;
    PFILLER0005 __added_filler_instance_334 () ;
    PFILLER0005 __added_filler_instance_335 () ;
    PFILLER0005 __added_filler_instance_336 () ;
    PFILLER0005 __added_filler_instance_337 () ;
    PFILLER0005 __added_filler_instance_338 () ;
    PFILLER0005 __added_filler_instance_339 () ;
    PFILLER0005 __added_filler_instance_340 () ;
    PFILLER0005 __added_filler_instance_341 () ;
    PFILLER0005 __added_filler_instance_342 () ;
    PFILLER0005 __added_filler_instance_343 () ;
    PFILLER0005 __added_filler_instance_344 () ;
    PFILLER0005 __added_filler_instance_345 () ;
    PFILLER0005 __added_filler_instance_346 () ;
    PFILLER0005 __added_filler_instance_347 () ;
    PFILLER20 __added_filler_instance_348 () ;
    PFILLER5 __added_filler_instance_349 () ;
    PFILLER1 __added_filler_instance_350 () ;
    PFILLER1 __added_filler_instance_351 () ;
    PFILLER1 __added_filler_instance_352 () ;
    PFILLER1 __added_filler_instance_353 () ;
    PFILLER20 __added_filler_instance_354 () ;
    PFILLER5 __added_filler_instance_355 () ;
    PFILLER1 __added_filler_instance_356 () ;
    PFILLER1 __added_filler_instance_357 () ;
```

```
    PFILLER1 __added_filler_instance_358 () ;
    PFILLER1 __added_filler_instance_359 () ;
    PFILLER20 __added_filler_instance_360 () ;
    PFILLER5 __added_filler_instance_361 () ;
    PFILLER1 __added_filler_instance_362 () ;
    PFILLER1 __added_filler_instance_363 () ;
    PFILLER1 __added_filler_instance_364 () ;
    PFILLER1 __added_filler_instance_365 () ;
    PFILLER20 __added_filler_instance_366 () ;
    PFILLER5 __added_filler_instance_367 () ;
    PFILLER1 __added_filler_instance_368 () ;
    PFILLER1 __added_filler_instance_369 () ;
    PFILLER1 __added_filler_instance_370 () ;
    PFILLER1 __added_filler_instance_371 () ;
    PFILLER20 __added_filler_instance_372 () ;
    PFILLER5 __added_filler_instance_373 () ;
    PFILLER1 __added_filler_instance_374 () ;
    PFILLER1 __added_filler_instance_375 () ;
    PFILLER1 __added_filler_instance_376 () ;
    PFILLER1 __added_filler_instance_377 () ;
    PFILLER0005 __added_filler_instance_378 () ;
    PFILLER0005 __added_filler_instance_379 () ;
    PFILLER0005 __added_filler_instance_380 () ;
    PFILLER0005 __added_filler_instance_381 () ;
    PFILLER0005 __added_filler_instance_382 () ;
    PFILLER0005 __added_filler_instance_383 () ;
    PFILLER0005 __added_filler_instance_384 () ;
    PFILLER0005 __added_filler_instance_385 () ;
    PFILLER0005 __added_filler_instance_386 () ;
    PFILLER0005 __added_filler_instance_387 () ;
    PFILLER0005 __added_filler_instance_388 () ;
    PFILLER0005 __added_filler_instance_389 () ;
    PFILLER0005 __added_filler_instance_390 () ;
    PFILLER0005 __added_filler_instance_391 () ;
    PFILLER0005 __added_filler_instance_392 () ;
    PFILLER0005 __added_filler_instance_393 () ;
    PFILLER0005 __added_filler_instance_394 () ;
    PFILLER0005 __added_filler_instance_395 () ;
    PFILLER0005 __added_filler_instance_396 () ;
    PFILLER0005 __added_filler_instance_397 () ;
    PFILLER0005 __added_filler_instance_398 () ;
    PFILLER0005 __added_filler_instance_399 () ;
    PFILLER0005 __added_filler_instance_400 () ;
    PFILLER0005 __added_filler_instance_401 () ;
    PFILLER0005 __added_filler_instance_402 () ;
    PFILLER0005 __added_filler_instance_403 () ;
    PFILLER0005 __added_filler_instance_404 () ;
    PFILLER0005 __added_filler_instance_405 () ;
    PFILLER0005 __added_filler_instance_406 () ;
    PFILLER0005 __added_filler_instance_407 () ;
    PFILLER0005 __added_filler_instance_408 () ;
    PFILLER0005 __added_filler_instance_409 () ;
    PFILLER0005 __added_filler_instance_410 () ;
    PFILLER0005 __added_filler_instance_411 () ;
```

```
    PFILLER0005 __added_filler_instance_412 () ;
    PFILLER0005 __added_filler_instance_413 () ;
    PFILLER0005 __added_filler_instance_414 () ;
    PFILLER0005 __added_filler_instance_415 () ;
    PFILLER0005 __added_filler_instance_416 () ;
    PFILLER0005 __added_filler_instance_417 () ;
    PFILLER0005 __added_filler_instance_418 () ;
    PFILLER0005 __added_filler_instance_419 () ;
    PFILLER0005 __added_filler_instance_420 () ;
    PFILLER0005 __added_filler_instance_421 () ;
    PFILLER0005 __added_filler_instance_422 () ;
    PFILLER0005 __added_filler_instance_423 () ;
    PFILLER0005 __added_filler_instance_424 () ;
    PFILLER0005 __added_filler_instance_425 () ;
    PFILLER0005 __added_filler_instance_426 () ;
    PFILLER0005 __added_filler_instance_427 () ;
    PFILLER0005 __added_filler_instance_428 () ;
    PFILLER0005 __added_filler_instance_429 () ;
    PFILLER0005 __added_filler_instance_430 () ;
    PFILLER0005 __added_filler_instance_431 () ;
    PFILLER0005 __added_filler_instance_432 () ;
    PFILLER0005 __added_filler_instance_433 () ;
    PFILLER0005 __added_filler_instance_434 () ;
    PFILLER0005 __added_filler_instance_435 () ;
    PFILLER0005 __added_filler_instance_436 () ;
    PFILLER0005 __added_filler_instance_437 () ;
    PFILLER0005 __added_filler_instance_438 () ;
    PFILLER0005 __added_filler_instance_439 () ;
    PFILLER0005 __added_filler_instance_440 () ;
    PFILLER0005 __added_filler_instance_441 () ;
    PFILLER0005 __added_filler_instance_442 () ;
    PFILLER0005 __added_filler_instance_443 () ;
    PFILLER0005 __added_filler_instance_444 () ;
    PFILLER0005 __added_filler_instance_445 () ;
    PFILLER0005 __added_filler_instance_446 () ;
    PFILLER0005 __added_filler_instance_447 () ;
    PFILLER0005 __added_filler_instance_448 () ;
    PFILLER0005 __added_filler_instance_449 () ;
    PFILLER0005 __added_filler_instance_450 () ;
    PFILLER0005 __added_filler_instance_451 () ;
    PFILLER0005 __added_filler_instance_452 () ;
    PFILLER0005 __added_filler_instance_453 () ;
    PFILLER0005 __added_filler_instance_454 () ;
    PFILLER0005 __added_filler_instance_455 () ;
    PFILLER0005 __added_filler_instance_456 () ;
    PFILLER0005 __added_filler_instance_457 () ;
    PFILLER0005 __added_filler_instance_458 () ;
    PFILLER0005 __added_filler_instance_459 () ;
    PFILLER0005 __added_filler_instance_460 () ;
    PFILLER0005 __added_filler_instance_461 () ;
    PFILLER0005 __added_filler_instance_462 () ;
    PFILLER0005 __added_filler_instance_463 () ;
    PFILLER0005 __added_filler_instance_464 () ;
    PFILLER0005 __added_filler_instance_465 () ;
```

```
    PFILLER0005 __added_filler_instance_466 () ;
    PFILLER0005 __added_filler_instance_467 () ;
    PFILLER0005 __added_filler_instance_468 () ;
    PFILLER0005 __added_filler_instance_469 () ;
    PFILLER0005 __added_filler_instance_470 () ;
    PFILLER0005 __added_filler_instance_471 () ;
    PFILLER0005 __added_filler_instance_472 () ;
    PFILLER0005 __added_filler_instance_473 () ;
    PFILLER0005 __added_filler_instance_474 () ;
    PFILLER0005 __added_filler_instance_475 () ;
    PFILLER0005 __added_filler_instance_476 () ;
    PFILLER0005 __added_filler_instance_477 () ;
    PFILLER0005 __added_filler_instance_478 () ;
    PFILLER0005 __added_filler_instance_479 () ;
    PFILLER0005 __added_filler_instance_480 () ;
    PFILLER0005 __added_filler_instance_481 () ;
    PFILLER0005 __added_filler_instance_482 () ;
    PFILLER0005 __added_filler_instance_483 () ;
    PFILLER0005 __added_filler_instance_484 () ;
    PFILLER0005 __added_filler_instance_485 () ;
    PFILLER0005 __added_filler_instance_486 () ;
    PFILLER0005 __added_filler_instance_487 () ;
    PFILLER0005 __added_filler_instance_488 () ;
    PFILLER0005 __added_filler_instance_489 () ;
    PFILLER0005 __added_filler_instance_490 () ;
    PFILLER0005 __added_filler_instance_491 () ;
    PFILLER0005 __added_filler_instance_492 () ;
    PFILLER0005 __added_filler_instance_493 () ;
    PFILLER0005 __added_filler_instance_494 () ;
    PFILLER0005 __added_filler_instance_495 () ;
    PFILLER0005 __added_filler_instance_496 () ;
    PFILLER0005 __added_filler_instance_497 () ;
    PFILLER0005 __added_filler_instance_498 () ;
    PFILLER0005 __added_filler_instance_499 () ;
    PFILLER0005 __added_filler_instance_500 () ;
    PFILLER0005 __added_filler_instance_501 () ;
    PFILLER0005 __added_filler_instance_502 () ;
    PFILLER0005 __added_filler_instance_503 () ;
    PFILLER0005 __added_filler_instance_504 () ;
    PFILLER0005 __added_filler_instance_505 () ;
    PFILLER0005 __added_filler_instance_506 () ;
    PFILLER0005 __added_filler_instance_507 () ;
    PFILLER0005 __added_filler_instance_508 () ;
    PFILLER0005 __added_filler_instance_509 () ;
    PFILLER0005 __added_filler_instance_510 () ;
    PFILLER0005 __added_filler_instance_511 () ;
    PFILLER0005 __added_filler_instance_512 () ;
    PFILLER0005 __added_filler_instance_513 () ;
    PFILLER0005 __added_filler_instance_514 () ;
    PFILLER0005 __added_filler_instance_515 () ;
    PFILLER0005 __added_filler_instance_516 () ;
    PFILLER0005 __added_filler_instance_517 () ;
    PFILLER0005 __added_filler_instance_518 () ;
    PFILLER0005 __added_filler_instance_519 () ;
```

```
    PFILLER0005 __added_filler_instance_520 () ;
    PFILLER0005 __added_filler_instance_521 () ;
    PFILLER0005 __added_filler_instance_522 () ;
    PFILLER0005 __added_filler_instance_523 () ;
    PFILLER0005 __added_filler_instance_524 () ;
    PFILLER0005 __added_filler_instance_525 () ;
    PFILLER0005 __added_filler_instance_526 () ;
    PFILLER0005 __added_filler_instance_527 () ;
    PFILLER0005 __added_filler_instance_528 () ;
    PFILLER0005 __added_filler_instance_529 () ;
    PFILLER0005 __added_filler_instance_530 () ;
    PFILLER0005 __added_filler_instance_531 () ;
    PFILLER0005 __added_filler_instance_532 () ;
    PFILLER0005 __added_filler_instance_533 () ;
    PFILLER0005 __added_filler_instance_534 () ;
    PFILLER0005 __added_filler_instance_535 () ;
    PFILLER0005 __added_filler_instance_536 () ;
    PFILLER0005 __added_filler_instance_537 () ;
    PFILLER0005 __added_filler_instance_538 () ;
    PFILLER0005 __added_filler_instance_539 () ;
    PFILLER0005 __added_filler_instance_540 () ;
    PFILLER0005 __added_filler_instance_541 () ;
    PFILLER0005 __added_filler_instance_542 () ;
    PFILLER0005 __added_filler_instance_543 () ;
    PFILLER0005 __added_filler_instance_544 () ;
    PFILLER0005 __added_filler_instance_545 () ;
    PFILLER20 __added_filler_instance_546 () ;
    PFILLER5 __added_filler_instance_547 () ;
    PFILLER1 __added_filler_instance_548 () ;
    PFILLER1 __added_filler_instance_549 () ;
    PFILLER1 __added_filler_instance_550 () ;
    PFILLER1 __added_filler_instance_551 () ;
    PFILLER0005 __added_filler_instance_552 () ;
    PFILLER0005 __added_filler_instance_553 () ;
    PFILLER0005 __added_filler_instance_554 () ;
    PFILLER0005 __added_filler_instance_555 () ;
    PFILLER0005 __added_filler_instance_556 () ;
    PFILLER0005 __added_filler_instance_557 () ;
    PFILLER0005 __added_filler_instance_558 () ;
    PFILLER0005 __added_filler_instance_559 () ;
    PFILLER0005 __added_filler_instance_560 () ;
    PFILLER0005 __added_filler_instance_561 () ;
    PFILLER0005 __added_filler_instance_562 () ;
    PFILLER0005 __added_filler_instance_563 () ;
    PFILLER0005 __added_filler_instance_564 () ;
    PFILLER0005 __added_filler_instance_565 () ;
    PFILLER0005 __added_filler_instance_566 () ;
    PFILLER0005 __added_filler_instance_567 () ;
    PFILLER0005 __added_filler_instance_568 () ;
    PFILLER0005 __added_filler_instance_569 () ;
    PFILLER0005 __added_filler_instance_570 () ;
    PFILLER0005 __added_filler_instance_571 () ;
    PFILLER0005 __added_filler_instance_572 () ;
    PFILLER0005 __added_filler_instance_573 () ;
```

```
    PFILLER0005 __added_filler_instance_574 () ;
    PFILLER0005 __added_filler_instance_575 () ;
    PFILLER0005 __added_filler_instance_576 () ;
    PFILLER0005 __added_filler_instance_577 () ;
    PFILLER0005 __added_filler_instance_578 () ;
    PFILLER0005 __added_filler_instance_579 () ;
    PFILLER0005 __added_filler_instance_580 () ;
    PFILLER0005 __added_filler_instance_581 () ;
    PFILLER0005 __added_filler_instance_582 () ;
    PFILLER0005 __added_filler_instance_583 () ;
    PFILLER0005 __added_filler_instance_584 () ;
    PFILLER0005 __added_filler_instance_585 () ;
    PFILLER0005 __added_filler_instance_586 () ;
    PFILLER0005 __added_filler_instance_587 () ;
    PFILLER0005 __added_filler_instance_588 () ;
    PFILLER0005 __added_filler_instance_589 () ;
    PFILLER0005 __added_filler_instance_590 () ;
    PFILLER0005 __added_filler_instance_591 () ;
    PFILLER0005 __added_filler_instance_592 () ;
    PFILLER0005 __added_filler_instance_593 () ;
    PFILLER0005 __added_filler_instance_594 () ;
    PFILLER0005 __added_filler_instance_595 () ;
    PFILLER0005 __added_filler_instance_596 () ;
    PFILLER0005 __added_filler_instance_597 () ;
    PFILLER0005 __added_filler_instance_598 () ;
    PFILLER0005 __added_filler_instance_599 () ;
    PFILLER0005 __added_filler_instance_600 () ;
    PFILLER0005 __added_filler_instance_601 () ;
    PFILLER0005 __added_filler_instance_602 () ;
    PFILLER0005 __added_filler_instance_603 () ;
    PFILLER0005 __added_filler_instance_604 () ;
    PFILLER0005 __added_filler_instance_605 () ;
    PFILLER0005 __added_filler_instance_606 () ;
    PFILLER0005 __added_filler_instance_607 () ;
    PFILLER0005 __added_filler_instance_608 () ;
    PFILLER0005 __added_filler_instance_609 () ;
    PFILLER0005 __added_filler_instance_610 () ;
    PFILLER0005 __added_filler_instance_611 () ;
    PFILLER0005 __added_filler_instance_612 () ;
    PFILLER0005 __added_filler_instance_613 () ;
    PFILLER0005 __added_filler_instance_614 () ;
    PFILLER0005 __added_filler_instance_615 () ;
    PFILLER0005 __added_filler_instance_616 () ;
    PFILLER0005 __added_filler_instance_617 () ;
    PFILLER0005 __added_filler_instance_618 () ;
    PFILLER0005 __added_filler_instance_619 () ;
    PFILLER0005 __added_filler_instance_620 () ;
    PFILLER0005 __added_filler_instance_621 () ;
    PFILLER0005 __added_filler_instance_622 () ;
    PFILLER0005 __added_filler_instance_623 () ;
    PFILLER0005 __added_filler_instance_624 () ;
    PFILLER0005 __added_filler_instance_625 () ;
    PFILLER0005 __added_filler_instance_626 () ;
    PFILLER0005 __added_filler_instance_627 () ;
```

```
    PFILLER0005 __added_filler_instance_628 () ;
    PFILLER0005 __added_filler_instance_629 () ;
    PFILLER0005 __added_filler_instance_630 () ;
    PFILLER0005 __added_filler_instance_631 () ;
    PFILLER0005 __added_filler_instance_632 () ;
    PFILLER0005 __added_filler_instance_633 () ;
    PFILLER0005 __added_filler_instance_634 () ;
    PFILLER0005 __added_filler_instance_635 () ;
    PFILLER0005 __added_filler_instance_636 () ;
    PFILLER0005 __added_filler_instance_637 () ;
    PFILLER0005 __added_filler_instance_638 () ;
    PFILLER0005 __added_filler_instance_639 () ;
    PFILLER0005 __added_filler_instance_640 () ;
    PFILLER0005 __added_filler_instance_641 () ;
    PFILLER0005 __added_filler_instance_642 () ;
    PFILLER0005 __added_filler_instance_643 () ;
    PFILLER0005 __added_filler_instance_644 () ;
    PFILLER0005 __added_filler_instance_645 () ;
    PFILLER0005 __added_filler_instance_646 () ;
    PFILLER0005 __added_filler_instance_647 () ;
    PFILLER0005 __added_filler_instance_648 () ;
    PFILLER0005 __added_filler_instance_649 () ;
    PFILLER0005 __added_filler_instance_650 () ;
    PFILLER0005 __added_filler_instance_651 () ;
    PFILLER0005 __added_filler_instance_652 () ;
    PFILLER0005 __added_filler_instance_653 () ;
    PFILLER0005 __added_filler_instance_654 () ;
    PFILLER0005 __added_filler_instance_655 () ;
    PFILLER0005 __added_filler_instance_656 () ;
    PFILLER0005 __added_filler_instance_657 () ;
    PFILLER0005 __added_filler_instance_658 () ;
    PFILLER0005 __added_filler_instance_659 () ;
    PFILLER0005 __added_filler_instance_660 () ;
    PFILLER0005 __added_filler_instance_661 () ;
    PFILLER0005 __added_filler_instance_662 () ;
    PFILLER0005 __added_filler_instance_663 () ;
    PFILLER0005 __added_filler_instance_664 () ;
    PFILLER0005 __added_filler_instance_665 () ;
    PFILLER0005 __added_filler_instance_666 () ;
    PFILLER0005 __added_filler_instance_667 () ;
    PFILLER0005 __added_filler_instance_668 () ;
    PFILLER0005 __added_filler_instance_669 () ;
    PFILLER0005 __added_filler_instance_670 () ;
    PFILLER0005 __added_filler_instance_671 () ;
    PFILLER0005 __added_filler_instance_672 () ;
    PFILLER0005 __added_filler_instance_673 () ;
    PFILLER0005 __added_filler_instance_674 () ;
    PFILLER0005 __added_filler_instance_675 () ;
    PFILLER0005 __added_filler_instance_676 () ;
    PFILLER0005 __added_filler_instance_677 () ;
    PFILLER0005 __added_filler_instance_678 () ;
    PFILLER0005 __added_filler_instance_679 () ;
    PFILLER0005 __added_filler_instance_680 () ;
    PFILLER0005 __added_filler_instance_681 () ;
```

```
    PFILLER0005 __added_filler_instance_682 () ;
    PFILLER0005 __added_filler_instance_683 () ;
    PFILLER0005 __added_filler_instance_684 () ;
    PFILLER0005 __added_filler_instance_685 () ;
    PFILLER0005 __added_filler_instance_686 () ;
    PFILLER0005 __added_filler_instance_687 () ;
    PFILLER0005 __added_filler_instance_688 () ;
    PFILLER0005 __added_filler_instance_689 () ;
    PFILLER0005 __added_filler_instance_690 () ;
    PFILLER0005 __added_filler_instance_691 () ;
    PFILLER0005 __added_filler_instance_692 () ;
    PFILLER0005 __added_filler_instance_693 () ;
    PFILLER0005 __added_filler_instance_694 () ;
    PFILLER0005 __added_filler_instance_695 () ;
    PFILLER0005 __added_filler_instance_696 () ;
    PFILLER0005 __added_filler_instance_697 () ;
    PFILLER0005 __added_filler_instance_698 () ;
    PFILLER0005 __added_filler_instance_699 () ;
    PFILLER0005 __added_filler_instance_700 () ;
    PFILLER0005 __added_filler_instance_701 () ;
    PFILLER0005 __added_filler_instance_702 () ;
    PFILLER0005 __added_filler_instance_703 () ;
    PFILLER0005 __added_filler_instance_704 () ;
    PFILLER0005 __added_filler_instance_705 () ;
    PFILLER0005 __added_filler_instance_706 () ;
    PFILLER0005 __added_filler_instance_707 () ;
    PFILLER0005 __added_filler_instance_708 () ;
    PFILLER0005 __added_filler_instance_709 () ;
    PFILLER0005 __added_filler_instance_710 () ;
    PFILLER0005 __added_filler_instance_711 () ;
    PFILLER0005 __added_filler_instance_712 () ;
    PFILLER0005 __added_filler_instance_713 () ;
    PFILLER0005 __added_filler_instance_714 () ;
    PFILLER0005 __added_filler_instance_715 () ;
    PFILLER0005 __added_filler_instance_716 () ;
    PFILLER0005 __added_filler_instance_717 () ;
    PFILLER0005 __added_filler_instance_718 () ;
    PFILLER0005 __added_filler_instance_719 () ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1250000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1250000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1250000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1250000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1250000 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
```

```
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1289200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1289200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1289200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1289200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1289200 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1328400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1328400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1328400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1328400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1328400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1367600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1367600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1367600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1367600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1367600 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1406800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1406800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1406800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
```

```
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1406800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1406800 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1446000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1446000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1446000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1446000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1446000 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1485200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1485200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1485200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1485200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1485200 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1524400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1524400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1524400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1524400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1524400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1563600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
```

```
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1563600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1563600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1563600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1563600 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1602800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1602800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1602800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1602800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1602800 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1642000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1642000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1642000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1642000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1642000 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1681200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x1608400y1681200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x1698000y1681200 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL1BWP7T \xofiller!FILL1BWP7T!x1709200y1681200 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
```

```
    FILL64BWP7T \xofiller!FILL64BWP7T!x1731600y1681200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2090000y1681200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL8BWP7T \xofiller!FILL8BWP7T!x2179600y1681200 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL4BWP7T \xofiller!FILL4BWP7T!x2224400y1681200 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1720400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x1608400y1720400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL4BWP7T \xofiller!FILL4BWP7T!x1698000y1720400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x1720400y1720400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL1BWP7T \xofiller!FILL1BWP7T!x1731600y1720400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x1754000y1720400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL8BWP7T \xofiller!FILL8BWP7T!x1843600y1720400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL4BWP7T \xofiller!FILL4BWP7T!x1888400y1720400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x1910800y1720400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1938800y1720400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2118000y1720400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL4BWP7T \xofiller!FILL4BWP7T!x2207600y1720400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2230000y1720400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL1BWP7T \xofiller!FILL1BWP7T!x2241200y1720400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
```

```
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1759600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1759600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1759600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1759600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1759600 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1798800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1798800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1798800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1798800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1798800 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1838000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1838000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1838000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1838000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1838000 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1877200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1877200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1877200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
```

```
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1877200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1877200 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1916400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1916400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1916400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1916400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1916400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1955600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1955600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1955600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1955600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1955600 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y1994800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y1994800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y1994800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y1994800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y1994800 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y2034000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
```

```
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y2034000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y2034000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y2034000 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y2034000 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y2073200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y2073200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y2073200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y2073200 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y2073200 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y2112400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y2112400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y2112400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y2112400 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y2112400 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y2151600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y2151600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y2151600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y2151600 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
```

```
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y2151600 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1250000y2190800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL64BWP7T \xofiller!FILL64BWP7T!x1608400y2190800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL32BWP7T \xofiller!FILL32BWP7T!x1966800y2190800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL16BWP7T \xofiller!FILL16BWP7T!x2146000y2190800 ( .VSS ( VSS
       ) ,
        .VDD ( VDD ) ) ;
    FILL2BWP7T \xofiller!FILL2BWP7T!x2235600y2190800 ( .VSS ( VSS )
       ,
        .VDD ( VDD ) ) ;
    endmodule
```

Cuadro 94: Verilog de un circuito *Not* con conexiones lógicas e instancias de componentes puramente físicos

15.2. Archivo .spf original obtenido para un sumador de 4 *bits* luego de LPE

```
    *
    *|DSPF 1.3
    *|DESIGN carry_look_ahead_4bit
    *|DATE "Wed Nov 13 11:57:05 2024"
    *|VENDOR "Synopsys"
    *|PROGRAM "StarRC"
    *|VERSION "V-2023.12-SP1"
    *|DIVIDER /
    *|DELIMITER :
    **FORMAT SPF
    *

    ** COMMENTS

    ** OPERATING_TEMPERATURE 25
    ** GLOBAL_TEMPERATURE 25
    **
    ** TCAD_GRD_FILE /home/nanoelectronica/Desktop/Folder_de_Trabajo
       /Rui/TSMC/nueva_integracion/sintesis_fisica_adder4b_noIO/
       Outputs/LPE/cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
    **   TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
    **   TCADGRD_VERSION 62

    .SUBCKT carry_look_ahead_4bit
```

```
*|GROUND_NET 0

*|NET N_10 0.00190297PF
*|I (I_D4A6F12410:N_3 I_D4A6F12410 N_3 B 0 268.6550 272.2800)
*|I (I_D4A6F12412:N_3 I_D4A6F12412 N_3 B 0 260.2400 271.7200)
Cg2_1 I_D4A6F12412:N_3 0 1.90297e-15
R2_1 I_D4A6F12410:N_3 I_D4A6F12412:N_3 28.0563
R2_2 I_D4A6F12412:N_3 N_10:6 0.0215527
R2_3 I_D4A6F12412:N_3 N_10:4 0.028
R2_4 I_D4A6F12412:N_3 N_10:3 0.0135474
R2_5 N_10:3 N_10:5 0.001
R2_6 N_10:4 N_10:6 0.001

*|NET N_11 0.00143142PF
*|I (I_D4A6F12410:N_4 I_D4A6F12410 N_4 B 0 269.2050 271.7200)
*|I (I_D4A6F12412:N_4 I_D4A6F12412 N_4 B 0 263.0400 271.7200)
Cg3_1 I_D4A6F12412:N_4 0 1.43142e-15
R3_1 I_D4A6F12410:N_4 I_D4A6F12412:N_4 27.2969
R3_2 I_D4A6F12412:N_4 N_11:3 0.001

*|NET N_12 0.00367749PF
*|I (I_D4A6F1241:N_4 I_D4A6F1241 N_4 B 0 275.3600 279.5600)
*|I (I_D4A6F12410:N_8 I_D4A6F12410 N_8 B 0 271.9850 272.3050)
*|I (I_D4A6F12412:N_6 I_D4A6F12412 N_6 B 0 264.1350 272.8400)
Cg4_1 I_D4A6F12410:N_8 0 3.67749e-15
R4_1 I_D4A6F1241:N_4 I_D4A6F12410:N_8 35.6522
R4_2 I_D4A6F1241:N_4 I_D4A6F12412:N_6 74.6504
R4_3 I_D4A6F12410:N_8 I_D4A6F12412:N_6 70.793

*|NET N_13 0.00228823PF
*|I (I_D4A6F12411:N_6 I_D4A6F12411 N_6 B 0 275.3600 268.3600)
*|I (I_D4A6F1244:N_3 I_D4A6F1244 N_3 B 0 286.0000 271.7200)
*|I (I_D4A6F1248:N_3 I_D4A6F1248 N_3 B 0 279.0250 267.1550)
Cg5_1 I_D4A6F1248:N_3 0 2.28823e-15
R5_1 I_D4A6F12411:N_6 I_D4A6F1248:N_3 38.4898
R5_2 I_D4A6F12411:N_6 I_D4A6F1244:N_3 45.9139
R5_3 I_D4A6F1244:N_3 I_D4A6F1248:N_3 43.6309

*|NET N_14 0.00520606PF
*|I (I_D4A6F12411:N_8 I_D4A6F12411 N_8 B 0 276.4650 268.3600)
*|I (I_D4A6F12413:N_6 I_D4A6F12413 N_6 B 0 270.2950 279.0000)
*|I (I_D4A6F1244:N_4 I_D4A6F1244 N_4 B 0 288.8000 271.7200)
Cg6_1 I_D4A6F12411:N_8 0 5.20606e-15
R6_1 I_D4A6F12411:N_8 I_D4A6F1244:N_4 36.4846
R6_2 I_D4A6F12411:N_8 I_D4A6F12413:N_6 41.3785
R6_3 I_D4A6F12413:N_6 I_D4A6F1244:N_4 125.762

*|NET N_15 0.00261148PF
*|I (I_D4A6F1245:N_4 I_D4A6F1245 N_4 B 0 283.7600 271.7200)
*|I (I_D4A6F1246:N_6 I_D4A6F1246 N_6 B 0 279.8900 274.8925)
*|I (I_D4A6F1248:N_5 I_D4A6F1248 N_5 B 0 279.5600 268.1300)
Cg7_1 I_D4A6F1248:N_5 0 2.61148e-15
R7_1 I_D4A6F1245:N_4 I_D4A6F1246:N_6 41.6183
R7_2 I_D4A6F1245:N_4 I_D4A6F1248:N_5 40.7633
```

```
R7_3 I_D4A6F1246:N_6 I_D4A6F1248:N_5 45.1144
R7_4 I_D4A6F1248:N_5 N_15:4 0.0153214

*|NET N_16 0.00373615PF
*|I (I_D4A6F1243:N_3 I_D4A6F1243 N_3 B 0 260.8000 276.2000)
*|I (I_D4A6F1247:N_5 I_D4A6F1247 N_5 B 0 271.1600 268.0800)
Cg8_1 I_D4A6F1243:N_3 0 3.73615e-15
R8_1 I_D4A6F1243:N_3 I_D4A6F1247:N_5 33.1472

*|NET N_17 0.00419252PF
*|I (I_D4A6F1243:N_4 I_D4A6F1243 N_4 B 0 263.6000 276.2000)
*|I (I_D4A6F1247:N_6 I_D4A6F1247 N_6 B 0 272.3800 268.3600)
Cg9_1 I_D4A6F1243:N_4 0 4.19252e-15
R9_1 I_D4A6F1243:N_4 I_D4A6F1247:N_6 33.6622

*|NET N_18 0.00413743PF
*|I (I_D4A6F12411:N_3 I_D4A6F12411 N_3 B 0 273.1350 267.6850)
*|I (I_D4A6F12413:N_3 I_D4A6F12413 N_3 B 0 266.3300 279.5600)
Cg10_1 I_D4A6F12413:N_3 0 4.13743e-15
R10_1 I_D4A6F12411:N_3 I_D4A6F12413:N_3 32.9125
R10_2 I_D4A6F12413:N_3 N_18:3 0.0106737
R10_3 N_18:3 N_18:4 0.001

*|NET N_19 0.00425119PF
*|I (I_D4A6F12411:N_4 I_D4A6F12411 N_4 B 0 273.9900 267.9700)
*|I (I_D4A6F12413:N_4 I_D4A6F12413 N_4 B 0 269.2000 279.3400)
Cg11_1 I_D4A6F12413:N_4 0 4.25119e-15
R11_1 I_D4A6F12411:N_4 I_D4A6F12413:N_4 33.6209

*|NET N_2 0.00482727PF
*|I (I_D4A6F1242:N_3 I_D4A6F1242 N_3 B 0 279.2800 284.0400)
*|I (I_D4A6F1243:N_6 I_D4A6F1243 N_6 B 0 264.7200 275.6400)
*|I (I_D4A6F1247:N_8 I_D4A6F1247 N_8 B 0 269.6200 268.3600)
Cg12_1 I_D4A6F1243:N_6 0 4.82727e-15
R12_1 I_D4A6F1242:N_3 I_D4A6F1243:N_6 38.2781
R12_2 I_D4A6F1242:N_3 I_D4A6F1247:N_8 173.259
R12_3 I_D4A6F1243:N_6 I_D4A6F1247:N_8 52.5292

*|NET N_20 0.00130897PF
*|I (I_D4A6F1246:N_4 I_D4A6F1246 N_4 B 0 278.7200 276.2000)
*|I (I_D4A6F1248:N_4 I_D4A6F1248 N_4 B 0 280.1200 268.9200)
Cg13_1 I_D4A6F1248:N_4 0 1.30897e-15
R13_1 I_D4A6F1246:N_4 I_D4A6F1248:N_4 15.1783
R13_2 I_D4A6F1248:N_4 N_20:3 0.001

*|NET N_21 0.000835738PF
*|I (I_D4A6F1245:N_3 I_D4A6F1245 N_3 B 0 280.9600 271.7200)
*|I (I_D4A6F1248:N_6 I_D4A6F1248 N_6 B 0 280.7800 268.3600)
Cg14_1 I_D4A6F1248:N_6 0 8.35738e-16
R14_1 I_D4A6F1245:N_3 I_D4A6F1248:N_6 13.6686

*|NET N_3 0.00509408PF
*|I (I_D4A6F1241:N_3 I_D4A6F1241 N_3 B 0 272.5600 279.5600)
*|I (I_D4A6F12410:N_6 I_D4A6F12410 N_6 B 0 270.8800 271.7200)
```

```
*|I (I_D4A6F1247:N_3 I_D4A6F1247 N_3 B 0 270.6250 267.1550)
Cg15_1 I_D4A6F1241:N_3 0 1.86709e-15
Cg15_2 I_D4A6F12410:N_6 0 1.65062e-15
Cg15_3 I_D4A6F1247:N_3 0 1.57637e-15
R15_1 I_D4A6F1241:N_3 I_D4A6F12410:N_6 114.153
R15_2 I_D4A6F1241:N_3 I_D4A6F1247:N_3 114.472
R15_3 I_D4A6F12410:N_6 N_3:4 0.0125709
R15_4 I_D4A6F12410:N_6 N_3:5 0.0159394
R15_5 I_D4A6F12410:N_6 I_D4A6F1247:N_3 74.2958

*|NET N_6 2.49374PF
*|I (I_D4A6F1241:N_2 I_D4A6F1241 N_2 B 0 276.1500 277.6450)
*|I (I_D4A6F12410:N_2 I_D4A6F12410 N_2 B 0 269.8300 269.8050)
*|I (I_D4A6F12411:N_2 I_D4A6F12411 N_2 B 0 272.8400 270.0400)
*|I (I_D4A6F12412:N_2 I_D4A6F12412 N_2 B 0 259.4000 270.0400)
*|I (I_D4A6F12413:N_2 I_D4A6F12413 N_2 B 0 265.5600 277.8800)
*|I (I_D4A6F1242:N_2 I_D4A6F1242 N_2 B 0 278.4400 285.7200)
*|I (I_D4A6F1243:N_2 I_D4A6F1243 N_2 B 0 259.9600 277.8800)
*|I (I_D4A6F1244:N_2 I_D4A6F1244 N_2 B 0 285.1600 270.0400)
*|I (I_D4A6F1245:N_2 I_D4A6F1245 N_2 B 0 280.8950 269.8050)
*|I (I_D4A6F1246:N_2 I_D4A6F1246 N_2 B 0 276.1500 277.8800)
*|I (I_D4A6F1247:N_2 I_D4A6F1247 N_2 B 0 269.8300 270.0400)
*|I (I_D4A6F1248:N_2 I_D4A6F1248 N_2 B 0 280.8950 270.0400)
*|I (I_D4A6F1249:N_3 I_D4A6F1249 N_3 B 0 432.1000 256.9600)
*|I (I_D4A6F1249:N_4 I_D4A6F1249 N_4 B 0 434.6000 273.7600)
*|I (I_D4A6F1249:N_5 I_D4A6F1249 N_5 B 0 432.1000 290.5600)
Cg18_1 I_D4A6F1242:N_2 0 1.14894e-14
Cg18_2 I_D4A6F1249:N_3 0 4.80368e-14
Cg18_3 I_D4A6F1249:N_5 0 2.17459e-14
Cg18_4 N_6:16 0 7.69312e-14
Cg18_5 N_6:17 0 5.93404e-13
Cg18_6 N_6:18 0 6.12987e-13
Cg18_7 N_6:19 0 3.0769e-13
Cg18_8 N_6:20 0 4.43173e-13
Cg18_9 N_6:21 0 3.18304e-13
Cg18_10 N_6:22 0 5.99755e-14
R18_1 I_D4A6F1241:N_2 I_D4A6F1248:N_2 516.889
R18_2 I_D4A6F1241:N_2 I_D4A6F1247:N_2 33.7693
R18_3 I_D4A6F1241:N_2 I_D4A6F1246:N_2 0.001
R18_4 I_D4A6F1241:N_2 N_6:21 34.5868
R18_5 I_D4A6F1241:N_2 N_6:20 18.7595
R18_6 I_D4A6F1241:N_2 N_6:19 6.38215
R18_7 I_D4A6F1241:N_2 I_D4A6F1244:N_2 404.311
R18_8 I_D4A6F1241:N_2 N_6:18 11.6274
R18_9 I_D4A6F1241:N_2 I_D4A6F1243:N_2 513.786
R18_10 I_D4A6F1241:N_2 N_6:16 900.445
R18_11 I_D4A6F1241:N_2 I_D4A6F1242:N_2 17.2574
R18_12 I_D4A6F1241:N_2 I_D4A6F12413:N_2 1.04809
R18_13 I_D4A6F1241:N_2 N_6:17 28.4019
R18_14 I_D4A6F1241:N_2 I_D4A6F1249:N_5 436036.0
R18_15 I_D4A6F1241:N_2 I_D4A6F12412:N_2 856.152
R18_16 I_D4A6F1241:N_2 I_D4A6F1249:N_4 34066.9
R18_17 I_D4A6F1241:N_2 I_D4A6F1249:N_3 605.989
R18_18 I_D4A6F1241:N_2 I_D4A6F12411:N_2 18.978
```

```
    R18_19 I_D4A6F12410:N_2 I_D4A6F1247:N_2 0.001
    R18_20 I_D4A6F12411:N_2 I_D4A6F1248:N_2 1.32598
    R18_21 I_D4A6F12411:N_2 I_D4A6F1247:N_2 0.477243
    R18_22 I_D4A6F12411:N_2 I_D4A6F1246:N_2 103.227
    R18_23 I_D4A6F12411:N_2 N_6:21 62.6288
    R18_24 I_D4A6F12411:N_2 N_6:20 39.6447
    R18_25 I_D4A6F12411:N_2 N_6:19 16.7927
    R18_26 I_D4A6F12411:N_2 I_D4A6F1244:N_2 2356.02
    R18_27 I_D4A6F12411:N_2 N_6:18 14.5289
    R18_28 I_D4A6F12411:N_2 I_D4A6F1243:N_2 805.609
    R18_29 I_D4A6F12411:N_2 N_6:16 975.587
    R18_30 I_D4A6F12411:N_2 I_D4A6F1242:N_2 28.3647
    R18_31 I_D4A6F12411:N_2 I_D4A6F12413:N_2 245.819
    R18_32 I_D4A6F12411:N_2 N_6:17 38.0953
    R18_33 I_D4A6F12411:N_2 I_D4A6F1249:N_5 264859.0
    R18_34 I_D4A6F12411:N_2 I_D4A6F12412:N_2 1301.73
    R18_35 I_D4A6F12411:N_2 I_D4A6F1249:N_4 34904.4
    R18_36 I_D4A6F12411:N_2 I_D4A6F1249:N_3 629.614
    R18_37 I_D4A6F12412:N_2 I_D4A6F1248:N_2 37676.3
    R18_38 I_D4A6F12412:N_2 I_D4A6F1247:N_2 1.72965
    R18_39 I_D4A6F12412:N_2 I_D4A6F1246:N_2 4935.69
    R18_40 I_D4A6F12412:N_2 N_6:21 651.466
    R18_41 I_D4A6F12412:N_2 N_6:20 482.988
    R18_42 I_D4A6F12412:N_2 N_6:19 283.275
    R18_43 I_D4A6F12412:N_2 I_D4A6F1244:N_2 19906.7
    R18_44 I_D4A6F12412:N_2 N_6:18 7.80242
    R18_45 I_D4A6F12412:N_2 I_D4A6F1243:N_2 402.096
    R18_46 I_D4A6F12412:N_2 N_6:16 4651.06
    R18_47 I_D4A6F12412:N_2 I_D4A6F1242:N_2 483.746
    R18_48 I_D4A6F12412:N_2 I_D4A6F12413:N_2 11756.6
    R18_49 I_D4A6F12412:N_2 N_6:17 56.8337
    R18_50 I_D4A6F12412:N_2 I_D4A6F1249:N_3 9861.7
    R18_51 I_D4A6F12413:N_2 I_D4A6F1248:N_2 6697.86
    R18_52 I_D4A6F12413:N_2 I_D4A6F1247:N_2 437.756
    R18_53 I_D4A6F12413:N_2 I_D4A6F1246:N_2 188.298
    R18_54 I_D4A6F12413:N_2 N_6:21 631.51
    R18_55 I_D4A6F12413:N_2 N_6:20 409.894
    R18_56 I_D4A6F12413:N_2 N_6:19 182.453
    R18_57 I_D4A6F12413:N_2 I_D4A6F1244:N_2 19953.6
    R18_58 I_D4A6F12413:N_2 N_6:18 155.409
    R18_59 I_D4A6F12413:N_2 I_D4A6F1243:N_2 0.929241
    R18_60 I_D4A6F12413:N_2 N_6:16 11793.7
    R18_61 I_D4A6F12413:N_2 I_D4A6F1242:N_2 236.547
    R18_62 I_D4A6F12413:N_2 N_6:17 386.667
    R18_63 I_D4A6F12413:N_2 I_D4A6F1249:N_4 441504.0
    R18_64 I_D4A6F12413:N_2 I_D4A6F1249:N_3 7874.7
    R18_65 I_D4A6F1242:N_2 I_D4A6F1248:N_2 771.242
    R18_66 I_D4A6F1242:N_2 I_D4A6F1247:N_2 50.4718
    R18_67 I_D4A6F1242:N_2 I_D4A6F1246:N_2 99.3076
    R18_68 I_D4A6F1242:N_2 N_6:21 33.6671
    R18_69 I_D4A6F1242:N_2 N_6:20 19.2741
    R18_70 I_D4A6F1242:N_2 N_6:19 7.28683
    R18_71 I_D4A6F1242:N_2 I_D4A6F1244:N_2 423.334
    R18_72 I_D4A6F1242:N_2 N_6:18 8.85457
```

```
    R18_73 I_D4A6F1242:N_2 I_D4A6F1243:N_2 273.43
    R18_74 I_D4A6F1242:N_2 N_6:16 1119.29
    R18_75 I_D4A6F1242:N_2 N_6:17 24.2729
    R18_76 I_D4A6F1242:N_2 I_D4A6F1249:N_4 44440.5
    R18_77 I_D4A6F1242:N_2 I_D4A6F1249:N_3 860.141
    R18_78 I_D4A6F1243:N_2 I_D4A6F1248:N_2 22834.2
    R18_79 I_D4A6F1243:N_2 I_D4A6F1247:N_2 1433.49
    R18_80 I_D4A6F1243:N_2 I_D4A6F1246:N_2 2970.06
    R18_81 I_D4A6F1243:N_2 N_6:21 412.896
    R18_82 I_D4A6F1243:N_2 N_6:20 306.58
    R18_83 I_D4A6F1243:N_2 N_6:19 180.58
    R18_84 I_D4A6F1243:N_2 I_D4A6F1244:N_2 11842.9
    R18_85 I_D4A6F1243:N_2 N_6:18 8.88656
    R18_86 I_D4A6F1243:N_2 N_6:16 4182.99
    R18_87 I_D4A6F1243:N_2 N_6:17 46.6489
    R18_88 I_D4A6F1243:N_2 I_D4A6F1249:N_3 8217.01
    R18_89 I_D4A6F1244:N_2 I_D4A6F1248:N_2 0.665759
    R18_90 I_D4A6F1244:N_2 I_D4A6F1247:N_2 4228.64
    R18_91 I_D4A6F1244:N_2 I_D4A6F1246:N_2 8376.95
    R18_92 I_D4A6F1244:N_2 N_6:21 134.145
    R18_93 I_D4A6F1244:N_2 N_6:20 40.3356
    R18_94 I_D4A6F1244:N_2 N_6:19 8.22418
    R18_95 I_D4A6F1244:N_2 N_6:18 441.789
    R18_96 I_D4A6F1244:N_2 N_6:16 62550.3
    R18_97 I_D4A6F1244:N_2 N_6:17 768.519
    R18_98 I_D4A6F1244:N_2 I_D4A6F1249:N_3 62544.6
    R18_99 I_D4A6F1245:N_2 I_D4A6F1248:N_2 0.001
    R18_100 I_D4A6F1246:N_2 I_D4A6F1248:N_2 2811.91
    R18_101 I_D4A6F1246:N_2 I_D4A6F1247:N_2 183.78
    R18_102 I_D4A6F1246:N_2 N_6:21 265.122
    R18_103 I_D4A6F1246:N_2 N_6:20 172.083
    R18_104 I_D4A6F1246:N_2 N_6:19 76.5978
    R18_105 I_D4A6F1246:N_2 N_6:18 65.244
    R18_106 I_D4A6F1246:N_2 N_6:16 4951.26
    R18_107 I_D4A6F1246:N_2 N_6:17 162.332
    R18_108 I_D4A6F1246:N_2 I_D4A6F1249:N_4 185353.0
    R18_109 I_D4A6F1246:N_2 I_D4A6F1249:N_3 3305.97
    R18_110 I_D4A6F1247:N_2 I_D4A6F1248:N_2 204.975
    R18_111 I_D4A6F1247:N_2 N_6:21 111.441
    R18_112 I_D4A6F1247:N_2 N_6:20 70.5434
    R18_113 I_D4A6F1247:N_2 N_6:19 29.8808
    R18_114 I_D4A6F1247:N_2 N_6:18 25.8526
    R18_115 I_D4A6F1247:N_2 N_6:16 1735.95
    R18_116 I_D4A6F1247:N_2 N_6:17 67.7864
    R18_117 I_D4A6F1247:N_2 I_D4A6F1249:N_5 323993.0
    R18_118 I_D4A6F1247:N_2 I_D4A6F1249:N_4 62108.4
    R18_119 I_D4A6F1247:N_2 I_D4A6F1249:N_3 1120.33
    R18_120 I_D4A6F1248:N_2 N_6:21 1699.88
    R18_121 I_D4A6F1248:N_2 N_6:20 1079.34
    R18_122 I_D4A6F1248:N_2 N_6:19 457.189
    R18_123 I_D4A6F1248:N_2 N_6:18 395.556
    R18_124 I_D4A6F1248:N_2 N_6:16 25338.3
    R18_125 I_D4A6F1248:N_2 N_6:17 1037.16
    R18_126 I_D4A6F1248:N_2 I_D4A6F1249:N_3 17264.5
```

```
    R18_127 I_D4A6F1249:N_3 N_6:21 25.1331
    R18_128 I_D4A6F1249:N_3 N_6:20 32.9381
    R18_129 I_D4A6F1249:N_3 N_6:19 44.2285
    R18_130 I_D4A6F1249:N_3 N_6:18 74.9103
    R18_131 I_D4A6F1249:N_3 N_6:16 1323.16
    R18_132 I_D4A6F1249:N_3 N_6:17 120.632
    R18_133 I_D4A6F1249:N_3 I_D4A6F1249:N_5 8.08498
    R18_134 I_D4A6F1249:N_3 I_D4A6F1249:N_4 0.441411
    R18_135 I_D4A6F1249:N_3 N_6:22 5338.16
    R18_136 I_D4A6F1249:N_4 N_6:21 1372.14
    R18_137 I_D4A6F1249:N_4 N_6:20 1812.5
    R18_138 I_D4A6F1249:N_4 N_6:19 2419.61
    R18_139 I_D4A6F1249:N_4 N_6:18 4075.82
    R18_140 I_D4A6F1249:N_4 N_6:16 145896.0
    R18_141 I_D4A6F1249:N_4 N_6:17 6456.0
    R18_142 I_D4A6F1249:N_4 I_D4A6F1249:N_5 0.441645
    R18_143 I_D4A6F1249:N_4 N_6:22 291.443
    R18_144 I_D4A6F1249:N_5 N_6:21 26888.9
    R18_145 I_D4A6F1249:N_5 N_6:20 38974.0
    R18_146 I_D4A6F1249:N_5 N_6:19 43402.7
    R18_147 I_D4A6F1249:N_5 N_6:18 48574.4
    R18_148 I_D4A6F1249:N_5 N_6:17 166902.0
    R18_149 I_D4A6F1249:N_5 N_6:22 5.30109
    R18_150 N_6:16 N_6:21 304.777
    R18_151 N_6:16 N_6:20 230.918
    R18_152 N_6:16 N_6:19 143.389
    R18_153 N_6:16 N_6:18 56.5246
    R18_154 N_6:16 N_6:17 35.1863
    R18_155 N_6:16 N_6:22 12.5264
    R18_156 N_6:17 N_6:21 15.3979
    R18_157 N_6:17 N_6:20 11.547
    R18_158 N_6:17 N_6:19 6.92698
    R18_159 N_6:17 N_6:18 1.71642
    R18_160 N_6:18 N_6:21 9.23283
    R18_161 N_6:18 N_6:20 6.4401
    R18_162 N_6:18 N_6:19 3.27358
    R18_163 N_6:19 N_6:21 3.28982
    R18_164 N_6:19 N_6:20 1.51331
    R18_165 N_6:20 N_6:21 1.24306

    *|NET N_7 2.47472PF
    *|I (I_D4A6F1241:N_5 I_D4A6F1241 N_5 B 0 271.7200 281.8000)
    *|I (I_D4A6F12410:N_7 I_D4A6F12410 N_7 B 0 268.3600 273.9600)
    *|I (I_D4A6F12411:N_7 I_D4A6F12411 N_7 B 0 272.8400 266.1200)
    *|I (I_D4A6F12412:N_5 I_D4A6F12412 N_5 B 0 260.4950 273.9600)
    *|I (I_D4A6F12413:N_5 I_D4A6F12413 N_5 B 0 265.5600 281.8000)
    *|I (I_D4A6F1242:N_5 I_D4A6F1242 N_5 B 0 278.4400 281.8000)
    *|I (I_D4A6F1243:N_5 I_D4A6F1243 N_5 B 0 260.4950 273.7250)
    *|I (I_D4A6F1244:N_5 I_D4A6F1244 N_5 B 0 285.1600 273.9600)
    *|I (I_D4A6F1245:N_5 I_D4A6F1245 N_5 B 0 280.1200 273.9600)
    *|I (I_D4A6F1246:N_5 I_D4A6F1246 N_5 B 0 275.0800 273.9600)
    *|I (I_D4A6F1247:N_7 I_D4A6F1247 N_7 B 0 269.4800 266.1200)
    *|I (I_D4A6F1248:N_7 I_D4A6F1248 N_7 B 0 277.8800 266.1200)
    Cg19_1 I_D4A6F1247:N_7 0 8.13205e-15
```

```
    Cg19_2 N_7:13 0 5.02028e-13
    Cg19_3 N_7:14 0 3.24664e-13
    Cg19_4 N_7:15 0 2.52871e-13
    Cg19_5 N_7:16 0 1.95313e-13
    Cg19_6 N_7:17 0 2.74112e-13
    Cg19_7 N_7:18 0 5.23605e-13
    Cg19_8 N_7:19 0 3.93994e-13
    R19_1 I_D4A6F1241:N_5 I_D4A6F12413:N_5 1.0223
    R19_2 I_D4A6F1241:N_5 I_D4A6F1242:N_5 1.11524
    R19_3 I_D4A6F12410:N_7 I_D4A6F12412:N_5 1.30526
    R19_4 I_D4A6F12410:N_7 I_D4A6F1246:N_5 1.09315
    R19_5 I_D4A6F12411:N_7 I_D4A6F1248:N_7 0.814342
    R19_6 I_D4A6F12411:N_7 I_D4A6F1247:N_7 0.557618
    R19_7 I_D4A6F12412:N_5 N_7:16 21.9954
    R19_8 I_D4A6F12412:N_5 N_7:17 51.2135
    R19_9 I_D4A6F12412:N_5 I_D4A6F1247:N_7 44.0258
    R19_10 I_D4A6F12412:N_5 N_7:19 152.691
    R19_11 I_D4A6F12412:N_5 I_D4A6F1244:N_5 279.688
    R19_12 I_D4A6F12412:N_5 I_D4A6F1243:N_5 0.001
    R19_13 I_D4A6F12412:N_5 I_D4A6F1242:N_5 423.027
    R19_14 I_D4A6F12412:N_5 N_7:18 131.479
    R19_15 I_D4A6F12412:N_5 N_7:15 10.9819
    R19_16 I_D4A6F12412:N_5 I_D4A6F12413:N_5 37.03
    R19_17 I_D4A6F12412:N_5 N_7:13 19.2917
    R19_18 I_D4A6F12412:N_5 I_D4A6F1248:N_7 374.867
    R19_19 I_D4A6F12412:N_5 N_7:14 29.8337
    R19_20 I_D4A6F12413:N_5 N_7:16 25.1473
    R19_21 I_D4A6F12413:N_5 N_7:17 67.9629
    R19_22 I_D4A6F12413:N_5 I_D4A6F1247:N_7 66.1777
    R19_23 I_D4A6F12413:N_5 N_7:19 199.089
    R19_24 I_D4A6F12413:N_5 I_D4A6F1244:N_5 367.488
    R19_25 I_D4A6F12413:N_5 I_D4A6F1242:N_5 537.483
    R19_26 I_D4A6F12413:N_5 N_7:18 148.852
    R19_27 I_D4A6F12413:N_5 N_7:15 16.1952
    R19_28 I_D4A6F12413:N_5 N_7:13 23.6388
    R19_29 I_D4A6F12413:N_5 I_D4A6F1248:N_7 506.815
    R19_30 I_D4A6F12413:N_5 N_7:14 33.9576
    R19_31 I_D4A6F1242:N_5 I_D4A6F1247:N_7 633.843
    R19_32 I_D4A6F1242:N_5 N_7:16 9.52033
    R19_33 I_D4A6F1242:N_5 N_7:17 17.5926
    R19_34 I_D4A6F1242:N_5 N_7:19 78.5828
    R19_35 I_D4A6F1242:N_5 I_D4A6F1244:N_5 45.5611
    R19_36 I_D4A6F1242:N_5 N_7:18 135.523
    R19_37 I_D4A6F1242:N_5 N_7:15 45.3656
    R19_38 I_D4A6F1242:N_5 N_7:13 170.236
    R19_39 I_D4A6F1242:N_5 I_D4A6F1248:N_7 79.784
    R19_40 I_D4A6F1242:N_5 N_7:14 223.053
    R19_41 I_D4A6F1244:N_5 I_D4A6F1247:N_7 406.003
    R19_42 I_D4A6F1244:N_5 N_7:16 7.97393
    R19_43 I_D4A6F1244:N_5 N_7:17 10.7666
    R19_44 I_D4A6F1244:N_5 I_D4A6F1245:N_5 0.836426
    R19_45 I_D4A6F1244:N_5 N_7:19 52.1118
    R19_46 I_D4A6F1244:N_5 N_7:18 109.591
    R19_47 I_D4A6F1244:N_5 N_7:15 26.2368
```

```
    R19_48 I_D4A6F1244:N_5 N_7:13 112.524
    R19_49 I_D4A6F1244:N_5 I_D4A6F1248:N_7 44.1205
    R19_50 I_D4A6F1244:N_5 N_7:14 162.701
    R19_51 I_D4A6F1245:N_5 I_D4A6F1246:N_5 0.836426
    R19_52 I_D4A6F1247:N_7 N_7:16 37.0327
    R19_53 I_D4A6F1247:N_7 N_7:17 73.7041
    R19_54 I_D4A6F1247:N_7 N_7:19 223.757
    R19_55 I_D4A6F1247:N_7 N_7:18 224.715
    R19_56 I_D4A6F1247:N_7 N_7:15 14.48
    R19_57 I_D4A6F1247:N_7 N_7:13 30.7883
    R19_58 I_D4A6F1247:N_7 I_D4A6F1248:N_7 529.72
    R19_59 I_D4A6F1247:N_7 N_7:14 51.0496
    R19_60 I_D4A6F1248:N_7 N_7:16 13.8771
    R19_61 I_D4A6F1248:N_7 N_7:17 13.9098
    R19_62 I_D4A6F1248:N_7 N_7:19 72.6485
    R19_63 I_D4A6F1248:N_7 N_7:18 189.732
    R19_64 I_D4A6F1248:N_7 N_7:15 31.6989
    R19_65 I_D4A6F1248:N_7 N_7:13 150.777
    R19_66 I_D4A6F1248:N_7 N_7:14 241.095
    R19_67 N_7:13 N_7:17 15.3797
    R19_68 N_7:13 N_7:16 5.98382
    R19_69 N_7:13 N_7:19 34.1473
    R19_70 N_7:13 N_7:18 22.552
    R19_71 N_7:13 N_7:15 3.49965
    R19_72 N_7:13 N_7:14 1.72257
    R19_73 N_7:14 N_7:17 15.6025
    R19_74 N_7:14 N_7:16 3.41072
    R19_75 N_7:14 N_7:19 29.8287
    R19_76 N_7:14 N_7:18 6.46879
    R19_77 N_7:14 N_7:15 8.3581
    R19_78 N_7:15 N_7:17 3.12587
    R19_79 N_7:15 N_7:16 4.88766
    R19_80 N_7:15 N_7:19 6.42635
    R19_81 N_7:15 N_7:18 22.9723
    R19_82 N_7:16 N_7:17 2.52981
    R19_83 N_7:16 N_7:19 7.15396
    R19_84 N_7:16 N_7:18 3.37734
    R19_85 N_7:17 N_7:19 2.10058
    R19_86 N_7:17 N_7:18 4.49957
    R19_87 N_7:18 N_7:19 3.14418

    *|NET N_8 0.00531585PF
    *|I (I_D4A6F12411:N_5 I_D4A6F12411 N_5 B 0 275.6400 267.2400)
    *|I (I_D4A6F1242:N_4 I_D4A6F1242 N_4 B 0 282.0800 284.0400)
    *|I (I_D4A6F1247:N_4 I_D4A6F1247 N_4 B 0 271.7200 268.9200)
    Cg20_1 I_D4A6F1242:N_4 0 5.26903e-15
    Cg20_2 N_8:6 0 4.68201e-17
    R20_1 I_D4A6F12411:N_5 N_8:5 0.001
    R20_2 I_D4A6F12411:N_5 I_D4A6F1247:N_4 34.1615
    R20_3 I_D4A6F12411:N_5 I_D4A6F1242:N_4 39.4193
    R20_4 I_D4A6F1242:N_4 I_D4A6F1247:N_4 136.829
    R20_5 I_D4A6F1247:N_4 N_8:4 0.001
    R20_6 N_8:5 N_8:6 0.00696429
    R20_7 N_8:5 N_8:7 0.0740442
```

```
*|NET N_9 0.002256PF
*|I (I_D4A6F1246:N_3 I_D4A6F1246 N_3 B 0 275.9200 276.2000)
*|I (I_D4A6F1248:N_8 I_D4A6F1248 N_8 B 0 278.1775 267.8000)
Cg21_1 I_D4A6F1246:N_3 0 2.256e-15
R21_1 I_D4A6F1246:N_3 I_D4A6F1248:N_8 28.5243

*
* Instance Section
*
XI_D4A6F1241 I_D4A6F1241:N_2 I_D4A6F1241:N_3 I_D4A6F1241:N_4
   I_D4A6F1241:N_5 N__generated_22 CKXOR2D0BWP7T
XI_D4A6F12410 I_D4A6F12410:N_2 I_D4A6F12410:N_3 I_D4A6F12410:N_4
    N__generated_26 I_D4A6F12410:N_6 I_D4A6F12410:N_7
   I_D4A6F12410:N_8 MOAI22D0BWP7T
XI_D4A6F12411 I_D4A6F12411:N_2 I_D4A6F12411:N_3 I_D4A6F12411:N_4
    I_D4A6F12411:N_5 I_D4A6F12411:N_6 I_D4A6F12411:N_7
   I_D4A6F12411:N_8 MOAI22D0BWP7T
XI_D4A6F12412 I_D4A6F12412:N_2 I_D4A6F12412:N_3 I_D4A6F12412:N_4
    I_D4A6F12412:N_5 I_D4A6F12412:N_6 XNR2D1BWP7T
XI_D4A6F12413 I_D4A6F12413:N_2 I_D4A6F12413:N_3 I_D4A6F12413:N_4
    I_D4A6F12413:N_5 I_D4A6F12413:N_6 XNR2D1BWP7T
XI_D4A6F1242 I_D4A6F1242:N_2 I_D4A6F1242:N_3 I_D4A6F1242:N_4
   I_D4A6F1242:N_5 N__generated_23 CKXOR2D0BWP7T
XI_D4A6F1243 I_D4A6F1243:N_2 I_D4A6F1243:N_3 I_D4A6F1243:N_4
   I_D4A6F1243:N_5 I_D4A6F1243:N_6 CKXOR2D0BWP7T
XI_D4A6F1244 I_D4A6F1244:N_2 I_D4A6F1244:N_3 I_D4A6F1244:N_4
   I_D4A6F1244:N_5 N__generated_24 CKXOR2D0BWP7T
XI_D4A6F1245 I_D4A6F1245:N_2 I_D4A6F1245:N_3 I_D4A6F1245:N_4
   I_D4A6F1245:N_5 N__generated_25 CKXOR2D0BWP7T
XI_D4A6F1246 I_D4A6F1246:N_2 I_D4A6F1246:N_3 I_D4A6F1246:N_4
   I_D4A6F1246:N_5 I_D4A6F1246:N_6 CKXOR2D0BWP7T
XI_D4A6F1247 I_D4A6F1247:N_2 I_D4A6F1247:N_3 I_D4A6F1247:N_4
   I_D4A6F1247:N_5 I_D4A6F1247:N_6 I_D4A6F1247:N_7 I_D4A6F1247:
   N_8 AOI22D0BWP7T
XI_D4A6F1248 I_D4A6F1248:N_2 I_D4A6F1248:N_3 I_D4A6F1248:N_4
   I_D4A6F1248:N_5 I_D4A6F1248:N_6 I_D4A6F1248:N_7 I_D4A6F1248:
   N_8 AOI22D0BWP7T
XI_D4A6F1249 N_4 I_D4A6F1249:N_3 I_D4A6F1249:N_4 I_D4A6F1249:N_5
    PVSS1CDG

.ENDS
```

Cuadro 95: Archivo .spf original obtenido para un sumador de 4 bits luego de LPE