Ejecución y utilización de un flujo de diseño para el desarrollo de un chip con tecnología nanométrica: Extracción de componentes parásitos y simulaciones en HSPICE

Charlie Ayenci Cruz Girón

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Ejecución y utilización de un flujo de diseño para el desarrollo de un chip con tecnología nanométrica: Extracción de componentes parásitos y simulaciones en HSPICE

Trabajo de graduación presentado por Charlie Ayenci Cruz Girón para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2021

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Ejecución y utilización de un flujo de diseño para el desarrollo de un chip con tecnología nanométrica: Extracción de componentes parásitos y simulaciones en HSPICE

Trabajo de graduación presentado por Charlie Ayenci Cruz Girón para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2021

Vo.Bo.:

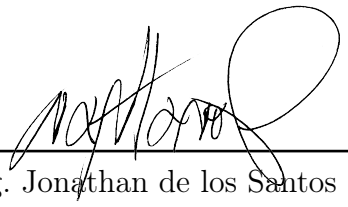
(f) _____

MSc. Carlos Esquit

Tribunal Examinador:

(f) _____

MSc. Carlos Esquit



(f) _____

Ing. Jonathan de los Santos

(f) _____

Ing. Kurt Kellner

Fecha de aprobación: Guatemala, 19 de enero de 2021.

Prefacio

Le agradezco a mi familia por haberme acompañado y guiado a lo largo de mi carrera, por ser mi fortaleza en los momentos de debilidad y por brindarme una vida llena de aprendizaje, experiencias y sobre todo felicidad.

Gracias a MSc Carlos Esquit por darme la oportunidad de ser parte de este proyecto, y sobre todo por habernos brindado su conocimiento y asesoría técnica. Al ingeniero Jonathan de los Santos por ayudarnos con el uso de las herramientas de *Synopsys*. Al Ing Kurt Kellner por su constante apoyo y toda el seguimiento que le dio al trabajo. A todos mis amigos, por ser excelentes compañeros de tesis, por haberme tenido la paciencia necesaria y por motivarme a seguir adelante en los momentos de desesperación.

Índice

Lista de figuras

Lista de cuadros

Resumen

El proyecto general consiste en la ejecución de un flujo de diseño para la elaboración de un chip con tecnología de 180nm. El flujo de diseño que se propuso consta de 10 pasos clave para la ejecución:

1. Síntesis HDL
2. Floorplan, placement and route
3. I/O ping
4. Simulaciones HDL/esquemático
5. DRC
6. LVS
7. Extracción de parásitos
8. Creación de archivos GDS
9. Verificación de manufactura
10. Pruebas de funcionalidad

Este proyecto se enfoca en la parte de extracción de parásitos y simulación en HSPICE. Tiene como objetivo principal la realización de una extracción exitosa para la simulación del chip y así comprobar si el diseño cumple con las funciones planteadas. Para lograr este objetivo se utilizaron la herramientas de Synopsys StarRC y HSPICE. Se realizaron distintas pruebas con distintos circuitos. Esto con el fin de poder validar cada paso del flujo de diseño propuesto. Cada comando, librería, software se documento a detalle para que en el futuro sirva como guía para que los estudiantes interesados en trabajar en esta área de investigación, puedan desarrollar su propio chip.

Abstract

The general project consists of the execution of a design flow for the elaboration of a chip with 180nm technology. The proposed design flow consists of 10 key execution steps:

1. HDL Synthesis
2. Floorplan, placement and route
3. I / O ping
4. HDL / schematic simulations
5. DRC
6. LVS
7. Parasite extraction
8. Create GDS files
9. Manufacturing verification
10. Functionality tests

This project focuses on the parasite extraction and simulation part of HSPICE. Its main objective is to carry out a successful extraction for the simulation of the chip and thus check if the design meets the proposed functions. To achieve this objective, the Synopsys StarRC and HSPICE tools were used. Different tests were carried out with different circuits. This in order to be able to validate each step of the proposed design flow. Each command, library, software is documented in detail so that in the future it can serve as a guide for students interested in working in this area of research to develop their own chip.

CAPÍTULO 1

Introducción

Con la tecnología escalando rápidamente, los efectos parásitos de las partes que componen un diseño en silicio se ha convertido en una influencia dominante en el desempeño de los circuitos *VLSI*. La tarea principal de una extracción de parásitos es la de calcular los efectos electromagnéticos en los cables y difusiones, tales como capacitancias, resistencias e inductancias. Este trabajo tiene como objetivo integrar al flujo existente la herramienta *StarRC*, para la realización de un *Layout Parasitic Extraction LPE* en un diseño en silicio. Así como la realización de una extracción exitosa. Y así poder crear un modelo analógico del circuito, para luego ser utilizado en simulaciones y verificaciones de funcionamiento.

Este trabajo esta dividido en cuatro capítulos con un orden bien definidos para que pueda ser utilizado como guía para futuras extracciones parásitas. El primer capítulo explica los conceptos básicos y archivos necesarios que sirven como entrada para la herramienta StarRC. Así como el flujo de diseño que se utilizara junto con la herramienta de *LVS, IC Vaildator*. El segundo capítulo se desarrolla enseñando como preparar todos los archivos de entrada (*nxtgrd, mapping, report file*) y su obtención. En el capítulo tres se muestra el resultado de una extracción a un diseño de una compuerta *not* y un diseño de un sumador completo, junto con todos los archivos obtenidos. Y el último capítulo expone las razones por lo cual no se pueden realizar simulaciones a nivel HSPICE del circuito diseñado utilizando las librerías de *TSMC*.

CAPÍTULO 2

Antecedentes

En el año 2009 el Ing. Carlos Esquit empieza a ocupar el puesto como nuevo director del departamento de Ingeniería Electrónica. Él posee estudios de Maestría en Ingeniería Electrónica a nanoescala en la Universidad de Texas A&M en Estados Unidos. Una de las primeras funciones que realizo como director, fue la de adaptación curricular para poder introducir cursos de diseño de circuitos a micro y nanoescala. [1]

Cuatro años más tarde gracias a los esfuerzos del Ing. Carlos Esquit, se imparte por primera vez un curso de diseño de circuitos a micro y nanoescala, que llevaba por nombre Introducción al Diseño de Sistemas VLSI. Al siguiente año, se logra un convenio con la compañía Synopsys para tener acceso al University Program que permite el acceso a herramientas de software y a librerías para la realización de diseños complejos.[1]

El siguiente gran paso fue el de involucrar VLSI en los trabajos de graduación. El primer trabajo consistió en el diseño de un sumador de 32 bits con tecnología de 28 nanómetros[2]. En el año 2015 el curso de Introducción al Diseño de Sistemas VLSI paso a llamarse nanoelectrónica y se dividió en dos partes. Luego en el año 2019 un grupo de estudiantes, en su trabajo de graduación, elaboraron un flujo de diseño para fabricar un chip a escala nanométrica con la compañía tailandesa Taiwan Semiconductor Manufacturing Company (TSMC).

CAPÍTULO 3

Justificación

El objetivo principal del diseño VLSI es producir los detalles de un dispositivo VLSI para que pueda realizarse físicamente. Este proceso debe cumplir con ciertos requerimientos de diseño, los cuales son proporcionados por la empresa que manufactura el chip. Este trabajo de investigación pretende dejar las bases para la elaboración de chips que tengan funcionalidades más especificas y complejas, lo que a su vez implica la posibilidad de realizar proyectos de investigación en nanoelectrónica dentro del país. Pues Guatemala no cuenta con el suficiente desarrollo en esta rama de la electrónica.

La extracción de componentes parásitos se utiliza para crear un modelo analógico preciso del circuito, de modo que se puedan realizar simulaciones y emular respuestas reales de circuitos digitales y analógicos. Las respuestas de circuitos digitales se utilizan para hacer cálculos de retardos de tiempo, análisis de potencia, análisis de integridad de señal etc. Las simulaciones realizadas al chip tienen como fin corroborar el correcto funcionamiento del chip antes de ser enviado a fabricar. [3]

CAPÍTULO 4

Objetivos

4.1 Objetivo general

Extraer componentes parásitos presentes en el diseño de un circuito integrado con tecnología nanométrica y realizar simulaciones que evidencien su funcionamiento.

4.2 Objetivos específicos

- Integrar al flujo de diseño existente la herramienta de Synopsis, StarRC, para la realización del LPE (Layout Parasitic Extraction).
- Realizar una extracción exitosa de componentes parásitos en el esquemático del circuito, para obtener una representación de un modelo analógico del circuito integrado con tecnología nanométrica.
- Probar la funcionalidad del circuito diseñado y realizar análisis del mismo mediante simulaciones detalladas utilizando la herramienta de simulación de Synopsis HSPICE.

CAPÍTULO 5

Alcance

En este trabajo de investigación se pretende integrar, al flujo de diseño existente, el flujo para la elaboración de una extracción de parásitos o *LPE*, como último paso de las distintas verificaciones existentes (*DRC, LVS*). Para lograr este objetivo se documentó cada librería, comando, programa y archivos necesarios, para que en el futuro sirva de guía para interesados en trabajar en esta fase del proyecto.

Para la realización de la extracción se utilizaron dos diseños en silicio, un diseño de una compuerta *not* y un diseño de un sumador completo. Todos los archivos de entrada y salida serán explicados, tanto su obtención como el uso correcto de los mismos. También se documentó e interpretaron los archivos importantes de salida para uso en el proceso de resolución de problemas. Las librerías proveídas por *TSMC*, por ser un *kit* académico, no cuentan con una descripción completa de las celdas. Por lo que la realización del *LVS*, y por lo tanto *LPE*, tendrá que adaptarse a un modelo *Black Box* que solo posee una descripción parcial de las celdas.

CAPÍTULO 6

Marco teórico

6.1 Transistores MOSFET

Los transistores son dispositivos semiconductores de tres terminales que son utilizados en electrónica digital como conmutadores controlados mediante señales eléctricas. El principio básico de funcionamiento del transistor involucra el uso de un voltaje entre dos terminales para controlar la corriente que fluye en la tercera terminal.[4]

El material base con que son fabricados los circuitos integrados es el silicio. El silicio puro consiste en un arreglo tridimensional de átomos. Este elemento pertenece al Grupo IV, por lo que forma enlaces covalentes con 4 electrones adyacentes como se ve en la Figura 1. Como todos los electrones de valencia están involucrados en los enlaces químicos, el silicio puro es un mal conductor.

La conductividad puede mejorarse introduciendo pequeñas cantidades de impurezas llamadas dopantes en el cristal de silicio. Un ejemplo de estos dopantes es el arsénico, este átomo tiene cinco electrones de valencia. Al introducir el arsénico como átomo dopante, se crean enlaces covalentes con cuatro electrones del arsénico dejando un electrón libre como se ve en la Figura 1. Este electrón libre permite el aumento de la corriente dentro del material provocando un aumento de la conductividad. Otro material utilizado como dopante es el boro, este a diferencia del arsénico solo posee 3 electrones de valencia. Dopar al silicio con boro y debido a la ausencia de un electrón para completar el enlace, provoca un tipo de reacción en cadena donde cada átomo de silicio le presta un electrón al átomo de boro y este a su vez le presta uno al átomo adyacente. El electrón faltante se propaga dentro del cristal actuando como una carga positiva. Los materiales que al ser dopados tiene mayor numero de portadores con carga negativa son llamados tipo n y los que tienen mayor numero de portadores con carga positiva tipo p. [4]

Un Semiconductor Metal-Oxido, MOS por sus siglas en ingles, se crea superponiendo varias capas de conductores y materiales aislantes, mediante procesos químicos. La tecnología CMOS utiliza dos tipos de transistores: un transistor tipo n (nMOS) y un transistor tipo

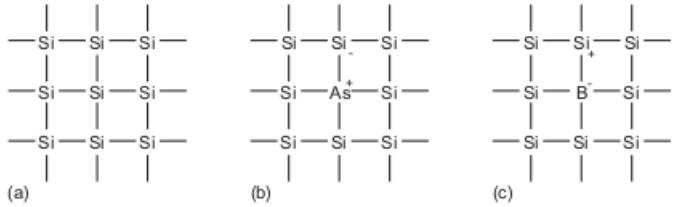


Figura 1: (a) Cristal de silicio puro. (b)Silicio dopado con arsénico. (c) Silicio dopado con boro.[4]

p (pMOS) ver Figura 2. Estos dispositivos son controlados mediante un campo eléctrico y por ellos son llamados transistores de efecto de campo (FETs) o también (MOSFETs). Cada transistor tiene una capa de material conductor llamada compuerta (gate). un material aislante de dióxido de silicio, un cuerpo o sustrato también llamado wafer. Un transistor nMOS es construida en un wafer de material tipo p y tiene regiones de material semiconductor tipo n adyacentes al gate llamadas source and drain.[4]

La compuerta(gate) funciona como entrada de control; afecta el flujo de corriente eléctrica entre el source y el drain. Por ejemplo un transistor nMOS posee un cuerpo tipo p que es conectado a tierra, así la unión p-n de el source y el drain están en un estado de no conducción. Entonces, el transistor se encuentra apagado. Si el voltaje en el gate se incrementa, se crea un campo eléctrico que atrae electrones libres al lado del dióxido de silicio y los materiales tipo n. Si el voltaje sigue creciendo el numero de electrones supera al numero de hoyos creando una región debajo del gate llamado canal y esta se comporta como material tipo n. Es de esta forma que puede fluir corriente entre el source y el drain.[4]

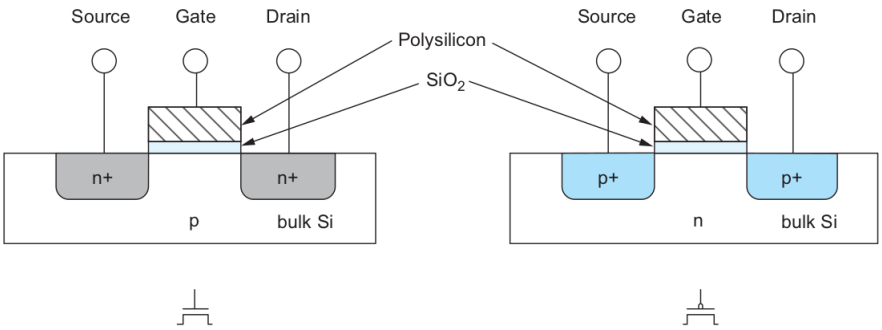


Figura 2: (a) Nmos. (b) Pmos.[4]

6.2 Very Large Scale Integration (VLSI)

Probablemente el evento mas importante que ha ocurrido en la electrónica ocurrió en 1947, cuando John Bardeen, Walter Brattain, and William Shockley en los laboratorios Bell inventaron el transistor bipolar. Luego en 1958, ocurrió una nueva invencion, casi de forma simultanea, del circuito integrado (IC) por Kilby en Texas Instruments y Noyce y Moore en Fairchild Semiconductor. La miniaturización de dispositivos que se puedo lograr a través de la tecnología IC provoco que los dispositivos electrónicos pudieran realizar operaciones cada

vez mas complejas con alto rendimiento y bajo costo. Esta invención tuvo mayor impacto en el área de la electrónica digital.[5]

El progreso de la miniaturización de circuitos integrados se muestra en la Figura 3. Como se puede ver en cuatro décadas desde 1970, el numero de transistores en un microprocesador se ha incrementado por un factor de un millón. Desde la introducción del circuito integrado el incremento de la densidad de transistores se ha logrado gracias a la continua reducción del ancho existente entre el drain y el source, también conocido en ingles como minimum feature size. Hoy en día esta característica ha alcanzado valores en el orden de los nanómetros. [5]

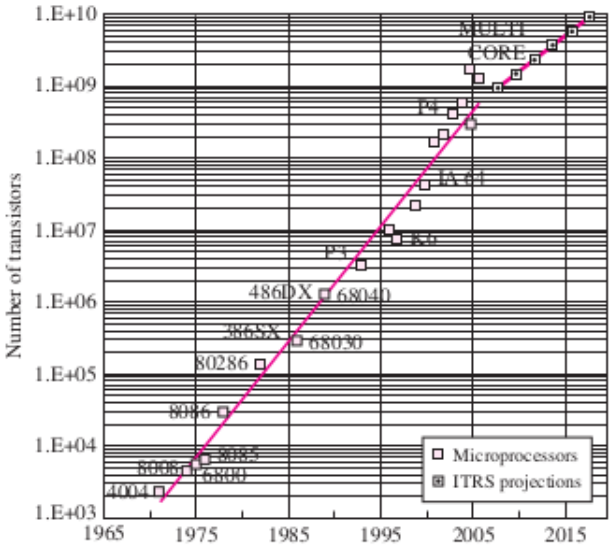


Figura 3: Complejidad de los microprocesadores contra tiempo [5]

A medida que el proceso de miniaturización continuaba, se fue usando una serie de abreviaciones para caracterizar los niveles de integración alcanzados. Los primeros IC, con menos de 100 componentes fueron caracterizados como small-scale integration o SSI. Luego conforme la densidad de componentes disminuia los IC fueron llamados medium-scale integration (MSI, 100-1000 componentes), luego large-scale integration (LSI, 10^3- 10^4 componentes), y very-large-scale integration (VLSI, 10^4 - 10^9 componentes).

6.3 CMOS VLSI Design Flow

El flujo de diseño se refiere al conjunto de pasos que deben seguir los diseñadores de un chip para llevar a cabo todas las especificaciones, hasta llevar el chip al proceso de fabricación. El flujo de diseño se divide en dos partes: Front-End y Back-End.

6.4 Front End

El flujo front end es responsable de determinar una solución a un problema y transformarlo en una descripción de circuito RTL. Describe la lógicas RTL (Register Transfer Level) en HDL. Para esto, se utiliza un lenguaje descriptor de hardware como Verilog y VHDL. Esta etapa de diseño contiene todos los detalles necesarios para la arquitectura de diseño, diagrama de bloques RTL, frecuencia de reloj, formas de onda, detalles de puertos, etc. A continuacion se enumeran los pasos importantes para la elaboración del diseño front end.[1]

1. Verilog/RTL: Elaboración del circuito en verilog de forma estructural o behavioral. A este circuito se le conoce como RTL.
2. Síntesis lógica: Se utiliza la herramienta Design Vision para convertir el RTL en un Netlist de compuertas lógicas.
3. Netlist: Circuito creado luego de aplicar la síntesis lógica. Se presenta de manera estructural y con componentes que hacer referencia a librerías.
4. Verificación: Se verifica la funcionalidad de la síntesis del circuito en plataformas como Formality, VCS o bien utilizando un FPGA.[1]

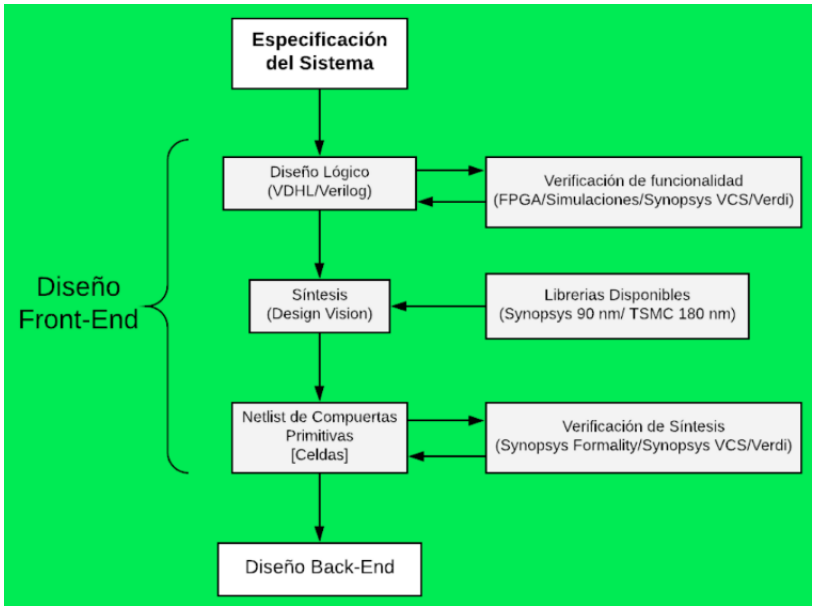


Figura 4: Diseño Front-End [1]

6.5 Back End

El proceso de back-end es responsable de la implementación física de un circuito. En este paso se transforma la descripción del circuito RTL en un diseño físico, compuesto por compuertas y sus interconexiones. En esta etapa del diseño se realiza el floorplanning, el power an ground routing, placement y routing.[6]

1. Floorplanning: Determina la forma y de subcircuitos y módulos.
2. Power and Ground Routing: Distribuye las conexiones de voltaje y tierra en todo el circuito.
3. Placement: Coloca cada celda de cada bloque en cierta posición espacial.
4. Routing: Asigna rutas a capas de metal específicas. [6]

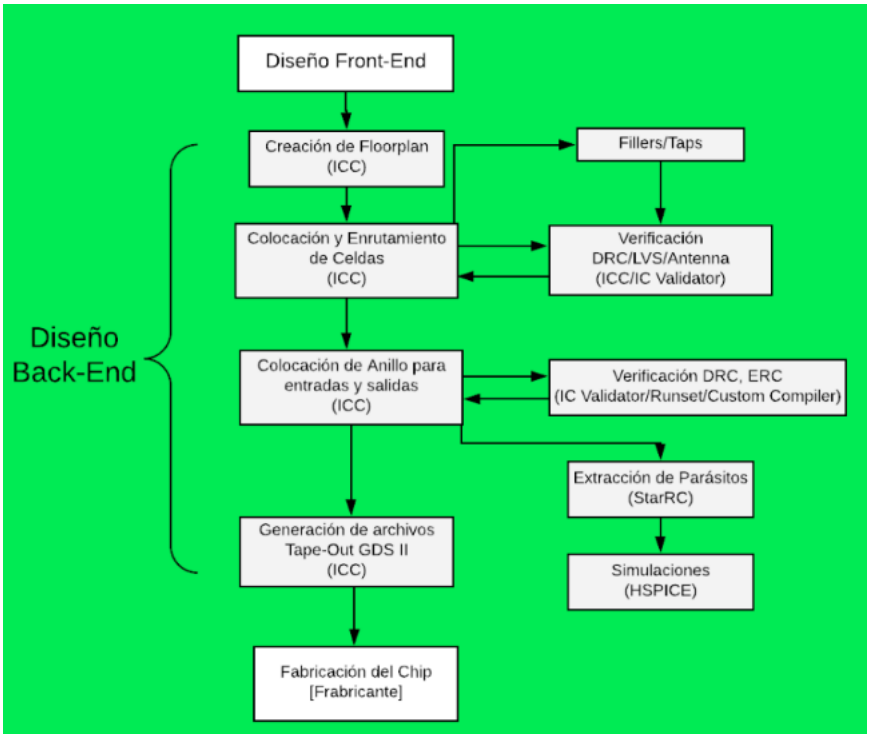


Figura 5: Diseño Back-End [1]

6.7 Verificación física

Luego de terminar el diseño físico, este debe ser completamente verificado para cumplir con la funcionalidad eléctricas y lógica. [6]

- Design Rule Check (DRC): Verifica que el diseño cumpla con todas las restricciones tecnológicas.
- Layout vs Schematic: Verifica la funcionalidad del diseño, se genera una netlist y se compara con la netlist generada durante la síntesis lógica.
- Parasitic Extraction: Se utiliza para verificar caracteristicas electricas del circuito.
- Antenna rule check: Previene efectos de antena, los cuales pueden dañar a los transistores.

- Electrical rule checking (ERC): Verifica que las conexiones de voltaje y tierra sean correctas.

6.8 Extracción de parásitos

En circuitos VLSI (very large-scale integration), el acoplamiento electromagnético entre los cables interconectados y las diferentes difusiones se vuelve cada vez mas importante. Con la llegada de las tecnologías de semiconductores nanométricas las interconexiones entre cables no pueden ser consideradas como un simple cable sin resistencia conectado a un equipotencial. Los efectos parásitos introducidos por los cables y difusiones muestran un comportamiento que tiende a ganar mas importancia conforme los dispositivos son reducidos y las velocidades de los circuitos incrementa.[7]

Un flujo de diseño típico para la creación de un circuito integrado puede verse en la Figura 6. Donde luego de un diseño del layout se procede con la extracción de parásitos para luego hacer una simulación a nivel compuerta. Cada paso en el flujo de diseño corresponde a la utilización de herramientas asistidas por computadora (CAD), lo cual garantiza un diseño VLSI moderno factible.

La tarea de una extracción de parásitos es la de modelar los efectos electromagnéticos en los cables y difusiones con componentes parásitos tales como capacitancia (C), resistencia (R) e inductancia (L). Así lograr realizar simulaciones a nivel compuerta, incluyendo un análisis de tiempo, potencia etc.

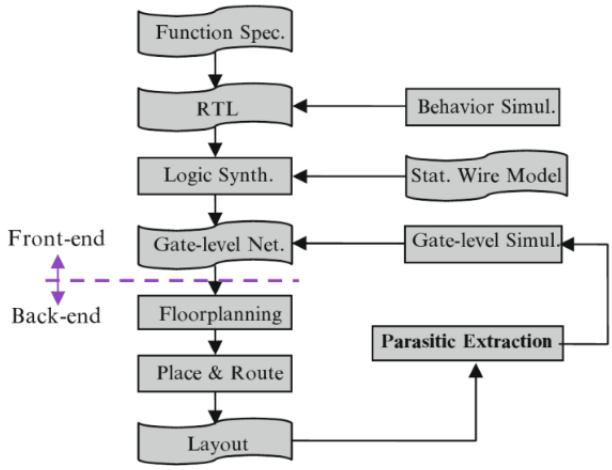


Figura 6: Flujo de diseño con extracción de parásitos.[7]

6.9 Synopsys

La empresa Synopsys se ha posicionado como líder mundial en proveer soluciones para el diseño y verificación avanzada de circuitos de silicio. Empresas como Intel, AMD, o Xilinx utilizan software de Synopsys para el desarrollo de nuevas tecnologías. Synopsys cuenta con

un amplio repertorio de herramientas que permiten maximizar productividad mientras se optimiza la potencia, rendimiento y costo de un chip. Cada una de estas herramientas esta pensada para desarrollar un segmento especifico del circuito integrado que se desea diseñar. [8] [9]

6.10 StarRC

StarRC es una herramienta de Synopsys que se utiliza para la extracción de componentes parásitos como resistencias, capacitores e inductores de layouts que representan diseños de circuitos integrados. La herramienta StarRC genera Netlists que son útiles para muchos tipos de análisis, como análisis temporal, ruido y funcionalidad. [3]

6.11 HSPICE

Es un simulador de circuitos analógicos y optimizador que pertenece a Synopsys. Puede ser utilizado para análisis de circuitos eléctricos en estado estacionario, transitorio y dominio de frecuencia. HSPICE permite hacer simulaciones rápidas y precisa de circuitos. Facilita el análisis a nivel de circuito mediante el uso de Monte Carlo, worst-case, barrido paramétrico y los análisis de barrido de tabla de datos, y emplea una capacidad de convergencia automática confiable ver Figura 7. [10]

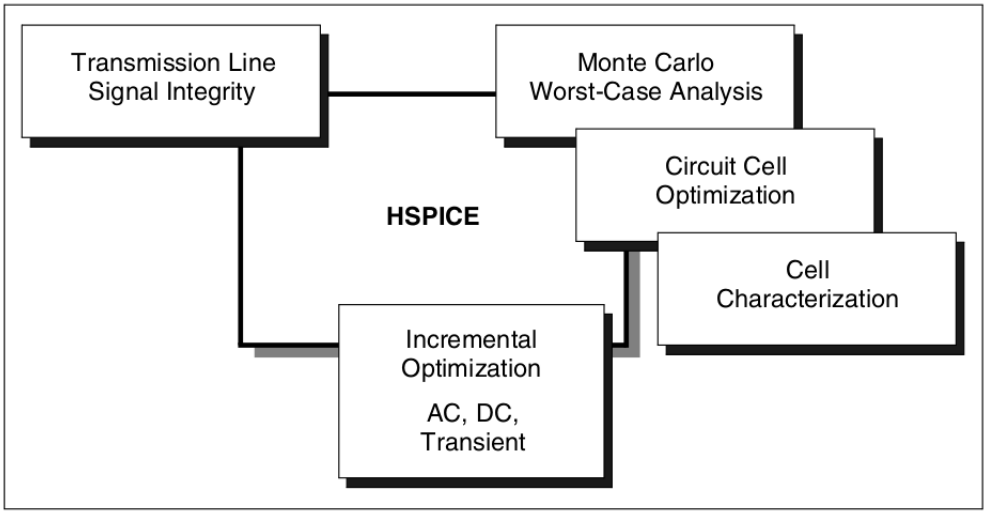


Figura 7: Características de diseño HSPICE [10]

CAPÍTULO 7

Flujos de diseño para extracción de parásitos

7.1. Uso de la herramienta StarRC

Esta herramienta analiza capacitancias y resistencias parásitas de diseños avanzados de circuitos. Esto lo logra comparando el diseño con un conjunto de estructuras primitivas cuyo elementos parásitos han sido calculados previamente. Esta herramienta puede ser utilizada de diferentes maneras dependiendo lo que se desea lograr. Los flujos principales son los siguientes: Extracción a nivel transistor y extracción a nivel compuerta.

7.2. Extracción a nivel transistor y extracción a nivel compuerta

El flujo a nivel transistor se enfoca en los componentes mas pequeños del diseño en gran detalle. Con esto se puede examinar los parásitos a nivel de componentes. Para realizar este flujo se necesita utilizar una herramienta de **Layout vs Schematic (LVS):**. También llamado extracción a nivel celda (o macros). Esta extracción es utilizada comúnmente para análisis temporales de todo el chip.

7.3. Archivo ITF y herramienta grdgenxo

Un archivo nxtgrd contiene referencias de componentes parásitos para el proceso de manufactura del chip. Este archivo se especifica en el archivo de comandos de StarRC utilizando el comando `TCAD_GRD FILE`. Un archivo nxtgrd es creado con la herramienta grdgenxo, un programa de utilidad de StarRC, como se muestra en la Figura 8. La herramienta grdgenxo

genera el archivo nxtgrd desde un archivo escrito en un lenguaje de descripción de procesos llamado Interconnect Technology Format (ITF). El archivo nxtgrd contiene capacitancias, resistencias, e información de las capas del chip, junto con las declaraciones ITF.

Se puede utilizar el archivo ITF de dos formas: Corriendo la herramienta grdgenxo para procesar el archivo ITF y crear un archivo nxtgrd para usar en la herramienta StarRC (Especificándolo con el comando `TCAD_GRD`. La otra forma de usar este archivo es leyéndolo directamente usando el comando `ITF_FILE` desde StarRC.

Figura 8: Generación del archivo de proceso [3]

7.4. Modo de operación Batch

Puede ejecutarse *StarRC* en modo batch mediante un archivo o archivos de comandos en la línea de consola. Esto se logra mediante el comando `StarXtract` que invoca a la herramienta StarRC. El comando StarRC utiliza la siguiente sintaxis:

```
StarXtract opcion1 opcion2 ... cmd_file1 [cmd_file2]
```

Opción	**Descripción**
cmd_file1	Archivo de comandos de *StarRC*
cmd_file2, ...	Archivo de comandos opcional.
-compare_parasitics	Comparo dos *netlists*
-convert_gpd_to_spef	Crea un *netlists* SPEF con un archivo GPD
-convert_gpd_to_spf	Crea un *netlists* SPF con un archivo GPD

Cuadro 1: Opciones del comando `StarXtract`

7.5. Bases de datos físicas y formatos soportados

La herramienta *StarRC* importa una descripción física del diseño para calcular los efectos parásitos de una de las siguientes fuentes: una base de datos ***Milkyway:***, *Hercules*, *IC Validator*, *Calibre* o una base de datos *LEF/DEF*.

Se pueden crear varios tipos de *Netlists* parásitos para usar como entrada a otra herramienta. Cada uno de ellos contiene ciertas diferencias en formato y estructura. Los siguientes tipos de*Netlists* disponibles son los siguientes: *Standard Parasitic Exchange Format(SPEF)* utilizado para análisis de tiempos, *Standard Parasitic Format(SPEF)* este formato contiene información a nivel dispositivo para entrada a herramientas de simulación. Formato *Netname* es un formato de salida tipo *SPICE* y contiene el nombre de los nodos internos, lo cual

provee mayor información acerca de los elementos parásitos a los que se esta conectado. *OpenAcces format (OA)* es un estándar libre para diseño de circuitos. Y por último el formato *STAR*, también es un formato tipo SPICE que se usa como entrada para herramientas de simulación.

7.6. Ejemplo de un *Netlist SPEF*

```
*SPEF "IEEE 1418-1999"
*DESIGN "ALU"
*DATE "Wed April 17 19:50:14 2006"
*VENDOR "Synopsys"
*PROGRAM "StarRC"
*VERSION "K-2015.06"
*DESIGN_FLOW "PIN_CAP NONE" "NAME_SCOPE LOCAL"
*DIVIDER /
*DELIMETER :
*BUS_DELIMITER []
*T_UNIT 1 NS
*C_UNIT 1 FF
*R_UNIT 1 HENRY

*NAME_MAP
*5 net1
*10 insta/net2
*14 U1
*16 insta/U1
*17 insta/U2

*PORTS
*5 I *C 3.6 0

*D_NET *5 1.02e+00

*CONN
*P *5 I *C 3.6 0
*I *14:I I *C 6.76 8.94 *L 4 *D inv

*CAP
1 *5 0.60481
2 *14:I 0.0413795

*RES
1 *5 *14:I 29.8492

*END
```

```
*D_NET *10 4.58e-01

*CONN
*I *16:ZN O *C 9.12 21.84
*I *17:I I *C 6.34 20.94 *L 4 *D inv

*CAP
1 *16:ZN 0.271701

*RES
1 *16:ZN *17:I 17.8989

*END
```

7.7. Ejemplo de un *Netlist SPF*

```
*
*|DSPF 1.0
*|DESIGN top
*|DATE "Wed␣April␣17␣13:34:43␣2000"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "2003.2.0.0"
*|DIVIDER /
*|DELIMETER :
*

.SUBCKT top net1

*|GROUND_NET 0

*|NET net1 9.60e-04PF
*|P (net1 I 0 3.6 0)
*|I (U1:I U1 I I 4e-15 6.76 8.94)
R1 net1 U1:I 29.8492
Cg1 net1 0 5.8737e-16
Cg2 U1:I 0 3.72441e-16

*|NET insta/net2 5.10e-04PF
*|I (insta/U1:ZN insta/U1 ZN O 0 9.12 21.84)
*|I (insta/U2:I insta/U2 I I 4e-15 6.34 20.94)
R2 insta/U1:ZN 0 2.96927e-16
Cg3 insta/U1:ZN 0 2.96927e-16
Cg4 insta/U2:I 0 2.12328e-16
*
* Instance Section
```

```
*
Xinsta/U2 insta/U2:I in2 VDD VSS inv
Xinsta/U1 in1 insta/U1:ZN VDD VSS inv
XU1 U1:I net2 VDD VSS inv

.ENDS
```

7.8. Ejemplo de un *Netlist NETNAME*

```
* SPICE Netlist
* VENDOR "Synopsys, Inc."
* PROGRAM "StarRC"
* Date "Thu April 16 16:26:00 2002"

**FORMAT SPICE

.SUBCKT AND2 B A OUT
XS1I1 B A S1N3 NAND2
XS1I2 OUT S1N3 INVA
.ENDS AND2

.SUBCKT AND3 C B A OUT
XS1I1 C B A SIN9 NAN3
XS1I2 OUT SIN9 INVA
.ENDS AND3

SUBCKT CS_ADD1 SUM COUT C B A
XS1I2 B A S1N29 AND2
XS1I3 C S1N9 S1N31 AND2
XS1I4 S1N43 S1N35 S1N39 AND2
XS1I5 SIN29 SIN31 S1N25 NOR2
XS1I6 SIN41 SIN39 S1N37 NOR2
XS1I7 C B A S1N41 AND3
XS1I8 C B A S1N43 OR2
XS1I16 B A S1N9 OR2
XS1I33 COUT S1N35 INVA
XS1I34 SUM S1N37 INVA
.ENDS CS_ADD1

.SUBCKT INVA OUT IN
MS1I1 OUT IN GND GND N ad=39p as=39p l=1u pd=32u ps=32u w=13u
MS1I2 VDD IN OUT VDD P ad=61.5p as=61.5p l=1u pd=47u ps=47u
w=20.5u
.ENDS INVA
*
*
*
```

ETC.

7.9. Flujo de diseño con IC Validator

Paro crear una *netlist* de parásitos con la herramienta de *IC Validator* para un flujo a nivel transistor, se debe tener una base de datos física, el *netlist* del esquemático, y un ***Runset:*** *script* para generar una base de datos de *IC Validator* como se ve en la Figura 9. La base de datos resultante o el *report file* pueden ser utilizadas para generar el *netlist* parásito, o una extracción del diseño desde un archivo GDS, o también de una librería ***Milkyway:***. Los pasos a seguir para la realización de este flujo son los siguientes: Especificar los archivos de entrada para *IC Validator*, Especificar el *pex_ runset_file* en el ***Runset:*** de ICV. Luego de la extracción todos los resultados obtenidos son almacenados en el ***Runset:*** *report file*.

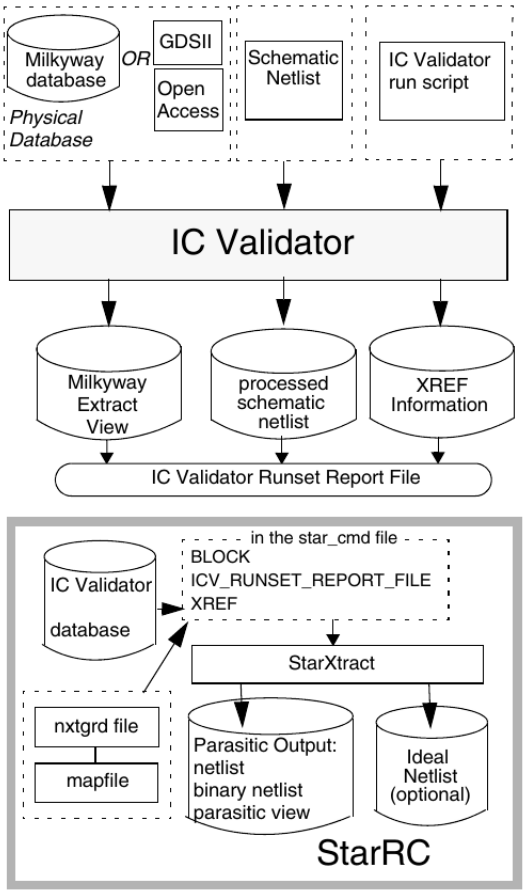


Figura 9: Flujo de diseño de IC Validator [3]

7.10. Preparación del *layout* para una extracción en Custom Compiler:

Antes de realizar las verificaciones (***DRC:***, ***Layout vs Schematic (LVS):*** y ***Layout Parasitic Extraction (LPE):***) es necesario obtener el *layout* del circuito mediante la herramienta *IC Compiler*, durante los pasos de la síntesis física. Para poder hacer una síntesis física con todos los pasos detallados inspeccionar el siguiente trabajo [1]. Una vez obtenido el diseño físico del circuito, procederemos a importarlo a ***Custom Compiler:***. Esta herramienta posee una interfaz gráfica que nos permite realizar verificaciones como **DRC:**, **Layout vs Schematic (LVS):**, y ***Layout Parasitic Extraction (LPE):***. En esta sección se explicó a detalle como importar *layouts* de circuitos previamente diseñados en ICC compiler y para ello debemos trabajar con un conjunto de librerías conocidas como *PDK*.

Las librerías *Process Design Kit (PDK)* son un conjunto de archivos utilizados para modelar procesos de fabricación junto con herramientas dedicadas al diseño de circuitos integrados [8]. Para importar las librerías *PDK* proveídas por la empresa ***TSMC:***, dirigirse a la siguinete ruta `/usr/Synopsys/TSMC/180/CMOS/G/IO3.3V/T-018-CM-SP-018-W1_1_0A` como se ve en la Figura 10. Una vez en esta carpeta abrir una terminal en esta misma ruta y ejecutar el comande `Custom_Compiler`. Luego se abrirá una ventana coma la que se muestra en la Figura 11.

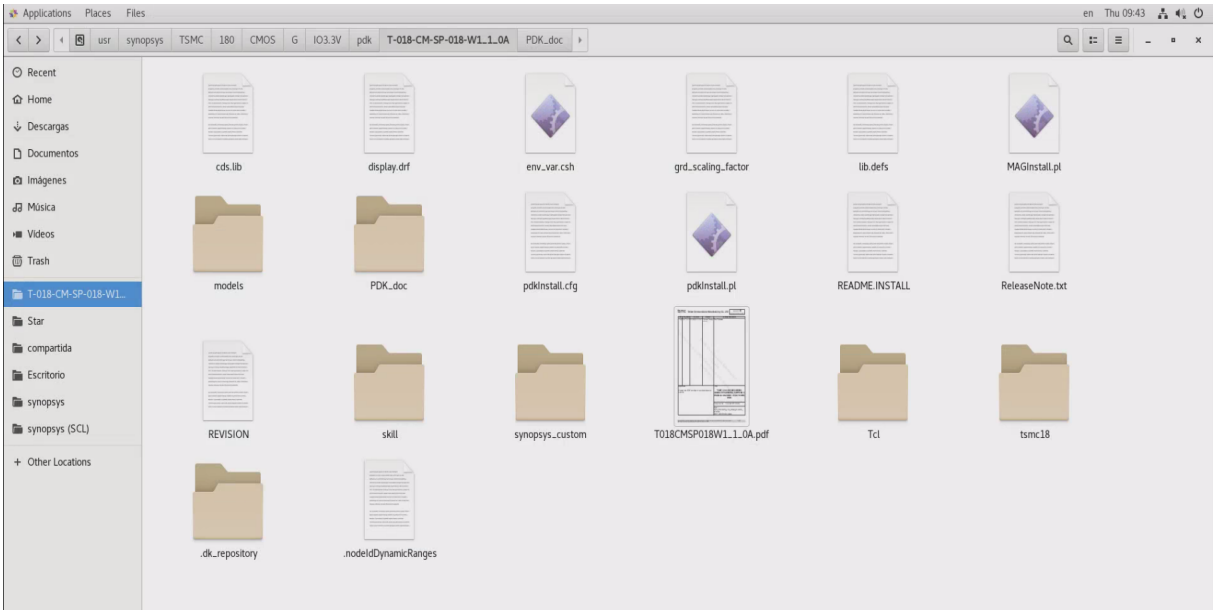


Figura 10: Carpeta de la librería *PDK* de *TSMC*

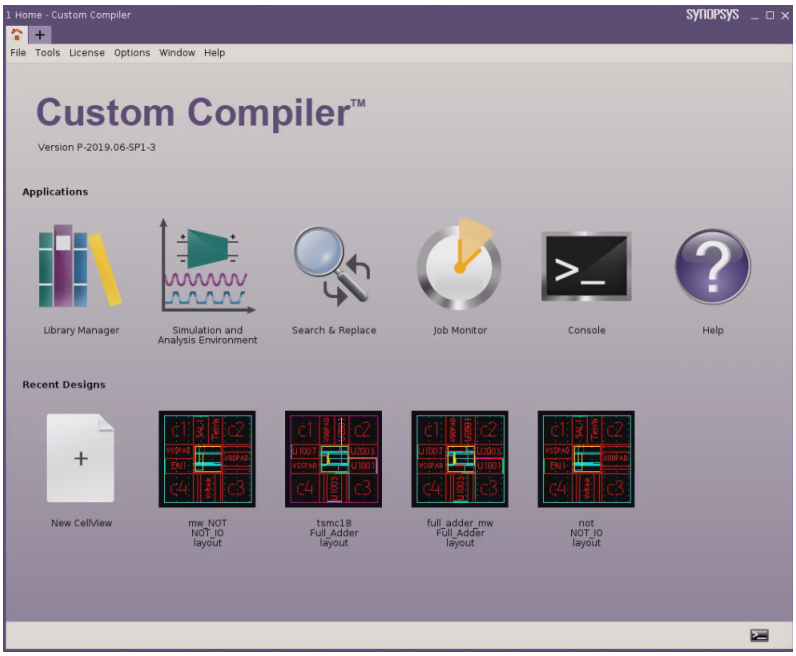


Figura 11: Ventana principal de *Custom Compiler*

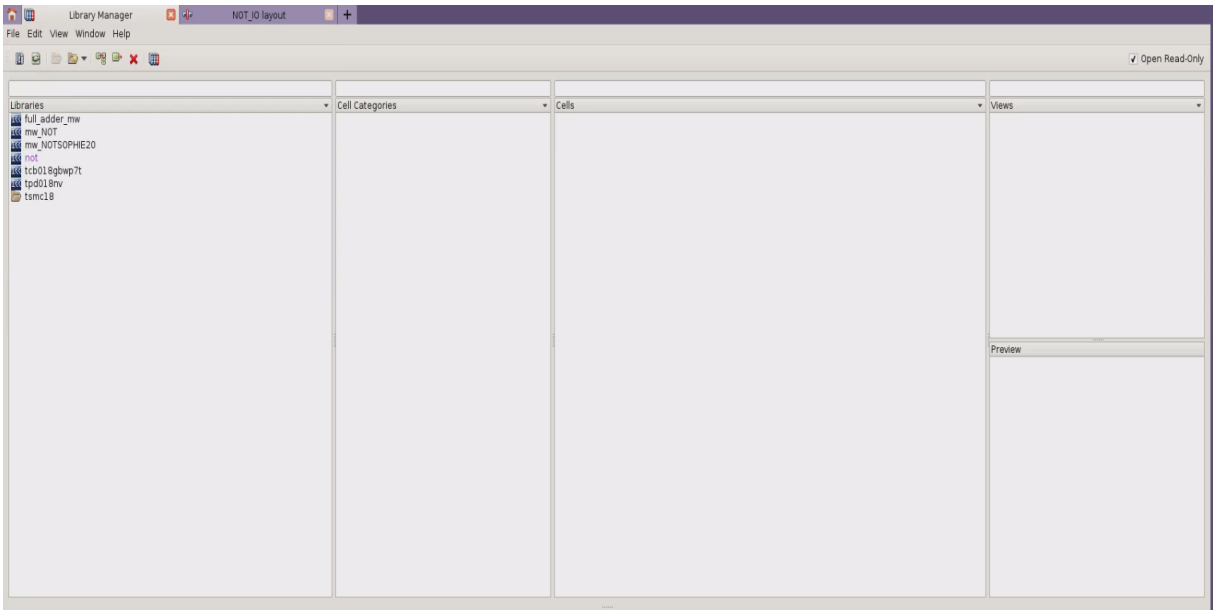


Figura 12: *Library Manager*

Continuamos dando click en el ícono *library Manager* y se mostrará una ventana como en la Figura 12. En esta ventana podemos ver las librerías que se importaron del *PDK* de ***TSMC:*** y libreras que es han importado desde *IC Compiler*. Junto a las librerías podemos observar las *Cell Categories*, aquí se muestran las celdas divididas por categorías tales como: compuertas lógica, transistores, módulos, etc. En el cuadro de *Cells* y *Views* podemos ver la celda seleccionada y las distintas vistas de la misma (*layout*, esquemático, símbolos, ect).

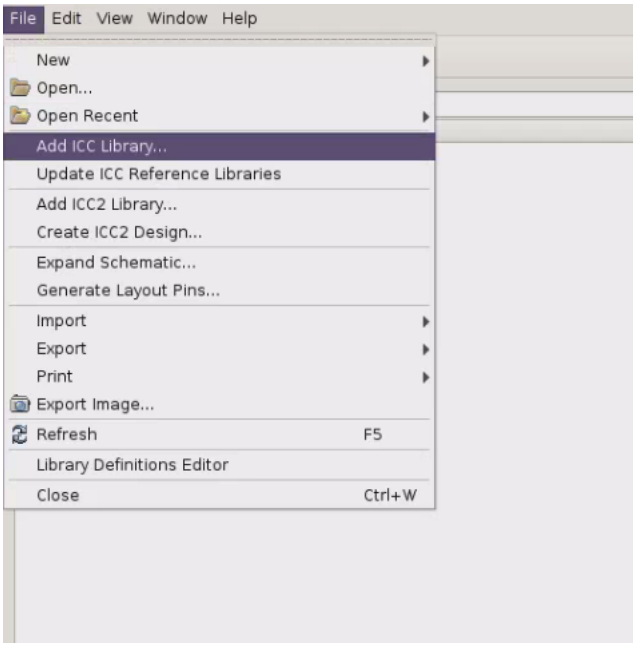


Figura 13: *Add ICC Library*

Para importar una librería hay que dirigirse a `File` y luego a `Add ICC Library` como se observa en la Figura 13. La librería a importar es la ***Milkyway:*** *Library* obtenida durante la síntesis física. Al seleccionar esta opción se abrirá una ventana como en la Figura 14. En esta ventana se debe nombrar la librería y especificar la ruta donde se encuentra la ***Milkyway:***, en este caso importaremos el diseño de una compuerta *not* que se encuentra en la ruta `/usr/synopsys/icc/bin/` con el nombre de `mw_NOT`. Al seleccionar la ***Milkyway:*** esta deberá aparecer en la sección de *Libraries*.

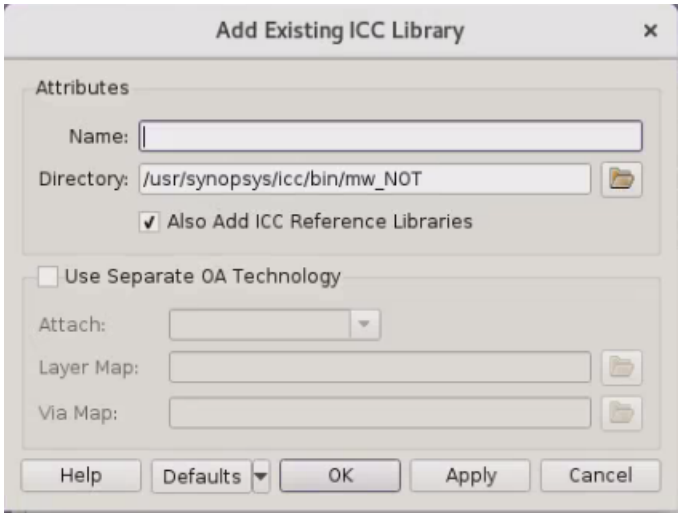


Figura 14: *Add existing ICC Library*

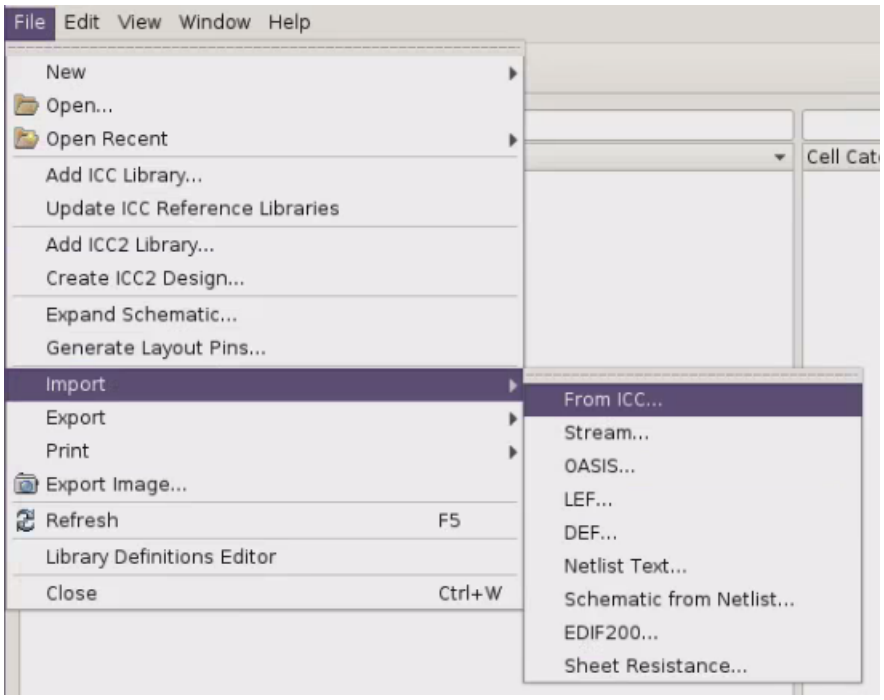


Figura 15: *Import ICC*

También tenemos la opción de importar la base de datos ***Milkyway:*** a una librería previamente creada dentro del paquete *PDK*. Esto le realizamos yendo a *Files*, luego a *Import* y por último a *From ICC* como en la Figura 15. Se abrirá una ventana como en la Figura 16 donde debe ingresarse la dirección de la ***Milkyway:*** y también debe especificarse las celdas que se desean importar. Por último debemos ingresar la carpeta *OA* en donde queremos que aparezca nuestra librería.

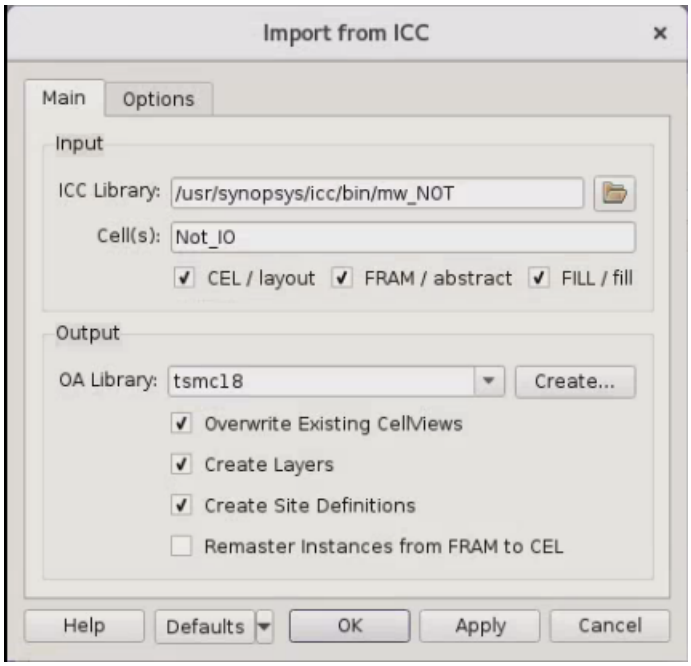


Figura 16: *Import from ICC*

CAPÍTULO 8

Archivo de comandos y preparación para la extracción de parásitos

8.1. Layouts en la herramienta Custom Compiler:

El primer paso para realizar la extracción de parásitos consiste en importar los *Layouts* a ***Custom Compiler:*** utilizando los pasos descritos en el capitula anterior. En este caso se utilizará un diseño de una compuerta not y un diseño de un sumador completo. Estos diseños fueron creados previamente en la herramienta *IC Compiler*. Luego de importarlos tendremos una vista de los circuitos en silicio como en la Figura 17 y en la Figura 18.

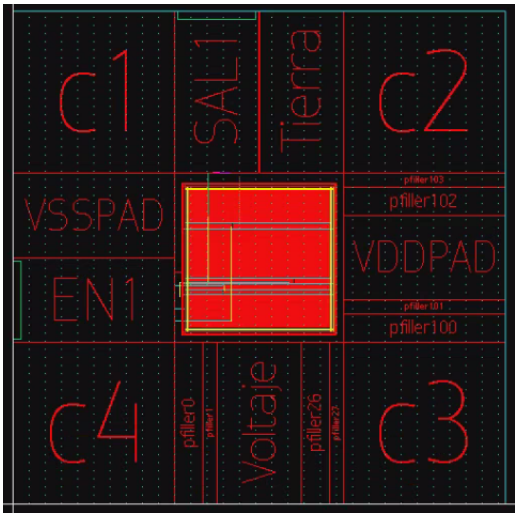


Figura 17: Diseño en silicio de una compuerta not

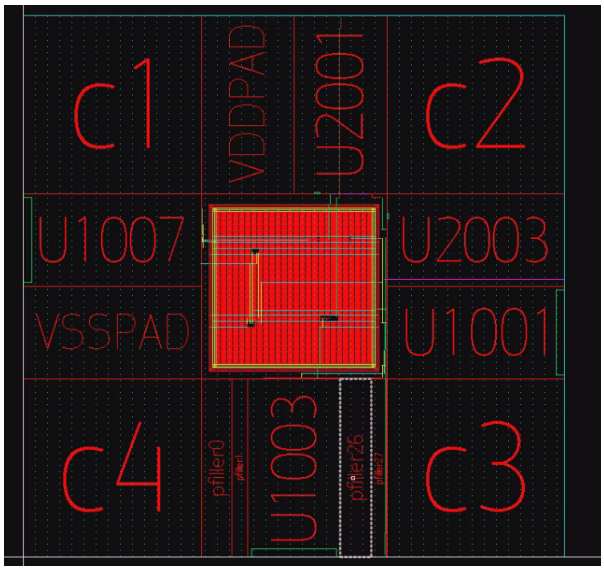


Figura 18: Diseño en silicio de un sumador completo

8.2. Obtención del archivo TCAD_GRD y el archivo Mapping

Estos archivos son indispensables para realizar la extracción de parásitos. El archivo *ITF* define el perfil de una sección transversal del proceso de fabricación y debe ir acompañado de un archivo *mapping* que se encarga de mapear cada capa de una base de datos física a una capa de proceso. Como se mencionó en el capítulo anterior, en la sección del archivo ITF. Podemos tomar dos caminos; correr la herramienta grdgenxo para procesar el archivo ITF y crear un archivo nxtgrd, o especificándolo directamente desde el archivo de comandos de StarRC. Independientemente del camino que tomemos, obtendremos el mismo resultado. Los archivos *ITF* y *TCAD_GRD* pueden encontrarse en la siguiente ruta `/usr/Synopsys/TSMC/180/CMOS/G/IO3.3V/util`. Al dirigirnos a la ruta nos encontraremos con lo archivos comprimidos que se observan en la Figura 19. Cada archivo comprimido representa diferentes valares de *Metal Scheme* y *Metal thickness*. Al descomprimir esta carpeta encontraremos los archivos *ITF* y *TCAD_GRD*. Por suerte ***TSMC:*** provee el archivo nxtgrd, por lo que no tenemos que generarlo. Si este no fuera el caso, tendríamos que ejecutar la herramienta *grdngenxo*.

Figura 19: Carpeta comprimida con los archivo ITF y TCAD_GRD

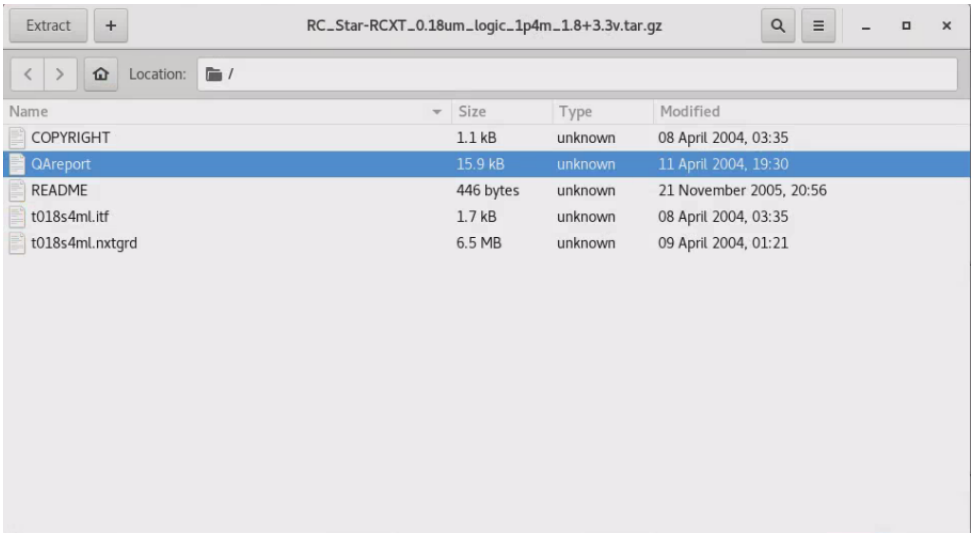


Figura 20: Archivos ITF y TCAD_GRD proveido por TSMC

El archivo *nxtgrd* describe la relación entre nuestro diseño y el proceso de manufactura, lo cual es información sumamente indispensable para calcular elementos parásitos. Para crear este archivo, como ya se mencionó , debemos utilizar la herramienta *grdgenxo*. Este es un programa de utilidad de *StarRC* que tiene como uso principal crear el archivo nxtgrd de un archivo de procesos *ITF*. Este programa se ejecuta desde consola y tiene la siguiente sintaxis: `grdgenxo itf_file`. En la Figura 21, podemos ver una corrida del programa grdgenxo al cual le pasamos el archivo itf, en este caso el archivo t018s6ml.itf. El programa tomará varios minutos en generar el archivo, por lo que hay que ser pacientes. Al finalizar se generará una carpeta comprimida con el archivo nxtgrd adentro, ver Figura 22.

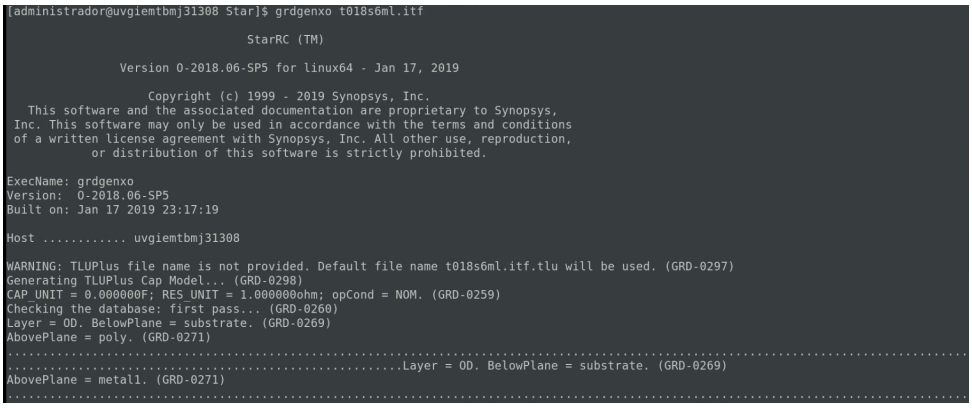


Figura 21: Generación del archivo nxtgrd mediante el programa grdgenxo

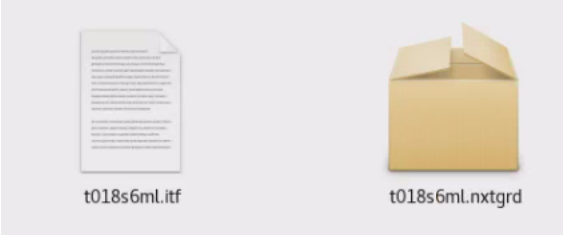


Figura 22: Archivo nxtgrd comprimido

8.3. Preparando el Runset: de IC Validator para un Layout vs Schematic (LVS):

Para obtener el *report file* que se utilizara en la extracción de parásitos. Es necesario hacer cambios al ***Runset:*** del ***Layout vs Schematic (LVS):***. Primero debemos configurar todos los archivos necesarios que *IC Validator* requiere para realizar el ***Layout vs Schematic (LVS):***. Y para ello buscamos en el archivo de ***Runset:*** *ENVIRONMENT SETUP*, aqui configuramos la ruta de *library_name* para que apunte en la direccion donde tenemos el archivo GDS generado en la síntesis física. También debemos de cambiar el nombre de la celda en *SCHEMATIC_TOPCELL*, y la ruta del esquemático cdl en *schematic_file*. Para mas información acerca de como generar el ***Runset:*** para el **Layout vs Schematic (LVS):** chequear [11].

A continuación se muestra parte del ***Runset:*** que debe configurarse:

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

library(
        library_name = "/home/administrador/Escritorio/Star/
            NOT_GDS.gds",
        cell = "Not_IO",
        format = GDSII
);

SCHEMATIC_TOPCELL : string = "Not_IO";// Set schematic top cell
    name here
sch_db = schematic(
        schematic_file = {{"/home/administrador/Escritorio/Star/
            NOT/allicv", ICV}},
        schematic_library_file = {{"/usr/synopsys/TSMC/
            SCRIPTS_NUEVOS/20191128-124344/unit.cdl", SPICE}},
        expand_multiple_devices = true,
        spice_settings = {slash_is_space = false}
);
```

También tenemos que editar la siguiente parte del código, descomentando el `#define RC_DECK`.

```
/* EDIT: The following section contains all of the runset
    variables for RC extraction tools.
*/
#define RC_DECK  // Turn on for LPE/RC extraction
```

La herramienta *StarRC* utiliza el ***Runset:*** *report file* de *IC Validator* para obtener resultados del ***Layout vs Schematic (LVS):*** y realizar extracción de parásitos. Para asegurarse que la extracción corra de forma correcta hay que verificar que el ***Runset:*** contenga la siguiente función. Esta función se encarga de generar los archivos necesarios que luego son utilizados como entrada de la herramienta *StarRc*. En la primera línea de código se genera el archivo *mapping* con el nombre de *STARRCXT.mapping*. Y en la segunda lníea el ***Runset:*** *report* con el nombre *STARRCXT.runset_rep*.

```
pex_map_file = pex_process_map_file(file = "STARRCXT.mapping");
runset_report_file = pex_runset_report_file(file = "STARRCXT.
    runset_rep");
xtr_out = milkyway_library(library_name = "MILKYWAY_XTR");
pex_generate_results(
        pex_matrix = pex_matrix,
        layout_database = xtr_out,
        pex_process_map_file = pex_map_file,
        pex_runset_report_file = runset_report_file,
        precision = 6
);
```

8.3.1. Archivo de comandos de StarRC y creación del Runset: para una extracción a nivel transistor

Un archivo de comandos StarRC es una lista de comandos que especifican las condiciones para una corrida de extracción. Este archivo de comandos es leído por la herramienta StarXtract. Para correr el flujo a nivel transistor con la herramienta IC Validator, se deben utilizar los siguientes comandos. Con el comando *ICV_RUNSET_REPORT_FILE* indicamos el reporte generado después de hacer la prueba del ***Layout vs Schematic (LVS):***, Con el comando BLOCK indicamos el nombre de la celda que queremos extraer. Y los comandos TCAD_GRD_FILE y MAPPING_FILE para especificar la ruta del archivo nxtgrd y mapping respectivamente.

```
ICV_RUNSET_REPORT_FILE: file_name
BLOCK: BLOCK
TCAD_GRD_FILE: technology.nxtgrd
MAPPING_FILE: mapping_file_name
```

TSMC: nos entrega un archivo de comandos que sirve como base para realizar una extracción. El siguiente ejemplo muestra un ejemplo del archivo de comandos utilizado para realizar la extracción de una compuerta *not* previamente diseñada.

```
BLOCK: Not_IO
CASE_SENSITIVE: NO
HIERARCHICAL_SEPARATOR: /

ICV_RUNSET_REPORT_FILE: file_name
TCAD_GRD_FILE: technology.nxtgrd
MAPPING_FILE: mapping_file_name

EXTRACTION: RC
REDUCTION: YES
DENSITY_BASED_THICKNESS: YES
 EXTRACT_VIA_CAPS: NO
REMOVE_FLOATING_NETS: YES
REMOVE_DANGLING_NETS: YES
POWER_NETS: VDD VSS
SKIP_CELLS: !*
TRANSLATE_RETAIN_BULK_LAYERS: YES

NETLIST_FORMAT: NETNAME
NETLIST_PASSIVE_PARAMS: YES
 NETLIST_FILE: adderstract.sp

COUPLE_TO_GROUND: NO
COUPLING_REPORT_FILE: cc.rep

XREF: YES
XREF_USE_LAYOUT_DEVICE_NAME: YES
CELL_TYPE: LAYOUT
NET_TYPE: LAYOUT

SKIP_PCELLS : cfmom* cfmom_mx* cfmom_rf* crtmom* crtmom_rf*
    ind_std* ind_std_40k* ind_sym* ind_sym_40k* ind_sym_ct*
    ind_sym_ct_40k* jvar* lcesd1_rf* lcesd2_rf* lowcpad_rf*
    mimcap_rf* mimcap_rf_2p0* mos_var* mos_var33* moscap_rf*
    moscap_rf33* moscap_rf33_nw* moscap_rf_nw* ndio_hia_rf*
    ndio_sbd_mac* pdio_hia_rf* rfnmos2v* rfnmos2v_6t* rfnmos3v*
    rfnmos3v_6t* rfpmos2v* rfpmos2v_5t* rfpmos2v_nw* rfpmos2v_nw_5t
    * rfpmos3v* rfpmos3v_5t* rfpmos3v_nw* rfpmos3v_nw_5t*
    rphpoly_rf* rphripoly_rf* rplpoly_rf* sbd_rf* sbd_rf_nw*
    spiral_std_m2u_a_33k* spiral_std_m2u_x_33k* spiral_std_mu_a_33k
    * spiral_std_mu_x_20k* spiral_std_mu_x_33k* spiral_std_mu_x_40k
    * spiral_sym_ct_m2u_u_a_33k* spiral_sym_ct_m2u_u_x_33k*
    spiral_sym_ct_mu_x_20k* spiral_sym_ct_mu_x_33k*
    spiral_sym_ct_mu_x_40k* spiral_sym_ct_mu_x_a_33k*
```

spiral_sym_m2u_u_33k* spiral_sym_mu_x_20k* spiral_sym_mu_x_33k* spiral_sym_mu_x_40k*

Comandos de StarRC

En esta sección se describen los comandos utilizados en el archivo de comandos de StarRC. Para mas información acerca de los comandos leer [3]

- **BlOCK**: Define el nombre del bloque o celda que va a ser extraído.
- **EXTRACTION**: Especifica es tipo de extraccion a realizar. Puede ser una extraccion RC, C, R , RLC.
- **DENSITY_BASED_THICKNES**: Habilita el calculo de variación de densidad y grosor durante la extracción.
- **EXTRACT_VIA_CAPS**: Realiza una extracción detallada capacitancias en las vias.
- **REMOVE_FLOATING_NETS**: Especifica si hay que remover las *nets* que no tienen conexiones conectándolos a tierra.
- **REMOVE_DANGLING_NETS**: Especifica si hay que identificar *nets* colgadas y reasignarlas a tierra (eliminándolas).
- **POWER_NETS**: Especifica *power nets* para que se les de un trata especial durante la extracción.
- **SKIP_CELLS**: Omite las listas de celdas especificadas durante la extracción.
- **TRANSLATE_RETAIN_BULK_LAYERS**: Especifica como tratar *mapped bulk layers* durante una extracción a nivel transistor.
- **NETLIST_FORMAT**: Define el formato de salida para del *netlist*. Entre los formatos tenemos: SPF, SPEF, OA, NETNAME, STAR, PARAMETERIZED_SPICE.
- **NETLIST_PASSIVE_PARAMS**: Genera o no parametros de dispasitivos pasivos en el *netlist*.
- **NETLIST_FILE**: Indica el nombre del *netlist* de salida. Por defecto se llama *block_name*.
- **COUPLE_TO_GROUND**: Indica si las capacitancias de acople son retenidas o agrupadas a tierra.
- **COUPLING_REPORT_FILE**: Genera un reporte con una lista de las capacitancias de acople por *net*.
- **XREF**: Especifica el conjunto de nombres usados en la *netlist* generada.
- **XREF_USE_LAYOUT_DEVICE_NAME**: Especifica si usar o no el nombre de los componentes del *layout*.

- **CELL_TYPE**: Especifica si usar el nombre de las celdas del esquematico o del *layout*.
- **NET_TYPE**: Especifica si usar nombres de *layout* o celdas durante selección de datos.

CAPÍTULO 9

Extracción y salida de StarRC

9.1. Verificación Layout vs Schematic

Antes de iniciar con la extracción de parásitos, debemos hacer la verificación de ***Layout vs Schematic (LVS):***. Para abrir la ventana de configuración de ***Layout vs Schematic (LVS):*** dentro del *layout* importado, seleccionamos *Verification*, luego damos *click* en la opción ***Layout vs Schematic (LVS):*** y por último en la opción *Setup and Run*. Seleccionada esta opción se abrirá una ventana como en la Figura 23. En esta ventana seleccionamos el ***Runset:*** de ***Layout vs Schematic (LVS):*** y seleccionamos todas las opciones necesarias para ejecutar la herramienta. Para mas información acerca de como correr ***Layout vs Schematic (LVS):*** chequear [11].

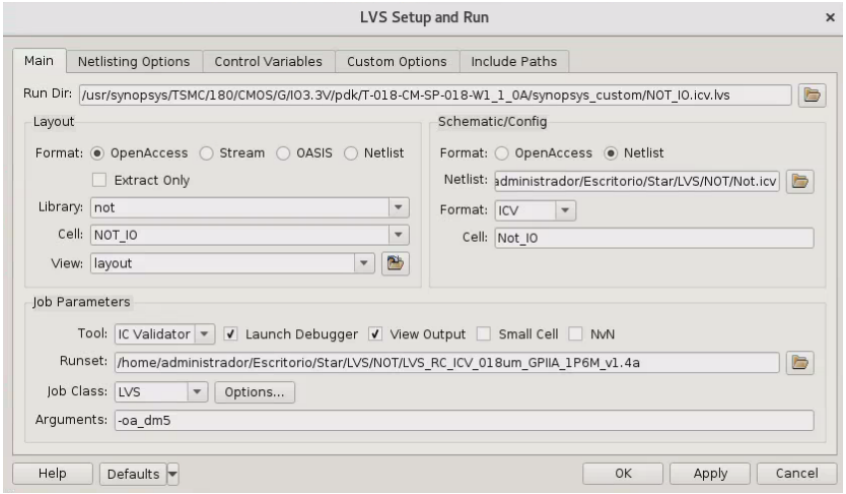


Figura 23: Configuración del LVS

Corremos la herramienta dando *click* en la opción *OK*. Se abrirá un programa auxiliar llamado *VUE* y nos indicará si hay un error de comparación entre el esquemático y el diseño en silicio. Si no hay errores, la herramienta nos dirá que la prueba ha pasado. En las Figuras 24 y 25 vemos el resultado de la verificación ***Layout vs Schematic (LVS):*** del diseño de la *not* y el sumador completo.

Figura 24: Verificación LVS exitosa de la compuerta not

Figura 25: Verificación LVS exitosa de un sumador completo

Después de haber realizado la verificación ***Layout vs Schematic (LVS):***, se generará una carpeta donde están todos los archivos creados por la herramienta ***Layout vs Schematic (LVS):***. La ruta de esta carpeta se puede indicar en las opciones de la ventana de configuración de ***Layout vs Schematic (LVS):***. Por defecto esta ruta apunta al directorio *synopsys_custom* y se guarda con el nombre de la celda, ejemplo: *celdaicv.lvs*. En la figura 26 , se observan los archivos generados por la herramienta de ***Layout vs Schematic (LVS):***. De esta carpeta nos interesan varios archivo. Uno de estos archivos es el *STARRCXT.runset_rep*. Tenemos que observar si se encuentran las carpetas *MILKYWAY_XTR* y *run_details*, ya que el ***Runset:*** *report file* hace referencia a ellas. El otro archivo importante es el *STARRCXT.mapping*. Los archivos ***Runset:*** *report file* y mapping se indican en el archivo de comandos de *StarRC* mediante los comandos ICV_RUNSET_-

REPORT_FILE y MAPPING_FILE.

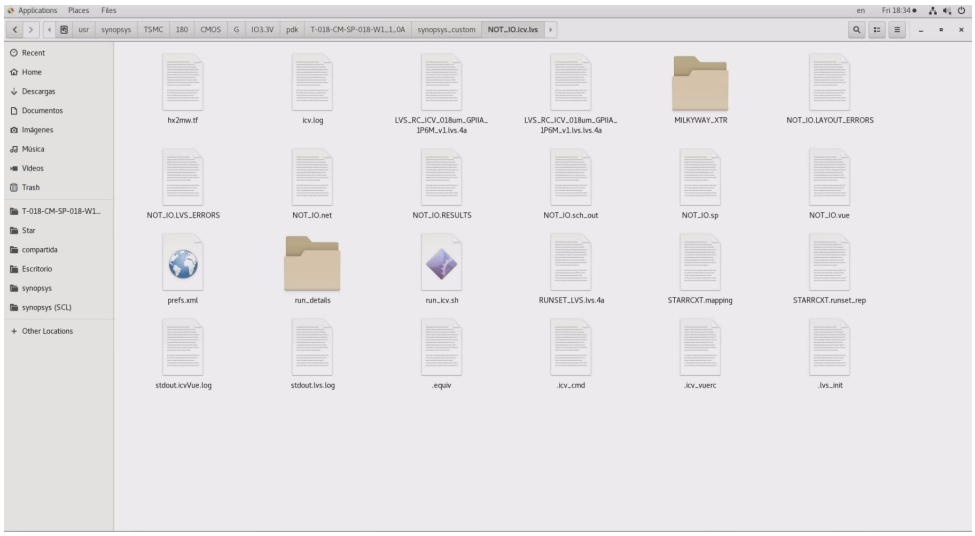


Figura 26: Carpeta de resultados después de la verificación LVS

9.2. Extracción a nivel transistor de un circuito inversor y un sumador completo con librerías de *TSMC:* utilizando la herramienta Custom Compiler:

La herramienta de **Custom Compiler:** es un ambiente de diseño que unifica el diseño, simulación, verificación física, extracción de parásitos y análisis de tiempo. La gran ventaja de utilizar **Custom Compiler:** es que el ambiente de trabajo provee de características que nos permite interactuar de forma fácil con los distintos archivos necesarios para realizar la extracción.

Para abrir la herramienta de verificación de ***Layout Parasitic Extraction (LPE):*** en *Custom compiler* hay que dirigirse a *verification*, ver Figura 27. Luego se abrirá una ventana con las configuraciones principales para la extracción. En esta ventana se especifica el directorio de trabajo, el diseño al cual se le realizará la extracción, el archivo de comandos y si es necesario argumentos para el comando StarXtract, ver Figura 28. Antes de realizar la extracción, es recomendable tener una carpeta con el archivo de comandos y el archivo *nxtgrd*, para hacer referencia a ellos desde la misma carpeta.

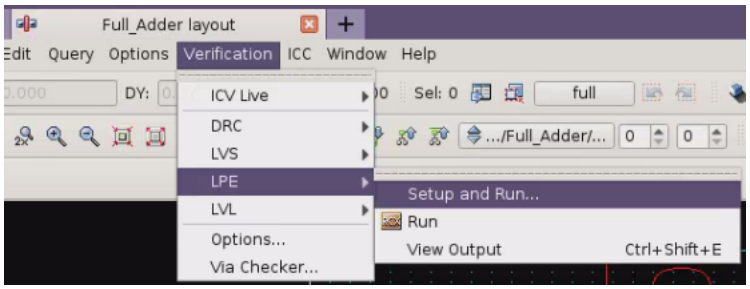


Figura 27: Ventana de verificación

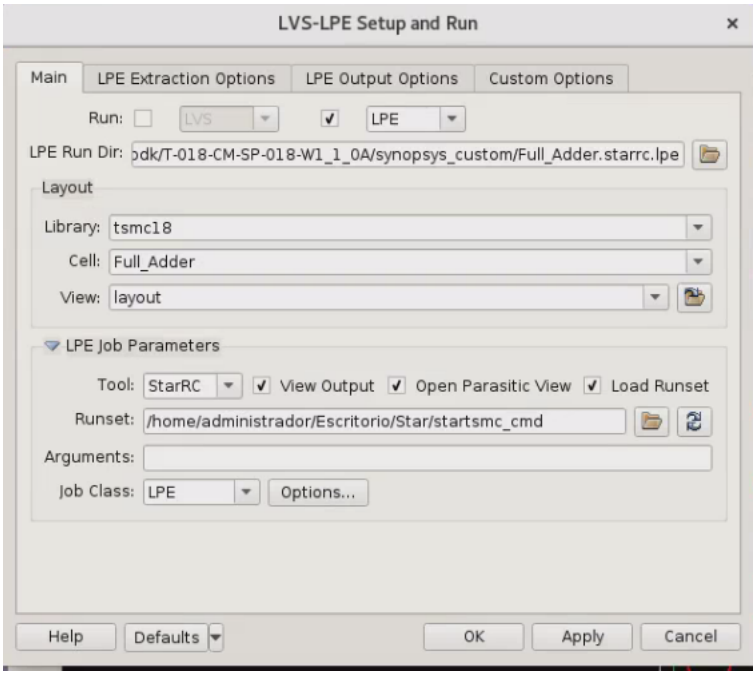


Figura 28: Ventana principal del LPE

En la ventana de opciones de extracción, se debe especificar el ***Runset:*** *report file*, el cual es creado por la herramienta de ***Layout vs Schematic (LVS):*** y el cual es necesario como entrada para la herramienta *StarRC*. También podemos especificar opciones tales como la *corner* de interés, la temperatura de operación, el tipo de extracción, y la cantidad de reducción que se realizara, ver Figura 29. El ***Runset:*** utilizado es el que se describió en el capítulo anterior, tanto para la extracción de la compuerta not, como el sumador completo.

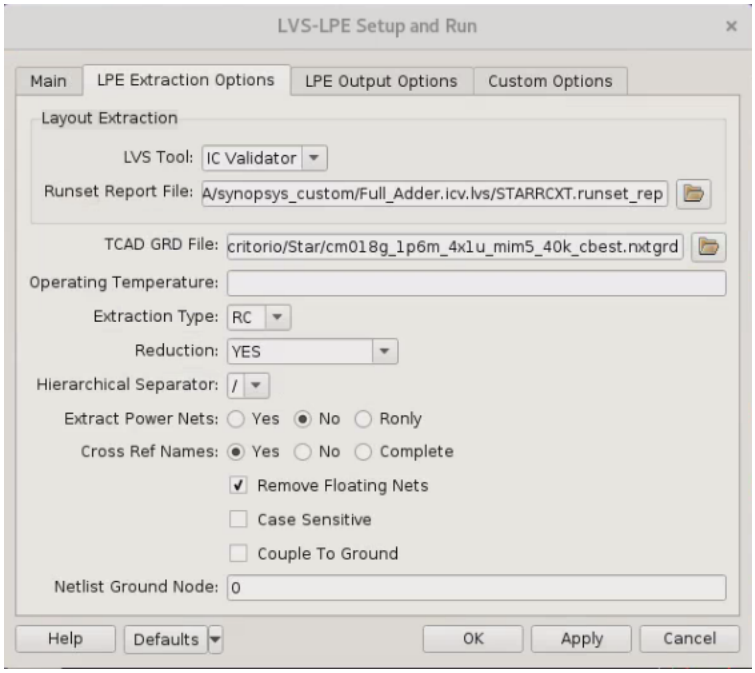


Figura 29: Opciones de extracción

En la ventana de salida podemos seleccionar el *netlist* de salida (SPEF, SPF. NETNAME ect). También especificamos la ruta del ***Runset:*** de salida, este archivo es una copia del archivo de comandos de *StarRC*. Por último especificamos el nombre del *netlist* de salida y verificar que la casilla *Use* ***Layout vs Schematic (LVS):*** *Netlist Port Order* este desmarcada, ver Figura 30

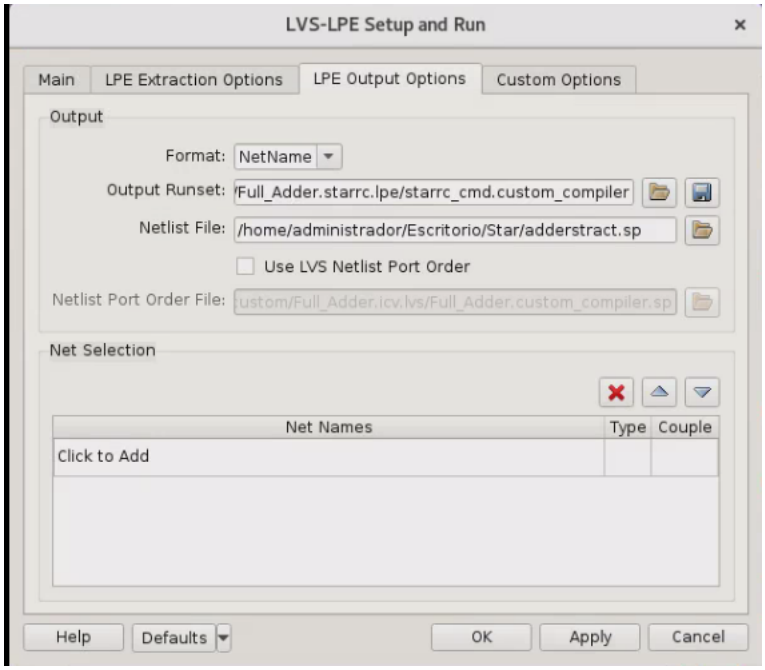


Figura 30: Opciones de salida de la extracción

Después de completar toda la configuración, damos *click* en *OK*, en la ventana de configuración del ***Layout Parasitic Extraction (LPE):***, y la herramienta *StarRC* iniciará a realizar la extracción. Al terminar la extracción la consola de ***Custom Compiler:*** nos mostrara un mensaje diciendo que la extracción se ha completado. Si hay errores la consola también nos lo hará saber. En la Figura 31 y Figura 32 podemos ver la extracción completada para la compuerta not y el sumador completo, sin errores.

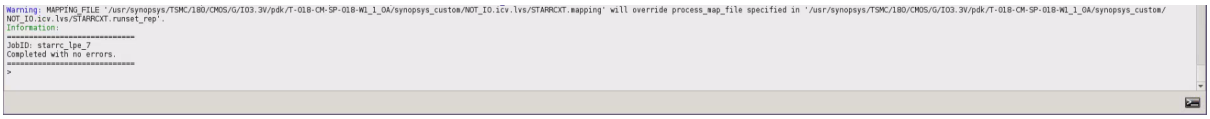


Figura 31: Extracción completada de la compuerta not

Figura 32: Extracción completada del sumador completo

9.3. Directorios y archivos

Al finalizar la extracción, *StarRC* creara una carpeta con archivos de reportes, errores, *netlist* de salida, el directorio *star*, entre otros. El directorio *star* puede crearse mediante el comando STAR_DIRECTORY desde el ***Runset:*** de *StarRC* o dentro de la carpeta de corrida (*run directory*), especificando la ruta desde la ventana de verificación principal de *LPE*, Figura 28.

En la Figura 33 podemos ver la carpeta generada luego de realizar la extracción de la compuerta Not. Otro archivo generado que también es importante es *summary file*. Este archivo contiene información, advertencias, y mensajes de error que ocurren durante la corrida del ***Runset:***. El archivo también reporta el tiempo de corrida durante la extracción, tiempo de *CPU* y memoria usada.

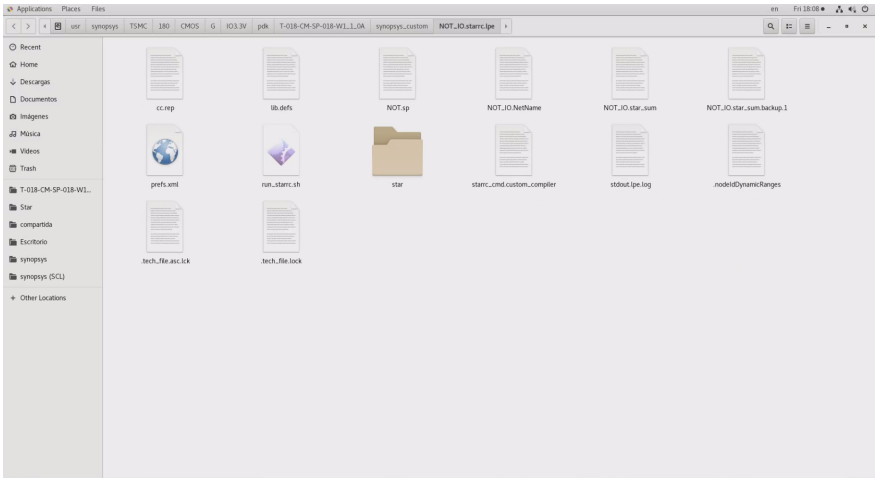


Figura 33: Directorio Star

9.4. Netlists de salida

A continuación se mostrarán los *netlist* obtenidos de la extracción de parásitos para la compuerta *not* y el sumador completo. Solo se añadieron ciertas partes del *netlist* ya que es un documenta extenso. Estos *netlists* fueron extraídos con el formato *Netname*, o sea son un documento *SPICE*. Podemos ver que al inicio el *netlist* tiene un encabezado con las opciones utilizadas para la extracción; como la temperatura de operación, el archivo *TCAD_GRD_FILE*, el formato del archivo y otros datos como la fecha y el nombre del diseño. Luego se muestra un subcircuito del diseño en silicio con todas las *nets* que lo componen. Cada *net* esta conectado a un conjunto de resistencias y capacitares que fueron obtenidos durante la extracción. Hasta el final del archivo tenemos las instancias de los componentes del diseño como transistores y compuertas lógicas.

Netlist obtenido de la compuerta Not

```
*
*|DSPF 1.3
*|DESIGN Not_IO
*|DATE "Mon␣Sep␣21␣12:16:52␣2020"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "O-2018.06-SP5"
*|DIVIDER |
*|DELIMITER :
**FORMAT NETNAME
*

** COMMENTS
```

```
** OPERATING_TEMPERATURE 25
** GLOBAL_TEMPERATURE 25
** TCAD_GRD_FILE /home/administrador/Escritorio/Star/
    cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
**    TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
**    TCADGRD_VERSION 62


.SUBCKT Not_IO

*|GROUND_NET 0

*|NET ln_VSSPAD|N__generated_11 0.038773PF
*|S (ln_VSSPAD|N__generated_11:485 87.4750 188.4000) // $llx
    =87.4750 $lly=179.4800 $urx=87.4750 $ury=197.3200 $lvl=3
*|S (ln_VSSPAD|N__generated_11:1822 115.0000 188.4000) // $llx
    =115.0000 $lly=179.4800 $urx=115.0000 $ury=197.3200 $lvl=3
Cg70_1 ln_VSSPAD|N__generated_11:485 0 6.61045e-16
R70_1 ln_VSSPAD|N__generated_11:485 ln_VSSPAD|N__generated_11:1822
     0.120345 $l=27.5250 $w=17.8400 $lvl=3

*|NET ln_VSSPAD|N__generated_10 0.0292841PF
*|S (ln_VSSPAD|N__generated_10:486 87.4750 205.2000) // $llx
    =87.4750 $lly=198.6800 $urx=87.4750 $ury=211.7200 $lvl=3
*|S (ln_VSSPAD|N__generated_10:1821 115.0000 205.2000) // $llx
    =115.0000 $lly=198.6800 $urx=115.0000 $ury=211.7200 $lvl=3
Cg69_2 ln_VSSPAD|N__generated_10:486 0 6.23207e-16
R69_2 ln_VSSPAD|N__generated_10:486 ln_VSSPAD|N__generated_10:1821
     0.164644 $l=27.5250 $w=13.0400 $lvl=3



*
* Instance Section
*
XXCom|XU1 ln_N_2:F49 ln_N_11:F50 ln_N_7:F51 ln_N_9:F52 ln_N_14:F53
     ln_N_15:F54 CKND0BWP7T
XXEN1 ln_N__generated_17:F65 ln_N__generated_20:F66
    ln_N__generated_19:F67 ln_N__generated_16:F68 ln_N_6:F69
    ln_N__generated_18:F70 ln_N_5:F71 ln_N_7:F72 ln_N_8:F55 ln_N_12
    :F56 ln_N_3:F57 ln_N_10:F58 ln_N_9:F59 ln_N_3:F60 ln_N_11:F61
    ln_N_13:F62 ln_N_14:F63 ln_N_15:F64 PDDW0204SCDG
XXSAL1 ln_N_3:F83 ln_N_3:F84 ln_N_3:F85 ln_N_3:F86 ln_N_4:F87
    ln_N_2:F88 ln_N_5:F89 ln_N__generated_26:F90 ln_N_8:F73 ln_N_12
    :F74 ln_N__generated_25:F75 ln_N__generated_23:F76
    ln_N__generated_21:F77 ln_N__generated_24:F78
    ln_N__generated_22:F79 ln_N_13:F80 ln_N_14:F81 ln_N_15:F82
    PDDW0204SCDG
```

```
XXU6 ln_N_3:F107 ln_N_7:F108 ln_N_9:F109 ln_N_14:F110 ln_N_15:F111
     TIELBWP7T
XXU7 ln_N_10:F112 ln_N_7:F113 ln_N_9:F114 ln_N_14:F115 ln_N_15:
    F116 TIEHBWP7T
.ENDS
```

Netlist obtenido del sumador completo

```
*
*|DSPF 1.3
*|DESIGN Full_Adder
*|DATE "Fri␣Sep␣18␣22:07:30␣2020"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "O-2018.06-SP5"
*|DIVIDER /
*|DELIMITER :
**FORMAT NETNAME
*

** COMMENTS

** OPERATING_TEMPERATURE 25
** GLOBAL_TEMPERATURE 25
** TCAD_GRD_FILE /home/administrador/Escritorio/Star/
    cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
**    TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
**    TCADGRD_VERSION 62


.SUBCKT Full_Adder

*|GROUND_NET 0


*|NET ln_N_32 0PF
*|I (ln_N_32:F198 U1007 ln_N_25 B 0 -0.0025 175.0000)
*|I (ln_N_32:F173 U1003 ln_N_25 B 0 72.5575 175.0000)
*|I (ln_N_32:F260 U5 ln_N_6 B 0 102.5575 175.0000)
*|I (ln_N_32:F223 U2001 ln_N_25 B 0 117.6175 175.0000)
*|I (ln_N_32:F265 U6 ln_N_6 B 0 211.5200 203.1650)
*|I (ln_N_32:F248 U2003 ln_N_25 B 0 117.6175 175.0000)
*|I (ln_N_32:F148 U1001 ln_N_25 B 0 119.9950 175.0000)
R26_3 ln_N_32:F198 ln_N_32:F173 0.001
R26_4 ln_N_32:F198 ln_N_32:F260 0.001
R26_5 ln_N_32:F260 ln_N_32:F223 0.001
R26_6 ln_N_32:F223 ln_N_32:F265 0.001
R26_7 ln_N_32:F265 ln_N_32:F248 0.001
```

```
R26_8 ln_N_32:F265 ln_N_32:F148 0.001

*|NET ln_N_31 4.97307PF
*|I (ln_N_31:F247 U2003 ln_N_24 B 0 235.8550 175.0000)
*|I (ln_N_31:F222 U2001 ln_N_24 B 0 235.2400 291.8375)
*|I (ln_N_31:F147 U1001 ln_N_24 B 0 235.8550 57.5000)
*|I (ln_N_31:F197 U1007 ln_N_24 B 0 114.3850 173.2600)
*|I (ln_N_31:F172 U1003 ln_N_24 B 0 145.1200 57.5000)
Cg25_3 ln_N_31:F222 0 6.62798e-15
Cg25_4 ln_N_31:F147 0 1.52132e-14
Cg25_5 ln_N_31:F197 0 4.76768e-15
Cg25_6 ln_N_31:690 0 3.91616e-14
R25_9 ln_N_31:1277 ln_N_31:F222 0.0235665
R25_10 ln_N_31:1277 ln_N_31:690 0.039632
R25_11 ln_N_31:F247 ln_N_31:1311 0.0397687
R25_12 ln_N_31:F247 ln_N_31:F147 0.0803035
R25_13 ln_N_31:1311 ln_N_31:F222 0.0405348
R25_14 ln_N_31:F147 ln_N_31:F172 0.099685
R25_15 ln_N_31:F197 ln_N_31:1220 0.0418934
R25_16 ln_N_31:F197 ln_N_31:690 0.171755
R25_17 ln_N_31:1220 ln_N_31:F172 0.0204292

*
* Instance Section
*
XU1001 ln_N_5:F142 ln_N_14 ln_N__generated_41 ln_N__generated_37
   ln_N_5:F153 ln_N__generated_40 ln_N__generated_36 ln_N_5:F156
   ln_N__generated_39 ln_N_8:F133 ln_N__generated_42 ln_N_5:F135
   ln_N__generated_35 ln_N_8:F137 ln_N_17:F138 ln_N__generated_34
   ln_N_5:F140 ln_N__generated_38 ln_N__generated_43 ln_N_6:F144
   ln_N_5:F145 ln_N_21 ln_N_31:F147 ln_N_32:F148 ln_N_33:F149
   PDDW0204SCDG
XU1003 ln_N__generated_49 ln_N_10 ln_N__generated_51
   ln_N__generated_47 ln_N__generated_54 ln_N_6:F179 ln_N_7 ln_N_5
   :F181 ln_N_6:F157 ln_N__generated_45 ln_N__generated_52 ln_N_5:
   F160 ln_N__generated_44 ln_N_8:F162 ln_N_8:F163 ln_N_6:F164
   ln_N_5:F165 ln_N__generated_48 ln_N__generated_55
   ln_N__generated_50 ln_N__generated_46 ln_N__generated_53
   ln_N_31:F172 ln_N_32:F173 ln_N_33:F174 PDDW0204SCDG
XU1007 ln_N__generated_60 ln_N_11 ln_N__generated_64
   ln_N__generated_59 ln_N__generated_67 ln_N__generated_63
   ln_N__generated_58 ln_N_5:F206 ln_N__generated_62
   ln_N__generated_57 ln_N__generated_66 ln_N_5:F185 ln_N_4:F186
   ln_N__generated_65 ln_N__generated_61 ln_N__generated_56 ln_N_5
   :F190 ln_N_17:F191 ln_N_17:F193 ln_N_4:F194 ln_N_4:F195 ln_N_20
    ln_N_31:F197 ln_N_32:F198 ln_N_33:F199 PDDW0204SCDG
XU2001 ln_N_4:F217 ln_N_2 ln_N_4:F226 ln_N_4:F227 ln_N_4:F228
   ln_N_4:F229 ln_N_12 ln_N_5:F231 ln_N__generated_74
```

```
    ln_N__generated_70 ln_N_9:F209 ln_N_5:F210 ln_N__generated_69
    ln_N__generated_76 ln_N__generated_73 ln_N__generated_68 ln_N_5
    :F215 ln_N__generated_72 ln_N__generated_78 ln_N__generated_75
    ln_N__generated_71 ln_N__generated_77 ln_N_31:F222 ln_N_32:F223
     ln_N_33:F224 PDDW0204SCDG
XU2003 ln_N_5:F242 ln_N_3 ln_N_5:F251 ln_N__generated_82 ln_N_5:
    F253 ln_N_5:F254 ln_N_13 ln_N_5:F256 ln_N_18:F232 ln_N_18:F233
    ln_N__generated_87 ln_N_5:F235 ln_N__generated_80
    ln_N__generated_86 ln_N__generated_84 ln_N__generated_79 ln_N_5
    :F240 ln_N__generated_83 ln_N_9:F243 ln_N__generated_85
    ln_N__generated_81 ln_N__generated_88 ln_N_31:F247 ln_N_32:F248
     ln_N_33:F249 PDDW0204SCDG
XU5 ln_N_4:F257 ln_N_15:F258 ln_N_19:F259 ln_N_32:F260 ln_N_33:
    F261 TIELBWP7T
XU6 ln_N_17:F262 ln_N_15:F263 ln_N_19:F264 ln_N_32:F265 ln_N_33:
    F266 TIEHBWP7T

.ENDS
```

CAPÍTULO 10

Simulación del diseño completo a nivel ***HSPICE:***

10.1. Celdas *Black-Box*

Cuando se trabaja con diseño de celdas no terminados, El *netlist* del diseño no posee los puertos de entrada, salida, y alimentación comparado con el esquemático. El diseño no esta lo suficientemente completo para tener todas las interacciones jerárquicas necesarias para crear puertos [12]. En la Figura 34, el puerto A no fue extraídos. Esto provocará en la verificación de ***Layout vs Schematic (LVS):*** aparezca el siguiente mensaje de error.

```
Proccesing black boxes' pins...
ERROR: Schematic port "A" does not have a corresponding port in
    the layout in blackbox {invb, invb}.
```

Schematic: ADD4.sp

```
.SUBCKT invb GND VDD A Z
M1 GND A Z GND n L=1u W=13u
M2 VDD A Z VDD p L=1u W=20.5u
.ENDS

.SUBCKT nor2b GND VDD QN A B
M1 QN B GND GND n L=1u W=13u
M2 QN A GND GND n L=1u W=13u
M3 n11 B QN VDD p L=1u W=20.5u
M4 n11 A VDD VDD p L=1u W=20.5u
.ENDS
```

Layout: add4.net

```
{CELL invb
  {PORT VDD GND Z }
  {PROP top=50.000000
bottom=0.000000
     left=0.000000 right=12.000000}
}

{CELL nor2b
  {PORT VDD GND QN A B}
  {PROP top=50.000000
bottom=0.000000
     left=0.000000 right=18.000000}
}
```

Figura 34: Puertos no extraídos

Entonces para solucionar este problema durante el ***Layout vs Schematic (LVS):*** es

necesario remover los puertos usando la siguiente función remove_schematic_ports como argumento de la función lvs_black_box_options(). C Por ejemplo:

```
lvs_black_box_options(
    equiv_cells          = {{"invb","invx"}},
    remove_schematic_ports = {"A"}
);
```

En la extracción que se mostrará a continuación de la compuerta *not* los puertos de entrada y salida del sub-circuito no están presentes debido al resultado del ***Layout vs Schematic (LVS):***.

```
*
*|DSPF 1.3
*|DESIGN Not_IO
*|DATE "Mon␣Sep␣21␣12:16:52␣2020"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "O-2018.06-SP5"
*|DIVIDER |
*|DELIMITER :
**FORMAT NETNAME
*

** COMMENTS

** OPERATING_TEMPERATURE 25
** GLOBAL_TEMPERATURE 25
** TCAD_GRD_FILE /home/administrador/Escritorio/Star/
   cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
**    TCAD_TIME_STAMP Tue Apr 23 22:51:06 2019
**    TCADGRD_VERSION 62


.SUBCKT Not_IO   Puertos_faltantes
```

Debido a que la empresa ***TSMC:*** solamente nos ha proveído de un *kit* académico, los componentes que encontraremos en las librerías serán solamente *black boxes*. Este diseño *black box* solamente contiene el diseño en silicio del componente, faltando entonces el *netlist* del esquemático ver Figura 37. Sin los modelos completos de los componentes, hacer simulaciones del circuito se vuelve una tarea complicada. Una solución a esto es hacer de forma manual las conexiones a los puertos de entrada, salida y alimentación, también sustituir las instancias de los componentes *black box*, con componentes completos. Y con base en esto hacer simulaciones que tienen como fin darnos una idea de cómo funcionará el circuito.

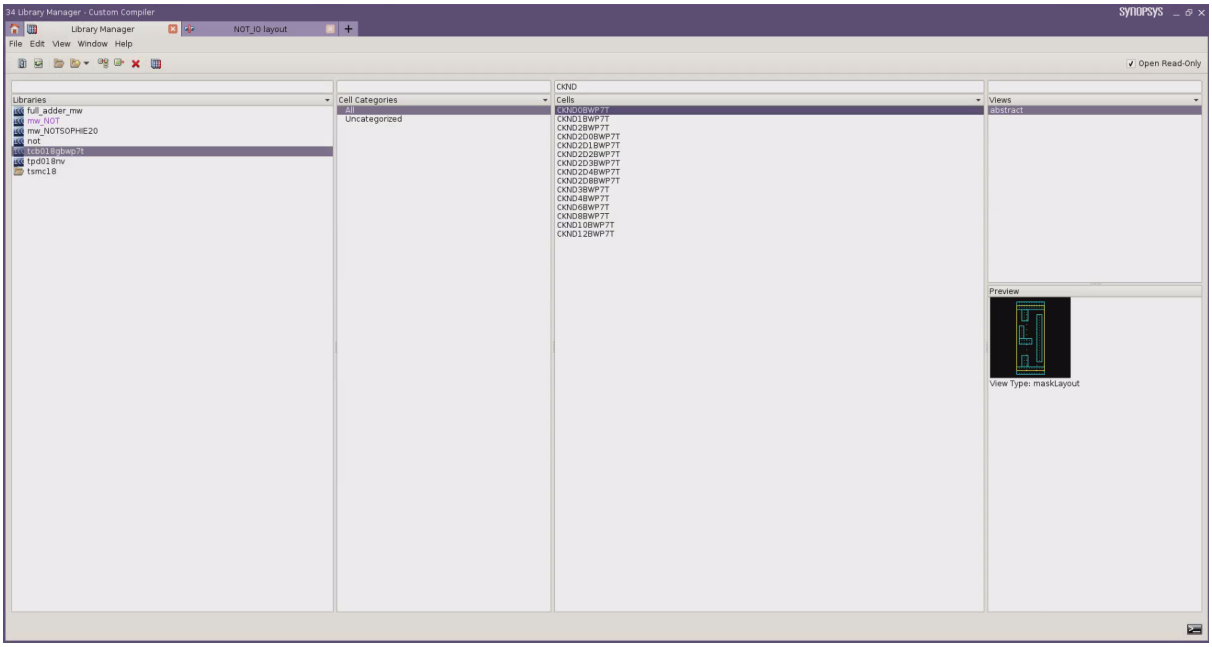


Figura 35: Componente de TSMC solo con esquemático

CAPÍTULO 11

Conclusiones

- Utilizar las herramientas *IC Validator* y *Custom Compiler*, permite que logremos hacer una integración sencilla de la herramienta *StarRC* al flujo existente, para la realización de extracción de componentes parásitos.
- Se logró hacer la extracción de dos diseños en silicio, un diseño correspondía a una compuerta *not* y el otro a un sumador completo.
- Se documento el uso correcto de la herramienta *StarRC* para que futuros estudiantes interesados en esta área de investigación puedan utilizarla.
- La descripciones incompletas de las celdas de las librerías *(black boxes)* hacen que verificar el funcionamiento del diseño en silicio mediante simulaciones sea una tarea difícil o imposible.

CAPÍTULO 12

Recomendaciones

1. Tener todas las herramientas de *Synoysys* actualizadas, así se evitan conflictos de versiones.
2. Utilizar la herramienta *Custom Compiler* para realizar extracciones. Ya que esta herramienta presenta una interfaz gráfica que ayuda a que el proceso de configuración sea más sencillo.
3. Leer los manuales proveidos por *Synopsys*, porque estos tienen información importante acerca de los flujos de extracción de parásitos.

CAPÍTULO 13

Bibliografía

[1] S. H. Rubio Vásquez, "Definición del Flujo de Diseño para Fabricación de un Chip con Tecnologia VLSI CMOS", Tesis de licenciatura Ingeniería Electrónica, Universidad del Valle de Guatemala, 2019.

[2] J. A. 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", Tesis de licenciatura Ingeniería Electrónica, Universidad del Valle de Guatemala, 2014.

[3] Solvnet.com, en *StarRC User Guide and Command Reference*, Synopsys, 2019.

[4] N. Weste y D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. Pearson Education Inc., 2011.

[5] R. C. Jaeger y T. N. Blalock, *Microelectronic circuit design*. McGraw-Hill, 2016.

[6] A. Kahng, J. Lienig, I. Markov y J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer Netherlands, 2011, ISBN: 9789048195916. dirección: https://books.google.com.gt/books?id=DWUGHyFVpboC.

[7] W. Yu y X. Wang, *Advanced Field-Solver Techniques for RC Extraction of Integrated Circuits*. Springer Berlin, 2016.

[8] L. A. Nájera Vásquez, "Implementación de circuitos sintetizados a nivel netlist a partir de un diseño en lenguaje descriptivo de hardware como primer paso en el flujo de diseño de un circuito integrado", Tesis de licenciatura Ingeniería Electrónica, Universidad del Valle de Guatemala, 2019.

[9] Solvnet.com, en *Synopsys Corporate Backgrounder Spring 2018*, Synopsys, 2018.

[10] ——, en, *HSPICE® User Guide: Basic Simulation and Analysis*, Synopsys, 2020.

[11] J. R. Girón Rubio, "Etapa de verificación física de Diseño en Silicio vs.Esquemático(LVS) en el flujo de diseño para un chip a nanoescala", Tesis de licenciatura Ingeniería Electrónica, Universidad del Valle de Guatemala, 20.

[12] Synopsys, *IC Validator LVS User Guide Q-2019.12*. SolvNet, 2020.

CAPÍTULO 14

Anexos

14.1. *Runsets* utilizados para realizar la extracción.

```
BLOCK: Full_Adder
* MILKYWAY_EXTRACT_VIEW: YES
*** Metal fill extraction
* METAL_FILL_POLYGON_HANDLING: FLOATING
* METAL_FILL_GDS_FILE:
* GDS_LAYER_MAP_FILE:
CASE_SENSITIVE: NO
HIERARCHICAL_SEPARATOR: /
*** RC Extraction options
* NETLIST_DEVICE_LOCATION_ORIENTATION : COMMENT
TCAD_GRD_FILE: /home/administrador/Escritorio/Star/cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
 MAPPING_FILE: /usr/synopsys/TSMC/180/CMOS/G/IO3.3V/pdk/T-018-CM-SP-018-W1_1_0A/synopsys_custom/Full_Adder.icv.lvs/STARRCXT.mapping
ICV_RUNSET_REPORT_FILE: /usr/synopsys/TSMC/180/CMOS/G/IO3.3V/pdk/T-018-CM-SP-018-W1_1_0A/synopsys_custom/Full_Adder.icv.lvs/STARRCXT.runset_rep
EXTRACTION: RC
REDUCTION: YES
DENSITY_BASED_THICKNESS: YES
*** For 90nm and below process
*EXTRACT_VIA_CAPS: YES
*** For 0.13um and above process
 EXTRACT_VIA_CAPS: NO
*** DataBase Processing
REMOVE_FLOATING_NETS: YES
REMOVE_DANGLING_NETS: YES
*REMOVE_FLOATING_PORTS: YES
POWER_NETS: VDD VSS
SKIP_CELLS: !*
TRANSLATE_RETAIN_BULK_LAYERS: YES
*** Netlist option
NETLIST_FORMAT: NETNAME
NETLIST_PASSIVE_PARAMS: YES
 NETLIST_FILE: adderstract.sp
* NETLIST_COMPRESS_COMMAND: gzip
*** Coupling Caps options
COUPLE_TO_GROUND: NO
COUPLING_REPORT_FILE: cc.rep
* COUPLING_ABS_THRESHOLD: 3e-15
* COUPLING_REL_THRESHOLD: 0.03
* COUPLING_REPORT_NUMBER: 1000
*** Subcket Pin's order for Simulation
* SPICE_SUBCKT_FILE: <user_define_file>
* XREF options
XREF: YES
XREF_USE_LAYOUT_DEVICE_NAME: YES
CELL_TYPE: LAYOUT
NET_TYPE: LAYOUT
*** For shrink flow
* MAGNIFICATION_FACTOR : 0.9
* MAGNIFY_DEVICE_PARAMS : NO
```

Figura 36: Componente de TSMC solo con esquemático

```
**  ------------------------------------------------------------------------------  **
**                                                                                  **
**  (c) Copyright 2013 Synopsys, Inc.                                               **
**                                                                                  **
**  ------------------------------------------------------------------------------  **
**                                                                                  **
**  Data contained in this file is created for educational and training purposes    **
**  only and is not recommended for fabrication                                     **
**                                                                                  **
**********************************************************************************


*****************************************************/

BLOCK:Full_Adder
MILKYWAY_DATABASE: /usr/synopsys/icc/bin/full_adder_mw
TCAD_GRD_FILE: /home/administrador/Escritorio/Star/cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
**MAPPING_FILE: /home/administrador/Escritorio/Star/STARRCXT.mapping

COUPLE_TO_GROUND: NO
COUPLING_MULTIPLIER: 1
EXTRACTION: RC
MILKYWAY_EXTRACT_VIEW: NO
CASE_SENSITIVE: YES
NETLIST_FORMAT: SPF
NETLIST_NODE_SECTION: YES
NETLIST_CONNECT_SECTION: YES
NETLIST_SUBCKT: YES
NETLIST_PASSIVE_PARAMS: YES
NETLIST_TAIL_COMMENTS: YES
NETLIST_DELIMITER: :

REDUCTION: NO
*XREF: YES
EXTRACT_VIA_CAPS: NO
IGNORE_CAPACITANCE: ALL
KEEP_VIA_NODES: NO
MAGNIFICATION_FACTOR: 1.0
MAGNIFY_DEVICE_PARAMS: YES
METAL_FILL_POLYGON_HANDLING: IGNORE
*MODE: 200
MOS_GATE_DELTA_RESISTANCE: NO
REMOVE_DANGLING_NETS: NO
REMOVE_FLOATING_NETS: NO
TRANSLATE_RETAIN_BULK_LAYERS: YES
SKIP_CELLS: !*
HIERARCHICAL_SEPARATOR: |
OBSERVATION_POINTS: *
EXTRA_GEOMETRY_INFO: NODE


NETLIST_FILE: full_adder_stracted.spf
```

Figura 37: Componente de TSMC solo con esquemático

CAPÍTULO 15

Glosario

Custom Compiler: Esta es una herramienta ofrece Synopsys para el diseño de circuitos integrados digitales, analógicos y mixtos.. 19, 24, 34, 37

DRC: Design Rule Check (DRC) término utilizado para describir la verificación de violaciones por reglas de diseño que puden ocurrir durante la síntesis física desde un circuito lógico hacia un *layout* fabricable. 19

HSPICE: Es un simulador de circuitos a nivel de dispositivo. HSPICE toma un archivo spice como entrada y produce una salida que describe la simulación solicitada del circuito.. 43

Layout Parasitic Extraction (LPE): Es el cálculo de los efectos parásitos tanto en los dispositivos diseñados como en las interconexiones de cableado requeridas de un diseño en silicio.. 19, 34, 37

Layout vs Schematic (LVS): Procedimiento a través del cual se compara el layout de un diseño contra el netlist para determinar si existe equivalencia entre ambos.. 13, 19, 27, 28, 32, 33, 35, 36, 43, 44

Milkyway: Se refiere a la librería que guarda el diseño en silicio del circuito obtenido mediante la síntesis física.. 14, 18, 21, 22

Runset: Este término se utiliza para describir un conjunto de instrucciones que son procesadas para realizar un extracción de parásitos.. 18, 27, 28, 32, 33, 35–37

TSMC: La empresa Taiwan *Semiconductor Manufacturing Company Limited (TSMC)* es una empresa fabricante de semiconductores de Taiwán. TSMC es el proveedor de algunos de los *runsets* y librerías que se utilizaron en el presente trabajo.. 19, 20, 25, 29, 34, 44