UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Síntesis física de un diseño electrónico, en lenguaje de descripción de *hardware*, para fabricación de un circuito integrado en silicio

Trabajo de graduación presentado por Diego Soler Castañeda para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2019

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

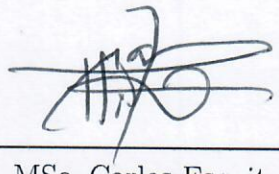
Síntesis física de un diseño electrónico, en lenguaje de descripción de *hardware*, para fabricación de un circuito integrado en silicio

Trabajo de graduación presentado por Diego Soler Castañeda para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2019

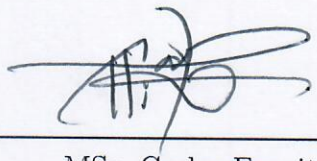
Vo.Bo.:



(f) _____

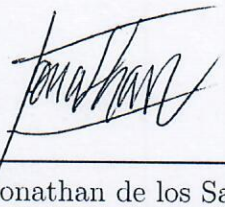
MSc. Carlos Esquit

Tribunal Examinador:



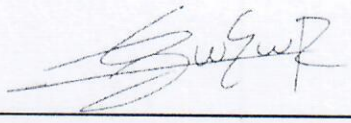
(f) _____

MSc. Carlos Esquit



(f) _____

Ing. Jonathan de los Santos



(f) _____

Ing. Guilmar Escobar

Fecha de aprobación: Guatemala, 10 de enero de 2020.

Índice

Lista de figuras

Lista de cuadros

Resumen

Este trabajo tiene como objetivo general realizar la síntesis física de un diseño electrónico en lenguaje de descripción de hardware para su fabricación en silicio, en el marco de realizar la fabricación del primer circuito integrado diseñado por estudiantes de ingeniería en Guatemala. El diseño electrónico a sintetizar es una máquina de estados finitos de 345 estados cuya función principal es generar una cadena de caracteres en codificación binaria para su posterior reproducción con un codificador de audio.

La síntesis física fue realizada con la herramienta *IC Compiler* de la empresa *Synopsys*, con la que se obtuvo un *layout* en silicio, con tecnología de fabricación CMOS de 180 nm, y se generaron archivos GDSII para la fabricación en silicio con un *foundry*. Se realizaron diversos pasos del flujo de diseño de una síntesis física, como *Floorplanning*, *Placement* y *Routing*, sin ningún objetivo particular de consumo de potencia o *timing*. El *foundry* escogido fue TSMC mediante la empresa EUROPRACTICE. El *die* del circuito integrado obtenido tiene un área total de 0.44 μm^2 y está formado por 510 celdas de la librería de diseño de 180 nm proveída por la empresa taiwanesa TSMC.

Abstract

This work has as general objective the physical synthesis of an electronic design in Hardware Description Language for its further manufacturing. This work is done under the framework of making the first integrated circuit designed by engineering students in Guatemala. The electronic design to be synthesized is a 345 states finite state machine whose main function is to generate a string of characters. The physical synthesis was performed with the IC Compiler tool of Synopsys, obtaining a silicon layout with 180 nm CMOS technology. The chosen foundry was TSMC through the european company EUROPRACTICE, which support academia and industry with a platform to develop smart integrated systems. Several steps of the design flow of physical synthesis were performed, such as Floorplanning and Placement and Route. The integrated circuit has a die area of 0.44 μm^2, and it is formed by 510 cells of 180 nm TSMC's library.

CAPÍTULO 1

Introducción

Un *chip* o circuito integrado, es uno de los objetos de ingeniería más complejos en el mundo. Los circuitos integrados son manufacturados en un material semiconductor llamado silicio, en el que también se fabrican componentes básicos como transistores, diodos, resistencias o capacitores. Un *chip* puede tener millones de estos componentes dependiendo de su complejidad. Desde el año 1959 (el año que Jack Kilby publicó la primera patente en relación a circuitos integrados) se han creado diferentes términos para reflejar el estado del desarrollo del diseño de circuitos integrados: integración a pequeña escala (SSI) para decenas de transistores en un chip, integración a mediana escala (MSI) para cientos de transistores por chip, integración a gran escala (LSI) con decenas de miles de transistores por chip, e integración a gran escala (VLSI) con cientos de miles de transistores. La integración a gran escala (VLSI) y el sistema en chip (SoC) son los últimos términos para cubrir el desarrollo de chips modernos. En diseños de integración a gran escala o VLSI, el proceso necesario para la fabricación de un circuito integrado está dividido en una serie de pasos detallados que hoy en día son la guía principal para los diseñadores de VLSI. Esta serie de pasos de conoce como flujo de diseño o *design flow* por su término en inglés.

Debido a que trabajar en una escala nanométrica no es nada simple, los ingenieros de VLSI necesitan herramientas que les aseguren tener diseños rápidos, libres de errores, con funciones definidas y fáciles de replicar para cumplir las altas demandas de la industria de circuitos integrados. Con el flujo de diseño los diseñadores han logrado obtener una metodología que les permite realizar implementaciones de circuitos integrados de forma más rápida, numerosa y libre de errores. El objetivo principal del flujo de diseño es transformar una idea o concepto de funcionalidad a una implementación física. Por ejemplo, inicialmente un ingeniero de VLSI para diseñar y mandar a fabricación un circuito integrado sumador de 32 bits, tuvo que pensar en el concepto general en relación a qué necesita que realice el circuito que quiere diseñar: en este caso, sumar 32 bits. Posteriormente ese concepto tuvo que trabajarlo con el flujo de diseño para que finalmente tuviera listo un diseño en silicio de un sumador de 32 bits apto para fabricación. Para lograr este objetivo del proceso de implementar un concepto a un circuito integrado, el flujo de diseño se divide en dos grandes

estructuras: la estructura de diseño lógico y la estructura de diseño físico.

Cada una de estas estructuras se divide en una serie de pasos, que se detallan en este trabajo, los cuales pueden variar en función del tipo de diseño u objetivo. Sin embargo, la función principal de cada estructura siempre es la misma. El diseño lógico es utilizado para transformar el concepto inicial a nivel de su comportamiento (sin ninguna implementación en circuito, solo el qué hace el circuito) a una descripción estructural, es decir, una descripción con compuertas lógicas y registros sin ninguna característica física. Con este paso los diseñadores ya tienen una idea inicial de la cantidad de compuertas y del tamaño total del circuito. Posteriormente esta descripción pasa a una síntesis física, que es la implementación del circuito en compuertas lógicas y registros con características física acordes a la tecnología de fabricación; y a una implementación física del circuito en un *layout* de silicio, listo para iniciar los procesos de fabricación.

Todo este interés de los ingenieros de diseño de VLSI en lograr metodologías para procesos de diseño más eficientes, ha tenido efecto en posicionar a la industria de los semiconductores y circuitos integrados en una de las industrias con más ingresos anuales, alcanzando cifras de hasta cuatrocientos billones de dólares [1]. Este es un mercado que no para de subir sus ganancias debido a la demanda provocada por los crecientes desarrollos tecnológicos, los cuales han permitido avances en comunicaciones, en eficiencia, en transporte y en la calidad de vida de las personas en general; y todo esto debido al progreso de los circuitos integrados.

Durante la historia, Guatemala normalmente se ha caracterizado por ser un país agrícola, sin embargo, en los tiempos modernos donde los países desarrollados apuestan cada vez más por las empresas de tecnología y las empresas de semiconductores o circuitos integrados; lograr que Guatemala ingrese a este tipo de mercado es un desafío a largo plazo en el país. Sin embargo, una buena pauta para iniciar es que las universidades y sus departamentos de ingeniería e investigación involucren al estudiante en estos ámbitos, particularmente a los ingenieros en electrónica. Es por ello que se ha planteado el objetivo de diseñar el primer circuito integrado en la historia de Guatemala, algo inédito no solo en el país sino que en la región a nivel pregrado, para demostrar que en Guatemala sí se puede lograr un desarrollo en este tipo de conocimiento.

Para lograr este objetivo, es necesario implementar y definir un flujo de diseño acorde al diseño a realizar. Lo importante en el diseño del primer circuito integrado en la historia de Guatemala, es dejar definido el proceso de flujo de diseño para lograr un producto exitoso, con el que en un futuro estudiantes de ingeniería puedan desarrollar sus propios circuitos integrados de aplicación específica en el país. Por tanto, la funcionalidad del circuito no debía ser algo tan complejo de implementar y desarrollar, por lo que se propuso una máquina de estados finitos que generara un mensaje para decodificar en un decodificador de audio. El flujo de diseño se implementó con herramientas de diseño provistas por *Synopsys*, empresa líder en el desarrollo de software especializado para el diseño de circuitos integrados complejos, que cuenta con un programa universitario del que el departamento de Ingeniería en Electrónica, Mecatrónica y Biomédica de la Universidad del Valle es miembro.

CAPÍTULO 2

Antecedentes

La nanoelectrónica y los temas relacionados a VLSI eran un tema desconocido en la Universidad del Valle de Guatemala hasta el año 2013, fecha en la que se empezó a impartir el curso de Introducción al diseño de sistemas VLSI a los estudiantes de Ingeniería en Electrónica por el MSc. Carlos Esquit. Desde entonces el departamento de Ingeniería en Electrónica, Mecatrónica y Biomédica de la Universidad del Valle de Guatemala inició a involucrarse más en el tema, consiguiendo el acceso a software de diseño de circuitos electrónicos de la empresa estadounidense *Synopsys*, mediante la suscripción al programa universitario que ofrecen [2].

Con el acceso a estas herramientas, los alumnos han podido involucrarse en un entorno de diseño de circuitos electrónicos vanguardista y muy acorde a la realidad de la industria, posicionando al departamento y a la Universidad en una posición privilegiada respecto a otras universidades en la región. Es por ello que es posible realizar este trabajo de graduación, apuntando a abrir las puertas de la universidad y del país a un ámbito de diseño de circuitos electrónicos mundial. Anteriormente en el departamento se realizó el trabajo de graduación relacionado con VLSI presentado en [3], en el que se diseña un sumador/restador de 32 bits con un PDK (*Process Design Kit*) de 128 nm de *Synopsys*. En dicho trabajo se implementan los módulos de compuertas lógicas presentados en el PDK utilizado en ese trabajo, para formar el sumador/restador. Dicho trabajo fue la primera publicación del departamento relacionado con nanoelectrónica, y el primer trabajo grande realizado con las herramientas de *Synopsys*. Si bien no se realizó un flujo de diseño completo como tal, el trabajo es buena referencia para consulta de la utilización del entorno de las herramientas de *Synopsys*.

Por otra parte, en Latinoamérica han habido trabajos de desarrollo en VLSI de parte de universidades sin mucha experiencia en esa área. Un ejemplo es el trabajo de A. Díaz[4]. En dicho trabajo de la Universidad de Chile, el autor realiza un flujo de diseño completo con herramientas de *Synopsys*, obteniendo como resultado los archivos GDSII necesarios para la fabricación del circuito integrado. Este trabajo es muy importante en la región, pues presenta una aplicación concreta de proyectos universitarios de VLSI y un vistazo al desarrollo en esta área de la que se puede ser capaz en Latinoamérica.

CAPÍTULO 3

Justificación

La investigación y el desarrollo en el campo de los circuitos integrados y semiconductores es un área inexplorada no solo en Guatemala sino que en gran parte de la región centroamericana. En Guatemala nunca ha habido una investigación o desarrollo de esta índole. En el campo de la Ingeniería Electrónica, el diseño de circuitos a muy grande escala es una de las ramas más estudiadas por los profesionales. Año tras año, las tecnologías de diseño avanzan y se diseñan procesadores y microprocesadores más pequeños y más rápidos, sin embargo, el flujo de diseño en términos generales sigue siendo el mismo.

Diseñar e investigar sobre el flujo de diseño para la fabricación de un circuito integrado sería de gran importancia para sentar las bases de un área de desarrollo profesional para un Ingeniero en Electrónica en último año; por otra parte, el tema sería vanguardista en la región y en la Universidad, por lo que resaltaría a Guatemala y a la Universidad del Valle en el ámbito tecnológico regional.

Por otra parte, para fabricar el circuito integrado se necesita la colaboración con un empresa que preste servicios de fabricación en programas educativos. EUROPRACTICE es una empresa europea que presta servicios de fabricación de circuitos integrados a universidades u organizaciones gubernamentales. Varias universidades europeas han trabajado con esta empresa como parte de sus programas de enseñanza en el ámbito de VLSI, sin embargo, EUROPRACTICE también ha prestado sus servicios a organizaciones de Europa y de Latinoamérica.

En su reporte anual del año 2018, consultado en [5], mencionan que en 2018 trabajaron un total de 624 diseños alrededor del mundo, incluído países como Brasil, México, Uruguay y Colombia. Por ejemplo, en Colombia prestaron su servicio de fabricación a la Universidad Industrial de Santander para fabricar un procesador pequeño [5]. Otro ejemplo del involucramiento de universidades latinoamericanas con empresas como estas, es el mencionado en [6]. Estudiantes de ingeniería en Electrónica en Valparaíso, Chile, trabajaron con MOSIS para la fabricación de un circuito integrado con funcionalidad de timer; y para el flujo de diseño y todos los pasos de diseño utilizaron herramientas de *Synopsys*. Por tanto, la colaboración de una universidad latinoamericana, en este caso la Universidad del Valle de Guatemala, con

empresas como EUROPRACTICE o MOSIS sí es posible y ya se ha hecho anteriormente para realizar un flujo de diseño de fabricación de un circuito integrado como un programa educativo a nivel pregrado en Latinoamérica.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Realizar la síntesis física de un diseño electrónico en lenguaje de descripción de hardware que produzca un *layout* listo para fabricación de un circuito integrado en silicio.

4.2. Objetivos específicos

- Trasladar un circuito en lenguaje de descripción de hardware hacia un textitlayout con celdas para fabricación en silicio.
- Realizar el *Place and Route* de celdas de silicio de un circuito electrónico.
- Diseñar anillos para los pines de entrada y salida del *layout* en silicio de un circuito electrónico.
- Realizar verificaciones de reglas de diseño y de *layout* contra esquemático para garantizar la fabricación correcta del circuito integrado.
- Generar los archivos GDSII para la fabricación en silicio de un circuito integrado.

CAPÍTULO 5

Alcance

Este trabajo tiene como objetivo diseñar y terminar un diseño electrónico, proveniente de un lenguaje de descripción de hardware, en una síntesis física que genere archivos GDSII listo para enviar a un *foundry* para la fabricación de un circuito integrado empaquetado. Estos archivos GDSII son una base de datos estándar para el intercambio de información entre las herramientas de diseño de un circuito integrado, y las herramientas de fabricación. Se busca documentar todo este proceso para que los estudiantes del departamento de Ingeniería Electrónica, Mecatrónica y Biomédica de la Universidad del Valle de Guatemala, tengan una referencia para realizar el diseño de circuitos de aplicación específica en proyectos futuros de la Universidad. Dicha documentación estará basada en videotutoriales sobre la instalación y utilización de las diferentes herramientas utilizadas en todo el proceso.

CAPÍTULO 6

Marco teórico

La fabricación de circuitos integrados lleva intrínseca una metodología de diseño o flujo de diseño sumamente estricto, con el fin de lograr un producto final funcional. Esta metodología es un conjunto de pasos estructurados para completar el diseño del *chip*, el cual incluye ciertas reglas, criterios, técnicas y herramientas para cada paso. Cada paso debe complementarse y entenderse con los demás, pues cualquier fallo en el diseño puede provocar un gran gasto de tiempo e inversión económica para los fabricantes en pasos posteriores, por tanto, son de gran importancia los diseños y predicciones del funcionamiento del circuito integrado. Entonces, debido a que el diseño en silicio es complicado, el rol de un buen diseño VLSI ayuda a reducir la complejidad, aumentar la productividad, y asegurar un producto funcional [7].

6.1. Metodología de diseño

Normalmente la familia lógica empleada en los circuitos integrados es la CMOS (*Complementary Metal-Oxide-Semiconductor*). El diseño de circuitos integrados CMOS consiste en definir las entradas y salidas del circuito, cálculos de funcionamiento, diseño (o *layout*) del circuito; y simulaciones y análisis de entradas/salidas después de una extracción de capacitancias y resistencias parásitas. Las especificaciones del circuito no se hacen antes de iniciar el diseño, pues estas pueden cambiar mientras el proceso de diseño madura; esto debido a que durante el proceso y posterior análisis se pueden identificar ciertos intercambios entre rendimiento y costo, manufactura del circuito integrado, o incluso en únicamente detalles que el cliente solicita. Por tanto una vez diseñado el circuito integrado y mandado a producción, ya no se pueden hacer cambios mayores en cuanto a especificaciones técnicas del *System-On-Chip* (SOC). [8]

Los medios mediante los cuales se puede lograr un buen diseño en VLSI se miden en parámetros como:

- Rendimiento, en términos de velocidad, potencia, funcionalidad, flexibilidad.
- Tamaño y costo por bloque del diseño en material semiconductor (*die*).
- Tiempo de diseño.
- Facilidad de verificación y testeo (conocido como *testability*).

La viabilidad de un circuito integrado se afecta en mayor medida por la productividad durante el diseño, es decir, la eficiencia con la que un diseño puede ser convertido desde un concepto hasta un *layout* físico listo para ser producido. Una forma muy efectiva para lidiar con los diseños complejos es creando un modelo con alto nivel de abstracción. Esto quiere decir que el comportamiento del sistema debe ser entendido de manera simple desde su nivel superior (circuito integrado como un todo) hasta a nivel de transistores en el diseño del circuito integrado [9]. Un buen diseño en VLSI debe ser descrito de forma eficiente y consistente en tres dominios de descripción:

- Comportamiento (*behavioral*)
  Se refiere a la implementación en software o en concepto de la funcionalidad del sistema [10]
- Estructural (*structural*)
  Se refiere a cómo los diferentes módulos del sistema se interconectan y funcionan entre sí [10].
- Físico (*physical*)
  Se refiere a la implementación física del sistema, es decir a nivel de *layout* en silicio [10].

Estos tres dominios de descripción tienen en común el lograr el comportamiento adecuado del sistema a diseñar, en función de los requerimientos de diseño, por lo que los ingenieros deben trabajar en los tres dominios en varios niveles de abstracción [10].

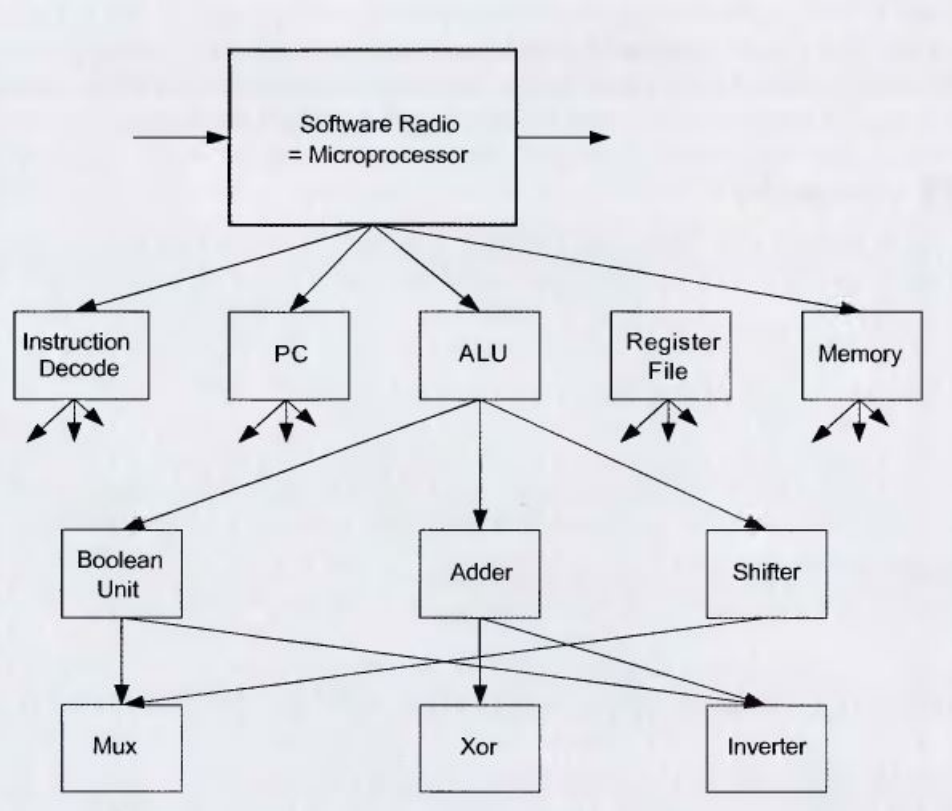
Con el paso del tiempo los circuitos se hacen más pequeños debido a la reducción de la tecnología de fabricación con la que funcionan los transistores dentro de los circuitos integrados. Por tanto es importante tener bien definidas algunas estrategias para hacer más simple los diseños complejos de circuitos integrados. Afortunadamente en el ámbito de VLSI hay cuatro estrategias de diseño bien definidas para lograr diseños eficientes.

- Descomposición o Jerarquía
  La mejor manera de lograr un trabajo es con la estrategia de divide y vencerás. Los diseñadores pueden separar el diseño en múltiples bloques para diseñarlos y testearlos individualmente, para al final interconectarlos todos ya funcionando [7]. Al dividir el sistema en módulos, y cada módulo en submódulos, el diseñador logra un nivel de detalle apropiadamente comprensible.

Una de las principales ventajas de la jerarquía en el diseño de circuitos integrados es que se pueden utilizar componentes o módulos prefabricados. Los módulos prefabricados generalmente son conocidos como *Intellectual Property* (IP) de los fabricantes, y cumplen funciones específicas. Por ejemplo, cierto fabricante puede poseer un sumador de 32 bits en su *Intellectual Property*, y si un diseñador lo necesita para su diseño lo puede utilizar e importar sin necesidad de tener que diseñar nuevamente un módulo sumador de 32 bits.

Un microprocesador típicamente contiene elementos como *decoders*, *program counter* (PC), unidad aritmética lógica (ALU), registros y memoria. Si un ingeniero de diseño quiere implementar un microprocesador puede dividir en módulos los elementos mencionados anteriormente, y la ALU la puede volver a dividir en otros submódulos, con el fin de tener el sistema bien jerarquizado. La siguiente figura muestra un microprocesador que en teoría es algo complejo, dividido en componentes más simples en un nivel de jerarquía [9].

Figura 1: Jerarquía de un microprocesador [9]



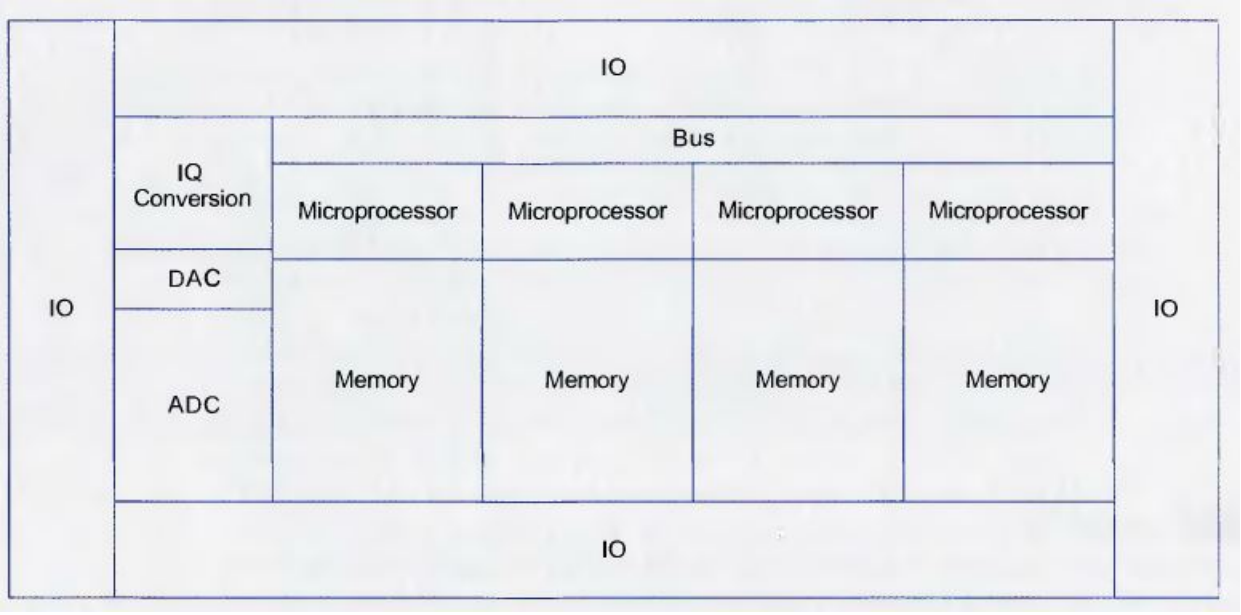
- Regularidad
  La jerarquía por sí sola no resuelve el problema de complejidad en el diseño de un circuito integrado. La regularidad es una guía que busca dividir la jerarquía en un set de bloques similares. Por ejemplo en un diseño a nivel de circuitería, se pueden utilizar transistores con tamaño uniforme fijado, mientras que a nivel de compuertas o en un diseño en HSPICE, se pueden utilizar librerías de diferentes tecnologías para utilizar compuertas lógicas con longitud variable.

La regularidad reduce el número de subcomponentes a validar y permite una operación más eficiente, por tanto simplifica la verificación. La regularidad se puede ver entonces como la descomposición de un sistema complejo en otro más simple y con bloques similares lo más numeroso posible [9].

- Modularidad
  La modularidad en términos simples es tener módulos con interfaces y funciones bien definidas [9]. Cada módulo debe estar bien definido en términos de sus funciones, nombre, tipo de señales, restricciones eléctricas y de tiempo en cada uno de sus puertos o capacitancia de carga si se refiere a puertos de entrada o salida. La modularidad es ventajosa en el sentido de que cada módulo se puede diseñar relativamente independiente de otros y es importante para que el diseñador tenga claridad al documentar posibles problemas al interconectar módulos en el diseño.

- Localidad
  La localidad se refiere a un buen diseño de posicionamiento de los módulos en el circuito integrado. Esto quiere decir que las conexiones se encuentren principalmente entre módulos vecinos, para evitar conexiones de grandes distancias dentro del *die*.
  Por ejemplo, en el siguiente diseño de un circuito integrado vemos que el convertidor analógico-digital (ADC) y el convertidor digital-analógico (DAC) se encuentran adyacentes a los pines de entrada y salida. Aquí se aplica de forma eficiente el concepto de localidad, pues los bloques analógicos generalmente manejan más corriente que los bloques digitales y necesitan buses más cortos con menor resistencia, aparte que se necesita que estén alejados de bloques digitales para evitar interferencia de ellos.

Figura 2: Ejemplo de localidad en un circuito integrado [9]



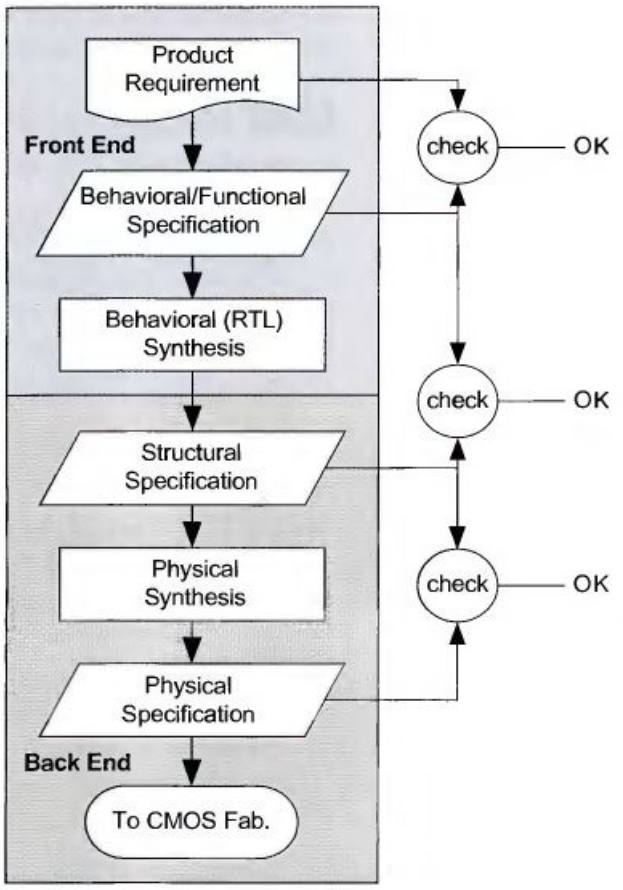
6.2. Flujo de diseño

Hoy en día los diseñadores de circuitos integrados deben afrontar el difícil reto de construir elementos con miles de millones de transistores de forma correcta y de forma rápida. Diseñar y llevar un circuito integrado de aplicación específica (ASIC) desde el concepto de su funcionamiento hasta su implementación en silicio de forma errónea, se traduce a riesgos económicos muy grandes que una empresa diseñadora de circuitos integrados no puede permitir; y estar repitiendo diseños hasta lograr uno correcto se traduce a pérdidas en tiempo de fabricación y diseño intolerables, pues las ventanas de tiempo se perderían y también tendría un impacto económico en la empresa. Existen miles de casos de productos con circuitos integrados de aplicación técnica muy buena que no fueron capaces de producir ningún dolar debido a sus retrasos de fabricación por errores. Para evitar estos contratiempos, los diseñadores de circuitos integrados definieron un flujo de diseño (o *design flow*) en el cual referenciarse para competir en una industria tan demandante como la de los circuitos integrados y materiales semiconductores [11]. En la Figura 3 se observa un flujo de diseño generalizado.

Un flujo de diseño, en el área de Ingeniería Electrónica, es un set de procedimientos con los que el diseñador se guía para la implementación final de un circuito como un circuito integrado, con una posibilidad decente de estar libre de errores [9]. Mediante el flujo de diseño los diseñadores pueden acortar el ciclo y tiempo de diseño con el fin de lograr producciones más rápidas.

El flujo de diseño contiene una secuencia de operaciones que transforman las intenciones del diseñador de circuitos integrados desde un concepto hasta una implementación física. El diseño inicia a un nivel *behavioral* y procede con un nivel *structural* (es decir compuertas y registros), lo que se conoce como *Register Transfer Level* (RTL) pues el diseño se implementa en un lenguaje de descripción de *hardware* (HDL) con elementos lógicos. Posteriormente esta descripción pasa a una síntesis física, en la que se transforma a una descripción física apta para la fabricación. Algunos diseñadores se enfocan solamente en los diseños y simulaciones a nivel *behavioral*, para que una empresa tercera se dedique a la implementación física [9].

Figura 3: Flujo de diseño generalizado [9]



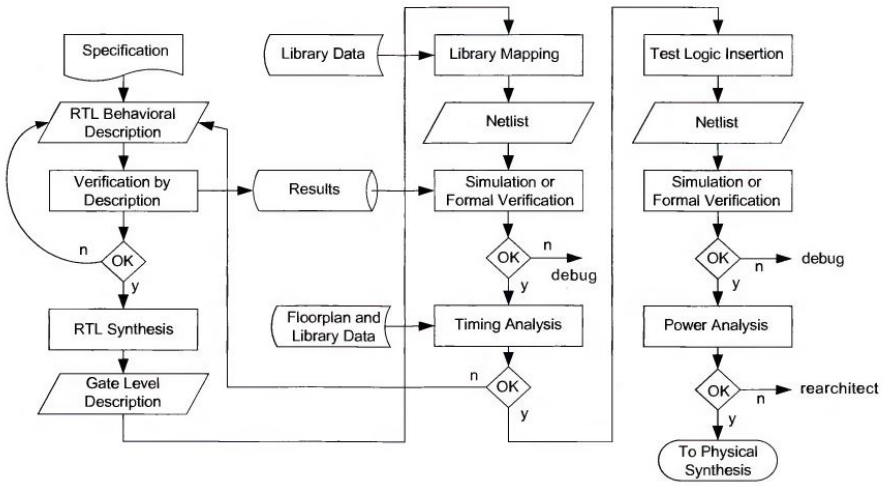
6.2.1. *Síntesis a nivel de comportamiento*

A este nivel la operación se realiza sin especificar la implementación del circuito. En esta síntesis se utilizan herramientas que permitan transformar directamente una descripción RTL a un *netlist* a nivel compuertas estructural. En la Figura 4 se muestra un diagrama de flujo de una síntesis a nivel *behavioral*.

Diseño lógico y verificación

El diseño inicia con una especificación del circuito, que puede ser una descripción en un texto, del circuito a implementar. El diseñador convierte este concepto a una descripción RTL *behavioral*, mediante la utilización de HDLs como *Verilog*. Posteriormente esta descripción es simulada para corroborar su correcto funcionamiento acorde a las especificaciones y requerimientos previos, con la utilización de plataformas de testeo adecuadas [9].

Figura 4: Flujo de síntesis a nivel *behavioral* [9]



Síntesis RTL

El siguiente paso es sintetizar la descripción del circuito, la cual se realiza mediante la conversión del RTL a compuertas y registros genéricos, optimizando la lógica para mejorar velocidad, y mapeando las compuertas a una librería de celdas. También se incluyen pasos como la optimización de potencia o de unidades lógicas. Por otra parte, se debe comprobar que estas conversiones de nivel *behavioral* a estructural funcionen igual que la descripción en el HDL, probando las mismas plataformas de testeo a estas nuevas implementaciones [9].

Conversión a librerías

El mapeo a librerías traduce una descripción genérica a nivel de compuertas lógicas con un HDL a un *netlist* con compuertas lógicas específicas correspondientes a una librería [9]. Por ejemplo, con una librería de compuertas CMOS en tecnología de fabricación de 32 nm, una descripción de un sumador de 32 bits en HDL ahora tendría correspondencia de las compuertas lógicas necesarias para su funcionamiento, a compuertas lógicas predefinidas con una tecnología de fabricación implementada.

Verificación funcional

En esta operación se debe comprobar que el *netlist* obtenido en el paso anterior tenga la misma función que lo descrito en el HDL. Volviendo al ejemplo del sumador de 32 bits, se debe comprobar que el sumador ahora implementado con compuertas lógicas de 32nm realmente sume 32 bits de manera correcta. Generalmente el mapeo a librerías no provoca errores, pero los HDLs siempre están sujetos a ambiguedades que pueden ocasionar traducciones incorrectas en los *netlist* generados, por lo que este paso siempre es necesario [9].

Análisis de *timing* estático

Una vez la relación y equivalencia entre las partes *behavioral* y *structural* se han establecido correctamente, es necesario chequear y analizar los requerimientos de *timing* del diseño. Es decir, ¿el circuito es suficientemente rápido en su funcionamiento? Este análisis se realiza con un *timing analyzer*. Herramientas como el *PrimeTime* o actualmente *NanoTime* de *Synopsys*, permiten análisis de *timing* muy precisos [9].

Testeo

En esta etapa se insertan registros escaneables con el fin de que el estado del circuito diseñado pueda ser fijado y monitoreado. En esta etapa se utilizan técnicas como la generación automática de patrones para verificar el funcionamiento correcto del circuito en función de varias entradas. También se utiliza la técnica de *Built-in Self Test (BIST)*, que es un proceso en el que el circuito realiza un auto testeo de su operación y funcionalidad sin necesidad de tener *hardware* extra conectado; esto es ventajoso principalmente para circuitos que no tengan conexiones directas a pines externos [9].

Análisis de potencia

Finalmente, antes de proceder a la fabricación física del circuito integrado, se realiza una estimación del consumo de potencia. El consumo de potencia va a depender del funcionamiento de las compuertas en función de las entradas que reciba el circuito integrado en un determinado momento. El análisis de potencia puede realizarse desde un conjunto particular de vectores de entrada mediante una simulación, y posterior evaluación del cambio de capacitancia en cada nodo del circuito en cada transición de reloj. Es importante mencionar que si durante esta etapa el consumo de potencia es muy alto, el diseño debe regresar a la etapa de arquitectura, para realizar un nuevo diseño y resolver el problema [9].

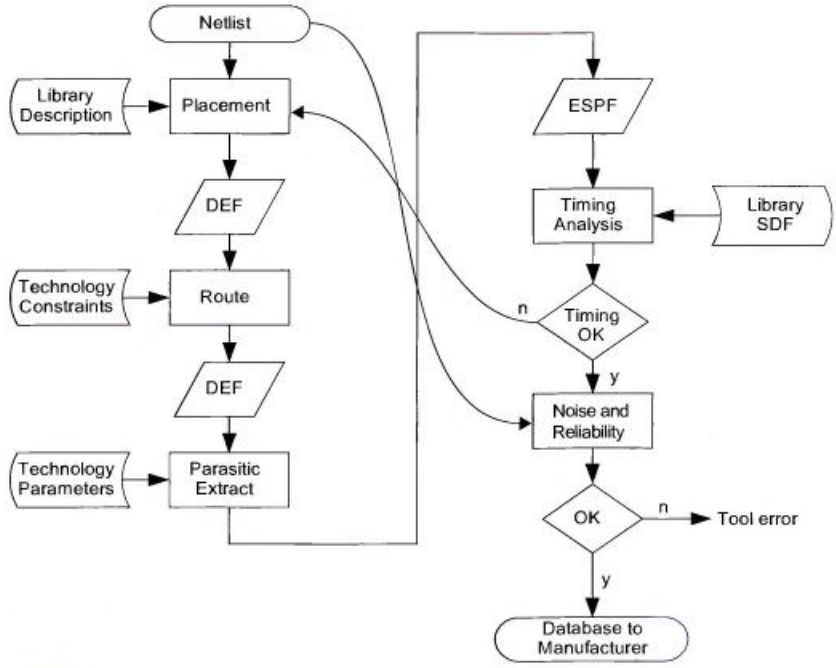
6.2.2. Síntesis física

La generación del *layout* en silicio es de los últimos pasos en el proceso para convertir un diseño de un circuito electrónico a un objeto manufacturable, es decir, convertir un diseño del dominio estructural al dominio físico. Este paso es llamado comúnmente como síntesis física. La Figura 5 muestra el flujo de diseño típico de la etapa de generación de *layout* [9].

Capas conductoras y sus características

Los procesos de VLSI permiten el acceso a diferentes capas para utilizar en el diseño. Estas capas pueden ser capas de materiales semiconductores como metal, polisilicio o las diferentes difusiones utilizadas en el proceso de fabricación. Estas capas contienen propiedades geométricas y eléctricas, sin embargo, suelen diferir entre ellas. [12]

Figura 5: Flujo de síntesis de la implementación física [9]



Propiedades geométricas y reglas de diseño

La transferencia del diseño durante la síntesis física hacia un *die* real en silicio se realiza con métodos fotolitográficos y quitar selectivamente material no deseado. Durante este proceso pueden ocurrir algunos efectos que impiden obtener la resolución deseada. Entre esos efectos se pueden mencionar las tolerancias y alineaciones de las fotomáscaras, difracciones de ondas y efectos de proximidad, tolerancia de los fotoresistores, reflexiones de capas inferiores, entre otros [12].

Para asegurar que la mayoría de diseños fabricados por oblea de silicio sean buenos, los ingenieros de diseño en VLSI han definido reglas de diseño (o técnicamente conocidas como *layout rules*). Estas reglas de diseño se tienen que cumplir para cada proceso de fabricación específico, si no el diseño no se podrá manufacturar con dicho proceso. Estas reglas se pueden ver como un compromiso entre la eficiencia de fabricación y diseño, las características eléctricas deseadas y la densidad del diseño [12]. En los *softwares* especializados para diseño de circuitos integrados, estas reglas de diseño se pueden analizar desde una herramienta llamada *Design Rule Check (DRC)*.

Como se mencionó anteriormente, cada fabricante decide las reglas de diseño que deben cumplir los interesados en fabricar con ellos. Dichas reglas contienen principalmente las siguientes restricciones geométricas:

- Anchura mínima/máxima
- Espaciado mínimo entre capas iguales
- Espaciado mínimo entre capas distintas
- Encerrado mínimo
- Extensión mínima

Propiedades eléctricas

Las capas utilizadas en el diseño están hechas con diferentes materiales y con diferentes grosores, por lo que difieren en sus características eléctricas. En este rubro, para los ingenieros de diseño es de suma importancia obtener resistencias bajas y capacitancias parásitas bajas para minimizar los tiempos de *switcheo* y los retardos en los nodos del circuito [12].
En el diseño VLSI, el plan de conexión entre las capas y nodos del circuito, en especial las de alimentación, deben hacerse en función tanto de la resistencia como de la carga de corriente. Si los conductores son afectados por una corriente mayor a la que pueden soportar, los conductores tienden a acercarse a un punto de ruptura, produciendo un fenómeno conocido como electromigración. Los ingenieros de diseño deben evitar que ocurra este fenómeno, realizando un diseño de colocación de celdas y de conexión adecuado [12].

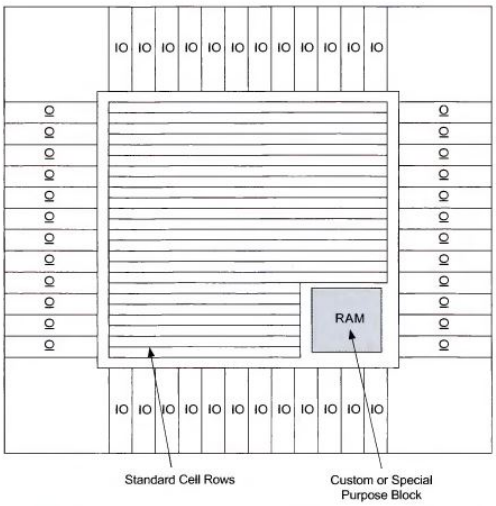
Floorplanning

Posterior a la síntesis lógica, una vez obtenido el *netlist* de compuertas en HDL, es primordial colocar las celdas de silicio en el diseño. Esto es conocido como *floorplanning*. El *floorplan* indica cómo el *die* de silicio se va a particionar para acomodar los diferentes bloques o celdas en los que se divide el circuito, las conexiones y buses entre ellos, la locación de las salidas y entradas, y así como la distribución de la alimentación y de señales importantes (como la señal de reloj por ejemplo) [12]. El *floorplan* tiene un impacto significativo en el rendimiento y costo del producto final, es decir, el circuito integrado físico. En este proceso se especifican elementos y detalles tales como:

- Particiones por bloques dentro del circuito
- Número y colocación, tamaño y forma anticipadas de los bloques
- Selección de empaquetado y utilización de pines/pads
- Buses y señales eléctricas críticas
- Restricciones de reloj y de alimentación

Todos estos detalles mencionados, culminan en un *floorplan* adecuado que sirve como objetivo para los ingenieros de diseño durante los pasos posteriores de la síntesis física [12]. En la siguiente figura se muestra un diseño de un circuito integrado, donde se identifican los diferentes módulos después de un *floorplanning* adecuado.

Figura 6: Circuito integrado genérico organizado por celdas [9]

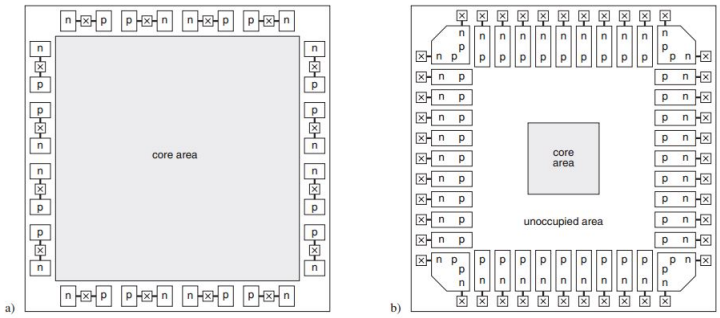


Por otra parte, es importante establecer un diseño adecuado para las entradas y salidas. La cantidad de pines de entradas y salidas es algo crítico durante el diseño de un circuito integrado, pues esto afecta directamente a la selección de un empaquetado para el *die* de silicio generado; y por tanto, los costos finales de fabricación por unidad. En la industria existen dos posibilidades para preparar los pines de entrada y salida:

- Diseño limitado por el núcleo.
  Un núcleo del circuito está rodeado por pocos pines, por tanto, el tamaño del *die* está dado principalmente por el tamaño del núcleo, es decir, la complejidad del funcionamiento del circuito.

- Diseño limitado por la cantidad de pads.
  Un núcleo pequeño del circuito está rodeado por una cantidad mayor de pines. En esta situación, la cantidad de pines necesarios para el circuito impone obligatoriamente un tamaño mínimo al *die*, por lo que se puede perder una gran cantidad de área.

Figura 7: a) Diseño limitado por núcleo vs b) Diseño limitado por pines [12]



Para estos casos, el *floorplanning* está destinado a impactar la arquitectura del diseño. La geometría de los pads se pueden elegir de tal manera que se minimice el tamaño del *die*. Los pads estrechos pero altos que se colocan perpendicularmente al perímetro del *chip*, son óptimos para diseños limitados por el número de pads. En cambio, las situaciones de diseños limitados por el núcleo, se sirven mejor con pads anchos y planas colocadas a lo largo del perímetro del chip. En la Figura 7 se ejemplifican estos casos.

Place and Route

El *Place and Route*, es el paso del flujo de diseño de un circuito integrado, en el que los ingenieros de diseño colocan e interconectan las celdas o bloques del circuito, dentro del area que ocupará el diseño, cumpliendo con los objetivos planteados en el paso previo de *floorplanning*. Para diseños muy grandes, los *floorplannings* tienden a reflejar una organización funcional. Esto facilita la descomposición en piezas de tamaño manejable que pueden ser diseñadas y verificadas por personas o equipos separados. En estos pasos, los diseñadores más experimentados generalmente comienzan con bloques que esperan que permanezcan estables durante el diseño, e incluyen una serie de compuertas adicionales en bloques que sospechan que necesitarán modificaciones o correcciones de errores en una fecha posterior.

Dichas celdas de repuesto permiten realizar cambios de última hora sin alterar demasiado el diseño previo; y en caso de que aparezca una pequeña falla después de que se hayan fabricado los prototipos, es probable que se pueda solucionar rehaciendo solo unos pocos metales y máscaras. Por otra parte, durante esta etapa puede ser necesaria una revisión importante del *floorplanning* inicial, si el área, el retraso de la ruta crítica, la disipación de potencia u otras cifras críticas difieren demasiado de las estimaciones iniciales [12].

Un punto importante es que las señales de reloj en el circuito se deben distribuir de manera adecuada para evitar un sesgo en la señal del reloj, lo cual afectaría severamente el funcionamiento del circuito. En algunas ocasiones no se encuentra una ruta o distribución adecuada de la señal de reloj en el *layout*, por lo que se debe repetir el diseño de las celdas; esto ha ocasionado que los diseñadores prefieran hacer el ruteo de la señal del reloj antes de hacer un *Place and Route* [9].

Extracción de parásitos

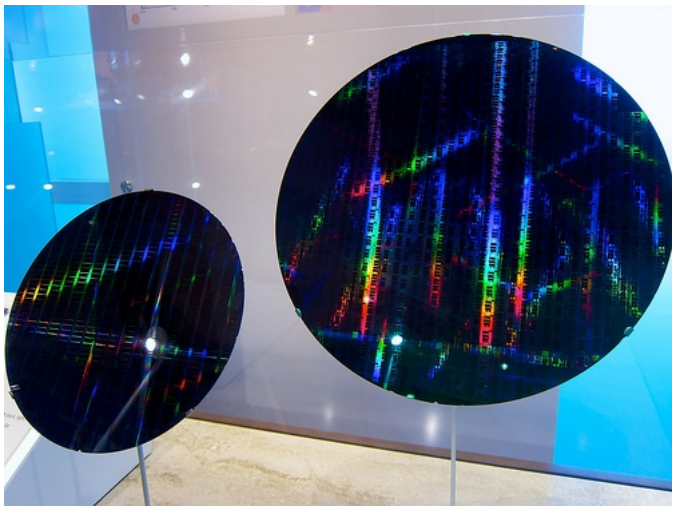
Otro paso de gran importancia es la extracción de parásitos de circuito generado en el *layout*. Durante este paso, se crea un nuevo *netlist* con las resistencias y capacitores asociados a todas las características físicas intrínsecas de los nets en el *layout*. El extractor de parásitos puede ser 2D o 3D. En una extracción 2D, la herramienta analiza la sección transversal asumiendo que todos los cables se extienden uniformemente fuera de la sección. Por otra parte, la extracción en 3D resuelve las ecuaciones de Maxwell en tres dimensiones para determinar con precisión la capacitancia de geometrías complejas [9].

Posteriormente, se vuelve a realizar un análisis de tiempo. En esta etapa, el análisis será más exacto pues ya se toman en cuenta las cargas conectadas a las compuertas en el circuito. Se realizan varias iteraciones para converger a los requerimientos necesarios de *timing*. En este punto se realizan análisis para revisar ruido, caída de IR en las líneas de distribución de energía al circuito y los límites de electromigración, es decir, cuán desgastados están las interconexiones después de tantas pruebas [9]. Finalmente, la estimación de disipación y consumo de potencia se realiza con los datos obtenidos de la extracción de parásitos, consiguiendo resultados más exactos [9].

6.3. Fabricación de un circuito integrado

Los circuitos integrados son fabricados en láminas delgadas circulares de silicio comúnmente conocidas como obleas de silicio (o *wafers* en inglés). Cada una de estas obleas puede contener miles de circuitos integrados o dies, en función del tipo de circuito diseñado. El diámetro de estas obleas puede rondar en unos 300 mm [13]. Todo este proceso de fabricación es realizado por un *foundry*, posterior a la obtención de un diseño proveniente de una etapa de diseño con los pasos mencionados anteriormente.

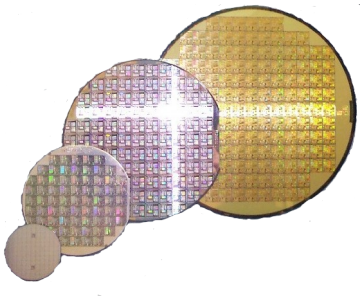
Figura 8: Oblea de silicio utilizada para fabricar un circuito integrado



6.3.1. Fotolitografía

La fotolitografía es el proceso en el que se selecciona áreas de interés en una oblea de silicio durante el proceso de manufactura de un circuito integrado, con el fin de definir regiones para los dopantes, el polisilicio, los metales y los contactos para el producto final. Estas áreas se seleccionan con la utilización de fotorresistencias [13].

Figura 9: Oblea de silicio después de un proceso fotolitográfico



El proceso de fotolitografía se resume en los siguientes pasos:

- Capas de óxido.
  Si bien este paso es opcional, es muy importante su utilización, pues se deposita una pequeña capa delgada de óxido de silicio sobre la oblea completa; y básicamente permite la formación de las compuertas de los transistores [13].

- Revestimiento de fotorresistencia.
  Mientras se hace girar la oblea de silicio se le aplica un polímero sensible a la luz, que originalmente es soluble en un solvente orgánico, pero tiene la propiedad de que al ser expuesto a la luz esas regiones se vuelven insolubles. Este tipo de polímero se llama fotorresistor negativo. Un polímero que cumple la propiedad inversa, es decir, soluble cuando se expone a la luz, es llamado fotorresistor positivo [13].

- Exposición.
  Un retículo de vidrio en el que se incluyen patrones que se quieren transferir al silicio, se acerca la oblea que se está procesando. Posteriormente, la máscara se opaca en las regiones a procesar, y se transparenta en las otras [13].

- Exposición al horno
  En este paso las obleas se exponen a un ácido o a una base para suprimir las áreas que no se han expuesto a la fotorresistencia. Después se remueven las fotorresistencias expuestas y la oblea está "horneada" a una temperatura baja, de modo de poder sostener las fotorresistencias restantes [13].

- Grabado de ácido.
  Se remueve el material de las áreas de la oblea que no han sido cubierta por el material fotorresistivo, mediante la utilización de ácidos, bases, y demás sustancias en función del material escogido [13].

- Girar y secar.
  En este paso se limpia la oblea de silicio con agua desionizada y posteriormente es secada con nitrógeno. Es importante destacar el hecho de que esta limpieza debe ocurrir en salones ultra limpios donde la cantidad de partículas de polvo por pie cúbico sea muy pequeña; esto porque en la época moderna de los semiconductores y circuitos integrados, la menor partícula de polvo puede destruir las propiedades de la oblea y por tanto del circuito a fabricar [13].

Posteriormente el área expuesta ya puede ser manipulada en los procesos de adición de impurezas para la fabricación de circuitos integrados.

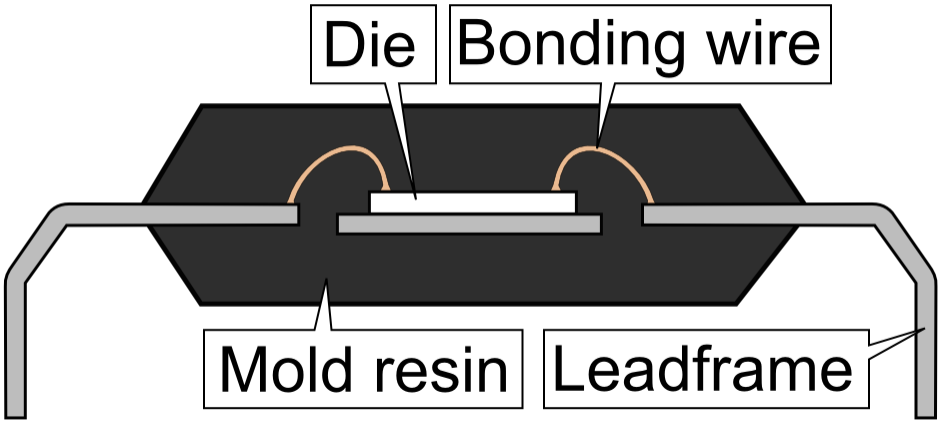
6.3.2. Empaquetado del circuito integrado

El empaquetado es una parte muy importante del diseño y fabricación de circuitos integrados, así como de su funcionamiento, pues provee una parte esencial de la transferencia de señal dentro del integrado y para la interconexión de los diferentes componentes dentro de él. Así mismo, el recubrimiento permite remover el calor generado, proveer soporte mecánico y proteger el circuito de daños y de humedad u otros factores externos que pudieran dañar el *die*. Además, la tecnología de empaquetado también tiene un impacto importante en el desempeño y la disipación de potencia del circuito, lo cual en los últimos años ha llevado a los productores a buscar nuevas formas de empaquetamiento; tal que el *delay* en las señales, causado por los materiales de empaquetado, sea menor utilizando capacitores e inductores especiales [14]. Por tanto, un buen empaquetado debe cumplir con una amplia variedad de requerimientos, tales como:

- Requerimientos eléctricos: los pines del recubrimiento deben tener poca capacitancia, resistencia e inductancia.
- Propiedades térmicas y mecánicas: la tasa de eliminación de calor debe ser alta, y aparte la reliabilidad mecánica requiere un buen acoplamiento entre las propiedades térmicas del integrado.
- Costo bajo: el costo del material de empaquetado es un punto clave para el desempeño del circuito integrado: aunque la cerámica tenga un desempeño superior al de los plásticos, la cerámica es más cara. También la cantidad de pines necesitados para un chip requiere un material más especial y caro, por lo que se debe analizar minuciosamente la aplicación que se le dará al chip para escoger el material de empaquetado más adecuado en relación al desempeño vs costo.

El empaquetado más común utiliza una estrategia de interconexión de dos niveles. La matriz del integrado se une en primer lugar a un soporte del chip o sustrato individual, y el cuerpo del envase tiene una cavidad en donde se coloca el *die*, la cual proporciona amplio espacio para conexiones de los circuitos integrados. El segundo nivel de interconexión lo conforman los conductores, los cuales conectan al chip al medio externo, que generalmente es cualquier dispositivo electrónico en el que se vaya a utilizar el integrado [14].

Figura 10: Empaquetado típico de un circuito integrado



Materiales del empaquetado

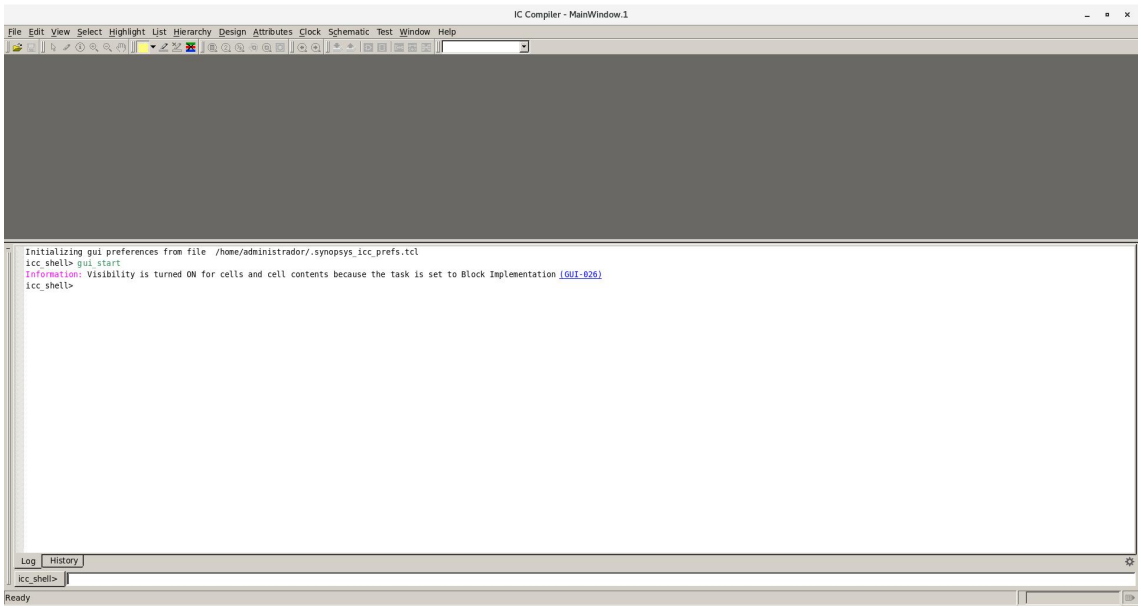
Los materiales más comunes utilizados para el recubrimiento del circuito integrado son cerámica y polímeros (o plásticos) [9]. Los plásticos son sustancialmente más baratos, pero carecen de propiedades térmicas que sí se encuentran en materiales cerámicos. Por ejemplo, el Al2O3 cerámico conduce el calor de una mejor manera que el SiO2 y su coeficiente de expansión térmica está sustancialmente más cerca de la de los metales de interconexión típicos [11], generando una mejor conducción eléctrica.

6.4. Síntesis física con *IC Compiler*

IC Compiler es un software desarrollado por la empresa estadounidense *Synopsys*, que permite realizar la síntesis física de diseños de circuitos electrónicos. *IC Compiler* incluye procesos innovativos para realizar todos los pasos de la síntesis física de forma eficiente, rápida y con resultados de alta calidad, lo que lo ha posicionado en la industria como un software líder para tales procesos. Esta herramienta funciona tanto en un entorno gráfico como en un entorno de comandos conocido como *Tool Command Language* (TCL).

La planeación de diseño es algo muy importante que los diseñadores de circuitos integrados deben seguir en un flujo de diseño. Esto se refiere a evaluar la factibilidad de diferentes estrategias de implementación de diseños en la síntesis física. Por ejemplo, para diseños más grandes, este proceso es crítico pues los diseños más grandes tienden a mostrar más problemas por interferencias, *delays* que hacen imposible cumplir con requerimientos de *timing*; por tanto se crea la necesidad de tener cuidado con una planificación de diseño temprana. *IC Compiler* soporta diferentes opciones para un diseño óptimo que permite exploraciones rápidas del diseño con el fin de reducir el tamaño del *die* e implementar un circuito optimizado y detallado [15].

Figura 11: Interfaz gráfica y línea de comandos en *IC Compiler*



La herramienta de *IC Compiler* permite que el usuario realice implementaciones físicas de diseños a muy gran escala, proveyendo las siguientes funcionalidades:

1. Inicialización del diseño
   Para inicializar un diseño se deben realizar tareas como las siguientes: importar el diseño proveniente de la síntesis lógica (extensión .ddc o .v), importar la librería, importar las restricciones de diseño, definir restricciones de tiempo, definir requerimientos de voltaje y/o crear el *floorplan*.

2. Optimización y colocación
   Durante este proceso, *IC Compiler* determina una locación física que sea adecuada para las celdas del diseño en función de las restricciones especificadas en el paso anterior.

3. Síntesis de la señal de reloj
   En este paso la herramienta trata de balancear el desfase entre las señales de reloj (si aplica en el diseño), en función de las restricciones de tiempo especificadas.

4. Enrutamiento
   En este paso la herramienta conecta todos los nodos y pines del circuito, en función de las restricciones y las reglas de diseño.

5. Optimización post-enrutamiento
   Después de las conexiones iniciales en el diseño, durante el paso de enrutamiento, en esta etapa se optimiza el diseño enrutado; en función de restricciones de tiempo, congestión, y restricciones de voltajes.

6. Finalización del circuito
   En esta etapa se agregan capacitores, metales y celdas para terminar el diseño. También se pueden hacer análisis de electromigración.

7. Escritura de datos
   En esta etapa final, el diseño es guardado y se generan archivos de reporte, base de datos del diseño, librería del diseño, y los archivos GDSII para enviar al *foundry*.

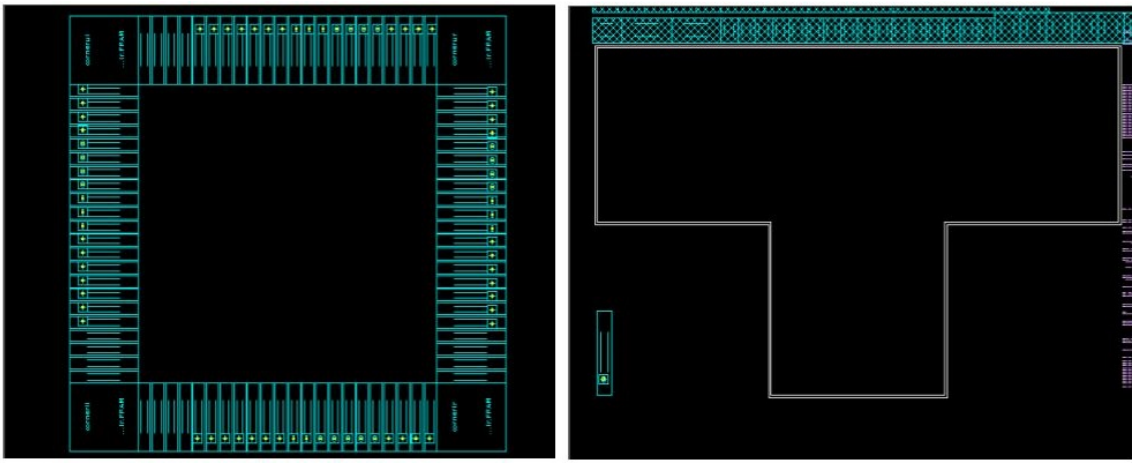
6.4.1. *Floorplanning* y *Placement* con *IC Compiler*

IC Compiler crea y refina un *floorplan* desde un *netlist* existente. Con esto se describe y crea el tamaño del núcleo del circuito, así como su forma y la posición para las celdas, restricciones y la ubicación de los pines de entrada/salida y alimentación sobre la periferia del circuito. *IC Compiler* permite crear una conexión lógica de los pines de alimentación y de tierra del diseño con las celdas, así como restricciones a los *pads* de entrada o salida; es decir, su orientación, posicionamiento o espaciado entre ellos [15].

Posteriormente, el diseñador está habilitado a crear el *floorplan* inicial. El *floorplan* puede ser de forma rectangular o rectilinear. Estos dos tipos de *floorplan* se muestran en la Figura 12.

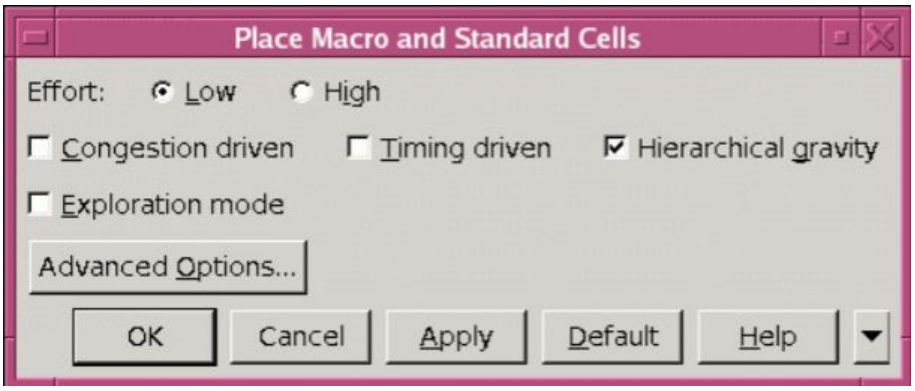
El *floorplan* puede ser personalizado a criterio del diseñador: modificar el ancho y el largo o la cantidad de filas para las celdas, así como cargar las restricciones en los pines de entrada y salida para realizar su posicionamiento [15].

Figura 12: *Floorplan* rectangular y rectilinear en *IC Compiler* [15]



Con *IC Compiler* se puede realizar un *Placement* simultáneo de celdas para el *chip* entero. Se inicia con una colocación plana inicial rápida realizada con fines de planificación del diseño. Para grupos o áreas de voltaje o celdas que aún no se han colocado, este *Placement* inicial es una herramienta que ayuda a decidir sobre las ubicaciones, los tamaños y las formas de bloques físicos de nivel superior. Esta ubicación es virtual, pues considera temporalmente el diseño para ser completamente plano, sin jerarquía. Después de decidir sobre las formas y ubicaciones de los bloques físicos, restaura la jerarquía de diseño y procede con el bloque por bloque del flujo de diseño físico. El *Placement* virtual plano puede controlarse para enfatizar las cualidades más importantes necesarias para el diseño en implementación, como el *timing* o la ubicación de pines [15]. El *Placement* se realiza típicamente después que se ha iniciado el *Floorplan* en la herramienta.

Figura 13: Ventana de *Placement* en *IC Compiler*

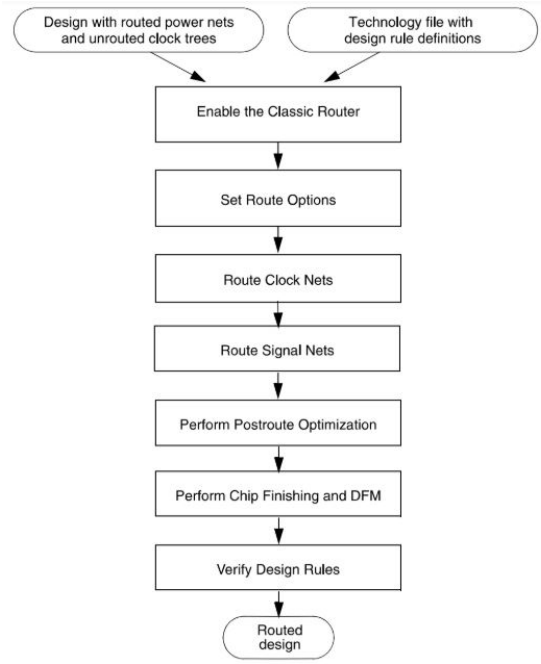


La interfaz principal del *Placement* se muestra en la Figura 13. Para realizar la colocación de celdas se puede seleccionar un modo *low effort*, en el que se producen buenos resultados de colocación para grupos grandes de celdas; y el modo *high effort*, en el que se consigue mejor calidad de resultados con el costo de mayor espera. De igual forma se pueden seleccionar diferentes tipos de optimización.

6.4.2. *Routing* con *IC Compiler*

IC Compiler contiene dos tipos de *ruteo*, el más común es el conocido como *Classic Router*, que es utilizado para tecnología de fabricación arriba de 45 nm [16]. En la Figura 14 se muestra el flujo de diseño básico que se puede realizar con *IC Compiler* para el *ruteo*.

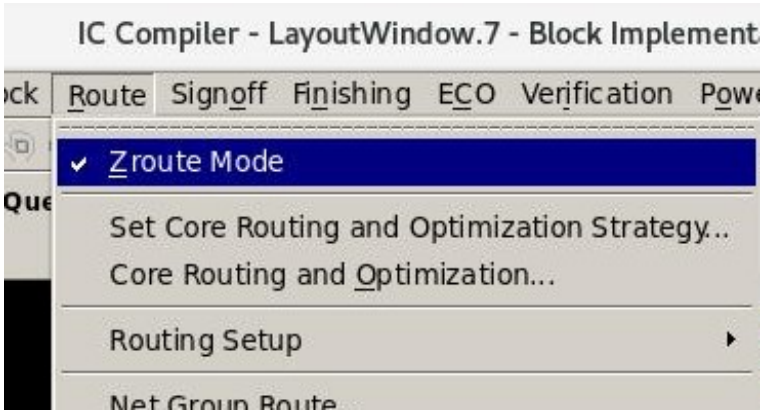
Figura 14: Flujo de ruteo básico con *IC Compiler*



Antes de iniciar un proceso de *ruteo*, *IC Compiler* sugiere que se cumplan las siguientes condiciones: los nodos y líneas de alimentación y de tierra deben estar conectadas antes de haber realizado la colocación de los elementos del circuito, haber realizado una optimización de la señal de reloj y que la congestión, el *timing* y la capacitancia máxima sean aceptables. *IC Compiler* ofrece la posibilidad de verificar que el diseño esté listo para ser procesado para *Routing* mediante la opción de *Check Routability*. Dicho proceso se ejecuta con el comando *check_routeability* en la línea de comandos, y si ocurre algún error, la herramienta genera un archivo detallando los errores encontrados [16].

Para habilitar el *Classic Router*, se debe deshabilitar la opción de *Zroute*, que es un proceso de *ruteo* para tecnologías de fabricación más complejas. En la interfaz de comandos se debe escribir *set_route_mode_options -zroute false* para habilitar el *Classic Router* o mediante la interfaz gráfica de la siguiente forma:

Figura 15: Habilitando el *Classic Router* en *IC Compiler*



IC Compiler permite que el diseñador pueda decidir qué capas del diseño (*layout*) sean habilitadas o deshabilitadas para realizar el *ruteo*, así como especificar qué capas sean únicamente para ciertos *nets* o nodos. Posterior a este tipo de restricciones, el diseño puede proseguir para hacer el *ruteo* de todos los nodos y señales. *IC Compiler* contiene diversas opciones de *ruteo*, lo que lo hace una herramienta muy flexible para cualquier tipo de diseño. El diseñador puede escoger primero las señales más críticas, como la señal de reloj, para rutearla primero y encontrar problemas. Por ejemplo, al tener toda la señal de reloj *ruteada* antes que cualquier otra señal, es posible hacer un análisis de *timing* para comparar con las restricciones de diseño [16].

Antes de *rutear* las otras señales, *IC Compiler* obliga que cualquier nodo identificado como señal de reloj esté ruteado previamente. Para rutear las demás señales se pueden utilizar diferentes métodos que provee *IC Compiler*:

- *Ruteo* global con asignación automática de *tracks*, reparación y una optimización post-ruteo.
- *Ruteo* independiente, para diseñadores más detallistas que quieran realizar la tarea de ruteo paso por paso y con una customización definida.

6.4.3. Entorno de Comandos (TCL)

Una de las ventajas que provee *IC Compiler* en su entorno TCL, es la automatización de los procesos mediante un *script* o código, que se puede cargar y ejecutar en la interfaz gráfica [17].

Script de *setup*

En este *script* se carga el diseño en *Verilog* proveniente de la parte de síntesis lógica en extensión .ddc, las librerías de la tecnología de fabricación a utilizar en extensión .tf, archivos de reglas de diseño en extensión .db, y archivos en extensión .tluplus para características físicas avanzadas de las celdas en relación a la extracción de parásitos. En el *script* se utilizan los siguientes comandos:

Comando	**Descripción**
lappend	Agrega elementos a una variable definida
link_library	Especifica la librería (extensión .db) utilizada en el diseño sintetizado lógicamente a enlazar
target_library	Especifica la librería (extensión .db) que contiene los componentes a utilizar durante la compilación del diseño
set_tlu_plus_-files -max_tluplus_file	Especifica el archivo TLUPLUS utilizado para el cálculo de la condición máxima de resistencia y capacitancia para la extracción de parásitos.
set_tlu_plus_-files -min_tluplus_file	Especifica el archivo TLUPLUS utilizado para el cálculo de la condición mínima de resistencia y capacitancia para la extracción de parásitos.
create mw_library	Crea una librería de Milkiway, la base de datos de diseños físicos y de herramientas de *Synopsys*
open mw_	Crea una librería de Milkiway, la base de datos de diseños físicos y de herramientas de *Synopsys*
open mw_library	Abre una librería de la base de datos de Milkiway
import_designs	Importa el diseño especificado, ya sea en extensión .v (*Verilog*) o .ddc
save_mw_cel	Guarda el diseño de Milkiway actual en formato de Milkiway.
close_mw_cel	Guarda el diseño de Milkiway actual en formato de Milkiway y posteriormente lo cierra
close_mw_lib	Guarda la librería de Milkiway actual, indicando si hubo error o no

Cuadro 1: Comandos utilizados en *setup.tcl* [17]

Script de *design*

En este *script* se carga la librería de Milkiway con el diseño configurado para realizar el *floorplanning*, *placement* y *routing*. Por otra parte, se listan comandos con sus restricciones respectivas para crear los anillos rectangulares de VDD y GND, para realizar el *floorplanning* de forma optimizada, para la ubicación optimizada de los pines de entrada y salida, para realizar una colocación de las celdas del diseño de forma optimizada, para realizar un ruteo eficiente en función de una ruta optimizada para la señal de reloj, y para crear restricciones sobre las capas de metal utilizadas. En el *script* se utilizan los siguientes comandos:

Comando	**Descripción**
open mw_library	Abre una librería de la base de datos de Milkiway
copy_mw_cel	Copia los diseños de Milkiway hacia la librería destino especificada
open mw_cel	Abre una librería de la base de datos de Milkiway y la establece como la librería actual a editar y diseñar
current mw_cel	El comando reporta el diseño de Milkiway en el que se está trabajando actualmente
create_floorplan	El comando realiza floorplanning en el diseño actual. Dependiendo de los argumentos, el comando puede crear restricciones sobre la forma de colocar las celdas dentro del diseño, distancias entre celdas y los pads, etc.
derive_pg_connection	Realiza la conexión de VDD y GND a las nets del diseño especificadas para voltaje y tierra.
create_rectangular_rings	Crea anillos rectangulares en el diseño. El comando crea de forma automática el anillo rectangular en la posición legal más cercana permitida, con el fin de no cometer errores que aparezcan en el DRC.
create fp_placement	Realiza una colocación plana virtual de las celdas del diseño. Con la opción de **-optimize-pins**, se realiza una asignación de los pines en relación a las restricciones de pines, es decir, restricciones de los costados del diseño.
set_ignored_layers	Restringe el routing únicamente entre las capas de metal especificadas.
clock_opt	Realiza síntesis de la señal de reloj, el routing de los nodos de reloj, extracción y optimización.
route_opt	Realiza routing y optimización simultánea en el diseño actual. Entre las funciones que puede realizar se encuentra el routing global, asignación de rutas, búsqueda y reparación y optimización de señales.

Cuadro 2: Comandos utilizados en *design.tcl* [17]

6.4.4. Entorno gráfico

Entorno gráfico para *setup*

En la interfaz gráfica de *IC Compiler*, en la pestaña de *File*, se encuentran todas las opciones relacionadas al paso de inicialización del diseño descrito en la sección anterior. En las siguientes figuras se muestra cómo acceder a ellas para configurar las librerías e importar el diseño .ddc.

Figura 16: Menú de *File* en *IC Compiler*

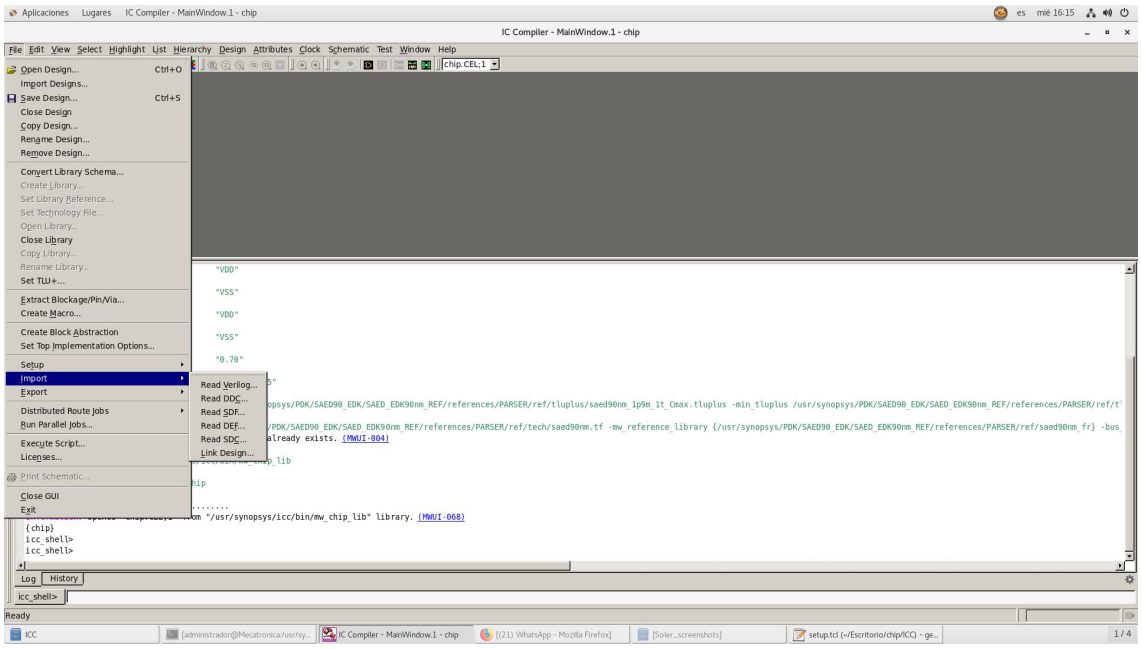


Figura 17: Menú de carga de archivos .tluplus

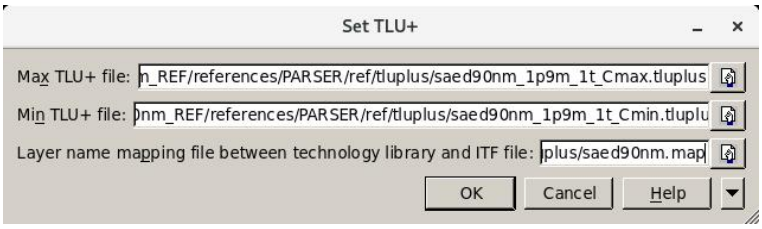


Figura 18: Menú de carga del archivo de la síntesis lógica, en extensión .ddc

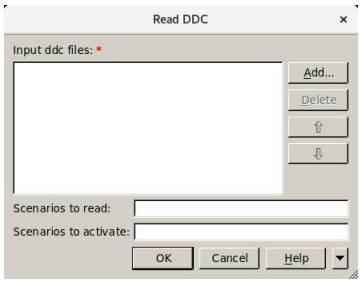


Figura 19: Menú de carga del archivo de la tecnología de fabricación utilizada

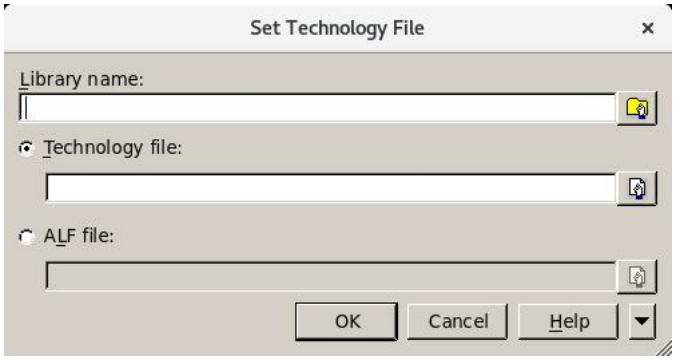
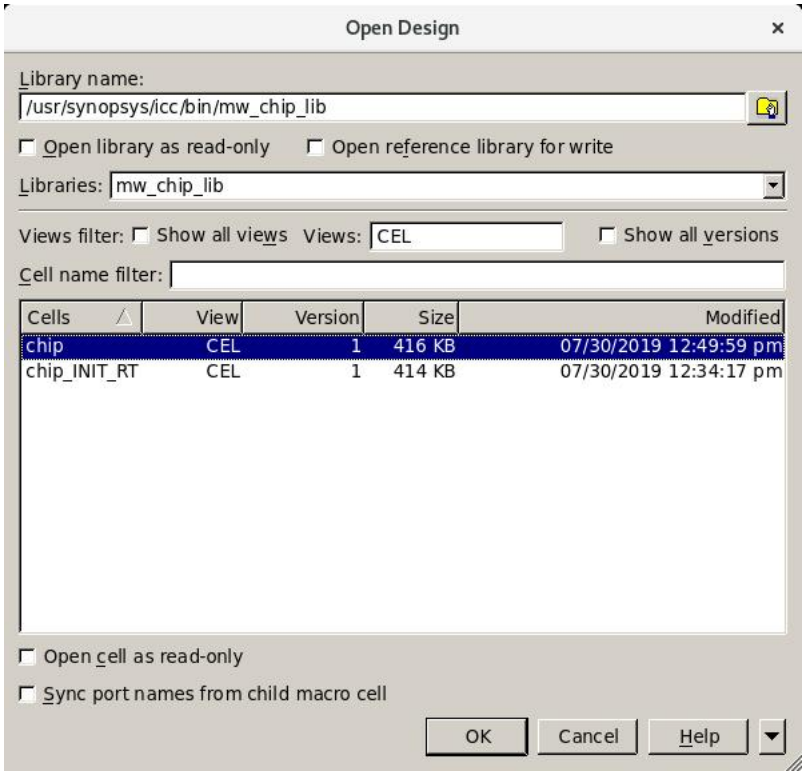
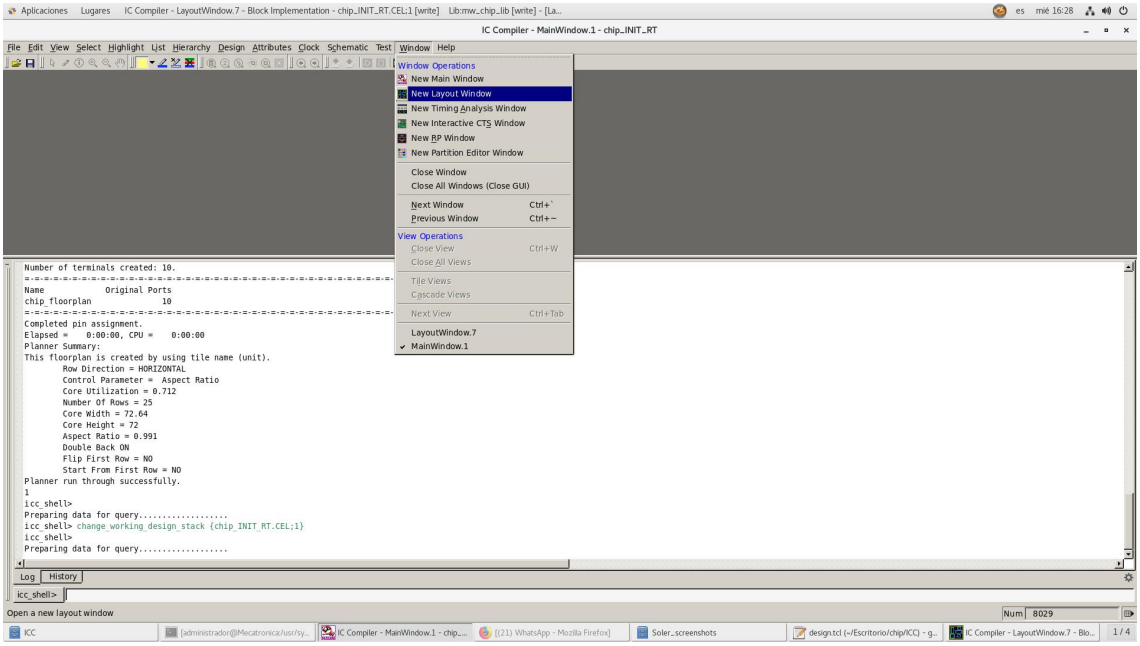


Figura 20: Menú para abrir una librería de diseño



Una vez cargada las librerías, archivos de configuración y archivo .ddc, el usuario está habilitado a ingresar a la interfaz gráfica propiamente de *layout* dentro de *IC Compiler*, para iniciar los pasos de la síntesis física descritos en la sección 6.2.2 [17]. Si bien se podría utilizar un *script* de *design*, como el descrito con anterioridad, la interfaz gráfica permite un manejo más amigable de las configuraciones que se pueden hacer para realizar la síntesis física. En la siguiente figura se muestra cómo ingresar al entorno gráfico de *layout*.

Figura 21: Menú para abrir el entorno de *layout*



Entorno gráfico para *design*

En la interfaz gráfica de *IC Compiler*, una vez abierto el entorno de *layout*, se pueden seleccionar las opciones para iniciar el flujo de diseño de la síntesis física.

Para el *Floorplanning* y el *Placement* de las celdas, se muestra en las siguientes figuras las opciones para acceder a los diferentes pasos mencionados en la sección 6.4.1.

Opción para conectar lógicamente los nodos de alimentación y de tierra del diseño.

Figura 22: Conectar alimentación y tierra

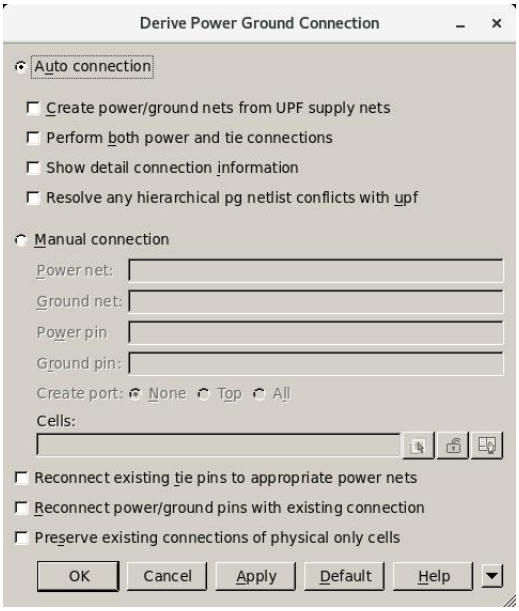


Figura 23: Menú para abrir las opciones de *Floorplan*

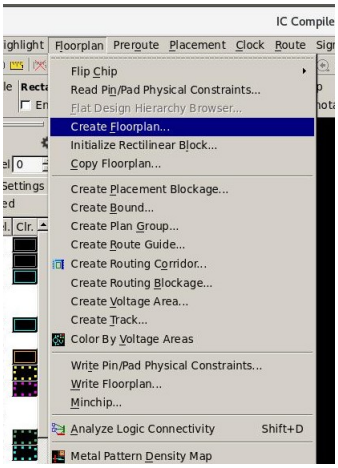


Figura 24: Opciones de *Floorplan*

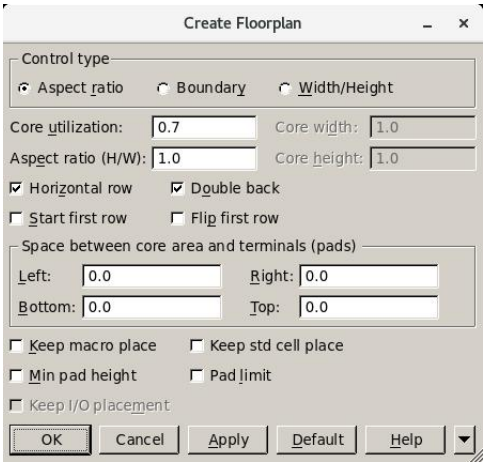
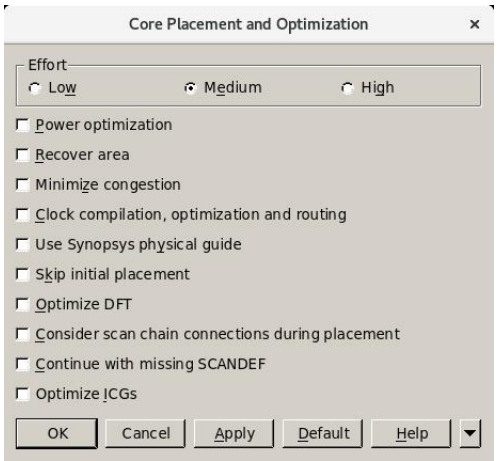


Figura 25: Opciones de *Placement*



En la Figura 25 se muestra cómo se pueden seleccionar los diferentes tipos de optimización para la colocación de celdas, y el tipo de esfuerzo para su colocación: *high* o *low*.

Para el *Routing* de las celdas, se muestra en las siguientes figuras las opciones para acceder a los diferentes pasos mencionados en la sección 6.4.2.

Ventana para seleccionar las capas de metal a ignorar. Como se mencionó anteriormente, el diseñador tiene la flexibilidad de escoger a su criterio las capas dentro del *layout* que desee ignorar o agregar, con el fin de optimizar su diseño.

Figura 26: Menú de *Route* e *Ignored Layers*

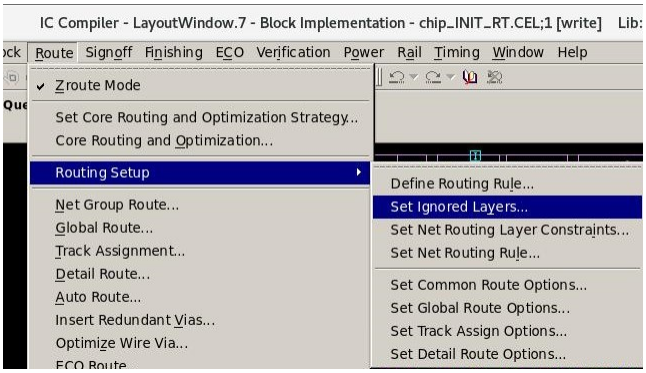
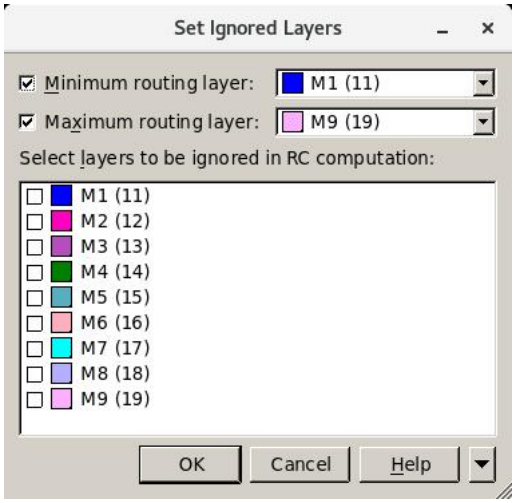
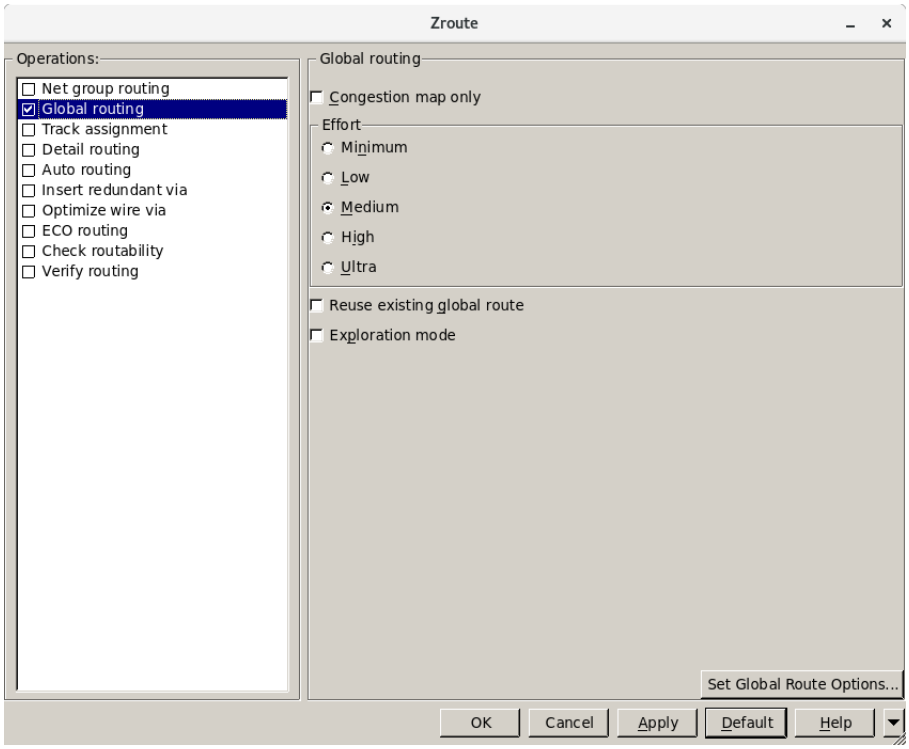


Figura 27: Opción para ignorar capas



Posteriormente se encuentran todas las opciones que provee *IC Compiler* para realizar el ruteo de las celdas y bloques dentro del *die* del circuito.

Figura 28: Opciones para *Routing*



CAPÍTULO 7

Herramientas necesarias para la síntesis física

7.1. Selección de fabricante y librerías

Los circuitos en VLSI deben tener bien definido la tecnología de fabricación en la que fueron diseñados, las reglas de diseño, restricciones físicas y características físicas. Todo esto va incluido en las librerías de diseño o PDK (*Design Process Kit)*, la cual puede variar en función del proceso de fabricación que se utilice, por tanto se debía seleccionar conjuntamente con una empresa fabricante de semiconductores y circuitos integrados.

Al ser estudiantes de pregrado de una universidad, se debía abocar a una empresa que prestara y facilitara servicios de fabricación en programas universitarios, por lo que se seleccionó a la empresa europea EUROPRACTICE, quienes se dedican a trabajar proyectos de fabricación y diseño de circuitos integrados con universidades e institutos de investigación alrededor del mundo; siendo el enlace entre el cliente (Universidad) y la empresa fabricante. Se creó un contrato legal Universidad-EUROPRACTICE para el acceso a PDKs de tecnología de fabricación de 180 nm de TSMC.

7.2. Selección de software para la síntesis física

El departamento de Ingeniería en Electrónica, Mecatrónica y Biomédica de la Universidad del Valle de Guatemala, cuenta con el acceso a *SolvNet* de *Synopsys*, el cual es un servicio que permite la obtención de diferentes recursos relacionados a software especializado de diseño de circuitos electrónicos. Esto incluye la descarga de herramientas, licencias, documentación y soporte técnico. Entre las diversas herramientas que provee *Synopsys* se encuentra *IC Compiler*, el cual es un software especializado que permite realizar la síntesis

física de diseños electrónicos.

Una vez descargado *IC Compiler* desde *SolvNet*, se procedió a investigar en la documentación de la herramienta los pasos y las funcionalidades necesarias para el objetivo del proyecto. La documentación incluye la explicación de todos los pasos específicos de un flujo de diseño completo, entiéndase optimización de rutas, optimización en la colocación de las celdas de silicio, requerimientos de *timing* y de potencia, optimización de consumo, optimización de la señal de reloj, etc.; sin embargo, con el fin principal de realizar el diseño del primer circuito integrado en la historia de Guatemala, no se busca ninguna meta específica de potencia, tiempo de respuesta u otro requerimiento a fondo; por tanto, se investigó y recolectó información sobre los pasos de la síntesis física de *Floorplanning*, *Place and Route*, y optimización de rutas de forma general. Todo esta recolección de información se realizó consultando la documentación oficial que provee *Synopsys* sobre *IC Compiler*, encontradas en las referencias [16], [15] y [17].

IC Compiler, como todas las herramientas de *Synopsys*, funcionan tanto en un entorno gráfico como en un entorno de líneas de comandos. El entorno de líneas de comandos es útil para la automatización de ciertos procesos de la síntesis física, por lo que se enfatizó la recolección de comandos y sus funcionalidades para realizar un *script* que realizara la síntesis de forma automática, una vez definido ciertos requerimientos generales del diseño. Se realizaron los *script* de *setup* y *design* (sección 6.4.3), descritos anteriormente, con el fin de automatizar la síntesis física realizada en este Proyecto de Graduación.

CAPÍTULO 8

Circuito seleccionado para realizar la síntesis física

El circuito a implementar en la síntesis física consiste en una máquina de estados finitos que genera el siguiente mensaje en codificación binaria:

"SOY EL PRIMER CHIP CON TECNOLOGIA NANOMETRICA DISEÑADO EN LA HISTORIA DE CENTROAMERICA. MIS CREADORES SON: CARLOS ESQUIT, DIEGO SOLER CASTAÑEDA, STEVEN HIRAM RUBIO, LUIS ARTURO NAJERA Y PABLO ORTIZ BARILLAS. DEPARTAMENTO DE INGENIERIA ELECTRONICA. UNIVERSIDAD DEL VALLE DE GUATEMALA. 2019"

El esquemático en formato de caja negra de la máquina de estados finitos se muestra en la Figura 30. Las entradas y salidas del circuito se distribuyen de la siguiente manera:

Entradas	**Salidas**
VDD	q[0]
GND	q[1]
RESET	q[2]
CLK	q[3]
	q[4]
	q[5]
	q[6]
	q[7]

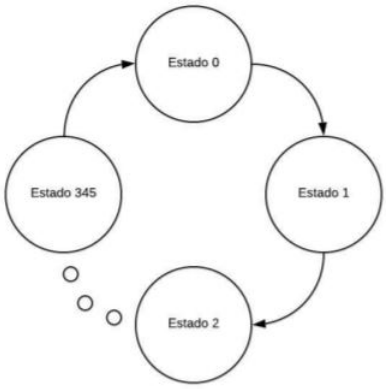
Cuadro 3: Entradas y salidas del circuito implementado en la síntesis física

Los pines de entrada consisten en 4 señales: una señal de reloj (`CLOCK`), una señal de *reset* (`RESET`), señal de alimentación (`VDD`) y señal de tierra (`GND`). Los pines de salida son 8,

los cuales representan los 8 bits correspondientes a cada carácter del mensaje generado. La máquina de estados finitos funciona de la siguiente manera: 345 estados basados en los 345 caracteres contenidos en el mensaje mostrado anteriormente, de forma secuencial y recursiva; es decir, transiciones ordenadas que recorren el mensaje carácter por carácter. La forma en la que ocurren las transiciones se muestran en la Figura 29. De la misma forma se muestra en el Cuadro 4 un fragmento de las transiciones de los 345 estados y la cadena de bits que representa cada carácter.

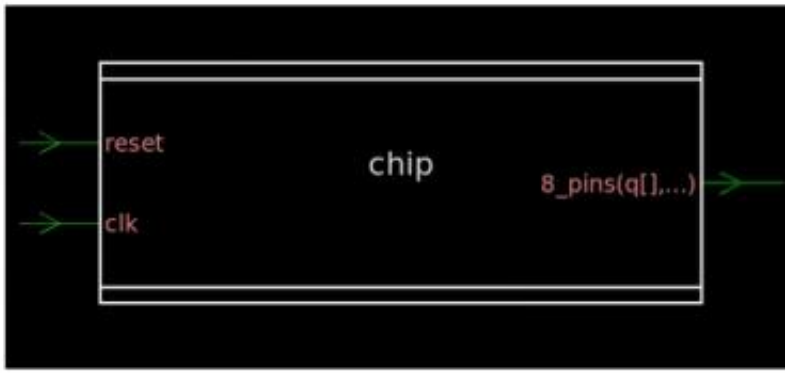
Posteriormente el circuito se conectará a un serializador para leer la cadena de caracteres generada en *Python*, y finalmente codificar el mensaje a audio y reproducirlo.

Figura 29: Transiciones de estado de la máquina de estados finitos diseñada



El acceso a las librerías finales para realizar el flujo de diseño completo, es decir, desde la síntesis lógica hasta la generación de los archivos GDSII necesarios para el *foundry*; no se obtuvo desde el inicio de este trabajo de graduación, por lo que se realizaron pruebas preliminares con librerías de tecnología de fabricación de 90 nm de *Synopsys*, incluídas en el *bundle* universitario que ofrecen y con el que cuenta el Departamento de Ingeniería en Electrónica de la Universidad del Valle. Se realizó parte del flujo de diseño completo con estas librerías para comprender en detalle los pasos necesarios, principalmente en el proceso de síntesis física.

Figura 30: Esquemático en caja negra del circuito a sintetizar



Estado actual			Estado siguiente		
Caracter	No. de estado	Binario	Caracter	No. de estado	Binario
S	0	000000000	O	1	000000001
O	1	000000001	Y	2	000000010
Y	2	000000010	ESPACIO	3	000000011
ESPACIO	3	000000011	E	4	000000100
E	4	000000100	L	5	000000101
L	5	000000101	ESPACIO	6	000000110
ESPACIO	6	000000110	P	7	000000111
P	7	000000111	R	8	000001000
R	8	000001000	I	9	000001001
I	9	000001001	M	10	000001010
M	10	000001010	E	11	000001011
E	11	000001011	R	12	000001100
R	12	000001100	ESPACIO	13	000001101
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
2	339	101010011	0	340	101010100
0	340	101010100	1	341	101010101
1	341	101010101	9	342	101010110
9	342	101010110	.	343	101010111
.	343	101010111	ESPACIO	344	101011000
ESPACIO	344	101011000	ESPACIO	345	101011001
ESPACIO	345	101011001	S	0	000000000

Cuadro 4: Tabla de transición de estados por caracter de la máquina de estados finitos diseñada

CAPÍTULO 9

Pruebas preliminares con librerías de 90 nm

9.1. Síntesis lógica

Antes de iniciar la síntesis física, se necesita obtener un diseño proveniente de la síntesis lógica con extensión *Verilog* (.v) o .ddc, como se mencionó en la sección 6.4. Una vez teniendo definido el funcionamiento del circuito, se realizó la descripción del circuito en un HDL. Para ello se implementó a nivel *behavioral* el circuito en *Verilog*. En esta parte se diseñó un *script* en *Verilog* que contara los 345 estados de la máquina de estados finitos diseñada, y en función del estado en el que está; encendiera o apagara los bits correspondientes a la representación binaria del carácter ASCII correspondiente en los 8 bits de salida.

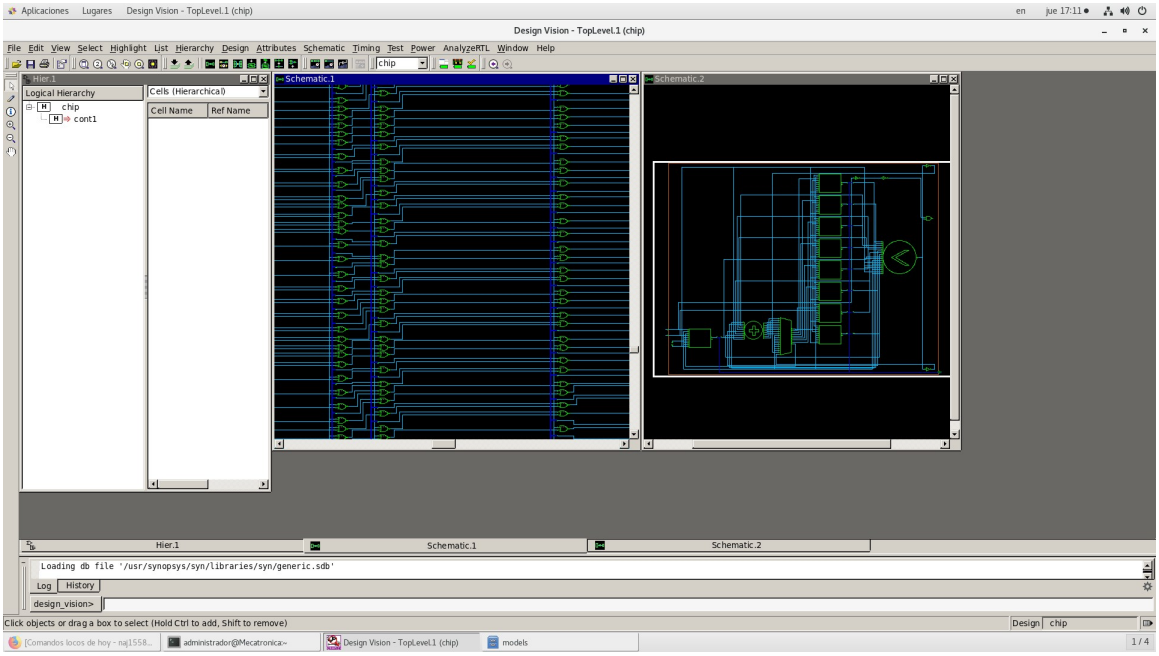
Posteriormente, este diseño se trasladó a la herramienta de *Synopsys* Design Vision, para realizar la síntesis lógica. Dicha implementación se ve en la Figura 31.

Realizado el proceso en Design Vision, se obtuvo un *netlist* de compuertas del circuito caracterizadas con sus celdas respectivas de la librería de 90 nm en extensión .ddc, listo para implementarse en la síntesis física.

9.2. Síntesis física

Con el *netlist* obtenido en la síntesis lógica con Design Vision, se procedió a iniciar la síntesis física con la herramienta *IC Compiler* de *Synopsys*. La herramienta necesita archivos .tluplus, .db y .tf, contenidos en la librería de 90 nm; para realizar la inicialización del diseño, descrita en la sección 6.4.

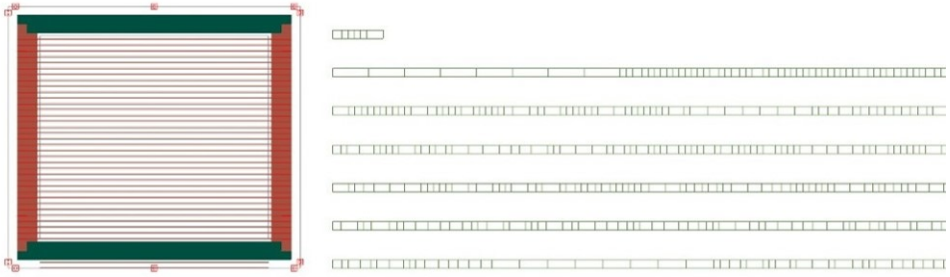
Figura 31: Utilización de *Design Vision* en la síntesis lógica



Seguidamente se importó el diseño proveniente de la síntesis lógica para inicializar los procesos de *Floorplanning* y *Placement & Route*. Se especificó un *search_path*, con el fin de ubicar todos los archivos necesarios para la inicialización del diseño. Dicho *search_path* utilizado para todos los archivos relacionados a la librería de 90 nm fue `/usr/Synopsys/PDK/SAED90_EDK/SAED_EDK90nm_REF/references/PARSER/ref/models/`

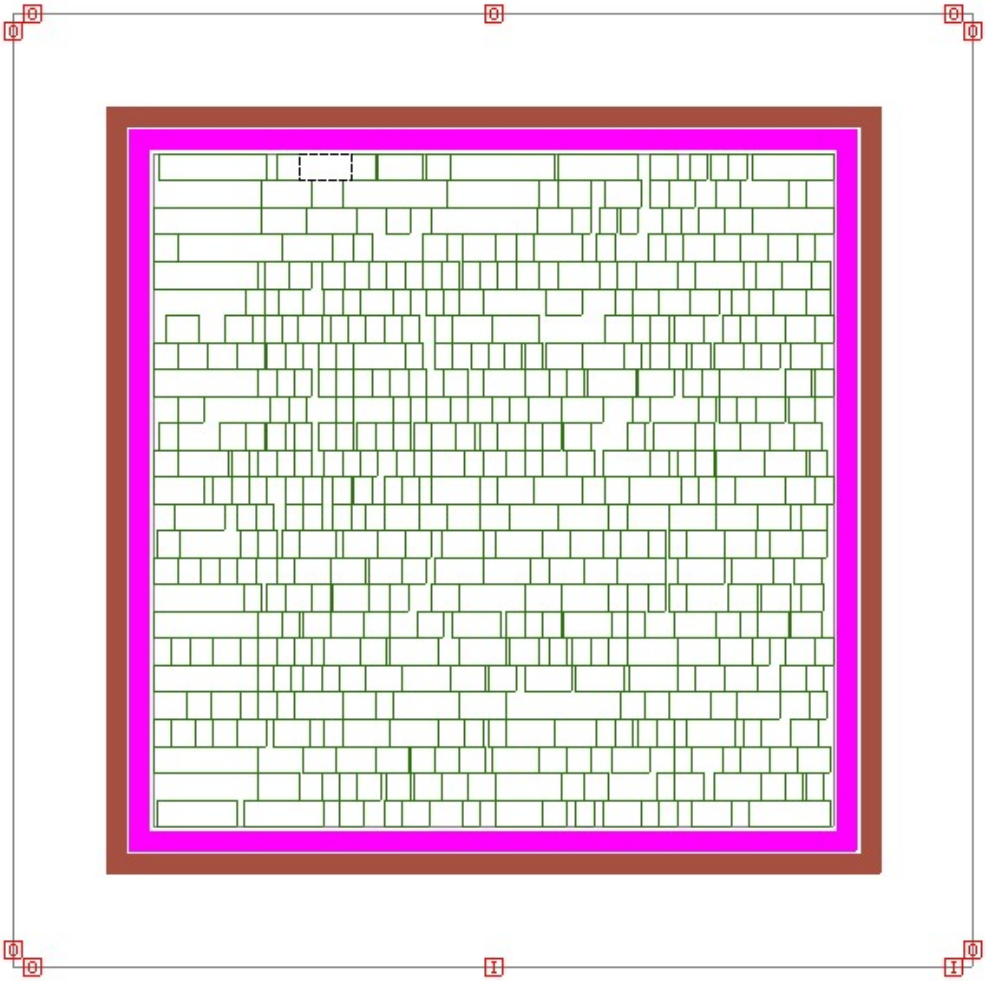
Para inicializar el diseño, se diseñó un *script* de *setup* como el descrito en la sección 6.4.3. El *script* completo se encuentra en la sección de Anexos. Los primeros resultados obtenidos fueron el *Floorplanning* y el diseño de anillos de VDD y GND, mostrados en la Figura 32. En la misma figura se observa a la derecha todas las celdas correspondientes a las diferentes compuertas lógicas contenidas en el circuito.

Figura 32: Resultado del *Floorplanning* con la librería de 90 nm



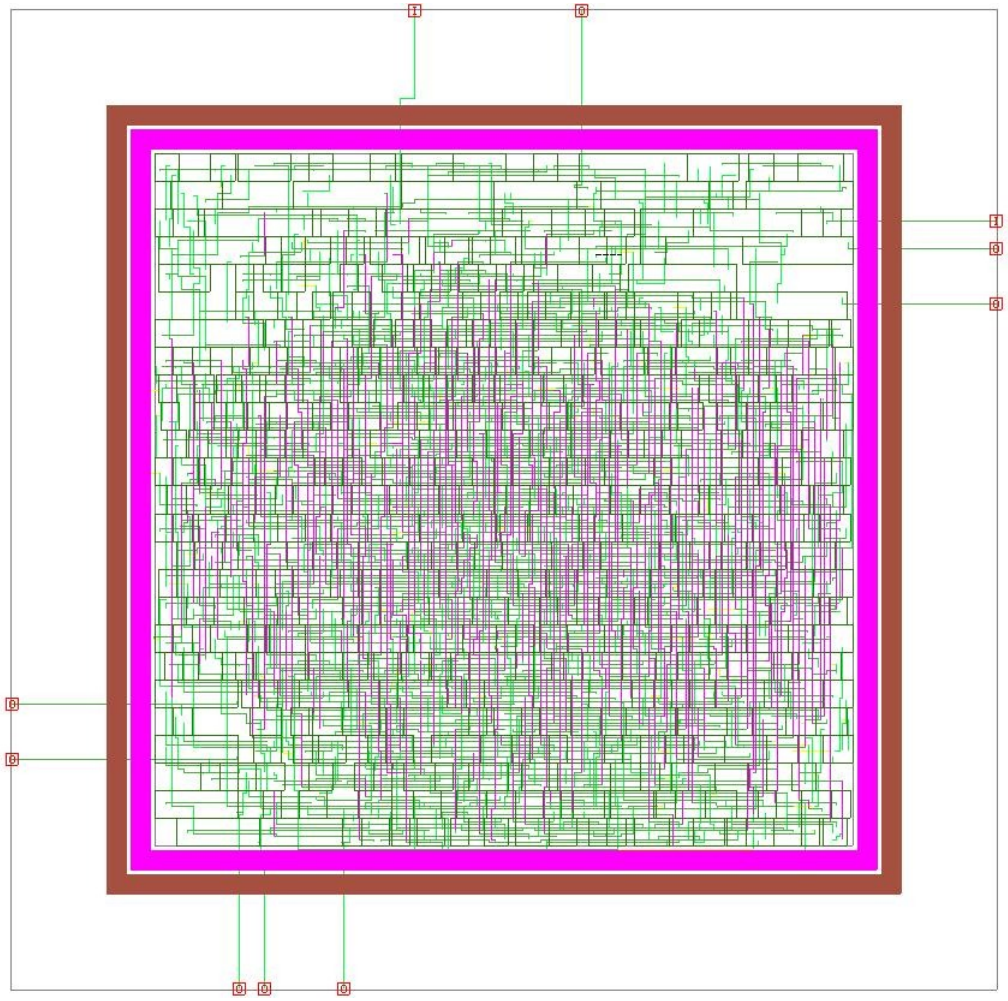
En la Figura 33 se muestra el resultado del *Placement* realizado. Se observa la forma en la que *IC Compiler* ubica los módulos y celdas del circuito dentro del *die* de forma optimizada y eficiente, para que las conexiones respecto a los pines de entrada y salida (en la periferia del *die* en la figura) sea de forma óptima.

Figura 33: Resultado del *Placement* con la librería de 90 nm



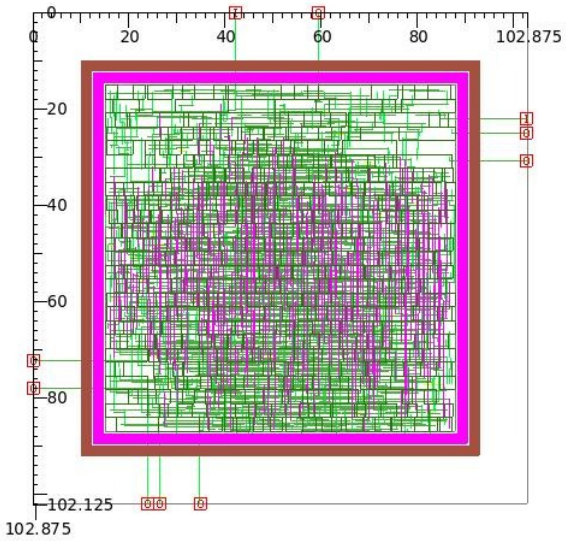
Posteriormente, las compuertas lógicas colocadas dentro del *die* en forma de celdas deben interconectarse en función de las conexiones descritas en el diseño importado de la síntesis lógica. Para ello se realizó el proceso de *Routing* con la opción de optimizar pines, descrito en la sección 6.4.2. Con esto se obtuvo el *die* mostrado en la Figura 34, en el cual se observan todas las interconexiones entre las celdas y los módulos del circuito, así como las de los pines de entrada y salida.

Figura 34: *Die* del circuito con tecnología de fabricación de 90 nm



Finalmente se obtuvo un reporte de diseño, en el que *IC Compiler* genera un archivo de texto con diferentes datos del circuito implementado en la síntesis física. Por otra parte, con la interfaz gráfica de *IC Compiler* en su ventana de *layout*, se pudo medir el área completa que utiliza el circuito integrado mediante la función de regla integrada en la herramienta. En la Figura 35 se observa este proceso. Dicho reporte se muestra en los Anexos. El Cuadro 5 muestra los datos más significados obtenidos en el reporte y en la medición de área.

Figura 35: Medición del área del circuito



Descripción	**Cantidad**
Puertos de entrada y salida	12
Celdas utilizadas en el circuito	510
Celdas combinacionales	489
Celdas secuenciales	20
Área total	0.11 mm^2

Cuadro 5: Información del diseño preliminar reportada por *IC Compiler*

CAPÍTULO 10

Pruebas finales con librerías de 180 nm de TSMC

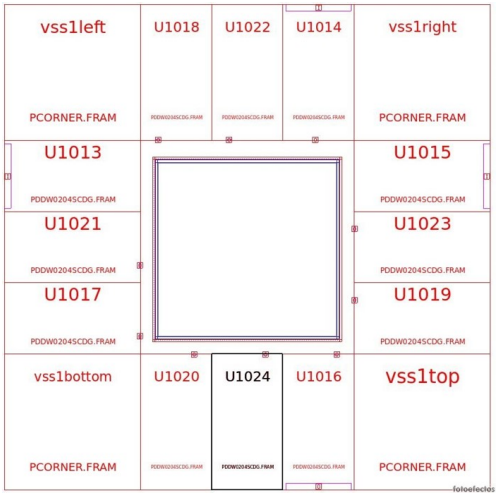
Posterior a haber realizado pruebas exitosas con las librerías de 90 nm proporcionadas por *Synopsys*, se obtuvo acceso al *PDK* de 180 nm de la empresa *TSMC*. Éste *PDK* contenía librerías y archivos más completos para un flujo de diseño completo, pues es un kit de diseño oficial de fabricación de circuitos integrados.

10.1. Síntesis completa

Se realizaron los mismos pasos de las síntesis lógica y física, descritos en la sección 9, con el agregado de poder utilizar las celdas de diseño de pines de entrada y salida, y las demás celdas necesarias para finalizar el diseño completo.

Durante la síntesis lógica, se colocaron en el archivo de *Verilog* las celdas necesarias para diseñar los pads de entrada y salida, como se muestra en la sección de Anexos. Al generar el *layout* durante la etapa de *Floorplanning*, se obtuvo lo mostrado en la Figura 36, donde se observan los pads correspondientes para las entradas y salidas.

Figura 36: *Floorplanning* del circuito en tecnología de fabricación de 180 nm con pads de entrada y salida incluidos



Al realizar los pasos descritos con anterioridad para la síntesis física, se obtuvieron los resultados mostrados en las siguientes figuras para las etapas de *Placement* y *Routing*.

Figura 37: *Placement* del circuito en tecnología de fabricación de 180 nm con pads de entrada y salida incluidos

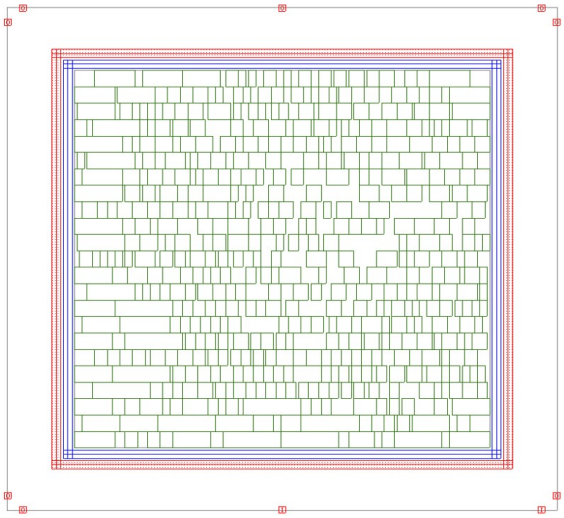
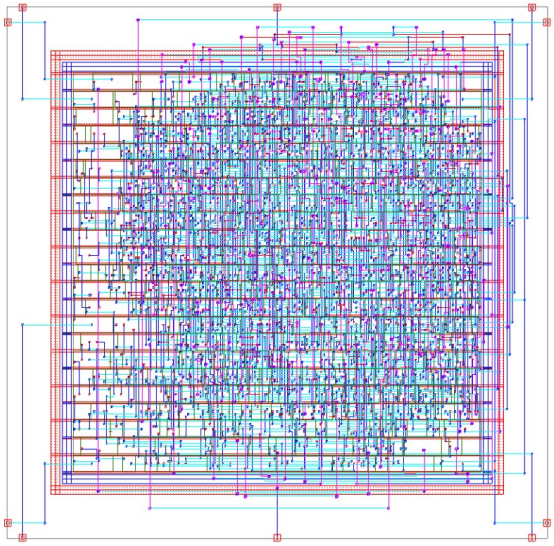


Figura 38: *Routing* del circuito con tecnología de fabricación de 180 nm



Finalmente se obtuvo el *die* completo con pads de entrada y salida, y sus *fillers* de dopante correspondientes, para obtener el circuito mostrado en la Figura 39, y poder generar los archivos GDSII que contienen la base de datos para la fabricación final del circuito integrado. Los archivos GDSII se generaron con la porción de código mostrada en la Figura 40. Al obtener el reporte de diseño con *IC Compiler*, se obtuvieron los datos mostrados en el Cuadro 6, relacionados a la cantidad de celdas y al área final del circuito.

Figura 39: *Die* del circuito finalizado con tecnología de fabricación de 180 nm

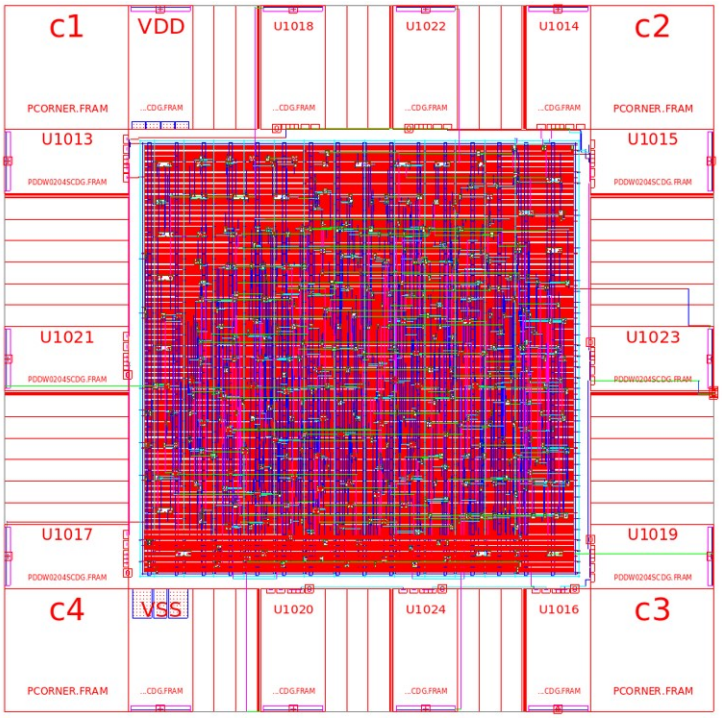


Figura 40: Generación de los archivos GDSII para la fabricación del circuito integrado

```
#Generacion de GDSII
set_write_stream_options -output_pin {text geometry} -keep_data_type
write_stream -lib_name mw_TSMC_chip_lib -format gds "chip.gds"
```

Descripción	**Cantidad**
Puertos de entrada y salida	12
Celdas utilizadas en el circuito	510
Celdas combinacionales	489
Celdas secuenciales	20
Área total	0.44 mm^2

Cuadro 6: Información del diseño final reportada por *IC Compiler*

CAPÍTULO 11

Conclusiones

- Con la síntesis física realizada al diseño lógico en Verilog proveniente de *Design Vision*, utilizando librerías con tecnología de fabricación CMOS de 180 nm, se obtuvo un *layout* de silicio correspondiente a dicho diseño mediante la utilización de la herramienta *IC Compiler*. Con *IC Compiler* se importó el diseño lógico en Verilog en extensión .ddc, y se comprobó la implementación y traslación exacta del diseño a nivel de compuertas hacia un diseño en *layout*. El *layout* obtenido contiene 12 puertos de entrada o salida y consiste en diferentes celdas de un *PDK* en tecnología de 180 nm de *Synopsys*. Con el *layout* diseñado se obtuvieron los archivos GDSII necesarios para enviar a un *foundry* y realizar la fabricación del circuito integrado. Si bien se hicieron pruebas preliminares satisfactorias con las librerías de 90 nm, al obtener las librerías de 180 nm proporcionadas por el fabricante, se replicó el proceso descrito en la sección 9.2 y se adicionó una etapa de diseño de *drivers* para los pines de entrada y salida con la información y reglas de diseño contenidas en el PDK proporcionado; logrando generar los archivos GDSII y tener la base de datos completa para su fabricación final.

- Se obtuvieron 510 celdas en un área de *die* de 0.44 μm^2 al realizar la síntesis física, con un *PDK* en tecnología de fabricación de 180 nm, del diseño proveniente de la síntesis lógica en *Verilog*. Para ello se realizaron etapas de la síntesis física como *Floorplanning* y *Placement and Route*, de las celdas de silicio generadas, dentro del *die* del circuito integrado diseñado. Se exploraron diferentes opciones de ruteo en la herramienta *IC Compiler*, y debido a que el rendimiento del circuito no apuntaba a ningún objetivo específico de potencia o de *timing*, se utilizó el ruteo global descrito en la sección 6.4.2.

CAPÍTULO 12

Recomendaciones

Los comandos y herramientas de *IC Compiler* utilizados permitieron el desarrollo de un circuito integrado cuyo funcionamiento es totalmente digital. Para proyectos digitales más complejos o proyectos que integren electrónica analógica, *IC Compiler* contiene una cantidad grande de metodologías de trabajo para todas las etapas descritas en éste trabajo, por lo que se recomienda revisar a detalle los manuales en [16], [15] y [17]; los cuales proveen la explicación profunda de todas las funciones de la herramienta para diversos circuitos que se deseen diseñar.

El diseño de circuitos integrados de aplicación específica requiere una etapa de planificación de diseño muy detallada en función de cumplir ciertos requerimientos de diseño. En éste trabajo de graduación se presenta un flujo de diseño de síntesis física para un circuito digital cuya función no apunta a cumplir ningún objetivo estricto de potencia o de *timing*. El objetivo principal del proyecto como un todo es dejar el antecedente del flujo de diseño completo en la Universidad y en el departamento de Ingeniería en Electrónica, Mecatrónica y Biomédica; para la fabricación de circuitos integrados de aplicación específica en proyectos futuros. *Synopsys* es una empresa líder en la industria en software para el diseño de circuitos electrónicos, por lo que provee más herramientas y funciones que las descritas en el presente trabajo.

Para proyectos que quieran cumplir objetivos específicos de potencia, se recomienda la herramienta *PrimePower* de *Synopsys*, la cual permite realizar reportes detallados de potencia y sugerir optimizaciones para cumplir dichos objetivos. Por otra parte, para proyectos que deban cumplir requerimientos de *timing* se recomienda el uso de la herramienta *NanoTime* de *Synopsys*, que provee análisis precisos de diversos efectos físicos que puedan causar impedimentos para cumplir dichos requerimientos, principalmente en circuitos con etapas analógicas.

CAPÍTULO 13

Bibliografía

[1] S. I. Association, *Annual Semiconductor Sales Increase 21.6 Percent, Top $400 Billion for First Time*, `https://www.semiconductors.org/annual-semiconductor-sales-increase-21.6-percent-top-400-billion-for-first-time/`, Accesado: 10-09-2019.

[2] Synopsys, *Electronic Design University Program*, `https://www.synopsys.com/community/university-program.html`, Accesado: 01-08-2019.

[3] J. A. de los Santos Chonay, *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*. Ciudad de Guatemala: Facultad de Ingeniería de la Universidad del Valle de Guatemala, 2014, 82 págs.

[4] D. A. Díaz, *Flujo de Diseño de Circuitos Integrados Digitales Aplicado al Desarrollo de un Controlador USB 2.0*. Universidad de Chile, 2010, 142 págs.

[5] EUROPRACTICE, *EUROPRACTICE Activity Report 2018*, 2018, 72 págs.

[6] E. MacDonald., *Enseñando a diseñar circuitos integrados digitales utilizando herramientas de Synopsys*. Valparaíso, Chile.: Universidad Técnica Federico Santa María, 2012, 15 págs.

[7] J. Carballo, *Chip Design for Non-designers*. PennWell Books, 2010, 162 págs.

[8] J. Rabaey, *Digital Integrated Circuits: A design perspective*. 2.ª ed. California: Universidad de California, 2000, págs. 31-78.

[9] D. Weste N. & Harris, *CMOS VLSI Design: A circuits and systems perspective*. 1.ª ed. Boston, Massachusetts.: Pearson Education, 2011, 867 págs.

[10] J. Baker, *CMOS Circuit Design, Layout and Simulation*. 1.ª ed. IEEE Series on Microelectronic Systems. John Wiley & Sons, Inc., 2010, 1214 págs.

[11] L. Xiu, *VLSI Circuit Design: Methodoly Demystified, A Conceptual Taxonomy*, 1.ª ed. IEEE Series on Microelectronic Systems. John Wiley & Sons, Inc., 2007, 222 págs.

[12] H. Kaeslin, *Digital Integrated Circuit Design. From VLSI Architectures to CMOS Fabrication*. New York: Cambridge University Press, 2008, 832 págs.

[13] V. Kishore K. & Prabhakar, *VLSI Design*, 1.ª ed. K. International Pvt Ltd., 2010, 414 págs.

[14] J. Lubaszewski & Jochen, *Design of Systems on a Chip*. Springer, 2006, 237 págs.

[15] Synopsys, *IC Compiler Design Planning User Guide*, ver. L-2016.03, Mountain View, California, 2016, 471 págs.

[16] ——, *IC Compiler Classic Router User Guide*, ver. J-2014.09, Mountain View, California, 2014, 139 págs.

[17] ——, *IC Compiler Implementation User Guide*, ver. M-2016.12, Mountain View, California, 2016, 1117 págs.

CAPÍTULO 14

Anexos

14.1. *Script de Setup*

```
##########################################################################
####                                                                  ####
####  Copyright (C) 2019 UVG.                                         ####
####                                                                  ####
##########################################################################
##########################################################################

# Common ORCA ICC setup file
# Variable setup
# source "scripts_icc/definitions.tcl"

lappend search_path /home/administrador/Escritorio/Chip_TSMC_IO/Libs
set link_library " * tcb018g3d3tc.db tcb018g3d3bc.db tcb018g3d3wc.db
    tpd018nvtc.db"
set target_library "tcb018g3d3tc.db tpd018nvtc.db"


source "/home/administrador/Escritorio/Chip_TSMC_IO/ICC/definitions.tcl"
set power                   "VDD"
set ground                  "VSS"
set powerPort               "VDD"
set groundPort              "VSS"
set core_util               "0.70"
set core_space              "3270.813965"

# Tell ICC where to look for files

set_tlu_plus_files -max_tluplus
    /usr/synopsys/TSMC/180/CMOS/G/IO3.3V/stclib/9-track/tcb018g3d3_280a_FE/
TSMCHOME/digital/Back_End/milkyway/tcb018g3d3_280a/techfiles/tluplus/
t018lo_1p6m_typical.
```

```
tluplus -min_tluplus
    /usr/synopsys/TSMC/180/CMOS/G/IO3.3V/stclib/9-track/tcb018g3d3_280a_FE/
TSMCHOME/digital/Back_End/milkyway/tcb018g3d3_280a/techfiles/
tluplus/t018lo_1p6m_typical.tluplus -tech2itf_map
    /usr/synopsys/TSMC/180/CMOS/G/IO3.3V/stclib/9-track/tcb018g3d3_280a_FE/
TSMCHOME/digital/Back_End/milkyway/tcb018g3d3_280a/techfiles/
tluplus/star.map_6M



#create_mw_lib \

create_mw_lib -technology
    /usr/synopsys/TSMC/180/CMOS/G/IO3.3V/stclib/9-track/tcb018g3d3_280a_FE/

TSMCHOME/digital/Back_End/milkyway/tcb018g3d3_280a/techfiles/tsmc018_6lm.tf

-mw_reference_library
    {/usr/synopsys/TSMC/180/CMOS/G/IO3.3V/stclib/9-track/tcb018g3d3_280a_FE/

TSMCHOME/digital/Back_End/milkyway/tcb018g3d3_280a/frame_only/tcb018g3d3
    /usr/synopsys/TSMC/180/CMOS/G/IO3.3V/iolib/LINEAR/tpd018nv_280a_FE/

TSMCHOME/digital/Back_End/milkyway/tpd018nv_280a/mt_2/6lm/frame_only/tpd018nv
    /usr/synopsys/TSMC/180/CMOS/G/stclib/7-track/tcb018gbwp7t_290a_FE/

TSMCHOME/digital/Back_End/milkyway/tcb018gbwp7t_270a/frame_only/tcb018gbwp7t}
    -bus_naming_style {[%d]}

-open /usr/synopsys/icc/bin/mw_TSMC_chip_lib

#Usar comandos
import_designs \
    -format ddc \
    -top $cell_name \
    -cel $cell_name
        {"/home/administrador/Escritorio/Chip_TSMC_IO/Out/chip_ddc_p_m.ddc"}


read_verilog {/home/administrador/Escritorio/Chip_TSMC_IO/Out/chip_syn_p_m.v}
read_sdc -version Latest
    "/home/administrador/Escritorio/Chip_TSMC_IO/Out/chip_sdc_p.sdc"


save_mw_cel -design "${cell_name}.CEL;1"

close_mw_cel

close_mw_lib
```

14.2. *Script de Design*

```
################################################################################
####                                                                        ####
#### Copyright (C) 2019 UVG.                                                ####
####                                                                        ####
################################################################################
################################################################################

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

#source "scripts_icc/definitions.tcl"
#Floorplaning

open_mw_lib /usr/synopsys/icc/bin/mw_TSMC_chip_lib

copy_mw_cel \
   -from_library /usr/synopsys/icc/bin/mw_TSMC_chip_lib \
   -from $cell_name \
   -to_library /usr/synopsys/icc/bin/mw_TSMC_chip_lib \
   -to "${cell_name}_floorplan"

set ::auto_restore_mw_cel_lib_setup false
open_mw_cel "${cell_name}_floorplan"
current_mw_cel "${cell_name}_floorplan"

create_cell {c1 c2 c3 c4} PCORNER
create_cell {VDD} PVDD1CDG
create_cell {VSS} PVSS1CDG
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


create_floorplan   \
    -start_first_row \
    -flip_first_row \
    -bottom_io2core 15 \
    -left_io2core 15 \
    -right_io2core 15 \
    -top_io2core 15

derive_pg_connection  \
   -power_net VDD \
   -power_pin VDD \
   -ground_net VSS \
   -ground_pin VSS
derive_pg_connection -power_net VDD -ground_net VSS -tie
```

```
create_rectangular_rings  \
    -nets {VDD}    \
    -left_offset 0.5 \
    -left_segment_layer METAL2 \
    -left_segment_width 2 \
    -right_offset 0.5  \
    -right_segment_layer METAL2 \
    -right_segment_width 2 \
    -bottom_offset 0.5 \
    -bottom_segment_layer METAL2 \
    -bottom_segment_width 2  \
    -top_offset 0.5 \
    -top_segment_layer METAL2 \
    -top_segment_width 2

create_rectangular_rings  \
     -nets {VSS}  \
     -left_offset 3 \
     -left_segment_layer METAL3 \
     -left_segment_width 2 \
     -right_offset 3    \
     -right_segment_layer METAL3 \
     -right_segment_width 2 \
     -bottom_offset 3   \
     -bottom_segment_layer METAL3 \
     -bottom_segment_width 2  \
     -top_offset 3      \
     -top_segment_layer METAL3  \
     -top_segment_width 2

create_power_straps \
    -direction vertical \
    -start_at 60   \
    -num_placement_strap 20 \
    -increment_x_or_y 25 \
    -nets {VDD} \
    -layer METAL2 \
    -width 3.0  \
    -do_not_route_over_macros

create_power_straps \
   -direction vertical \
   -start_at 55  \
   -num_placement_strap 16 \
   -increment_x_or_y 28 \
   -nets {VSS}  \
   -layer METAL3 \
   -width 2.0 \
   -do_not_route_over_macros
#-----------

#Tapp Cell para facilitar enrutamiento
add_tap_cell_array -master_cell_name TAPCELLBWP7T -distance 30
```

```
create_fp_placement -effort High -optimize_pins


preroute_standard_cells \
    -connect horizontal \
    -port_filter_mode off \
    -cell_master_filter_mode off \
    -cell_instance_filter_mode off \
    -voltage_area_filter_mode off

preroute_instances \
   -ignore_pads -ignore_cover_cells

verify_pg_nets

clock_opt
route_opt
#Se corre para reparar violaciones de DRC
route_search_repair -rerun_drc -loop "5"
#Well fillers
insert_well_filler -layer {PIMP} -enclosure_only 2
insert_well_filler -layer {NIMP} -enclosure_only 2
#Fillers del core
insert_stdcell_filler -cell_without_metal "FILL1BWP7T FILL2BWP7T FILL4BWP7T
    FILL8BWP7T FILL16BWP7T FILL32BWP7T" -connect_to_power "VDD"
    -connect_to_ground "VSS"
#Fillers del Pad
insert_pad_filler -cel "PFILLER1 PFILLER5 PFILLER05 PFILLER0005 PFILLER10
    PFILLER20" -overlap_cell "PFILLER0005"


#Check de Antenna Rule
uplevel #0 source
    /home/administrador/Escritorio/Chip_TSMC_Ring/ICV/antennaRule_018_6lm_Mod.tcl
#Verificacion DRC, Antenna
verify_zrt_route


#Generacion de GDSII
set_write_stream_options -output_pin {text geometry} -keep_data_type
write_stream -lib_name mw_TSMC_chip_lib -format gds "chip.gds"

save_mw_cel -design "${cell_name}_floorplan.CEL;1"
close_mw_cel

close_mw_lib
#end of floorplan
```

14.3. Reporte generado por *IC Compiler*

Figura 41: Reporte de diseño generado

```
****************************************
Report : area
Design : chip
Version: O-2018.06-SP5
Date   : Wed Sep 25 17:34:55 2019
****************************************

Library(s) Used:

    saed90nm_typ (File: /usr/synopsys/PDK/SAED90_EDK/SAED_EDK90nm_REF/references/PARSER/ref/models/saed90nm_typ.db)

Number of ports:                          10
Number of nets:                          436
Number of cells:                         415
Number of combinational cells:           397
Number of sequential cells:               17
Number of macros/black boxes:              0
Number of buf/inv:                        61
Number of references:                     32

Combinational area:               3210.610004
Buf/Inv area:                      342.860013
Noncombinational area:             505.958004
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                  3716.568008
Total area:                 undefined
```

CAPÍTULO 15

Glosario

Behavioral Descripción a nivel de comportamiento de un circuito electrónico. Generalmente la descripción a este nivel se realiza de forma parecida a un lenguaje de programación en alto nivel, con un código secuencial. 12, 15–18, 49

Celdas Módulos en el diseño de circuitos VLSI en los que se divide un circuito dentro del die. Contiene información lógica y física de ese bloque específicamente. 7, 17, 20–23, 29, 30, 33, 34, 38–41, 44, 49–52, 59

CMOS Familia lógica empleada en la fabricación de circuitos integrados actuales. Se utilizan transistores pMOS y nMOS. 11, 17, 59

Delay Cantidad de tiempo en la que tarda una señal en propagarse de un nodo a otro. 26, 28

Design Rule Check Proceso de la síntesis física en la que se comprueba si el diseño realizado cumple las reglas de diseño del fabricante. 20

Design Vision Software especializado, desarrollado por Synopsys, para realizar síntesis lógica a nivel de compuertas en el proceso de diseño de un circuito integrado. 49, 50, 59

Die Bloque de material semiconductor en el que se crea un circuito integrado. 12, 14, 19, 20, 22, 24, 26, 28, 41, 51, 52, 57, 59

Driver Etapa de potencia que permite un manejo adecuado de la corriente y voltaje entre los pines físicos del circuito integrado empaquetado, con los pines del circuito integrado dentro del die. 59

EUROPRACTICE Empresa que provee servicios de bajo costo para el diseño y fabricación de circuitos integrados. 5, 6, 43

Flujo de diseño Serie de pasos meticulosos que los ingenieros diseñadores de circuitos integrados realizan para su fabricación. 1–3, 5, 6, 15, 18, 22, 28, 30, 31, 38, 44, 46, 61

Foundry Término con el que se le conoce a una empresa dedicada a la fabricación de semiconductores y circuitos integrados. 9, 24, 29, 46, 59

GDSII Formato de archivo de base de datos utilizado para el intercambio de datos de diseños físicos de circuitos integrados para su fabricación. 3, 7, 9, 29, 46, 57–59

HDL Hardware Description Language. Lenguaje de computadora especializado en describir en estructura y comportamiento un circuito electrónico. 15–17, 20, 49

HSPICE Simulador de circuitos desarrollado por la empresa estadounidense Synopsys. 13

IC Compiler Software desarrollado por Synopsys especializado en la síntesis física durante el diseño de un circuito integrado. 28–33, 35, 36, 38, 41, 43, 44, 49, 51, 53, 57–59, 61

Intellectual Property Unidad lógica reusable en el diseño físico de un circuito electrónico, desarrollado por una empresa de diseño de circuitos integrados con el fin de vender la licencia de dicho módulo. 13

Layout Representación de un circuito integrado de forma plana, en función de los materiales semiconductores de los que está hecho. 2, 7, 11, 12, 18, 23, 32, 36–38, 40, 53, 55, 59

Netlist Descripción de la conectividad de un circuito, en forma de lista de los componentes empleados y los nodos en los que se interconectan. 16, 17, 20, 23, 29

PDK Process Design Kit. Es un grupo de archivos utilizado para modelar un proceso de fabricación para las herramientas de diseño de un circuito integrado. El PDK es creado por el fabricante, con cierta tecnología de diseño, y proveído al cliente para sus diseños. 3, 43, 55, 59

Python Lenguaje de programación de alto nivel. 46

Ruteo Proceso en el que se determinan los caminos dentro de un circuito integrado para las conexiones. 23, 31, 32, 34, 41, 59

Script Programa almacenado en un archivo de texto que cumple una función específica. 33, 36, 44, 49, 50

Structural Descripción a nivel de interconexiones de un circuito electrónico. Generalmente la descripción a este nivel se realiza con instancias de módulos y con conexiones de señales . 12, 15, 18

Synopsys Empresa estadounidense dedicada al desarrollo de software especializado para diseño de circuitos electrónicos integrados. 2, 3, 5, 18, 28, 33, 43, 44, 46, 49, 50, 55, 59, 61

Tecnología de fabricación Longitud del canal en un transistor fabricado mediante procesos de VLSI. 2, 12, 17, 31, 33, 36, 43, 46, 52, 56, 57, 59

Timing Tiempo en el que se modifica una señal. 18, 23, 28, 30–32, 44, 59, 61

TSMC Empresa que provee servicios relacionados a la fabricación de circuitos integrados y semiconductores. 43, 55

Verilog Lenguaje de descripción de hardware para modelar circuitos electrónicos. 16, 33, 49, 55, 59

VLSI Integración a gran escala. Proceso de crear un circuito integrado compuesto por cientos de miles de transistores en un único chip. 1–3, 5, 11, 12, 18, 20, 43