Corrección de anillo de entradas/salidas y pruebas de antenna y ERC para la definición del flujo de diseño del primer chip con tecnología nanométrica desarrollado en Guatemala

Marvin Geovanni Flores Espino

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Corrección de anillo de entradas/salidas y pruebas de antenna y ERC para la definición del flujo de diseño del primer chip con tecnología nanométrica desarrollado en Guatemala

Trabajo de graduación presentado por Marvin Geovanni Flores Espino
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2021

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Corrección de anillo de entradas/salidas y pruebas de antenna y ERC para la definición del flujo de diseño del primer chip con tecnología nanométrica desarrollado en Guatemala

Trabajo de graduación presentado por Marvin Geovanni Flores Espino
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2021

Vo.Bo.:

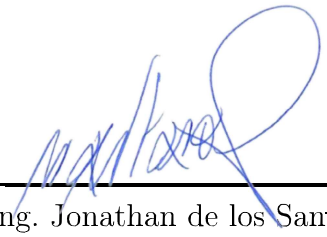
(f) _____

Ing. Carlos Esquit

Tribunal Examinador:

(f) _____

Ing. Carlos Esquit



(f) _____

Ing. Jonathan de los Santos

(f) _____

Ing. Kurt Kellner

Fecha de aprobación: Guatemala, 15 de enero de 2021.

Prefacio

Agradezco principalmente a doña Isabel Gutiérrez de Bosch (QEPD), ya que gracias a su compromiso con la educación de Guatemala fue posible que pueda estudiar en esta universidad, recibiendo la educación de más alta calidad en el país. Agradezco a todo su equipo de la Fundación Juan Bautista Gutiérrez por los seguimientos mensuales y demás apoyo brindado en todos los aspectos para que pudiera lograr este objetivo.

Agradezco también al Ing. Miguel Zea por el asesoramiento para llevar a cabo este trabajo, al M. Sc. Carlos Esquit por promover la investigación en el campo de la nanoelectrónica y al Ing. Luis Nájera por su apoyo en la resolución de dudas de este proyecto.

Por último quiero agradecer a mi familia, la cual anheló siempre lo mejor para mi vida y me apoyó en todo momento.

Índice

Lista de figuras

Lista de cuadros

Resumen

El flujo de diseño de un circuito integrado puede definirse como una serie de pasos a seguir para la obtención de los planos utilizados para su fabricación a partir de su diseño en un lenguaje descriptor de hardware. Este flujo de diseño se puede dividir en dos grandes secciones: síntesis lógica y síntesis física.

Este trabajo se enfoca en la implementación del anillo de entradas/salidas y en la corrida de pruebas de antena y reglas de diseño eléctrico, las cuales son verificaciones físicas que el circuito integrado debe pasar para poder ser fabricado.

En el proceso de implementación del anillo, se escogen los *pads* de entrada/salida adecuados para la aplicación que necesitamos, también se crean los *pads* de voltaje y tierra.

También se busca evitar los efectos de antena, los cuales pueden dañar los *gates* de los transistores durante la fabricación de pasos de grabado de plasma a través de la acumulación de exceso de carga en cables metálicos que no están conectados a los nodos de unión PN, estos efectos pueden aumentar la fuga del *gate* de un transistor, cambiar el voltaje de umblar y reducir el tiempo de vida del mismo. Para realizar las pruebas de antena, se ejecutan dos archivos principales: el runset de antena en la herramienta IC Validator y el archivo con las reglas de antena, en la herramienta IC Compiler.

Por último, se verifican las reglas de diseño eléctico o *ERC*, son seis reglas principales las que se verifican: *Path check, ntap check, MOS s/d power and ground check, Gate directly connecting to power or ground, Floating Gate y Floating Well.* Estas seis reglas que se describen en detalle más adelante se verifican en el proceso de *LVS.* Al realizar las verificaciones de estas reglas se generan varios archivos de resultados, el archivo que sirve para interpretar los resultados de las reglas de diseño eléctrico son los errores de *Layout.*

Abstract

The design flow of an integrated circuit can be defined as a series of steps to follow to obtain the plans used for its manufacture from its design in a hardware descriptor language. This design flow can be divided into two main sections: logical synthesis and physical synthesis.

This work focuses on the implementation of the input / output ring and the antenna test run and Electrical Design Rules, which are physical verifications that the integrated circuit must pass in order to be manufactured.

In the process of implementing the ring, the appropriate input / output pads are chosen for the application we need, the voltage and ground pads are also created.

It also seeks to avoid the effects of antenna, which can damage the gates of the transistors during the manufacture of plasma engraving steps through the accumulation of excess charge in metallic cables that are not connected to the PN junction nodes, These effects can increase the leakage of the gate of a transistor, change the threshold voltage and reduce the lifetime of the transistor. To perform the Antenna tests, two main files are run: the antenna runset in the IC Validator tool and the file with the antenna rules, in the IC Compiler tool.

Finally, the electrical design rules or ERC are verified, there are six main rules that are verified: path check, ntap check, MOS s / d power and ground check, gate direct connect to power or ground, floating gate and floating well . These six rules described in detail later are verified in the LVS process. When verifying these rules, several result files are generated, the file that is used to interpret the results of the electrical design rules are the layout errors.

CAPÍTULO 1

Introducción

Un circuito integrado debe pasar por un flujo de diseño. Este flujo de diseño es una serie de pasos o módulos, cada módulo cumple una función específica dentro del proceso de diseño y es de suma importancia que cada módulo sea realizado con éxito ya que las corridas sin errores de cada uno de estos módulos hace posible la fabricación del mismo. Cada módulo realiza la tarea de implementación de una parte del circuito integrado o verifica la integridad del diseño en silicio del mismo.

Tanto la implementación de cada parte como la verificación de la integridad del diseño son partes fundamentales en el flujo de diseño, ya que nos da seguridad del funcionamiento deseado del circuito.

Este trabajo se enfoca en desarrollar y documentar la implementación del anillo de entradas/salidas y de la correcta ejecución de dos pruebas de integridad del diseño: antena y reglas de diseño eléctrico. Los resultados obtenidos son satisfactorios, ya que se logró la implementación del anillo y las corridas de ambas pruebas sin errores.

CAPÍTULO 2

Antecedentes

2.1 Antecedentes de la línea de investigación

El primer acercamiento del departamento de ingeniería electrónica y mecatrónica (y de la UVG en general) con el estudio de circuitos con tecnología nanométrica se dio en el año 2013, año en el cual el MSc. Carlos Esquit impartió el curso de *Introducción a diseño de sistemas VLSI*.

En el año 2014, el departamento de ingeniería electrónica y mecatrónica logró un acuerdo académico con la empresa Synopsis, acuerdo en el que se le permite el acceso al departamento a sus herramientas y librerías de estudiantes para poder hacer procesos de diseño más completos. El primer trabajo de graduación relacionado con nanoelectrónica data de ese mismo año, siendo el diseño de un sumador/restador elaborado con herramientas de synopsis con librerías de tecnología de 28 nanómetros[2].

2.2 Antecedentes de este trabajo

La base de este proyecto inició en 2019, cuando un grupo de cuatro estudiantes iniciaron el proceso de definición del flujo de diseño del chip.

El grupo saliente encontró que este proceso puede dividirse dos categorías: síntesis lógica (Front-End) y física (Back-End). Además, plantearon un esquema a seguir tomando en cuenta fabricantes y herramientas disponibles, generaron guías de instalación y uso de las herramientas para cada parte del flujo de diseño.

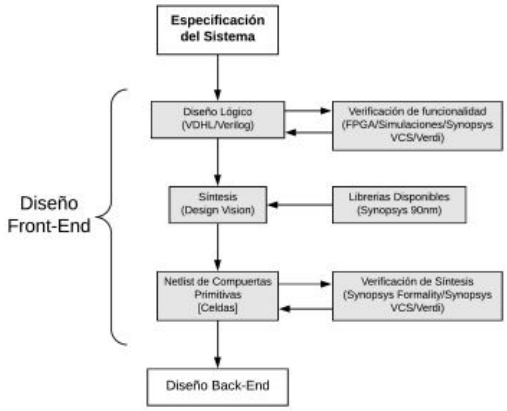


Figura 1: Fase front-end del flujo de diseño dejado por grupo saliente [1].

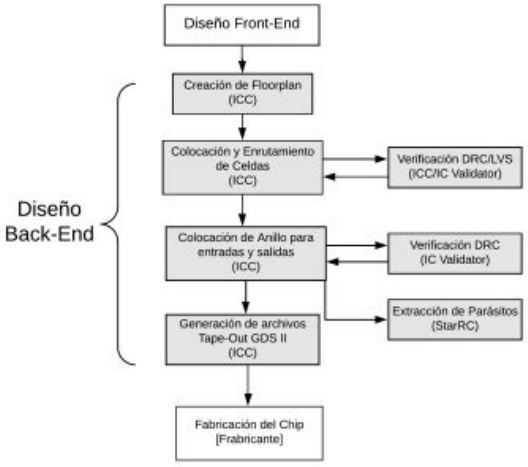


Figura 2: Fase back-end del flujo de diseño dejado por grupo saliente [1].

El flujo de diseño inició con ayuda de las herramientas de synopsis. Además, este grupo aplicó el proceso de diseño con librerías de TSMC, empresa que fabricará el chip. Sin embargo, el chip aún cuenta con una serie de errores que no permiten su fabricación. Los errores del anillo de entradas/salidas y la correcta corrida de las pruebas Antenna y Electrical Rule Check (ERC) forman el objetivo de este trabajo de graduación.

CAPÍTULO 3

Justificación

Como se mencionó en los antecedentes, la fabricación del chip ha sido imposible gracias a los errores dados por el *Design Rule Check (DRC)*, parte de estos errores se deben al diseño incorrecto del anillo de entradas/salidas, específicamente la conexión del core con los pads VDD Y VSS (Figura 3).

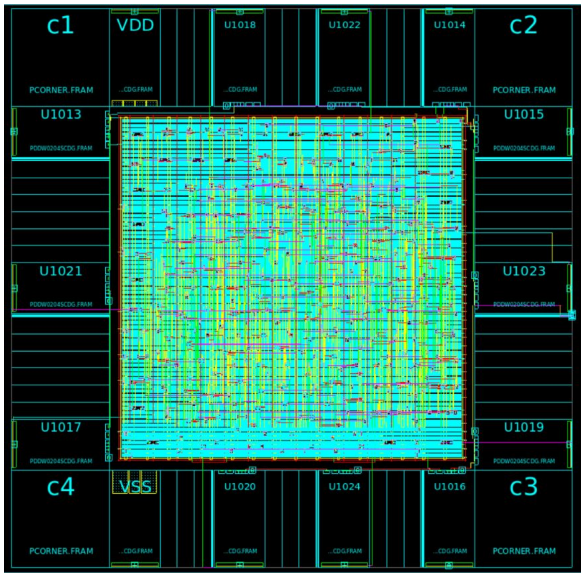


Figura 3: Chip sintetizado, listo para verificación de errores.

Más allá de arreglar los errores, es muy importante entender que el anillo de entradas/salidas forma una parte fundamental del chip, ya que cuenta con los pines que tendrán contacto con el mundo exterior, cada pin de entrada debe cumplir con las restricciones necesarias y cada pad de entrada y salida debe ser correctamente elegido, ya que cada pad cuenta con diferentes comportamientos en las salidas al aplicar las mismas entradas.

Existe la posibilidad de verificar que no existan sobrecargas en los alambres de metal que no están conectados a los canales PN de los transistores[3] y que las conexiones de power y ground sean correctas[3], para esto usamos las verificaciones de Antenna y ERC, respectivamente. Esto es importante ya que al ejecutarse sin errores nos da la seguridad de que las reglas de diseño se cumplen para estos análisis.

Elaborar el flujo de diseño de un circuito integrado puede tener un impacto muy importante en la tecnología del país. El flujo permite realizar un circuito integrado para una aplicación específica, lo que permite ya no depender de los circuitos integrados comerciales, que son generalmente de propósito general.

La ventaja de un flujo bien diseñado es que es siempre el mismo, por lo que para futuras investigaciones bastaría con cambiar el código de descripción de hardware y aplicar los pasos del flujo ya establecidos. Un flujo de diseño bien definido puede automatizarse, por lo que el proceso puede tomar menos tiempo y los futuros estudiantes pueden dedicarse a realizar aplicaciones más complejas o a aprender a utilizar nuevas herramientas y librerías.

La elaboración de circuitos integrados para aplicaciones específicas puede representar la base para el desarrollo de la tecnología nanométrica en Guatemala, lo que puede abrir brechas laborales en este campo.

CAPÍTULO 4

Objetivos

4.1 Objetivo general de la línea de investigación

Entregar un flujo de diseño bien definido y funcional para la fabricación de chips con tecnología nanométrica, para que sea utilizado por futuros estudiantes de la UVG.

4.2 Objetivo general de este trabajo

Realizar un anillo de entradas/salidas funcional para un circuito integrado con tecnología nanométrica y verificación de efecto de antena y reglas de diseño eléctrico.

4.3 Objetivos específicos

- Estudiar flujo de diseño anterior para comprender la sección de I/O Ring, antena y ERC.
- Estudiar a profundidad la documentación de Synopsis y TSMC para ver los comandos que pueden ser útiles para dejar un I/O ring funcional libre de errores, y para la elaboración de las verificaciones de antena y reglas de diseño eléctrico.
- Elaborar el script funcional para el anillo de entradas/salidas.
- Elaborar corridas de antena y ERC sin errores.
- Documentar el uso correcto de esta parte del flujo de diseño para futuros estudiantes de la UVG.

CAPÍTULO 5

Alcance

Debido a la pandemia del Covid-19, el departamento de electrónica de UVG gestionó con la empresa TSMC el permiso legal de acceder remotamente a las computadoras del laboratorio de la UVG. Dado el trabajo que esto implica, como instalación y configuración del software en todas las computadoras de la nube, instalación y gestión de la herramienta de acceso remoto Splashtop, las pruebas se retrasaron hasta finales de julio de 2020. El acceso remoto se hizo por medio de la aplicación Splashtop business.

El retraso producido por la pandemia del Covid-19 provocó limitaciones experimentales, ya que no se le pudo hacer pruebas de rendimiento o *brenchmarking* al circuito integrado (como estaba originalmente planeado) sino que la totalidad de tiempo de trabajo se utilizó para realizar las pruebas en software y arreglar los errores de diseño del chip. Tanto la fabricación como las pruebas de rendimiento del chip se dejan como trabajo futuro en esta línea de investigación.

Por otra parte, como se mencionó en el objetivo específico 5, este trabajo servirá como referencia para futuros estudiantes de la UVG. Este trabajo formará parte de la documentación de todo el flujo de diseño de circuitos integrados que se realicen en la universidad. Aunque cada chip sea diseñado y fabricado para distintos propósitos, todos los chips deben ser diseñados con el mismo flujo de diseño.

CAPÍTULO 6

Marco teórico

6.1 Very Large-Scale Integration VLSI

Desde la invención del transistor bipolar en 1947, ha habido un gran crecimiento en la industria de los semiconductores[5]. El área mas fuerte de la industria de los semiconductores ha sido la tecnología VLSI. El término VLSI se refiere al proceso de crear un chip integrado por millones de transistores en un solo circuito integrado.

En la Figura 4 podemos observar los elementos más importantes en el desarrollo de la tecnología VLSI ordenados cronológicamente.

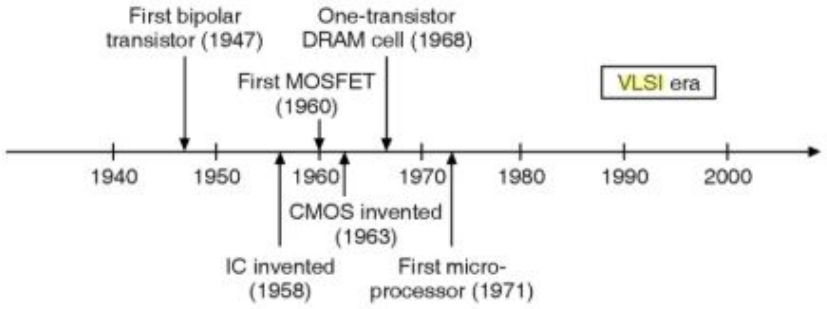


Figura 4: Cronología de eventos sobresalientes en el desarrollo de la tecnología VLSI [5].

6.2 Flujo de diseño

El flujo de diseño comprende una serie de pasos con el objetivo de fabricar un chip a partir de las especificaciones del sistema. El flujo de diseño puede separarse en dos etapas: síntesis lógica y síntesis física[1].

La síntesis lógica comprende la descripción del comportamiento lógico del sistema, a nivel *behavioural*. La síntesis lógica comprende la fase front-end del flujo de diseño. La síntesis física describe el funcionamiento del sistema a nivel estructural o *structural*. La síntesis física comprende la fase back-end del flujo de diseño **(Rubio)**.

En la Figura 5 podemos observar un flujo de diseño estándar para tecnología VLSI, con especial énfasis en los pasos del diseño físico. El diseño físico es la parte del flujo de diseño en la que se enfoca este trabajo.

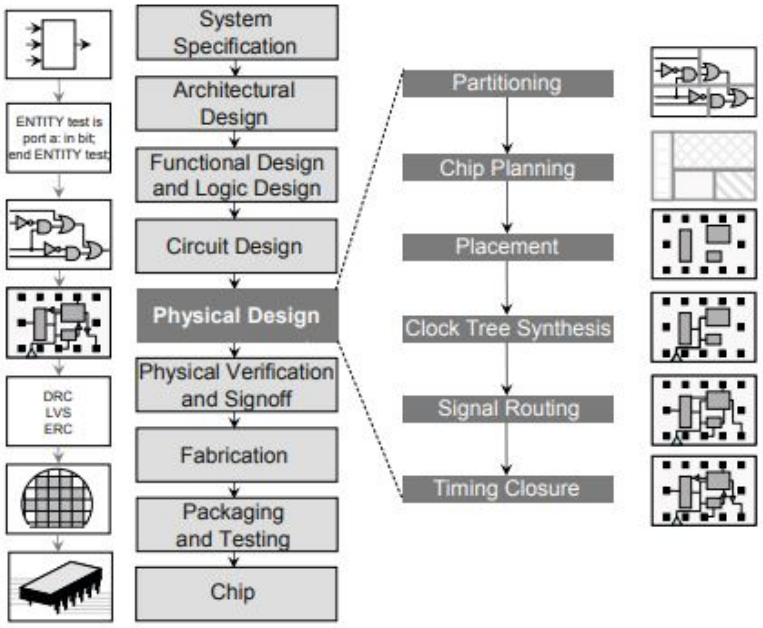


Figura 5: Flujo de diseño para un circuito integrado [3].

El proceso de front-end, es decir la síntesis lógica, no da como resultado un archivo RTL (Register Transfer level) a partir de un diseño hecho en un lenguaje descriptor de hardware HDL, en este caso verilog. Luego, en la síntesis física se obtiene una descripción física del sistema a partir de su descripción lógica, esta descripción física será nuestro *Layout*.[1]

6.2.1 Diseño Front-End

El flujo de diseño inicia especificando la funcionalidad del sistema, como se puede observar en la Figura 6.

Podemos definir este proceso como una serie de pasos:

- Diseño lógico: En esta parte diseñaremos nuestro circuito en un HDL. Por simplicidad, suele hacerse de forma behavioural. Este archivo es el RTL .
- Síntesis: Para esta parte del proceso de utiliza la herramienta *Design Vision*, la cual convierte el archivo RTL en una lista de compuertas lógicas, conocida como Netlist .

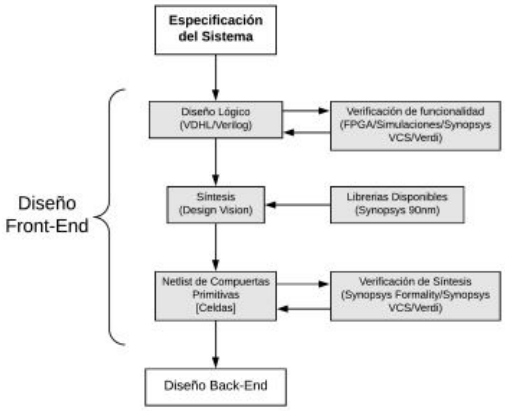


Figura 6: Diseño front-end.

- Netlist de compuertas: Es el resultado del paso anterior, es una lista de compuertas lógicas ordenadas de manera estructural y referenciadas a librerías existentes (pueden ser genéricas como las de synopsis o las comerciales que utiliza TSMC) .

Cabe mencionar que en cada paso se pueden hacer verificaciones para asegurar el correcto funcionamiento y sintetizado del circuito, la verificación del diseño lógico puede hacerse con simulaciones y software como *Synopsis VCS* o *Verdi*, o con herramientas físicas como un field-programmable gate array *FPGA*. La verficación del correcto sintetizado puede realizarse con herramientas como *Formality* [1].

6.2.2 Diseño back-end

. El diseño back-end comprende la síntesis física del circuito. A partir del Netlist obtenemos un layout con la topología del circuito, los pasos de esta etapa pueden observarse en la Figura 7.

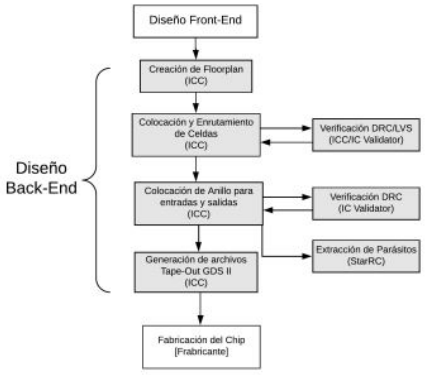


Figura 7: Diseño back-end.

Al igual que el diseño front-end, esta etapa se define como una serie de pasos:

- Placement: Este paso consiste en colocar las celdas de un diseño en un **Floorplan**.
- Routing: Una vez hecho todo el placement, las celdas deben interconectarse, esto se hace por medio de **nets**, este proceso es conocido como enrutamiento.
- Colocación de anillo de entradas y salidas: Esta parte del diseño comprende la correcta elección de **PADS** y su conexión, estos pines serán el contacto con el mundo exterior.
- Extracción de parásitos: es un proceso en el cual se extraen archivos que describen todos los elementos capacitivos o resistivos asociados a cada nodo del layout.

Al igual que en la síntesis lógica, en la síntesis física también se pueden hacer verificaciones usando la herramieta IC Validator, con esta herramienta podemos hacer pruebas de ERC, LVS, antena y ERC:.

6.3 IC Compiler

IC Compiler es un software capaz de realizar la fase back-end del flujo de diseño, es decir la síntesis física. Como entrada le suministramos a la herramienta un Netlist: (resultado del diseño front-end) para llevarlo a los archivos finales de fabricación GDSII. Combina la síntesis física y el enrutamiento para implementaciones de diseño lógico y físico en todo el flujo de diseño[6].

6.4 IC Validator

IC Validator o ICV se ejecuta con base en runsets. Los Runset: contienen información del layout como las especificaciones de componentes, asignacion de capa de metales, verificaciones de la base de datos del diseño ylas verificaciones físicas[7].

IC Validator genera reportes sobre los análisis, indicando si se cumplen o no las reglas de diseño para cada uno.[1]

6.5 Anillo de entradas/salidas

El anillo de entradas/salidas es la parte del circuito integrado que contiene los para la interacción con el mundo exterior, este puede contener pines de **Tierra o VSS:**, **Voltaje de alimentación o VDD:** y los pines de entrada y salida necesarios.

Ya que los PADS de entrada/salida del chip se encuentran sin errores, nos centraremos en los pines VDD y VSS.

El enrutamiento de VDD y VSS en circuitos integrados digitales modernos generalmente tiene una topología de malla que es creado a través de los siguientes cinco pasos[3]:

Paso 1. Crear el anillo

Generalmente el anillo de entradas/salidas se construye para rodear al core del chip. El propósito de este anillo es conectar las celdas de entradas/salidas y posiblemente estructuras de protección contra descargas electrostáticas. Para baja resistencia, estas conexiones y el anillo en sí están en muchas capas de metal, por ejemplo el anillo puede usar las capas de metal2 a la capa metal8 (cada capa excepto la metal1) [3].

Paso 2. Conectar los pads I/O al anillo

Cada Pad de entrada/salida tendrá una cantidad de "dedos"que van en cada una de varias capas de metal. Estos deben estar conectados al máximo al anillo de potencia para minimizar resistencia y maximizar la capacidad de llevar corriente al núcleo. En la parte superior izquierda de la Figura 8 podemos observar las conexiones de los pads hacia el anillo [3].

Paso 3. Crear una malla

La malla de poder consiste en un conjunto de franjas en pasos definidos en dos o más capas (Figura 8). El ancho de las rayas se determinan a partir de la potencia estimada. Las rayas están puestas en pares, alternando como VDD-GND, VDD-GND, etc.

La malla utiliza lac capas superiores y más gruesas, y es más escasa en cualquier capa inferior para evitar la congestión de enrutamiento. Las rayas en las capas adyacentes generalmente están conectadas con tantas vias como sea posible, con el fin de minimizar la resistencia [3].

Paso 4. Crear los rieles de Metal1

La capa de Metal1 es donde la distribución de VDD-VSS concurre con las compuertas lógicas del diseño. El ancho (capacidad de suministro de corriente) y el *pitch* de los rieles de metal1 generalmente están determinados por la *Standard cell Library*.

Paso 5. Conectar los rieles de Metal 1 a la malla

Los rieles Metal1 están conectados a la malla con vías apiladas. La Figura 8 ilustra el enfoque de malla para la distribución VDD-VSS [3].

6.6 Atenna Rule Check

Antenna Rule Check busca evitar los efectos de antena, que pueden dañar puertas de transistores durante la fabricación de pasos de grabado de plasma a través de la acumulación de exceso de carga en cables metálicos que no están conectados a los nodos de unión PN [3].

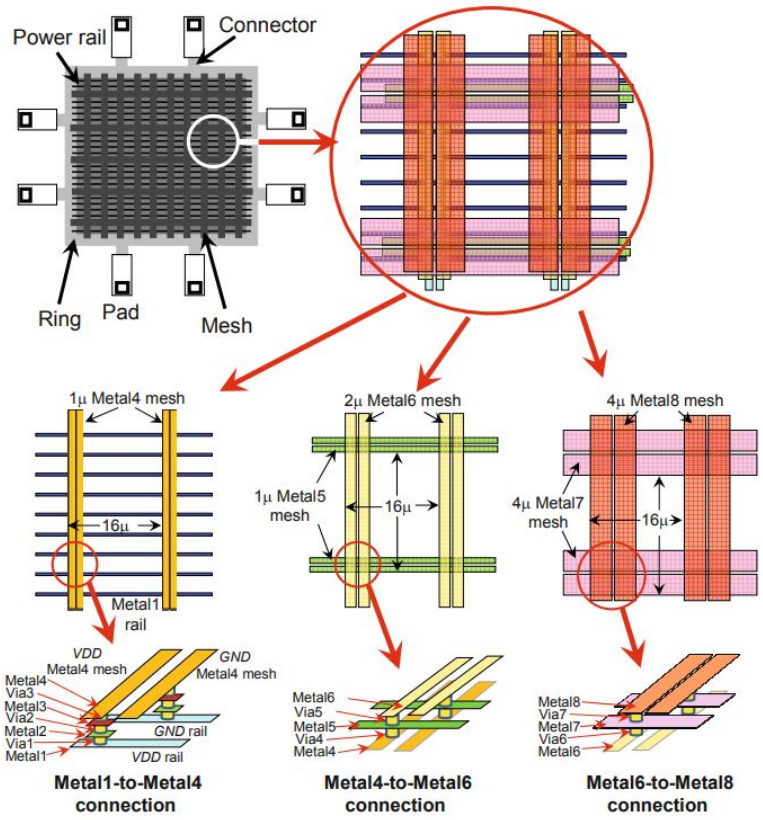


Figura 8: Construcción de malla y su distribución [3].

Antenna funciona a base de runsets y se realiza con la ayuda de la herramienta IC compiler.

Reglas de antena

Cuando un cable de metal en contacto con el gate de un transistor se graba en plasma, puede cargar voltajes suficientemente altos para eliminar los delgados óxidos del gate del transistor, a este efecto se le conoce daño de óxido de puerta inducido por plasma o simplemente efecto de antena [4]. Esto puede aumentar la fuga del gate, cambiar el voltaje de umblar y reducir el tiempo de vida de un transistor. Durante el proceso de grabado con plasma a altas temperaturas, los diodos formados por las difusiones de source y drain pueden conducir cantidades importantes de corriente, estos diodos se descargan de los cables antes que se dañe el óxido del gate [4].

Las reglas de antena especifican el área máxima de metal que se puede conectar a un gate sin el source o el drain para actuar como elemento de descarga.Las reglas de diseño normalmente definen la relación máxima entre el área de metal y el área del gate de un transistor de modo que la carga sobre el metal no dañe la puerta. Las relaciones pueden variar de 100:1 a 5000:1[4].

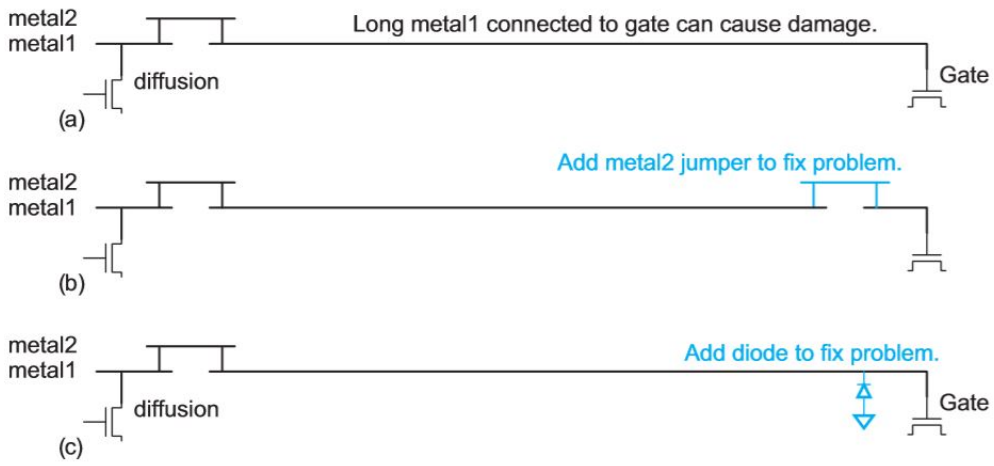


Figura 9: Ejemplo de error de antena y dos posibles soluciones.

En la Figura 20 se puede observar un ejemplo de error de antena y dos formas de solucionarlo. En la Figura 20(a), una línea larga de metal1 está conectada a un gate de un transistor, no tiene conexión la difusión hasta que se forma el metal2, por lo que el gate puede dañarse durante el proceso de grabado con plasma de metal1. En la Figura 20(b) la línea de metal1 se interrumpe con un puente de metal2, lo que causa que la cantidad de carga que podría golpear al gate durante el grabado de metal1 se reduzca, en la Figura 20(c) se agrega un diodo de antena, el cual funciona como ruta de descarga durante el proceso de grabado, este diodo tiene polarización inversa durante el funcionamiento normal, por lo que no interfiere con el funcionamiento del circuito[4].

6.7 Electrical Rule Check (ERC)

Electrical Rule Check es una opción especial para los diseñadores de circuitos a nanoescala [9], tiene la capacidad de verificar siete clases de reglas:

6.7.1 Soft Connect Check

Esta regla ayuda a los diseñadores a buscar cual contacto conecta con el N-Well:. Define si se pasa la conectividad establecida de los polígonos de la capa superior a los polígonos de la capa inferior especificados (unidireccional).[9]

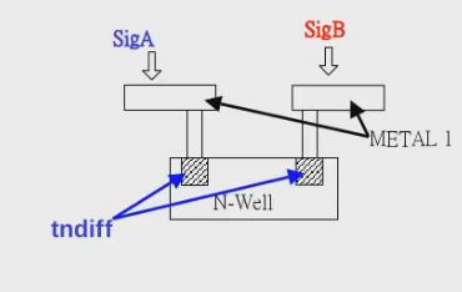


Figura 10: Ejemplo de polígonos.

6.7.2 Path Check

Esta regla ayuda a los diseñadores a verificar si se olvidó algo de los cuatro tipos de reglas que verifica [9] :

- Nodos con un conectado a alimentación, pero no a tierra.
- Nodos con un *Path* conectado a tierra, pero no a alimentación.
- Nodos sin un *Path* a tierra o a alimentación.
- Nodos sin un *Path* a cualquier etiqueta de una *Net:*.

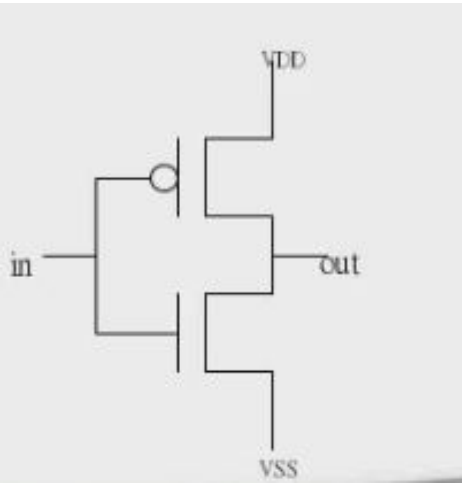


Figura 11: Ejemplo de etiquetas de net.

6.7.3 PTAP/NTAP connectivity check

Verifica si el PTAP está conectado a alimentación y si el NTAP está conectado a tierra[9].

6.7.4 MOSFET Power and Ground Check

Verifica que para cada mosfet tipo N y P, uno de ambos (*Source (Mosfet): y Drain (Mosfet):*) se conecte a alimentación y el otro a tierra[9].

6.7.5 Gate directly connected to power or ground

Esta regla sirve para la protección de descargas electrostáticas. Si el gate: de un mosfet tipo p está conectado directamente al nodo de alimentación, el *gate* será dañado.Si el *gate* de un mosfet tipo n está conectado directamente al nodo de tierra, el *gate* será dañado[9].

6.7.6 Floating Gate

Verifica si hay contactos interactuando con el *poly gate*[9].

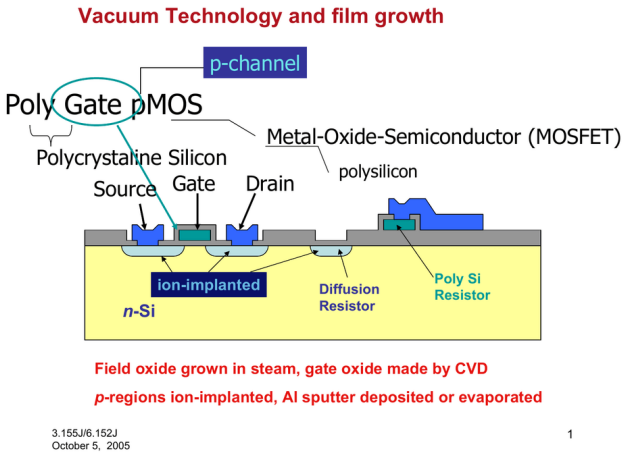


Figura 12: Ejemplo de etiquetas de poly gate.

6.7.7 Floating Well

Verifica si hay conexiones de alimentación o tierra con el *nwell* o el sustrato tipo p[9].

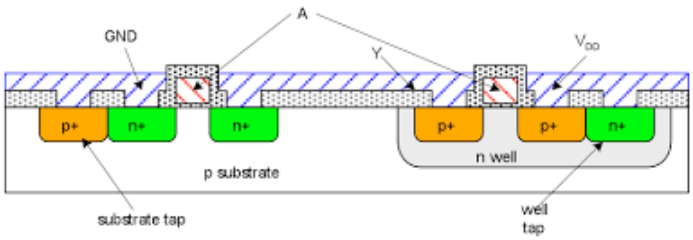


Figura 13: Ejemplo de nwell y sustrato tipo p.

CAPÍTULO 7

Materiales y métodos

7.1 Herramientas de software

El grupo saliente preparó el ecosistema para poder llevar a cabo el flujo de diseño. Este proceso implicó desde una investigación inicial sobre el flujo de diseño de circuitos ASIC y obtención e instalación de las herramientas y las librerías necesarias. Después de indagar en la página de la empresa Synopsis, el equipo encontró que el sistema operativo a usar era Centos 7 de Linux y las herramientas para hacer la síntesis lógica y física serían Design Vision e IC Compiler, respectivamente. Además, se tomaron en cuenta las herramientas VCS y Formality que permiten realizar las verificaciones de funcionamiento del circuito tanto antes como después de la síntesis, brindando así la seguridad de la correcta construcción[8]. El grupo saliente también instaló estos programas y obtuvo las librerías tanto de Synopsis como de TSMC para realizar el flujo de diseño con el circuito integrado con tecnología de 180nm.

7.1.1 Herramientas iniciales

El grupo saliente dejó instaladas las herramientas Design Vision, IC compiler, VCS y Formality, con runsets con comandos para generar los archivos del circuito integrado. También se obtuvieron las librerías con las que la empresa TSMC fabrica sus circuitos integrados, por lo que son de alta confidencialidad.

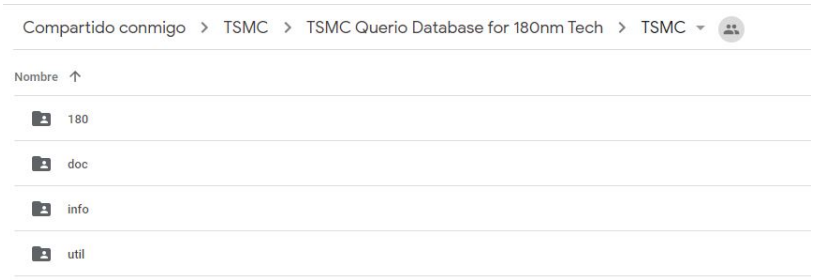


Figura 14: Organización de información de TSMC de 180nm.

7.2 Design Vision

En Design Vision se realiza la implementación del anillo de entradas y salidas, también es posible crear los *pads* de alimentación y tierra desde esta herramienta.

7.3 IC Compiler

Las pruebas de antena y reglas de diseño eléctrico se hacen en la herramienta IC Compiler e IC Validator, ya que forman parte del diseño back-end.

7.4 Métodos

Se accedió a la computadora con las herrramientas por medio de Splashtop, el cual es una herramienta de acceso remoto de distribución linux.

Para la realización de las pruebas, se siguió las siguientes metodologías:

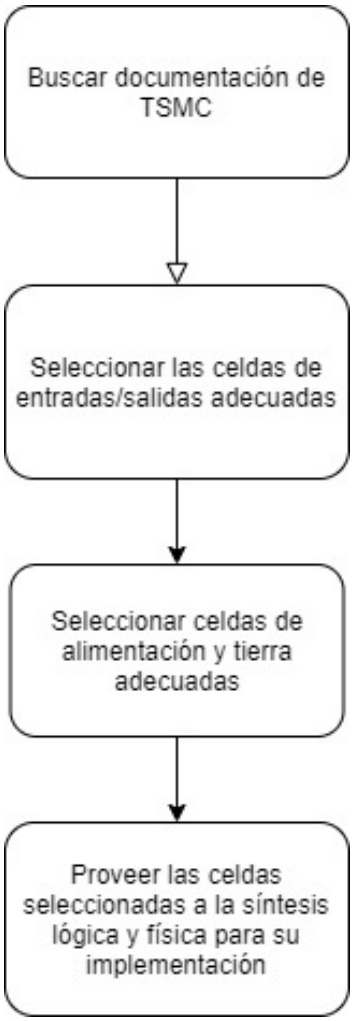


Figura 15: Metodología para el anillo de entradas/salidas.

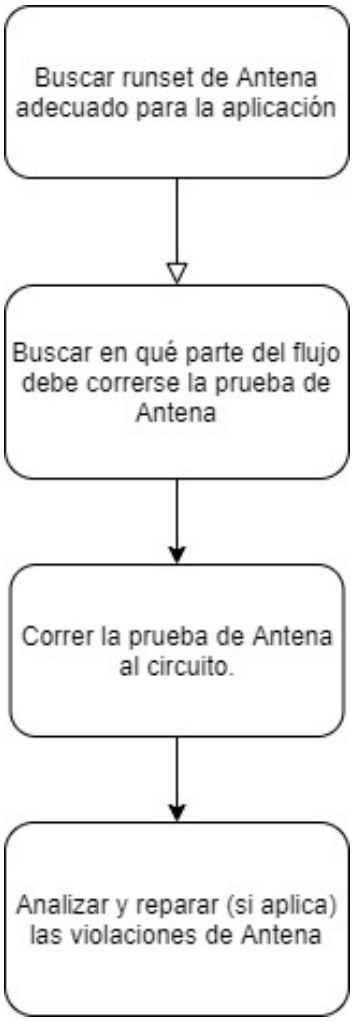


Figura 16: Metodología para la prueba de antena.

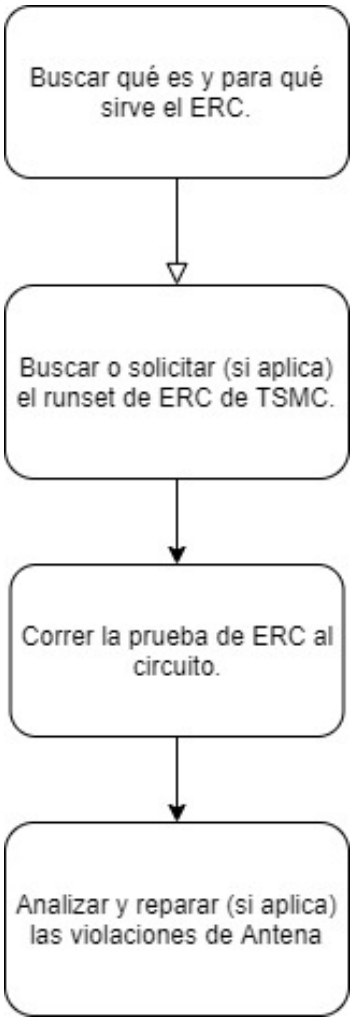


Figura 17: Metodología para la prueba de ERC.

Para estas metodologías se trabajaron tres circuitos antes de trabajar el circuito final: una Not, un Full Adder y un Ripple Carry Adder. Se escogió la Not ya que es el circuito más simple y con menos tendencia a errores para las pruebas, también se escogió el Full Adder y el Ripple Carry Adder debido a su fácil implementación y al ser más complejo que la Not dió un panorama más completo de todos los posibles errores que pueden existir al trabajar circuitos más complejos.

CAPÍTULO 8

Pruebas

8.1 Pruebas iniciales del flujo de diseño completo

La primer prueba se realizó replicando el flujo de diseño hasta donde lo dejó el grupo anterior con un Full Adder.

```
module Full_Adder_Structural_Verilog(
  input X1, X2, Cin,
  output S, Cout
  );
    wire a1, a2, a3;
    xor u1(a1,X1,X2);
 and u2(a2,X1,X2);
 and u3(a3,a1,Cin);
 or u4(Cout,a2,a3);
    xor u5(S,a1,Cin);
endmodule
```

Figura 18: Circuito Behavioural de un Full Adder.

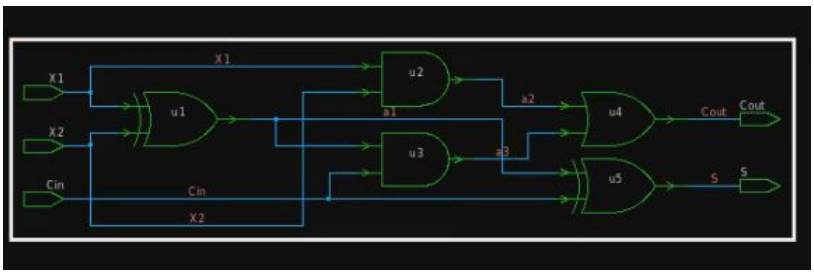


Figura 19: Vista esquemática del circuito Full Adder.

Al validar este circuito en el software Design Vision, devolvió un 1 lo que significa que el circuito está correcto.

```
design_vision> check_design
1
```

Figura 20: Validación del circuito en verilog con Design Vision.

Una vez sintetizado lógicamente en Design Vision, se observó que el circuito ya presenta componentes referenciados en las librerías de synopsis, estas librerías son genéricas, los componentes reales del fabricante a utilizar son los de la empresa TSMC.

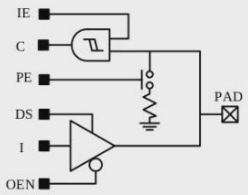
```
module Full_Adder_Structural_Verilog ( X1, X2, Cin, S, Cout );
  input X1, X2, Cin;
  output S, Cout;
  wire   n3, n4;

  CKXOR2D0 U5 ( .A1(n3), .A2(n4), .Z(S) );
  MOAI22D0 U6 ( .A1(n4), .A2(n3), .B1(X1), .B2(Cin), .ZN(Cout) );
  CKND0 U7 ( .I(X2), .ZN(n3) );
  XNR2D0 U8 ( .A1(X1), .A2(Cin), .ZN(n4) );
endmodule
```

Figura 21: Validación del circuito en verilog con Design Vision.

En este punto, se eligieron los in-outs correctos entre todos los que da el fabricante, en este caso TSMC. Se revisó en la documentación de TSMC para el Typical Case (hay best, typical y worst case) y se escogió el in-out correcto.

8.2 PDDW0204SCDG

Dual-Driving Regular I/O Cell with Schmitt Trigger Input, and Enable-Controlled Pull-Down Resistor



Truth Table

INPUT						OUTPUT	
DS	OEN	I	PAD	PE	IE	PAD	C
0/1	0	0	-	0/1	0	0	0
0/1	0	0	-	0/1	1	0	0
0/1	0	1	-	0/1	0	1	0
0/1	0	1	-	0/1	1	1	1
0/1	1	0/1	0	0/1	0	-	0
0/1	1	0/1	0	0/1	1	-	0
0/1	1	0/1	1	0/1	0	-	0
0/1	1	0/1	1	0/1	1	-	1
0/1	1	0/1	Z	0	0	-	0
0/1	1	0/1	Z	0	1	-	X
0/1	1	0/1	Z	1	0	L	0
0/1	1	0/1	Z	1	1	L	L

Figura 22: In-out seleccionado del fabricante.

Como se observa en la Figura 22, el objetivo es que lo que este en el pin PAD (contacto con el mundo exterior) llegue al pin C (contacto con el circuito), utilizando como guía la tabla de verdad, se precedió a colocarlos en el circuito.

```
module Full_Adder ( X1, X2, Cin, S, Cout,dummy);
  input X1, X2, Cin;
  output S, Cout;

  wire X1_w;
  wire X2_w;
  wire Cin_w;
  wire S_w;
  wire Cout_w;
  output [1:0] dummy;


  Full_Adder_Structural_Verilog U23 (X1_w, X2_w, Cin_w, S_w, Cout_w);

  PDDW0204SCDG U1001(1'b0,1'b0,1'b1, X1,X1_w,1'b0,1'b1);
  PDDW0204SCDG U1003(1'b0,1'b0,1'b1, X2,X2_w,1'b0,1'b1);
  PDDW0204SCDG U1007(1'b0,1'b0,1'b1, Cin,Cin_w,1'b0,1'b1);


  PDDW0204SCDG U2001(S,1'b0,1'b0,S_w,dummy[0],1'b0,1'b0);
  PDDW0204SCDG U2003(Cout,1'b0,1'b0,Cout_w,dummy[1],1'b0,1'b0);
  PVDD1CDG U666(VDD);
  PVSS1CDG U0666 (VSS);

endmodule
```

Figura 23: In-out del fabricante seleccionado.

Luego, se repitió el proceso con el circuito sintetizado para que Design Vision le coloque los PADS de entradas y salidas correctos referenciados en las librerías reales del fabricante.

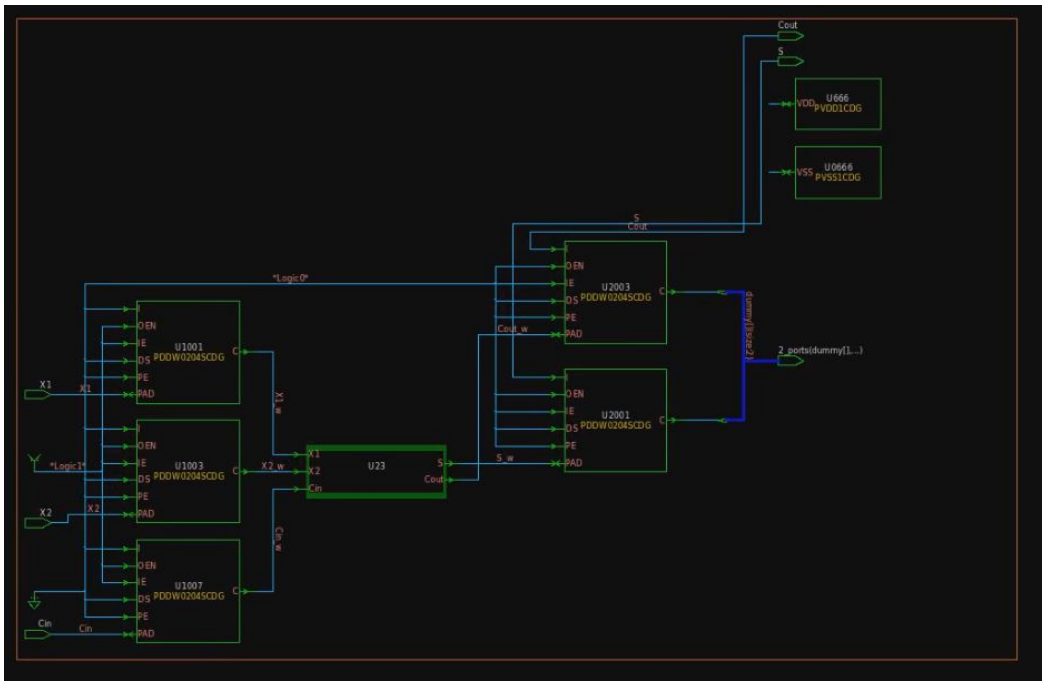


Figura 24: vista esquemática de los pads de entradas y salidas.

Como se observa en la Figura 25, ya tenemos la vista esquemática del circuito, pero con los pads de entradas y salidas ya colocados

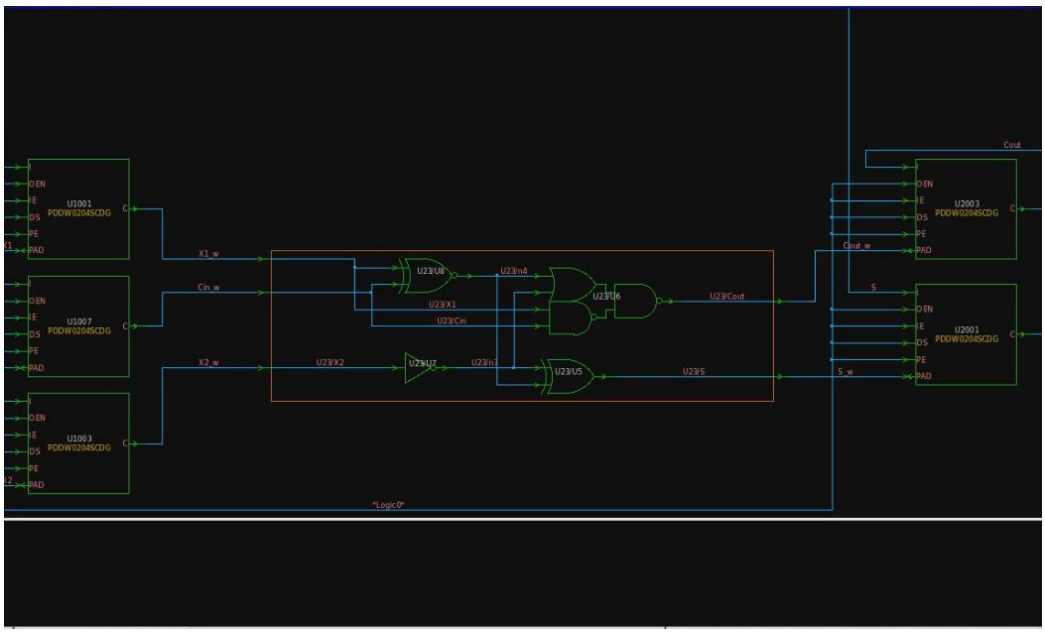


Figura 25: Vista esquimática del circuito con los pads de entradas y salidas.

Se chequeó el diseño y se compiló, se observa que Design Vision devolvió un 1, lo que indica que todo se realizó correctamente.

```
  Optimization Complete
  ---------------------
1
design_vision>
```

Figura 26: Circuito sintetizado y compilado correctamente.

Dando como resultado el circuito que será utilizado para la siguiente parte del flujo de diseño.

```
////////////////////////////////////////////////////////////////
// Created by: Synopsys DC Expert(TM) in wire load mode
// Version   : O-2018.06-SP5
// Date      : Mon Aug  3 14:39:34 2020
////////////////////////////////////////////////////////////////


module Full_Adder_Structural_Verilog ( X1, X2, Cin, S, Cout );
  input X1, X2, Cin;
  output S, Cout;
  wire   n1, n2;
  tri   S;
  tri   Cout;

  XOR2D4 U5 ( .A1(n1), .A2(n2), .Z(S) );
  XNR2D0 U6 ( .A1(X1), .A2(Cin), .ZN(n2) );
  MOAI22D4 U7 ( .A1(n2), .A2(n1), .B1(X1), .B2(Cin), .ZN(Cout) );
  CKND0 U8 ( .I(X2), .ZN(n1) );
endmodule


module Full_Adder ( X1, X2, Cin, S, Cout, dummy );
  output [1:0] dummy;
  input X1, X2, Cin;
  output S, Cout;
  wire   n4, X1_w, X2_w, Cin_w, n2;
  tri   X1;
  tri   X2;
  tri   Cin;
  tri   S_w;
  tri   Cout_w;
  assign Cout = n4;
  assign S = n4;

  Full_Adder_Structural_Verilog U23 ( .X1(X1_w), .X2(X2_w), .Cin(Cin_w), .S(
        S_w), .Cout(Cout_w) );
  PDDW0204SCDG U1001 ( .I(n4), .OEN(n2), .IE(n2), .PAD(X1), .DS(n4), .PE(n4),
        .C(X1_w) );
  PDDW0204SCDG U1003 ( .I(n4), .OEN(n2), .IE(n2), .PAD(X2), .DS(n4), .PE(n4),
        .C(X2_w) );
  PDDW0204SCDG U1007 ( .I(n4), .OEN(n2), .IE(n2), .PAD(Cin), .DS(n4), .PE(n4),
        .C(Cin_w) );
  PDDW0204SCDG U2001 ( .I(n4), .OEN(n4), .IE(n4), .PAD(S_w), .DS(n4), .PE(n4),
        .C(dummy[0]) );
  PDDW0204SCDG U2003 ( .I(n4), .OEN(n4), .IE(n4), .PAD(Cout_w), .DS(n4), .PE(
        n4), .C(dummy[1]) );
  TIEL U5 ( .ZN(n4) );
  TIEH U6 ( .Z(n2) );
endmodule
```

Figura 27: Circuito sintetizado completamente en Design Vision.

Al terminar la síntesis lógica, se procedió a ejecutar la síntesis física con el archivo "design.tcl"que dejó el gurpo saliente, este archivo contiene los comandos para que IC Compiler realice la síntesis física. El resultado de esta síntesis puede verse en la Figura 28.

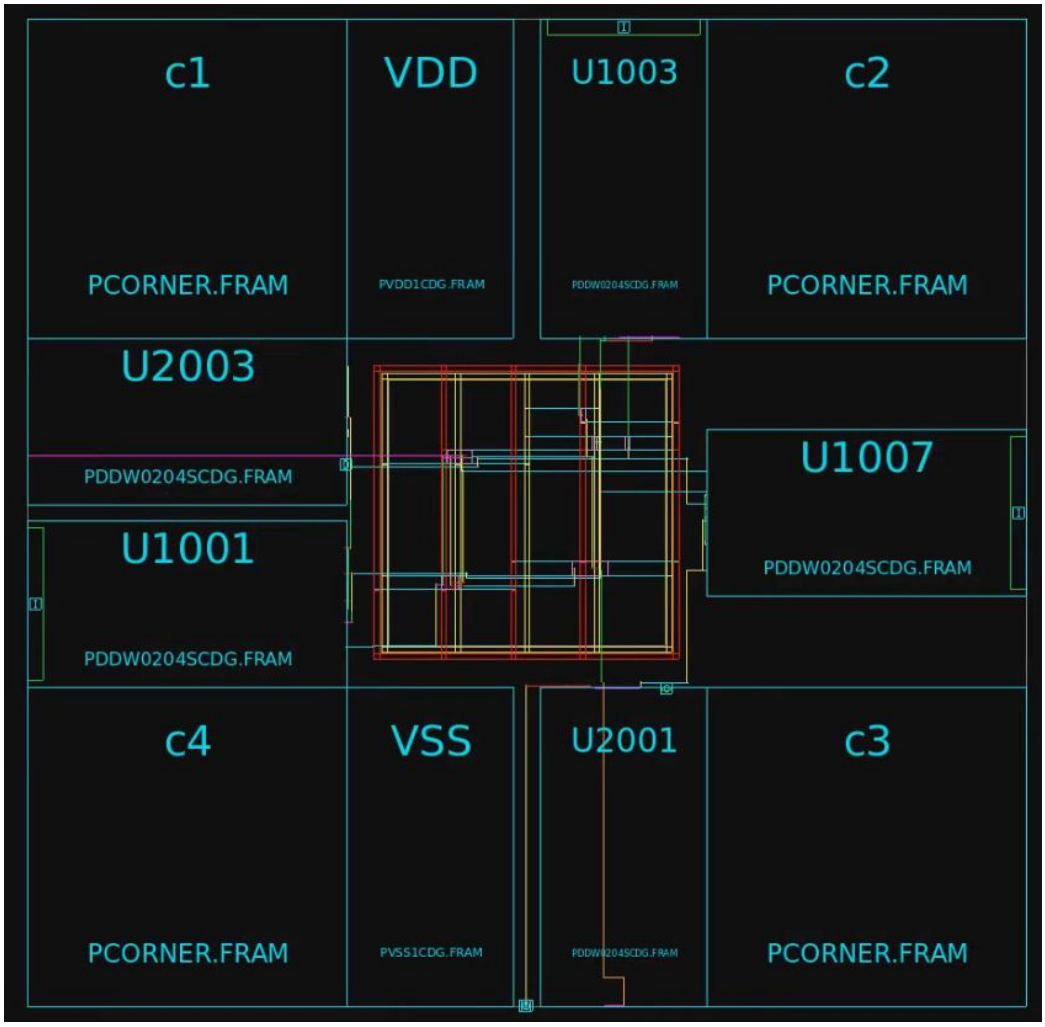


Figura 28: Circuito sintetizado físicamente en IC Compiler.

El problema en este punto es que al correr las pruebas de integridad como el DRC, el circuito presenta errores siento el más notable la falta de conexión hacia el pin VDD y VSS (alimentación y tierra,respectivamente). Hasta este punto logró llegar el grupo saliente en el flujo de diseño.

CAPÍTULO 9

Anillo de entradas y salidas

9.1 Celdas de entradas y salidas

TSMC envió los archivos necesarios para la correcta colocación de las celdas de entradas y salidas. Estos archivos son las librerías y la documentación de las mismas.

Para colocar los pads de entradas y salidas, se obtuvo la librería (con extensión .db), por lo que revisando el documento “AN 001 miniasic information 20170125” se concluyó que la librería que se utilizará en el flujo de diseño será la llamada "tpd018nv", la cual funciona a niveles de voltaje 1.8V/3.3V y es la librería estándar regular para pads de entradas y salidas.

TSMC provee cuatro distintos tipos de archivos .db correspondientes a la librería tpd018nv, estos cuatro archivos funcionan para worst case, best case y typical case, por lo que el archivo a utilizar es el tpd018nvtc el cual posee las características de un caso típico.

Se utilizó el documento "DB-TPD018NV-TC.pdf"para revisar la información de los pads, éstos contienen distintas descripciones funcionales, para esta aplicación se seleccionó la funcionalidad *Dual-Driving Regular I/O Cell with Schmitt Trigger Input, and Enable-Controlled Pull-Down Resistor*, existen ocho celdas con esta funcionalidad, por lo que se puede seleccionar cualquiera, en este trabajo en específico se escogió la celda PDDW0204SCDG.

9.2 Celda PDDW0204SCDG

Como se mencionó anteriormente, esta celda corresponde a un *Dual-Driving Regular I/O Cell with Schmitt Trigger Input, and Enable-Controlled Pull-Down Resistor*.

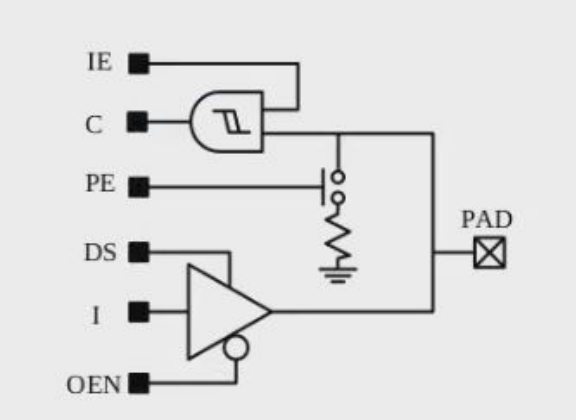


Figura 29: Circuito de la celda DDW0204SCDG.

En la Figura 29 se observa el circuito de la celda seleccionada para pad de entrada/salida. Lo que se necesitó es que la información que entre por el PAD (contacto del core del circuito con el mundo exterior) llegue al pin C (contacto con el circuito).

Truth Table

INPUT						OUTPUT	
DS	OEN	I	PAD	PE	IE	PAD	C
0/1	0	0	-	0/1	0	0	0
0/1	0	0	-	0/1	1	0	0
0/1	0	1	-	0/1	0	1	0
0/1	0	1	-	0/1	1	1	1
0/1	1	0/1	0	0/1	0	-	0
0/1	1	0/1	0	0/1	1	-	0
0/1	1	0/1	1	0/1	0	-	0
0/1	1	0/1	1	0/1	1	-	1
0/1	1	0/1	Z	0	0	-	0
0/1	1	0/1	Z	0	1	-	X
0/1	1	0/1	Z	1	0	L	0
0/1	1	0/1	Z	1	1	L	L

Figura 30: Tabla de verdad de la celda DDW0204SCDG.

Para lograr lo anteriormente mencionado se tomó como base la tabla de verdad de la Figura 30, esta tabla proporcionó las configuraciones de bits de input para lograr distintas configuraciones de output. Por lo tanto, se seleccionaron las configuraciones de bits donde PAD es un input y C es output y ambos tienen el mismo valor.

En el archivo tpd018nv.v están las descripciones de los pads en un archivo de verilog, este nos proporciona el orden en el que están los parámetros para configurar el pad, como se observa en la Figura 31.

```
`celldefine
module PDDW0204CDG (I,DS,OEN,PAD,C,PE,IE);
```

Figura 31: Descripción en verilog de la celda DDW0204SCDG.

Para el anillo de entradas y salidas es posible que las celdas físicas para la colocación de alimentación, tierra y corners no formen parte del netlist sintetizado lógicamente y se deben agregar al diseño en IC Compiler(síntesis física).Para esto, se utilizó el comando *create-cell* de la forma como se indica en la Figura 32.

```
create_cell {c1 c2 c3 c4} PCORNER
create_cell {VDDPAD} PVDD1CDG
create_cell {VSSPAD} PVSS1CDG
```

Figura 32: Creación de pads de VDD, VSS y corners.

En la Figura 32, el comando create-cell contiene los nombres de cada celda entre llaves, seguido de la referencia del elemento en la librería de TSMC. A estos pads se les colocó restricciones como se muestra en la Figura 33.

```
set_pad_physical_constraints -pad_name "c1" -side 1
set_pad_physical_constraints -pad_name "c2" -side 2
set_pad_physical_constraints -pad_name "c3" -side 3
set_pad_physical_constraints -pad_name "c4" -side 4
set_pad_physical_constraints -pad_name "VDD" -side 2 -order 2
set_pad_physical_constraints -pad_name "VSS" -side 4 -order 2
```

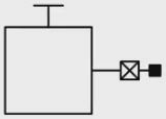
Figura 33: Creación de las restricciones de pads de VDD, VSS y corners.

Con ayuda del comando set-pad-physical-constraints,se colocó el nombre del pad con el parámetro pad-name, con el parámetro side especifíca en que borde del floorplan rectangular irá el pad (donde 0 es la restricción default, 0 es borde izquierdo, 1 es borde superior, 3 es borde derecho y 4 es borde inferior), en este caso cada corner irá en un borde distinto, el pad de VDD irá en el borde superior y el pad de VSS irá en el borde inferior, el parámetro order especifica el número de orden de colocación del pad. El orden de ubicación es una restricción de orden en el sentido de las agujas del reloj para los lados izquierdo y superior. El orden de colocación es una restricción de orden en sentido antihorario para los lados derecho e inferior. El argumento de orden debe ser un número entero positivo.El valor predeterminado es 0, lo que significa que el pad no tiene una restricción de orden.

También es posible crear los *pads* de alimentación y tierra desde la síntesis lógica, colocando los *pads* PVDD1CDG y PVSS1CDG respectivamente.

8.35 PVDD1CDG

Vdd Pad for Core Power Supply



Cell Information

	Value	Unit
Pad Number	1	-

Leakage Power

	Value	Unit
VDD	45.2453	nW

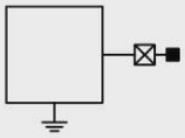
Pin Capacitance

	Value	Unit
VDD	3.1734	pF

Figura 34: Pad de VDD o alimentación para crearlo desde la síntesis lógica.

8.40 PVSS1CDG

Vss Pad for Core Ground Supply



Cell Information

	Value	Unit
Pad Number	1	-

Leakage Power

	Value	Unit
VDD	0.0000	nW
VDDPST	0.0980	nW

Pin Capacitance

	Value	Unit
VSS	1.7012	pF

Figura 35: Pad de VSS o tierra para crearlo desde la síntesis lógica.

CAPÍTULO 10

Prueba de antena

10.1 Detección y reparación de violaciones de antena

En la fabricación de circuitos integrados, el óxido del gate se puede dañar fácilmente por una descarga electrostática. La carga estática que se acumula en los cables durante el proceso de metalización multinivel puede dañar el dispositivo. IC compiler define el área de antena metálica como el área del pin de metal sumado con el área de encierro de metal. El siguiente ejemplo utiliza IC Compiler para el cálculo del área de la antena:

Gate area = 0.01um,

Metal pin area = 0.05um,

==> If metal enclosure of via = 0.1um (area = 0.01um),

==> antenna ratio = (metal pin area + metal enclosure area) / gate area

==> antenna ratio = (0.05+(0.01*2)) / 0.01 = 7

IC Compiler incluye el área metálica de la caja de vía para el cálculo del área metálica de la antena. El enrutador en IC Compiler duplica el recuento de áreas metálicas si la caja metálica de via se superpone con un alfiler de metal o una forma de metal. Según el cálculo anterior, IC Compiler no considera la condición de fusión de cada capa de metal. Para evitar problemas de antena, la herramienta IC Compiler verifica que para cada pin de entrada el área de antena metálica dividida por el área del gate sea menor que la relación máxima de antena dada por la fundición:

$$(antenna - area)/(gate - area) < (max - antenna - ratio) \quad (1)$$

Figura 36: Flujo de antena.

Para poder llevar a cabo el flujo de antena,se empezó definiendo las reglas (Fig.36). Ya que el circuito integrado trabajado en esta línea de investigación será enviado a fabricación depende de cada fabricante que tan rigurosas son las reglas, es por esto que se le solicitó a la empresa TSMC el runset de antena.

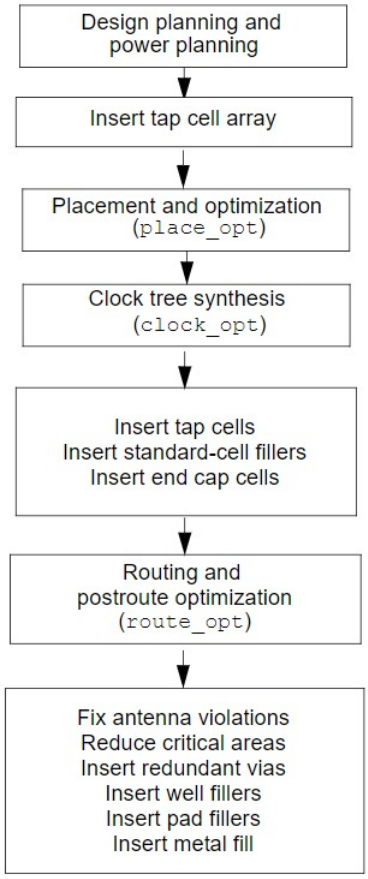


Figura 37: Flujo de síntesis física.

Se estudió el manual de IC Compiler, se obtuvo que la prueba de antena debe llevarse a cabo después de la optimización del ruteo, como se muestra en la Figura 37.

10.2 Archivos necesarios para ejecutar la prueba de antena

Un runset es un script que contiene los comandos/reglas de determinado proceso de cada fabricante (existe un runset para DRC,LVS,etc).

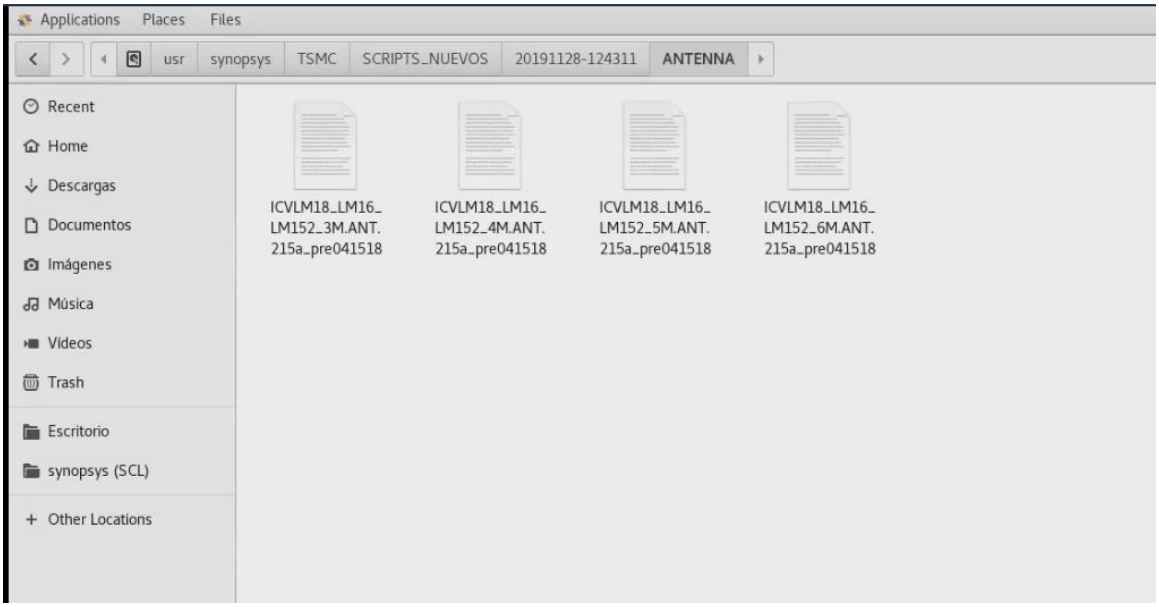


Figura 38: Runsets de antena.

Se encontró el runset proveído por TSMC, el cual se configuró para un grosor de 40K. Además de este runset, TSCM envió un script de extensión .tcl, el cual contiene las reglas de antenna para la síntesis física, este script funciona para la configuración de la relación de antena que puede acelerar la resolución de la violación de antena. Tanto el script .tcl como el runset deben ejecutarse para esta prueba.

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

//#define MIMCAP_2F                     /* turn on only when customer use 2fF/um2 MIMCAP. otherwise please turn off it */
#define THICK_40K                     /* turn on only when 40KA 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 */
M1_THICKNESS : double = 0.53;        /* The Thickness of First Metal is 5.3K A */
M2_THICKNESS : double = 0.53;        /* The Thickness of Inter Metal is 5.3K A */
M3_THICKNESS : double = 0.53;        /* The Thickness of Inter Metal is 5.3K A */
M4_THICKNESS : double = 0.53;        /* The Thickness of Inter Metal is 5.3K A */
M5_THICKNESS : double = 0.53;        /* The Thickness of Inter Metal is 5.3K A */
#ifdef THICK_40K
   M6_THICKNESS : double = 4.6;      /* The Thickness of Top Metal is 40K A */
#else
   #ifdef THICK_20K
      M6_THICKNESS : double = 2.34;    /* The Thickness of Top Metal is 20K A */
   #else
      M6_THICKNESS : double = 0.99;    /* The Thickness of Top Metal is 9.9K A */
   #endif
#endif
MD_THICKNESS : double = 0.99;
PO_THICKNESS : double = 0.2;
MX_S_1 : double = 0.28;
BALANCE_RATIO : double = 6000;
UNBALANCE_RATIO : double = 100;
DIO_AREA : double = 0.203;
vDNW_R_7 : double = 500000;
MT_THICKNESS : double = 0.99;

library(
    cell         = "RCA_IO_CORRUPT",
    format       = GDSII,
    library_name = "/home/administrador/Escritorio/LVS/RCA.gds"
```

Figura 39: Código a cambiar en runset

En la Figura 39, se observa parte del código del runset, este se editó para leer el archivo gds deseado poniendo el nombre de la *Top Cell* y la dirección completa del archivo gds que se está trabajando, también puede ejecutarse con la librería *Milkyway*, colocando el formato MILKYWAY. Como se mencionó anteriormente, este se configura a un grosor de 40k.

Se encontró que el runset debe ejecutarse en IC validator al finalizar la síntesis física y el archivo tcl debe ejecutarse en IC Compiler, después de optimizar el ruteo.

TSMC envió el runset para circuitos con 3,4, 5 o 6 capas de metal, se utilizó el runset de 6 capas de metal ya que las librerías que usamos son de 6 metales, es decir, de este numero de capas será el circuito integrado. TSMC envió dos formatos, .tcl para IC Compiler y .scm para la herramiento Astro Router, por lo tanto se decidió utilizar el runset de Antena para 6 capas de metal en extensión tcl.

Figura 40: Archivos con extensión tcl de antena.

Este archivo está compuesto de dos tipos de reglas: Regla del área de la pared lateral de una sola capa de metal y Regla de área de pared lateral de una sola vía.

Para correr el runset de antena, es necesario obtener el archivo extensión gds del circuito que se está trabajando. Existen dos formas de obtener este archivo: Desde IC Compiler y desde Custom Compiler.

10.2.1 Obtención de archivo gds desde Custom Compiler

Se puede obtener el archivo gds de dos formas distintas: desde IC Compiler y desde Custom Compiler. Para obtenerlo desde Custom Compiler es necesario haber hecho la síntesis física (librería *milkyway* creada). Primero es necesario dirigirse a File -> Add ICC Library, donde se debe colocar el nombre y el directorio de la librería *milkyway* creada, como se observa en la Figura 40.

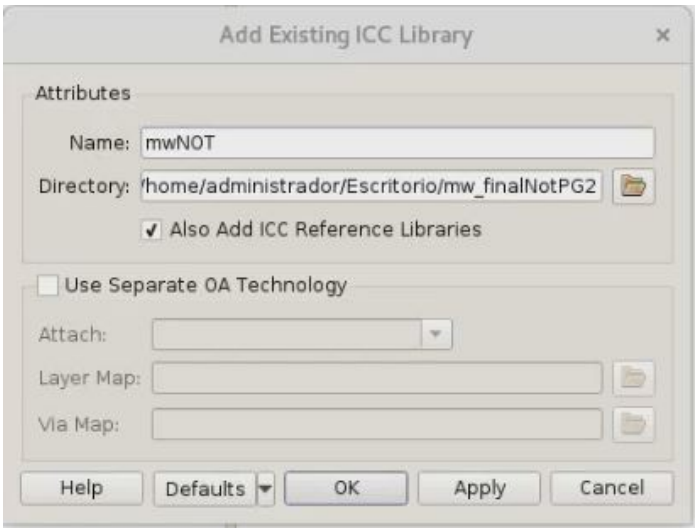


Figura 41: agregar librería milkyway.

Posteriormente se debe dirigir a File -> Import -> From ICC, donde se debe colocar la librería *milkyway* trabajada y el nombre de la *Top Cell*, como se muestra en la Figura 41.

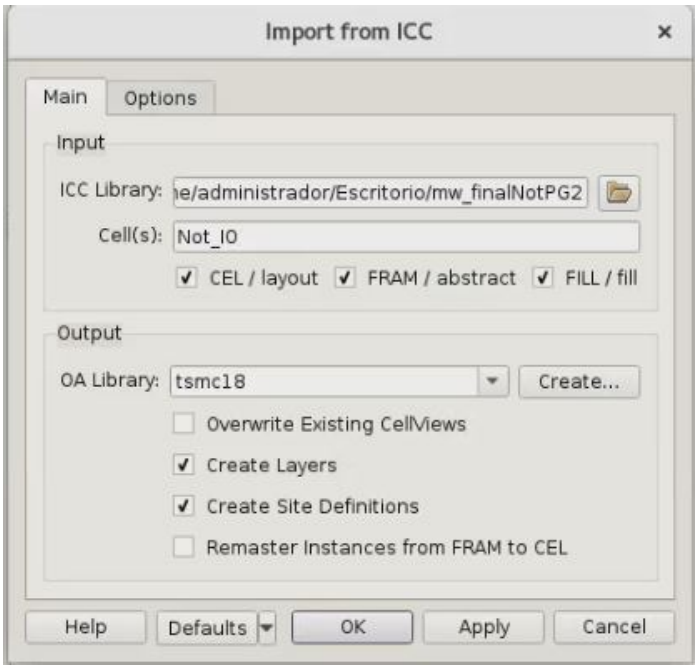


Figura 42: importar librería y crear el layer.

Por último, se debe dirigir a File -> Export -> Stream, donde se debe colocar la librería *milkyway*, el nombre de la *Top Cell* y la vista que por defecto es Layout, como se observa en la Figura 42.

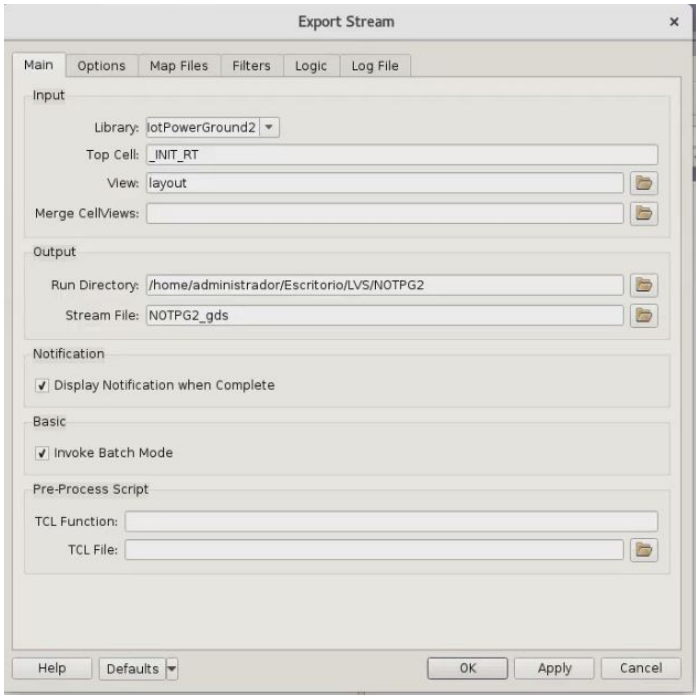


Figura 43: Exportación de stream.

Los detalles de este procedimiento se describen en la tesis del módulo de LVS.

10.2.2 Comandos para las pruebas de antena en IC Compiler

define-antenna-rule:

Define una regla avanzada para un modo de verificación de antena específico y almacena la regla en una librería Milkyway.

define-antenna-layer-rule:

Define una regla de antena avanzada para una capa específica y almacena la regla en una librería Milkyway.

remove-antenna-rules:

Elimina las reglas de antena almacenadas en una librería milkiway.

10.3 Reglas que verifica el archivo tcl de antena de TSMC

10.3.1 Regla del área de la pared lateral de una sola capa de metal

Se pueden definir reglas de antena de capa metálica tanto globales como específicas de la capa. Las reglas específicas de la capa, cuando están presentes, anulan las reglas globales; las reglas globales se aplican a las capas que no tienen sus propias reglas específicas. Las reglas globales se definen con el comando define-antenna-rule y las reglas específicas de la capa con el comando define-antenna-layer-rule.

La regla global de este tipo utiliza el área de la pared lateral, ignorando los segmentos de la capa inferior (modo de una sola capa). La protección de diodos es limitada; si se conecta más de un diodo, se usa la suma de todos los valores de protección de diodos para todos los diodos para calcular la relación máxima de antena. TSMC especifica la relación máxima permitida entre el área de la antena metálica y el área del gate de los transistores, la cual es 400.

Posteriormente TSMC define una regla de capa por cada capa de metal del circuito integrado, en la cual se especifica en que capa se aplica la regla, el modo, que en este caso utiliza el área de la pared lateral, ignorando los segmentos de la capa inferior (modo de una sola capa). También se especifica la relación máxima permitida entre el área de la antena metálica y el área del gate si la antena no está protegida por ningún diodo, que en este caso es de 400. Además, especifica la relación máxima permitida entre el área de la antena metálica y el área del gate si la antena está protegida por un diodo, expresada como un vector de valores en forma de v0 v1 v2 v3 [v4], El valor v4 representa el límite superior de la protección del diodo y es opcional. Si no especifica el valor v4, se supone que es 0, lo que significa que no hay limite superior.

```
define_antenna_rule /home/administrador/Escritorio/mw_RCA \
  -mode 4 \
  -diode_mode 4 \
  -metal_ratio 400 \
  -cut_ratio 0

define_antenna_layer_rule /home/administrador/Escritorio/mw_RCA \
            -mode 4 \
            -layer "METAL6" \
            -ratio 400 \
            -diode_ratio {0.203 0 8000 30000}

define_antenna_layer_rule /home/administrador/Escritorio/mw_RCA \
            -mode 4 \
            -layer "METAL5" \
            -ratio 400 \
            -diode_ratio {0.203 0 400 2200}

define_antenna_layer_rule /home/administrador/Escritorio/mw_RCA \
            -mode 4 \
            -layer "METAL4" \
            -ratio 400 \
            -diode_ratio {0.203 0 400 2200}

define_antenna_layer_rule /home/administrador/Escritorio/mw_RCA \
            -mode 4 \
            -layer "METAL3" \
            -ratio 400 \
            -diode_ratio {0.203 0 400 2200}

define_antenna_layer_rule /home/administrador/Escritorio/mw_RCA \
            -mode 4 \
            -layer "METAL2" \
            -ratio 400 \
            -diode_ratio {0.203 0 400 2200}

define_antenna_layer_rule /home/administrador/Escritorio/mw_RCA \
            -mode 4 \
            -layer "METAL1" \
            -ratio 400 \
            -diode_ratio {0.203 0 400 2200}
```

Figura 44: Reglas del área de la pared lateral de una sola capa de metal

10.3.2 Regla del área de pared lateral de una sola vía

Esta regla utiliza el área del polígono, ignorando los segmentos de la capa inferior (modo de una sola capa).La protección de diodos es limitada; si se conecta más de un diodo, se usa la suma de todos los valores de protección de diodos para todos los diodos para calcular la relación máxima de antena. Evita la verificación de la relación entre el área de metal y el área del *gate*.

```
define_antenna_rule /home/administrador/Escritorio/mw_RCA \
  -mode 1 \
  -diode_mode 4 \
  -metal_ratio 0 \
  -cut_ratio 20

define_antenna_layer_rule /home/administrador/Escritorio/mw_RCA \
            -mode 1 \
            -layer "VIA12" \
            -ratio 20 \
            -diode_ratio {0.203 0 83.33 75}

define_antenna_layer_rule /home/administrador/Escritorio/mw_RCA \
            -mode 1 \
            -layer "VIA23" \
            -ratio 20 \
            -diode_ratio {0.203 0 83.33 75}

define_antenna_layer_rule /home/administrador/Escritorio/mw_RCA \
            -mode 1 \
            -layer "VIA34" \
            -ratio 20 \
            -diode_ratio {0.203 0 83.33 75}

define_antenna_layer_rule /home/administrador/Escritorio/mw_RCA \
            -mode 1 \
            -layer "VIA45" \
            -ratio 20 \
            -diode_ratio {0.203 0 83.33 75}

define_antenna_layer_rule /home/administrador/Escritorio/mw_RCA \
            -mode 1 \
            -layer "VIA56" \
            -ratio 20 \
            -diode_ratio {0.203 0 83.33 75}
```

Figura 45: Reglas del área de pared lateral de una sola vía

10.4 Archivo tcl

Como se mencionó anteriormente, TSMC envió un archivo con extensión tcl para verificar las reglas de antena en la síntesis física, este archivo debe leerse después de la optimización de ruteo. Para saber como leerlo se estudió el manual *Using Tcl with synopsis tools*, en el que se encontró que este tipo de archivo soporta los siguientes comandos:

after	exec	history*	open	split
append	expr	if	package	string
array	exit*	incr	pid	subst
binary	fblocked	info	proc	switch
bgerror	fconfigure	interp	puts	tell
break	fcopy	join	pwd	time
catch	file	lappend	read	trace
cd	fileevent	lindex	regexp	unset
clock	filename	linsert	regsub	update
close	flush	list	rename*	uplevel
concat	for	llength	return	upvar
continue	foreach	lrange	scan	variable
encoding	format	lreplace	seek	vwait
eof	gets	lsearch	set	while
error	glob	lsort	socket	
eval	global	namespace	source*	

Figura 46: Comandos de IC Compiler para archivos Tcl.

De la Figura 46, se estudiaron los comandos y se encontraron dos relevantes para su funcionamiento: uplevel y source.

Comando source

Este comando lee un archivo y lo evalúa como un script Tcl.

Comando uplevel

Un procedimiento puede acceder a variables dentro del alcance global o desde el alcance de otro procedimiento llamándolo usando los comandos upvar y uplevel de IC Compiler. El comando uplevel evalúa el comando en un ámbito diferente en lugar del ámbito actual, la sintaxis 0 indica el alcance global.

verify-zrt-route

Adicionalmente, se debe agregar el comando verify-zrt-route, el cual verifica e informa sobre violaciones de restricciones de reglas de diseño de enrutamiento (DRC), aperturas de red, violaciones de reglas de antena y violaciones de reglas de área de voltaje.

Contenido del archivo

El archivo tcl contiene 13 reglas de antena descritas con los comandos anteriormente mencionados, de este archivo solo debe modificarse la dirección de la librería *milkyway* que se está trabajando, ya que todas las configuraciones son las que TSMC envío para sus reglas.

10.5 Comandos para ejecutar runset y archivo tcl de antena

```
1 uplevel #0 source DireccionDeArchivoTcl
2 verify_zrt_route
```

Se deben utilizar los comandos anteriores para correr el archivo tcl de antena en la herramienta IC Compiler y mostrar sus resultados (la dirección varía depende donde se guarde el archivo tcl). El comando uplevel 0 no es estrictamente necesario, ya que este evalúa las variables implicadas en las reglas de antenna en un ámbito global pero en este caso todo está bajo un mismo ámbito.

El siguiente es el comando para correr el runset de antena en una terminal:

```
1 icv -i DireccionCompletaDeArchivogds -c NombreCelda -sf ICV -vue
    DireccionCompletaRunsetAntena
```

Cabe mencionar que el runset también puede ejecutarse desde la interfaz gráfica de IC Validator.

10.6 Archivos que se generan al ejecutar runset de antena

Figura 47: Archivos que se generan al ejecutar runset de antena.

Se debe crear una carpeta por cada circuito que se trabaje, esto para manejar de la manera más ordenada posible los archivos de antena, los archivos importantes son los archivos .RESULTS y .LAYOUT-ERRORS.

```
--------------------------------------------------------------------------------
Results Summary
--------------------------------------------------------------------------------

Rule and DRC Error Summary

44 total rules were run.
0 rules NOT EXECUTED.
0 rules have violations.
There are 0 total violations.
Refer to RCA_IO_CORRUPT.LAYOUT_ERRORS
```

Figura 48: Archivo de resultados al correr runset de antena a un RCA.

En la Figura 48 vemos los resultados de la prueba de antena al correrla en un *Ripple Carry Adder*, se ejecutaron 44 reglas y no se obtuvieron violaciones. Para más información se debe revisar el arhivo de errores de *Layout*.

Figura 49: Errores de layout del RCA al correr runset de antena.

En la Figura 49 se observan los errores de **layout**, en este caso la prueba ha sido exitosa (puede verse **clean**).

10.7 Prueba de antena con una compuerta not con comando

Se realizó la prueba de antena para la compuerta not, con el fin de minimizar todos los errores posibles, primero se corrió la prueba del archivo tcl llamada antennaRule-018-6lm.tcl en IC Compiler, obteniendo los resultados de las figuras 50 y 51.

```
set lib [current_mw_lib]
remove_antenna_rules $lib
define_antenna_rule $lib -mode 1 -diode_mode 4 -metal_ratio 0 -cut_ratio 20
define_antenna_layer_rule $lib -mode 1 -layer "VIA12" -ratio 20 -diode_ratio {0.203 0 83.33 75}
define_antenna_layer_rule $lib -mode 1 -layer "VIA23" -ratio 20 -diode_ratio {0.203 0 83.33 75}
define_antenna_layer_rule $lib -mode 1 -layer "VIA34" -ratio 20 -diode_ratio {0.203 0 83.33 75}
define_antenna_layer_rule $lib -mode 1 -layer "VIA45" -ratio 20 -diode_ratio {0.203 0 83.33 75}
define_antenna_layer_rule $lib -mode 1 -layer "VIA56" -ratio 20 -diode_ratio {0.203 0 83.33 75}

define_antenna_rule $lib -mode 4 -diode_mode 4 -metal_ratio 400 -cut_ratio 0
define_antenna_layer_rule $lib -mode 4 -layer "METAL6" -ratio 400 -diode_ratio {0.203 0 8000 30000}
define_antenna_layer_rule $lib -mode 4 -layer "METAL5" -ratio 400 -diode_ratio {0.203 0 400 2200}
define_antenna_layer_rule $lib -mode 4 -layer "METAL4" -ratio 400 -diode_ratio {0.203 0 400 2200}
define_antenna_layer_rule $lib -mode 4 -layer "METAL3" -ratio 400 -diode_ratio {0.203 0 400 2200}
define_antenna_layer_rule $lib -mode 4 -layer "METAL2" -ratio 400 -diode_ratio {0.203 0 400 2200}
define_antenna_layer_rule $lib -mode 4 -layer "METAL1" -ratio 400 -diode_ratio {0.203 0 400 2200}
```

Figura 50: Reporte de las reglas de antena que se están ejecutando.

```
Verify Summary:

Total number of nets = 8, of which 0 are not extracted
Total number of open nets = 0, of which 0 are frozen
Total number of excluded ports = 0 ports of 0 unplaced cells connected to 0 nets
                                 0 ports without pins of 0 cells connected to 0 nets
                                 0 ports of 0 cover cells connected to 0 non-pg nets
Total number of DRCs = 0
Total number of antenna violations = 0
Total number of voltage-area violations = no voltage-areas defined
Total number of tie to rail violations = not checked
Total number of tie to rail directly violations = not checked
```

Figura 51: Resultados de la prueba.

En la Figura 52 se muestra el resultado del archivo .RESULTS que se genera al correr el runset de antena, siendo exitosos los resultados ya que no se muestran violaciones.

```
--------------------------------------------------------------------------------
Results Summary
--------------------------------------------------------------------------------

Rule and DRC Error Summary

44 total rules were run.
0 rules NOT EXECUTED.
0 rules have violations.
There are 0 total violations.
Refer to Not_IO.LAYOUT_ERRORS
```

Figura 52: Resultados del runset con una Not.

Como se observa en la Figura 52, para la compuerta not no se tienen violaciones de antena.

```
                LAYOUT ERRORS RESULTS: CLEAN

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

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

Library name:     /home/administrador/Escritorio/LVS/NOTPG2/NOTPG2_gds
Structure name:   _INIT_RT
Generated by:     IC Validator RHEL64 Q-2019.12-SP2-4.5500898 2020/04/25
Runset name:      /usr/synopsys/TSMC/SCRIPTS_NUEVOS/20191128-124311/ANTENNA/ICVLM18_LM16_LM152_6M.ANT.215a_pre041518
User name:        administrador
Time started:     2020/09/19 11:57:26PM
Time ended:       2020/09/19 11:57:28PM

Called as: icv -i /home/administrador/Escritorio/LVS/NOTPG2/NOTPG2_gds -c _INIT_RT -sf ICV -vue /usr/synopsys/TSMC/SCRIPTS_NUEVOS/20191128-124311/ANTENNA/ICVLM18_LM16_LM152_6M.ANT.215a_pre041518
```

Figura 53: Resultados de la prueba de antena.

Al correr el runset de antena tomando como archivo de entrada el archivo con extensión gds exportado desde custom compiler de la compuerta Not, se obtuvo el resultado de la Figura 53 indicando que pasó la prueba.

10.8 Prueba de antena con una compuerta NOT con VUE tool

El runset de antena también puede ejecutarse desde la interfaz gráfica de IC Validator *(VUE tool)*, esta herramienta permite realizar las mismas funciones que el comando.

En la interfaz debe especificarse el archivo de Runset a utilizar, el directorio donde se guardarán todos los archivos de salida y el *Layout* que se está utilizando, este puede ingresarse en formato GDSII o Milkyway.

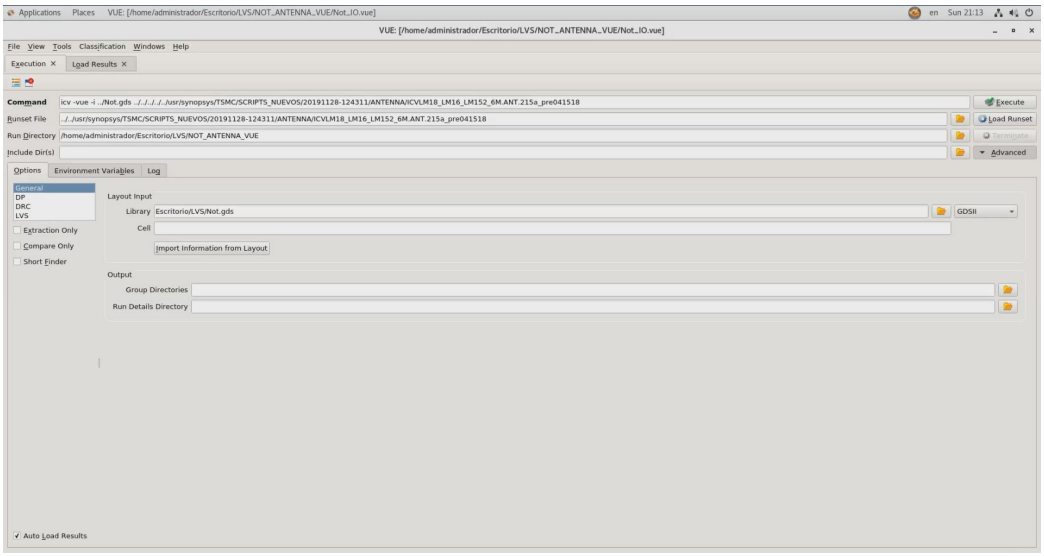


Figura 54: Configuración en VUE tool para prueba de antena con compuerta Not

Esta prueba dio resultado equivalente al hecho con comando, siendo ambas corridas sin errores.

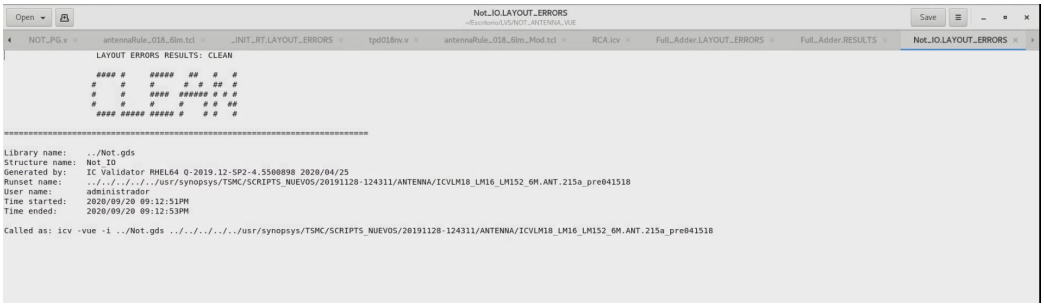


Figura 55: Resultados de la prueba de antena en VUE tool para una compuerta not.

10.9 Prueba de antena con el circuito Full Adder con comando

Teniendo la seguridad de que la prueba funciona para la compuerta Not, se corrió las pruebas para un circuito más complejo, en este caso un Full Adder. Para el Full Adder se corrió el archivo tcl en IC Compiler dando como resultado lo observado en la Figura 56. Como se puede observar, no hay violaciones de antena para este circuito.

```
Verify Summary:

Total number of nets = 15, of which 0 are not extracted
Total number of open nets = 0, of which 0 are frozen
Total number of excluded ports = 0 ports of 0 unplaced cells connected to 0 nets
                                 0 ports without pins of 0 cells connected to 0 nets
                                 0 ports of 0 cover cells connected to 0 non-pg nets
Total number of DRCs = 0
Total number of antenna violations = 0
Total number of voltage-area violations = no voltage-areas defined
Total number of tie to rail violations = not checked
Total number of tie to rail directly violations = not checked

1
icc_shell>
```

Figura 56: Corrida de archivo tcl sin violaciones de antena para un Full Adder.

Después, se ejecutó el runset de antena por medio del comando en terminal dando como resultado lo observado en las figuras 57 y 58. Se puede observar que el circuito Full Adder pasó la prueba de antena.

```
---------------------------------------------------------------------------
Results Summary
---------------------------------------------------------------------------

Rule and DRC Error Summary

44 total rules were run.
0 rules NOT EXECUTED.
0 rules have violations.
There are 0 total violations.
Refer to Full_Adder.LAYOUT_ERRORS
```

Figura 57: Resultados de la prueba para un Full Adder

```
                LAYOUT ERRORS RESULTS: CLEAN

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

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

Library name:    /mnt/nfs/compartida/FullAdderFiles/GDS/FullAdder.gds
Structure name:  Full_Adder
Generated by:    IC Validator RHEL64 Q-2019.12-SP2-4.5500898 2020/04/25
Runset name:     /usr/synopsys/TSMC/SCRIPTS_NUEVOS/20191128-124311/ANTENNA/ICVLM18_LM16_LM152_6M.ANT.215a_pre041518
User name:       administrador
Time started:    2020/09/16 01:40:55PM
Time ended:      2020/09/16 01:40:58PM

Called as: icv -i /mnt/nfs/compartida/FullAdderFiles/GDS/FullAdder.gds -c Full_Adder -sf ICV -vue /usr/synopsys/TSMC/SCRIPTS_NUEVOS/20191128-124311/ANTENNA/ICVLM18_LM16_LM152_6M.ANT.215a_pre041518
```

Figura 58: Corrida de runset sin errores para un Full Adder.

10.10 Prueba de antena con el circuito Full Adder con VUE tool

Se configuraron los parámetros descritos anteriormente en la interfaz VUE tool: de IC Validator, como se observa en la Figura 59.

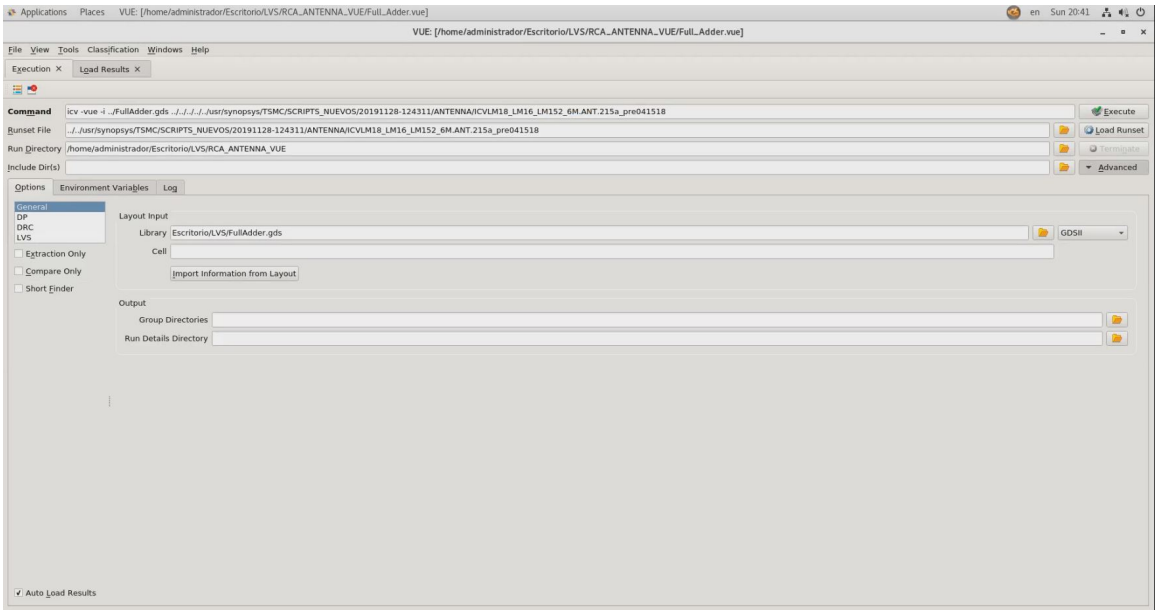


Figura 59: Configuración en VUE tool para un Full Adder.

Al igual que en el caso anterior, se recomienda crear un directorio para tener los archivos de salida organizados.

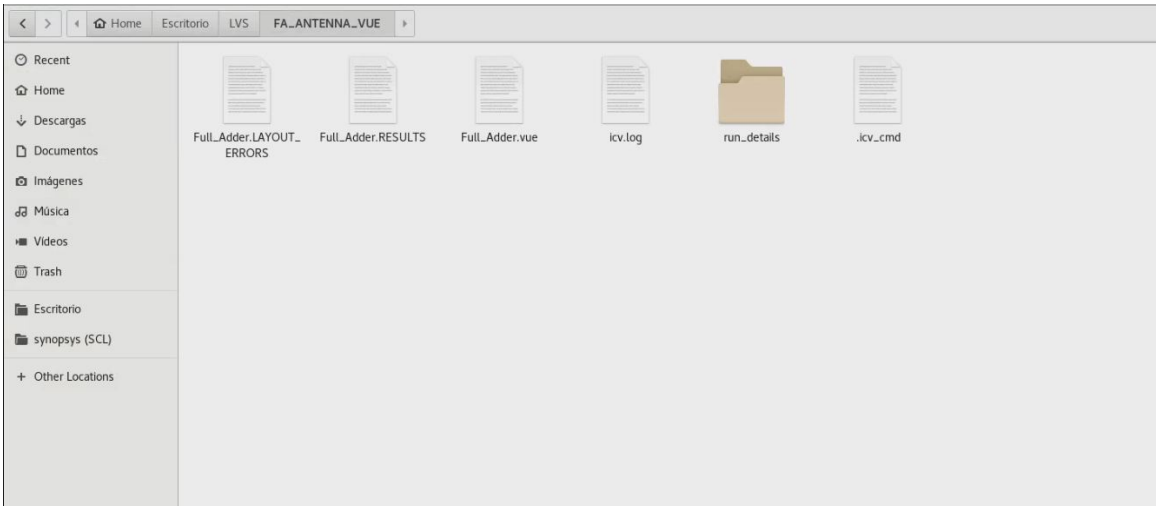


Figura 60: Directorio para guardar archivos de salida de la prueba de antena.

Esta prueba fue exitosa para un Full Adder, como se muestra en la Figura 61.

```
                 LAYOUT ERRORS RESULTS: CLEAN

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

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

Library name:    /home/administrador/Escritorio/LVS/FullAdder.gds
Structure name:  Full_Adder
Generated by:    IC Validator RHEL64 Q-2019.12-SP2-4.5500898 2020/04/25
Runset name:     ../../../../../usr/synopsys/TSMC/SCRIPTS_NUEVOS/20191128-124311/ANTENNA/ICVLM18_LM16_LM152_6M.ANT.215a_pre041518
User name:       administrador
Time started:    2020/09/20 08:12:21PM
Time ended:      2020/09/20 08:12:23PM

Called as: icv -vue ../../../../../usr/synopsys/TSMC/SCRIPTS_NUEVOS/20191128-124311/ANTENNA/ICVLM18_LM16_LM152_6M.ANT.215a_pre041518
```

Figura 61: Corrida de runset sin errores para un Full Adder.

10.11 Prueba de antena con el circuito Ripple Carry Adder con comando

Se ejecutó la prueba de antena para un RCA. Al igual que en las dos pruebas anteriores, primero se ejecutó la prueba mediante comandos de IC Compiler durante la síntesis física dando como resultado una prueba sin violaciones, como se puede observar en la Figura 62.

```
Verify Summary:

Total number of nets = 37, of which 0 are not extracted
Total number of open nets = 1, of which 0 are frozen
Total number of excluded ports = 80 ports of 16 unplaced cells connected to 24 nets
                                 0 ports without pins of 0 cells connected to 0 nets
                                 0 ports of 0 cover cells connected to 0 non-pg nets
Total number of DRCs = 498
Total number of antenna violations = 0
Total number of voltage-area violations = no voltage-areas defined
Total number of tie to rail violations = not checked
Total number of tie to rail directly violations = not checked

1
icc_shell>
```

Figura 62: Corrida de archivo tcl en IC compiler para un RCA

Posteriormente se ejecutó el runset de antena proveído por TSMC al circuito RCA, teniando un total de cero violaciones y una prueba limpia, como se observan en las figuras 63 y 64.

```
--------------------------------------------------------------------------------
Results Summary
--------------------------------------------------------------------------------

Rule and DRC Error Summary

44 total rules were run.
0 rules NOT EXECUTED.
0 rules have violations.
There are 0 total violations.
Refer to RCA_IO_CORRUPT.LAYOUT_ERRORS
```

Figura 63: Corrida de archivo tcl en IC compiler para un RCA

Figura 64: Corrida de archivo tcl en IC compiler para un RCA

10.12 Prueba de antena con el circuito Ripple Carry Adder con VUE tool

Se ejecutó la prueba para un *Ripple Carry Adder*, siendo exitosa la prueba para este circuito.

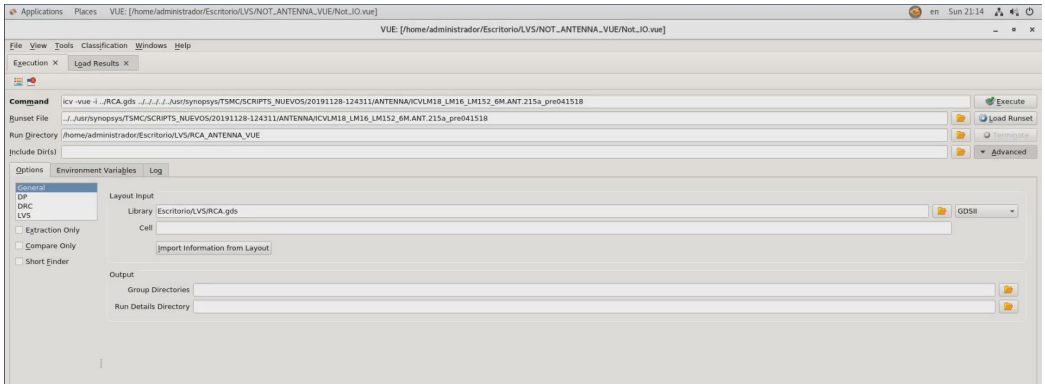


Figura 65: Configuración en VUE tool para un RCA.

Figura 66: Resultados de la prueba de antena para un RCA mediante VUE tool

CAPÍTULO 11

Reglas de Diseño Eléctrico (ERC)

Como se mencionó en el marco teórico, *ERC* es una opción que tiene capacidad de verificar siete clases de reglas: *Soft Connect Check, Path Check, PTAP/NTAP Check, MOSFET power and ground check, Gate directly connected to power or ground, floating Gate y Floating Well*. Para la descripción de cada regla referirse al Marco Teórico de este trabajo. El runset proveído por TSMC verifica estas siete reglas. El Runset de ERC está incorporado en el mismo script junto con el runset de LVS:. Las pruebas realizadas de ERC solo se realizaron con el runset de TSMC ya que Synopsis no cuenta con este runset.

11.1 Comando para ejecutar el runset desde la terminal.

El comando para ejecutar el runset de ERC, el cual es el mismo que el de LVS, es el siguiente:

```
1 icv -i DireccionArchivogds -c NombreCelda -s DireccionArchivoicv -sf
    ICV -vue DireccionRunset
```

Los archivos gds e icv se describen en la sección 11.2.1 y 11.2.2, respectivamente.

11.2 Archivos necesarios para la verificación de las reglas de diseño eléctrico

En comparación con la prueba de antena, esta prueba de integridad necesita un archivo más para su funcionamiento. Los archivos necesarios para ejecutar esta prueba son:

- Nombre de la librería (Archivo gds)

- Archivo esquemático (Archivo icv)

11.2.1 Archivo gds

Como se mencionó anteriormente, este archivo puede generarse a través de IC Compiler o desde Custom Compiler, este archivo es necesario para la prueba porque contiene toda la información sintetizada del circuito trabajado.

11.2.2 Archivo icv

Este archivo es un archivo esquemático. Este archivo contiene la información del circuito a nivel de celdas. Para generar este *netlist* a nivel esquemático se tomó como guía el documento FS-AN-006-How-To-Generate-CDL-Netlist-From-Verilog enviado por IMEC. Los detalles de la creación de este archivo se encuentran en la sección 10.2 de la tesis del módulo de LVS de esta línea de investigación.

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

library(
        library_name = "/home/administrador/Escritorio/LVS/NOTPG2/NOTPG2_gds",
        cell = "_INIT_RT",
        format = GDSII
);

SCHEMATIC_TOPCELL : string = "Not_IO";          // Set schematic top cell name here
sch_db = schematic(
        schematic_file = {{"/mnt/nfs/compartida/allicv", ICV}},
```

Figura 67: Configuración de directorios de archivos necesarios

En la Figura 67 se muestra como debe estar configurado el runset.

11.3 Configuración del runset de LVS para ejecutar ERC.

Para ejecutar las funciones de ERC es necesario apagar la función de extracción de parásitos, ya que el *runset* crea las conexiones necesarias para ERC si esta función no está definida ya que solo puede ejecutar LVS con extracción o LVS con ERC (no ambas opciones a la vez).

```
#ifndef RC_DECK
/* CONNECTIONS FOR ERC */
cdb = incremental_connect(
        cdb,
        {
                {{ntap}, ntap_not_var},
                {{gate1_not_IO1}, poly}
        }
);
#endif
```

Figura 68: Conexiones necesarias para ERC apagando la función de extracción

Para verificar todas las reglas de ERC, se deben habilitar todas las reglas y deshabilitar la extraccióon, como se muestra en la Figura 69.

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

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

Figura 69: Configuración de opciones para ejecutar ERC

Regla	Configuración por defecto	Configuración aplicada
WELL TO PG CHECK	Habilitado	Habilitado
GATE TO PG CHECK	Deshabilitado	Habilitado
PATH CHECK	Deshabilitado	Habilitado
DS TO PG CHECK	Habilitado	Habilitado
FLOATING GATE CHECK	Habilitado	Habilitado
FLOATING WELL CHECK	Habilitado	Habilitado
NW RING	Deshabilitado	Deshabilitado

Cuadro 1: Configuraciones por defecto y aplicadas a las reglas de diseño eléctrico (ERC).

Como se observa en la Figura 69 y en el Cuadro 1, algunas funciones del ERC estaban deshabilitadas, para correr ERC de forma completa es necesario habilitar todas las reglas, exceptuando *NW RING*, ya que si se habilita esta función no se realizará el *Soft connect check* para el *nwell* y el sustrato tipo p.

11.4 Archivos generados al correr ERC

Similar a la prueba de Antena, ERC genera archivos de resultados siendo los más importantes los de extensión **.RESULTS** y **.LAYOUT-ERRORS**.

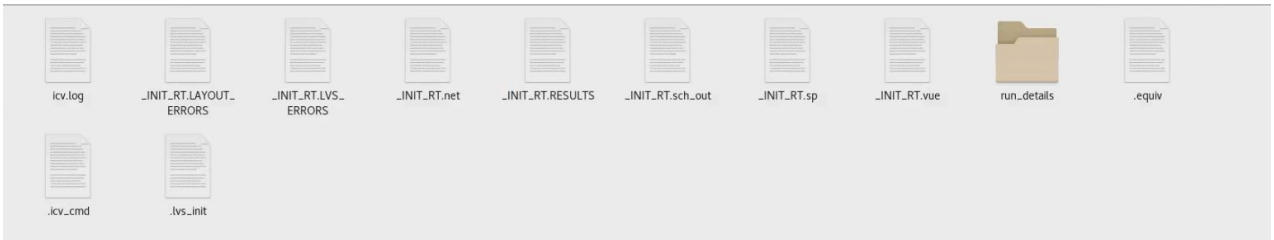


Figura 70: Archivos generados al correr runset de LVS con ERC.

En el archivo de extensión **.RESULTS** se deben obtener resultados satisfactorios, tanto en la prueba de LVS como en la de DRC y extracción (que en este caso son los resultados de ERC), como se observa en la Figura 71.

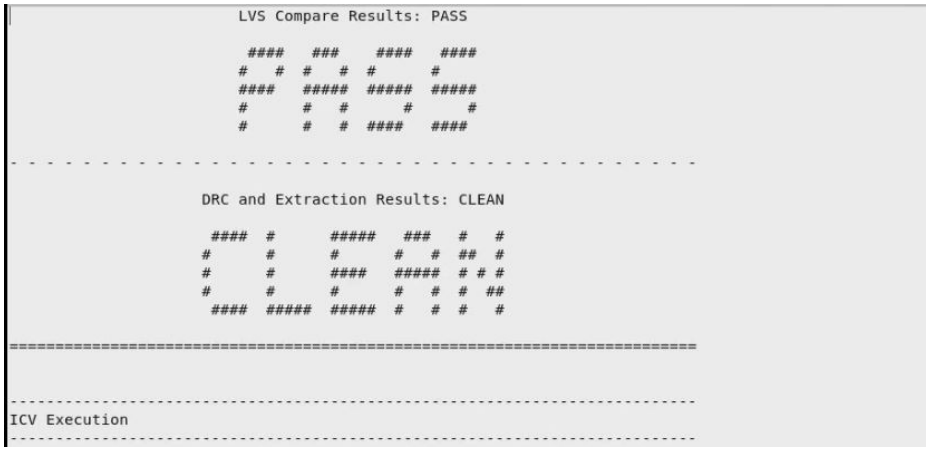


Figura 71: Archivos generados al correr runset de LVS con ERC.

Adicionalmente, en este archivo también se encuentra el resumen de resultados tanto de LVS como de ERC. En la Figura 72 se muestra un ejemplo para una NOT con las pruebas ejecutadas exitosamente.

```
--------------------------------------------------------------------------------
Results Summary
--------------------------------------------------------------------------------

Compare Error Summary
Refer to _INIT_RT.LVS_ERRORS.

LVS Compare Result: PASS
TOP equivalence point:
                [Not_IO, _INIT_RT]

    7 Successful blackbox cells
    0 Failed blackbox cells

    1 Successful equivalence points
    0 Failed equivalence points

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

LVS Device Extraction Error Summary

79 total rules were run.
0 rules NOT EXECUTED.
0 rules have violations.
There are 0 total violations.
Refer to _INIT_RT.LAYOUT_ERRORS
```

Figura 72: Archivos generados al correr runset de LVS con ERC.

En el archivo de extensión **.LAYOUT-ERRORS** se encuentran los detalles de los errores (si existiesen) de ERC. En las figuras 73 y 74 se muestran los casos donde la prueba se ejecuta satisfactoriamente y donde la prueba se ejecuta con violaciones de ERC, respectivamente.

```
                LAYOUT ERRORS RESULTS: CLEAN

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

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

Library name:    /home/administrador/Escritorio/LVS/NOTPG2/NOTPG2_gds
Structure name:  _INIT_RT
Generated by:    IC Validator RHEL64 Q-2019.12-SP2-4.5500898 2020/04/25
Runset name:     /mnt/nfs/compartida/RUNSET_LVS.4a
User name:       administrador
Time started:    2020/09/21 11:42:24AM
Time ended:      2020/09/21 11:42:35AM

Called as: icv -i /home/administrador/Escritorio/LVS/NOTPG2/NOTPG2_gds -c _INIT_RT -s /mnt/nfs/compartida/allicv -sf ICV -vue /mnt/nfs/compartida/RUNSET_LVS.4a
```

Figura 73: Archivos generados al correr runset de LVS con ERC.

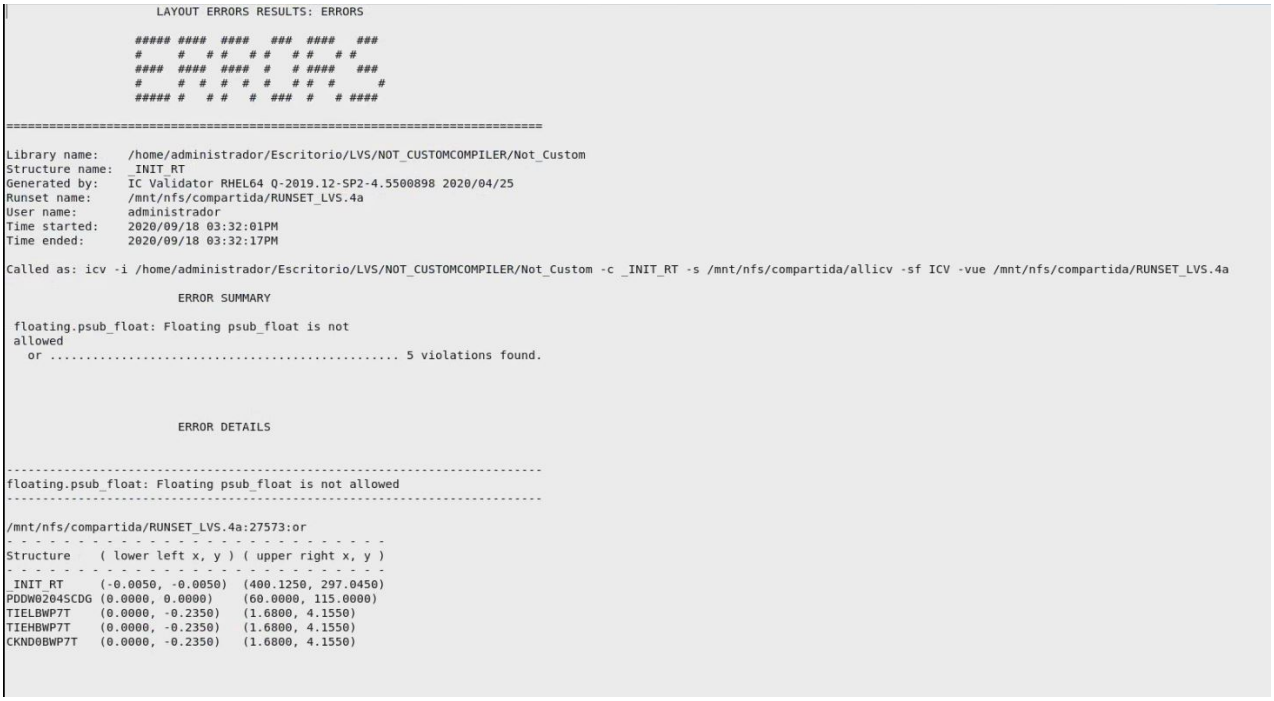


Figura 74: Archivos generados al correr runset de LVS con ERC.

11.5 Primeras pruebas, advertencias y errores generados

Al ejecutar el runset de ERC (el cual es el mismo que el de LVS) se presentaba en repetido número de ocasiones la misma advertencia: El pin *Bulk* de la función que recopila información de configuración de extracción sobre dispositivos genéricos estaba configurado como una terminal(por defecto) como se muestra en la Figura 75, este error se corrigió agregando el tipo de pin correspondiente, que en este caso es Bulk, a cada línea función donde se marcaba la advertencia, como se muestra en la Figura 76.

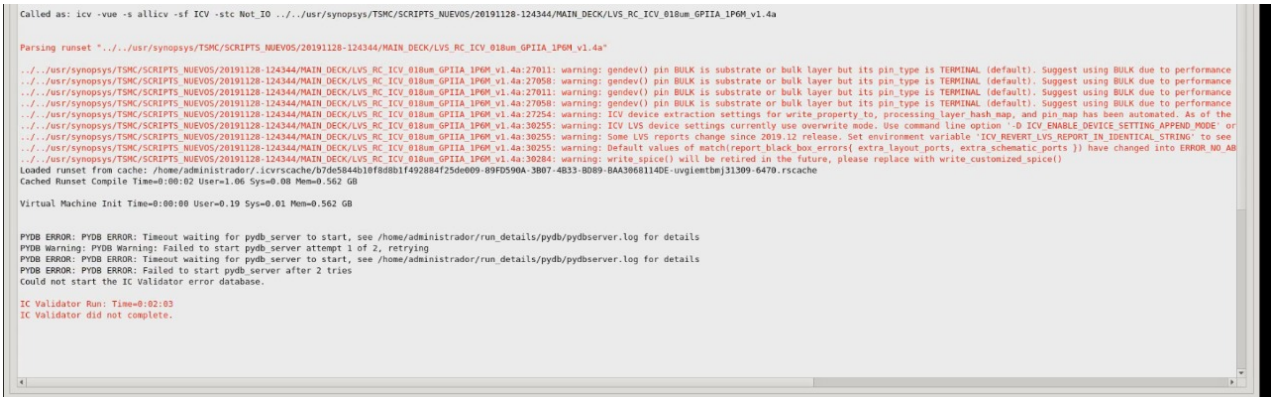


Figura 75: Advertencias presentadas.

```
gendev(
        matrix = device_matrix,
        device_name = "spiral_sym_ct_40k",
        simulation_model_name = "spiral_sym_ct_40k",
        device_body = ct_ind,
        device_layers = { {metal6,"PLUS"},{metal6,"MINUS"},{psub,"BULK",pin_type=BULK},{metal4,"CTAP"} },
        schematic_devices = { { "spiral_sym_ct_40k", {"PLUS","MINUS","BULK","CTAP"} } },
        swappable_pins = {{ "PLUS","MINUS" }},
        properties = { { "NR" }, { "W", DOUBLE, MICRO }, { "S", DOUBLE, MICRO }, { "RAD", DOUBLE, MICRO }, { "LAY" }},
        property_function = ct_ind_func,
        processing_layer_hash = { "aux1" => { sym_rad },"aux2" => { ct_ind },"aux3" => { ind_btm_sym },"aux4" => { segs_not_end },"aux5" => { spacing_3u },"aux6" => { re_sym_ind }},
        merge_parallel = true,
        bulk_relationship = INTERACT
);
```

Figura 76: Solución a la advertencia.

Las primeras pruebas se realizaron con una compuerta Not, en donde se presentaron 5 violaciones de ERC como se muestran en la Figura 75.

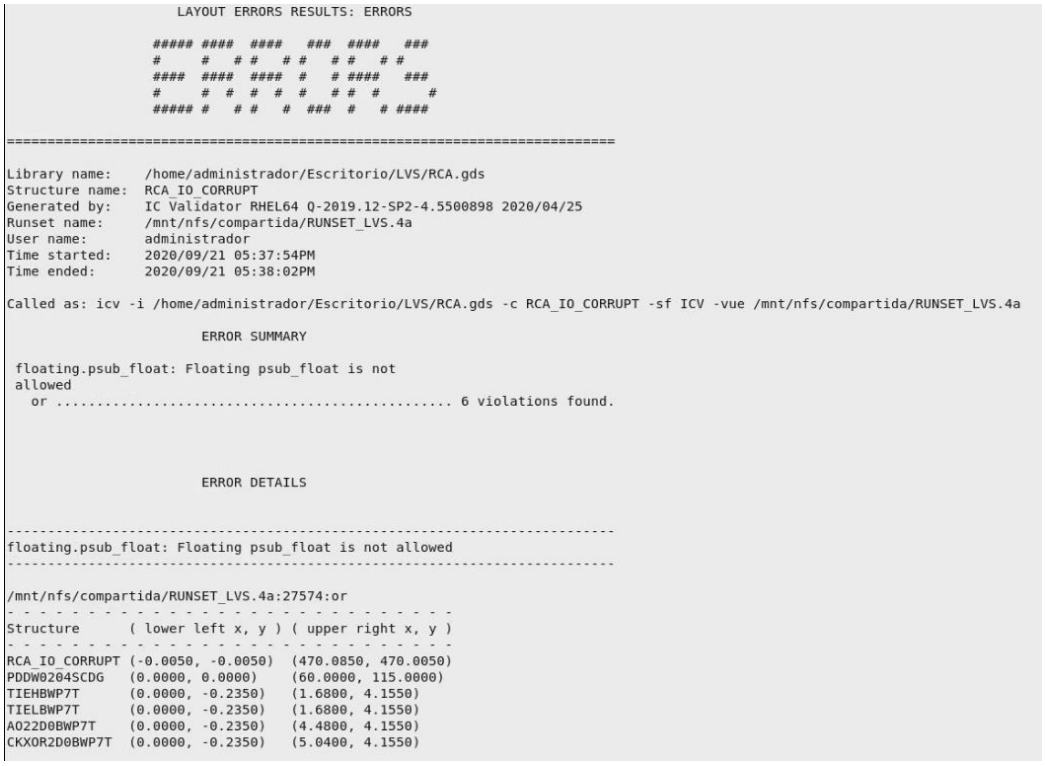
```
                    LAYOUT ERRORS RESULTS: ERRORS

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

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

Library name:    /home/administrador/Escritorio/LVS/NOT_CUSTOMCOMPILER/Not_Custom
Structure name:  _INIT_RT
Generated by:    IC Validator RHEL64 Q-2019.12-SP2-4.5500898 2020/04/25
Runset name:     /mnt/nfs/compartida/RUNSET_LVS.4a
User name:       administrador
Time started:    2020/09/18 03:32:01PM
Time ended:      2020/09/18 03:32:17PM

Called as: icv -i /home/administrador/Escritorio/LVS/NOT_CUSTOMCOMPILER/Not_Custom -c _INIT_RT -s /mnt/nfs/compartida/allicv -sf ICV -vue /mnt/nfs/compartida/RUNSET_LVS.4a

                        ERROR SUMMARY

 floating.psub_float: Floating psub_float is not
 allowed
   or ................................................ 5 violations found.



                        ERROR DETAILS


------------------------------------------------------------------------------
floating.psub_float: Floating psub_float is not allowed
------------------------------------------------------------------------------

/mnt/nfs/compartida/RUNSET_LVS.4a:27573:or
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Structure     ( lower left x, y ) ( upper right x, y )
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
_INIT_RT      (-0.0050, -0.0050)  (400.1250, 297.0450)
PDDW0204SCDG (0.0000, 0.0000)    (60.0000, 115.0000)
TIEL8WP7T     (0.0000, -0.2350)   (1.6800, 4.1550)
TIEH8WP7T     (0.0000, -0.2350)   (1.6800, 4.1550)
CKND0BWP7T    (0.0000, -0.2350)   (1.6800, 4.1550)
```

Figura 77: Violaciones de ERC de una compuerta Not.

Al revisar el detalle de los errores, se encontró que era el mismo error repetido tantas veces como pads declarados tuviera el *netlist* de Verilog. Se encontró que el error se debía a la regla *Floating Well Check* que realiza el ERC. También se ejecutaron las pruebas para un *Full Adder* y un *Ripple Carry Adder*.

```
                    LAYOUT ERRORS RESULTS: ERRORS

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

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

Library name:    /home/administrador/Escritorio/LVS/FullAdder.gds
Structure name:  Full_Adder
Generated by:    IC Validator RHEL64 Q-2019.12-SP2-4.5500898 2020/04/25
Runset name:     /mnt/nfs/compartida/RUNSET_LVS.4a
User name:       administrador
Time started:    2020/09/21 05:30:05PM
Time ended:      2020/09/21 05:30:14PM

Called as: icv -i /home/administrador/Escritorio/LVS/FullAdder.gds -c Full_Adder -sf ICV -vue /mnt/nfs/compartida/RUNSET_LVS.4a

                        ERROR SUMMARY

 floating.psub_float: Floating psub_float is not
 allowed
   or ................................................ 7 violations found.



                        ERROR DETAILS


--------------------------------------------------------------------------
floating.psub_float: Floating psub_float is not allowed
--------------------------------------------------------------------------

/mnt/nfs/compartida/RUNSET_LVS.4a:27574:or
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Structure     ( lower left x, y ) ( upper right x, y )
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Full_Adder    (-0.0050, -0.0050)  (350.2450, 350.0050)
PDDW0204SCDG  (0.0000, 0.0000)    (60.0000, 115.0000)
TIELBWP7T     (0.0000, -0.2350)   (1.6800, 4.1550)
TIEHBWP7T     (0.0000, -0.2350)   (1.6800, 4.1550)
CKXOR2D4BWP7T (0.0000, -0.2350)   (12.3200, 4.1550)
AO22D2BWP7T   (0.0000, -0.2350)   (5.6000, 4.1550)
CKXOR2D0BWP7T (0.0000, -0.2350)   (5.0400, 4.1550)
```

Figura 78: Violaciones de ERC de un Full Adder.



```
                    LAYOUT ERRORS RESULTS: ERRORS

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

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

Library name:    /home/administrador/Escritorio/LVS/RCA.gds
Structure name:  RCA_IO_CORRUPT
Generated by:    IC Validator RHEL64 Q-2019.12-SP2-4.5500898 2020/04/25
Runset name:     /mnt/nfs/compartida/RUNSET_LVS.4a
User name:       administrador
Time started:    2020/09/21 05:37:54PM
Time ended:      2020/09/21 05:38:02PM

Called as: icv -i /home/administrador/Escritorio/LVS/RCA.gds -c RCA_IO_CORRUPT -sf ICV -vue /mnt/nfs/compartida/RUNSET_LVS.4a

                        ERROR SUMMARY

 floating.psub_float: Floating psub_float is not
 allowed
   or ................................................ 6 violations found.



                        ERROR DETAILS


--------------------------------------------------------------------------
floating.psub_float: Floating psub_float is not allowed
--------------------------------------------------------------------------

/mnt/nfs/compartida/RUNSET_LVS.4a:27574:or
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Structure      ( lower left x, y ) ( upper right x, y )
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
RCA_IO_CORRUPT (-0.0050, -0.0050)  (470.0850, 470.0050)
PDDW0204SCDG   (0.0000, 0.0000)    (60.0000, 115.0000)
TIEHBWP7T      (0.0000, -0.2350)   (1.6800, 4.1550)
TIELBWP7T      (0.0000, -0.2350)   (1.6800, 4.1550)
AO22D0BWP7T    (0.0000, -0.2350)   (4.4800, 4.1550)
CKXOR2D0BWP7T  (0.0000, -0.2350)   (5.0400, 4.1550)
```

Figura 79: Violaciones de ERC de un Ripple Carry Adder.

11.6 Solución a errores

Los errores en todos los circuitos probados fue el mismo repetido distinto número de veces. El error presentado indicó que el sustrato tipo p de los transistores nMOSFET del circuito estaba **flotando**, lo cual no está permitido. Se analizó el caso y se llegó a la conclusión de que el sustrato tipo p no estaba contacto con el *NWELL* de los transistores.

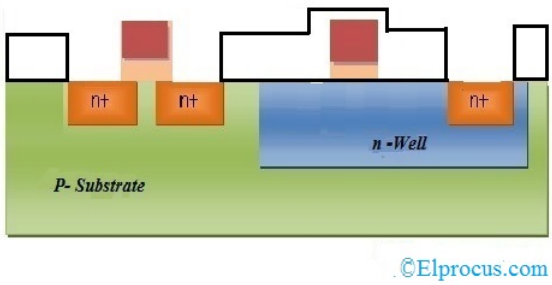


Figura 80: Sustrato tipo p y NWELL de un transistor

El runset de LVS está escrito en lenguaje PXL, este lenguaje define sus conexiones mediante 5 comandos: **interacting, not-interacting, and, or y not**.

Se llegó a la conclusión de que los errores presentados se deben a que originalmente el sustrato tipo p excluía al NWELL en la definición de conexiones para el sustrato tipo p.

En el runset se encontró la conexión de esta forma:

```
psub = psube not nxwell
```

La línea anterior se cambió en el runset por

```
psub = psube and nxwell
```

Conectando de esta forma el sustrato tipo p con el NWELL, solucionando los errores de ERC.

11.7 Verificación de reglas de diseño eléctrico con una compuerta NOT con comandos y VUE tool

Se verificó la prueba ERC para una compuerta Not desde el comando en terminal y desde la *VUE tool*, en la Figura 81 se puede observar el archivo **NOT.LAYOUT-ERRORS**, el cual nos da detalles de los errores de ERC. En este caso la prueba ha sido ejecutada exitosamente.

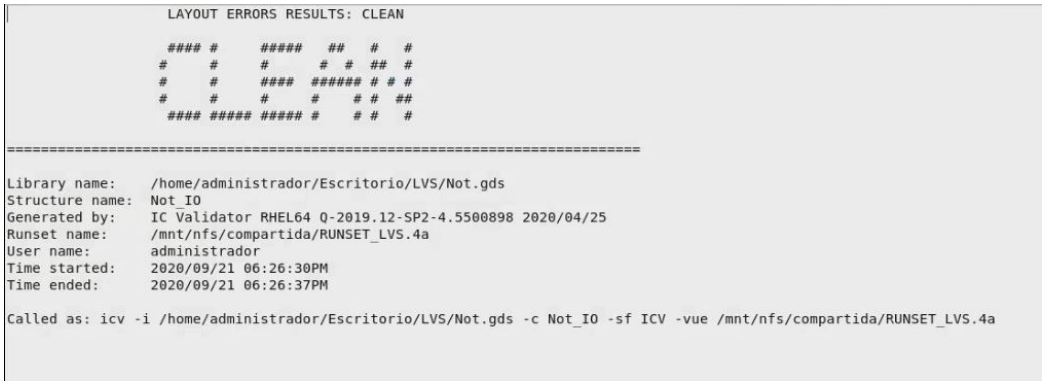


Figura 81: Prueba de ERC exitosa para una compuerta NOT.

En la Figura 82, se puede observar el archivo **NOT.RESULTS**, el cual contiene la información de los resultados de LVS y los de ERC/Extracción. En este caso, tanto la prueba de LVS como la de ERC han sido ejecutadas exitosamente.

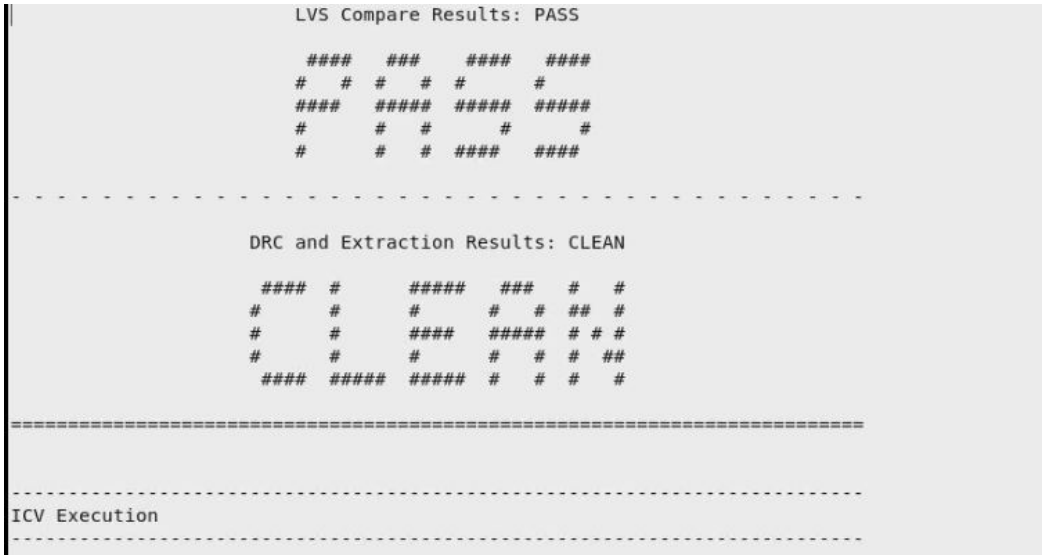


Figura 82: Pruebas de ERC y LVS pasadas exitosamente.

11.8 Verificación de reglas de diseño eléctrico con una compuerta NOT con Custom Compiler

La prueba de ERC también se puede realizar desde la herramienta Custom Compiler, en donde se importa el *Layout* de la librería *Milkyway* creada en el proceso de síntesis física. En la Figura 83 se puede observar el *layout* de una compuerta Not.

En la Figura 84 se muestra la configuración del runset. Para llegar a esta ventana de configuración se debe hacer click en la pestaña **Verification** -> **LVS** -> **Setup and Run**.

En la Figura 85 se muestra la pestaña **NOT.LAYOUT-RESULTS**, donde se observa que la prueba de ERC se ha ejecutado sin violaciones.

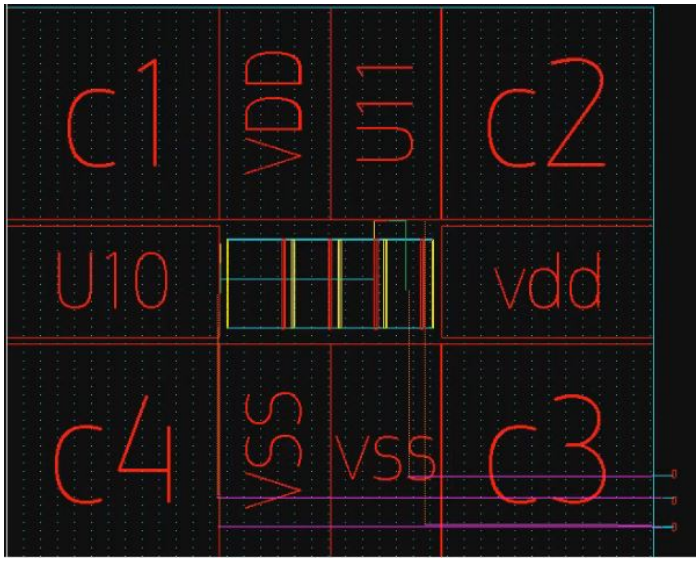


Figura 83: Configuración del runset en Custom Compiler para una compuerta Not

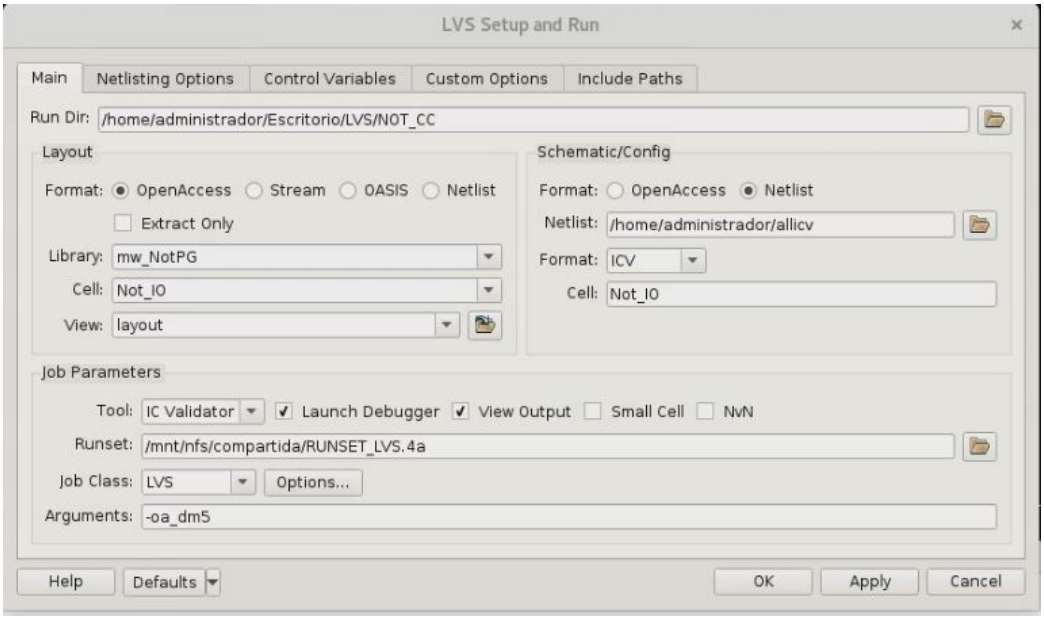


Figura 84: Configuración del runset en Custom Compiler para una compuerta Not

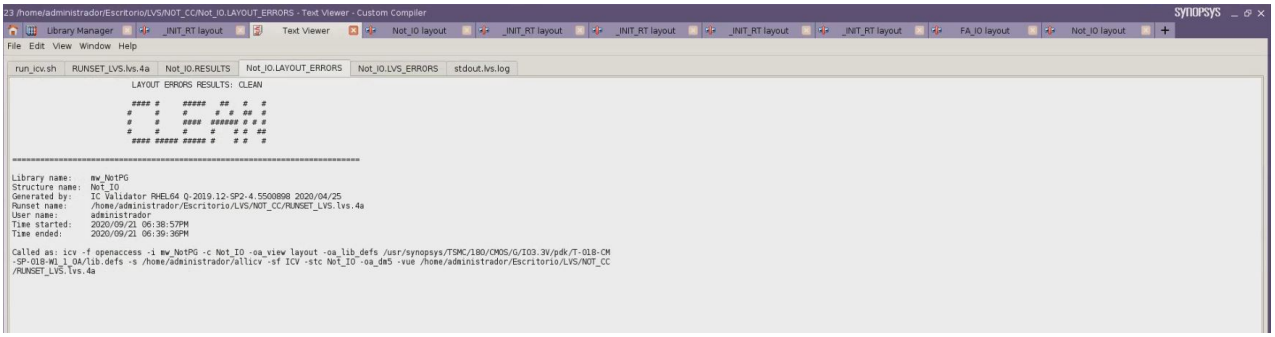


Figura 85: Pruebas de ERC ejecutada exitosamente para una Not.

11.9 Verificación de reglas de diseño eléctrico con un Full Adder con comandos y VUE tool

Se verificó la prueba ERC para un circuito *Full Adder* desde el comando en terminal y desde la *VUE tool*, en la Figura 86 se puede observar el contenido del archivo **FULL-ADDER.LAYOUT-ERRORS**, el cual nos da detalles de los errores de ERC. En este caso la prueba ha sido ejecutada exitosamente.

En la Figura 87 se muestra el contenido del archivo **FULL-ADDER.RESULTS**, en donde se observa que las pruebas tanto de LVS como de ERC se han ejecutado sin violaciones.

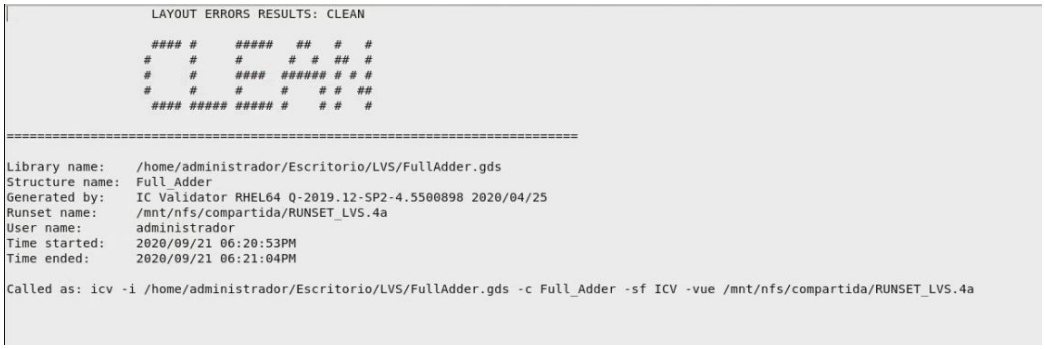


Figura 86: Prueba de ERC exitosa para un Full Adder.

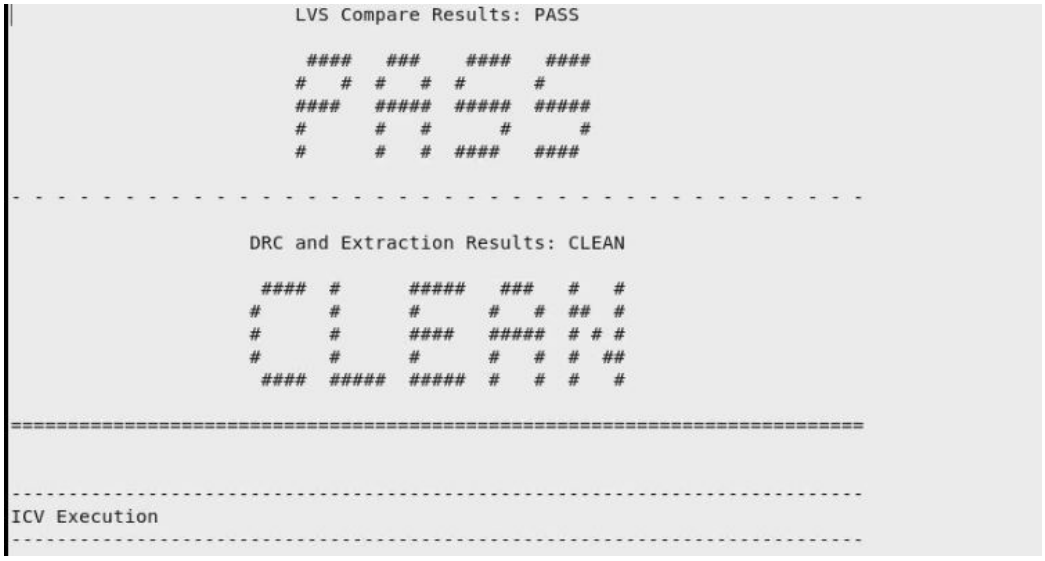


Figura 87: Pruebas de ERC y LVS pasadas exitosamente.

11.10 Verificación de reglas de diseño eléctrico con un Full Adder con Custom Compiler

Se importó el *Layout* de la librería *Milkyway* creada en el proceso de síntesis física. En la Figura 88 se puede observar el *layout* de un circuito *Full Adder*.

En la Figura 89 se muestra la configuración del runset. Para llegar a esta ventana de configuración se debe hacer click en la pestaña **Verification** -> **LVS** -> **Setup and Run**.

En la Figura 90 se muestra la pestaña **FULL-ADDER.LAYOUT-RESULTS**, donde se observa que la prueba de ERC se ha ejecutado sin violaciones.

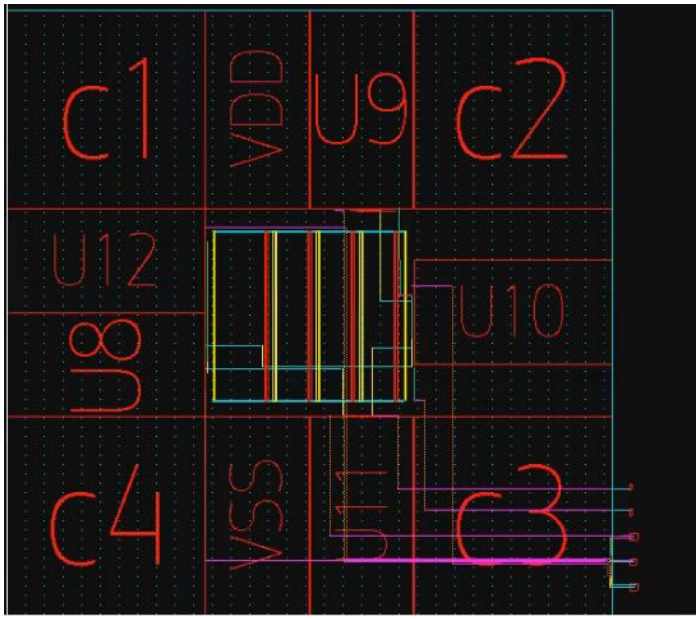


Figura 88: Layout de un Full Adder visto en Custom Compiler.

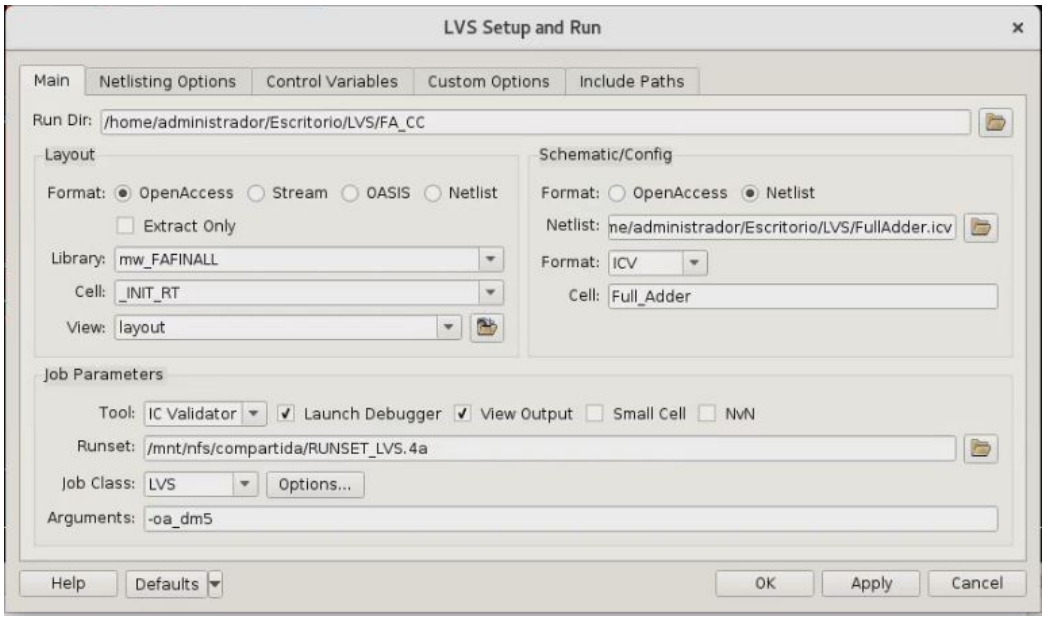


Figura 89: Configuración en Custom Compiler para ejecutar ERC y LVS a un Full Adder.

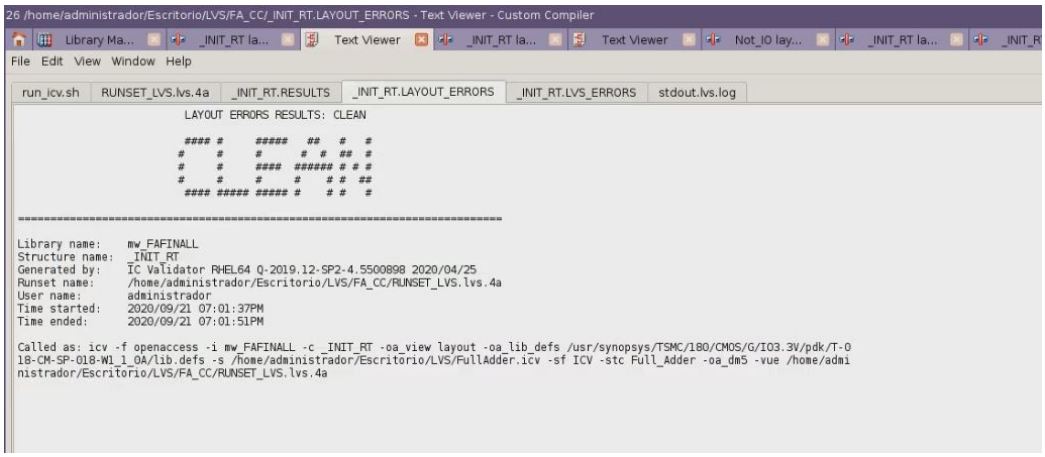


Figura 90: Prueba de ERC ejecutada exitosamente.

11.11 Verificación de reglas de diseño eléctrico con un Ripple Carry Adder con comandos y VUE tool

Se verificó la prueba ERC para un circuito *Ripple Carry Adder* desde el comando en terminal y desde la *VUE tool*, en la Figura 91 se puede observar el contenido del archivo **RCA.LAYOUT-ERRORS**, el cual nos da detalles de los errores de ERC. En este caso la prueba ha sido ejecutada exitosamente.

En la Figura 92 se muestra el contenido del archivo **RCA.RESULTS**, en donde se observa que las pruebas tanto de LVS como de ERC se han ejecutado sin violaciones.

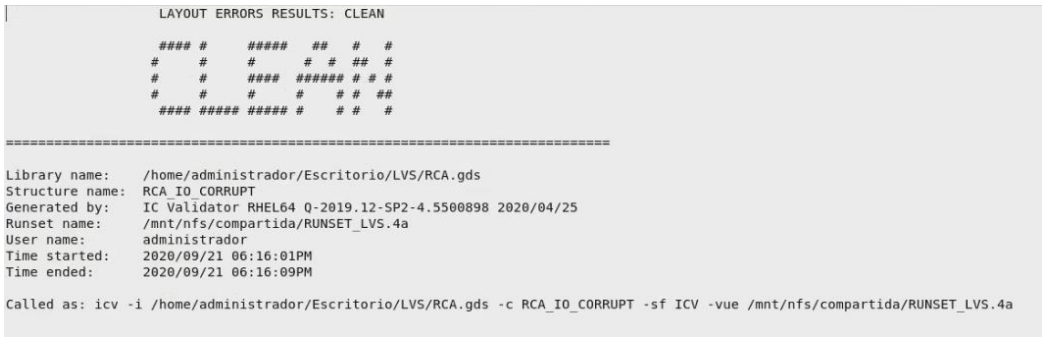


Figura 91: Prueba de ERC exitosa para un Ripple Carry Adder.

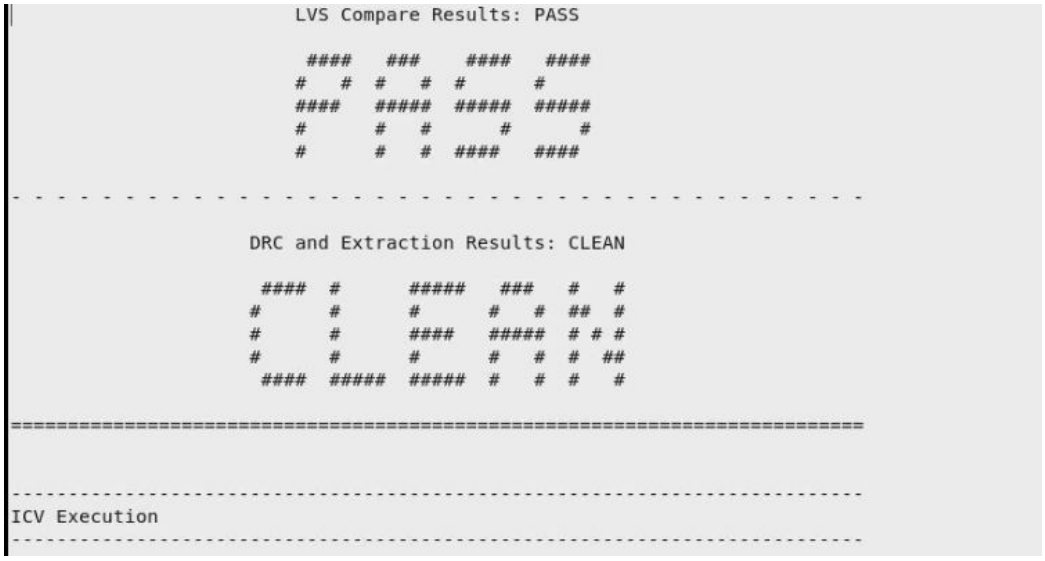


Figura 92: Pruebas de ERC y LVS pasadas exitosamente.

11.12 Verificación de reglas de diseño eléctrico con un Ripple Carry Adder con Custom Compiler

Se importó el *Layout* de la librería *Milkyway* creada en el proceso de síntesis física. En la Figura 93 se puede observar el *layout* de un circuito *Ripple Carry Adder*.

En la Figura 94 se muestra la configuración del runset. Para llegar a esta ventana de configuración se debe hacer click en la pestaña **Verification** -> **LVS** -> **Setup and Run**.

En la Figura 95 se muestra la pestaña **RCA.LAYOUT-RESULTS**, donde se observa que la prueba de ERC se ha ejecutado sin violaciones.

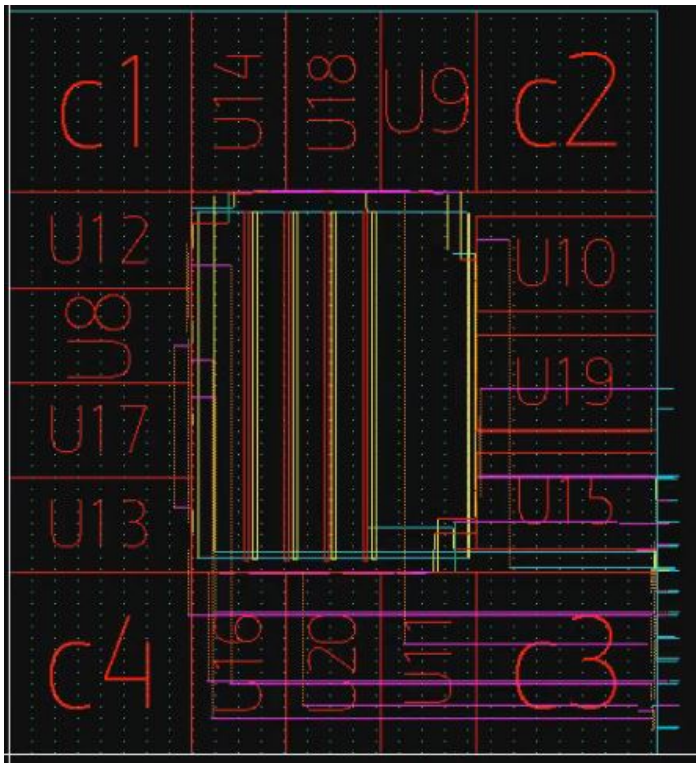


Figura 93: Layout de un Ripple Carry Adder visto en Custom Compiler.

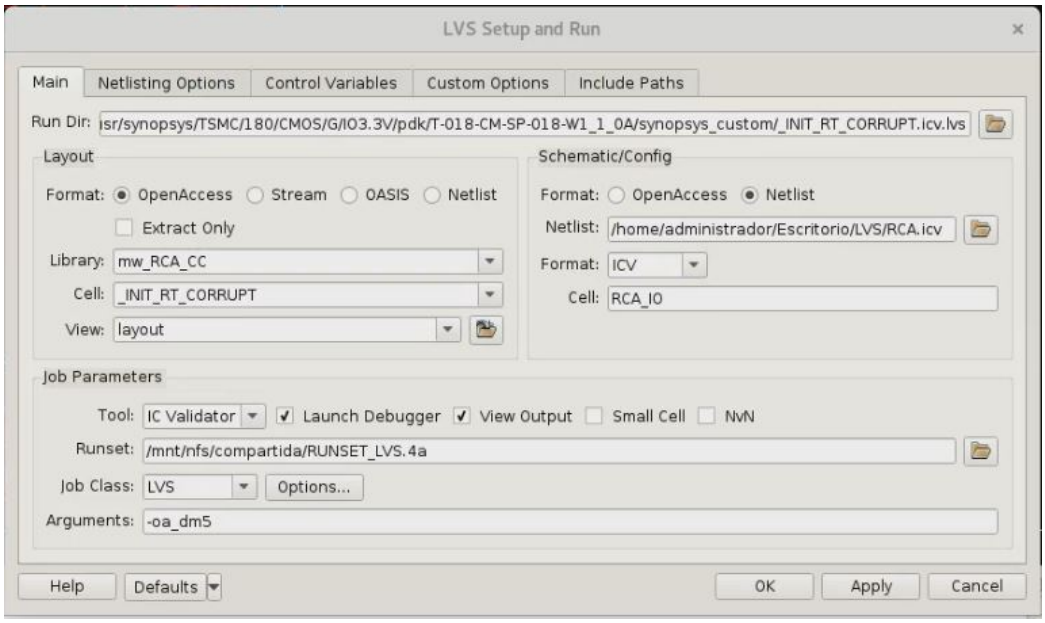


Figura 94: Configuración en Custom Compiler para ejecutar ERC y LVS a un RCA.

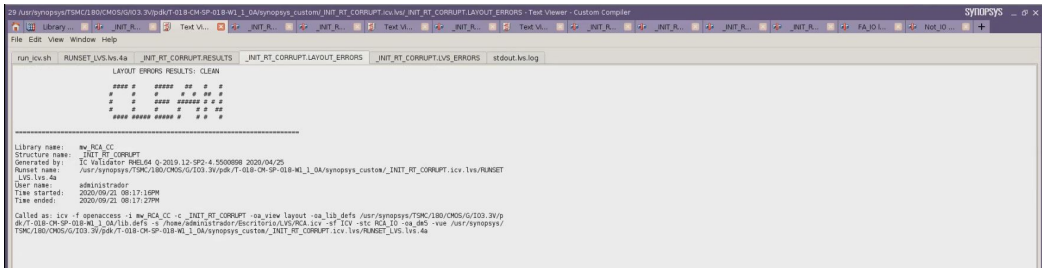


Figura 95: Prueba de ERC ejecutada exitosamente.

CAPÍTULO 12

Conclusiones

1. Se comprendió exitosamente la sección de anillo de entradas/salidas, antena y reglas de diseño eléctrico del flujo de diseño de un circuito integrado logrando su implementación en el proyecto de la línea de investigación.
2. Se estudió a profundidad la documentación de Synopsis y TSMC para elaborar las pruebas de antena y ERC.
3. Se seleccionó la celda adecuada para el anillo de entradas/salidas.
4. Se elaboraron corridas de antena y ERC sin errores.
5. Se documentó el uso correcto de estas partes del flujo de diseño para futuros estudiantes involucrados en esta línea de investigación.

CAPÍTULO 13

Recomendaciones

1. Se recomienda el uso de toda la documentación disponible en la página de Synopsis y Solvnet para conocer el funcionamiento a detalle de cada herramienta.
2. Se recomienda tener las últimas versiones de las herramientas y de su documentación, ya que una fuente de error puede ser la incompatibilidad de versiones.
3. Se recomienda realizar tanto la prueba de antena como la de ERC con comandos desde una terminal, ya que es la forma más simple de ejecutar las pruebas y esto hará que los archivos generados por las pruebas se creen en el directorio donde se abrió la terminal.
4. Se recomienda estudiar los trabajos de graduación anteriores de esta línea de investigación, ya que contienen procedimientos o información importante sobre los pasos del flujo de diseño.

CAPÍTULO 14

Bibliografía

1. Rubio Vásquez, Steven Hiram.*Definición del Flujo de diseño para Fabricación de un Chip con tecnología VLSI CMOS* . Tesis de Licenciatura de Ingeniería Electrónica.Universidad del Valle de Guatemala. 2019.

2. Santos Chonay, Jonathan A. *Diseño de un sumador/restador de 32 bits con tecnología CMOS en un proceso de 28 nanómetros usando aplicaciones de diseño de la empresa Synopsys* .Tesis de Licenciatura de Ingeniería Electrónica.Universidad del Valle de Guatemala. 2014.

3. A. Kahng, J. Lienig, I. Markov y J. Hu. *CMOS VLSI Design: A circuits and systems perspective*. Pearson India, 2015, ISBN: 9789332559042. Extraído de: https://books.google.com.gt /books?id=0RAwDwAAQBAJ.

4. N. Weste y D. Harris. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer Netherlands, 2011, ISBN: 9789048195916. extraído de: https://books.google.com.gt /books?id=DWUGHyFVpboC.

5. RAJ, A.A. y LATHA,T. *VLSI Design*. PHI Learning, 2008.ISBN:9788120334311. Extraído de: https://boo ks.google.com.gt /books?id=CO8zq6_vcr8C.

6. Synopsys, Solvnet. *IC Compiler Classic Router User Guide*, version J-2014.09. Manual. Synopsys. 2020.

7. Synopsys, Solvnet. *IC Compiler Design Planning User Guide*, version L-2016.03-SP2. Manual. Synopsys. 2020.

CAPÍTULO 15

Anexos

CAPÍTULO 16

Glosario

Drain (Mosfet): terminal de drenaje o salida de un transistor mosfet. 16

ERC: siglas de Electrical Rule Check. 11

gate: terminal compuerta de un transistor mosfet. Funciona mediante la aplicación de una tensión (con un valor mínimo llamada tensión umbral). 16

LVS: significa *Layout vs Schematic*, la cual es una prueba de integridad física que compara equivalencias entre los componenetes del layout y del esquemático. 52

N-Well: Se refiere al pozo N de un transistor mosfet,es decir, el sustrato es tipo p dopado. 14

Net: se refiere a una colección de dos o más componentes interconectados. 15

Netlist: Es la descripción de un circuito a nivel de compuertas. 11

Runset: se refiere al conjunto de reglas y comandos para ejecutar un proceso o un conjunto de procesos específico. 11

Source (Mosfet): Terminal de alimentación de un transistor mosfet.. 16

VDD: Nodo de Alimentación de un circuito. 11

VSS: Nodo tierra de un circuito. 11

VUE tool: Software utilizado para correr las pruebas de integridad física mediante una interfaz gráfica de la herramienta IC Compiler. 48