Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: uso avanzado de StarRC para la generación de un archivo Hspice con componentes parásitos para su correcta simulación

Carlos Andrés Letona Torres

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: uso avanzado de StarRC para la generación de un archivo Hspice con componentes parásitos para su correcta simulación

Trabajo de graduación presentado por Carlos Andrés Letona Torres
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: uso avanzado de StarRC para la generación de un archivo Hspice con componentes parásitos para su correcta simulación

Trabajo de graduación presentado por Carlos Andrés Letona Torres
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

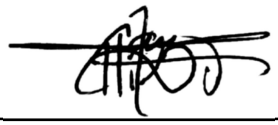
2023

Vo.Bo.:

(f) _____

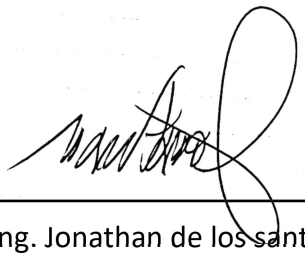
MSc. Carlos Esquit

Tribunal Examinador:



(f) _____

MSc. Carlos Esquit



(f) _____

Ing. Jonathan de los santos

(f) _____

Ing. Luis Nájera

Fecha de aprobación: Guatemala, 5 de enero de 2023.

Prefacio

La realización de este trabajo no hubiera sido posible sin el apoyo incondicional de varias personas. En primer lugar, agradezco a mi familia por el apoyo brindado, a mis padres por el esfuerzo que hicieron por darme una educación de calidad, a Mariana Rivas por su apoyo y motivación. Agradezco al MSc. Carlos Esquit por haber sido mi asesor y habernos transmitido parte de su conocimiento sobre nanoelectrónica en la Universidad del Valle de Guatemala. Le doy las gracias al Ing. Jonathan de los Santos por todo su apoyo con las herramientas de Synopsys. Finalmente, quiero agradecer al grupo de trabajo del proyecto del nanochip por haber brindado suficiente documentación para poder involucrarnos en el proyecto de forma sencilla, y a todos mis compañeros de la promoción de ingeniería electrónica.

Índice

Lista de figuras

Lista de cuadros

Resumen

En esta investigación se busca lograr un uso avanzado en StarRC para poder identificar comandos que sean de mucha utilidad para el desarrollo de un **CI** ya que esta herramienta de Synopsys contiene demasiadas opciones útiles por descubrir. A pesar de que la documentación de esta herramienta es bastante extensa, se busca poder identificar procesos, opciones, archivos de entrada y salida que sean de mucha importancia para poder concluir con la creación del primer nanoChip en Guatemala.

De esta manera, se busca poder generar un CMD el cual nos permita obtener un archivo con parásitos haciendo uso de la herramienta de Synopsys conocida como StarRC. Para realizar lo anteriormente mencionado se cuenta con la documentación de promociones pasadas, las cuales serán de mucha utilidad para poder replicar los resultados, entender y comprender el proceso para saber que es lo que se debe utilizar. Además, se documentarán opciones que puedan ser de utilidad para realizar distintos análisis con la herramienta y la identificación de comandos útiles para facilitar el uso de esta.

Por otro lado, se identificarán las compuertas utilizadas en el diseño del nanoChip y se crearán decks para cada compuerta que cumpla con el comportamiento lógico de cada una para poder simular el archivo de salida de la extracción de parásitos. Además, se propondrán diferentes formas de realizar la extracción y distintos flujos para generar las bases de datos.

Abstract

This research seeks to achieve an advanced use in StarRC in order to identify commands that are very useful for the development of a **CI** since this Synopsys tool contains too many useful options to discover. Although the documentation of this tool is quite extensive, it seeks to be able to identify processes, options, input and output files that are very important to be able to conclude with the creation of the first nanoChip in Guatemala.

In this way, it seeks to be able to generate a CMD which allows us to obtain a file with parasites using the Synopsys tool known as StarRC. To carry out the aforementioned, there is documentation of past promotions, which will be very useful to be able to replicate the results, understand and comprehend the process to know what should be used. In addition, options that may be useful to perform different analyzes with the tool and the identification of useful commands to facilitate its use will be documented.

On the other hand, the gates used in the design of the nanoChip will be identified and decks will be created for each gate that keep the logical behavior of each one in order to simulate the output file of the parasite extraction. In addition, different ways of performing the extraction and different flows to generate the databases will be proposed.

CAPÍTULO 1

Introducción

En la actualidad, los productos tecnológicos tienen dentro de ellos algún chip el cual debe realizar alguna función en específico. Esto se logró con la revolución tecnológica que se produjo al momento de la creación del primer transistor en 1947. Con el paso del tiempo, los circuitos han ido evolucionando de forma muy rápida, creándose así circuitos cada vez más pequeños y complejos. De esta manera, para poder crear circuitos pequeños, se deben crear transistores de menor tamaño.

Para lograr lo anteriormente mencionado, se utiliza un flujo de diseño específico. Para poder realizar este flujo se necesitan herramientas específicas las cuales han sido desarrolladas por la empresa Synopsys y a las cuales la Universidad del Valle de Guatemala nos da acceso. En los siguientes capítulos se encontrará información valiosa para poder hacer uso avanzado de la herramienta designada para realizar la extracción de parásitos conocida como StarRC. Esta extracción de parásitos se encuentra casi al final del flujo de diseño, ya que nos permite poder realizar diferentes tipos de análisis para circuitos los cuales ya han sido sintetizados físicamente.

CAPÍTULO 2

Antecedentes

Este proyecto de graduación es posible debido a los avances de compañeros en años pasados quienes han apoyado con el diseño y apoyo en diferentes etapas en la creación del primer nanoChip llamado "El Gran Jaguar" creado por una universidad centroamericana. Inicialmente, se replicaron los proyectos de graduación de años anteriores para poder entender el proceso de diseño del nanoChip. Afortunadamente, la documentación utilizada fue de mucha ayuda para entender cada proceso sin que quedaran dudas en las etapas correspondientes en la creación del nanochip. Por lo tanto, se cambió el verilog inicial del flujo para poder agregar los nombres de el nuevo grupo de estudiantes, y se replicaron todos los pasos hasta la síntesis física y sus respectivas verificaciones.

2.1 Instalación de aplicaciones de Synopsys

Inicialmente, es importante poder realizar una correcta instalación de las aplicaciones de diseño de la empresa Synopsys. Esta instalación se realizó siguiendo los pasos que podemos encontrar en [1].

2.2 Creación del archivo verilog

Luego, en varios de los proyectos anteriores se realizó un código en python para poder generar un archivo verilog para poder describir el comportamiento del nanoChip que se desea realizar. En este código se colocaron los dos textos que se deseaban que el chip mostrara en sus salidas y se realizó una conversión de letra por letra a un valor hexadecimal y se coloca ese valor de forma binaria en la salida del chip, obteniendo así una salida de un bus de 8 bits. Esta conversión se realizó colocando en una entrada una señal de reloj la cual con cada flanco positivo se aumenta el valor de un contador el cual es el encargado de recorrer el texto. Por último, este python genera el archivo verilog el cual se utilizará en la síntesis lógica.

2.3 Síntesis lógica

En esta etapa se utilizó el verilog de la etapa anterior para ejecutar 2 diferentes procesos. Estos procesos se dividieron en dos partes debido a que se presentaron problemas al intentar sintetizar lógicamente el nanoChip ya que había que agregar los módulos de entradas y salidas manualmente. Por lo tanto, los procesos son:

Primer proceso: En este proceso se sintetizó lógicamente el comportamiento del nanoChip con una herramienta de Synopsys llamada "Design Vision". Esta herramienta busca el poder replicar la lógica descrita por el archivo verilog con compuertas, multiplexores y flip-flops.

Segundo proceso: En el primer proceso obtuvimos un nuevamente un verilog de salida, en este verilog encontramos nuevamente el mismo comportamiento pero con compuertas, como podemos visualizar en la Figura 1. Sin embargo, como podemos observar en la Figura 2, este verilog le hacen faltas las compuertas de entrada y salidas físicas. Por lo que manualmente se agregaron e interconectaron estos módulos y se volvió a realizar la síntesis con la herramienta anteriormente mencionada. El resultado de este segundo proceso lo podemos ver en la Figura 8. Este último proceso nos brindará un nuevo verilog el cual está listo para utilizarse en la síntesis física.

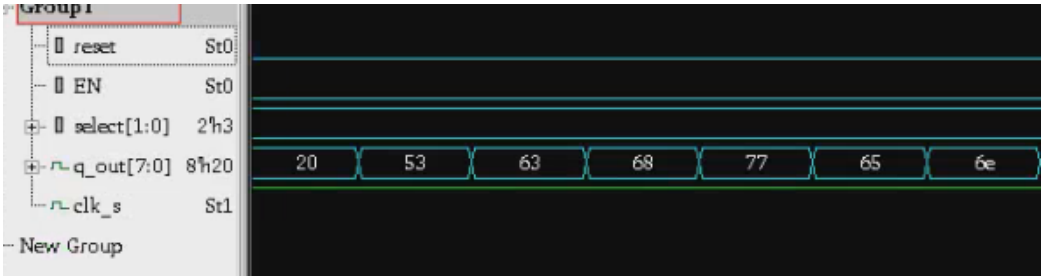


Figura 1: Comportamiento de salida luego de simular señales de entrada.

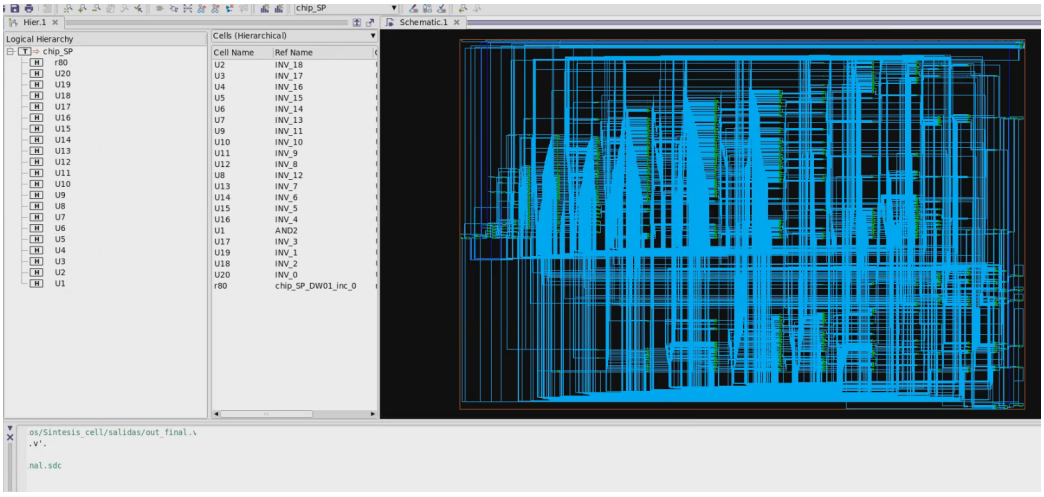


Figura 2: Núcleo lógico del nanoChip.

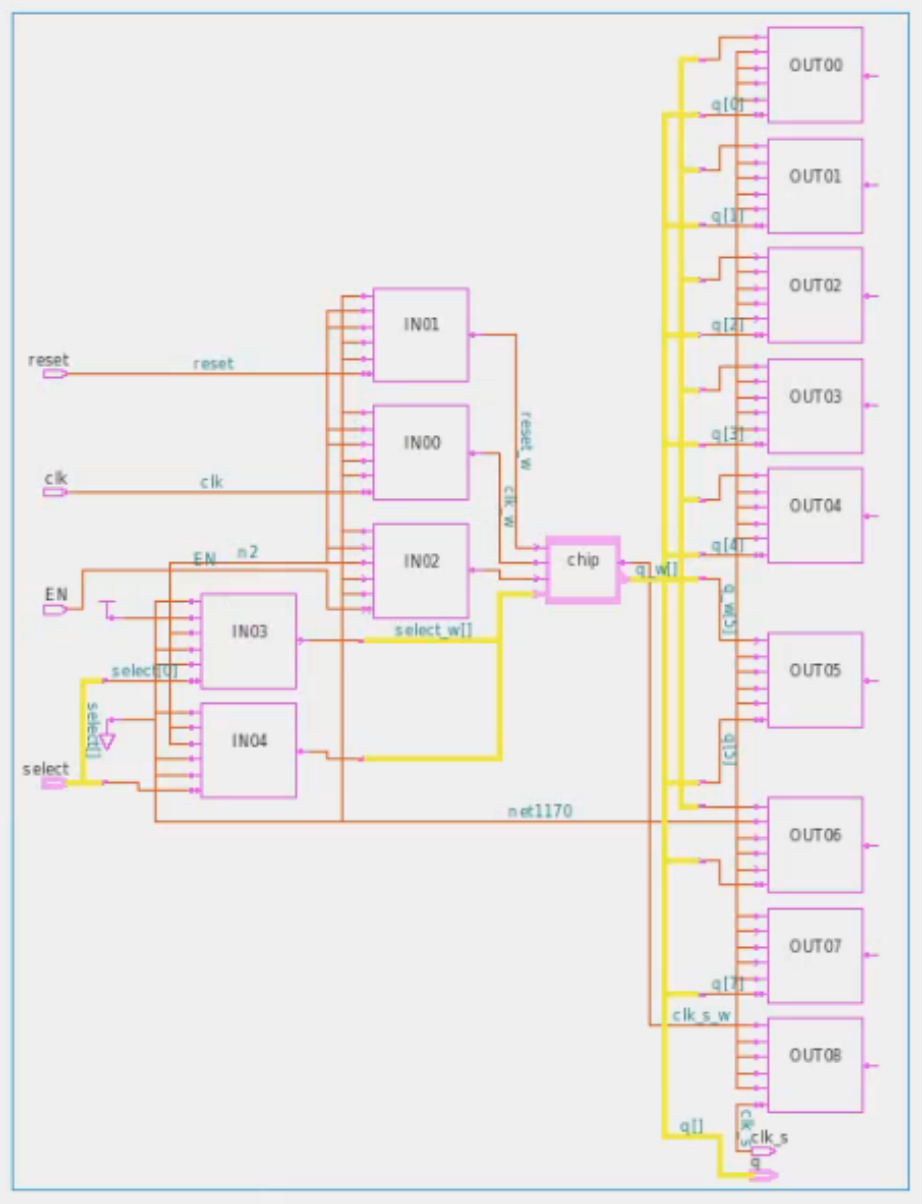


Figura 3: Núcleo lógico del nanoChip con pines de entrada y salida.

2.4 Síntesis física

En esta etapa se realizaron diferentes procesos para poder representar el verilog de salida de la síntesis lógica de forma física. Inicialmente, se crearon librerías con los archivos brindados por TSMC para seguir sus respectivas reglas de diseño y poder hacer uso de sus "Black boxes". Seguido, se crean celdas conocidas como "corners", las cuales no tienen ninguna función lógica y van a ser ubicadas en las esquinas del nanoChip. También, se crean las celdas de "Vdd" y "Vss" las cuales se encargarán de alimentar nuestro chip.

También es importante mencionar que se tuvo que realizar una ubicación estratégica de los pads de entradas y salidas alrededor de un "floor plan". Así mismo, se creó un anillo conocido como "power ground" el cual consiste en un anillo estratégico seguido de un patrón conocido como "mesh" para poder alimentar todas las celdas utilizadas de manera adecuada sin generar corto circuitos o inconsistencias como podemos observar en la Figura 4. Por último, se utilizó un comando para poder hacer un ruteo entre los componentes del chip.

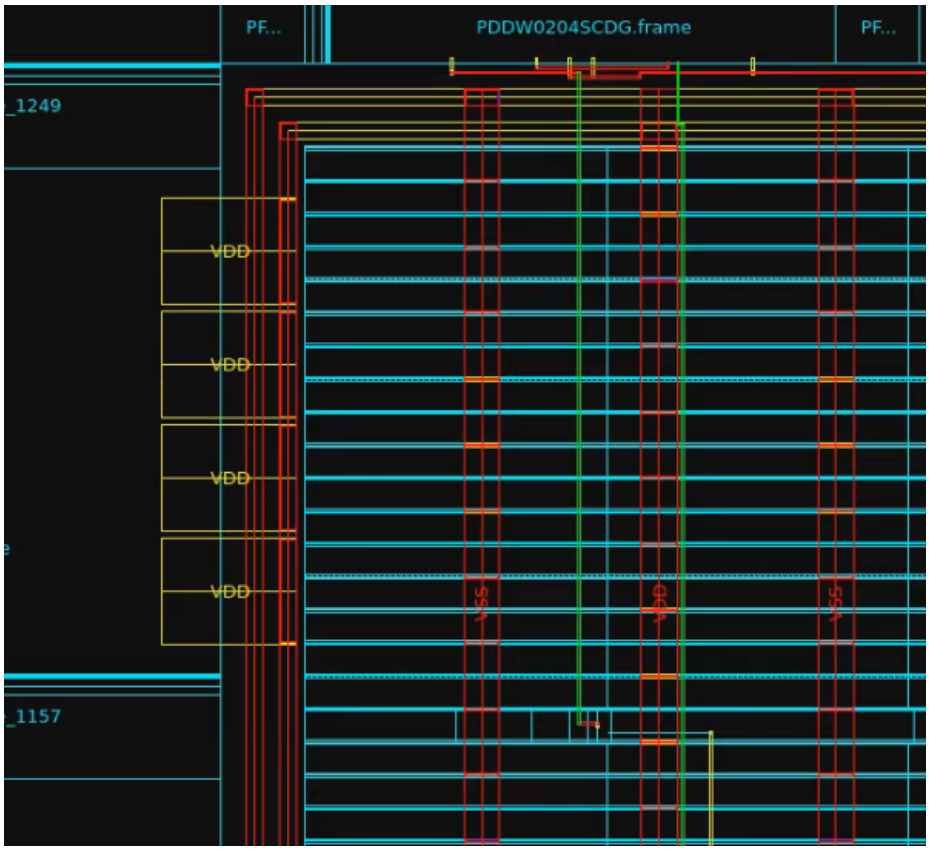


Figura 4: Núcleo lógico del nanoChip con pines de entrada y salida.

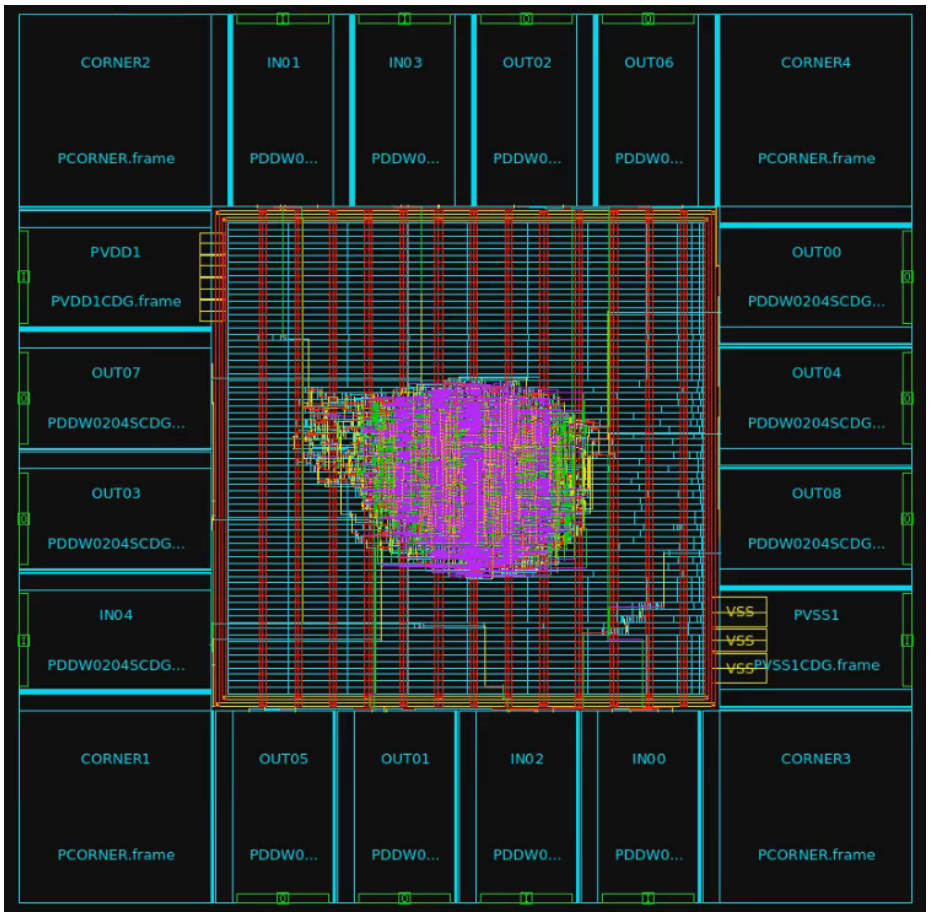


Figura 5: NanoChip sintetizado físicamente.

Finalmente, se realizaron diferentes pruebas para poder verificar que todos los procesos fueron realizados con éxito y que el diseño del nanoChip es completamente robusto para poder ejecutar su proceso de fabricación. Entre estas pruebas realizadas podemos encontrar:

- **Design Rule Check (DRC):** En esta se verifica que todos los requerimientos de fabricación. Esta verificación se ve limitada por la tecnología de las herramientas de fabricación de la fábrica, estas limitaciones lo que nos dictan, son las distancias que debemos tener entre metales, difusiones, polisilicios. Además de vernos en la necesidad de poner esquinas redondeadas, y ser suficientemente robusto en el diseño por posibles perdidas en la resolución en la manufactura. Precisamente, en esta etapa es donde encontramos los 6 errores que actualmente están deteniendo el proceso de fabricación del chip.
- **Layout vs Schematic (LVS):** Esta verificación lo que sencillamente hace, es el comparar que los *netlists* obtenidos en la síntesis física y en la síntesis lógica sean comparados para evitar tener inconsistencias que arruinen nuestro diseño.
- **Antenna Rule Check:** Con esta fase lo que deseamos es poder evitar efectos antena en nuestro circuito. Este efecto se da cuando los conductores acumulan cargas, y esto

es un problema debido a que puede llegar a dañar los transistores y dañar alguna compuerta.

- **Electrical Rule Check:** Con esta verificación podemos verificar la calidad de la conexión a tierra, a su respectiva alimentación, y los tiempos de transmisión de señales.
- **Parasitic Extraction (LPE):** Esta verificación utiliza los parámetros eléctricos (capacitancias y resistencias) para verificar las características eléctricas del circuito.En esta verificación se genera un archivo **Hspice** con todas estas capacitancias y resistencias parásitas presentes.

```
                    LAYOUT ERRORS RESULTS: ERRORS

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

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

Library name:    EL_GRAN_JAGUAR.ndm
Structure name:  chip_IO
Generated by:    IC Validator RHEL64 S-2021.06-SP3-2.7200131 2022/01/06
Runset name:     /home/nanoelectronica/Documents/Gran_Jaguar_DENSITY/Sintesis_f
User name:       nanoelectronica
Time started:    2022/05/06 05:11:29PM
Time ended:      2022/05/06 05:15:01PM

Called as: icv -icc2 -f NDM -i EL_GRAN_JAGUAR.ndm -p /home/nanoelectronica/Docu
slnFile.txt -icc_density_blockage -icc2_error_categories -I /home/nanoelectroni
nanoelectronica/Documents/Gran_Jaguar_DENSITY/Sintesis_fisica/DRC/signoff_check
Gran_Jaguar_DENSITY/Sintesis_fisica/requisitos_tsmc/ICVLM18_LM16_LM152_6M.215a_
CLF: -sln "16 18 28 31 33 38 17 27 29 32 39 "

                        ERROR SUMMARY

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

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

 M3.R.1 : Min M3 area coverage < 30%
   density ..................................... 1 violation found.

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

 M5.R.1 : Min M5 area coverage < 30%
   density ..................................... 1 violation found.

 M6.R.1 : Min M6 area coverage < 30%
   density ..................................... 1 violation found.
```

Figura 6: Errores de densidad.

La prueba llamada “design rule check” la cual verifica que cumplimos con todas las reglas de diseño del fabricante. La segunda prueba a realizar se llama “electric rule check” la cual como su nombre indica, verifica las reglas de diseño eléctricas. De la misma forma, se realizó una prueba que llama “layout vs schematic” la cual verifica que no tengamos inconsistencias entre el layout de las compuertas a utilizar y de su esquemático correspondiente. Finalmente, la última prueba a ejecutar es la importante prueba de “antenna”. Afortunadamente todas las pruebas salieron exitosas, excepto por la prueba de DRC la cual se tratará más adelante.

CAPÍTULO 3

Justificación

El proceso de la creación de un chip es tan complicado que solo países bien desarrollados tienen fabricas dedicadas a la creación de estos. Afortunadamente, la Universidad del Valle de Guatemala, logró tener acceso a herramientas de fabricación de estos chips, lo cual es algo histórico en el área de desarrollo tecnológico del país. Debido que abre las puertas a que muchos estudiantes de esta casa de estudios aprendan y sepan utilizar todas las herramientas necesarias para la fabricación e investigación de chips nanométricos en Guatemala.

Los estudiantes de Ingeniería Electrónica poseen una característica única que nos diferencia de las demás universidades, esta característica es su conocimiento del área de la nanoelectrónica. El objetivo principal de este proyecto es el poder concluir con el proyecto del nanoChip con la resolución de los 6 errores de densidad, para poder así ser el primer país en centroamérica en crear un nanoChip. Al lograrlo, no solo se estaría creando el primer nanoChip, si no que también se abre una cantidad importante de oportunidades para la sociedad guatemalteca, ya que nos lograríamos posicionar como un país capaz de desarrollar proyectos ambiciosos, destacar poder aspirar a poder trabajar en estas importantes empresas en la tecnología moderna.

Por lo anteriormente mencionado, es importante poder garantizar que el nanoChip se comporta de la forma que se desea. Para poder garantizarlo, debemos ser capaces de encontrar una solución al momento de realizar la debida extracción de parásitos con StarRC para poder generar un archivo que contenga las entradas y salidas del nanoChip. Ya que este archivo es el que el grupo especializado en Hspice podrá simular para verificar que el comportamiento es el que se planteo inicialmente. Esta etapa prácticamente es una garantía de que todos las iteraciones anteriormente realizadas funcionan adecuadamente.

CAPÍTULO 4

Objetivos

4.1 Objetivo general

Aprender el uso avanzado de StarRC para poder generar un archivo Hspice a partir de la extracción de parásitos el cual contenga todos los componentes necesarios para poder simular un comportamiento transiente y así poder garantizar que el nanoChip cumple con la función deseada.

4.2 Objetivos específicos

- Análisis y distribución de compuertas lógicas utilizadas en el nanoChip.
- Crear decks que cumplan con el comportamiento lógico encontrado en la documentación sobre las compuertas utilizadas.
- Hacer uso avanzado de StarRC para identificar posibles comandos útiles al hacer la extracción de parásitos.
- Apoyar en la búsqueda de la solución para los 6 errores de densidad pendientes.
- Proponer nuevas opciones de configuración de StarRC que beneficien a la linea de investigación.
- Experimentar con diferentes formas de hacer la extracción de parásitos.

CAPÍTULO 5

Alcance

Este trabajo busca la definición y la correcta ejecución de una extracción de parásitos de un circuito específico para la respectiva generación de un archivo Hspice. Esta ejecución se limita al uso de la herramienta de Synopsys conocida como StarRC. De la misma manera, este trabajo muestra la ejecución de la debida extracción haciendo uso de archivos brindados por el textitfoundry TSMC y librerías educativas brindadas por Synopsys.

Por lo tanto, este trabajo se ve limitado a realizar la debida extracción haciendo uso de archivos resultantes de una verificación exitosa de LVS, de esta verificación haremos uso de un archivo mapping, de una base de datos Milkyway y de un report file. Finalmente, se mostrará como crear un CMD para ejecutar comandos útiles en la debida extracción, cabe mencionar que se estará mostrando los campos necesarios y opcionales que se pueden utilizar, sin embargo no se estará haciendo uso de algunos comandos avanzados.

CAPÍTULO 6

Marco teórico

6.1. Very Large Scale Integration

VLSI es el proceso completo para la fabricación de circuitos integrados. Estos circuitos están conformados por miles de millones de transistores, el hecho de tener tantos transistores en un lugar tan pequeño lleva una gran complejidad en su proceso de fabricación. Para esto, utilizamos un flujo de diseño para el cuál se utilizarán herramientas dedicadas a este proceso a las cuales la empresa llamada *Synopsys* le da acceso a la Universidad del Valle de Guatemala.

En la actualidad encontramos una diversidad de dispositivos los cuales cuentan con un **CI**. Este circuito integrado es el encargado de realizar varias tareas de computo y distribuirlo a otros dispositivos. Las empresas líderes en el diseño de **CI** haciendo uso del proceso de **VLSI** son Intel, AMD y Samsung.

6.2. Herramientas de Synopsys

- **Hspice:** Este es un simulador para circuitos analógicos el cual es capaz de realizar análisis transientes, de estado estacionario y de dominio de frecuencia. Hspice es la mejor herramienta para poder simular circuitos ya que esta nos ofrece modelos de componentes MOS los cuáles están certificados.
- **Custom WaveView:** Esta herramienta es un entorno gráfico que nos permite observar las salidas de circuitos simulados. Esto nos permite poder observar las salidas y entradas de nuestro **CI** para poder verificar el comportamiento.
- **Design Vision:** Con esta herramienta podemos realizar la síntesis lógica con la tecnología a utilizar y se generan archivos de salida los cuales son de mucha importancia para las siguientes etapas del flujo de diseño.

- **IC Compiler 2 (ICC 2):** Esta herramienta es la encargada de realizar los pasos de diseño que se pueden observar en la Figura 7 en la etapa de la síntesis física.
- **IC Validator:** Con este software logramos comprobar verificaciones de diseño de un *Layout* en específico.
- **Star RC:** Este software nos permite realizar extracción de parásitos. Este proceso nos brinda un modelo de el circuito junto con los efectos físicos de sus componentes.

6.3. Flujo de diseño

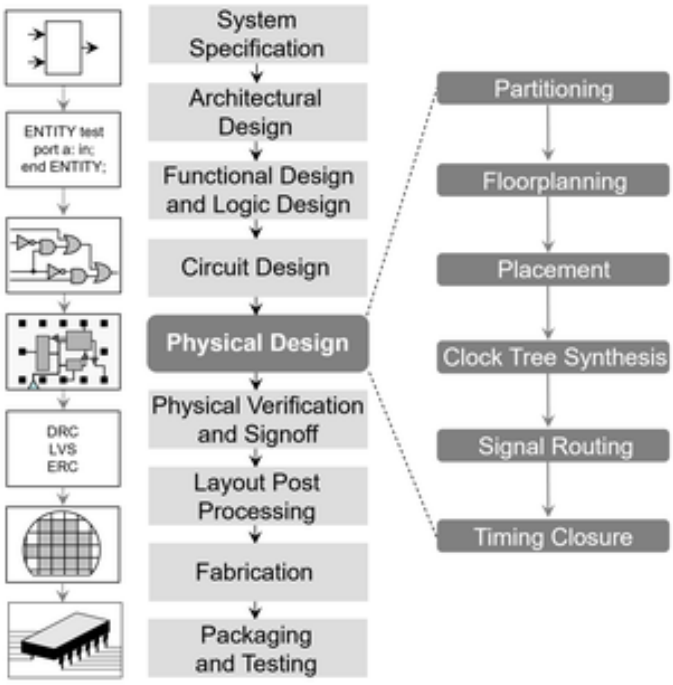


Figura 7: Flujo de diseño para el diseño de un Circuito Integrado.

Inicialmente, para poder hacer realidad la creación de un circuito integrado, debemos asegurarnos que el **CI** sea completamente fabricable y funcional. Para poder garantizar estos dos puntos, debemos seguir un flujo de diseño cómo se puede visualizar en la Figura 7 y asegurar que en cada etapa se cumplen con todos los requisitos de las etapas correspondientes.

6.4. Diseño de la capa física

Partiendo de un *netlist* creado en la síntesis lógica, se debe describir como se va a colocar en un espacio físico un material semiconductor como lo es el silicio. Este bloque de silicio tiene partes con difusiones distintas, pines, polisilicios, metales e interconexiones. En esta etapa como observamos en la Figura 7, precisamente en la etapa de diseño físico, encontramos varias sub-etapas las cuales son planificadas en esta etapa.

Por lo tanto, como podemos observar una de estas sub-etapas es el ruteo de nuestros componentes. Este ruteo se hace en varios metales, en nuestro nuestro **CI** utiliza desde metal 1 hasta metal 6, es por esto que al realizar la prueba de DRC obtenemos 6 errores de densidad como podemos ver en 6 ya que nuestra densidad por tipo de metal en el bloque designado es muy bajo para cumplir con las reglas de diseño provistas por Synopsys.

6.5. Extracción de parásitos

En la herramienta StarRC de Synopsys se pueden extraer parásitos como resistencias, capacitores e inductores que representan diseños de diseño de circuitos integrados. Una vez que se termina el diseño del layout, se debe probar la sincronización del circuito. Un análisis de tiempo preciso requiere que todas las resistencias y capacitancias parásitas se tomen en cuenta, ya que así podemos tener una idea precisa de los tiempos que el **CI** va a tener. La herramienta StarRC usa el diseño del chip junto con la descripción del proceso (en nuestro caso es un archivo verilog creado en la síntesis lógica) para extraer millones de parásitos. Finalmente, al realizar la extracción de parásitos, se puede generar un archivo Hspice, el cuál nos servirá para poder simular nuestro **CI** luego de haber realizado todas las etapas y así poder garantizar su adecuado comportamiento.

StarRC analiza capacitancias y resistencias parásitas del diseño, estos parásitos se obtienen de un archivo *nxtgrd* el cual posee una base de datos de caracterización brindada por el fabricante según el proceso de manufactura. Como podemos observar en la Figura 8, el archivo anteriormente mencionado es una de las entradas a nuestra herramienta. Además, es importante notar como la herramienta a utilizar acepta diseños Milkyway, LEF/DEF, e incluso bibliotecas de diseño NDM las cuales son creadas en ICC2. De igual manera, StarRC nos pide un archivo *mapping* el cual obtenemos de la verificación de LVS. Finalmente, nos pide un último archivo conocido como CMD el cual contiene todos los comandos que vamos a necesitar para realizar la extracción de parásitos y restricciones a tomar a cuenta. Una vez realizado el proceso, StarRC nos brinda un archivo binario (GPD) con los datos de todos los parásitos extraídos. Sin embargo, a la herramienta se le puede pedir que genere un archivo SPF el cual es el que nosotros vamos a necesitar para poder mandarlo al equipo encargado de Hspice para su debida simulación.

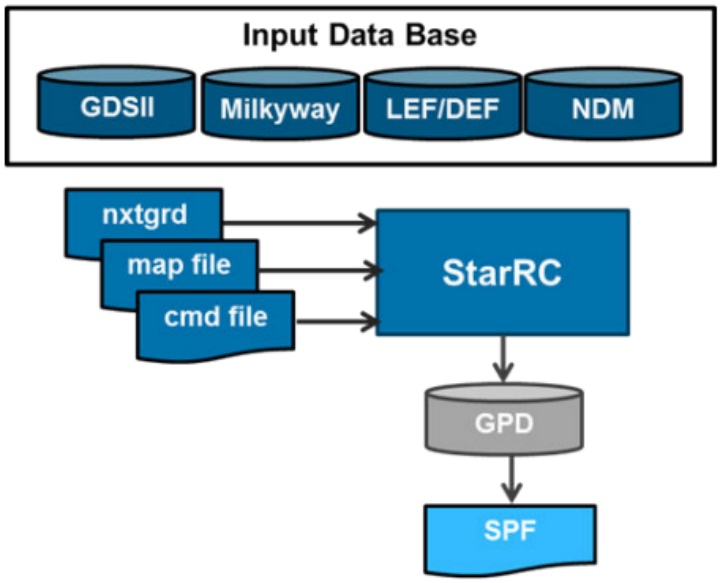


Figura 8: Proceso para el uso de StarRC. [17]

La herramienta de Synopsys contiene tiene varias formas de lograr diferentes objetivos. Por ejemplo, StarRC tiene la opción de realizar una extracción a nivel de flujo de compuertas lógicas, este tipo de extracción es comúnmente utilizada para poder hacer un análisis de *timing* en un chip. Por otro lado, tenemos el análisis a nivel transistor, el cual está enfocado en el análisis con mayor detalle ya que este examina parásitos en cada nodo y depende del archivo *mapping* obtenido de la etapa de LVS. Cabe mencionar que también se pueden realizar análisis de inductancia en el nodo del reloj y diferentes análisis de campo.

StarRC es precisa y eficiente en todos los análisis que se pueden analizar, desde un análisis tan complejo como puede ser un chip con miles de millones de transistores, hasta el análisis de un par de nodos críticos. Algunos de los principales análisis que se pueden realizar con esta herramienta son los siguientes:

- Extracción completa de un chip con verificaciones de ruido, *timing* y electromigración.
- La opción de poder usar archivos de caracterización brindados por las fábricas mas grandes de microcontroladores.
- Extracción a nivel de compuerta y a nivel transistor.
- Analizar diseños en etapa temprana los cuales pueden poseer violaciones en las reglas de diseño.

Por otro lado, es importante mencionar que StarRC cuenta con unas excelentes *man pages*, las cuales nos serán de mucha utilidad durante todo el proceso, o incluso para poder identificar errores y soluciones recomendadas. Para poder acceder a estas *man pages* debemos identificar el error que nos muestra nuestra consola al ejecutar el comando de la extracción, y luego hacer uso del comando 1.

```
starrc_man SX-0000
```

Cuadro 1: Comando para acceder a las *man pages* de StarRC.

Finalmente, para poder realizar la extracción de parásitos debemos utilizar el comando 2 en la consola del sistema operativo. Es importante resaltar que este comando puede incluir opciones útiles como podemos observar en el Cuadro 3. Si se desea más información sobre todos los comandos opcionales, se pueden encontrar en [17].

```
StarXtract option1 option2 ... cmd_file1 [cmd_file2] ...
```

Cuadro 2: Comando para correr la extracción de parásitos. [17]

Opción	**Descripción**
cmd_file1	StarRC *command file*.
cmd_file2	*command file* opcional para la extracción.
-cleanN	Regenera el netlist de salida para extracción a nivel transistor.
-compare_parasitics	Compara dos netlists.
-gdscheck	Analiza errores en un archivo GDS
-convert_gpd_to_spef	Crea un archivo SPEF de un GPD
-convert_gpd_to_spf	Crea un archivo SPF de un GPD
-convert_gpd_to_oa	Crea un archivo OA de un GPD
-set_gpd_config	Aplica una configuración a un archivo GPD
-reset_gpd	Regresa el archivo GPD a su estado original

Cuadro 3: Tabla de opciones. [17]

6.5.1. Flujos para realizar el diseño de las bases de datos

En la herramienta de StarRC se calculan los efectos parásitos que se obtendrán de nuestro diseño físico del **CI**. Normalmente, utilizamos los archivos de salida de la prueba física de LVS .

Tipo	Descripción
Librería	NDM creadas en ICC2.
Librería	Formato Milkyway creadas con ICC.
Librería	Formato LEF/DEF.
Archivo	Salida de LVS usando ICV
Archivo	Salida de LVS usando Hercules
Archivo	Salida de LVS usando Mentor Graphics Calibre

Cuadro 4: Tabla de opciones. [17]

6.5.2. IC Validator LVS Tool Flow

Como se puede observar en el flujo de la Figura 9, para crear un netlist con parásitos haciendo uso de IC validator, debemos proveer una base de datos físicos, un netlist de un esquemático y un runset *script* para generar la base de datos de ICV, de igual forma, los archivos como la base de datos de ICV y el runset report file pueden ser documentos de entrada en la herramienta de StarRC para realizar la extracción.

En el flujo de ICV, inicialmente debemos de seleccionar los datos de entrada y especificar cual será el *pex_runset_report_file* del runset *script*. Todos los resultados y la información de la base de datos es guardado en el *pex_runset_report_file*. Finalmente, al realizar la extracción de parásitos se debe colocar el comando 5 en el CMD que se ejecutará.

```
ICV_RUNSET_REPORT_FILE : pex_runset_report
```

Cuadro 5: Comando para agregar el *pex_runset_report_file* en StarRC. [17]

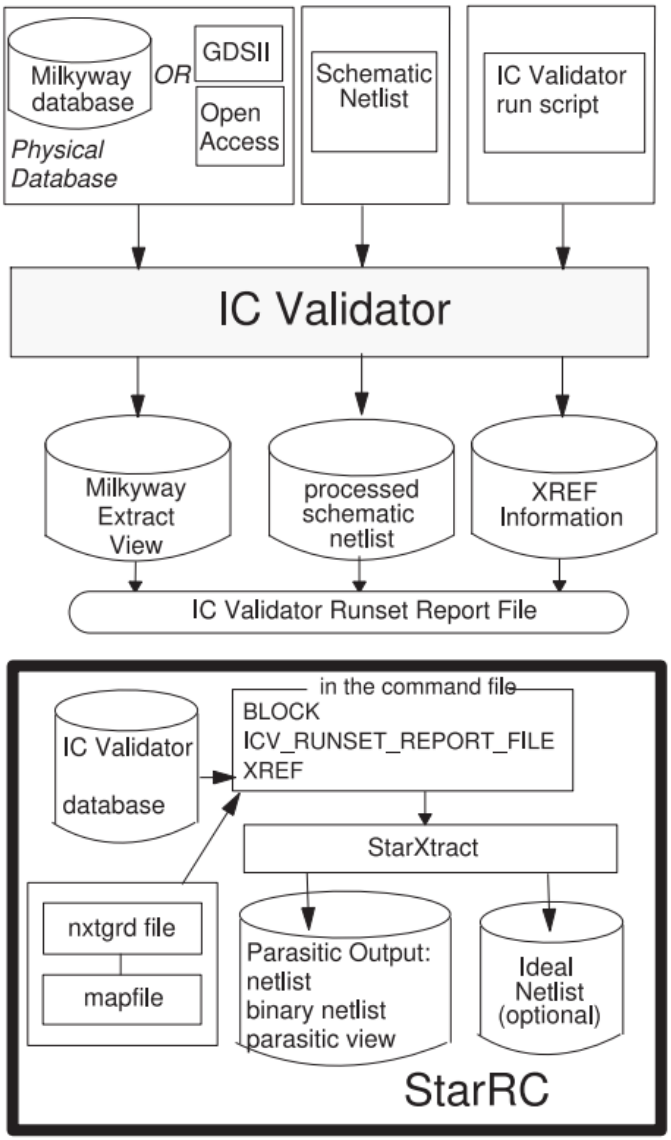


Figura 9: IC Validator flow. [17]

6.5.3. IC Compiler II Flow

La herramienta de ICC2 genera una librería de diseño en formato NDM, esta librería puede ser leída directamente por la herramienta de StarRC como se puede ver en la Figura 10. Este flujo a diferencia del flujo anterior, no necesita de una verificación *clean* de LVS en el layout de nuestro circuito para poder realizar satisfactoriamente la extracción. Al hacer uso de este flujo, se puede generar un *report_file* agregando el comando 6 el cual define el nombre que se le dará al *report_file*. En este archivo se puede encontrar información de los *cells* de nuestro circuito y del *top cell*.

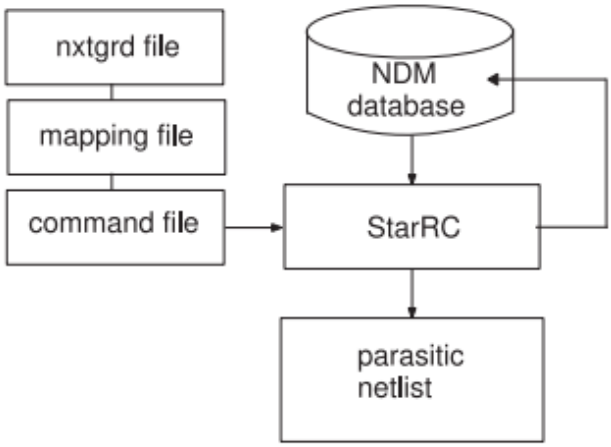


Figura 10: ICC II Flow.

```
NDM_CELL_REPORT_FILE: ndm_report
```

Cuadro 6: Comando para crear un *report_file* del NDM. [17]

De igual manera, en el CMD a utilizar, se debe agregar el comando 7, el cuál es obligatorio para poder realizar la extracción haciendo uso de este flujo. El nombre a colocar en este comando es el mismo que se usó en el comando *open_ lib* en ICC2. De igual manera, en el comando *BLOCK* del CMD se debe colocar el nombre del *top-level block* el cual es el mismo que se utilizó en el comando *open_ block* de la herramienta de ICC2. Cabe mencionar que el comando 7 soporta únicamente diseños a nivel de compuertas.

```
NDM_DATABASE: design_library
```

Cuadro 7: Comando para especificar el nombre de la librería de diseño generada por ICC2. [17]

Asimismo, encontramos el comando 8, con el cual hace referencia a una lista de celdas específicas para las cuales la herramienta de StarRC hace uso de una vista de diseño y no de una vista por *frame* durante la extracción. Es importante resaltar que este comando también puede aplicarse a celdas omitidas.

```
NDM_DESIGN_VIEW: list_of_cells
```

Cuadro 8: Comando para especificar la lista de las celdas para la vista por diseño. [17]

Finalmente, con el comando 9 se estarán listando celdas para las cuales se tendrá una vista por *layout* durante la extracción. Es importante mencionar que para cada celda omitida, debe existir su respectiva vista por *frame* o por diseño, ya que la herramienta de StarRC obtiene información de la conectividad en el siguiente orden:

- De la vista de diseño, si la vista existe y la celda especificada en el comando 8.
- De lo contrario, revisa si existe en la vista por *frame*.
- Del un archivo GDSII si la celda está especificada en el comando *gds_file*.
- En el archivo especificado en la vista por layout.

```
NDM_LAYOUT_VIEW: list_of_cells
```

Cuadro 9: Comando para especificar la lista de las celdas para la vista por *layout* [17]

6.5.4. IC Compiler (Milkyway) Database Flow

Como podemos observar en la Figura 11 se muestra el flujo para generar una base de datos usando ICC. Las bases de datos *Milkyway* se pueden leer directamente en StarRC. Además, la prueba de LVS no es necesaria que sea *clean* para poder realizar exitosamente la extracción. De igual manera que en el flujo anterior, tenemos comandos importantes que se deben agregar en el CMD. *Milkyway* guarda información del diseño en archivos llamados *views*.Por lo tanto, para poder leer *views* adicionales a la base de datos utilizamos el comando 10.

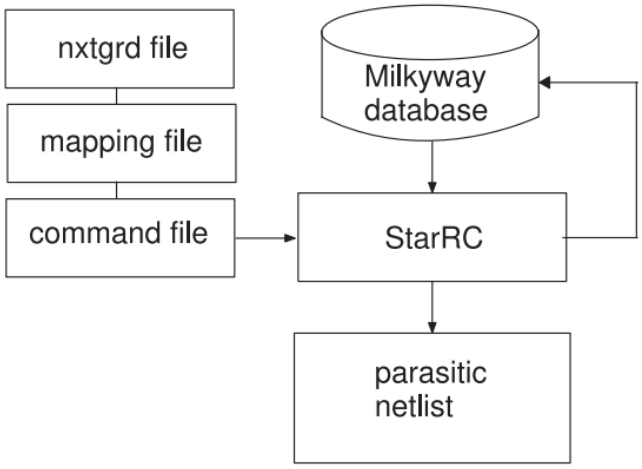


Figura 11: Milkyway Design Extraction Flow. [17]

```
MILKYWAY_ADDITIONAL_VIEWS: view_name
```

Cuadro 10: Comando para agregar *views* adicionales [17]

De igual manera, tenemos el comando para poder especificar una lista de *cells* para los cuales la herramienta de StarRC utilizará la vista por *layout* al momento de realizar la extracción. También es sumamente importante utilizar el comando 12, con este comando se especifica la ubicación de la carpeta de la base de datos que posee todas las *views* que se utilizan en nuestro diseño.

```
MILKYWAY_CELL_VIEW: list_of_cells
```

Cuadro 11: Comando para agregar *cell views* adicionales [17]

```
MILKYWAY_DATABASE: layout_library
```

Cuadro 12: Comando para especificar la librería de la base de datos *milkyway*. [17]

6.5.5. StarRC Command File

El *command file* es un archivo el cual se especifica al utilizar el comando *StarXtract*, en este archivo se incluyen comandos generales para la configuración inicial y comandos específicos que se desean utilizar al momento de realizar la extracción de parásitos. Cabe mencionar que el comando *StarXtract* acepta como entrada varios archivos CMD, pero la herramienta lee inicialmente el primer archivo colocado, luego el segundo, luego el tercero y así sucesivamente, reescribiendo instancias creadas inicialmente. Por otra parte, los comandos que incluyen listas u objetos se van acumulando.

Por lo tanto, es importante comprender como debe crearse un *command file* de forma correcta. Inicialmente, la sintaxis que se utiliza en el CMD es el que se puede apreciar en 13. De esta manera, existen también caracteres especiales para poder utilizar en estar archivo, algunos de los caracteres más importantes se pueden observar en el Cuadro 14. Es importante mencionar que se puede hacer uso de *strings*, *arrays*, listas y de operadores matemáticos como se detalla en [16].

```
command_name [command_argument] [-option [argument]]
```

Cuadro 13: Sintaxis para colocar comandos en el CMD [17]

Caracter	Descripción
...	Indica que un comando puede ser repetido varias veces.
[]	Indica un comando opcional.
\	Indica la continuación de una linea de comandos.
/	Indica el nivel de una estructura de directorios.
*	Da inicio a un comentario y como comodín de caracteres.

Cuadro 14: Tabla de caracteres especiales.

Como primer punto, para poder crear nuestro archivo con comandos, debemos de especificar de forma obligatoria ciertos comandos. Para esto tenemos que tener claro que tipo de base de datos estamos utilizando y el nombre de nuestro *top block*. Por ejemplo, en 15 podemos ver como se define el nombre del bloque principal usando el comando *BLOCK*, la ubicación de nuestra base de datos (en este caso en específico se está utilizando una base de datos *Milkyway* el cual viene del flujo de ICC) haciendo uso del comando *MILKYWAY_DATABASE:*, la ubicación del archivo *nxtgrd* usando el comando *TCAD_GRD_FILE:*, y finalmente se define la ubicación del archivo *mapping* y del *runset_report_file* haciendo uso de los comandos *MAPPING_FILE:* y *ICV_RUNSET_REPORT_FILE:* respectivamente.

De la misma manera, se deben definir los comandos para obtener nuestro archivo de salida. Esto lo logramos utilizando los comandos que se pueden ver en 16. Es importante tomar en cuenta que el primer comando *NETLIST_FORMAT:* permite varias opciones de entrada las cuales se encuentran detalladas en 18, de igual manera, con el comando *NETLIST_FILE:* se define la dirección y el nombre del archivo que se generará después de realizar la extracción.

Es importante mencionar, que existen varios comandos adicionales que se pueden utilizar, por ejemplo, el comando *COUPLING_REPORT_FILE:* y *PLACEMENT_INFO_FILE:* no son obligatorios, pero el primer comando genera un reporte de las capacitancias de acoplamiento por nodo y el segundo genera información de la ubicación de las celdas en nuestro diseño. Lo cual puede ser de mucha utilidad al querer tener mayor información del diseño.

```
BLOCK: chip_IO
MILKYWAY_DATABASE: ./MILKYWAY_XTR
TCAD_GRD_FILE: ./cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
MAPPING_FILE:./STARRCXT.mapping
ICV_RUNSET_REPORT_FILE: ./STARRCXT.runset_rep
```

Cuadro 15: Ejemplo de comandos importantes.

```
NETLIST_FORMAT: SPF | SPEF | NETNAME | OA | STAR
NETLIST_FILE: chip_IO.spf
COUPLING_REPORT_FILE: chip_IO.spf
PLACEMENT_INFO_FILE: YES
PLACEMENT_INFO_FILE_NAME: PLACEMENT_chipIO
```

Cuadro 16: Comandos para generar el archivo de salida.

Argumento	**Descripción**
SPF	Es el formato estándar y soporta únicamente *EXTRACTION: RC* con el comando *COUPLE_ TO_ GROUND: YES | NO*
SPEF	Es flexible y compacto, todas los nombres se encuentran mapeados internamente, reduciendo así el tamaño del *netlist*.
OA	Crea un archivo *OpenAccess* y solo se puede utilizar con una extracción a nivel transistor.
NETNAME	Exclusivo para la extracción a nivel transistor, genera nodos internos con nombres como *netname:1*, *netname:2*, etc.
STAR	Exclusivo para la extracción a nivel transistor, es compacto y utiliza nomenclatura de subnodos tipo *SPICE*.
PARAMETERIZED_SPICE	Es un *netlist SPICE* parametrizado que permite un análisis Monte Carlo.

Cuadro 17: Argumentos para el archivo de salida de la extracción. [17]

Comando opcional	Descripción
POWER_NETS:	Ayuda a ahorrar tiempo de ejecución y memoria. En este comando se específican los nodos de poder y se genera un archivo llamado *power_net_names* el cual posee información de los nodos identificados como *power_nets*
PARASITIC_EXPLORER_ENABLE_ANALYSIS:	Se debe escoger entre *YES* o *NO*. Con este comando podemos decidir si creamos un GPD el cual serviría para usar en el *Parasitic Explorer Tool.*
REMOVE_FLOATING_NETS:	Se define si queremos colocar los nodos que no tienen ninguna conexión a tierra.
REMOVE_DANGLING_NETS:	Definimos si deseamos aterrizar nodos identificados como flotantes.
COUPLE_TO_GROUND:	Este comando sirve para decidir si deseamos retener o aterrizar las capacitancias de acoplamiento.
EXTRACT_VIA_CAPS:	Realiza una extracción detallada vía capacitancias.
DENSITY_BASED_THICKNESS:	Permite el cálculo de la variación de densidad y espesor durante la extracción.
EXTRACTION:	Permite especificar el tipo y el alcance de la extracción.
REDUCTION:	Permite especificar el tipo de reducción a realizarse.
CASE_SENSITIVE:	Permite especificar si deseamos distinguir entre minúsculas y mayúsculas en los nombres de las celdas.
HIERARCHICAL_SEPARATOR:	Permite especificar el simbolo a utilizarse para separar delimitadores jerárquicos.
XREF:	Este comando determina qué conjunto de nombres debe tomar en cuenta para la generación del archivo de salida y cuales son los flujos de análisis y qué dispositivos y redes conservar si están presentes tanto los nombres de diseño como los nombres de esquemáticos con referencias cruzadas.
XREF_USE_LAYOUT_DEVICE_NAME:	Especificar si deben usarse los nombres de los layouts en dado caso el comando *XREF* se encontrara activo.
PLACEMENT_INFO_FILE:	Especifica si queremos crear un archivo que contenga información de la colocación de las celdas.
PLACEMENT_INFO_FILE_NAME:	Especificamos el nombre del archivo en dado caso el comando *PLACEMENT_INFO_FILE:* se encuentre activado.

Cuadro 18: Comandos opcionales útiles. [17]

6.5.6. Electromigration Analysis

La electromigración se refiere al efecto físico que perjudicial que se produce al manejar altas densidades de corriente en un circuito integrado. Este efecto toma mucha importancia conforme se van disminuyendo las tecnologías de transistores que se utilizan, ya que estos al ser más pequeños son más propensos a romperse o a generar falsas conexiones como podemos observar en 12. Por lo tanto, para poder generar un *netlist* haciendo uso de la herramienta de análisis de electromigración, se recomienda utilizar los comandos que se pueden ver en 19.

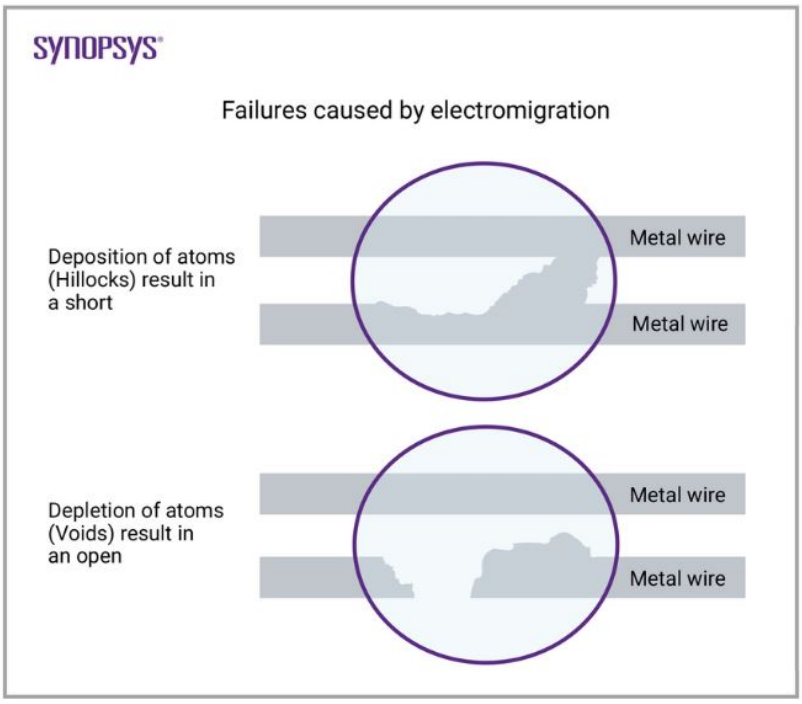


Figura 12: Milkyway Design Extraction Flow. [20]

```
REDUCTION: NO
EXTRA_GEOMETRY_INFO: NODE RES
KEEP_VIA_NODES YES
SHORT_PINS: NO
NETLIST_TAIL_COMMENTS: YES
NETLIST_NODE_SECTION: YES
NETLIST_CONNECT_SECTION: YES
```

Cuadro 19: Comandos para realizar una extracción con análisis de electromigración.

6.5.7. Custom Compiler Extraction Flow

Luego de haber visto como realizar la extracción de parásitos con diferentes flujos y la correcta creación del CMD, es importante mencionar que existe el flujo de *Custom Compiler* el cuál posee una interfaz gráfica para poder realizar la extracción. Como se puede apreciar en la Figura 13, este flujo sigue utilizando todos los archivos mencionados anteriormente. Por lo tanto, para demostrar el funcionamiento de este flujo se realizó un *ring oscilator* en el capítulo 8 y se realizaron todas las configuraciones necesarias como se puede observar en las figuras 44, 45, 46.

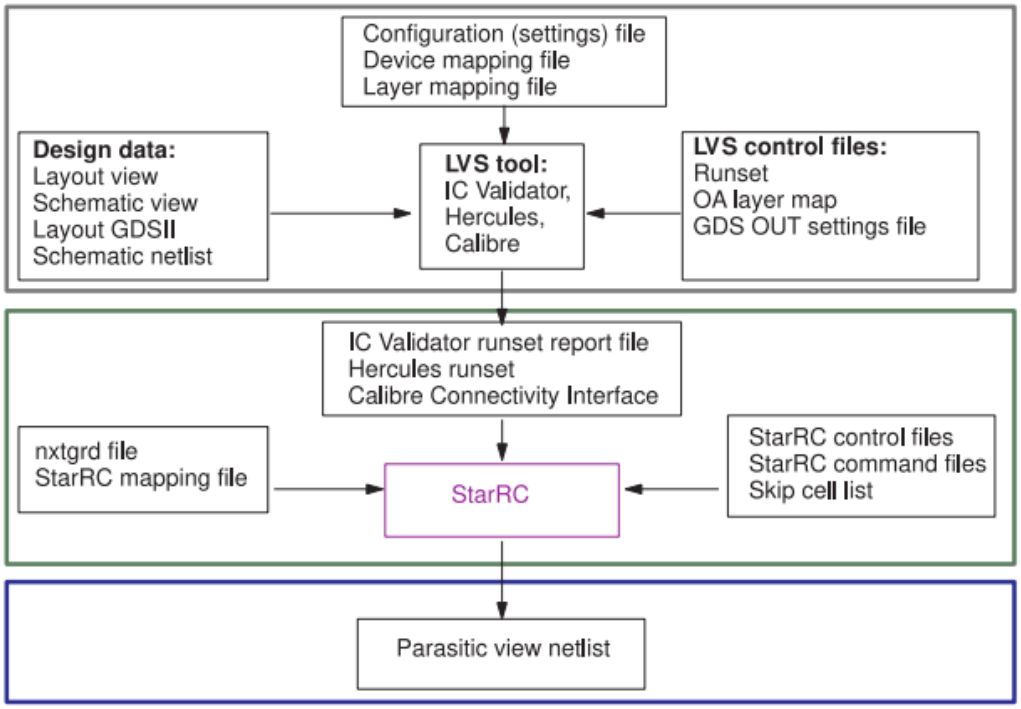


Figura 13: Custom Compiler Extraction Flow. [17]

CAPÍTULO 7

Compuertas lógicas del chip UVG

7.1. Compuertas a simular

Al realizar la extracción de parásitos nos encontramos con el problema de que el archivo no era simulable debido a que dentro del archivo Hspice, se instanciaban compuertas de las cuales no teníamos su comportamiento lógico en algún arhivo de texto para poder simularlo. Por lo tanto, para poder corregir esta problemática, se encontró en la documentación brindada por TSMC el comportamiento lógico de cada compuerta utilizada en el **CI** que anteriormente diseñamos.

En consiguiente, se crearon a nivel transistor cada una de las compuertas, y se le realizaron pruebas en asdasd para poder verificar que las compuertas creadas fueran fieles al comportamiento que TSMC nos mencionaba que estas deben tener.

7.1.1. AN4D1BWP7T

INPUT				OUTPUT
A1	A2	A3	A4	Z
1	1	1	1	1
0	x	x	x	0
x	0	x	x	0
x	x	0	x	0
x	x	x	0	0
x	x	x	x	0

Cuadro 20: Comportamiento lógico de la compuerta AN4D1BWP7T

```
*** Subcircuit for AN4D1BWP7T Gate ***
* D G S B
*AN4D1BWP7T

.include 'And_gate.sp'
.include 'Params.sp'

.SUBCKT AN4D1BWP7T A1 A2 A3 A4 out vsource
       X1 A1 A2 Z1 vsource AndGate size=1
       X2 A3 A4 Z2 vsource AndGate size=1
       X3 Z1 Z2 out vsource AndGate size=1
.ENDS
```

Figura 14: Circuito en HSPICE de la compuerta AN4D1BWP7T.

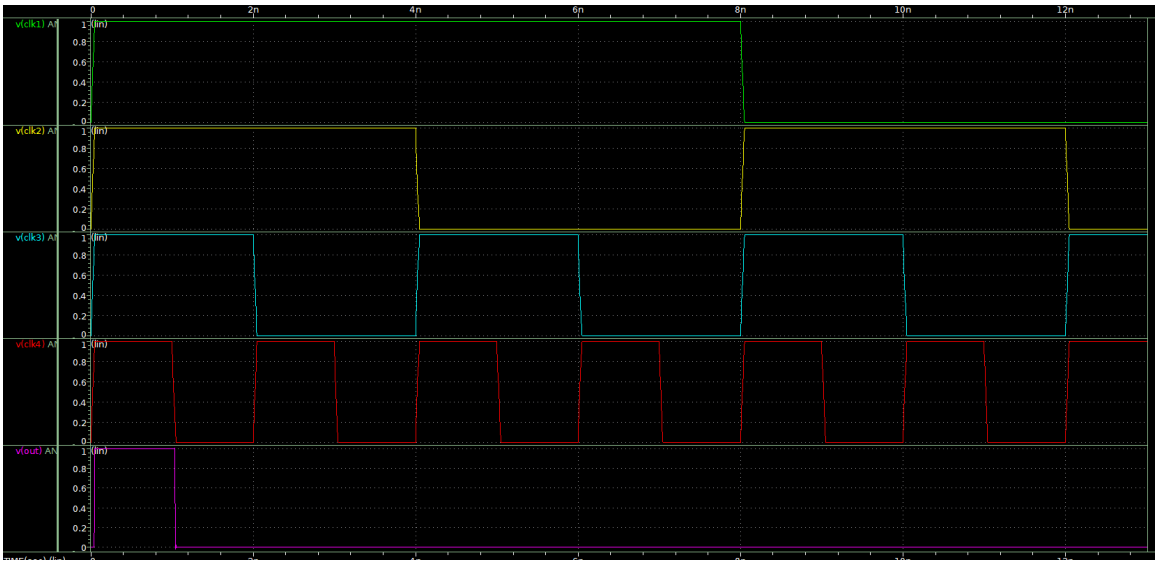


Figura 15: Verificación del comportamiento lógico de la compuerta AN4D1BWP7T.

INPUT				OUTPUT
A1	A2	A3	B	Z
1	1	1	x	1
x	x	x	1	1
0	x	x	0	0
x	0	x	0	0
x	x	0	0	0

Cuadro 21: Comportamiento lógico de la compuerta AO31D1BWP7T

7.1.2. AO31D1BWP7T

```
*** Subcircuit for AO31D1BWP7T Gate ***
* D G S B
*AO31D1BWP7T

.include 'And_gate.sp'
.include 'Or_gate.sp'
.include 'Params.sp'

.SUBCKT AO31D1BWP7T A1 A2 A3 B out vsource
       X1 A1 A2 Y1 vsource AndGate size=1
       X2 Y1 A3 Y2 vsource AndGate size=1
       X4 Y2 B out vsource OrGate size=1
.ENDS
```

Figura 16: Circuito en HSPICE de la compuerta AO31D1BWP7T.

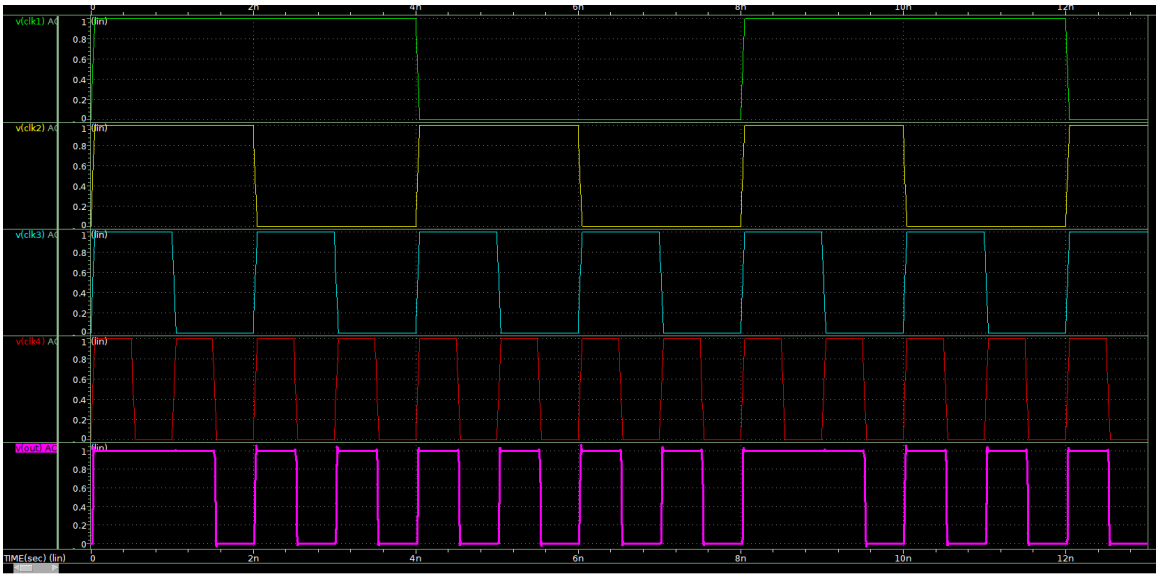


Figura 17: Verificación del comportamiento lógico de la compuerta AO31D1BWP7T.

INPUT			OUTPUT
A1	A2	B	ZN
1	1	x	0
x	x	1	0
0	x	0	1
x	0	0	1

Cuadro 22: Comportamiento lógico de la compuerta AOI21D0BWP7T

7.1.3. AOI21D0BWP7T

```
*** Subcircuit for AOI21D0BWP7T Gate ***
* D G S B
*AOI21D0BWP7T

.include 'And_gate.sp'
.include 'Nor_Gate.sp'
.include 'Params.sp'

.SUBCKT AOI21D0BWP7T A1 A2 B out vsource
       X1 A1 A2 Z1 vsource AndGate size=1
       X3 Z1 B out vsource NorGate size=1
.ENDS
```

Figura 18: Circuito en HSPICE de la compuerta AOI21D0BWP7T.

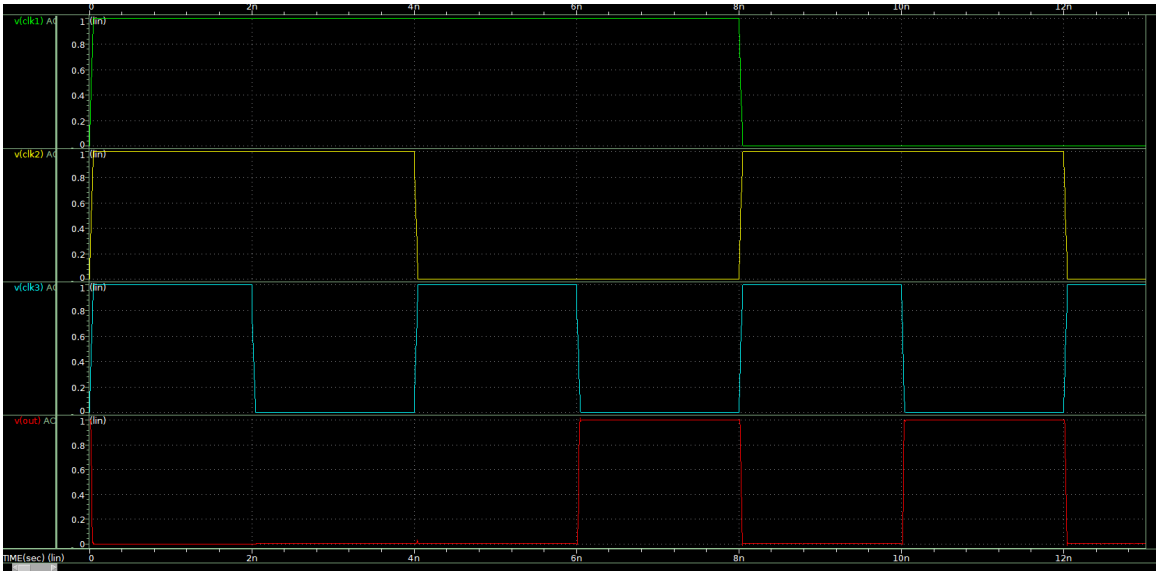


Figura 19: Verificación del comportamiento lógico de la compuerta AOI21D0BWP7T.

7.1.4. AOI222D0BWP7T

INPUT						OUTPUT
A1	A2	B1	B2	C1	C2	ZN
1	1	x	x	x	x	0
x	x	1	1	x	x	0
x	x	x	x	1	1	0
0	x	x	0	x	0	1
x	0	x	0	x	0	1
0	x	0	x	x	0	1
x	0	0	x	x	0	1
0	x	x	0	0	x	1
x	0	x	0	0	x	1
0	x	0	x	0	x	1
x	1	0	x	0	x	1

Cuadro 23: Comportamiento lógico de la compuerta AOI222D0BWP7T

```
*** Subcircuit for AOI222D0BWP7T Gate ***
* D G S B
*AOI222D0BWP7T

.include 'And_gate.sp'
.include 'Nor_Gate.sp'
.include 'Params.sp'

.SUBCKT AOI222D0BWP7T A1 A2 B1 B2 C1 C2 out vsource gnd
        X1 A1 A2 Y1 vsource AndGate size=1
        X2 B1 B2 Y2 vsource AndGate size=1
        X3 C1 C2 Y3 vsource AndGate size=1
        X4 Y1 Y2 Z1 vsource NorGate size=1
        X5 Z1 Y3 out vsource NorGate size=1
.ENDS
```

Figura 20: Circuito en HSPICE de la compuerta AOI222D0BWP7T.

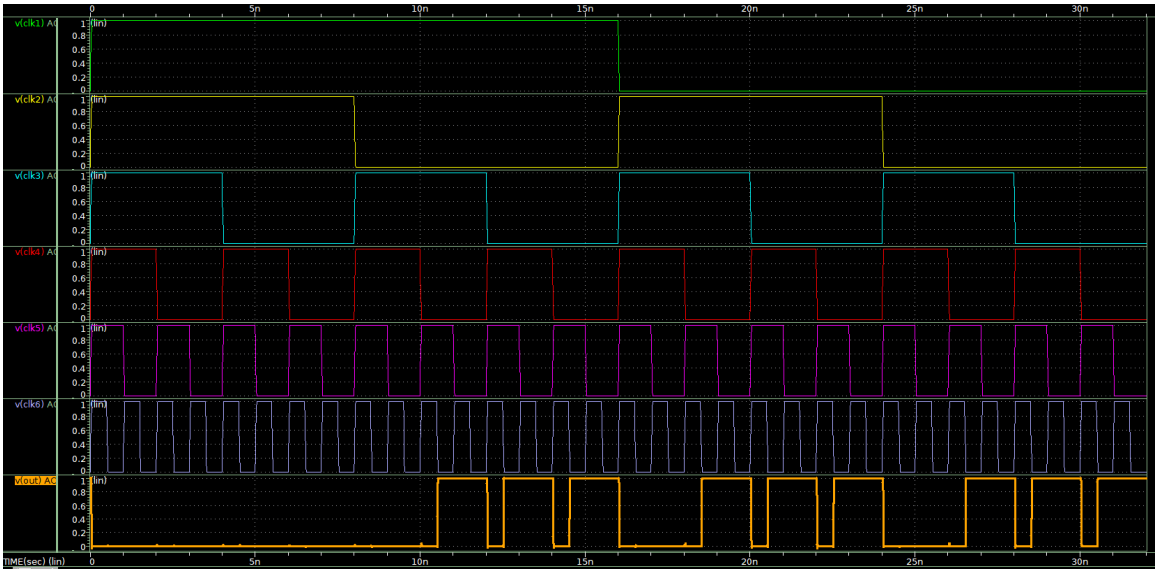


Figura 21: Verificación del comportamiento lógico de la compuerta AOI222D0BWP7T.

7.1.5. CKAN2D1BWP7T

INPUT		OUTPUT
A1	A2	Z
1	1	1
0	x	0
x	0	0

Cuadro 24: Comportamiento lógico de la compuerta CKAN2D1BWP7T

```
*** Subcircuit for CKAN2D1BWP7T Gate ***
* D G S B
*CKAN2D1BWP7T

.include 'And_gate.sp'
.include 'Params.sp'

.SUBCKT CKAN2D1BWP7T A1 A2 out vsource
       X1 A1 A2 out vsource AndGate size=1
.ENDS
```

Figura 22: Circuito en HSPICE de la compuerta CKAN2D1BWP7T.

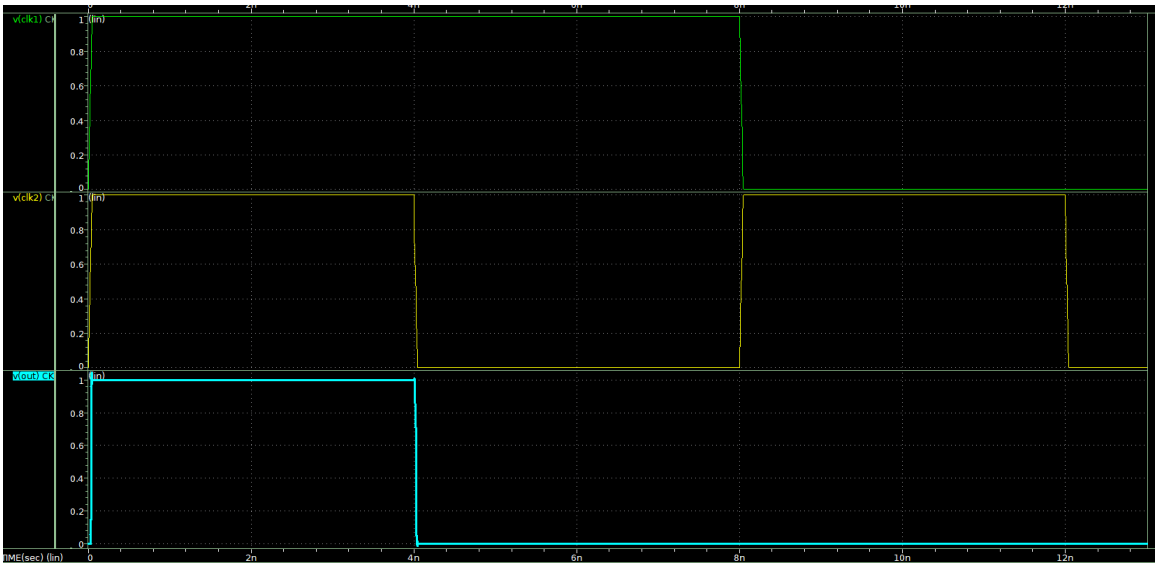


Figura 23: Verificación del comportamiento lógico de la compuerta CKAN2D1BWP7T.

7.1.6. CKXOR2D1BWP7T

INPUT		OUTPUT
A1	A2	Z
0	0	0
0	1	1
1	0	1
1	1	0

Cuadro 25: Comportamiento lógico de la compuerta CKXOR2D1BWP7T

```
*** Subcircuit for CKXOR2D1BWP7T Gate ***
* D G S B
*CKXOR2D1BWP7T

.include 'Xor_Gate.sp'
.include 'Params.sp'

.SUBCKT CKXOR2D1BWP7T A1 A2 out vsource
       X1 A1 A2 out vsource XorGate size=1
.ENDS
```

Figura 24: Circuito en HSPICE de la compuerta CKXOR2D1BWP7T.

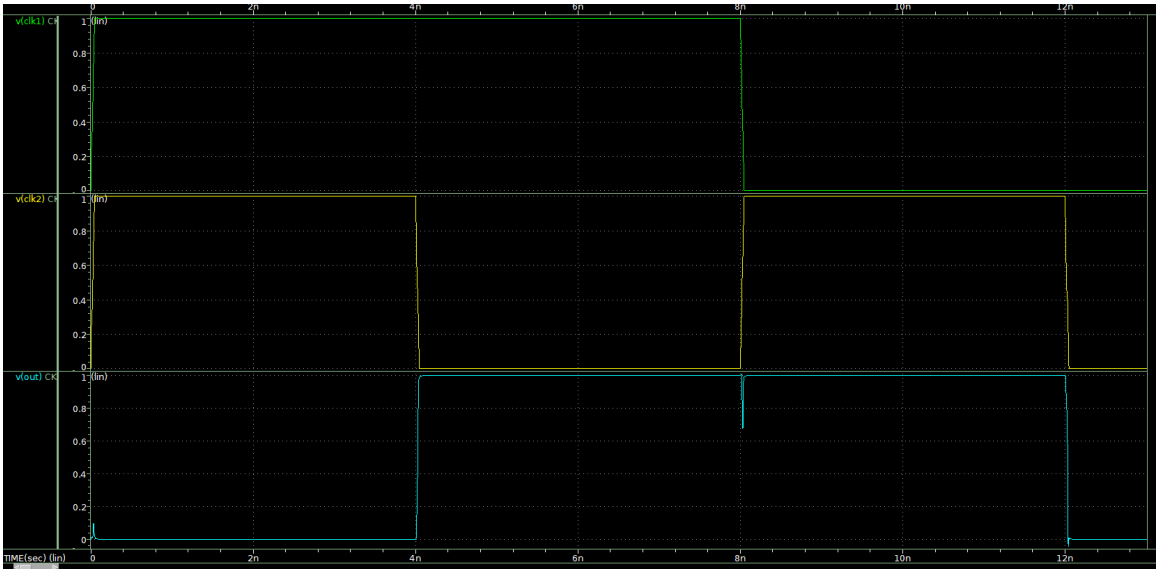


Figura 25: Verificación del comportamiento lógico de la compuerta CKXOR2D1BWP7T.

7.1.7. INR3D0BWP7T

INPUT			OUTPUT
A1	B1	B2	ZN
1	0	0	1
0	x	x	0
x	1	x	0
x	x	1	0

Cuadro 26: Comportamiento lógico de la compuerta INR3D0BWP7T

```
*** Subcircuit for INR3D0BWP7T Gate ***
* D G S B
*INR3D0BWP7T

.include 'Not_gate.sp'
.include 'Nor_Gate.sp'
.include 'Params.sp'

.SUBCKT INR3D0BWP7T A1 B1 B2 out vsource
       X1 A1 A1_neg vsource NotGate size = 1

       X2 A1_neg B1 Z1 vsource NorGate size=1
       X3 Z1 B2 out vsource NorGate size=1
.ENDS
```

Figura 26: Circuito en HSPICE de la compuerta INR3D0BWP7T.

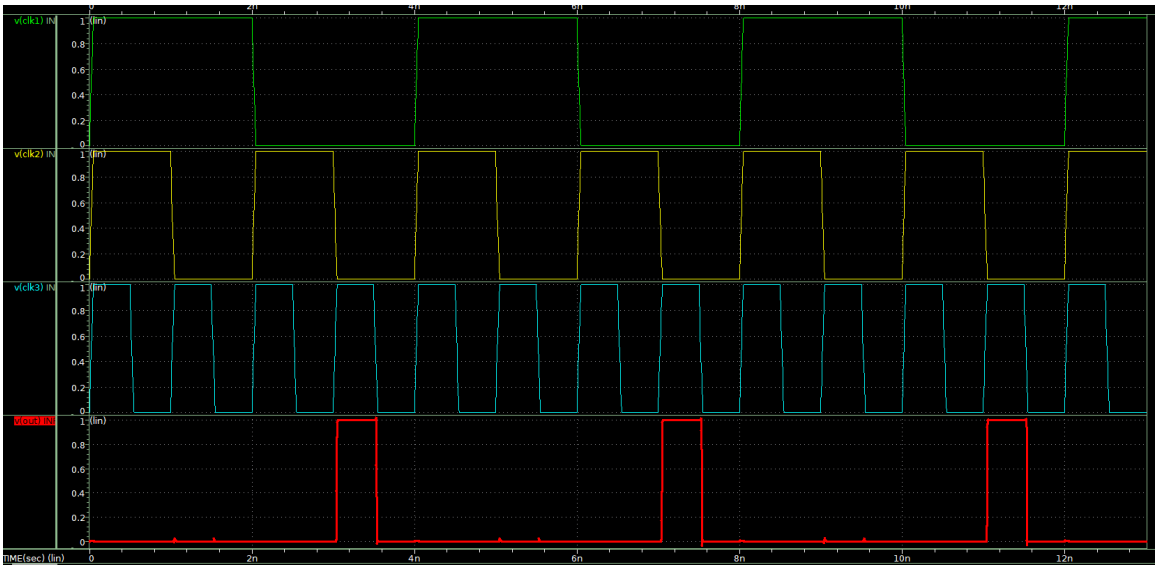


Figura 27: Verificación del comportamiento lógico de la compuerta INR3D0BWP7T.

7.1.8. INR4D0BWP7T

INPUT				OUTPUT
A1	B1	B2	B3	ZN
1	0	0	0	1
0	x	x	x	0
x	1	x	x	0
x	x	1	x	0
x	x	x	1	0

Cuadro 27: Comportamiento lógico de la compuerta INR4D0BWP7T

```
*** Subcircuit for INR4D0BWP7T Gate ***
* D G S B
*INR4D0BWP7T

.include 'Not_gate.sp'
.include 'Nor_Gate.sp'
.include 'Params.sp'

.SUBCKT INR4D0BWP7T A1 A2 B1 B2 out vsource
        X1 A1 A1_neg vsource NotGate size = 1

        X2 A1_neg A2 Z1 vsource NorGate size=1
        X3 B1 B2 Z2 vsource NorGate size=1
        X4 Z1 Z2 out vsource NorGate size=1
.ENDS
```

Figura 28: Circuito en HSPICE de la compuerta INR4D0BWP7T.

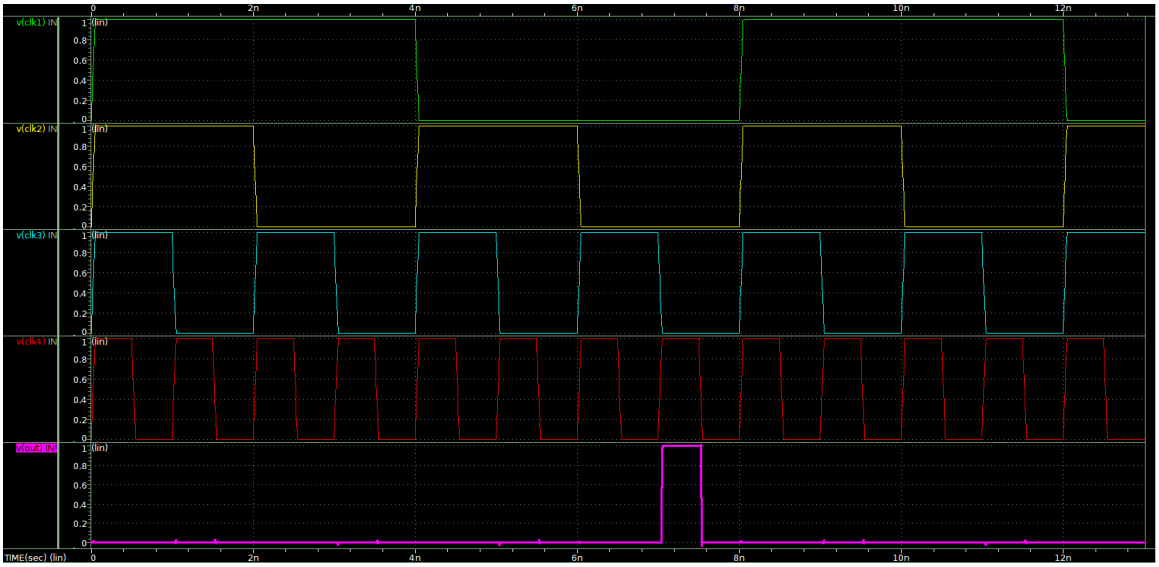


Figura 29: Verificación del comportamiento lógico de la compuerta INR4D0BWP7T.

7.1.9. ND2D1BWP7T

INPUT		OUTPUT
A1	A2	ZN
1	1	0
0	x	1
x	0	1

Cuadro 28: Comportamiento lógico de la compuerta ND2D1BWP7T

```
*** Subcircuit for ND2D1BWP7T Gate ***
* D G S B
*ND2D1BWP7T

.include 'Nand_gate.sp'
.include 'Params.sp'

.SUBCKT ND2D1BWP7T A1 A2 out vsource
        X1 A1 A2 out vsource NandGate size=1
.ENDS
```

Figura 30: Circuito en HSPICE de la compuerta ND2D1BWP7T.

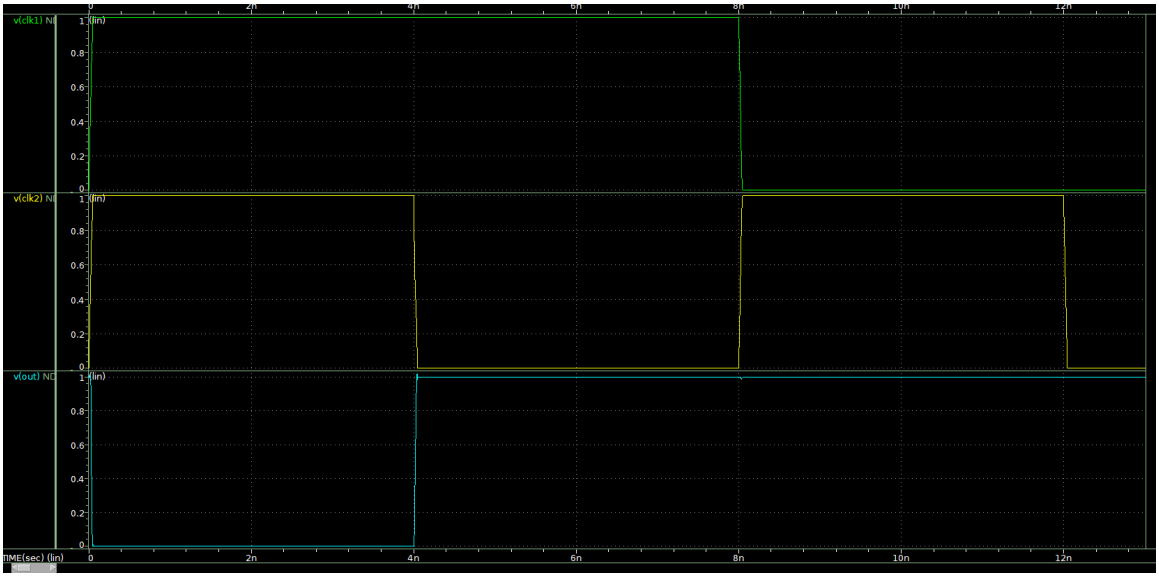


Figura 31: Verificación del comportamiento lógico de la compuerta ND2D1BWP7T.

7.1.10. OAI22D0BWP7T

INPUT				OUTPUT
A1	A2	B1	B2	ZN
0	0	x	x	1
x	x	0	0	1
1	x	x	1	0
x	1	x	1	0
1	x	1	x	0
x	1	1	x	0

Cuadro 29: Comportamiento lógico de la compuerta OAI22D0BWP7T

```
*** Subcircuit for OAI22D0BWP7T Gate ***
* D G S B
*OAI22D0BWP7T

.include 'Nand_gate.sp'
.include 'And_gate.sp'
.include 'Or_Gate.sp'
.include 'Params.sp'

.SUBCKT OAI22D0BWP7T A1 A2 B1 B2 out vsource
        X1 A1 A2 Z1 vsource OrGate size=1
        X2 B1 B2 Z2 vsource OrGate size=1
        X4 Z1 Z2 out vsource NandGate size=1
.ENDS
```

Figura 32: Circuito en HSPICE de la compuerta OAI22D0BWP7T.

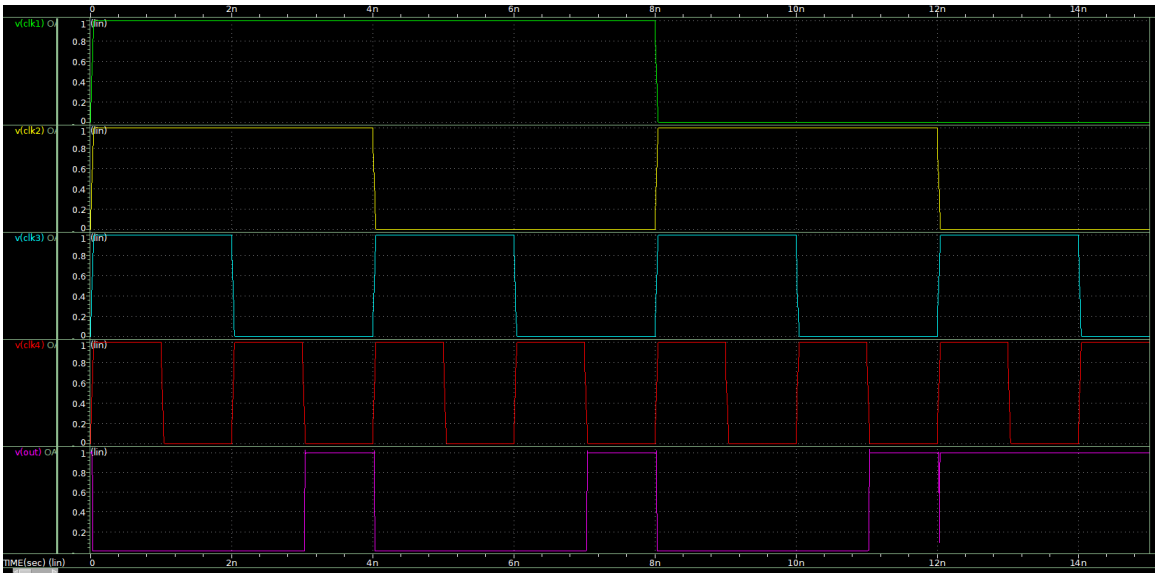


Figura 33: Verificación del comportamiento lógico de la compuerta OAI22D0BWP7T.

7.1.11. OAI221D0BWP7T

```
*** Subcircuit for OAI221D0BWP7T Gate ***
* D G S B
*OAI221D0BWP7T

.include 'Nand_gate.sp'
.include 'And_gate.sp'
.include 'Or_Gate.sp'
.include 'Params.sp'

.SUBCKT OAI221D0BWP7T A1 A2 B1 B2 C out vsource
        X1 A1 A2 Y1 vsource OrGate size=1
        X2 B1 B2 Y2 vsource OrGate size=1
        X3 Y1 Y2 Z1 vsource AndGate size=1
        X4 Z1 C out vsource NandGate size=1
.ENDS
```

Figura 34: Circuito en HSPICE de la compuerta OAI221D0BWP7T.

INPUT					OUTPUT
A1	A2	B1	B2	C	ZN
0	0	x	x	x	1
x	x	0	0	x	1
x	x	x	x	0	1
1	x	x	1	1	0
x	1	x	1	1	0
1	x	1	x	1	0
x	1	1	x	1	0

Cuadro 30: Comportamiento lógico de la compuerta OAI221D0BWP7T

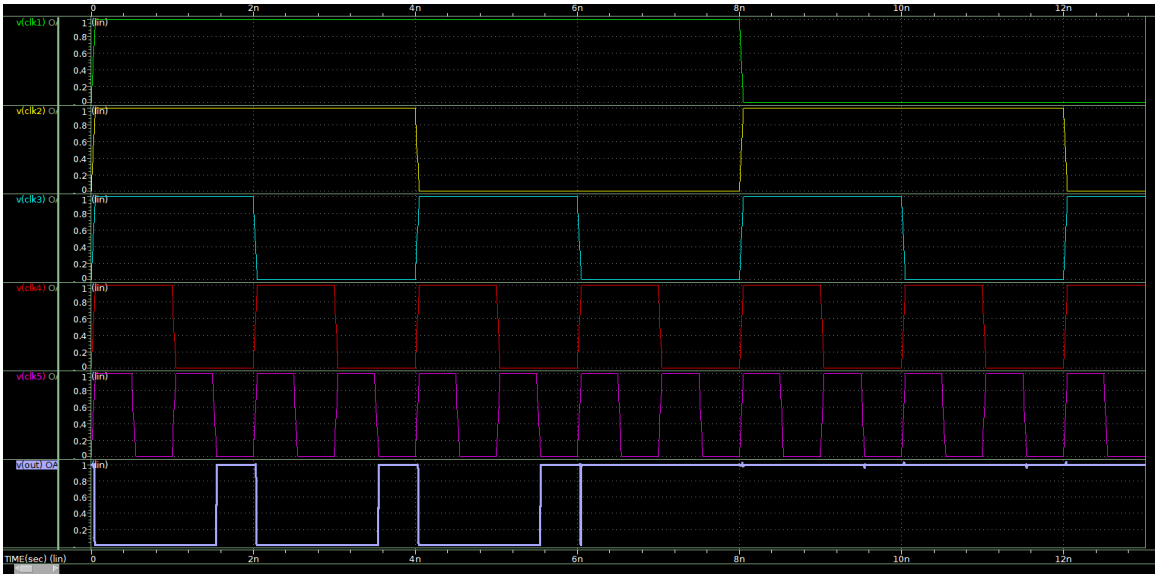


Figura 35: Verificación del comportamiento lógico de la compuerta OAI221D0BWP7T.

7.1.12. OAI222D0BWP7T

INPUT						OUTPUT
A1	A2	B1	B2	C1	C2	ZN
0	0	x	x	x	x	1
x	x	0	0	x	x	1
x	x	x	x	0	0	1
1	x	x	1	x	1	0
x	1	x	1	x	1	0
1	x	1	x	x	1	0
x	1	1	x	x	1	0
1	x	x	1	1	x	0
x	1	x	1	1	x	0
1	x	1	x	1	x	0
x	1	1	x	1	x	0

Cuadro 31: Comportamiento lógico de la compuerta OAI222D0BWP7T

```
*** Subcircuit for OAI222D0BWP7T Gate ***
* D G S B
*OAI222D0BWP7T

.include 'Nand_gate.sp'
.include 'And_gate.sp'
.include 'Or_Gate.sp'
.include 'Params.sp'

.SUBCKT OAI222D0BWP7T A1 A2 B1 B2 C1 C2 out vsource
        X1 A1 A2 Y1 vsource OrGate size=1
        X2 B1 B2 Y2 vsource OrGate size=1
        X3 C1 C2 Y3 vsource OrGate size=1
        X3 Y1 Y2 Z1 vsource AndGate size=1
        X4 Z1 Y3 out vsource NandGate size=1
.ENDS
```

Figura 36: Circuito en HSPICE de la compuerta OAI222D0BWP7T.

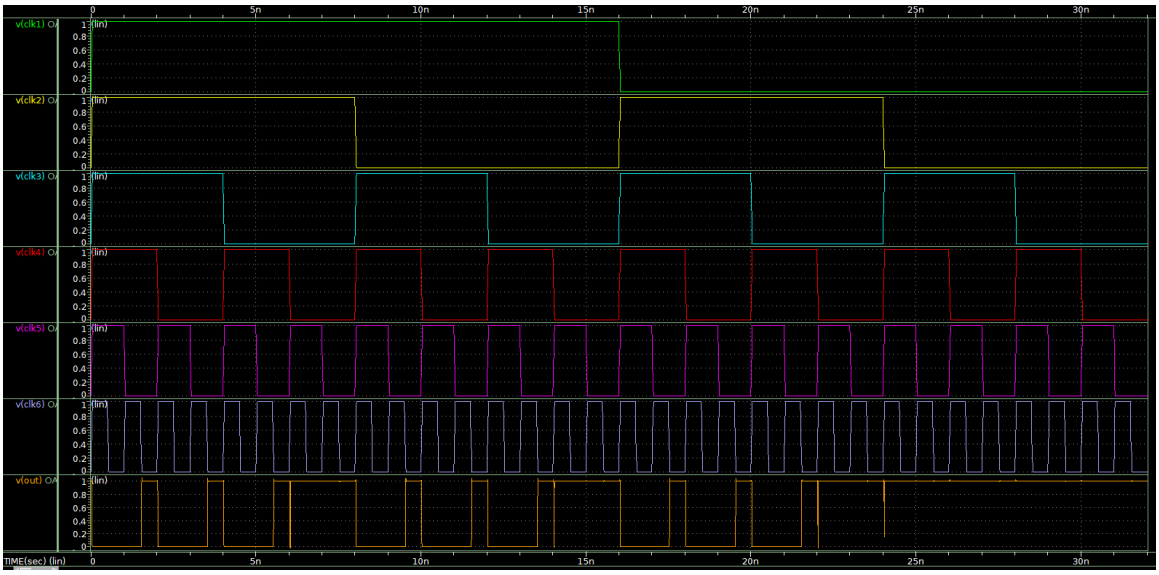


Figura 37: Verificación del comportamiento lógico de la compuerta OAI222D0BWP7T.

7.1.13. OR3D1BWP7T

```
*** Subcircuit for OR3D1BWP7T Gate ***
* D G S B
*OR3D1BWP7T

.include 'Or_Gate.sp'
.include 'Params.sp'

.SUBCKT OR3D1BWP7T A1 A2 A3 out vsource
        X1 A1 A2 Z1 vsource OrGate size = 1
        X2 A3 A3 out vsource OrGate size=1
.ENDS
```

Figura 38: Circuito en HSPICE de la compuerta OR3D1BWP7T.

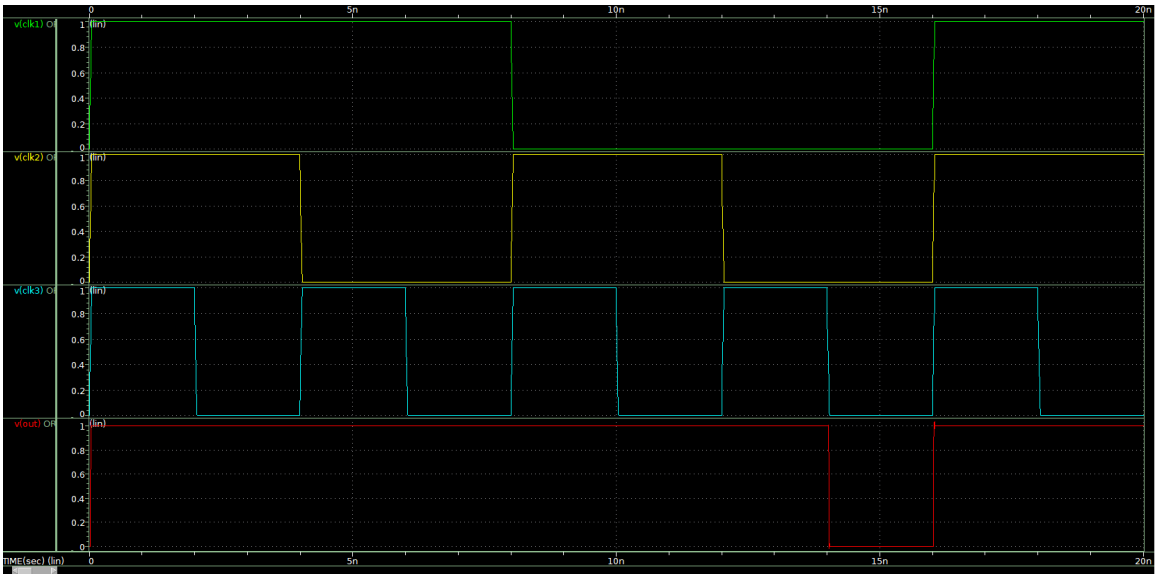


Figura 39: Verificación del comportamiento lógico de la compuerta OR3D1BWP7T.

INPUT			OUTPUT
A1	A2	A3	ZN
0	0	0	0
1	x	x	1
x	1	x	1
x	x	1	1

Cuadro 32: Comportamiento lógico de la compuerta OR3D1BWP7T

CAPÍTULO 8

Simulación de un *ring oscillator*

8.1. Diseñando el *ring oscillator*

Para poder verificar que el CMD creado para realizar la extracción de parásitos funcionara de forma adecuada, se creo un *ring oscillator* con las librerías educativas de Synopsys con su respectivo layout y esquemático como se puede observar en lasa figuras 40 y 41 respectivamente. Por lo tanto, para poder asegurarnos que la extracción no encontraría ningún tipo de conflicto, se realizaron pruebas de DRC y LVS las cuales resultaron en *clean* para el primer caso y en *pass* para el segundo como se puede ver en las figuras 42 y 43.

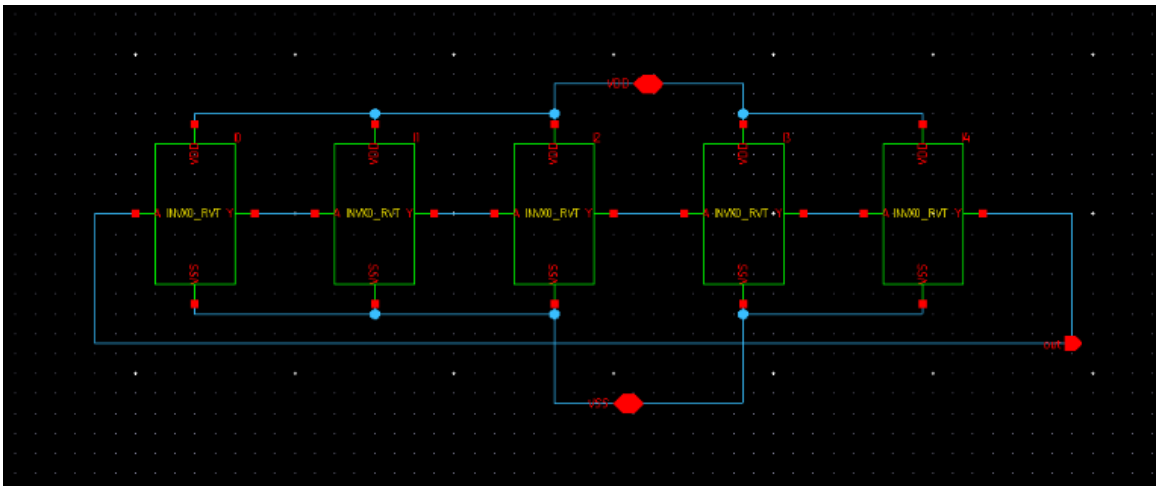


Figura 40: Esquemático del *ring oscillator*.

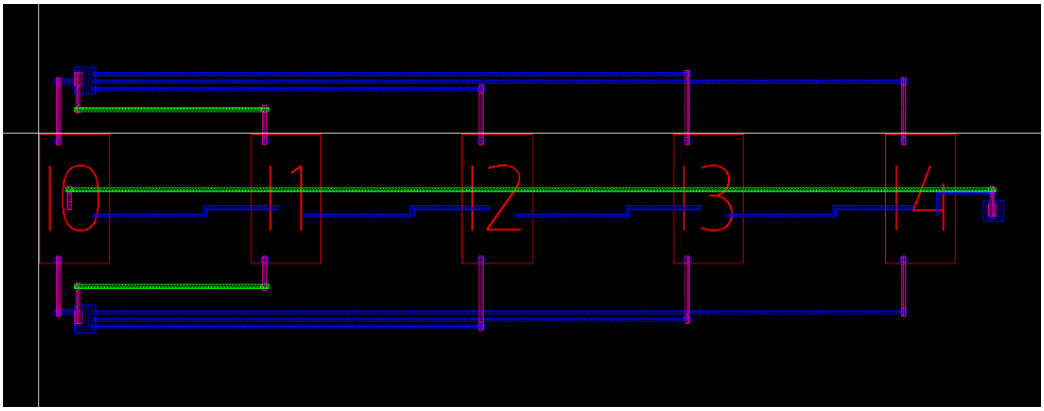


Figura 41: Layout del *ring oscillator*.

LAYOUT ERRORS RESULTS

CLEAN

Model: Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz

DRC Error Statistics

```
Library name:     ROscprueba
Structure name:   ROscprueba
Generated by:     IC Validator RHEL64 S-2021.06-SP3-2.7200131 2022/01/06
Runset name:      /usr/synopsys/iPDK/SAED_PDK32nm/icv/drc/saed32nm_lp9m_drc_rules.rs
User name:        nanoelectronica
Time started:     2022/08/27 12:20:34AM
Time ended:       2022/08/27 12:20:42AM

Called as: icv -f openaccess -i ROscprueba -c ROscprueba -oa_view layout -oa_lib_defs /home/nanoelectronica/Desktop/Folder_de
```

Figura 42: DRC clean del *ring oscillator*.

LVS Compare Results: PASS

DRC and Extraction Results: CLEAN

[ROscprueba, ROscprueba]

Model: Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz

Netlist Extraction Statistics

```
Library name:     ROscprueba
Structure name:   ROscprueba
Generated by:     IC Validator RHEL64 S-2021.06-SP3-2.7200131 2022/01/06
Runset name:      /usr/synopsys/iPDK/SAED_PDK32nm/icv/lvs/saed32nm_lp9m_lvs_rules.rs
User name:        nanoelectronica
Time started:     2022/08/27 12:21:41AM
Time ended:       2022/08/27 12:21:50AM

Called as: icv -f openaccess -i ROscprueba -c ROscprueba -oa_view layout -oa_lib_defs /home/nanoelectronica/Desktop/Folder_de
```

Layout vs. Schematic Statistics

```
Schematic: /home/nanoelectronica/Desktop/Folder_de_Trabajo/synopsys_custom/pvjob_ROscprueba.ROscprueba.icv.lvs/ROscprueba.sch
```

LVS Errors:

```
    1 Successful equivalence points
    0 Failed equivalence points
```

Figura 43: LVS clean del *ring oscillator*.

8.2. Realizando la extracción de parásitos.

Luego, para realizar la extracción de parásitos se usó *custom compiler* y se configuraron las opciones y el archivo de salida como se puede apreciar en las figuras 44, 45 y 46. Y finalmente como se puede observar en el listing 8.1 el archivo de salida SPF contiene el nombre del *top cell* junto con su respectivo pin de entrada y salida (el cual es el mismo debido a la naturaleza del *ring oscillator*) y logró ser completamente simulado en la herramienta hspice obteniendo una frecuencia de 22.25 GHz.

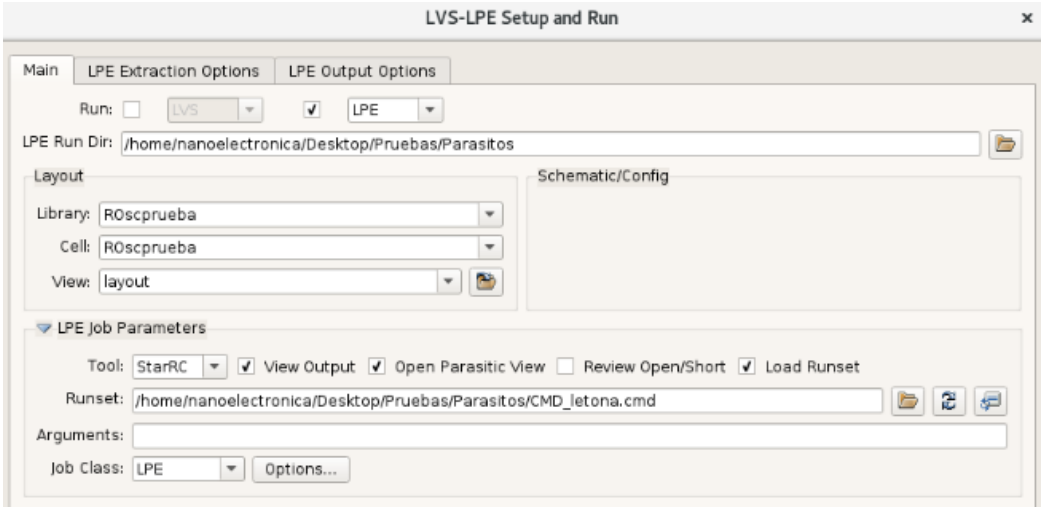


Figura 44: Configuración principal para realizar la extracción de parásitos del *ring oscillator*.

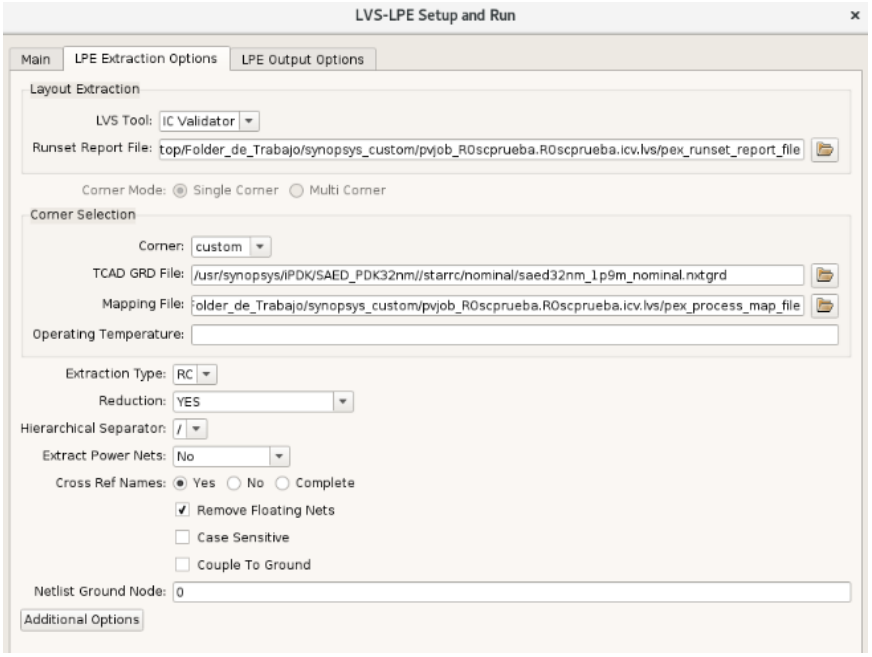


Figura 45: Configuración opcional para realizar la extracción de parásitos del *ring oscillator*.

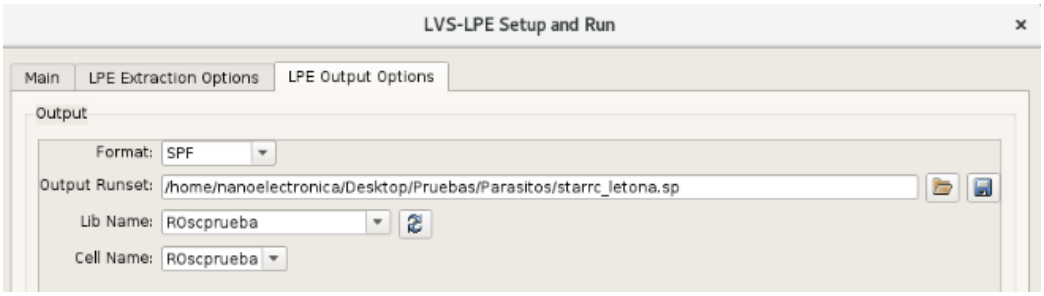


Figura 46: Configuración del archivo de salida de la extracción de parásitos del *ring oscillator*.

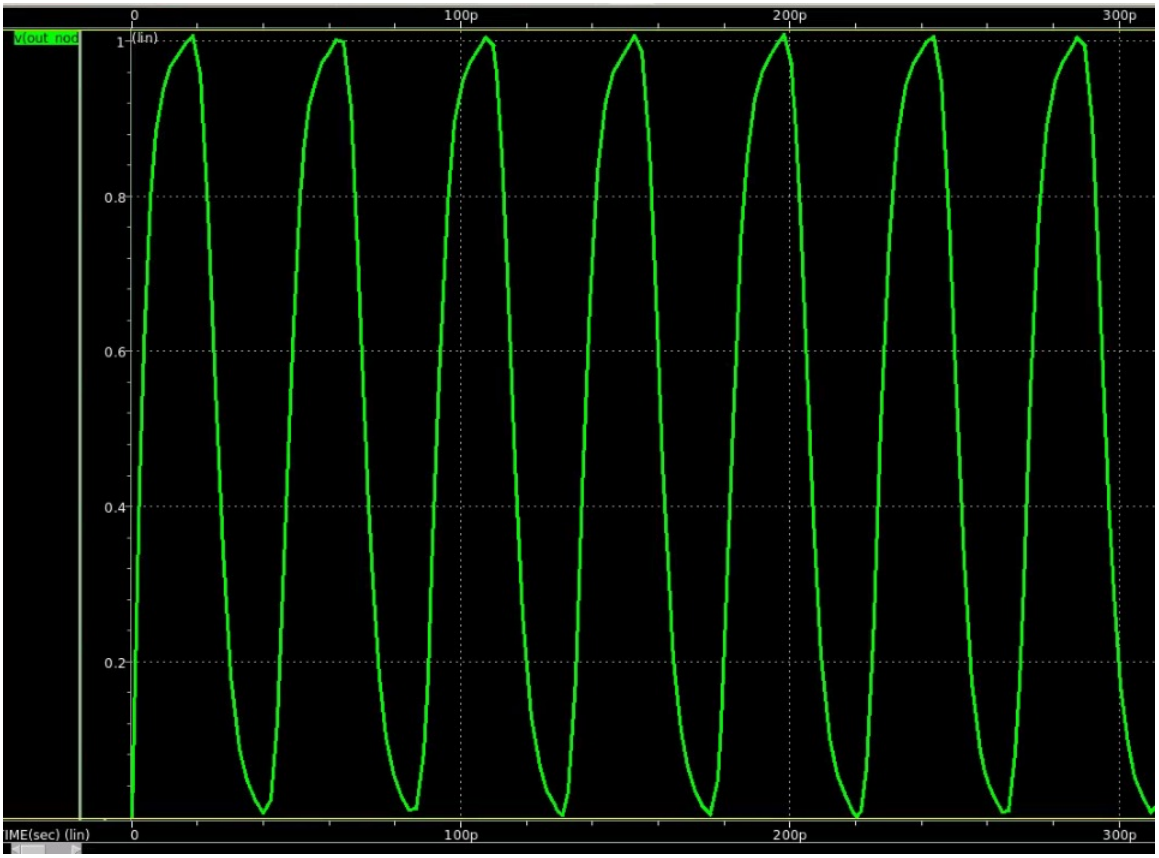


Figura 47: Comportamiento a la salida del *ring oscillator*.

```
****** transient analysis tnom=  25.000 temp=  25.000 ******
period=  44.9401219912p  targ=  47.9200094970p   trig=   2.9798875058p
frequency=  22.2518310074g

runlvl= 3 sets ".opt sim_accuracy= 1"
```

Figura 48: Especificaciones del *ring oscillator*.

```
*
*|DSPF 1.3
*|DESIGN ROscprueba
*|DATE "Sat Aug 6 00:14:05 2022"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "S-2021.06-SP4"
*|DIVIDER /
*|DELIMITER :
**FORMAT SPF
*

** COMMENTS

** OPERATING_TEMPERATURE 25
** DENSITY_OUTSIDE_BLOCK 0
** GLOBAL_TEMPERATURE 25
**
** TCAD_GRD_FILE /usr/synopsys/iPDK/SAED_PDK32nm/starrc/nominal/
    ↪ saed32nm_1p9m_nominal.nxtgrd
** TCAD_TIME_STAMP Wed Mar 16 14:34:54 2011
** TCADGRD_VERSION 72

.SUBCKT ROscprueba out

*|GROUND_NET 0

*|NET out 0.00131888PF
*|P (out B 0 13.4960 -1.0850)
*|I (XI4/MN:DRN XI4/MN DRN B 0 12.4800 -1.3470)
*|I (XI0/MP:GATE XI0/MP GATE I 1.56e-17 0.5050 -0.7570)
*|I (XI0/MN:GATE XI0/MN GATE I 8.1e-18 0.5050 -1.3470)
*|I (XI4/MP:DRN XI4/MP DRN B 0 12.4800 -0.7570)
...
...
...
```

Listing 8.1: Archivo SPF del *ring oscillator*.

CAPÍTULO 9

Conclusiones

- Existen varias formas de realizar la base de datos necesaria para realizar correctamente una extracción de parásitos.
- La extracción de parásitos depende mucho de las etapas anteriores para obtener puertos en el bloque que se desea simular.
- Para realizar una correcta extracción, se debe tener muy presente el tipo de *cells* que se utilizarán, ya que en iteraciones pasadas del nanoChip UVG, se estaba intentando realizar una extracción a nivel transistor, siendo este análisis imposible de realizar debido a la naturaleza de los archivos brindados por TSMC.
- Existen varios comandos útiles para poder obtener información significativa de los **CI** a los cuales se les realiza la extracción.
- Al tener un correcto CMD para realizar la extracción, esta se puede realizar desde *custom compiler* o desde la terminal.
- Con la documentación de TSMC se pueden crear archivos HSPICE que cumplan con el comportamiento lógico de sus compuertas.

CAPÍTULO 10

Recomendaciones

- Realizar la extracción de parásitos utilizando *frame views* desde *ICC2* para evitar utilizar *black boxes* en los flujos de diseño.
- Experimentar la creación de las bases de datos con flujos distintos.
- Tener claro las *cells* que se utilizan en el flujo para evitar hacer análisis incorrectos.
- Automatizar la creación del CMD para evitar invertir tiempo en búsqueda de comandos.
- Utilizar el manual de usuario de StarRC [Star RC] para poder ampliar la información en los comandos opcionales.

CAPÍTULO 11

Bibliografía

[1] A. Altuna Hernández. 2021. *Diseño de un Circuito Integrado con Tecnología de 180nm usando Librerías de Diseño de TSMC: Ejecución de la Síntesis Física, Verificación de Reglas de Diseño y Corrección de Errores Obtenidos.* Tesis Universidad del Valle de Guatemala. págs. 8-97.

[2] Cision. *SMIC Standardizes on Synopsys' StarRC for Signoff Parasitic Extraction:* https://www.prnewswire.com/news-releases/smic-standardizeson-synopsys-starrc-for-signoff-parasitic-extraction-300351456.html.

[3] C. A. Cruz Girón. 2020. *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*. Tesis Universidad del Valle de Guatemala. págs. 1-59.

[4] Design-reuse. *SMIC Standardizes on Synopsys StarRC for Signoff Parasitic Extraction*: https://www.design-reuse.com/news/40824/smic-synopsysstarrc-signoff-parasitic-extraction.html.

[5] E. Carlos. 2018. *Extracción de los Elementos Parásitos de un Circuito Integrado a Partir del Layout*," Tesis Microelectrónica de la Universidad Nacional de Rosario. págs. 1-5.

[6] J. Ayala Escobar. 2021. *Diseño de un Circuito Integrado con Tecnología de 180 nm Usando Librerías de Diseño de TSMC: Ejecución de la Síntesis Física, Verificaciones de Antena y Corrección de Errores Obtenidos.* Tesis Universidad del Valle de Guatemala. págs. 24-62.

[7] J. A. Ruiz Orozco. 2021. *Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: ejecución de la fase de verificación física Layout vs Schematic (LVS)*. Tesis Universidad del Valle de Guatemala. págs. 14-66.

[8] J. E. Shin. 2021. *Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: ejecución de la síntesis física, validación de reglas eléctricas y corrección de errores obtenidos*. Tesis Universidad del Valle de Guatemala. págs. 2-76.

[9] J, Girón. 2020. *Etapa de verificación física de Diseño en Silicio vs. Esquemático (LVS) en el flujo de diseño para un chip a nanoescala*. Tesis Universidad del Valle de Guatemala. págs. 1-74.

[10] J. N. Ruano Orellana. 2020. *Definición del flujo en la herramienta VCS para la simulación de HDLs en la Fabricación de un Chip con Tecnología Nanométrica CMOS*. Tesis Universidad del Valle de Guatemala. págs. 12-56.

[11] K. S. Cardona Polanco. 2021. *Mejoramiento del proceso de síntesis lógica llevada a cabo para la elaboración de un circuito integrado a escala nanométrica*. Tesis Universidad del Valle de Guatemala. págs. 1-83.

[12] L. Nájera Vásquez. 2019. *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 Universidad del Valle de Guatemala. págs. 1-52.

[13] L. Abadía López. 2021. *Posicionamiento e interconexión entre componentes de un circuito sintetizado para el flujo de diseño de un circuito a escala nanométrica utilizando la herramienta de IC Compiler*. Tesis Universidad del Valle de Guatemala. págs. 1-115.

[14] M. Flores Espino. 2020. *Corrección de anillo de entradas/salidas y pruebas de antenna y ERC para la definición del flujo de diseño del primer chip con tecnología nanométrica desarrollado en Guatemala*. Tesis Universidad del Valle de Guatemala. págs. 1-72.

[15] M. Illescas. 2020. *Verificación de reglas de diseño (DRC) para el desarrollo deun flujo funcional de un circuito integrado con tecnología nanométrica*. Tesis Universidad del Valle de Guatemala. págs. 1-79.

[16] *PrimeSim™ Hspice S-2021.09-SP2 User Documentation*, ver. S-2021.09-SP2, Synopsys, Inc, sep. de 2021.

[17] *StarRC™ User Guide and Command Reference*, Version T-2022.03, Synopsys, Inc, mar. de 2022.

[18] S. H. Rubio Vasquez. 2019. *Definición del Flujo de Diseño para Fabricación de un Chip con Tecnologia VLSI CMOS*. Tesis Universidad del Valle de Guatemala. págs. 2-29.

[19] Synopsys, "StarRC Parasitic Extraction," Synopsys, 2015, págs. 1-7.

[20] Synopsys, "*StarRC™ Custom: Next-Generation Modeling and Extraction Solution for Custom IC Designs*," Synopsys, 2010, págs. 1-6.

[21] S. Scholar. *Starrc Parasitic Extraction Starrc Solution:* https://www.semanticscholar.org/paper/Starrc-Parasitic-Extraction-Starrc-SolutionDatasheet-Parasitic/9a624a5edc32ca0ffd54ec66a60959035587bf34.

CAPÍTULO 12

Anexos

12.1. CMD para la extracción de parásitos

```
**-------------------------------------------------------**
** Name:  CMD_letona.cmd
** Date:  18 Nov 2022
** Author:  Carlos Letona
** Description: El siguiente CMD es utilizado para realizar la extraccion de
    ↪  parasitos
**  del nanoChip UVG.
**
**  Based on TSMC template made for customer reference.
**-------------------------INPUT-----------------------------**


BLOCK: chip_IO
MILKYWAY_DATABASE: /home/nanoelectronica/Desktop/Pruebas/Parasitos/
    ↪ MILKYWAY_XTR
TCAD_GRD_FILE: /home/nanoelectronica/Desktop/Pruebas/Parasitos/
    ↪ cm018g_1p6m_4x1u_mim5_40k_cbest.nxtgrd
MAPPING_FILE: /home/nanoelectronica/Desktop/Pruebas/Parasitos/STARRCXT.
    ↪ mapping
ICV_RUNSET_REPORT_FILE: /home/nanoelectronica/Desktop/Pruebas/Parasitos/
    ↪ STARRCXT.runset_rep


*Subcket Pin's order for Simulation
*SPICE_SUBCKT_FILE: /home/nanoelectronica/Desktop/Pruebas/Sintesis_Fisica/
    ↪ ERC/netlist.icv
```

```
**---------------------Ring_oscilator----------------------**

**BLOCK: ROscprueba
**MILKYWAY_DATABASE: ./INVX8.icv.lvs/XTROUT
*TCAD_GRD_FILE: /usr/synopsys/iPDK/SAED_PDK32nm/starrc/nominal/
    ↪ saed32nm_1p9m_nominal.nxtgrd
*MAPPING_FILE: /home/nanoelectronica/Desktop/Folder_de_Trabajo/
    ↪ synopsys_custom/pvjob_ROscprueba.ROscprueba.icv.lvs/
    ↪ pex_process_map_file
*ICV_RUNSET_REPORT_FILE: /home/nanoelectronica/Desktop/Folder_de_Trabajo/
    ↪ synopsys_custom/pvjob_ROscprueba.ROscprueba.icv.lvs/
    ↪ pex_runset_report_file


**-----------------------OUTPUT-----------------------------**

NETLIST_FORMAT: SPF
NETLIST_PASSIVE_PARAMS: YES
NETLIST_FILE: chip_IO.spf
*NETLIST_FILE: starrc_letona.spf


**----------------------OPTIONS-----------------------------**

PLACEMENT_INFO_FILE: YES
PLACEMENT_INFO_FILE_NAME: PLACEMENT_chipIO
*NET XREF WRITE LNXF
CASE_SENSITIVE: NO
HIERARCHICAL_SEPARATOR: /


COUPLE_TO_GROUND: YES
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

POWER_NETS: VDD VSS

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*
```

12.2. Archivo de salida SPF del ring oscillator

```
*
*|DSPF 1.3
*|DESIGN ROscprueba
*|DATE "Sat Aug 6 00:14:05 2022"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "S-2021.06-SP4"
*|DIVIDER /
*|DELIMITER :
**FORMAT SPF
*

** COMMENTS

** OPERATING_TEMPERATURE 25
** DENSITY_OUTSIDE_BLOCK 0
** GLOBAL_TEMPERATURE 25
**
** TCAD_GRD_FILE /usr/synopsys/iPDK/SAED_PDK32nm/starrc/nominal/
    ↪ saed32nm_1p9m_nominal.nxtgrd
** TCAD_TIME_STAMP Wed Mar 16 14:34:54 2011
** TCADGRD_VERSION 72

.SUBCKT ROscprueba out

*|GROUND_NET 0

*|NET out 0.00131888PF
*|P (out B 0 13.4960 -1.0850)
*|I (XI4/MN:DRN XI4/MN DRN B 0 12.4800 -1.3470)
*|I (XI0/MP:GATE XI0/MP GATE I 1.56e-17 0.5050 -0.7570)
*|I (XI0/MN:GATE XI0/MN GATE I 8.1e-18 0.5050 -1.3470)
*|I (XI4/MP:DRN XI4/MP DRN B 0 12.4800 -0.7570)
Cg1_1 out 0 5.49422e-17
```

```
Cg1_2 XI4/MN:DRN 0 1.80697e-17
Cg1_3 XI0/MN:GATE 0 1.1567e-16
Cg1_4 XI4/MP:DRN 0 5.11274e-17
Cg1_5 out:8 0 9.10812e-17
Cg1_6 out:9 0 5.25342e-17
Cg1_7 out:12 0 8.36724e-16
R1_1 out out:12 0.581429
R1_2 XI4/MN:DRN out:8 50.1661
R1_3 XI0/MP:GATE XI0/MN:GATE 0.001
R1_4 XI0/MN:GATE out:11 0.001
R1_5 XI0/MN:GATE out:10 0.001
R1_6 XI0/MN:GATE out:12 160.517
R1_7 XI4/MP:DRN out:7 50.1021
R1_8 XI4/MP:DRN out:6 150.0
R1_9 XI4/MP:DRN out:9 150.0
R1_10 out:6 out:7 0.184
R1_11 out:6 out:9 0.175
R1_12 out:8 out:9 0.5
R1_13 out:8 out:12 3.24024

*|NET net8 0.000593207PF
*|I (XI0/MP:DRN XI0/MP DRN B 0 0.5200 -0.7570)
*|I (XI1/MN:GATE XI1/MN GATE I 8.1e-18 3.4950 -1.3470)
*|I (XI0/MN:DRN XI0/MN DRN B 0 0.5200 -1.3470)
*|I (XI1/MP:GATE XI1/MP GATE I 1.56e-17 3.4950 -0.7570)
C5_1 XI1/MN:GATE XI0/MN:GATE 1.7775e-22
C5_2 net8:6 XI0/MN:GATE 3.16223e-18
C5_3 net8:7 XI0/MN:GATE 3.94068e-18
C5_4 XI0/MP:DRN out:12 5.2989e-19
C5_5 XI0/MN:DRN out:12 3.76131e-19
C5_6 XI1/MP:GATE out:12 2.95561e-21
C5_7 net8:6 out:12 9.30984e-18
C5_8 net8:7 out:12 3.21386e-17
Cg5_9 XI0/MP:DRN 0 3.16383e-17
Cg5_10 XI1/MN:GATE 0 5.93936e-17
Cg5_11 XI0/MN:DRN 0 1.70079e-17
Cg5_12 XI1/MP:GATE 0 5.01816e-17
Cg5_13 net8:6 0 5.35573e-17
Cg5_14 net8:7 0 2.99314e-16
R5_1 XI0/MP:DRN net8:5 37.6607
R5_2 XI0/MP:DRN net8:6 150.0
R5_3 XI1/MN:GATE net8:7 140.602
R5_4 XI1/MN:GATE XI1/MP:GATE 0.001
R5_5 XI1/MN:GATE net8:8 0.001
R5_6 XI0/MN:DRN net8:7 50.1661
R5_7 XI1/MP:GATE net8:9 0.001
R5_8 net8:5 net8:6 0.175
R5_9 net8:6 net8:7 0.5
```

```
*|NET net12 0.000616888PF
*|I (XI2/MN:GATE XI2/MN GATE I 8.1e-18 6.4850 -1.3470)
*|I (XI1/MN:DRN XI1/MN DRN B 0 3.5100 -1.3470)
*|I (XI2/MP:GATE XI2/MP GATE I 1.56e-17 6.4850 -0.7570)
*|I (XI1/MP:DRN XI1/MP DRN B 0 3.5100 -0.7570)
C6_1 XI2/MN:GATE XI1/MN:GATE 1.7775e-22
C6_2 net12:8 XI1/MN:GATE 3.94302e-18
C6_3 net12:7 XI1/MP:GATE 3.17694e-18
C6_4 net12:7 net8:6 1.1919e-19
C6_5 net12:8 net8:6 4.71678e-22
C6_6 XI1/MN:DRN net8:7 4.71673e-19
C6_7 XI1/MP:DRN net8:7 5.915e-19
C6_8 net12:7 net8:7 6.30932e-18
C6_9 net12:8 net8:7 1.80414e-17
Cg6_10 XI2/MN:GATE 0 5.93269e-17
Cg6_11 XI1/MN:DRN 0 1.81584e-17
Cg6_12 XI2/MP:GATE 0 5.01769e-17
Cg6_13 XI1/MP:DRN 0 5.2349e-17
Cg6_14 net12:7 0 5.82359e-17
Cg6_15 net12:8 0 3.13205e-16
R6_1 XI2/MN:GATE net12:8 140.602
R6_2 XI2/MN:GATE XI2/MP:GATE 0.001
R6_3 XI2/MN:GATE net12:9 0.001
R6_4 XI1/MN:DRN net12:8 50.1661
R6_5 XI2/MP:GATE net12:10 0.001
R6_6 XI1/MP:DRN net12:6 50.1021
R6_7 XI1/MP:DRN net12:5 150.0
R6_8 XI1/MP:DRN net12:7 150.0
R6_9 net12:5 net12:6 0.184
R6_10 net12:5 net12:7 0.175
R6_11 net12:7 net12:8 0.5

*|NET net16 0.000616608PF
*|I (XI3/MN:GATE XI3/MN GATE I 8.1e-18 9.4750 -1.3470)
*|I (XI2/MN:DRN XI2/MN DRN B 0 6.5000 -1.3470)
*|I (XI3/MP:GATE XI3/MP GATE I 1.56e-17 9.4750 -0.7570)
*|I (XI2/MP:DRN XI2/MP DRN B 0 6.5000 -0.7570)
C7_1 XI3/MN:GATE XI2/MN:GATE 1.7775e-22
C7_2 net16:8 XI2/MN:GATE 3.94334e-18
C7_3 net16:7 XI2/MP:GATE 3.17694e-18
C7_4 net16:7 net12:7 2.49322e-19
C7_5 net16:8 net12:7 3.92411e-21
C7_6 XI2/MN:DRN net12:8 4.71673e-19
C7_7 XI2/MP:DRN net12:8 5.91928e-19
C7_8 net16:7 net12:8 6.31579e-18
C7_9 net16:8 net12:8 1.80291e-17
Cg7_10 XI3/MN:GATE 0 5.93581e-17
```

```
Cg7_11 XI2/MN:DRN 0 1.80958e-17
Cg7_12 XI3/MP:GATE 0 5.01769e-17
Cg7_13 XI2/MP:DRN 0 5.23221e-17
Cg7_14 net16:7 0 5.81342e-17
Cg7_15 net16:8 0 3.12946e-16
R7_1 XI3/MN:GATE net16:8 140.602
R7_2 XI3/MN:GATE XI3/MP:GATE 0.001
R7_3 XI3/MN:GATE net16:9 0.001
R7_4 XI2/MN:DRN net16:8 50.1661
R7_5 XI3/MP:GATE net16:10 0.001
R7_6 XI2/MP:DRN net16:6 50.1021
R7_7 XI2/MP:DRN net16:5 150.0
R7_8 XI2/MP:DRN net16:7 150.0
R7_9 net16:5 net16:6 0.184
R7_10 net16:5 net16:7 0.175
R7_11 net16:7 net16:8 0.5

*|NET net20 0.000614572PF
*|I (XI4/MN:GATE XI4/MN GATE I 8.1e-18 12.4650 -1.3470)
*|I (XI3/MN:DRN XI3/MN DRN B 0 9.4900 -1.3470)
*|I (XI4/MP:GATE XI4/MP GATE I 1.56e-17 12.4650 -0.7570)
*|I (XI3/MP:DRN XI3/MP DRN B 0 9.4900 -0.7570)
C8_1 net20:8 XI4/MN:DRN 4.71673e-19
C8_2 net20:8 XI4/MP:DRN 5.91945e-19
C8_3 XI4/MN:GATE out:8 3.20618e-18
C8_4 net20:7 out:8 2.45257e-21
C8_5 net20:8 out:8 4.93615e-18
C8_6 XI4/MN:GATE out:9 7.36315e-19
C8_7 XI4/MP:GATE out:9 3.18371e-18
C8_8 net20:7 out:9 2.53175e-19
C8_9 net20:8 out:9 1.97234e-17
C8_10 XI4/MP:GATE out:12 2.95561e-21
C8_11 net20:7 out:12 2.45744e-18
C8_12 net20:8 out:12 1.37059e-17
C8_13 XI4/MN:GATE XI3/MN:GATE 1.7775e-22
C8_14 net20:8 XI3/MN:GATE 3.94304e-18
C8_15 net20:7 XI3/MP:GATE 3.17694e-18
C8_16 net20:7 net16:7 2.49322e-19
C8_17 net20:8 net16:7 3.92411e-21
C8_18 XI3/MN:DRN net16:8 4.71673e-19
C8_19 XI3/MP:DRN net16:8 5.91519e-19
C8_20 net20:7 net16:8 6.31579e-18
C8_21 net20:8 net16:8 1.80407e-17
Cg8_22 XI4/MN:GATE 0 5.91942e-17
Cg8_23 XI3/MN:DRN 0 1.80958e-17
Cg8_24 XI4/MP:GATE 0 4.99947e-17
Cg8_25 XI3/MP:DRN 0 5.22582e-17
Cg8_26 net20:7 0 5.53815e-17
```

```
Cg8_27 net20:8 0 2.97584e-16
R8_1 XI4/MN:GATE XI4/MP:GATE 0.001
R8_2 XI4/MN:GATE net20:9 0.001
R8_3 XI4/MN:GATE net20:8 140.602
R8_4 XI3/MN:DRN net20:8 50.1661
R8_5 XI4/MP:GATE net20:10 0.001
R8_6 XI3/MP:DRN net20:6 50.1021
R8_7 XI3/MP:DRN net20:5 150.0
R8_8 XI3/MP:DRN net20:7 150.0
R8_9 net20:5 net20:6 0.184
R8_10 net20:5 net20:7 0.175
R8_11 net20:7 net20:8 0.5

*
* Instance Section
*
XXI4/MN XI4/MN:DRN XI4/MN:GATE VSS VSS n105 SCA=155.435 SCB=0.0349949 SCC
    ↪ =0.0136582 ad=0.02754p as=0.02754p l=0.03u m=1 nf=1 nrd=0.377778 nrs
    ↪ =0.377778 pd=0.744u ps=0.744u w=0.27u
XXI0/MP XI0/MP:DRN XI0/MP:GATE VDD VDD p105 SCA=182.158 SCB=0.101797 SCC
    ↪ =0.0183349 ad=0.05304p as=0.05304p l=0.03u m=1 nf=1 nrd=0.196154 nrs
    ↪ =0.196154 pd=1.244u ps=1.244u w=0.52u
XXI3/MN XI3/MN:DRN XI3/MN:GATE VSS VSS n105 SCA=154.121 SCB=0.034853 SCC
    ↪ =0.0136582 ad=0.02754p as=0.02754p l=0.03u m=1 nf=1 nrd=0.377778 nrs
    ↪ =0.377778 pd=0.744u ps=0.744u w=0.27u
XXI2/MN XI2/MN:DRN XI2/MN:GATE VSS VSS n105 SCA=154.121 SCB=0.034853 SCC
    ↪ =0.0136582 ad=0.02754p as=0.02754p l=0.03u m=1 nf=1 nrd=0.377778 nrs
    ↪ =0.377778 pd=0.744u ps=0.744u w=0.27u
XXI1/MN XI1/MN:DRN XI1/MN:GATE VSS VSS n105 SCA=154.121 SCB=0.034853 SCC
    ↪ =0.0136582 ad=0.02754p as=0.02754p l=0.03u m=1 nf=1 nrd=0.377778 nrs
    ↪ =0.377778 pd=0.744u ps=0.744u w=0.27u
XXI0/MN XI0/MN:DRN XI0/MN:GATE VSS VSS n105 SCA=156.918 SCB=0.0363658 SCC
    ↪ =0.013662 ad=0.02754p as=0.02754p l=0.03u m=1 nf=1 nrd=0.377778 nrs
    ↪ =0.377778 pd=0.744u ps=0.744u w=0.27u
XXI4/MP XI4/MP:DRN XI4/MP:GATE VDD VDD p105 SCA=182.158 SCB=0.101797 SCC
    ↪ =0.0183349 ad=0.05304p as=0.05304p l=0.03u m=1 nf=1 nrd=0.196154 nrs
    ↪ =0.196154 pd=1.244u ps=1.244u w=0.52u
XXI3/MP XI3/MP:DRN XI3/MP:GATE VDD VDD p105 SCA=182.158 SCB=0.101797 SCC
    ↪ =0.0183349 ad=0.05304p as=0.05304p l=0.03u m=1 nf=1 nrd=0.196154 nrs
    ↪ =0.196154 pd=1.244u ps=1.244u w=0.52u
XXI2/MP XI2/MP:DRN XI2/MP:GATE VDD VDD p105 SCA=182.158 SCB=0.101797 SCC
    ↪ =0.0183349 ad=0.05304p as=0.05304p l=0.03u m=1 nf=1 nrd=0.196154 nrs
    ↪ =0.196154 pd=1.244u ps=1.244u w=0.52u
XXI1/MP XI1/MP:DRN XI1/MP:GATE VDD VDD p105 SCA=182.158 SCB=0.101797 SCC
    ↪ =0.0183349 ad=0.05304p as=0.05304p l=0.03u m=1 nf=1 nrd=0.196154 nrs
    ↪ =0.196154 pd=1.244u ps=1.244u w=0.52u

.ENDS
```