UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un Circuito Integrado con Tecnología de 180 nm usando Librerías de Diseño de TSMC: Ejecución de la Síntesis física, Verificación de Reglas de Diseño y Corrección de Errores Obtenidos.

Trabajo de graduación presentado por Antonio Altuna Hernandez para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2021

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un Circuito Integrado con Tecnología de 180 nm usando Librerías de Diseño de TSMC: Ejecución de la Síntesis física, Verificación de Reglas de Diseño y Corrección de Errores Obtenidos.

Trabajo de graduación presentado por Antonio Altuna Hernandez para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2021

Vo.Bo.:

(f) _____

Ing. Luis Nájera

Tribunal Examinador:

(f) _____

Ing. Luis Nájera

(f) _____

MSc. Carlos Esquit

(f) _____

Ing. Luis Pedro Montenegro

Fecha de aprobación: Guatemala, 5 de diciembre de 2021.

Prefacio

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras vitae eleifend ipsum, ut mattis nunc. Pellentesque ac hendrerit lacus. Cras sollicitudin eget sem nec luctus. Vivamus aliquet lorem id elit venenatis pellentesque. Nam id orci iaculis, rutrum ipsum vel, porttitor magna. Etiam molestie vel elit sed suscipit. Proin dui risus, scelerisque porttitor cursus ac, tempor eget turpis. Aliquam ultricies congue ligula ac ornare. Duis id purus eu ex pharetra feugiat. Vivamus ac orci arcu. Nulla id diam quis erat rhoncus hendrerit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Sed vulputate, metus vel efficitur fringilla, orci ex ultricies augue, sit amet rhoncus ex purus ut massa. Nam pharetra ipsum consequat est blandit, sed commodo nunc scelerisque. Maecenas ut suscipit libero. Sed vel euismod tellus.

Proin elit tellus, finibus et metus et, vestibulum ullamcorper est. Nulla viverra nisl id libero sodales, a porttitor est congue. Maecenas semper, felis ut rhoncus cursus, leo magna convallis ligula, at vehicula neque quam at ipsum. Integer commodo mattis eros sit amet tristique. Cras eu maximus arcu. Morbi condimentum dignissim enim non hendrerit. Sed molestie erat sit amet porttitor sagittis. Maecenas porttitor tincidunt erat, ac lacinia lacus sodales faucibus. Integer nec laoreet massa. Proin a arcu lorem. Donec at tincidunt arcu, et sodales neque. Morbi rhoncus, ligula porta lobortis faucibus, magna diam aliquet felis, nec ultrices metus turpis et libero. Integer efficitur erat dolor, quis iaculis metus dignissim eu.

Índice

Lista de figuras

Lista de cuadros

Resumen

El objetivo de esta investigación es mejorar la síntesis física, actual, como parte del flujo de diseño para el desarrollo de circuitos integrados digitales a nanoescala. Esta por ser la tercera iteración del proyecto, para realizar el flujo de diseño, se cuenta con la documentación de las promociones pasadas. El primer paso para realizar este proyecto es el análisis de los resultados anteriores. Para hacer esto se replicarán los trabajos realizados anteriormente por: Luis Nájera y Luis Abadilla, que fueron los encargados de la síntesis física en su respectiva promoción. Luego de esto se migrarán estos procesos de la herramienta de Custom Compiler I a Custom Compiler II. Además se verificarán las librerías de **TSMC** que se están utilizando para corroborar que estén actualizadas y sean compatibles con el proceso de manufactura para el que se está diseñando.

Para etapa de verificación y validación de los resultados se realizarán los siguientes tres procesos de verificación: *Design Rule Check*, *Electrical Rule Check* y *Antenna Rule Check*. Para estos procesos de verificación también se cuenta con la documentación de los trabajos de Matthias Sibrian y Marvin Flores. Con las librerías en orden en conjunto con la síntesis física, se estarán haciendo múltiples pruebas con diferentes circuitos eléctricos digitales como: una compuerta NOT, un Full Adder, un Ripple Carry Adder y un chip de diseño personalizado. Estos cuatro circuitos serán pasados por el proceso de síntesis física y las verificaciones mencionadas anteriormente para validar que los resultados obtenidos por la síntesis física se puedan manufacturar por la empresa **TSMC**.

Abstract

This is an abstract of the study developed under the

CAPÍTULO 1

Introducción

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque eget consequat risus. Praesent a quam lacinia, consequat eros id, auctor tellus. Phasellus a dapibus arcu, vitae luctus leo. Aliquam erat volutpat. Suspendisse ac velit quam. Nullam risus nibh, lobortis vehicula elit non, pellentesque volutpat odio. Donec feugiat porta sapien gravida interdum. Cras odio nunc, lobortis sed pellentesque imperdiet, facilisis eu quam. Praesent pharetra, orci at tincidunt lacinia, neque nulla ornare lacus, ut malesuada elit risus non mi. Fusce pellentesque vitae sapien sed mollis. Curabitur viverra at nulla vitae porta. In et mauris lorem.

Vestibulum faucibus fringilla justo, eget facilisis elit convallis sit amet. Morbi nisi metus, hendrerit quis pellentesque non, faucibus at leo. Proin consectetur, est vel facilisis facilisis, arcu felis vestibulum quam, et fringilla metus neque at enim. Nunc justo mauris, egestas quis maximus eget, viverra vehicula nunc. Fusce eu nulla elementum, condimentum diam at, aliquam leo. Nullam sed sodales enim, eu imperdiet risus. Aliquam ornare augue leo, fringilla mattis nunc facilisis eget. Nam faucibus, libero a aliquet fermentum, magna arcu ultrices lacus, a placerat tortor turpis ut purus.

Integer eget ligula non metus egestas rutrum sit amet ut tellus. Aliquam vel convallis est, eu sodales leo. Proin consequat nisi at nunc malesuada gravida. Aliquam erat volutpat. Aliquam finibus interdum dignissim. Etiam feugiat hendrerit nisl, hendrerit feugiat ex malesuada in. Cras tempus eget arcu vitae congue. Ut non tristique mauris. Vivamus in mattis ipsum. Cras bibendum, enim bibendum commodo accumsan, ligula nulla porttitor ex, et pharetra eros nisl eget ex. Morbi at semper arcu. Curabitur massa sem, maximus id metus ut, molestie tempus quam. Vivamus dictum nunc vitae elit malesuada convallis. Donec ac semper turpis, non scelerisque justo. In congue risus id vulputate gravida. Nam ut mattis sapien.

CAPÍTULO 2

Antecedentes

La Universidad del Valle de Guatemala (UVG) a lo largo de los años se ha caracterizado por estar a la vanguardia en la investigación de nuevas tecnologías y metodologías en el área de las ciencias aplicadas. El Ing. Carlos Esquit, en el 2009, ingresó como nuevo director del departamento de Ing. Electrónica y Mecatrónica de la universidad y consigo trajo las destrezas y aprendizajes adquiridos en Estados Unidos en las ramas de micro y nano electrónica. Al iniciar, realizó una reforma al mapa curricular de Ing. Electrónica para enseñar dichas disciplinas aprendidas en el extranjero.

En el 2013, se inicia a impartir el curso de Introducción al diseño de sistemas **VLSI**. Al inicio las herramientas con las que se impartió este curso fueron gratuitas y permitían realizar diseños básicos a micro y nano escala de circuitos. Al año siguiente, el ingeniero formó una alianza con la empresa Synopsys. Esta se encuentra a la vanguardia en lo que se refiere a soluciones que facilitan el diseño en silicio de chips y su verificación. Esta proporcionó herramientas que actualmente utilizan los líderes de la industria de semiconductores. Con estas herramientas se realizó el primer diseño en silicio a nanoescala de 28nm, realizado por el Ing. Jonathan de los Santos [1] en 2014.

En la nueva reforma curricular, que entró en vigencia en 2015, se añadieron los cursos de Nanoelectrónica 1 y 2. El fin de este cambio, que actualmente sigue vigente, es impartir teoría de **VLSI** y desarrollar en los alumnos competencias en el área de investigación del diseño de chips a micro y nano escala. Esto utilizando las herramientas que el software Synopsys le proporciona a la UVG. Con estas herramientas en el 2019 y 2020 se realizó el diseño para el primer chip con tecnología CMOS a nanoescala en Guatemala.

En el año 2019 los alumnos Luis Nájera y Steven Rubio en conjunto con el apoyo de Interuniversity Microelectronics Centre (**IMEC**) y bajo la supervisión del Ing. Carlos Esquit desarrollaron lo que fue la primera iteración de este proyecto. Los resultados de este equipo fueron la implementación de las librerías de **TSMC** en el desarrollo de un flujo de diseño para la fabricación del primer circuito integrado en la historia de Guatemala, abriendo paso a una nueva área de investigación en la UVG. Los trabajos que documentan dichos resultados se encuentran en [2] y [3].

Los antecesores directos a este proyecto que trabajaron en el ciclo 2020 a 2021 en lo que se refiere a la síntesis física, Veriricación **ERC** (Electrical Rule Check), Antenna Rule Check y verificación **DRC** (Design Rule Check) fueron el quipo de: Luis Abadilla [4], Marvin Flores [5] y Matthias Sibrian [6]. Los resultados obtenidos por ellos fueron diferentes simulaciones funcionales, con las librerías de **TSMC** en los programas IC Validator y IC Compiler I. Adicionalmente dejaron documentación para utilizar los programas y scripts para replicar su trabajo.

CAPÍTULO 3

Justificación

La fabricación de Circuitos Integrados (**CI**) puede ser un proceso largo y riguroso. El elaborar un flujo de diseño para la fabricación de estos **CI** abre nuevas oportunidades para lo que es el diseño y manufactura de estos dispositivos facilitando y automatizando el proceso. En el mercado actualmente hay una gran variedad de **CI** de propósito general. Sin embargo el poder crear **CI** dedicados permite que los dispositivos electrónicos que los utilizan mejoren su desempeño (velocidad, consumo de potencia, etc...). Esto se debe a que están diseñados para cumplir una única funcionalidad. A diferencia de aquellos de propósito general que cuentan con múltiples módulos internos que realizan tareas varias, que si bien lo hacen más versátil, hacen que este no dedique el 100 % de sus recursos a una tarea.

Es por esto que el crear un flujo de diseño funcional, con herramientas de estado del arte, permite impulsar el desarrollo de la industria de semiconductores en Guatemala. Por medio de la creación de plazas laborales que permitan el diseño y manufactura de estos dispositivos en el país así como la comercialización de estos productos.

Los beneficios inmediatos de este proyecto son el facilitar la utilización del flujo de diseño a quienes proceden esta investigación. Para que se pueda invertir tiempo en únicamente a mantener actualizado el flujo de diseño y permitir que aquellos que se quieran beneficiar de este puedan crear proyectos de alto nivel en los departamentos de Electrónica, Macarrónica y Biomédica de la UVG. Además, a largo plazo, que otras carreras puedan colaborar de forma interdisciplinaria para la creación de proyectos más ambiciosos y con un mayor alcance para beneficio de la comunidad UVG y del país.

CAPÍTULO 4

Objetivos

Objetivo General

Mejorar el proceso actual de síntesis física como parte del flujo de diseño para el desarrollo de un circuito integrado a nanoescala, trabajando en conjunto con los demás grupos, para mejorar la interconexión con los demás módulos, adicionalmente someter el *layout* a las diferentes verificaciones y trabajar en conjunto con los integrantes de mi grupo para solventar violaciones a las normas de diseño de forma eficiente para así obtener un *layout* funcional sin errores de DRC.

Objetivos Específicos

- Migrar la síntesis física de la herramienta IC Compiler I a IC Compiler II.
- Verificar la compatibilidad de las librerías de **TSMC** utilizadas en el desarrollo de la síntesis física.
- Posicionar de forma estratégica los diferentes componentes del circuito integrado para que el proceso de ruteo sea eficiente y minimice los problemas en las etapas posteriores del flujo de diseño, adicionalmente que se cumpla con las restricciones de área, brindada por **TSMC**, que es de $2mm^2$.
- Validar los resultados de la síntesis física por medio de la verificación DRC, comprendiendo las diferentes violaciones a las normas de diseño y corrigiéndolas hasta que el *layout* este libre de errores.
- Mantener la comunicación con mi grupo de trabajo y los demás para solucionar los errores obtenidos de forma rápida y consolidar el flujo de diseño.
- Obtener y delegar los archivos y *scripts* resultantes del proceso de síntesis física y DRC al grupo encargado de la automatización del flujo de diseño.

CAPÍTULO 5

Alcance

El alcance de este trabajo de graduación se limita a tres grandes ejes. El primero es migrar la síntesis física, heredada por los estudiantes de la cohorte anterior, de la herramienta *IC Compiler I* a la herramienta de *IC Compiler II*. Esta migración se hace con el fin de actualizar el flujo de diseño ya que las herramientas necesarias para las otras partes del flujo de diseño como *Custom Compiler* han empezado a desistir de algunas facilidades con la versión de *IC Compiler I*.

El segundo eje es la mejora del proceso de síntesis física en la herramienta de *IC compiler II* utilizando los archivos, *runsets* y documentación de **TSMC** por el intermediario y patrocinador **IMEC** utilizando los programas de la suite de Synopsys a nuestra disposición. Este trabajo pretende ser una guía para los futuros equipos de investigación con el fin de facilitar la comprensión de los archivos, los comandos y el proceso de síntesis física en general. Esto es de vital importancia ya que agiliza la curva de aprendizaje de la siguiente generación y sirve como guía para replicar resultados previos.

El tercer y último eje es la verificación y corrección de reglas de diseño por medio de la verificación *Design Rule Check* o **DRC** utilizando los *runsets*, archivos y documentación de **TSMC** por medio de **IMEC** utilizando las herramientas de la suite de Synopsys a nuestra disposición. Esta verificación se hace con el fin de minimizar los errores de diseño de los diferentes diseños a nanoescala realizados y documentados en este trabajo.

CAPÍTULO 6

Marco teórico

VLSI

Very Large Scale Integration o **VLSI** es el proceso completo por el cual se fabrica un circuito integrado (**CI**), los cuales están compuestos, actualmente, de millones de transistores a nanoescala. El proceso de diseño y simulación se le conoce como Flujo de Diseño. [3] Dicho proceso sigue una estructura top-down la cual implica que el proceso se lleva a cabo desde la jerarquía más alta (macro) hasta el proceso más específico (micro).

Actualmente las computadoras, teléfonos y demás dispositivos electrónicos con capacidad de computo cuentan con un **CI** denominado procesador. El procesador de estos dispositivos es el encargado de realizar el cómputo de la información y distribuirla a lo largo de las diferentes piezas de hardware que integran a estos dispositivos. Las empresas líder en el diseño **VLSI** para procesadores de dispositivos electrónicos actualmente son Intel, AMD, Samsung y recientemente se incorporó a este campo, de diseñó a gran escala, Apple.

Apple actualmente cuenta con su primera generación de procesadores diseñados por ellos especializados para mejorar el actual rendimiento de sus computadoras, tablets y laptops. Este procesador se conoce como *M1* y cuenta con 16 miles de millones de transistores con una tecnología de 5nm[7]. A continuación una imagen:

Figura 1: Primer procesador dedicado para computadoras Apple, M1 con 16 miles de millones de transistores con una tecnología de 5nm. [7]

Flujo de Diseño [6]

Es el proceso por el cual se obtiene el diseño de un **CI** fabricable y funcional. Este proceso puede dividirse en múltiples niveles de abstracción que conforman las diferentes capas del diseño de un **CI**. Las seis capas en las que se divide este proceso son: sistema, algoritmo, arquitectura, lógica, física y *singoff*. A continuación se describe a mayor detalle cada uno de ellas. Cabe resaltar que la necesidad de dividir todo el proceso de fabricación en capas surge ya que la complejidad de diseño es tal que requiere de especialización en diferentes disciplinas. El segmentar en capas permite que cada especialista trabaje en un nivel de abstracción específico aportado al flujo de diseño flexibilidad, cooperación y calidad.

Ingenieros de diseño de Intel Corp pertenecientes al equipo de NIKE Design Technology mencionan que en sus procesos de trabajo para el diseño **VLSI** es necesario particionar las diferentes tareas a realizar en el flujo de diseño. Cada fase a cargo de ingenieros a especializados. Sin embargo es importante recalcar que puede existir una partición innecesaria (sobresegmentación del flujo) de tareas que puede resultar en optimización local de cada uno de los procesos. Esto en mayor esquema puede presentar problemas en la optimización general del **CI** ya que en la marcha surgen cambios que afectan diferentes segmentos del flujo de diseño y puede tomar días arreglar estos problemas. [8] Es por esto que es importante la segmentación apropiada del flujo del diseño y que sean equipos de ingenieros especializados que permiten agilizar este proceso. A continuación se sugiere un particionamiento apropiado para el equipo de trabajo con el que se cuenta.

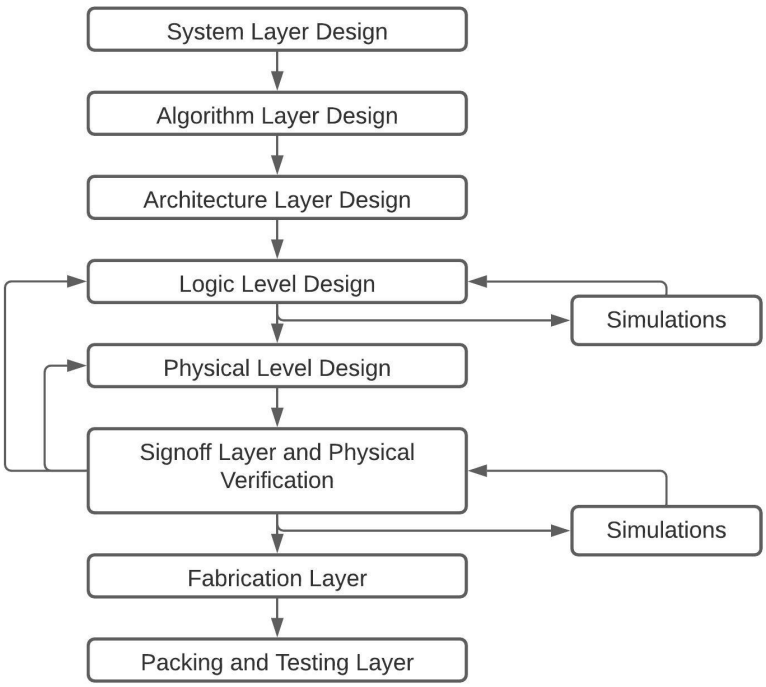

Figura 2: Flujo de diseño simplificado para el diseño de un **CI** digital a nanoescala

Diseño en la capa Sistema

Esta es la jerarquía más alta de diseño, el objetivo que se desea conseguir en esta etapa es diseñar un sistema que cumpla con las necesidades básicas de su aplicación. Las principales características en las que se especializa esta capa son: funcionalidad, desempeño, condiciones de trabajo, dimensiones físicas, empaquetado, pinout, tecnología de fabricación, costos, la potencia consumida y la necesidad de comunicación interna y externa.

Diseño en la capa Algoritmo

Esta capa busca definir una estructura en software que permita una implementación a harware funcional y sencilla. En esta etapa es importante considerar la capacidad de computo del **CI**, la cantidad de memoria que este requiere, se define la arqutectura del conjunto de instrucciones (**ISA**) por sus siglas en inglés [9]. En este punto debe evaluarse que tan complejo, que tantos recursos computacionales y que tanta exactitud se requiere según los objetivos establecidos en la capa superior como los protocolos de comunicación y las tareas que debe realizar.

Diseño en la capa Arquitectura

Así como la capa superior busca distribuir los recursos computacionales de forma eficiente, esta capa busca distribuir de forma eficiente los recursos físicos (hardware) para cada una de las tareas que se van a realizar. Aquí se debe de prestar atención a los objetivos del diseño en la capa de sistema como: desempeño, costo, potencia, dimensiones, etc... En este punto la operación de la máquina (**MO**), por sus siglas en inglés, toma lugar y se refiere a como el hardware implementa el **ISA** definido en la capa superior [9]. La definición de *datapaths*, tamaño de los buses, memorias de acceso aleatorio (**RAM**) y su relación con los demás componentes son las tareas de prioridad.

Diseño en la capa Lógica

En esta capa se busca generar una descripción lógica del circuito y hacer una síntesis de la misma para generar lo que se denomina *netlist*. Ambas tareas críticas para el diseño del **CI**

Ya con una arquitectura, que define todo el proceso de interacción del hardware, se requiere definir la conectividad de cada módulo. En el nivel de abstracción más alto de esta capa se traduce en *black boxes* las cuales contienen entradas, salidas y una sincronización temporal definida (reloj). La cual se puede describir utilizando un lenguaje descriptor de hardware o **HDL** por sus siglas en inglés. Los lenguajes más comunes para poder describir el harware son: Verilog y VHDL. Estos módulos deben ser simulados y verificados para asegurar su funcionalidad.

El siguiente paso es la traducción de estos bloques de **HDL** a elementos de bajo nivel de un circuito y como estos están interconectados entre sí. A el resultado de esta traducción se le conoce como *netlist*. Herramientas de síntesis lógica permiten automatizar estos procesos, lo que facilita esta tarea de traducir de **HDL** a un *netlist*.

Los *netlists* utilizan algo llamado *cell librarys* que contienen los componentes más elementales de un circuito. Los transistores son definidos y conforman subcircuitos que representan compuertas lógicas las cuales en conjunto con los elementos básicos como capacitores, resistencias, entre otros... son descritos tanto sus propiedades como su interconexión. Esto se puede simular y validar utilizando una herramienta de simulación *hspice*.

Diseño en la capa Física

A partir de este punto cada subcircuito descrito en el *netlist* es colocado en un espacio físico en una oblea de material semiconductor (Silicio) la cual está conformada de las diferentes difusiones, metales, pines y sus interconexiones. Debido a que el fabricante tiene requerimientos de diseño, que se deben cumplir para garantizar la funcionalidad del diseño, es imperativo que se tomen en cuenta y se verifiquen de forma rigurosa. Ya que este proceso toma en cuenta factores como: rendimiento, área, confiabilidad, potencia, entre otros... el proceso se divide en diferentes secciones: *Partitioning, FloorPlanning, Placement, Power and Ground Routing, Signal Routing y Closure* de las cuales se discutirán a mayor detalle. Al

final de todo este proceso un *netlist* asociado es generado el cual contiene todos los cambios y restricciones colocadas en los diferentes pasos de esta capa de síntesis física.

Diseño en la capa *Signoff* y Verificación Física

Una vez se culmina el diseño este debe de ser verificado de forma rigurosa para asegurar que este cumpla con los estándares de fabricación, que concuerde el diseño en silicio con el de el circuito propuesto, etc... Cualquier problema debe ser solucionado en la capa anterior de diseño. Las verificaciones que se realizan se mencionan a continuación:

Design Rule Checking (DRC)

Verifica que el *layout* cumpla con todos los requerimientos de fabricación los cuales están directamente relacionadas a la tecnología que se está trabajando. El realizar un proceso de manufactura a escala nanométrica es complejo debido a que se ve limitado por la tecnología de las herramientas de fabricación con las que cuenta la fábrica. Limitaciones como las distancias entre los metales, difusiones, policilicio, o complicaciones como la pérdida de resolución, las esquinas redondeadas, las conexiones reducidas durante el proceso de manufactura, etc... deben mitigares y esto se hace cumpliendo con los requerimientos de los fabricantes ya que ellos toman en cuenta estos problemas a la hora de fabricarlo. [10]

Layout vs. Schematic (LVS)

Verifica la funcionalidad del diseño. El *netlist* generado por la capa superior, síntesis física, y el *netlist* generado por la capa de síntesis lógica son comparados para verificar si ambos concuerdan en términos de funcionalidad.

Antenna Rule Checking

Esta fase sirve para prevenir los efectos de antena en el circuito. Los conductores al acumular cargas pueden presentar efectos de antena. Estos pueden acumularse debido a las descargas del rayo laser al momento del grabado, generando descargas no deseadas en las compuertas del transistor. Esto representa un problema ya que puede causar daños a los transistores, en especial a la compuerta del gate, o reducir su tiempo de vida.

Electrical Rule Checking (ERC)

Verifica la calidad de la conexión de tierra y alimentación así como los tiempos de transmisión de señales. Adicionalmente verifica cargas capacitivas y la correcta conexión de los *fanouts*. Una compuerta lógica puede excitar un número finito de entradas a otras compuertas en un mismo nodo (*fanout*). Esta excitación se ve limitada por la corriente que provee el circuito. Por lo tanto debe tomarse en cuenta para la frecuencia de operación y la potencia

entregada máxima de una compuerta. Factores que afectan tanto el rendimiento como el tiempo de vida del *CI*.

Parasitic Extraction (LPE)

Este deriva de los componentes geométricos, a nivel nanométrico, sus parámetros eléctricos (Capacitancias y Resistencias) para verificar las características eléctricas del circuito. A esta verificación se ingresa un *netlist* y con la ayuda de la herramienta de Synopsys *starRC* se genera un *hspice* con todas estas capacitancias y resistencias parásitas presentes en el *layout*.

Final Simulation

Luego de haber pasado todas las simulaciones en **CLEAN** o **PASS**, utilizando el *hspice* generado en la verificación de **LPE**, se hace una simulación que sirve para validar que los resultados obtenidos por el circuito descrito por el *layout* que se diseñó cumplan con las mismas funcionalidades que el circuito descrito en la capa de síntesis lógica. Si esto es congruente, entonces se puede concluir la etapa de verificaciones.

Diseño en la capa de Fabricación

Al finalizar todas las simulaciones, se genera un archivo de extensión **.GDS2** el cual contiene toda la información que el fabricante necesita para empezar el proceso de manufactura. Este archivo es enviado a una fábrica que se dedica a manufacturar chips en silicio. En el caso del proyecto, el fabricante es **TSMC**. El proceso en donde se manda el archivo a la fábrica se le conoce como *tapeout*. Aquí se generan las máscaras y por el proceso de fotolitografía, patrones expuestos a los rayos láser se graban en el silicio, así como los diferentes dopajes. El diseño se hace sobre obleas de silicio redondas las cuales varían en tamaño desde los 200mm hasta los 300mm. Por último se clasifican según su estado: funcional o defectuoso, establecido por las pruebas de potencia y velocidad que se establecieron.[10]

Diseño en la capa de Empaquetado y Pruebas

Aquí se cortan los **CI** individualmente de la oblea de silicio donde se fabricaron. A cada **CI** se le conoce como *die* y este es empaquetado según la necesidad del cliente. Los empaquetados pueden ser: **DIP, PGS, BGA, etc...** Se conectan los pines dentro del empaquetado con el *die* ya sea con *bonding wires o solder bumps* y se sella. Como punto final del flujo de diseño, este se prueba en un laboratorio por el cliente para ver que este funcionando de la forma deseada y con los requerimientos propuestos. Concluyendo el flujo de diseño del **CI**

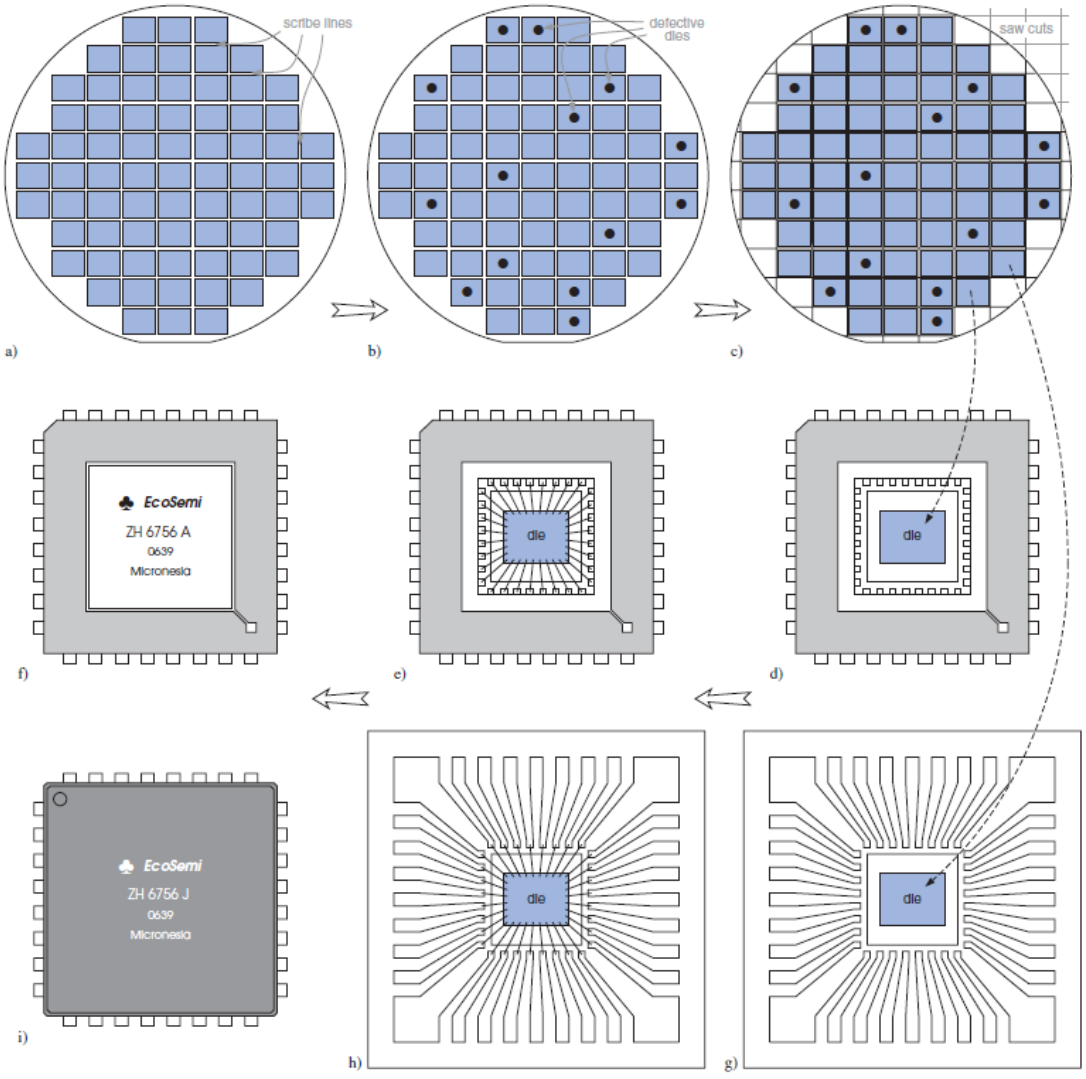

Fig. 11.11 Testing and encapsulation steps (simplified, not drawn to scale). Processed wafer (a), probed wafer with defective circuits inked (b), wafer after sawing (c), a good die attached to package cavity (d), wires bonded to lead frame (e), final IC after sealing, testing, and stamping (f). (g,h,i) correspond to (d,e,f) for a plastic package.

Figura 3: Segmentación de la oblea de silicio resultante en **die**, proceso de empaquetado, colocación de *bonding wires* y *CI* final [11]

Síntesis Física, *Design Rule Check (DRC), Electrical Rule Check (ERC) y Antena Rule Check*

Esta es la parte del flujo de diseño que se realizará este trabajo de investigación. Por lo tanto se desarrolla a más detalle la información de los procesos que se realizarán.

Síntesis Física

La síntesis física es una de las partes fundamentales para el desarrollo de este proyecto, ya que aquí se transformará la síntesis lógica a un conjunto de componentes físicos posicionados en silicio y unidos a través de *interconects*. Esta etapa de diseño está segmentada en diferentes secciones las cuales se describen a continuación:

Partitioning

En esta etapa se secciona el circuito en subcircuitos con el fin de que estos puedan luego ser posicionados y colocados de forma estratégica.

Floor Planning

En esta etapa se hace una relación espacial entre los pines de entras y salidas del **CI** y los subcircuitos. Esto se hace con la finalidad de establecer prioridades a las secciones del circuito original que deben estar más cerca de estos puertos ya que se desea minimizar retardos y material utilizado.

Placement

En esta etapa se plantean secciones del espacio que se va a utilizar para realizar el **CI**. En el caso de este proyecto se cuentan con $2mm^2$ de espacio para la fabricación del circuito deseado. Las partes seccionadas en la primera etapa se les asigna de forma estratégica una posición física en el silicio con el fin de obtener los mejores resultados en cuanto a la minimización de retardos y distancia entre las entradas y salidas de estos subcircuitos con los demás.

Power and Ground Routing

En esta etapa se hace un ruteo de los planos de alimentación y tierra. Esto se hace de primero ya que todo circuito lógico compuesto de **MOSFETs** tendrá la necesidad de estos, lo que hace crítico que estas señales se hagan primero.

Signal Routing

Se generan *interconects* entre todos los circuitos lógicos. El software utilizado permite optimizar estas trayectorias a manera de que utilicen menor material. Ya que entre más distancias y más material la cantidad de parásitos incrementará lo que impacta directamente con el rendimiento del **CI**.

Closure

Este es el paso antes de pasar a las verificaciones físicas, en esta fase se realizan las últimas revisiones del *layout*. Se puede realizar un **DRC** básico que proporciona la herramienta de IC Compiler II.

Proceso de Síntesis Física Anterior [12]

A continuación se muestra el proceso de la síntesis física realizada por el estudiante Luis Abadilla en el 2020. Los comandos que utilizó se describen y están colocados en el orden en el que fueron utilizados. Toda la síntesis física se realiza en la herramienta de IC Compiler I, y cuenta con cuatro partes fundamentales.

- **Importación de librerías y creación de *MilkyWay Library*:** En esta etapa se importan las librerías de **TSMC** a utilizar, se crean los archivos donde se documenta el historial de todas las acciones que se realizan en la síntesis física. Además se importan los archivos que provienen de la síntesis lógica. Adicionalmente se crea un directorio de trabajo y se llaman a las librerías link y target las cuales definen el caso típico.
- **Floor Plan**
  - **Conexiones Lógicas de alimentación y tierra:** Esta sección se dedica a crear las variables y las celdas que le corresponden a los puertos de alimentación.
  - **Creación de Pads y Corners:** Aquí se plantean las esquinas y los *pads*, llamados celdas, los cuales definen y acotan el contorno del **CI**
  - **Configuración de las restricciones:** Una de las partes fundamentales del Floor Plan, ya que aquí se restringe todas las posibles orientaciones, posiciones, conexiones o arreglos que puedan causar conflictos en el ruteo. Indispensable para realizar un *layout* de calidad.
  - **Creación del Floor Plan:** se ejecuta y verifica el Floor Plan propuesto por todas las restricciones y configuraciones previas.
- **Placement:** Aquí se colocan los componentes según las reglas de Floor Plan, además se pueden configurar arreglos que optimicen: *delays*, *fanouts*, distancias y *X-Talk*. También es verificable y reconfigurable.
- **Routing:** Por último se realizan todas las interconexiones, empezando por los puertos de alimentación y luego por las demás interconexiones. Aquí se realiza nuevamente una verificación de la efectividad del proceso y se concluye la síntesis física.

Comando	**Funcionalidad**
Comandos para la importación de librerías y creación de la *MilkyWay* y directorios necesarios para el proyecto	
./ icc_shell -gui - shared_license	Iniciar el programa y su interfaz gráfica en consola
lappend search_path / home / administrador / Escritorio /FA_TSMC_DRC / Libs / tcb018gbwp7t_290a _FE / tcb018gbwp7t /LM	Comando para definir el directorio en donde se encuentran las librerías que se van a utilizar
set link_library " tcb018gbwp7ttc .db tcb018gbwp7twc .dbtcb018gbwp7tbc .db tpd018nvtc .db "*	Hacer mención directa de las librerías que se van a utilizar de todo el catálogo de **TSMC** (link)
set target_library "tcb018gbwp7ttc .db "	Hacer mención directa de la librería (target) con la que se estará trabajando: caso típico
set_tlu_plus_files -max_tluplus / home / administrador / Escritorio/ FA_TSMC_DRC / Libs / tcb018gbwp7t_290a_FE / tluplus / t018lo_1p6m_typical . tluplus - min_tluplus / home / administrador / Escritorio / FA_TSMC_DRC / Libs / tcb018gbwp7t_290a_FE / tluplus / t018lo_1p6m_typical . tluplus - tech2itf_map / home / administrador / Escritorio / FA_TSMC_DRC / Libs / tcb018gbwp7t_290a_FE / tluplus / star . map_6M	Comando para agregar la TUPLUS y el archivo .MAP específico dentro de las librerías de **TSMC**
create_mw_lib -technology / home / administrador / Escritorio / FA_TSMC_DRC / Libs / tcb018gbwp7t_290a_FE /tf/ tsmc018_6lm .tf - mw_reference_library {/ home / administrador / Escritorio / FA_TSMC_DRC / Libs / tcb018gbwp7t_290a_FE / tcb018gbwp7t / home / administrador / Escritorio / FA_TSMC_DRC / Libs / iolib / tpd018nv } - bus_naming_style {[% d]} -open / home / administrador / Escritorio / LUIS_ABADIA / Milky_Aba	Crear un directorio y una librería *MilkyWay* para la documentación de todos los cambios que se harán dentro de la síntesis física.
Hasta este punto se importan los archivos de extensión verilog (.v) y extensión (.sdc) resultantes de la síntesis física.	
Floor Plan Parte 1: Conexiones lógicas de VDD y VSS	
set_app_var mw_logic1_net "VDD"	Definición de una variable para la net de poder del voltaje

set_app_var mw_logic0_net "VSS"	Definición de una variable para la net de poder de tierra
derive_pg_connections -power_net VDD -power_pin VSS -ground_net VSS -ground_pin VSS	Configuración de los pines de poder y las nets de poder según las variables que se definieron en los comandos anteriores.
derive_pg_connections -power_net VDD -ground_net VSS -tie	
report_cell_physical -connections	Verificación de que los comandos anteriores se ejecutaron con éxito. Si el resultado es 1 es exitoso, si es 0 es fallido.
Floor Plan Parte 2: Creación de celdas (Pads y Corners)	
create_cell {C1 C2 C3 C4} PCorner	Creación de las 4 esquinas del chip.
create_cell { VDD} PVDD1CDG	Creación de la conexión de VDD (Pad)
create_cell { VDD} PVSS1CDG	Creación de la conexión de VSS (Pad)
Floor Plan Parte 3: Configuración de las restricciones	
set_pad_physical_constraints -pad_name ç1side 1	Configuración del primer pad de esquina en el lado 1
set_pad_physical_constraints -pad_name ç2side 2	Configuración del segundo pad de esquina en el lado 2
set_pad_physical_constraints -pad_name ç3side 3	Configuración del tercer pad de esquina en el lado 3
set_pad_physical_constraints -pad_name ç4side 4	Configuración del cuarto pad de esquina en el lado 4
set_pad_physical_constraints -pad_name "VDDside 3 -order 2	Configuración del pad de VDD en el lado 3 en segunda posición
set_pad_physical_constraints -pad_name "VSSside 4 -order 2	Configuración del pad de VSS en el lado 4 en segunda posición
Floor Plan Parte 4: Creación del Floor Plan	
create_floorplan - control_type width_and_height - core_utilization 0.7 - core_width 5 -core_height 50 - no_double_back - top_io2core 10 - bottom_io2core 10 - left_io2core 5 - right_io2core 5	Creación de la segmentación del espacio a utilizar por los componentes (Floor Plan)
check_physical_design	Verificación del proceso de diseño hasta este punto. Si el resultado es 1 es correcto, si es 0 es incorrecto y se debe corregir.
adjust_fp_floorplan	Este comando termina de hacer los ajustes necesarios para asegurar que este esté bien acotado.
Configuración y creación del placemnet	
set_fp_placement_strategy - adjust_shapes on	Ajuste de las geometrías dentro del Floor Plan

create_fp_placement -effort High -max_fanout 800 - optimize_pins - congestion_driven - timing_driven - consider_scan	Comando para realizar el posicionamiento , con mejor esfuerzo, un gran *fanout*, optimizando la congestión, el *delay* y la posición de los pines de entradas y salidas con respecto a los *pads*.
check_physical_design - post_initial_placement	Este comando permite realizar una verificación del placement si el resultado es 0 esta mal y si es 1 esta correcto.
place_opt -effort high	Optimizar el placement al final para tener mejores resultados maximizando el esfuerzo para que se cumplan las condiciones previamente establecidas.
Creación del Routing	
create_rectangular_rings -nets {VDD VSS } -around core	Creación del anillo de alimentación al rededor de todos los componentes para hacer el routing más eficiente.
create_power_straps -direction vertical -start_at 155 - num_placement_strap 4 - increment_x_or_y 50 -nets {VDD} -layer METAL2 -width 2 - do_not_route_over_macros	Este comando crea también un fácil acceso a los planos de alimentación y tierra. Los *power straps* dependen de la cantidad de componentes y saturación del espacio.
set_route_mode_options -zroute false	Para cambiar el comando clásico se utiliza este comando.
check_routeability	Este preceso sirve para hacer un análisis preliminar para saber si la herramienta es capaz de rutear el circuito propuesto. De ser cierto el resultado es 1, si es 0 se debe cambiar o la configuración o el placement.
set_route_opt_strategy	Este comando permite analizar si el circuito tiene problemas de delay, de X-talk o errores generales de diseño.
set_preroute_drc_strategy	En este punto se hace un análisis de DRC muy básico, esto es solamente como una pre revisión, antes del análisis DRC por IC Validator.
preroute_standard_cells -mode net -connect both -nets { VDD}	Comando para conectar networks con el anillo de VDD.
190 preroute_standard_cells -mode net -connect both -nets { VSS}	Comando para conectar networks con el anillo de VSS.
preroute_instances	Coando para conectar todas las celdas con los anillos.
route_opt	Conexión entre celdas y fuentes de alimentación.

verify_route	La última verificación de routing se hace aquí, si el valor es 0 debe de verificarse nuevamente y probar con nuevas estrategias de ruteo, si es 1 todo está en orden y se puede proceder a DRC.
Finaliza la Síntesis Física.	

Cuadro 1: Proceso de síntesis física realizado en el 2020, comandos y su descripción.

DRC

Design Rule Check, esta etapa es crítica y juega un papel fundamental en conjunto con la síntesis física. Aquí se comparan los resultados obtenidos de la síntesis física con un *runset*, que en nuestro caso es proporcionado por **TSMC**, el cual contiene todas las restricciones de fabricación. Aquí se hace una exhaustiva revisión de las normas como el espaciado de metales, las interconexiones, el tamaño de los *pads*, etc... se cumplan y concuerden con todas las normas de **TSMC**. Esta etapa no trata acerca de la funcionalidad ni la coherencia entre el diseño lógico y físico que es tarea del **LVS**. Cualquier problema resultante de esta verificación debe ser gestionado por la capa de diseño superior (síntesis física) y de forma iterativa solucionar cualquier problema que se presente hasta obtener un resultado **CLEAN**. [6]

Para esta etapa se pueden presentar una gran cantidad de errores y no existe un orden específico, ni un formato definido para presentarlos. Es por esto que se debe de contar con la documentación del software que se estará utilizando para la realización de este proceso de validación. En el caso de Synopsys, la herramienta que ofrecen es IC Validator. Debido a que existen demasiadas reglas de diseño a continuación se mencionan algunas reglas de diseño comunes. [10]

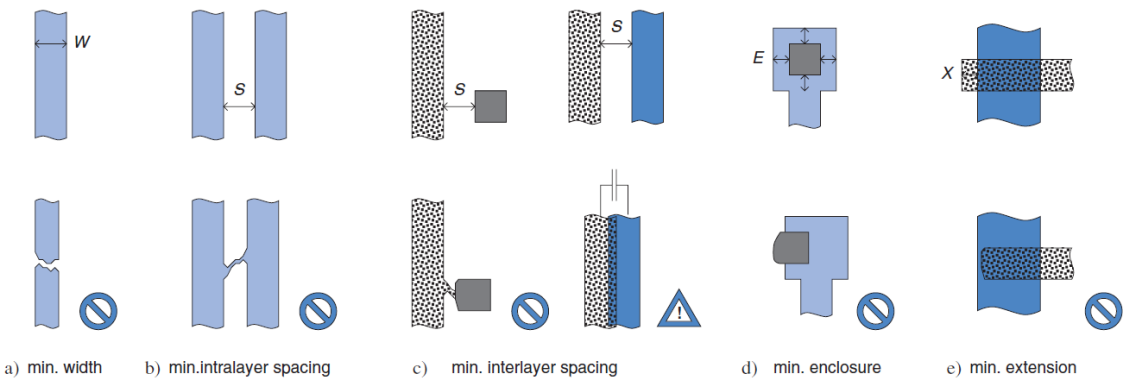

Fig. 11.1 Minimum size rules (top row) and the likely consequences of violating them (bottom row).

Figura 4: Ilustración de las reglas de diseño de condiciones mínimas y como estas se infringen [11]

Minimum Width

Se establece un grosor mínimo de las estructuras para evitar que estas colapsen o se abra el circuito por falta de material. Particularmente las lineas largas y delgadas, sin elementos en sus vecindades, son más susceptibles a sobre-estirarse a diferencia que un conjunto de lineas del mismo grosor. Es por esto que se necesita un mayor grosor.[11]

Minimum Intralayer Spacing

Esta regla limita la distancia entre las diferentes estructuras. Esto se hace para evitar cortos circuitos entre los elementos adyacentes, También puede depender si estas estructuras tienen una conexión eléctrica existente. Por ejemplo, en un proceso de 130nm exige una separación de *Well-to-Well* de 630nm si estos manejan el mismo potencial. De no ser así este es de 1000nm. [11]

Minimum Interlayer Spacing

Esta regla permite verificar la distancia entre un plano y otro, para evitar acoplamientos e interacciones indeseadas entre sí. Problemas como la inflación de capacitancias por contactos entre polisilicio y difusiones o corrientes de fuga por acoplamientos ente uniones PN y una *gate* de policilicio. [11]

Minimum Enclosure

Estas reglas están directamente relacionadas con las estructuras del *layout* en diferentes planos. Estas reglas son uniformes a lo largo de todos los bordes. Por ejemplo para las vias que el contacto sea el adecuado con las superficies que conecta ya que el proceso de manufactura tiene cierta tolerancia en las máscaras y su alineación además de las imperfecciones del proceso de *etching*. Otro ejemplo es que el metal de más alto nivel este debajo de *overglass passivation layer* a lo largo de todas las aperturas para los *pads* así el sellado es hermético. [11]

Minimum Extension

Debido a que existen múltiples capas es importante mantener reglas que delimiten como se sobreponen las estructuras. La forma en la que esto se verifica es por medio de la orientación en la que se sobreponen. El *self-aligned perocess* permite que el policilicio actúe como una máscara para el proceso de implantación de iones [10]. Por lo que si estas capas de policilicio no están correctamente alineadas es posible que ya sea el *drain* o el *source* no se dopen de forma correcta. Esto afectaría considerablemente el desempeño y la funcionalidad del **CI**. Estas reglas ya toman en cuenta las imperfecciones del proceso de *etching* y problemas en la alineación de las máscaras. [11]

Maximum Width

Las vías y en general los contactos se fabrican por medio del proceso de *etching* en la capa de dieléctrico para luego depositar rellenos de tungsteno para llenar las cavidades, antes de depositar la siguiente capa metálica. Para asegurar una calidad uniforme y una superficie plana es necesario delimitar el grosor máximo de estas cavidades. Lo más común es delimitar un único tamaño de vía o contacto. Si en algún caso se desea agregar un contacto

más grande, estas deben segmentares en subgrupos de contactos. A esta técnica se le conoce como *stipple contact/via.*[11]

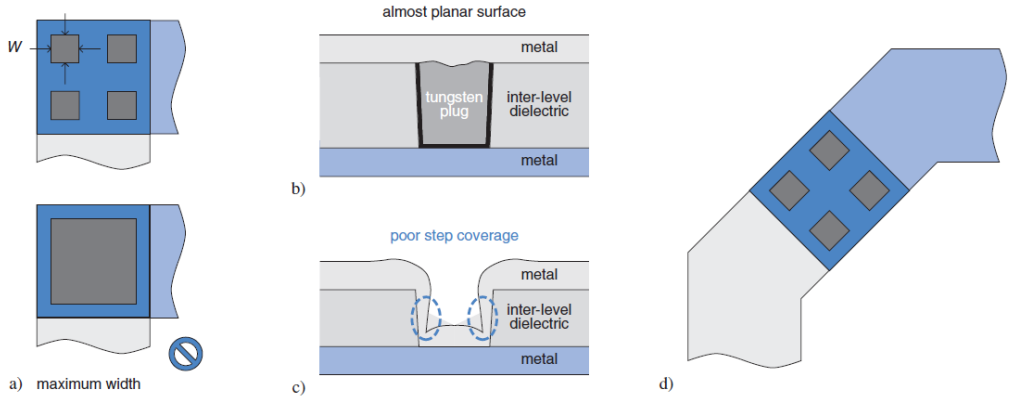

Fig. 11.2 Maximum size rule for contacts and vias. Stipple contact/via versus an oversize single contact/via (a). Cross section of a plugged via (b) and of a historical sink-in via (c). Electromigration-aware stipple contact/via (d), to be explained in section 11.6.1.

Figura 5: Ilustración de la regla de *Maximum Width*, como esta afecta el diseño y la técnica de *stipple contact/via* [11]

Density Rules

La densidad planar de una capa de un layout es definida como el área ocupada por todos los elementos en esa capa, dividido el área de toda la capa. Estas normas permiten limitar la ocupación del plano con cota inferior y superior. Por ejemplo se puede definir que la ocupación esté en promedio entre el 20 % y el 80 % por cada milímetro por milímetro de región. Algunas prácticas como llenar de *dummys*, sin conectar, una región en donde la ocupación no se llegue a cumplir resultan útiles en el proceso de diseño.[11]

Extension Rules

Si bien estas reglas están mejor definidas en el proceso de *Antenna Rule Checking*, el proceso de **DRC** hace una verificación únicamente de la extensión del material conductor. Estas reglas se detallaran con mayor detenimiento en la verificación de reglas de antena.[11]

Verificación de DRC Anterior

En la iteración previa a este proyecto, la cual fue dirigida y ejecutada por Matthias Sibrian y precedida por Luis Nájera el año anterior, cuenta con una metodología específica para realizar el proceso de verificación DRC. El proceso de ejecución de esta verificación esta

segmentado en cuatro pilares fundamentales: Instalar y abrir Custom Compiler, Cargar la celda sintetizada, abrir la vista de *layout* y correr la verificación DRC.

La parte fundamental de este proceso, para el nivel al que se trabajará este proyecto, se centra principiante en la ejecución de la verificación de las reglas de diseño, comprensión, corrección de errores y revalidación. Es un proceso iterativo que requiere consultar la documentación de TSMC y sus restricciones de diseño. Esta directamente enlazado con el proceso de síntesis física ya que en este se corrigen los errores que esta verificación presenta. A continuación se muestra un diagrama que muestra el proceso de corrección de errores de DRC.

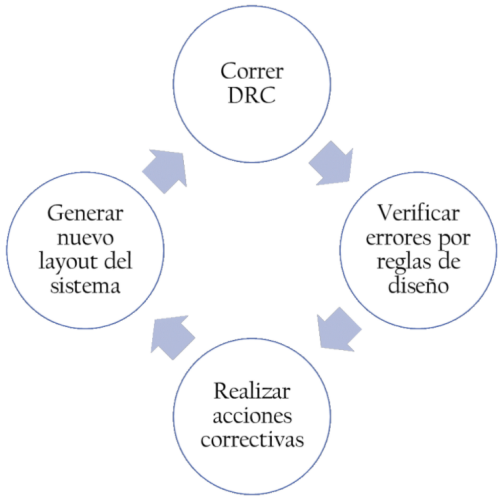

Figura 6: Imagen del proceso de verificación DRC de años anteriores [13]

Debido a problemas con la compatibilidad de IC Validator con IC Compiler I los grupos anteriores se vieron obligados a utilizar Custom Compiler como la herramienta de verificación y su integración con IC Validator. Es por esto que el proceso mostrado debe ser abordado de forma manual, utilizando una interfaz gráfica. Uno de los desafíos que presenta el actual proyecto es utilizar IC Compiler II para poder utilizar IC Validator en consola y abrir paso a una verdadera automatización del flujo de diseño.

ERC [5]

Electrical Rule Check es una etapa de la capa de *singoff* la cual permite verificar que el rendimiento eléctrico del circuito cumpla con las especificaciones del fabricante. Las reglas que verifica esta etapa son las siguientes:

Soft Connect Check

Esta regla establece si la conectividad (unidireccional) de las difusiones que se encuentran en el *N-Well* y la capa superior es la especificada.

Path Check

Esta regla valida lo siguiente: Nodos con un *path* conectado a alimentación pero no tierra, Nodos con un *path* a tierra pero no a alimentación, Nodos sin *path* a tierra o alimentación o Nodos sin un *path* a cualquier otra *net*.

PTAP/NTAP Connectivity Check

Verifica si el *PTAP* se conecta a alimentación y el *NTAP* a tierra.

MOSFET Poer and Ground Check

Verifica que para cualquier **MOSFET**, ya sea tipo N o P, *source* **MOSFET** y *drain* **MOSFET** estén conectados correspondientemente a alimentación y a tierra.

Gate Directly Connected to Power or Ground

Esta regla sirve para verificar que exista protección contra descargas electro-estáticas. Si el *gate* de un **MOSFET** tipo P se conecta a alimentación de forma directa podría dañarse el *gate*, así como un **MOSFET** tipo N conectado a tierra directamente.

Floating Gate

Verifica si los contactos están interfiriendo con el *poly gate*

Floating Well

Verifica si existe alguna conexión entre el *N-Well* o *P-Subrate* y alimentación o tierra.

Antenna Rule Checking [5]

Esta verificación busca evitar que el **CI** tenga efectos de antena que pueden dañar las terminales de los transistores (*gate, drain, source o bulk*) durante el proceso de fabricación. Esto puede ocurrir en el proceso de grabado de los caminos, por medio de rayos plasma o en la implantación de iones. Estos rayos pueden hacer que se acumulen cargas no deseadas en

los cables metálicos, los cuales pueden estar conectadas a nodos de uniones PN. Algunos de los efectos presentes pueden ser: daño de oxido inducido por plasma (efecto antena), también conocido como *Fowler-Nordheim tunneling*, e inducción de corriente por altas temperaturas.[11] Efectos que pueden quemar o reducir el tiempo de vida de un transistor afectando el rendimiento y tiempo de vida del **IC** completo.

Esta verificación al igual que el **DRC** cuenta con un *runset* específico y utiliza el software de IC Compiler II. Las reglas de antena se centran en especificar el área máxima de metal que se puede conectar a un gate sin el source o drain para actuar como elemento de descarga. Las relaciones entre gate y los demás metales son: 100:1 hasta 5000:1 para que la carga sobre el metal no dañe la compuerta. Algunas formas de solucionar estos problemas pueden ser: seccionar el metal en conexiones de metales más pequeños para reducir la potencia del rayo plasma durante la fabricación o la colocación de un diodo entre el metal y tierra para evitar la conducción de corriente y que esta pueda causar una descarga no deseada por las altas temperaturas.

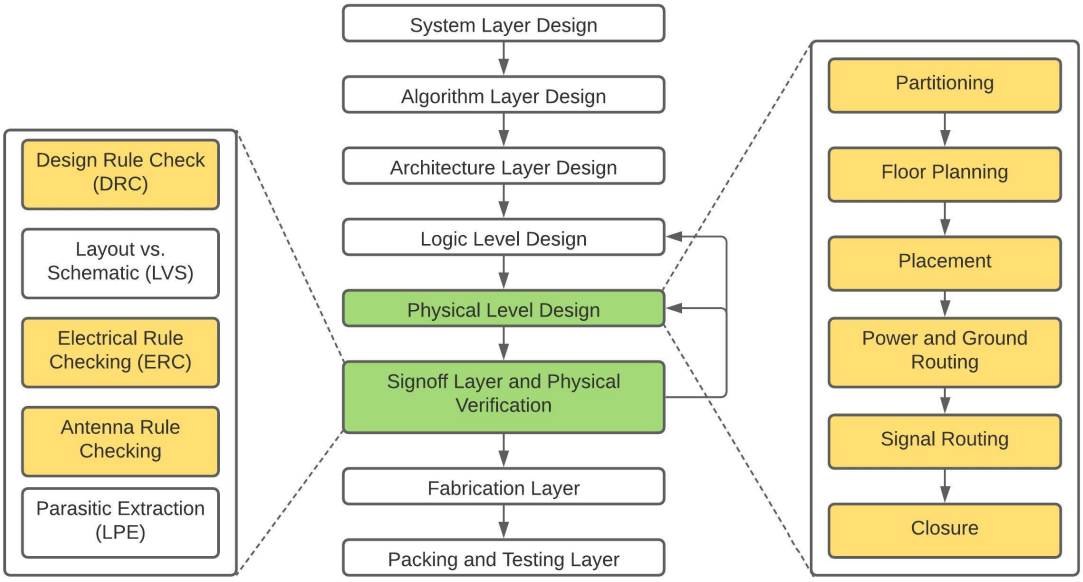

Figura 7: Flujo de diseño que ilustra las partes específicas a realizar de la síntesis física y la etapa de *Signoff* en amarillo

Electronic Design Automation (EDA)

La industria **EDA** existe con el fin de proporcionar herramientas de software para diseñadores electrónicos, para automatizar los procesos de diseño y fabricación de **CIs**. Esto permite que los precios para la fabricación de estos dispositivos disminuya, además de la especialización en cada una de las herramientas de software y separación del proceso de fabricación con el de diseño. Esto genera equipos de trabajo especializados en el área de diseño de **CIs** y por otro lado las compañías que fabrican dichos dispositivos pueden proporcionar

las restricciones y normas de diseño con las que pueden fabricar los dispositivos. Si bien ambas disciplinas se separan estas convergen con las herramientas **EDA**, siendo el puente en entre los diseñadores y la industria de semiconductores, reduciendo costos, generando equipos especializados y creando una nueva industria en el mundo.[14]

Actualmente las dos compañías más grandes en la industria **EDA** son Cadence y Synopsys. Para fines de esta investigación se estarán utilizando las herramientas **EDA** de Synopsys. Esta cuenta con todas las herramientas para completar el flujo de diseño propuesto en la Figura 3 hasta la etapa de *Signoff and Physical Verification*. Esto se debe a que Synopsys no es una empresa que se dedica a la fabricación de **CIs**. Además como se mencionó anteriormente este es el puente entre el fabricante y el diseñador, por lo que las restricciones de diseño y fabricación han sido proporcionadas por **TSMC** la empresa a la que se mandará a hacer el **CI**, en donde se podrán concluir las últimas dos etapas del flujo de diseño y obtener el **CI** físico.[15]

Synopsys

Synopsys es una herramienta de diseño y simulación para cada una de las diferentes etapas del flujo de diseño. Estas herramientas son proporcionadas por el departamento de Ing. Electrónica, Mecatrónica y Biomédica de la UVG. Las cuales se aprenden a utilizar en los cursos de Nano Electrónica 1 y 2 son: HSpice, Wave Viewer, IC Compiler, IC Compiler II, Custom Compiler, Verdi3, VCS, Design Vision, Nanotime, IC Validator, StarRC, Formality, Milky Way, entre otras. En los siguientes apartados se desarrolla acerca de las herramientas que son de interés para el desarrollo del proyecto de graduación en lo que serán las etapas de Síntesis Física y en la etapa de *Signoff* y la Verificación Física: **DRC**, **ERC** y *Anttenna Rule Checking* [1]

IC Compiler II

La sucesora a la herramienta IC Compiler I, es una de las herramientas líder para lo que es Place and Route. Esta herramienta es esencial para la síntesis física. En esta herramienta permite procesar el *verilog* proveniente de la síntesis lógica para convertirlo en un *layout* que represente el circuito físico en silicio, el cual será luego verificado.[16]

IC Validator

IC Validator es una de las herramientas de Synopsys para la etapa de *Signoff* y Verificación física. Este software permite realizar **DRC**, *Antenna* y **ERC**, verificaciones necesarias para asegurar que el *layout* generado en la etapa anterior pueda ser fabricado.[17]

CAPÍTULO 7

Migración de la Síntesis Física a IC Compiler II

Anteriormente el proceso de síntesis física se realizó en la herramienta de Synopsys IC Compiler I (**ICCI**). Sin embargo, debido a las recomendaciones de los proyectos anteriores y programas actuales que están dejando de utilizarlo de forma progresiva se realizó una migración completa de esta etapa a IC Compiler II (**ICCII**). Además, durante el proceso de migración, se aprendieron nuevas técnicas que permitieron mejorar el flujo de diseño anterior.

7.1. Creación de Librerías Estándar *New Data Model* (NDM)

La primera diferencia significativa entre el proceso de síntesis física anterior y el actual es que la herramienta **ICCII** trabaja con librerías denominadas *New Data Model* o por sus siglas en inglés (**NDM**) y su extensión es *.ndm*. A diferencia de **ICCI** que utiliza la librería *MilkyWay* extensión *.mw*.

Las librerías **NDM** al igual que las *MilkyWay* son una colección de librarías que contienen toda la información de las celdas que se estarán usando dentro de un diseño, esta puede ser tanto una colección para tener todos los archivos de referencia como una librería que contenga toda la síntesis física. Nuestro *foundary*: **TSMC** nos brinda tres tipos de archivos que se utilizarán a lo largo de este proyecto para lo que es la librería de referencia. Estos tes archivos son: *Technology File*, *DB-Technology File* y *Library Exchange Format File*.

7.1.1. *Technology File*

El primer requerimiento para poder empezar a trabajar dentro de la síntesis física es el *Technology File*, también conocido como *techfile* en el entorno de Synopsys, es un archivo

de extensión *.tf* que contiene toda la información referente a las reglas específicas que debe seguir el diseño según la tecnología que se está trabajando. Este archivo contiene información como: tamaño de una celda estándar, tamaño de la tecnología que se está trabajando, colores para las diferentes capas de metal y características de los diseños, dimensiones de los metales, extensiones máximas, mínimas, entre otras... En esta investigación se utiliza un *techfile* proporcionado por **TSMC** para una tecnología de 180nm y 6 *layer map* que se refiere a las capas de metal que se están utilizando.

Nombre del archivo:

- ***tsmc018_6lm.tf***

Directorio:

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

7.1.2. *DB-Technology File*

Tanto las celdas estándar como los *pads* de entradas y salidas cuentan con este tipo de archivo de extensión *.db*. Este archivo es una versión compilada de otro tipo de archivo que también proporciona **TSMC** denominado *Liberty File* extensión *.lib*. Estos dos archivos análogos contienen la información lógica de las celdas y *pads* que se van a estar utilizando en la síntesis física. Algunas de estas características lógicas son: timing, área, potencia, voltaje, entre otros...

Nombre de los archivos para celdas estándar:

- ***tcb018gbwp7tbc.db***
- ***tcb018gbwp7tlt.db***
- ***tcb018gbwp7tml.db***
- ***tcb018gbwp7ttc.db***
- ***tcb018gbwp7twc.db***
- ***tcb018gbwp7twcl.db***

Directorio para las celdas estándar:

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

Nombre del archivo para los *pads* de entradas y salidas:

- ***tpd018nvtc.db***

Directorio para los *pads* de entradas y salidas:

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

7.1.3. *Library Exchange Format File*

Por el momento se han mencionado únicamente las características lógicas de las celdas y *pads*, sin embargo la representación física de estos es igual de indispensable para la síntesis física. Estos archivos de extensión *.lef* son los que contienen todas las representaciones de las celdas y *pads* en silicio, así como algunas reglas e información de las celdas. Este archivo por ser de **TSMC** únicamente nos permite observar la simplificación de caja negra de las celdas y sus pines en metal 1, así como algún *Routing Blockage* o **RB**. La descripción es legible ya que está en texto plano. A continuación se especifica el archivo y el directorio que se usó para este proyecto:

Nombre del archivo para las celdas estándar:

- ***tcb018gbwp7t_6lm.lef***

Directorio para las celdas estándar:

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

Nombre del archivo para los *pads* de entradas y salidas:

- ***tpd018nv_6lm.lef***

Directorio para los *pads* de entradas y salidas:

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

7.1.4. *ICCII Library Manager*

Debido a que los trabajos de años anteriores se basan en librerías *MilkyWay* el proyecto se vio obligado a cambiar desde el inicio. Explorando en la documentación de Synopsys se encontró una interfaz auxiliar de **ICCII** la cual se llama *Library Manager*. Para poder invocar a esta herramienta se utiliza el comando: ***icc2_lm_shell -gui***. Esta herramienta sirve para crear librerías *.ndm* a través de diferentes flujos. El flujo depende de los archivos

con los que se cuente y las celdas que uno desee generar en las librerías. Sin embargo todas las librerías siguen la misma metodología para generar una **NDM**. Este proceso se describe a continuación:

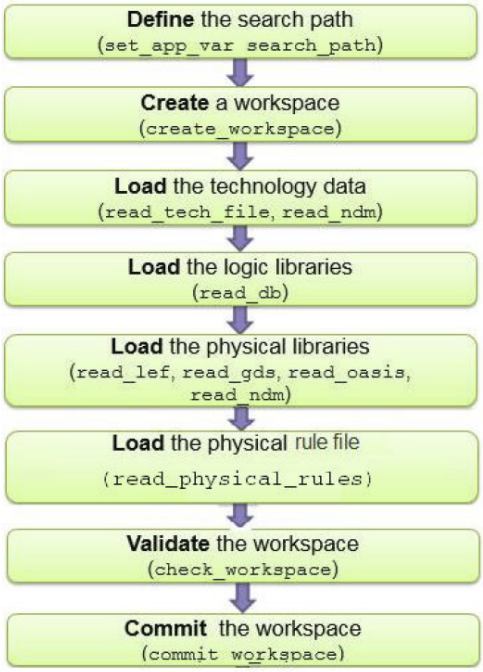

Figura 8: Proceso para crear una librería **NDM** y sus comandos respectivos extraído de: [18]

Para generar la librería de referencia del proyecto se utilizaron 3 flujos distintos: *Normal Flow, Physical Only Flow* y *Aggregate Flow*. Todos estos flujos requieren de un *technology file*, que para fines de este proyecto es el mismo para todas las celdas ya que se está trabajando con la misma tecnología de 180nm. A continuación se describe cada uno de los flujos que se implementaron y los comandos necesarios para ejecutarlos.

Normal Flow

Este flujo permite crear una librería **NDM** que cuente con: *tech file, db file* y *lef file*. Las celdas generadas en esta **NDM** serán exclusivamente aquellas que su información física tenga asociada una información lógica. En otras palabras que tanto en el *.db* como en el *.lef* exista la misma celda describiendo la información lógica y física respectivamente. Por lo tanto estas celdas contienen tanto información del *layout* como información **PVT**. Si en algún caso no existe una pareja, es decir hay celdas sin alguna de las dos descripciones, estas son expulsadas de la *.ndm*. Estas librerías son útiles para la creación de *pads* de entradas y salidas como las celdas estándar con fines de fabricación y simulación. A continuación la ejecución de las librerías utilizadas en el proyecto en el *shell* del *library manager*.

Ejemplo de un *Normal Flow* para la creación de una **NDM** de celdas estándar:

1. Creamos un *workpace* de flujo normal:

```
> create_workspace -flow normal -technology usr/synopsys/TSMC
/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/TSMCHOME/digital/
Back_End/milkyway/tcb018gbwp7t_270a/techfiles/tsmc018_6lm.tf
NormalWorkspace
```

2. Importamos los *.db files* donde están las celdas estándar:

```
> read_db { usr/synopsys/TSMC/180/CMOS/G/stclib/7-track/
tcb018gbwp7t_290a_FE/TSMCHOME/digital/Front_End/timing_power_noise/
NLDM/tcb018gbwp7t_270a/tcb018gbwp7tbc.db usr/synopsys/TSMC/180/CMOS/
G/stclib/7-track/tcb018gbwp7t_290a_FE/TSMCHOME/digital/Front_End/
timing_power_noise/NLDM/tcb018gbwp7t_270a/tcb018gbwp7tlt.db usr/
synopsys/TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/
TSMCHOME/digital/Front_End/timing_power_noise/NLDM/tcb018gbwp7tml.db
 usr/synopsys/TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/
TSMCHOME/digital/Front_End/timing_power_noise/NLDM/tcb018gbwp7ttc.db
 usr/synopsys/TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/
TSMCHOME/digital/Front_End/timing_power_noise/NLDM/tcb018gbwp7twc.db
 usr/synopsys/TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/
TSMCHOME/digital/Front_End/timing_power_noise/NLDM/tcb018gbwp7twcl.
db }
```

3. Importamos el *.lef file* de las celdas estándar:

```
> read_lef /usr/synopsys/TSMC/180/CMOS/G/stclib/7-track/
tcb018gbwp7t_290a_FE/TSMCHOME/digital/Back_End/lef/tcb018gbwp7t_270a
/lef/tcb018gbwp7t_6lm.lef
```

4. Hacemos el *check* del *workspace* y desplegamos la ventana de errores para verificar que todo esté en orden:

```
> current_workspace; check_workspace
```

```
> gui_create_window -type MessageBrowserWindow
```

```
> open_ems_database check_workspace.ems
```

5. Hacemos *commit* y *save* a la librería *.ndm* con el nombre **StandardWorkspace.ndm**

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

Ejemplo de un *Normal Flow* para la creación de una **NDM** de *pads* de entradas y salidas:

1. Creamos un *workpace* de flujo normal:

```
> create_workspace -flow normal -technology usr/synopsys/TSMC
  /180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/TSMCHOME/digital/
  Back_End/milkyway/tcb018gbwp7t_270a/techfiles/tsmc018_6lm.tf
  NormalWorkspace
```

2. Importamos los *.db files* donde están los *pads* de entradas y salidas:

```
> read_db { TSMC/180/CMOS/G/IO3.3V/iolib/LINEAR/tpd018nv_280a_FE/
  TSMCHOME/digital/Front_End/timing_power_noise/NLDM/tpd018nv_280a/
  tpd018nvtc.db }
```

3. Importamos el *.lef file* donde están los *pads* de entradas y salidas:

```
> read_lef TSMC/180/CMOS/G/IO3.3V/iolib/LINEAR/tpd018nv_280a_FE/
  TSMCHOME/digital/Back_End/lef/tpd018nv_280a/mt_2/6lm/lef/
  tpd018nv_6lm.lef
```

4. Hacemos el *check* del *workspace* y desplegamos la ventana de errores para verificar que todo esté en orden:

```
> current_workspace; check_workspace
```

```
> gui_create_window -type MessageBrowserWindow
```

```
> open_ems_database check_workspace.ems
```

5. Hacemos *commit* y *save* a la librería *.ndm* con el nombre **PadsWorkspace.ndm**

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

Physical Only Flow

Este flujo, a diferencia del *Normal Flow*, escoge únicamente a aquellas celdas cuya información se encuentre exclusivamente en el archivo *.lef*, es decir solamente toma en cuenta la información física de la celda, como su nombre lo indica. Estas librerías se crearon ya que hay ciertas celdas que son indispensables para la creación del chip como las esquinas y los *non-metal fillers* y que no cuentan con información **PVT** por parte del *foundary*. Estas librerías contienen celdas cuya funcionalidad se ve en el proceso de fabricación, pero no en el proceso de simulación. A continuación se describen las librerías creadas para el proyecto.

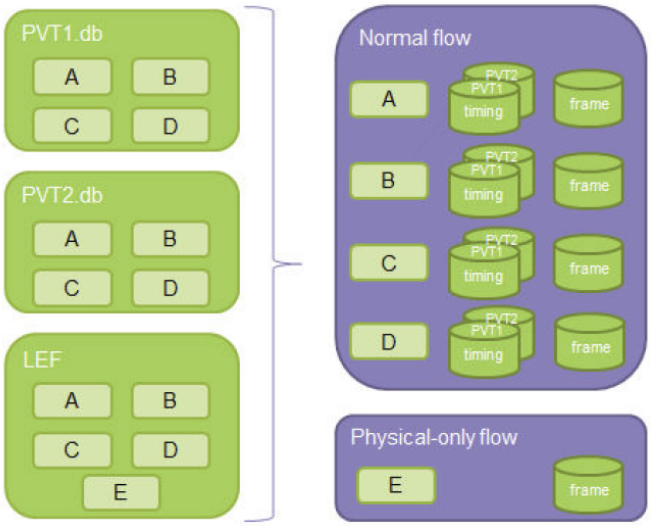

Figura 9: Diagrama que muestra como se crean las diferentes **NDM** según la información y el flujo utilizado, extraído de: [18]

Ejemplo de un *Physical Only Flow* para la creación de una **NDM** de celdas estándar que contengan los *non-metal fillers*:

1. Creamos un *workpace* de flujo únicamente físico:

```
> create_workspace -flow physical_only -technology usr/synopsys/
TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/TSMCHOME/digital
/Back_End/milkyway/tcb018gbwp7t_270a/techfiles/tsmc018_6lm.tf
PhysicalOnlyWorkspace
```

2. Importamos los *.db files* donde están las celdas estándar:

```
> read_db { usr/synopsys/TSMC/180/CMOS/G/stclib/7-track/
tcb018gbwp7t_290a_FE/TSMCHOME/digital/Front_End/timing_power_noise/
NLDM/tcb018gbwp7t_270a/tcb018gbwp7tbc.db usr/synopsys/TSMC/180/CMOS/
G/stclib/7-track/tcb018gbwp7t_290a_FE/TSMCHOME/digital/Front_End/
timing_power_noise/NLDM/tcb018gbwp7t_270a/tcb018gbwp7tlt.db usr/
```

```
synopsys/TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/
TSMCHOME/digital/Front_End/timing_power_noise/NLDM/tcb018gbwp7tml.db
 usr/synopsys/TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/
TSMCHOME/digital/Front_End/timing_power_noise/NLDM/tcb018gbwp7ttc.db
 usr/synopsys/TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/
TSMCHOME/digital/Front_End/timing_power_noise/NLDM/tcb018gbwp7twc.db
 usr/synopsys/TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/
TSMCHOME/digital/Front_End/timing_power_noise/NLDM/tcb018gbwp7twcl.
db }
```

3. Importamos el *.lef file* de las celdas estándar:

```
>       read_lef /usr/synopsys/TSMC/180/CMOS/G/stclib/7-track/
tcb018gbwp7t_290a_FE/TSMCHOME/digital/Back_End/lef/tcb018gbwp7t_270a
/lef/tcb018gbwp7t_6lm.lef
```

4. Hacemos el *check* del *workspace* y desplegamos la ventana de errores para verificar que todo esté en orden:

```
>       current_workspace; check_workspace
```

```
>       gui_create_window -type MessageBrowserWindow
```

```
>       open_ems_database check_workspace.ems
```

5. Hacemos *commit* y *save* a la librería *.ndm* con el nombre **FillersWorkspace.ndm**

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

Ejemplo de un *Physical Only Flow* para la creación de una **NDM** *pads* de entradas y salidas que contengan los *I/O non-metal fillers* y las esquinas:

1. Creamos un *workpace* de flujo únicamente físico:

```
>       create_workspace -flow physical_only -technology usr/synopsys/
TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/TSMCHOME/digital
/Back_End/milkyway/tcb018gbwp7t_270a/techfiles/tsmc018_6lm.tf
PhysicalOnlyWorkspace
```

2. Importamos los *.db files* donde están los *pads* de entradas y salidas:

```
> read_db { TSMC/180/CMOS/G/IO3.3V/iolib/LINEAR/tpd018nv_280a_FE/
TSMCHOME/digital/Front_End/timing_power_noise/NLDM/tpd018nv_280a/
tpd018nvtc.db }
```

3. Importamos el *.lef file* donde están los *pads* de entradas y salidas:

```
> read_lef TSMC/180/CMOS/G/IO3.3V/iolib/LINEAR/tpd018nv_280a_FE/
TSMCHOME/digital/Back_End/lef/tpd018nv_280a/mt_2/6lm/lef/
tpd018nv_6lm.lef
```

4. Hacemos el *check* del *workspace* y desplegamos la ventana de errores para verificar que todo esté en orden:

```
> current_workspace; check_workspace
```

```
> gui_create_window -type MessageBrowserWindow
```

```
> open_ems_database check_workspace.ems
```

5. Hacemos *commit* y *save* a la librería *.ndm* con el nombre **CornersWorkspace.ndm**

```
> current_workspace PhysicalOnlyWorkspace; commit_workspace  -
output CornsersWorkspace.ndm
```

Aggregate Flow

Una vez creadas estas cuatro librerías *.ndm* se debe crear una única librería a la que se le refiere en el proceso de la síntesis física como *reference library*. Esta librería cuenta con toda la información necesaria para poder hacer un mapeo de las celdas de la síntesis lógica, entre otras, necesarias para realizar la síntesis física. Un *Aggregate Flow* es una compilación de múltiples librerías *.ndm* referidas en una única librería. A continuación se crea la librería usada en este proyecto:

Ejemplo de un *Aggregate Flow* para la creación de una **NDM** que contenga a las cuatro librerías anteriormente creadas.

1. Creamos un *workpace* de flujo agregado:

```
> create_workspace -flow aggregate AggregateWorkspace
```

2. Importamos las *.ndm* previamente creadas:

```
> read_ndm /mnt/nfs/compartida/ASA/LibreriasNDM/CornersWorkspace.
ndm
```

```
> read_ndm /mnt/nfs/compartida/ASA/LibreriasNDM/FillersWorkspace.
ndm
```

```
> read_ndm /mnt/nfs/compartida/ASA/LibreriasNDM/PadsWorkspace.ndm
```

```
> read_ndm /mnt/nfs/compartida/ASA/LibreriasNDM/StandardWorkspace
.ndm
```

4. Hacemos el *check* del *workspace* y desplegamos la ventana de errores para verificar que todo esté en orden:

```
> current_workspace; check_workspace
```

```
> gui_create_window -type MessageBrowserWindow}
```

```
> open_ems_database check_workspace.ems}
```

5. Hacemos *commit* y *save* a la librería *.ndm* con el nombre **TSMCWorkspace.ndm**

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

7.2. Migración de La Síntesis Física de IC Compiler I a IC Compiler II

Una vez finalizada la librería de referencia, el proceso de síntesis física puede iniciar. Para que esto se pueda efectuar de forma correcta se debe seguir una linea de trabajo ordenada. La herramienta de **ICCII** cuenta con un *Task Assistant* el cual segmenta, organiza y guía al usuario por el proceso de síntesis con una interfaz gráfica. Sin embargo, no todas las

opciones para el proceso de este proyecto se encuentran dentro de esta guía, por lo tanto el apoyarse con los manuales de guía de usuario[19] y guía de diseño [20] es un excelente forma de complementar a esta interfaz. Adicionalmente el manual de comandos de **ICC2** [21] y las opciones dentro de la aplicación [22] proporcionan mayor detalle y algunos ejemplos complementarios a las guías de usuario. El comando para invocar a **ICCII** desde consola es: `icc2_shell -gui`

7.2.1. Creando un Bloque de Trabajo

Para poder iniciar a trabajar en el proceso de síntesis física primero se debe crear una librería **NDM** en donde se almacenará toda la información del bloque que se estará creando. Luego se le debe cargar el archivo de síntesis lógica de extensión verilog *.v* que contiene la información de la interconexión de las celdas estándar y los *pads* de entradas y salidas. Es por esto que es de suma importancia la librería de referencia contenga toda la información de estas celdas, tanto física como lógica. Si alguna de estas celdas no existiese se presentarían errores desde un comienzo. A continuación se describe como realizar este proceso en **ICCII**.

Ejemplo de la creación de un bloque y la importación de librerías necesarias para la síntesis física asociadas a este, de una compuerta NOT.

1. Creamos la librería **NDM** que almacenará todas las cosas que se realicen en la síntesis física y le asociamos su librería de referencia. (La librería de referencia es la que fue creada en la sección anterior). El nombre de dicha librería es **NOT_SYN.ndm**

```
> create_lib NOT_SYN.ndm -technology /usr/synopsys/TSMC/180/CMOS/
  G/stclib/7-track/tcb018gbwp7t_290a_FE/TSMCHOME/digital/Back_End/
  milkyway/tcb018gbwp7t_270a/techfiles/tsmc018_6lm.tf -ref_libs /mnt/
  nfs/compartida/ASA/LibreriasNDM/TSMCWorkspace.ndm
```

2. Se lee el archivo verilog de la compuerta NOT que ya pasó por el proceso de síntesis lógica. El nombre de este archivo es **NOT_IO_syn.v**. A partir de aquí se crea un bloque con el nombre del módulo de mayor jerarquía del archivo de verilog. En este caso el nombre de este módulo es **NOT_IO**. Esta información será relevante más adelante.

```
> read_verilog /mnt/nfs/compartida/NOT/SintesisLogica/
  salidas_not_io/NOT_IO_syn.v
```

3. Se lee el archivo de extensión *.sdc* para importar los requerimientos de diseño impuestos en la síntesis lógica. Este archivo se llama **NOT_IO.sdc**

```
> read_sdc -echo -syntax_only /mnt/nfs/compartida/NOT/
  SintesisLogica/salidas_not_io/NOT_IO.sdc
```

4. Por último, se importan los archivos TLU+, estos contienen toda la información de los coeficientes de resistencia y capacitancia, esenciales para la extracción de parásitos. Adicionalmente se importa en conjunto el *layermap* que es una descripción de las características de la cantidad de capas de metal que el diseño tiene.

```
> read_parasitic_tech -tlup /usr/synopsys/TSMC/180/CMOS/G/stclib
/7-track/tcb018gbwp7t_290a_FE/TSMCHOME/digital/Back_End/milkyway/
tcb018gbwp7t_290a/tluplus/t018lo_1p6m_typical.tluplus -layermap /usr
/synopsys/TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/
TSMCHOME/digital/Back_End/milkyway/tcb018gbwp7t_290a/tluplus/star.
map_6M
```

Figura 10: Instancias del archivo verilog en el bloque de trabajo

7.2.2. Instanciar *pads* de alimentación y esquinas

Antes de iniciar con el proceso de *floorplaning* es indispensable que todas las celdas que se van a utilizar en las entradas y salidas estén definidas. Por lo tanto en esta etapa se definen los *pads* de VDD y VSS que no se definieron en la síntesis lógica. Adicionalmente, las esquinas del chip se deben definir, esto ya que son las únicas celdas de entradas y salidas que son cuadradas y le otorgan orden, simetría al chip. Además carecen de conexiones físicas en el circuito por la posición en donde se encuentran.

A continuación se instancian las celdas de VDD y VSS, así como las esquinas de una NOT.

1. De primero instanciamos los *pads* de entradas y salidas. Estos provienen de la librería de referencia, por lo que es indispensable que estén definidos en ella. En nuestro caso

se encuentran dentro del flujo normal *PadsWorkspace.ndm* dentro de *TSMCWorkspace.ndm* y los nombres de estas celdas son: **PVDD1CDG** y **PVSS1CDG**. Los nombres de cada instancia son: **PVDD** y **PVSS**, respectivamente, para distinguirlos de las nets de alimentación.

```
> create_cell {PVDD} PVDD1CDG}
```

```
> create_cell {PVSS} PVSS1CDG}
```

2. Definimos las esquinas del chip, estas las definimos como **CORNER1, CORNER2, CORNER3** y **CORNER4** y provienen de la librería de referencia *TSMCWorkspace.ndm* en la librería *CornersWorkspace.ndm*. El nombre de la celda que se está utilizando es: **PCORNER**.

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

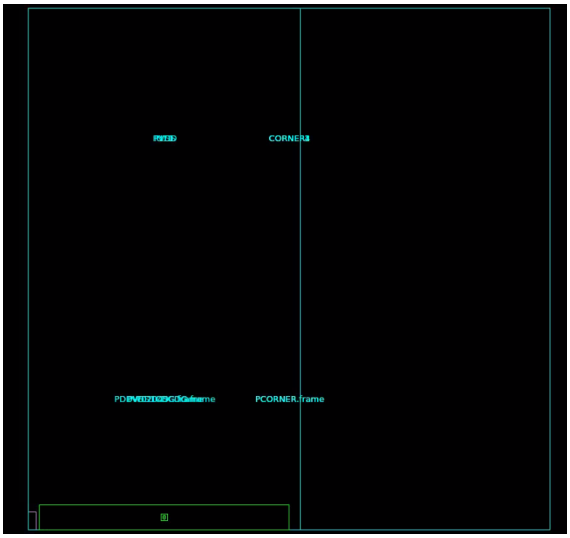

Figura 11: Instancias de las esquinas y *pads* de VDD y VSS

7.2.3. Creación de PG *nets*

Debido a que en la síntesis lógica los *pads* de alimentación no existen, es crítico que se defina una conexión lógica de todos los pines de las celdas con esa nomenclatura entre sí. Asimismo es importante que estas conexiones estén dirigidas hacia los pines de acople con los *pads* de entradas y salidas. De omitir este paso las conexiones de **PG** causan problemas de **DRC** ya que no son celdas de una misma *net*.

Flujo para definir las conexiones lógicas de las celdas, *pads* y pines de **PG**. Extraído del manual de usuario *Design Planning User Manual* [20].

1. Debido a que no estamos haciendo un diseño que tenga un *Unified Power Format* **UPF**, sino que el flujo esta basado en un archivo de verilog, entonces el siguiente comando deriva dos conexiones por defecto con los nombres de **VDD** y **VSS** los cuales y los establece como fuentes de alimentación principales.

```
> resolve_pg_nets
```

2. En seguida especificamos los nombres de las *nets* de **PG** que vamos a crear.

```
> create_net -power VDD
```

```
> create_net -ground VSS
```

3. Luego de crear estas **nets**, se deben asignar a todos los pines, para esto el siguiente comando agarra todos aquellos elementos físicos cuya interconexión se llame **VDD** o **VSS** y los asocia a las **nets** que se crearon.

```
> connect_pg_net -net VDD [get_pins -physical_context *VDD]
```

```
> connect_pg_net -net VSS [get_pins -physical_context *VSS]
```

4. Luego se debe asegurar que todas aquellas celdas que por algún motivo el programa obvió, se asocien a estas *nets*.

```
> connect_pg_net -automatic
```

5. Por último se solicita un reporte de estas interconexiones para validar que estas existan y esten asociadas a una *net*.

```
> report_cells -power
```

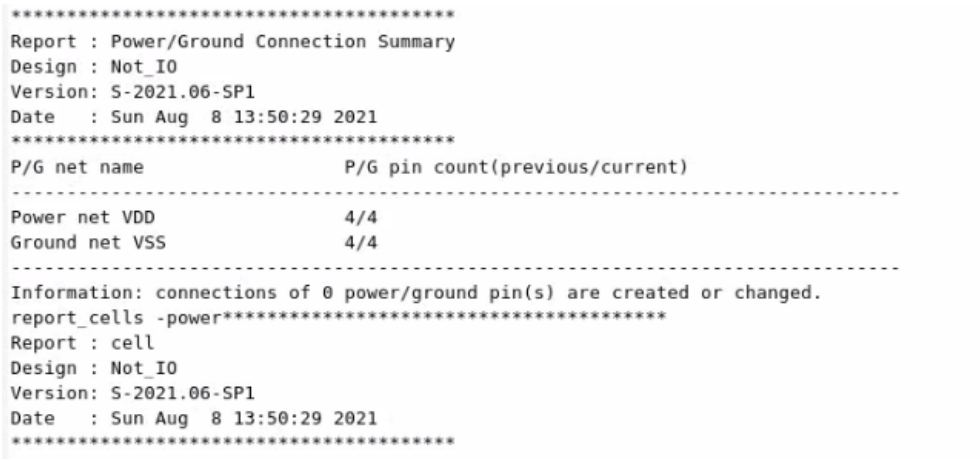

Figura 12: Resumen de consola del reporte de conexiones con 4 de 4 conexiones exitosas de **PG**

7.2.4. Creación del FloorPlan Inicial con Anillo de Entradas y Salidas

Una vez instanciadas todas aquellas celdas que se están utilizando para el diseño, así como todas aquellas conexiones lógicas entre ellas, se puede empezar a crear un *FloorPlan* y el anillo de Entradas y salidas. El *FloorPlan* es la parte del proceso de diseño en donde se define el espacio físico en donde se estarán posicionando los elementos, tanto del nucleo del **CI** como el espacio designado para los *pads* de entradas y salidas. El anillo de entradas y salidas define el espacio en donde estsrán encerrados los pads de entradas y salidas. Son guias virtuales que luego pueden ser utilizadas como directrices de posicionamiento para los pads.

1. El siguiente comando tiene multiples argumentos. En conjunto estos crean el *FloorPlan*.

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

   a. El primer argumento `-site def unit` define el tamaño de una celda básica. En el *techfile* está definido este argumento que es de $3.92\mu m$ bajo el nombre de **unit**. Esto se hace ya que el core debe ser un múltiplo de este argumento para generar filas exactas. Por ejemplo, en este caso se define un core de $100 * 100\mu m^2$ sin embargo el programa aproxima a $98 * 98\mu m^2$ debido a que son 25 filas exactas de $3.92\mu m$.
   b. El segundo argumento `-use_site_row` especifica que se desea utilizar la dimensión **unit** pra definir las filas del core.
   c. El tercer argumento `-keepall` mantiene todas aquellas celdas que se instanciaron, esto para no deshacer procesos anteriores que se encuentren sobre la misma área de trabajo.
   d. El cuarto arguemtno `-side_length {100 100}` define un núcleo con las dimensiones de $100 * 100\mu m^2$.

e. El último argumento `-core_offset {125}` es la distancia a la que se encuentra el borde del núcleo al borde del margen externo.

2. Se define el anillo de entradas y salidas en el margen externo del **CI** además de especificarle que la altura de las esquinas es de $115\mu m$. Para futuras referencias el nombre de este anillo es **anillo_IO**.

```
> create_io_ring -name anillo_IO -bbox {{0.000 0.000} {349.680
  348.000}} -corner_height 115}
```

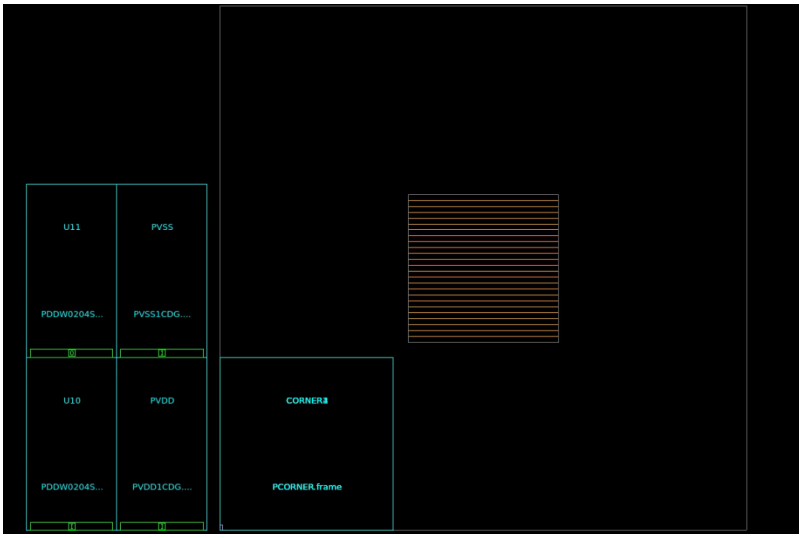

Figura 13: *FloorPlan* con las *site rows* desplegadas y el anillo creado al rededor del margen externo

7.2.5. IO *Placement* y Creación del Anillo de PG

Para poder crear conexiones funcionales entre los *pads* de alimentación y el circuito es necesario crear un **PG** *ring*; este es un anillo que sirve como puente entre las celdas y los *pads* de alimentación. Para esto primero hay que posicionar los pads en el anillo de IO y luego crear el anillo.

A continuación se muestra un ejemplo de un *placement* de IO así como la creación de un anillo de entradas y salidas para una compuerta NOT.

1. Este comando es el indicado cuando se desea hacer el *placement* de los *pads* de entradas y salidas. Cabe resaltar que como esta es una compuerta NOT no se requirió de ningúna especificación adicional. A diferencia de otros casos que se presentarán más adelante.

```
> place_io
```

2. Los siguientesd comandos son necesarios para crear y definir el anillo de **PG**. De primero se crea un patron de anillo. Este patrón se define de forma que los *tracks* horizontales estén en metal 2 y los *tracks* verticales en metal 3. Esta decisión se tomó ya que los pads de **VDD** y **VSS** cuentan con sus pines en metal 2, lo que en un futuro evitará cortos en las redes de alimentación. El espaciado y las distancias entre *straps* fueron extraídas de los ejemplos del *Task Assistant*

```
> create_pg_ring_pattern ring_pattern -horizontal_layer METAL2
 -horizontal_width {2} -horizontal_spacing {2}    -vertical_layer
METAL3 -vertical_width {2} -vertical_spacing {2}
```

3. Luego se crea una estrategia. Las estrategias utilizan patrones para luego ser compiladas. Esta estrategia se crea en el anillo del núcleo, utiliza la estrategia definida anteriormente y la replica 2 veces, una para la net **VDD** y la otra es para la net **VSS** de forma concentrica. Adicionalmente estas están separadas del núcleo por $1\mu m$

```
> set_pg_strategy core_ring    -pattern {{name: ring_pattern}
{nets: {VDD VSS}} {offset: {1 1}}} -core
```

3. Por último se compila la estrategia definida para que esta sea colocada en el diseño.

```
> compile_pg -strategies core_ring
```

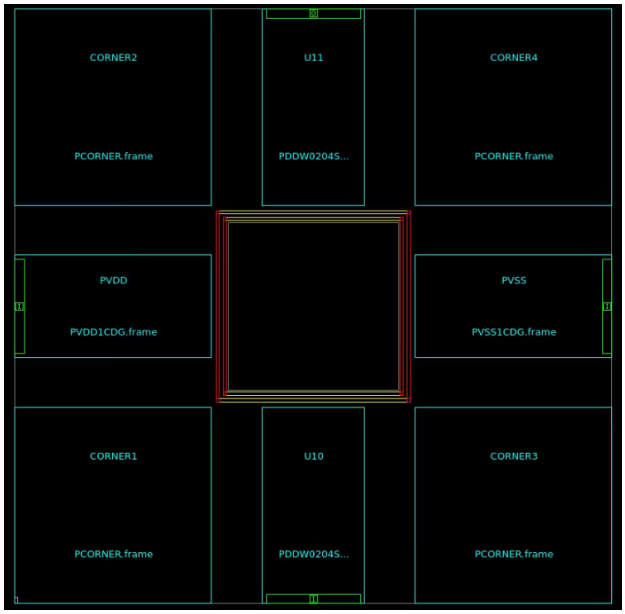

Figura 14: Colocación del anillo de **PG** y *placement* de los *pads* en el anillo de IO

7.2.6. *Cell Placement* y Legalización

Para poder crear un *placement* de celdas exitoso es importante que

1. Esta opción se coloca en **FALSE** para que cuando se haga el *placement* este se efectue sin poner las celdas en posiciones fijas por si requieren de algún movimiento. Si en algún caso se desean fijas, esto se coloca en **TRUE**. Además se colocan los bloqueos como automáticos. Esto se hace en dado caso sean necesarios, el diseñador no tenga que especificarlos manualmente.

```
> set_app_options -name place.coarse.fix_hard_macros -value false
```

```
> set_app_options -name plan.place.auto_create_blockages -value
  auto
```

2. Se crea el *FloorPlan* con las opciones de: optimizar para *timing*, congestiones y que el esfuerzo sea el máximo posible. Para diseños pequeños y sin relojes, no hay mucha diferencia que estos argumentos estén, sin embargo son muy útiles y es preferible dejarlos.

```
> create_placement -floorplan -timing_driven -congestion -effort
  high -congestion_effort high
```

3. Esta es la última parte del flujo del *placement*. Aquí se efectúan todas las descripciones anteriores y se colocan las celdas. Todo esto se registra en la **NDM** del proyecto.

```
> legalize_placement
```

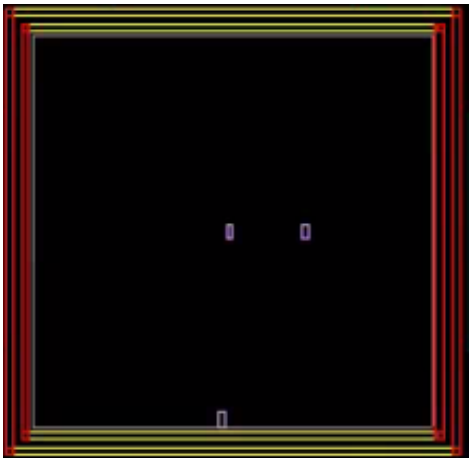

Figura 15: *Cell Placement* del **CI** vista únicamente del core

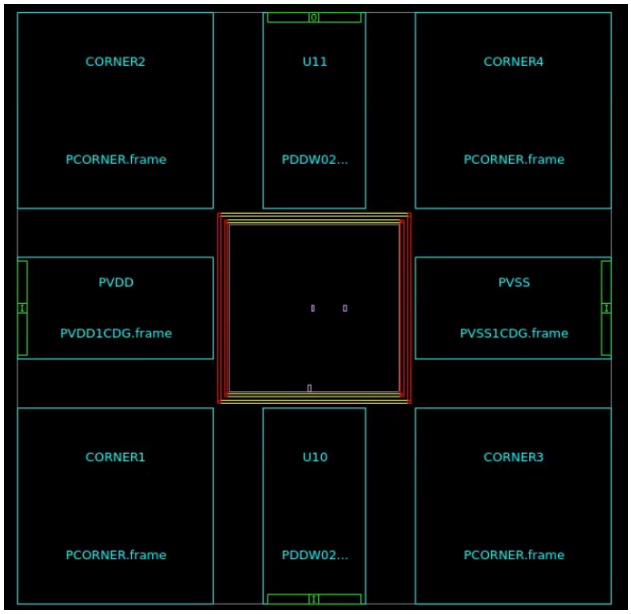

Figura 16: *Cell Placement* con vista de todo el **CI**

7.2.7. Conexión del anillo de PG con los pads de alimentación

7.2.8. Reconección de PG

7.2.9. Creación del *Mesh* y los *Cell Rows*

7.2.10. Unión del PG *Planing*

7.2.11. Ruteo

7.2.12. Creación de *filler cells* para los *IO*

7.2.13. Creación de *filler cells* para el *core*

7.3. Configuración y ejecución de DRC en ICCII

7.4. Circuitos adicionales y cambios en el proceso de síntesis física

7.5. Comandos adicionales

CAPÍTULO 8

Conclusiones

CAPÍTULO 9

Recomendaciones

CAPÍTULO 10

Bibliografía

[1] J. de los Santos, "Diseño de un sumador/restador completo de 32 bits con tecnología CMOS en un proceso de 28 nanómetros usando aplicaciones de diseño de la empresa Synopsys.," en *Trabajo de Graducacion, Modalidad Tésis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2014.

[2] S. Rubio, "Definición del Flujo de Diseño para Fabricación de un Chip con Tecnologia VLSI CMOS," en *Trabajo de Graducacion, Modalidad Tésis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2019.

[3] L. Nájera, "Implementación de circuitos sintetizados a nivel netlist a partir de un diseño en lenguaje descriptivo de hardware como primer paso en el flujo de diseño de un circuito integrado.," en *Trabajo de Graducacion, Modalidad Tésis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2019.

[4] L. Abadía, "Posicionamiento e interconexión entre componentes de un circuito sintetizado para el flujo de diseño de un circuito a escala nanométrica utilizando la herramienta de IC Compiler," en *Trabajo de Graducacion, Modalidad Tésis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2020.

[5] M. Flores, "Corrección de anillo de entradas/salidas y pruebas de antenna y ERC para la definición del flujo de diseño del primer chip con tecnología nanométrica desarrollado en Guatemala," en *Trabajo de Graducacion, Modalidad Tésis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2020.

[6] M. Sibrian, "Verificación de reglas de diseño (DRC) para el desarrollo de un flujo funcional de un circuito integrado con tecnología nanométrica," en *Trabajo de Graducacion, Modalidad Tésis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2020.

[7] "Apple unleashes M1," *Apple Newsroom*, nov. de 2020. dirección: `https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/`.

[8] K. G. Krishna. B, "Circuit Design Enviroment and Layout Planning," *Intel Technology Journal*, págs. 13-20, 1999.

[9] C. Esquit, *Computer Architecture Lecture 1: Introduction and Five Components of a Computer*, jul. de 2020.

[10] ——, *Introduction to VLSI System Design Lecture: Fabrication and Layout*, mayo de 2021.

[11] K. H, "Digital Integrated Circuit Design From VLSI Architectures to CMOS Fabrication," *Intel Technology Journal*, págs. 13-20, 1999.

[12] L. A. López, "Manual Síntesis Física," en *Manual Síntesis Física para IC Compiler I*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2020.

[13] M. Sibrian, "Manual de DRC," en *MANUAL DE USUARIO PARA LA VERIFICACIÓN DE REGLAS DE DISEÑO EN UN CIRCUITO INTEGRADO CON TECNOLOGÍA NANOMÉTRICA*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2020.

[14] *EDA101 - Introduction to Electronic Design Automation*. Cadence Design Systems, abr. de 2020. dirección: `https://www.youtube.com/watch?v=HkT363NUOBA`.

[15] Synopsys. dirección: `https://www.synopsys.com/silicon-design.html`.

[16] ——, dirección: `https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html`.

[17] ——, dirección: `https://www.synopsys.com/silicon/mask-synthesis/icv-workbench.html`.

[18] ——, *Library Manager User Guide*, Synopsys Inc., USA, 2021.

[19] ——, *IC Compiler ™II Implementation User Guide: Version S-2021.06, June 2021*, Synopsys Inc., USA, 2021.

[20] ——, *Design Planning User Guide: Version S-2021.06, June 2021*, Synopsys Inc., USA, 2021.

[21] ——, *IC Compiler ™II Tool Commands: Version S-2021.06, June 2021*, Synopsys Inc., USA, 2021.

[22] ——, *IC Compiler ™II Application Options and Attributes: Version S-2021.06, June 2021*, Synopsys Inc., USA, 2021.

CAPÍTULO 11

Anexos

11.1. Planos de construcción