

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



**Diseño de un circuito integrado con tecnología de 180 nm
usando librerías de diseño de TSMC: ejecución de la fase de
verificación física Layout vs Schematic (LVS)**

Trabajo de graduación presentado por José Alejandro Ruiz Orozco
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2021

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



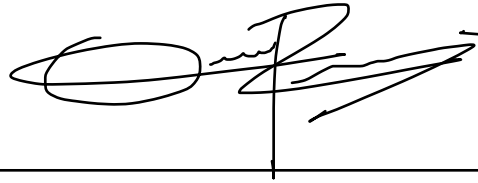
**Diseño de un circuito integrado con tecnología de 180 nm
usando librerías de diseño de TSMC: ejecución de la fase de
verificación física Layout vs Schematic (LVS)**

Trabajo de graduación presentado por José Alejandro Ruiz Orozco
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,


2021

Vo.Bo.:



(f) _____
Ing. Ricardo Girón

Tribunal Examinador:



(f) _____
Ing. Ricardo Girón



(f) _____
MSc. Carlos Esquit



(f) _____
Ing. Jonathan de los Santos

Fecha de aprobación: Guatemala, 8 de enero de 2021.

Prefacio

Este trabajo es dedicado a mi familia, especialmente a mis padres, por todo el esfuerzo y dedicación que han hecho para apoyar mi crecimiento académico, mental y espiritual. Agradezco a mis amigos, por los increíbles momentos que se convirtieron en atesorados recuerdos e historias. Agradezco finalmente a mis maestros, por enseñarme que si se puede imaginar, se puede crear.

Será como árbol plantado a la orilla del río, que da su fruto a su tiempo y su hoja no cae; y todo lo que hace, prosperará.

En memoria de Pipo.

Prefacio	v
Lista de figuras	XIII
Lista de cuadros	XV
Resumen	XVII
Abstract	XIX
1. Introducción	1
2. Antecedentes	3
3. Justificación	5
4. Objetivos	7
Objetivo general	7
Objetivos específicos	7
5. Alcance	9
6. Marco teórico	11
6.1. Etapas del flujo de diseño	11
6.1.1. <i>Product Requirement</i>	11
6.1.2. <i>Behavioral/Functional Specification</i>	12
6.1.3. <i>Behavioral (RTL) synthesis</i>	12
6.1.4. <i>Structural Specification</i>	13
6.1.5. <i>Physical Synthesis</i>	13
6.1.6. <i>Physical Specification</i>	13
6.2. <i>Layout vs. Schematic (LVS)</i>	14
6.3. Herramientas utilizadas	16
6.3.1. <i>CDL</i>	16
6.3.2. <i>Netlist</i>	16

6.3.3.	NetTran	16
6.3.4.	<i>GDS</i>	17
6.3.5.	<i>Custom Compiler</i>	17
7.	Proceso de LVS	19
7.1.	Archivos previos	19
7.2.	Archivos generados	24
7.2.1.	Archivo <i>ICV</i>	24
7.3.	Librerías a <i>Custom Compiler</i>	25
7.3.1.	Archivo <i>.ndm</i>	25
7.3.2.	Ejecución de <i>Custom Compiler</i>	26
7.4.	Archivo <i>GDS</i>	30
7.5.	Archivo <i>Runset</i>	32
8.	Circuitos desarrollados	35
8.1.	<i>Layouts</i> de circuitos	35
8.1.1.	Compuerta <i>NOT</i>	35
8.1.2.	Compuerta <i>XOR</i>	36
8.1.3.	Circuito <i>Full Adder</i>	37
8.1.4.	Circuito <i>ALU</i>	37
8.1.5.	Circuito <i>Counter</i>	38
8.1.6.	Memoria <i>RAM</i>	38
9.	Métodos de ejecución de LVS	39
9.1.	<i>Custom Compiler</i>	39
9.2.	Comandos en terminal	44
9.3.	Archivos generados	47
10.	Retos y soluciones	51
10.1.	Comandos en terminal	51
10.2.	Archivo <i>Runset</i>	51
10.3.	Exportar archivo <i>GDS</i>	54
10.4.	Ejecutar LVS en <i>Custom Compiler</i>	54
11.	Resultados	57
11.1.	<i>NOT</i>	60
11.2.	<i>XOR</i>	61
11.3.	<i>Full Adder</i>	62
11.4.	<i>ALU</i>	63
11.5.	<i>4 Bit Counter</i>	64
11.6.	<i>RAM Memmory</i>	65
12.	Proyecto final: El Gran Jaguar	67
12.1.	Documentos de Síntesis Lógica	68
12.2.	Documentos de Síntesis Física	68
12.3.	Documentos generados	69
12.4.	Vista de <i>Layout</i>	71
12.5.	Resultados de LVS	71

13.Conclusiones	73
14.Recomendaciones	75
15.Bibliografía	77

Lista de figuras

1.	Flow Design	12
2.	Pasos de VLSI	13
3.	LVS basics	14
4.	LVS flow	15
5.	Salidas Síntesis Lógica	20
6.	Archivo <i>Verilog</i> de Síntesis Lógica	20
7.	Salidas Síntesis Física	21
8.	Carpeta <i>.ndm</i> de Síntesis Física	21
9.	Carpeta de Trabajo Compartida	22
10.	Carpeta de Compuerta <i>NOT</i>	22
11.	Documentos recuperados	23
12.	Concatenación de librerías	24
13.	Traducción de archivos	24
14.	Extracción de <i>ICV</i>	25
15.	Carpeta local de trabajo para la compuerta <i>XOR</i>	26
16.	Inicio de <i>Custom Compiler</i>	27
17.	<i>Library Manager</i> dentro de <i>Custom Compiler</i>	27
18.	Menú de <i>File</i>	28
19.	Opción <i>Add ICC2 Library</i>	28
20.	Copia de carpeta <i>.ndm</i>	29
21.	Nombre de nueva librería	29
22.	Apartado <i>Attributes</i>	30
23.	Librería agregada a <i>Library Manager</i>	30
24.	Opción <i>Export Stream</i>	31
25.	Configuración <i>Export Stream</i>	32
26.	<i>Environment Setup</i> en <i>Runset</i>	33
27.	Formato de <i>black boxes</i> en <i>Runset</i>	33
28.	Layout de compuerta <i>NOT</i>	36
29.	Layout de compuerta <i>XOR</i>	36
30.	Layout de circuito <i>Full Adder</i>	37
31.	Layout de circuito <i>ALU</i>	37

32.	Layout de circuito <i>Counter</i>	38
33.	Layout de memoria <i>RAM</i>	38
34.	Vista de <i>Layout</i> en <i>Custom Compiler</i>	40
35.	<i>Setup</i> para ejecución de LVS	40
36.	Configuración de LVS, <i>Run Directory</i>	41
37.	Configuración de LVS, <i>Layout</i>	42
38.	Configuración de LVS, <i>Schematic</i>	42
39.	Configuración de LVS, <i>Runset</i>	43
40.	Ubicación de archivo <i>Runset</i> para LVS	43
41.	Archivos de salida de LVS desde <i>Custom Compiler</i>	44
42.	Archivos necesarios 1	46
43.	Archivos necesarios 2	47
44.	Carpeta de pruebas en terminal	47
45.	Archivos de salida de LVS desde terminal	48
46.	Archivos compartidos de LVS desde <i>Custom Compiler</i>	49
47.	Archivos compartidos de LVS desde terminal	49
48.	Error en LVS mediante terminal	52
49.	Error en archivo <i>Runset</i> 1	52
50.	Error en archivo <i>Runset</i> 1	53
51.	Error en archivo <i>Runset</i> 1	54
52.	Error en archivo <i>Runset</i> 1	55
53.	Error en archivo <i>Runset</i> 1	56
54.	<i>Layout Clean</i> desde terminal	57
55.	<i>LVS Pass</i> desde terminal	58
56.	Resultado completo de LVS desde terminal	58
57.	<i>LVS Pass</i> desde <i>Custom Compiler</i>	59
58.	Resultado de LVS desde <i>Custom Compiler</i>	59
59.	LVS de compuerta <i>NOT</i> a través de terminal	60
60.	LVS de compuerta <i>NOT</i> a través de <i>Custom Compiler</i>	60
61.	LVS de compuerta <i>NOT</i> a través de terminal	61
62.	LVS de compuerta <i>NOT</i> a través de <i>Custom Compiler</i>	61
63.	LVS de <i>Full Adder</i> a través de terminal	62
64.	LVS de <i>Full Adder</i> a través de <i>Custom Compiler</i>	62
65.	LVS de <i>ALU</i> a través de terminal	63
66.	LVS de <i>ALU</i> a través de <i>Custom Compiler</i>	63
67.	LVS de <i>Counter</i> de 4 bits a través de terminal	64
68.	LVS de <i>Counter</i> de 4 bits a través de <i>Custom Compiler</i>	64
69.	LVS de memoria <i>RAM</i> a través de terminal	65
70.	LVS de memoria <i>RAM</i> a través de <i>Custom Compiler</i>	65
71.	Documentos de salida generados en Síntesis Lógica	68
72.	Documentos de salida generados en Síntesis Física	68
73.	Archivos de ICV y GDS generados	69
74.	Archivo de <i>Runset</i> modificado	69
75.	Archivos de salida al ejecutar LVS en terminal	70
76.	Archivos de salida al ejecutar LVS en <i>Custom Compiler</i>	70

77.	LVS de memoria <i>RAM</i> a través de <i>Custom Compiler</i>	71
78.	Ubicaciones para guardar archivos generados por LVS	71
79.	Resultado de LVS a través de Terminal	72
80.	Resultado de LVS a través de <i>Custom Compiler</i>	72

Lista de cuadros

1. Ubicación de archivos importantes 34
2. Lista de comandos 45

Este trabajo desarrolla el proceso de ejecución para el módulo de verificación física, *Layout Versus Schematic* (LVS), en el proceso de diseño de un circuito a nano escala. Tiene como finalidad, definir el flujo correcto y el método más eficiente para la ejecución del LVS, tomando en consideración las actualizaciones que tuvieron los programas utilizados en este proceso.

Para poder realizar esta verificación física se hizo una revisión de la literatura actualizada, ya que en años anteriores se había definido el proceso adecuado con las herramientas vigentes en ese momento, tales como *Custom Compiler*, *ICC*, *IC Validator*, entre otras. Sin embargo, estas herramientas se actualizaron y los procesos que antes se utilizaban deben modificarse para obtener los resultados esperados y continuar con las fases de diseño.

Siguiendo las guías de instrucciones proveídas en investigaciones anteriores y las *User Guides* de TSMC paso a paso, se encontraron distintos retos, principalmente en los tipos de archivos compatibles con las nuevas versiones de los programas utilizados. A pesar de esto, se logró modificar los archivos necesarios para obtener los resultados esperados en esta etapa de diseño y poder continuar con el flujo del proyecto.

El proceso que se mostrará en este documento es una explicación a detalle de cómo se debe ejecutar el LVS para cualquier circuito. Esto parte de la participación de la síntesis física y síntesis lógica y los archivos de salida que estos generan. Se explica el proceso para obtener los archivos necesarios para la ejecución de LVS, las complicaciones con los mismos y la manera de solucionarlo. Se mostrará el proceso realizado en circuitos básicos como las compuertas *NOT* y *XOR*, en circuitos con complejidad media, como un *Full Adder* y una *ALU*, hasta circuitos más completos como lo son un *Counter* de 4 bits y una memoria *RAM*.

Finalmente, se analizarán los resultados obtenidos como archivos de salida al ejecutar el LVS, los desafíos que se encontraron en todas las fases de este proceso, la manera de como resolverlos hasta lograr los resultados esperados.

This research develops the execution process for the physical verification module *Layout Versus Schematic* (LVS), in the design process of a nano scale circuit. The purpose of this work, is to define the correct flow and the most efficient method for the execution of the LVS, considering the updates on the programs used in this process.

In order to perform this physical verification, the recent literature and sources were reviewed, since in past researches the proper flow was defined according to the programs used at the time, such as *Custom Compiler*, *ICC*, *IC Validator*, among others. Nevertheless, these tools were updated, and the processes that were being used then, must be modified in order to obtain the expected outcomes and proceed to the next design phases.

Following the instruction guides provided in previous researches and the *User Guides* of TSMC step by step, many challenges were encountered, mainly regarding the type of files compatible with the newer versions of the utilized programs. Despite this, the modification of the files was a success, and the expected results in this design stage were obtained in order to advance with the project flow.

The process that will be shown in this document is a detailed explanation of how the LVS should be executed for any circuit. This starts from the participation of physical synthesis and logical synthesis and the output files that they generate. It explains the process to obtain the files necessary for the execution of LVS, the complications with them and the way to solve it. The process carried out in basic circuits such as *NOT* and *XOR* gates, in circuits with medium complexity, such as a *Full Adder* and an *ALU*, up to more complete circuits such as a 4-bit *Counter* and *RAM* memory will be shown.

Finally, the results obtained as output files will be analyzed when executing the LVS, the challenges that were found in all the phases of this process, the way to solve them until the expected results are achieved.

El flujo de diseño de un circuito a nano escala contiene distintas etapas, cada una dedicada a un área específica para asegurar la integridad y el funcionamiento adecuado al obtener el producto terminado. Dentro de estas etapas se encuentra el LVS, el cual, tiene como finalidad, verificar la funcionalidad del diseño. Esto parte de una comparación entre las dos fases que componen esta etapa. La primera es la extracción del *netlist* tanto del *layout* como del esquemático y la segunda es la comparación de estos archivos o *netlists*. Sin embargo, este proceso tiene diferentes maneras para llevarse a cabo.

Para el desarrollo del *Layout Versus Schematic* (LVS) se utilizan distintos programas y herramientas, dependiendo del método implementado, siendo estos *IC Validator*, *NetTran*, *Black Box LVS*, *Custom Compiler*, *VUE Tool*, entre otras. A lo largo de este documento, se explicará a detalle el proceso adecuado para realizar esta verificación, los problemas que se encontraron al migrar las versiones de los programas que se utilizaron, los resultados obtenidos para cada circuito que se analizó y las recomendaciones para que esta etapa pueda realizarse en un futuro de una manera mucho más eficiente, simple y directa para cualquier circuito que se quiera diseñar, para obtener los resultados deseados y llegar hasta su fabricación en silicio.

A pesar de estas migraciones o actualizaciones de las aplicaciones y programas utilizadas en este proceso, se logró ejecutar la prueba *LVS* de manera adecuada a través de líneas de comando desde una terminal. Aun sabiendo y tomando en cuenta que la mejor manera, o la más recomendada, para efectuar esta prueba era a través de *Custom Compiler*, se tuvo que optar por realizarlo por líneas de comando y los resultados fueron satisfactorios. Esto se debió a que al tratar de realizar la prueba desde *Custom Compiler*, el programa no se ejecutaba de manera adecuada, terminando en resultados ausentes o resultados no deseados. Esto, junto con todo el proceso detallado, se describirá en capítulos siguientes en este documento.

Desde su fundación en 1966, la Universidad del Valle de Guatemala se ha caracterizado por alcanzar la excelencia en todas las áreas de estudio y la nanoelectrónica no es una excepción. En el 2009 el ingeniero Carlos Esquit inició su carrera como director del departamento de Ingeniería Electrónica y Mecatrónica. El ingeniero Esquit, con su fascinación por la tecnología nanométrica, introdujo este campo en el departamento de Electrónica.

En el 2013 se iniciaron los cursos introductorios de VLSI, *Very Large Scale Integration* por sus siglas en inglés. Esta fue la herramienta que se utilizó al comienzo de todos los proyectos orientados a la nanoelectrónica dentro de la universidad, siempre a cargo y dirigidos por el ingeniero Esquit. Sin embargo, el alcance y oportunidades de aplicación de este sistema de diseño eran limitadas, ya que los recursos utilizados en ese entonces eran gratuitos y no permitían cubrir un área más extensa en cuanto a proyectos más complejos.

En 2014, se creó una alianza con la empresa Synopsys. Esta es una empresa con más de 30 años de experiencia en el área de los nano chips, en su diseño y fabricación. Pasan del diseño en las herramientas, a la fabricación de estos circuitos en silicio para poder llegar a cubrir todas las necesidades que presente el mercado actual. Con esta alianza, nuevas puertas a proyectos mucho más ambiciosos se abrieron y el departamento no dudó para comenzar a promoverlos. Este mismo año se realizó el primer diseño en silicio en la escala nanométrica a cargo de Jonathan de los Santos [1], actual asesor del proyecto de diseño y fabricación del nanochip.

En el 2016, se readecuaron los cursos del pénsum de Ingeniería Electrónica y se realizó un cambio del curso de introducción a VLSI por dos cursos de nanoelectrónica (1 y 2), cubriendo así una parte mucho más extensa del campo y profundizando en técnicas que ayudarían al desarrollo del proyecto de fabricación de un nanochip. Continuando con los proyectos impulsados por el departamento y por el ingeniero Carlos Esquit, en los siguientes años se trabajó en varias fases del proceso de diseño como parte de trabajos de graduación, entre estos vale mencionar los aportes de los ingenieros Steven Rubio, Luis Nájera y Ricardo Girón [2], [3] y [4] respectivamente. Este trabajo de graduación hará constantes referencias al trabajo citado en [4], ya que ocupó en gran parte las etapas que se pretenden perfeccionar.

Dentro de los alcances en [4] vale la pena mencionar todos los archivos que se lograron generar durante el proceso de LVS. En este trabajo se quiso obtener los *netlist* a nivel compuerta mediante la síntesis lógica y los *netlist* de esquemático mediante el *NetTran Tool*. Todo esto se llevó a cabo de tres diferentes métodos de LVS, mediante comandos, *VUE Tool* y *Custom Compiler*. Todos los archivos generados tenían que cumplir las reglas de diseño avaladas por TSMC *Taiwan Semiconductor Manufacturing Company* por sus siglas en inglés. TSMC es la empresa líder de producción y fabricación de semiconductores en el mundo. Además, es quien define las reglas de diseño para la fabricación de los circuitos a nanoescala. Es por esto, que en el proceso de diseño se utilizan las librerías brindadas por ellos, con el fin de cumplir con las reglas ya mencionadas y tener un proceso más ágil.

De la misma manera en que la Universidad del Valle y su departamento de ingeniería electrónica y mecatrónica buscan alcanzar nuevas metas en la tecnología y ser competitivos en el mercado actual de la nanoelectrónica, se crean procesos mucho más complejos en el diseño y la fabricación de estos componentes. Siendo así, una tarea más demandante y con exigencias más rigurosas de como pudo haber sido unos años atrás. Es por esto que durante el proceso de diseño de un chip en escala nanométrica se deben cumplir ciertas directrices establecidas por las empresas que los fabrican.

Dentro del proceso de fabricación, el chip debe pasar por un Flujo de Diseño. Este es un filtro utilizado para asegurar el cumplimiento de varias directrices establecidas por las empresas fabricantes. Empresas reconocidas en el campo de la nanoelectrónica tales como Apple o Intel, frecuentemente delegan ciertas etapas de este Flujo de Diseño, con el fin de tener a las personas más expertas y capacitadas para cada área específica. De esta manera, se aseguran que el resultado de diseño será un circuito sin errores y listo para avanzar en el proceso de fabricación.

Dentro de las áreas mencionadas anteriormente se encuentra la verificación física del flujo de diseño *Layout Versus Schematic*, por sus siglas en inglés LVS. La importancia de esta etapa recae en que es el punto crítico en donde ambas fases, *Layout* y *Schematic*, compueban estar en perfecta sincronía. Sin esta etapa no se podría tener la certeza que la lógica que debe cumplir el circuito diseñado, coincida con la estructura física del mismo. Este trabajo pretende documentar el proceso correcto y más eficiente para realizar el LVS y con esto avanzar en las demás etapas del flujo de diseño hasta llegar a su fabricación. Esta será una etapa fundamental para completar el diseño y conseguir la fabricación del primer chip en escala nanométrica desarrollado por una universidad en Centro América.

Objetivo general

Realizar la prueba Layout Vs. Schematic dentro del flujo de diseño establecido y mantener una comunicación eficiente dentro del grupo de trabajo para resolver de manera oportuna los errores encontrados, basado en información técnica proveída por TSMC y la obtenida en los resultados de la prueba LVS con el fin de informar a los demás grupos en dónde se encuentran los errores y sus causas específicas.

Objetivos específicos

- Definir el método más eficiente para ejecutar el LVS de manera automatizada.
- Documentar el proceso para ejecutar el LVS de un circuito cualquiera.
- Proveer los archivos necesarios al equipo encargado de la realización de extracción de parásitos para lograr esta tarea con éxito.
- Proveer toda la información técnica necesaria al equipo de automatización del flujo de diseño para que la etapa de LVS pueda ser automatizada mediante scripts.

Este trabajo cubre todo el proceso de *Layout Versus Schematic* (LVS), partiendo de los archivos de salida que se obtienen después de la síntesis física y la síntesis lógica hasta la ejecución correcta de este proceso, en el cual se obtiene una comparación limpia y sin errores en ambas etapas, tanto en los documentos del *layout* como en los documentos del *schematic*.

Adicionalmente, comprobará que los archivos generados en la síntesis lógica y síntesis física sean los correctos para el flujo de LVS, siendo estos los siguientes:

1. Archivo con formato *.v*
2. Archivo con formato *.fdc*
3. Archivo con formato *.sdc*
4. Librerías con formato *.ndm*

De la misma manera, se asegurará, a través de los archivos de salida de LVS, que la prueba se ejecutó sin errores o *CLEAN* y que pasa los requerimientos de la misma. Para poder trasladar los archivos necesarios para las etapas siguientes en el flujo de diseño.

Finalmente, se presentará una guía paso a paso para poder replicar este proceso para cualquier circuito a diseñar, con las recomendaciones necesarias para tener un proceso mucho más eficiente y rápido; asegurando, por medio de imágenes, que se obtengan los resultados deseados en cada etapa.

El proceso de diseño y fabricación de un chip a escala nanométrica contiene lineamientos específicos para asegurar que el circuito se comportará como debe y no tendrá errores ni en su estructura ni en los resultados al momento de operación. Para llegar a esta efectividad se debe seguir un proceso riguroso con etapas muy meticulosas en las cuales, las empresas más especializadas en diseño de circuitos dedican a sus mejores expertos. Este proceso se denomina *Design Flow* y está dividido en dos partes: *Front End* y *Back End* [4] y [5].

Como se puede observar en la Figura #1, el *Front End* está compuesto por las etapas a un *behavioral level*, o nivel de funcionalidad, mientras que el *Back End* está compuesto por etapas a un *structural level*, o nivel estructural o físico. Ambas etapas son críticas en el proceso de diseño, ya que si una falla, el circuito no podrá ser fabricado. Se debe tomar en cuenta que este proceso cubre desde las especificaciones que debe cumplir el circuito, en cuanto a su funcionalidad o el propósito que debe satisfacer, hasta la fabricación del mismo para poder integrarlo en una aplicación real. A continuación se explicará cada una de las etapas de este proceso con más detalle.

6.1. Etapas del flujo de diseño

6.1.1. *Product Requirement*

Esta es la etapa previa al *Front End*, en esta se definen los requerimientos de funcionalidad para el circuito, es decir las funciones básicas que este debe cumplir.

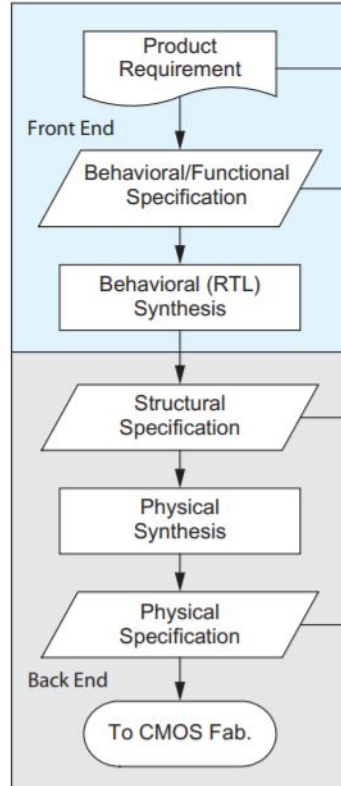


Figura 1: Flow Design

6.1.2. *Behavioral/Functional Specification*

En esta, se deben definir los objetivos y el alcance que tendrá el chip, la velocidad, *performance*, tecnología que se usará para su diseño y específicamente la arquitectura del mismo.

6.1.3. *Behavioral (RTL) synthesis*

Register Transfer Level, por sus siglas en inglés. Esta es la última etapa al *behavioral level* y en ella se elabora el circuito en verilog, optimizando la lógica para obtener mejores resultados de área y velocidades de operación, se traduce a un *netlist* utilizando *Design Vision* o *Design Compiler* y finalmente se verifica la funcionalidad con la ayuda de herramientas como *Formality*. En este punto se debe verificar que los resultados coincidan con las especificaciones de funcionalidad descritas al inicio [6]. En este punto se podrían utilizar diferentes lenguajes de descripción de *hardware* o HDLs por sus siglas en inglés. Sin embargo, los dos más utilizados son Verilog y VHDL [7].

6.1.4. *Structural Specification*

Esta es la primera etapa del *Back End*. Aquí, al igual que en el inicio del diseño, se deben definir parámetros específicos, pero esta vez orientados a detalles físicos del chip. Estas especificaciones persiguen cumplir lo que se traduzca de los RTL a los *netlists* obtenidos en la etapa anterior. Esta etapa está compuesta por varias sub fases: *Partitioning*, *Floorplanning*, *Routing*, *Placement*, *Clock network synthesis*, *Global routing*, *Detailed routing*, y *Timing closure* [7] como puede verse en la Figura #2.

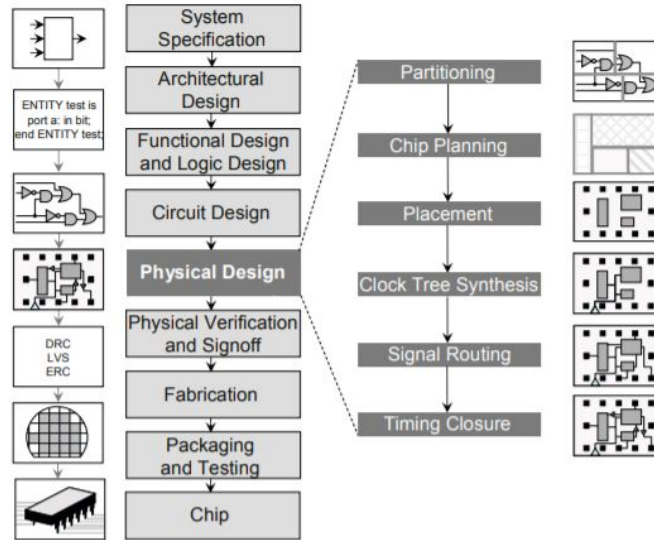


Figura 2: Pasos de VLSI

6.1.5. *Physical Synthesis*

Dentro de la síntesis física se inicia el proceso de manufactura del circuito como tal. Sin embargo esto empieza con la propia creación del *Layout*. Este, tiene que editarse para cumplir con los requisitos y lineamientos de fabricación. Para evitar errores se utilizan verificaciones internas como *Design Rule Check (DRC)*, *Parasitic Extraction*, *Antenna Rule Check*, *Electrical Rule Check* y *Layout vs. Schematic (LVS)*.

6.1.6. *Physical Specification*

En esta etapa solo se definen los parámetros que el circuito debe de cumplir al momento de ser fabricado. Estos parámetros son trasladados a la empresa responsable y encargada de la fabricación del circuito para que ellos velen por el cumplimiento de los mismos.

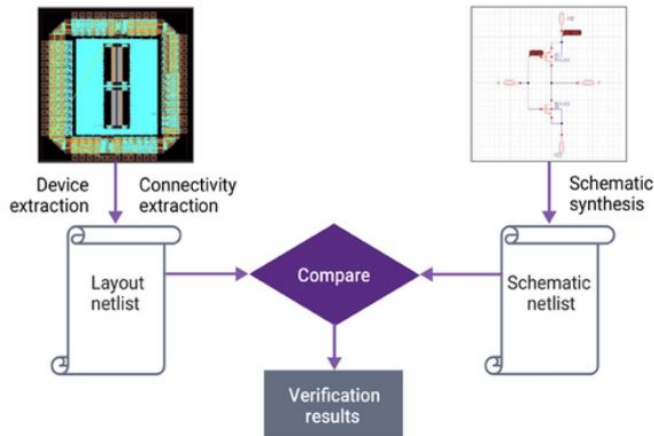


Figura 3: LVS basics

6.2. *Layout vs. Schematic (LVS)*

El LVS verifica la funcionalidad del diseño. Del *layout*, se derivan los *netlist* y son comparados con los *netlist* originales producidos de la síntesis lógica o RTL [7]. Al momento de realizar la comparación entre ambos *netlists* se debe comprobar que todas las nets y dispositivos del esquemático coincidan con los del *layout*. El proceso de LVS consta de dos etapas que deben realizarse en paralelo y luego en secuencia, debido a que se necesitan comparar los resultados obtenidos del *behavioral* y *structural level* en el mismo punto para luego determinar los puntos que se deben modificar para que la comparación produzca un resultado libre de errores.

El LVS se realiza en conjunto con otros procesos de verificación en la etapa de verificación o síntesis física. Este, tiene como función principal resaltar errores en el proceso de diseño que se deben corregir para poder continuar en el proceso y que finalmente el circuito o el chip que se este diseñando logre ser fabricado sin ningún problema y cumpla el proposito principal para el que fue pensado. Como se muestra en la Figura #3 y como se ha mencionado anteriormente, durante el proceso de LVS se debe realizar una comparación de *netlist*, tanto de los provenientes del diseño en silicio como del esquemático. Para poder obtener estos dos archivos se debe pasar por otros procesos anteriores y utilizar distintos programas de diseño que serán comentados próximamente.

El proceso de LVS se ejecuta con la ayuda de la herramienta IC Validator de Synopsys. Es a causa de la implementación de esta herramienta en el proceso de diseño que se debe tener ambos *netlist*, ya que es el formato utilizado por la herramienta para comparar el layout y el esquemático del circuito. En la Figura #4 se puede observar la división del flujo del LVS en el *behavioral* y *structural level*. Del lado izquierdo se observa todo el proceso definido con anterioridad como el front end, donde se definen, diseñan y prueban las especificaciones a nivel behavioral. Es en esta parte en donde se obtienen los *netlist* del esquemático del circuito. Estos, tienen que ser después traducidos a un *netlist* en formato compatible con IC Validator para poder realizar la comparación deseada, esto se lleva a cabo a través de *NetTran*.

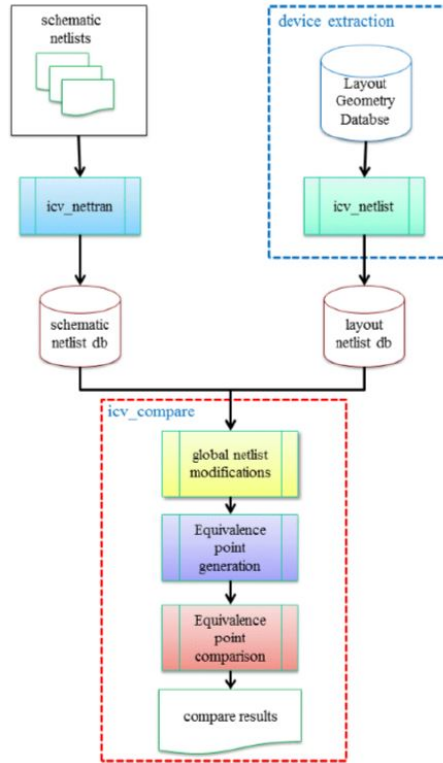


Figura 4: LVS flow

De forma paralela al proceso anterior, se extrae el *netlist* del *back end* o nivel *structural*. Este es el *netlist* correspondiente al *layout* del circuito. A diferencia de su contraparte, este *netlist* ya se encuentra en un formato compatible con IC Validator por lo que no se tiene que traducir. Seguido de esto, al tener ambos *netlist* en el formato deseado se parte a realizar modificaciones globales a los *netlists*. Esto con el fin de poder efectuar una comparación más eficiente, ya que habrán menos factores de diferencia entre los *netlist* y esto, a su vez, reducirá de gran manera los puntos de error que se podrían obtener si no se modificaran estas variables. Estas variables pueden deberse a que se tengan conexiones o elementos en serie o en paralelo en ambos *netlist* lo cual, al poderse simplificar, reduce la complejidad de compararlos.

La herramienta IC Validator puede también ejecutar modificaciones de manera automatizada, de este modo puede filtrar aquellos elementos que no requieran ser comparados o que su análisis no será beneficioso para presentar resultados. Adicionalmente puede combinar dispositivos que se encuentren en serie o en paralelo, buscando así ser aún más eficiente en este proceso y reduciendo la cantidad de puntos de comparación.

El siguiente paso es la generación de puntos de equivalencia. Este puede ser operado de manera automática por la herramienta, o bien ser configurado manualmente por el usuario en el *Runset*. Cabe mencionar, que si el proceso se desea hacer más eficiente, la mejor opción es definirlo manualmente, ya que se puede llegar a ser más específico y meticuloso de lo que podría tolerar la herramienta. En este proceso, se definen celdas del *layout* y del esquemático que compartan lógica similar o potencialmente similar. Luego de esto se pasa a comparar estas celdas que fueron configuradas como parejas entre cada uno de los procesos y de estos se

generan los resultados que ayudarán a definir la dirección a seguir del diseño. Los resultados de comparación del LVS contienen: 1) El resultado de la prueba, es decir si es aceptable o no (*PASS* o *FAIL*), 2) Número de celdas equivalentes aceptables y las que no lo son, 3) Mensajes de compilación y finalmente 4) Diagnósticos. Dependiendo de estos 4 puntos, se debe tomar la decisión de que etapas deben de ser revisadas nuevamente o qué puntos deben cambiarse, si los resultados de las pruebas fallan; de lo contrario, se puede seguir adelante en el proceso de diseño.

6.3. Herramientas utilizadas

Dependiendo del proceso o el método seleccionado para ejecutar el LVS, existen distintas herramientas que se tienen que implementar para poder obtener los resultados de la comparación que se mencionó en párrafos anteriores. A pesar que la herramienta mencionada con mayor frecuencia en este proceso es el IC Validator, existen otras que dependiendo el método seleccionado, forman gran parte del proceso del LVS y son de suma importancia.

6.3.1. *CDL*

Circuit Design Language por sus siglas en inglés. Este es el lenguaje utilizado en el *behavioral* level para describir el esquemático del circuito, es decir, este es el *netlist* generado directamente del esquemático. Este luego es el que se debe traducir para estar en un formato compatible con IC Validator.

6.3.2. *Netlist*

Los *netlist* son archivos que representan al circuito después de realizar la síntesis lógica. Estos pueden ser una representación en nivel *behavioral* o *structural*. Como su nombre en inglés lo indica es una lista de las nets del circuito, es decir una lista de las interconexiones del mismo.

6.3.3. NetTran

Esta herramienta se encarga de traducir el *netlist* proveniente del esquemático a un formato de *netlist* aceptable en IC Validator para que pueda ser comparado con el *netlist* del *layout*. El NetTran puede traducir los *netlist* provenientes de HSPICE o de Verilog, ya que a pesar de ser similares con el de ICV no son exactamente iguales y esto impide la comparación en LVS.

6.3.4. *GDS*

Graphic Data System por sus siglas en inglés. El GDS es un archivo que presenta información física del circuito o del *layout*. Este contiene especificaciones de las figuras geométricas que son las representaciones de los distintos componentes que integran al circuito. Este archivo es utilizado para comparar el diseño en silicio del circuito [4].

6.3.5. *Custom Compiler*

Esta herramienta de Synopsys es la encargada de la ejecución del LVS. Custom Compiler permite realizar simulaciones, análisis de diseño y análisis físicos de los circuitos. Esta, a través de una interfaz gráfica, permite visualizar los resultados de estos análisis y simulaciones para poder saber la dirección que se debe tomar en cuanto a las modificaciones necesarias en caso de tener resultados no deseados.

Para poder comprender y explicar en su totalidad el proceso de ejecución de LVS, es necesario identificar el punto donde esta etapa se encuentra en el flujo de diseño. Para ejecutar la prueba de LVS, son necesarios varios archivos que son los resultados de dos fases o etapas previas en el proceso de diseño. Estas etapas son la síntesis lógica y la síntesis física. En algunos casos, estos archivos de salida se deben modificar para que se encuentren en los formatos necesarios y compatibles con los programas y herramientas empleadas en LVS.

7.1. Archivos previos

Dentro de la síntesis lógica, la cual es la primera etapa de diseño que afecta o nutre a LVS, como se explico anteriormente, se obtienen tres archivos a su salida, como se puede observar en la Figura #5 en donde se encuentran archivos con formatos *.ddc* *.sdc* y *.v*. Este último es el más importante para la ejecución de LVS. Este documento en formato *verilog*, Figura #6 se utiliza con otros documentos para extraer una salida en formato *.icv* que son formatos de archivos compatibles con *IC Validator*.

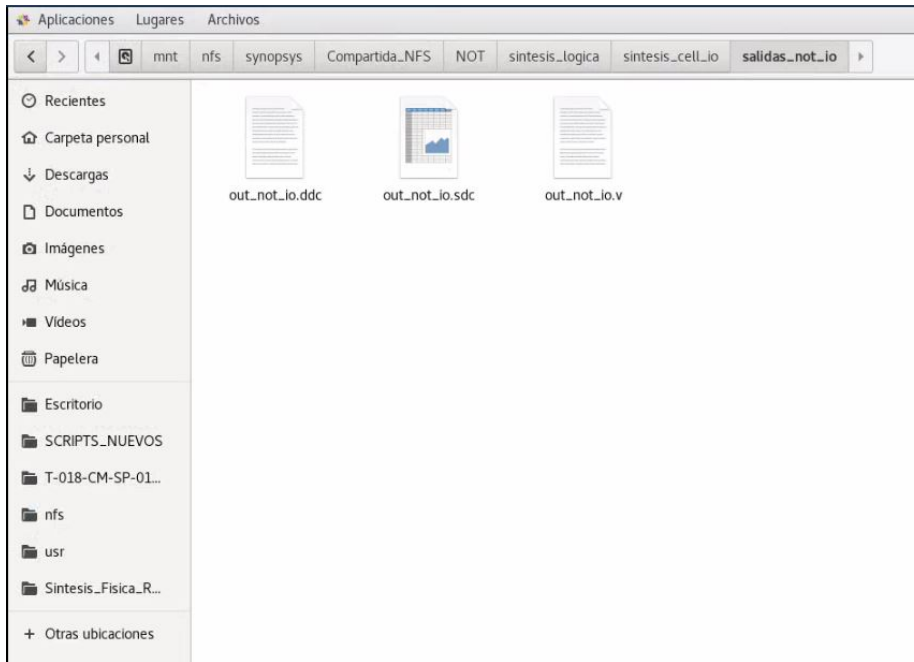


Figura 5: Salidas Síntesis Lógica

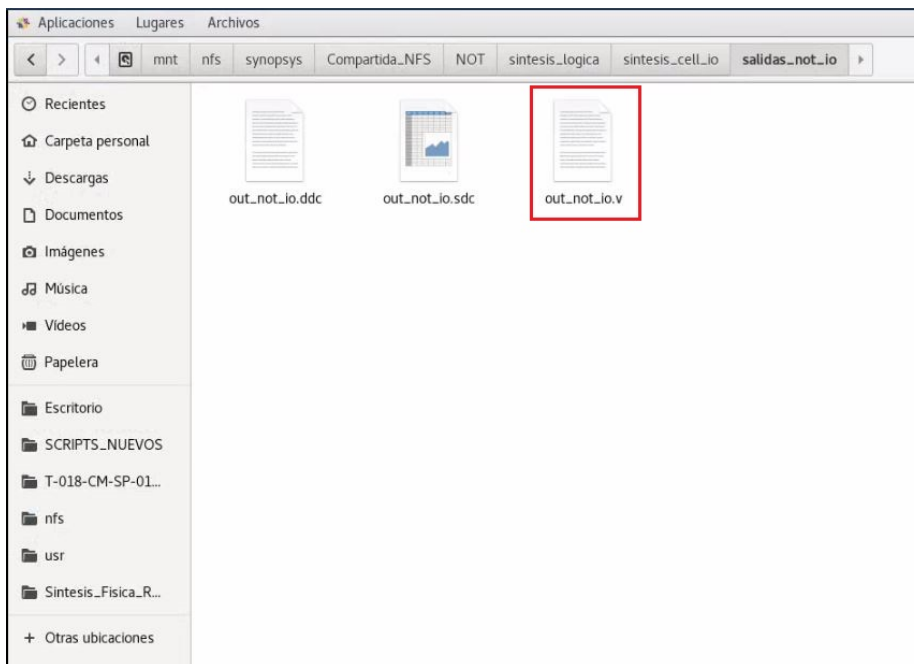


Figura 6: Archivo *Verilog* de Síntesis Lógica

Por el otro lado, los archivos generados después de la síntesis física son más variados, como se muestra en la Figura #7. Dentro de todos estos archivos únicamente se utilizará la carpeta *.ndm*, que está marcada en la Figura #8 la cual contiene librerías de *ICC2* que serán utilizadas para la extracción del archivo *GDS* y la visualización en *Custom Compiler* del *layout* del circuito a trabajar.

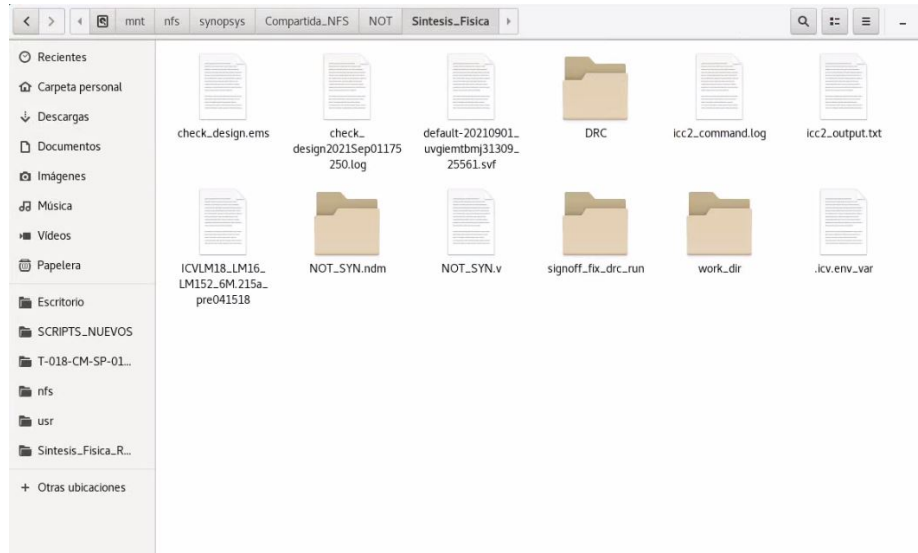


Figura 7: Salidas Síntesis Física

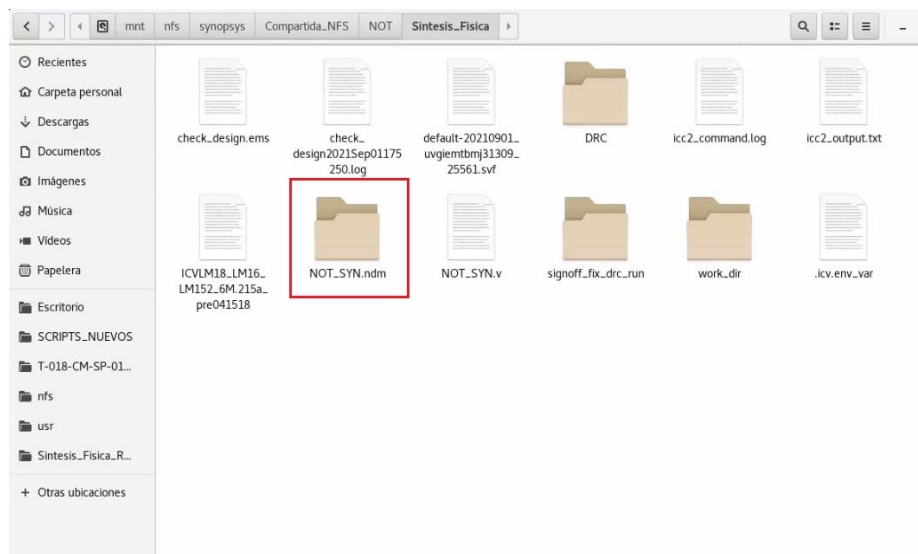


Figura 8: Carpeta *.ndm* de Síntesis Física

Los archivos y carpetas anteriormente mostradas son todas del proceso de diseño de una compuerta *NOT*; sin embargo, estos documentos y carpetas también se generan para cualquiera que sea el circuito a diseñar. Las ubicaciones de estos documentos se encuentran en una carpeta compartida entre todos los integrantes del grupo de diseño, para poder tener un acceso a los mismos de manera ágil y rápida. La ruta para estos archivos y para la carpeta compartida en general es la siguiente:

`/mnt/nfs/synopsys/compartida`

Dentro de esta ubicación se encuentran diferentes sub carpetas de los circuitos trabajados como se muestra en la Figura #9

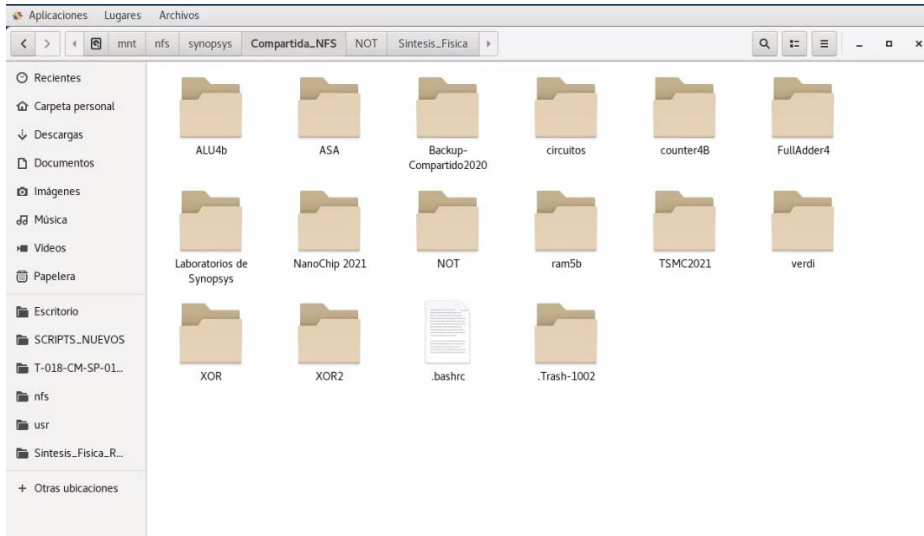


Figura 9: Carpeta de Trabajo Compartida

Luego, dentro de estas subcarpetas se encuentran otras carpetas de cada etapa de diseño: síntesis física, síntesis lógica, LVS, Antena y LPE, como se muestra en la Figura #10

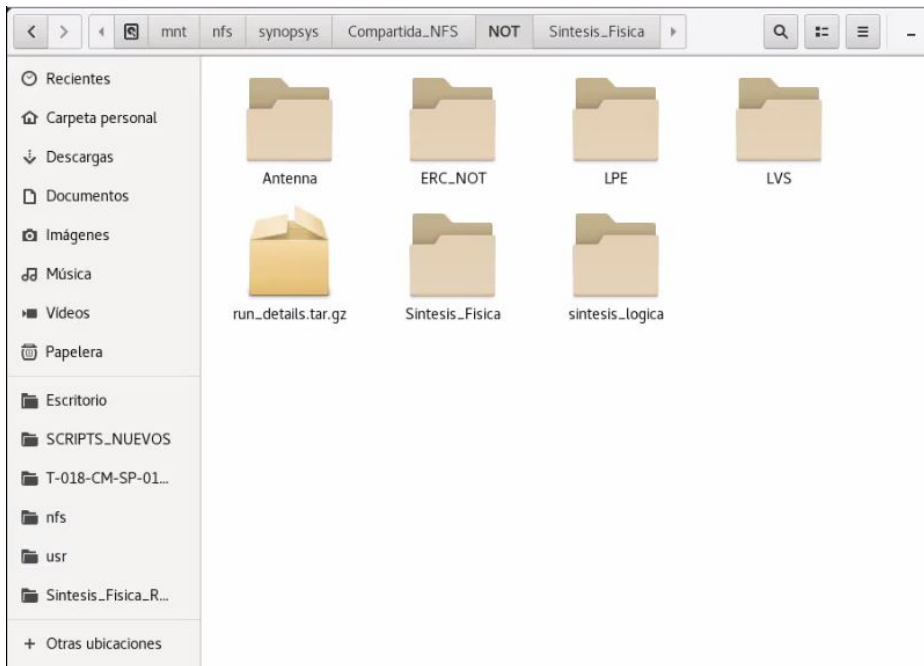


Figura 10: Carpeta de Compuerta NOT

Adicional a estos archivos y carpetas, que son generados por fases o etapas de diseño previas a LVS, existen otros archivos necesarios para la correcta ejecución de esta última etapa. Gracias a los avances y todo el trabajo realizado en [1], [3] y [4], especialmente en este último, se contaba con esta serie de archivos mencionados. Todos ellos se almacenaron en la copia de seguridad de la carpeta compartida utilizada por el grupo de diseño anterior, quien también, al igual que el equipo de diseño actual, realizó todas las etapas de la misma manera que se están realizando ahora, pero con la versión anterior de todas las herramientas. Los archivos recuperados se muestran en la Figura #11.

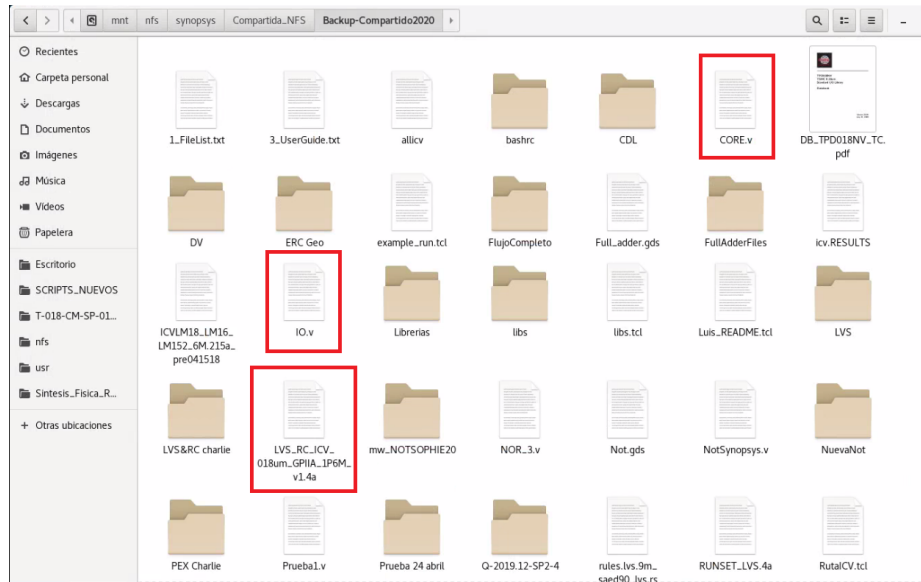


Figura 11: Documentos recuperados

Estos archivos son *CORE.v*, *IO.v* y *LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a*. Los primeros dos son archivos en formato *Verilog* y continen las librerías estándar o *core* y las librerías de entradas y salidas de los circuitos a analizar. El último, es el archivo de *Runset*, el cual fue compartido dentro de la documentación proveída por *TSMC*. En este, se deben verificar siempre los parámetros introducidos en el *environment setup* y la adición de las celdas o *black boxes* que fueron utilizadas en el diseño de cada circuito. Es decir, este archivo debe modificarse para cumplir todos los parametros de cada circuito que se desee implementar. En su mayoría de información, los distintos archivos de *Runset* contendrán lo mismo, pero debe alterarse para ser compatible con cada circuito. En capítulos siguientes se mostrarán estas diferencias. Es recomendable hacer una copia de estos archivos a una ubicación local para un mejor control y facilidad de acceso. Estos documentos se encuentran ahora en la carpeta compartida utilizada por el equipo de diseño en la siguiente dirección:

`/mnt/nfs/compartida/Backup-Compartido2020`

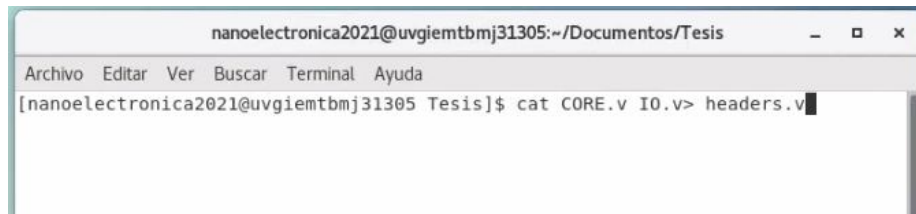
En la mayoría de imágenes se muestra una dirección distinta a la que se menciona en referencias anteriores. Esto se debe a que la carpeta compartida tuvo que ser reubicada por cuestiones de disponibilidad de espacio interno. Sin embargo, al seguir las rutas especificadas se pueden encontrar los documentos y las carpetas como se ha explicado anteriormente. Como referencia, también pueden observarse todos los documentos que se han mencionado hasta ahora con su ubicación exacta dentro de la carpeta compartida en el Cuadro #1.

7.2. Archivos generados

7.2.1. Archivo *ICV*

Los archivos en formato *verilog*, mencionados en el párrafo anterior, son utilizados en el proceso de extracción del archivo *ICV*, el cual, es uno de los archivos necesarios para la ejecución del LVS. Lo primero que se debe realizar es una concatenación de estos archivos, para obtener a la salida un solo archivo con todas las librerías en conjunto. Esto se realiza desde líneas de comandos en una terminal como se observa en la Figura #12.

El archivo de salida se nombró como `headers.v` y también está en formato *verilog* esto es importante y se debe tener en mente el nombre para poder comprender en su totalidad el proceso para la extracción del archivo *ICV*. En toda esta sección se muestra una misma dirección de donde se ejecutó la terminal, se sugiere realizar esto en una carpeta diferente por cada circuito, ya que los archivos de salida de cada comando se ubicarán en donde se haya ejecutado la terminal.



```
nanoelectronica2021@uvgiemtbnj31305:~/Documentos/Tesis
Archivo Editar Ver Buscar Terminal Ayuda
[nanoelectronica2021@uvgiemtbnj31305 Tesis]$ cat CORE.v IO.v > headers.v
```

Figura 12: Concatenación de librerías

Después de obtener este archivo, se implementó la herramienta *NetTran*, la que permitirá realizar una traducción de un archivo en formato *verilog* a uno en formato *spice* o *CDL*, ya que este es el formato compatible para extraer el archivo *ICV* para LVS. En la Figura #13 se muestra el comando para la traducción de este archivo a uno en formato *spice*, el cual se definió con el nombre *headers.sp*.



```
nanoelectronica2021@uvgiemtbnj31305:~/Documentos/Tesis
Archivo Editar Ver Buscar Terminal Ayuda
[nanoelectronica2021@uvgiemtbnj31305 Tesis]$ icv_nettran -verilog headers.v -outType SPICE -outName headers.sp
```

Figura 13: Traducción de archivos

Como último paso previo a la extracción del archivo *ICV* se debe modificar el archivo obtenido en el paso anterior, *headers.sp*. En el cual, se deben agregar las siguientes líneas de texto al inicio del archivo.

```
.GLOBAL VDD VSS VDDPST VSSPST
*.EQUATION
*.SCALE METER
*.MEGA
.PARAM
```



```
.INCLUDE /usr/synopsys_old/TSMC/SCRIPTS_NUEVOS/20191128-124344/  
source.added
```

El último archivo es brindado por TSMC. Este mismo contiene la declaración de los subcircuitos de los dispositivos como transistores, diodos, capacitores y resistencias definidos en el runset. Es sumamente importante incluirlo, al igual que lo mencionado anteriormente y tener correcta la ubicación de donde se está extrayendo. Siempre se debe asegurar que los archivos que se agregan o se hacen referencia se encuentren en la dirección donde se indican. Este proceso anterior, involucrando los archivos o *Netlist IO.v*, *CORE.v*, *headers.v*, y *headers.sp* se puede ejecutar una sola vez, ya que el archivo que se utilizará será el *headers.sp* y puede copiarse y ser utilizado en cada subcarpeta creada para cada circuito a analizar.

Finalmente, se debe ejecutar el comando para exportar el archivo en formato *ICV*, utilizando los archivos *headers.sp* y el netlist en formato *verilog* obtenido en la etapa de síntesis lógica, en este caso definido como *out_not_io.v* (Figura #6). El comando para extracción del archivo *ICV* se muestra en la Figura #14. Todos los comandos que se han utilizado hasta este punto, así como los que se utilizarán durante la ejecución de *LVS* se encuentran en Cuadro #2. Este *ICV* se definió como *CDL_netlist.icv* y será utilizado posteriormente para la ejecución de *LVS*.



```
nanoelectronica2021@uvgiemtmbmj31305:~/Documentos/Tesis  
Archivo Editar Ver Buscar Terminal Ayuda  
[nanoelectronica2021@uvgiemtmbmj31305 Tesis]$ icv_nettran -verilog out_not_io.v -sp headers.sp -outType  
ICV -outName CDL_netlist.icv
```

Figura 14: Extracción de *ICV*

7.3. Librerías a *Custom Compiler*

Tal como se utilizaron algunos archivos de salida de la síntesis lógica para la extracción de *Netlist* en formato *ICV* a través de la herramienta *NetTran* para luego implementarlo en la ejecución de *LVS*, la síntesis física también exporta ciertos archivos que son necesarios para esta ejecución. La carpeta *.ndm* que se mostró en la Figura #8, es la que se necesita para poder importar una nueva librería a *Custom Compiler* y así poder obtener todas las visualizaciones de *layouts* que se presentarán en el siguiente capítulo. A continuación se detalla el proceso de importación de estas librerías a *Custom Compiler* a través de imágenes. El ejemplo se realizó con la compuerta *XOR*.

7.3.1. Archivo *.ndm*

El primer paso es ubicar el archivo o carpeta que se exportó en la etapa de síntesis física, como se muestra en la Figura #8. Luego de haberlo ubicado, se recomienda copiarlo a una ubicación local, como en pasos anteriores, esto facilita la accesibilidad del mismo. En la Figura #15 se observan una serie de sub carpetas y documentos incluyendo la carpeta *.ndm* de la síntesis física, los cuales se utilizarán en la ejecución de *LVS*.

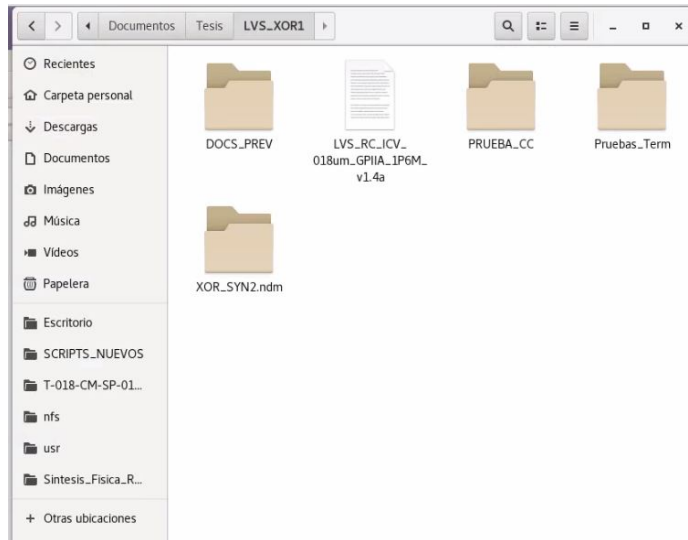


Figura 15: Carpeta local de trabajo para la compuerta *XOR*

7.3.2. Ejecución de *Custom Compiler*

El siguiente paso requiere la implementación de *Custom Compiler* para importar directamente la librería proveniente de *ICC2*. Sin embargo, para poder ejecutar la aplicación, se debe hacer a través de líneas de comando en una terminal que se ejecute en una ubicación específica. La razón de este procedimiento es que al acceder a *Custom Compiler* desde esta carpeta o ubicación, se asegura que se contarán con todas las librerías brindadas por TSMC. La carpeta desde donde se tiene que ejecutar el comando fue proporcionada por TSMC y su ubicación es la siguiente:

```
/usr/synopsys/TSMC/180/CMOS/G/I03.3V/pdk/T-018-CM-SP-018-W1_1_0A
```

Al estar en esta ubicación se debe ejecutar el siguiente comando en terminal:

```
custom_compiler
```

Una vez dentro de *Custom Compiler* se debe realizar, en el mismo orden, el procedimiento que a continuación se describe.

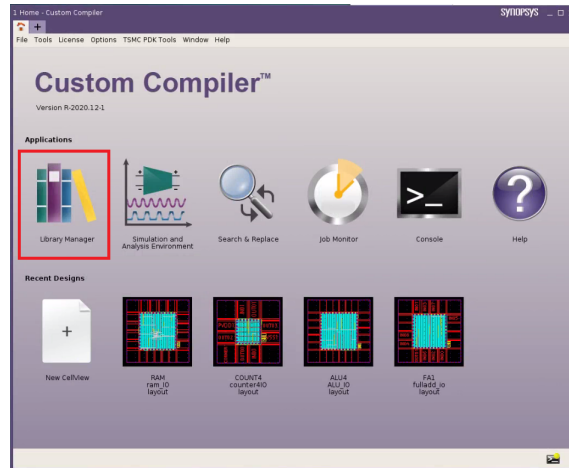


Figura 16: Inicio de *Custom Compiler*

En la Figura #16 se muestra la página de inicio de *Custom Compiler*, en rojo está resaltada la opción de *Library Manager*, se debe hacer click en esta opción para acceder a las librerías incluidas en la herramienta.

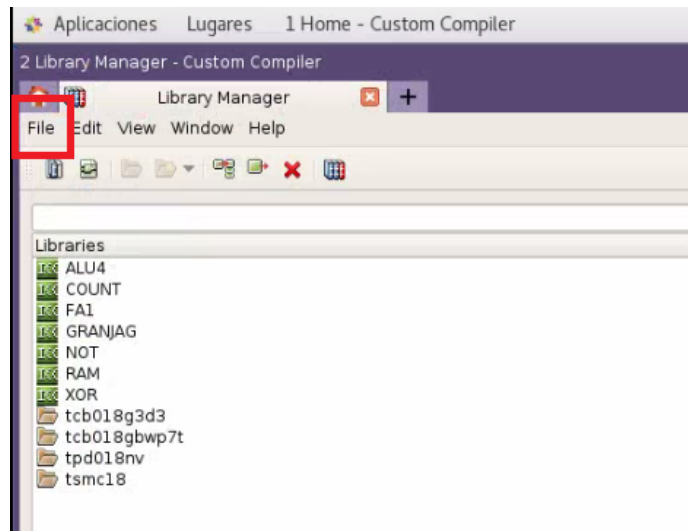


Figura 17: *Library Manager* dentro de *Custom Compiler*

Dentro del *Library Manager* se debe hacer click en el menú de *File* en la esquina superior izquierda, está marcado en rojo en la Figura #17.

Al acceder al menú de *File* se cuenta con varias opciones, pero se seleccionará en la opción de *Add ICC2 Library* tal como se muestra en la Figura #18.

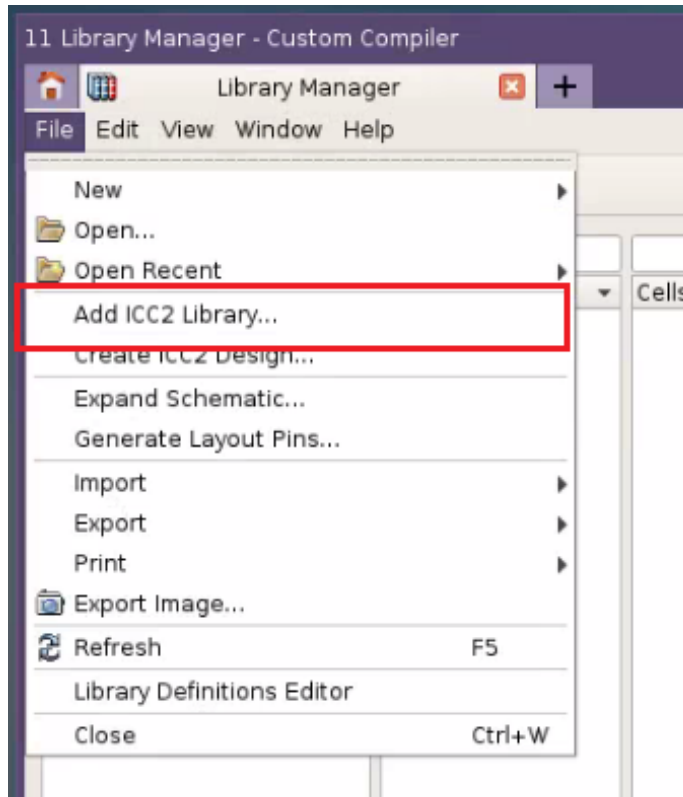


Figura 18: Menú de *File*

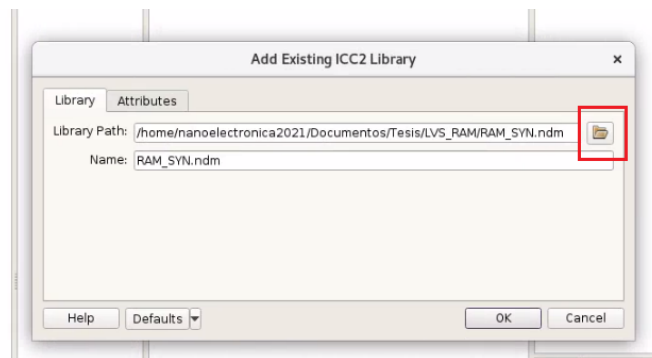


Figura 19: Opción *Add ICC2 Library*

Como se observa en la Figura #19, dentro de esta opción aparecen dos pestañas. La primera se denomina *Library*, en ella se encuentran dos recuadros, uno de *Library Path* y otro de *Name*. En la primera opción se debe indicar la ubicación exacta de la carpeta *.ndm*, la cual se copió en una ubicación local. Para poder ubicar esta dirección se puede realizar de dos maneras. La primera es haciendo click en el ícono marcado en rojo en la Figura #19, al lado del primer recuadro. Esto permitirá navegar por el buscador de archivos hasta ubicar el deseado. La segunda opción es ubicando la carpeta localmente y hacer click derecho sobre ella y luego hacer click en la opción copiar, como se muestra en la Figura #20, después se debe hacer click derecho sobre el recuadro de *Library Path* y hacer click en pegar, o bien presionar **Ctrl+v** para pegar la ubicación de la carpeta.

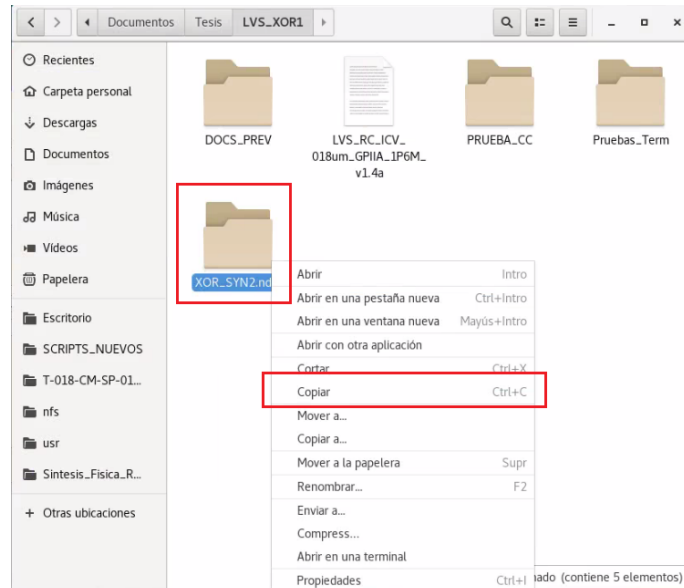


Figura 20: Copia de carpeta .ndm

En el recuadro de *Name* que se muestra resaltado en la Figura #21, se debe definir el nombre que se desea para la nueva librería que se está importando. En este caso, se define como XOR, para poder tener una referencia amigable y sencilla al momento de buscarla en el *Library Manager*.

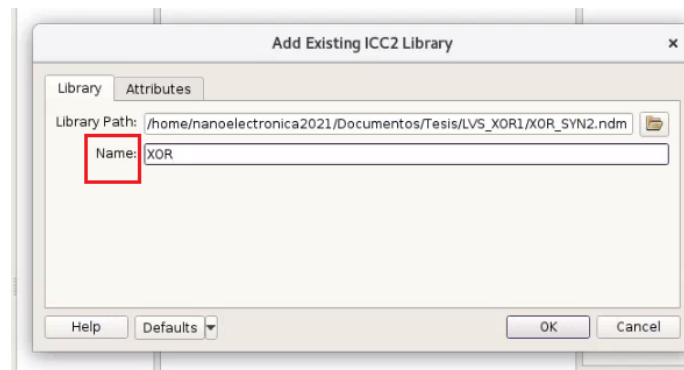


Figura 21: Nombre de nueva librería

Finalmente se debe revisar el apartado de *Attributes* dentro de esta configuración. En la Figura #22 se muestra que en el apartado de *OA Tech Library* se especifica una librería ya existente en los archivos de *Library Manager*, como se resalta en rojo. Esta es la razón por la cual se ejecutó el comando para iniciar *Custom Compiler* desde una carpeta específica, ya que se utilizarán las referencias de esta librería que se encuentra dentro de esta carpeta, para poder importar las futuras librerías de los circuitos trabajados. Al asegurar que esta configuración es correcta se presiona OK y se importará la librería deseada.

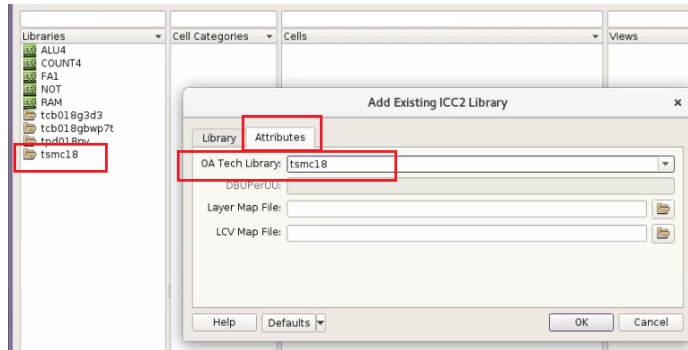


Figura 22: Apartado *Attributes*

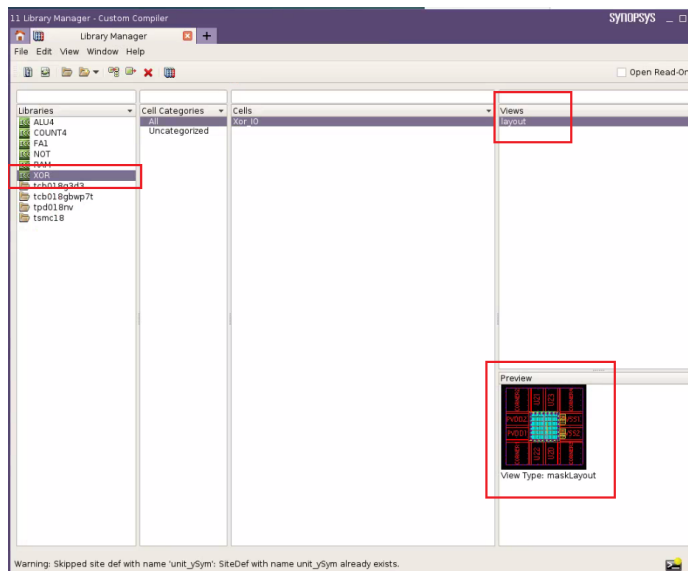


Figura 23: Librería agregada a *Library Manager*

Una vez completado este proceso se puede observar, como se muestra en la Figura #23, a la nueva librería *XOR* en el apartado de *Libraries* de lado izquierdo, y su vista de *layout* en el apartado de *Views* de lado derecho. Este proceso se debe realizar por cada circuito a trabajar para poder obtener los *layouts* que se mostraron en la sección anterior.

7.4. Archivo *GDS*

Parte de las responsabilidades del grupo encargado de la síntesis física es exportar el archivo en formato *.gds*, ya que este es utilizado también en el proceso de ejecución de LVS desde la terminal a través de líneas de comando. Sin embargo, para enriquecer este trabajo se definió el proceso para exportar este archivo, ya que se encontraron ciertos desafíos y problemas en cuanto a compatibilidad con el *GDS* exportado por el equipo previo. La razón de estos problemas de compatibilidad es la migración o actualización de las herramientas utilizadas en esta ocasión, ya que a diferencia del proyecto realizado el año pasado, la exportación se hizo desde *IC Compiler* y ahora se utilizó *IC Compiler II*. Esto no fue un problema

para el equipo de trabajo, ya que se logró obtener el archivo adecuado a través de *Custom Compiler*.

Luego de haber importado las librerías en *Custom Compiler* como se definió en la sección anterior, se vuelve a utilizar el menú de *File* como se muestra en la Figura #18, pero en esta ocasión se hará click en la opción *Export*, seguido de la opción *Stream...* como se ve en la Figura #24.

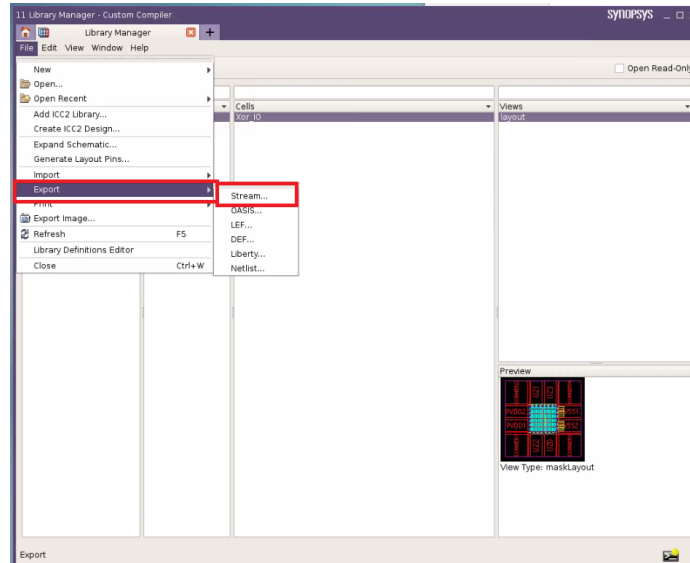


Figura 24: Opción *Export Stream*

Una vez en la opción *Export Stream* se debe configurar ciertos parámetros, que se resaltan con rojo en la Figura #25. En el apartado de *Main* en la esquina superior izquierda del configurador, se debe revisar y modificar los parámetros en las secciones de *Input* y *Output*. Para la primera sección se debe especificar la librería que se está trabajando, con el nombre que se haya definido en el proceso de importar la librería a *Custom Compiler* (revisar Figura #21), luego se debe especificar el nombre de la *Top Cell*, el nombre de esta aparece en *Custom Compiler* en la columna *Cell*, y por último asegurarse que en *View* este definido *layout* que es el tipo de vista disponible de la librería. Luego de esto, se debe dirigir el archivo de salida a la ruta que se desee en la opción de *Run Directory* en la sección de *Output*, al mismo tiempo definir el nombre del archivo que se requiere a la salida en la opción *Stream File* de la misma sección. Este nombre debe llevar *.gds* al final, para guardarlo en este formato. Finalmente se hace click en *OK* y se obtiene el archivo *GDS* en la carpeta que se especificó. Este proceso se debe hacer para cada circuito a trabajar.

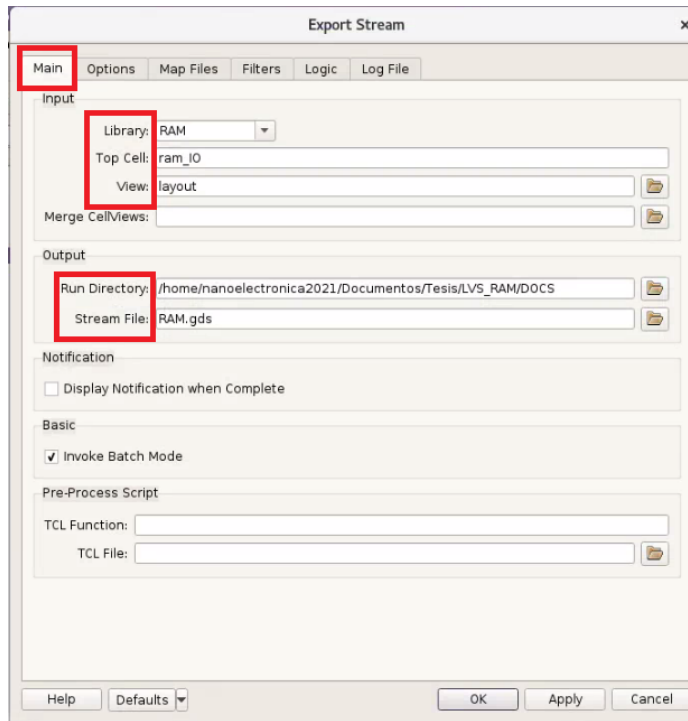


Figura 25: Configuración *Export Stream*

7.5. Archivo *Runset*

El último archivo que se debe revisar y modificar dependiendo de que circuito se este trabajando es el *Runset*. Este archivo es proporcionado por TSMC, sin embargo este debe ser modificado para cada circuito. A pesar que las modificaciones y ediciones son mínimas, se debe ser cuidadoso ya que de no contar con el *Runset* correcto, el LVS no se ejecutará de manera adecuada. Como primer paso de modificación se debe verificar el *environment setup*. Como se muestra en la Figura #26, se deben modificar varios parámetros, se explicará cada recuadro marcado en rojo en orden descendente. En el segundo recuadro se debe insertar la dirección completa del archivo *GDS* generado en la sección anterior, seguido del nombre de la celda utilizada, esta será la que se definió en el mismo proceso de extracción del *GDS* como *Top Cell* y, por último, se especifica el formato del archivo, en este caso *GDSII*. En el tercer recuadro se define nuevamente el nombre de la *Top Cell* del esquemático. En el cuarto recuadro se inserta la dirección exacta del archivo *ICV* que se generó, al igual que el *CDL* de *SPICE* que se encuentra en la carpeta compartida de los archivos recuperados del proyecto anterior. Verificar si el archivo se encuentra en esta ubicación para estar seguro. Por último, en el quinto recuadro, se definen los archivos que se generarán a la salida del LVS. Esto es sumamente importante debido a que estos son los archivos que se trasladarán al equipo de extracción de parásitos o *LPE*.


```

LVS_RC_ICV_018um_GPIIA_IP6M_v1.4a
#error This PXL runset was generated to run with ICV version 2015.12 and newer.
#endif
#endif

////////////////////////////////////
// ENVIRONMENT SETUP //
////////////////////////////////////

library(
    library_name = "/home/nanoelectronica2021/Documentos/Tesis/LVS_RAM/DOCS/RAM.gds",
    cell = "ram_I0",
    format = GDSII
);

SCHEMATIC_TOPCELL : string = "ram_I0"; // Set schematic top cell name here
sch_db = schematic(
    schematic_file = {"~/home/nanoelectronica2021/Documentos/Tesis/LVS_RAM/DOCS/RAM.icv", ICV},
    schematic_library_file = {"~/usr/synopsys-01d/TSMC/SCRIPTS_NUEVOS/20191128-124344/unit.cdl", SPICE}),
    expand_multiple_devices = true,
    spice_settings = {slash_is_space = false}
);

//define USER EQUIV FILE // Turn on for user-specified equivalent point file flow
#ifdef USER EQUIV FILE
#include ".user.equiv" // Set equivalent point file here
#endif

/* EDIT: The following section contains all of the runset variables for RC extraction tools. */
#define RC_DECK // Turn on for LPE/RC extraction
#define CROSS_REFERENCE // Turn on for source cross reference in LPE extraction
#define ZERO_NRS_NRD // Turn off when this deck would calculate NRS and NRD
#define FILTER_DGS_TIED_MOS // Turn on to filter MOS with D, G and S tied together (default filter MOS with all pins tied)
#define WELL_TO_PG_CHECK // Default is on. Turn on to highlight if mwell connects to ground or psub connects to power.
#define GATE_TO_PG_CHECK // Default is off. Turn on to highlight if a mos gate directly connects to power or ground.
#define PATH_CHECK // Default is off. Turn on to highlight if
// (1) nodes have a path to power but no path to ground
// (2) nodes have a path to ground but no path to power
// (3) nodes have no path to power or ground
// (4) nodes have no path to any label net
#define DS_TO_PG_CHECK // Default is on. Turn on to highlight if drain connects to power and source connects to ground.
#define FLOATING_GATE_CHECK // Default is on. Turn on to highlight if there are floating gates.
#define FLOATING_WELL_CHECK // Default is on. Turn on to highlight if well does not connect to power or ground.
// The mwell of moscaps and mwell-resistor are excluded
#define HW_RING // Turn on to enable HW ring to separate the node from BULK

```

Figura 26: Environment Setup en Runset

Por último, se debe revisar el apartado *ICV OPTIONS*, en este, se encuentran todas las *black boxes* o cajas negras de las celdas utilizadas en la fabricación y diseño de cada circuito. Estas varían de circuito en circuito, y a medida que la complejidad del circuito aumenta, así también la cantidad de *black boxes* que se deben incluir, debido a que se utilizan distintos circuitos y compuertas para implementar circuitos más complejos. En la Figura #27 se observan algunos ejemplos del formato en el que se deben agregar las *black boxes* necesarias en cada circuito. Esto permite visualizar estas celdas como una caja negra, en la cual no se sabe lo que ocurre adentro y solo se especifican las entradas y salidas de la misma.

```

);

lvs_black_box_options(
    equiv_cells = {{schematic_cell = "CKX0R2D4BWP7T", layout_cell = "CKX0R2D4BWP7T"}},
    remove_schematic_ports = {"A1", "A2", "Z"}
);

lvs_black_box_options(|
    equiv_cells = {{schematic_cell = "A022D2BWP7T", layout_cell = "A022D2BWP7T"}},
    remove_schematic_ports = {"A1", "A2", "B1", "B2", "Z"}
);

lvs_black_box_options(
    equiv_cells = {{schematic_cell = "TIELBWP7T", layout_cell = "TIELBWP7T"}},
    remove_schematic_ports = {"ZN"}
);

lvs_black_box_options(
    equiv_cells = {{schematic_cell = "PVDD1CDG", layout_cell = "PVDD1CDG"}},
    remove_schematic_ports = {"VDD"}
);

lvs_black_box_options(
    equiv_cells = {{schematic_cell = "PVSS1CDG", layout_cell = "PVSS1CDG"}},
    remove_schematic_ports = {"VSS"}
);

```

Figura 27: Formato de black boxes en Runset

En el Cuadro #1 se encuentra una lista de los diferentes archivos que se han mencionado con anterioridad y su ubicación exacta. Esto se realizó con la intención que para cualquier proyecto futuro, encontrar estos archivos sea algo mucho más sencillo y no se pierda tiempo buscandolos. Sin importar si los archivos se recuperaron del equipo de diseño del año anterior, fueron proveídos por TSMC, o bien fueron generados en los pasos antes descritos, aquí se encuentran todas las ubicaciones.

Archivo	Fuente	Ubicación
CORE.v	Documentación de equipo anterior	/mnt/nfs/compartida/Backup-Compartido2020/CORE.v
IO.v	Documentación de equipo anterior	/mnt/nfs/compartida/Backup-Compartido2020/IO.v
Runset	Documentación de equipo anterior	/mnt/nfs/compartida/Backup-Compartido2020/LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a
Verilog	Archivo generado en síntesis lógica	/mnt/nfs/compartida/NOT/sintesis_logica/sintesis_cell_io/salidas_not_io/out_not_io.v
Carpeta .ndm	Archivo generado en síntesis física	/mnt/nfs/compartida/NOT/Sintesis_Fisica/NOT_SYN.ndm
ICV	Archivo generado en proceso de LVS	/mnt/nfs/compartida/NOT/LVS/NOT.icv
GDS	Archivo generado en proceso de LVS	/mnt/nfs/compartida/NOT/LVS/Not.gds

Cuadro 1: Ubicación de archivos importantes

Para el archivo de *Runset* se debe tener en cuenta que la ubicación que se define es donde se encuentra el archivo *Runset* general, sin embargo, como se mencionó en secciones anteriores, se debe modificar para cada circuito a trabajar. Adicionalmente, los archivos de Verilog, ICV, GDS, y la carpeta .ndm que se definen en este cuadro, son específicamente para el caso de la compuerta *NOT*. Para los otros circuitos se deben modificar las direcciones, buscando en las carpetas compartidas de cada compuerta o circuito trabajado. A pesar de tener las direcciones, se recomienda utilizarlas solo como guías, no como la dirección precisa, ya que estas pueden variar, por lo que siempre se recomienda revisar la ubicación exacta de los archivos aquí mencionados.

Circuitos desarrollados

Con el fin de poder realizar el LVS de manera adecuada se definieron distintos circuitos para analizar y desarrollarlos en todas las etapas del flujo de diseño. Estos circuitos fueron diversos e incrementaban en su complejidad para asegurar que el proceso definido en cada etapa fuese correcto y se obtuvieran los resultados esperados en cada uno. A continuación se muestran los *layouts* de los circuitos con los que se trabajaron en la etapa de LVS en orden ascendente de complejidad y en el orden cronológico en el que se trabajaron. Todas estas visualizaciones son a través de *Custom Compiler* y el proceso para importarlos se explica a continuación.

8.1. *Layouts* de circuitos

8.1.1. Compuerta *NOT*

En la Figura #28 se puede observar el *layout* de una compuerta *NOT* desde la herramienta *Custom Compiler*.

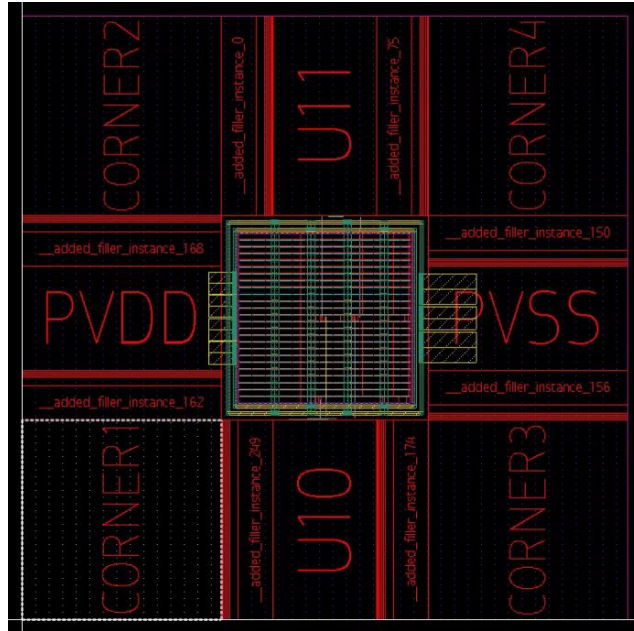


Figura 28: Layout de compuerta *NOT*

8.1.2. Compuerta XOR

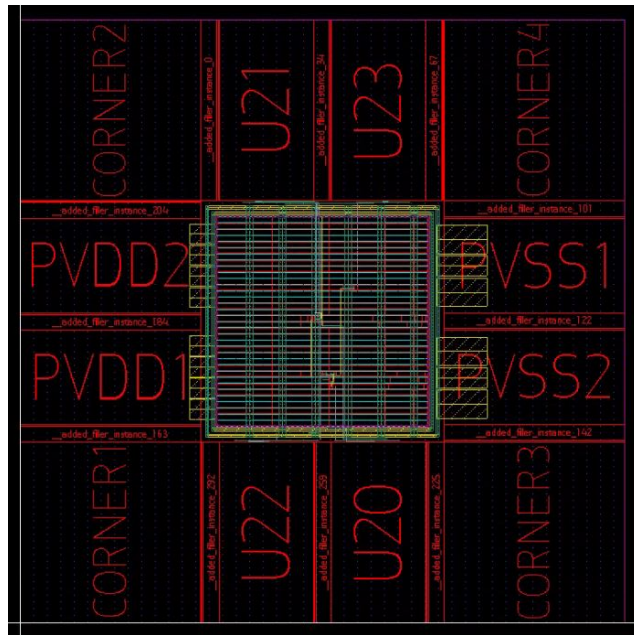


Figura 29: Layout de compuerta *XOR*

8.1.3. Circuito *Full Adder*

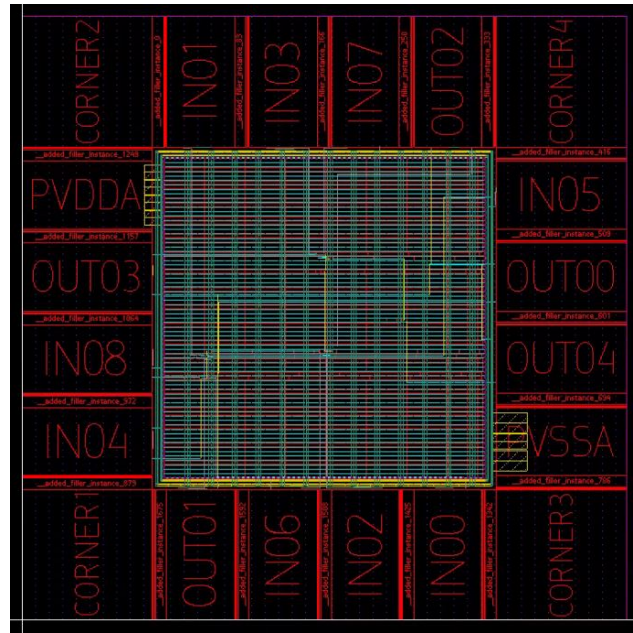


Figura 30: Layout de circuito *Full Adder*

8.1.4. Circuito *ALU*

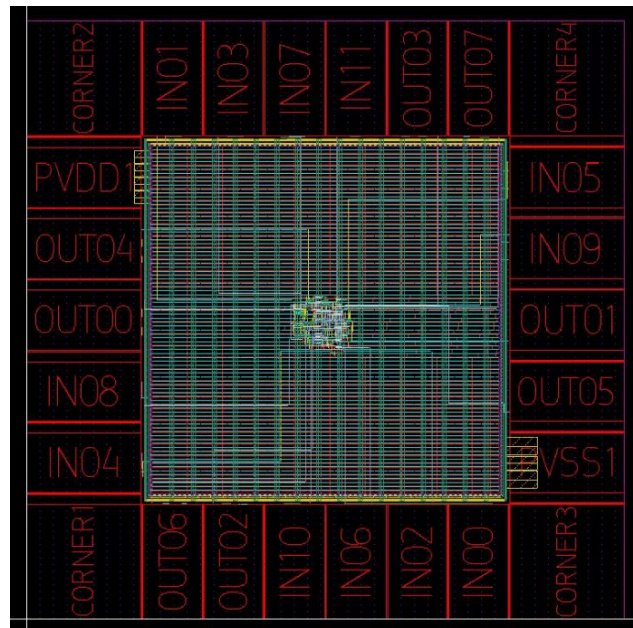


Figura 31: Layout de circuito *ALU*

8.1.5. Circuito *Counter*

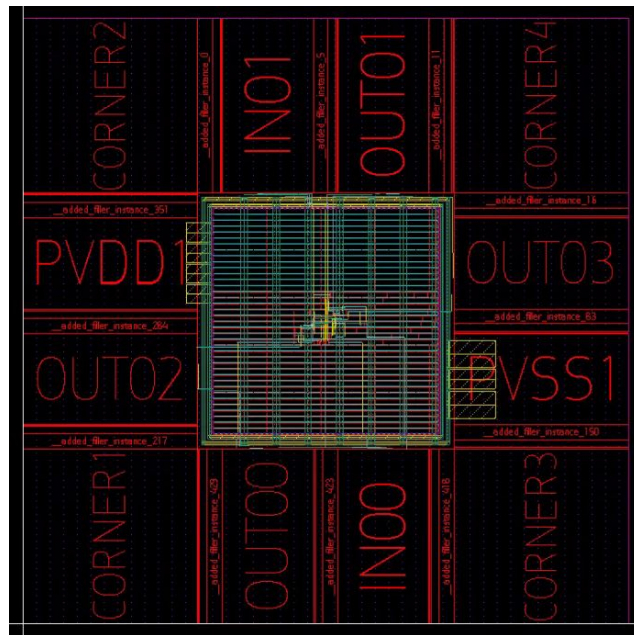


Figura 32: Layout de circuito *Counter*

8.1.6. Memoria *RAM*

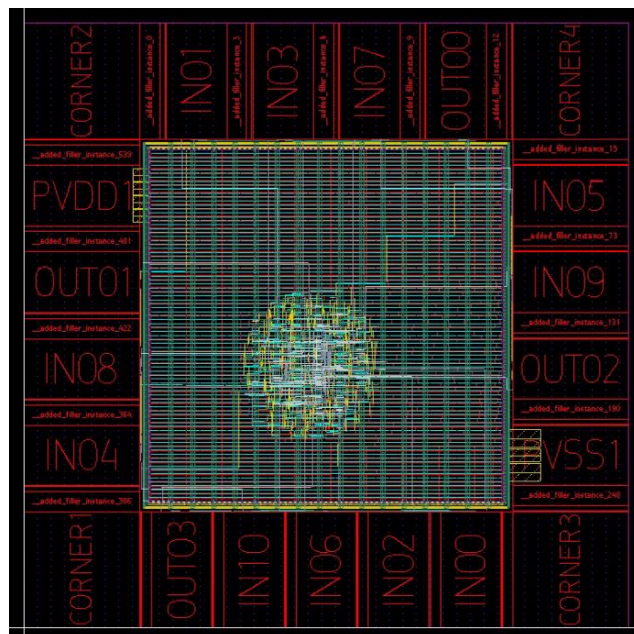


Figura 33: Layout de memoria *RAM*

Métodos de ejecución de LVS

En este capítulo se describen y se muestran dos métodos que se implementaron para poder ejecutar LVS. Se definirán los archivos necesarios para cada uno, los pasos que se deben seguir para la correcta ejecución de LVS, los retos que se encontraron al obtener resultados no deseados, las estrategias implementadas para corregir estos errores y finalmente se mostrarán los resultados obtenidos y los archivos generados.

9.1. *Custom Compiler*

El primer método que se trató de implementar fue la utilización de *Custom Compiler* como herramienta para realizar la verificación de LVS. En [4], se puede observar la implementación de esta herramienta, y al autor haciendo la mención en sus recomendaciones, que esta es la herramienta más agradable debido a que presenta los resultados de la prueba de una manera mucho más amigable para el usuario. Sin embargo, se tuvo que comprobar que aún con la actualización de las herramientas utilizadas, el proceso y los resultados seguían siendo los mismos, ya que se tuvo una modificación en el proceso, al migrar de *IC Compiler* a *IC Compiler II*. En esta migración se encontró el primer desafío, al no obtener un archivo *GDS* desde *IC Compiler II* compatible con *Custom Compiler*. Sin embargo, tal como se describirá en una sección futura, se logró solventar esta situación y ejecutar la prueba con esta herramienta.

A pesar de poder utilizar una interfaz gráfica mucho más amigable con el usuario y partir de una herramienta dedicada para las etapas de diseño del nano chip, se debe tener en cuenta que para realizar el LVS es necesario contar con todos los archivos mencionados en las secciones anteriores del documento, con la excepción del archivo *GDS*, ya que este puede ser generado en el mismo proceso en el que se ejecuta LVS a través de esta herramienta. Tal y como se mostró el proceso para poder importar una librería en *Custom Compiler* a través

de imágenes con una detallada descripción de cada una, se mostrará el proceso paso a paso para ejecutar LVS para cualquier circuito.

Lo primero que se debe hacer es ejecutar el comando `custom_compiler`, de la manera que se describió en la sección 7.3.2 de este documento. Una vez dentro de *Custom Compiler* se debe ingresar a la *Library Manager* para seleccionar la librería del circuito que se estará trabajando. Se debe ingresar a la vista del *layout* del circuito tal y como se mostró en la Figura #23.

Una vez dentro de la vista del *layout*, tal y como se muestra en la Figura #34, se debe acceder a la opción de *Verification*, luego *LVS* y finalmente a *Setup and Run* como se muestra en la Figura #35.

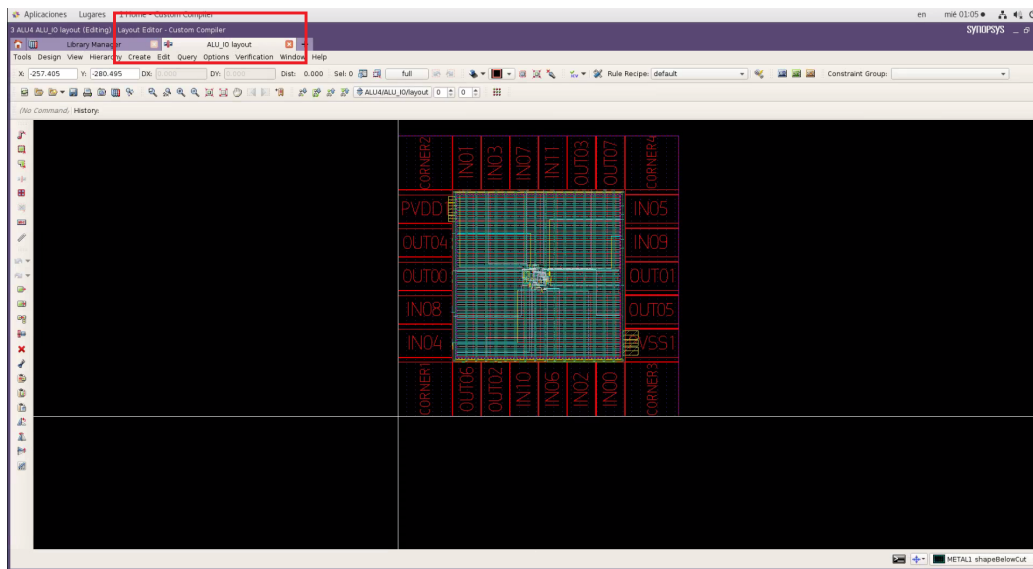


Figura 34: Vista de *Layout* en *Custom Compiler*

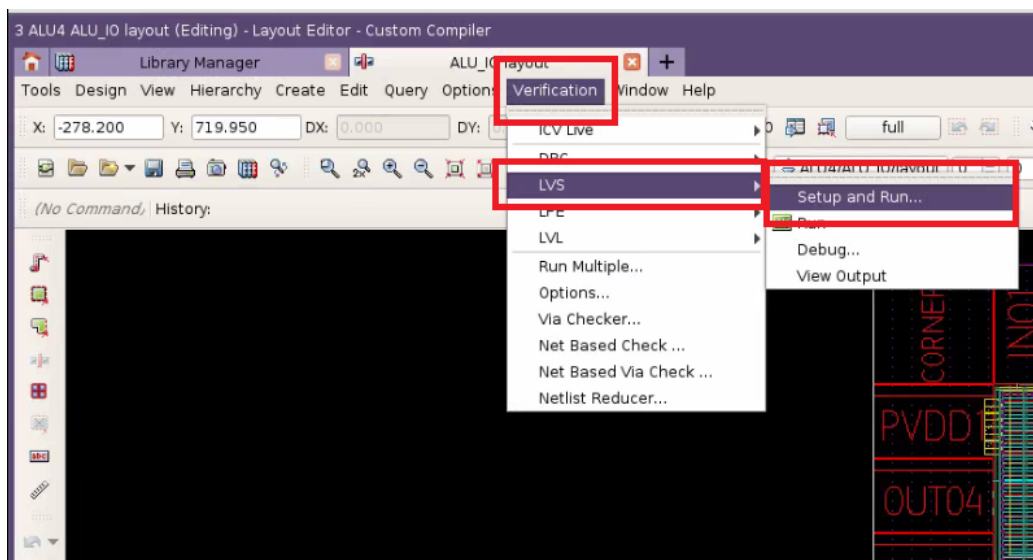


Figura 35: *Setup* para ejecución de LVS

Al acceder a esta opción aparecerá el cuadro que se muestra en la Figura #36. Lo primero que se debe modificar dentro de este configurador es la dirección en donde se ejecutará el comando. En la imagen está resaltado en rojo el espacio donde se debe definir esta ubicación. Es por esto que se recomienda generar carpetas con un nombre específico para cada una de las pruebas realizadas. Esto simplificará de gran manera el proceso, ya que se volverá algo mecánico al solo variar los nombres de las carpetas por cada circuito.

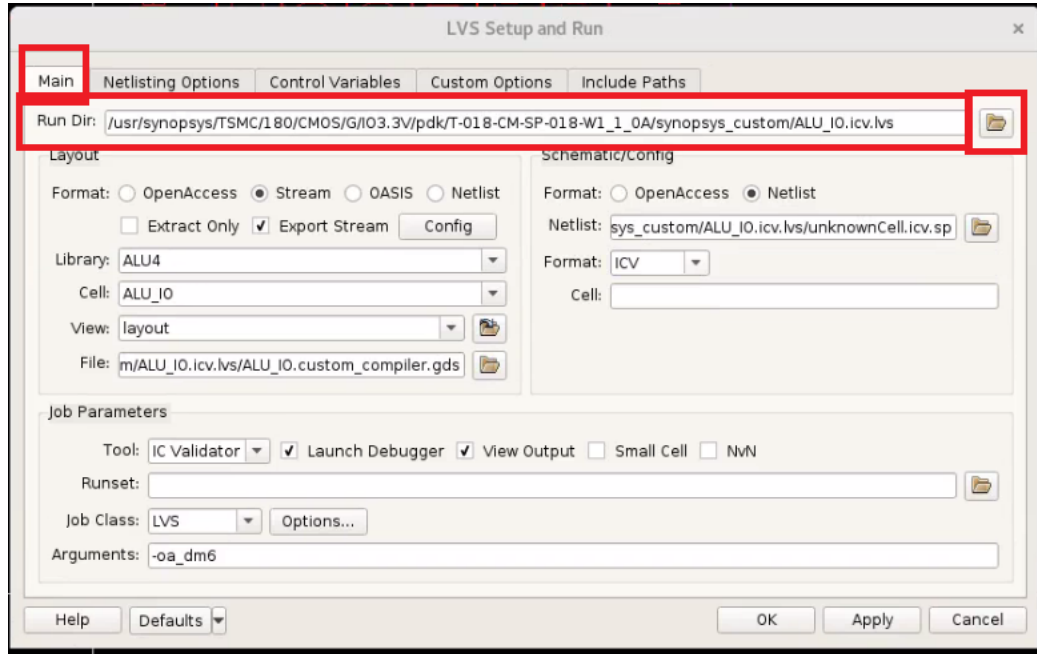


Figura 36: Configuración de LVS, *Run Directory*

Seguido de esta modificación se debe editar la sección de *Layout* que está resaltada en rojo en la Figura #37. Dentro de esta configuración se debe definir el formato de archivo que se está utilizando, para este caso específico se realizó el LVS de la *ALU*. Al seleccionar el formato *Stream*, la siguiente opción es exportar este archivo, el cual es el archivo GDS. Las siguientes casillas resaltadas definen la librería utilizada, en este caso *ALU4*, el nombre de la celda, *ALU_IO*, el tipo de vista que es *layout* y finalmente el nombre del archivo en formato GDS que se exportará. Todos estos parámetros se auto definen, por lo que hay que asegurarse que todo este en orden para cada circuito.

A continuación se modifica el apartado de *Schematic/Config*, resaltado en rojo en la Figura #38. Se debe seleccionar el formato *Netlist* y definir la ruta específica hacia el archivo ICV del circuito. Recordar que al mantener un orden para guardar todos los documentos generados, estos pasos se facilitan. Luego, se debe establecer el formato ICV del documento y, por último, definir el nombre de la celda que se está trabajando, el cual debe corresponder con el nombre de la celda en el archivo ICV.

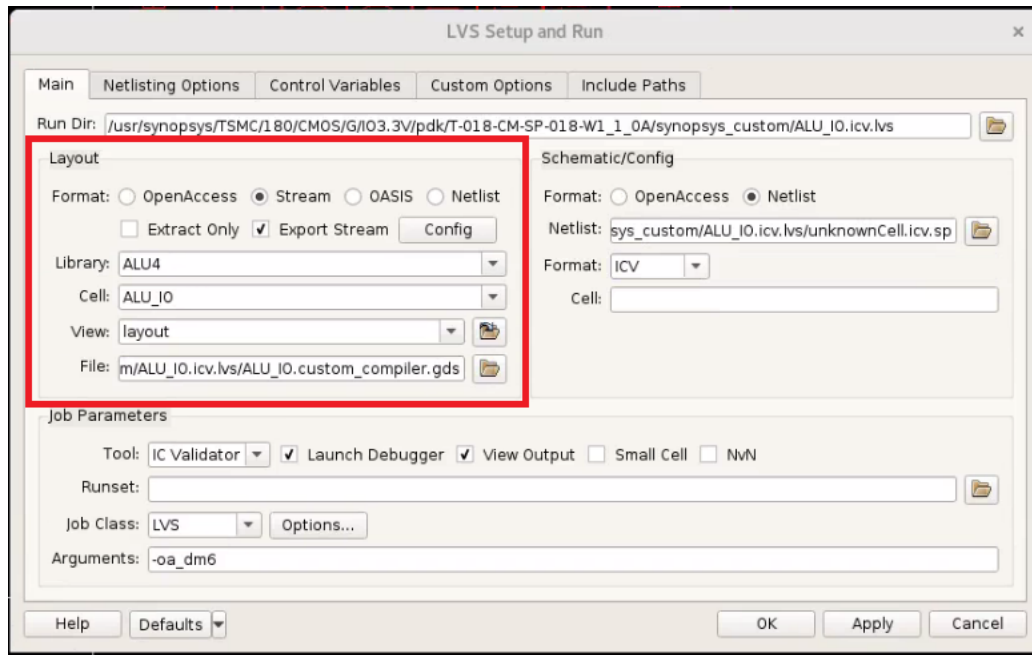


Figura 37: Configuración de LVS, *Layout*

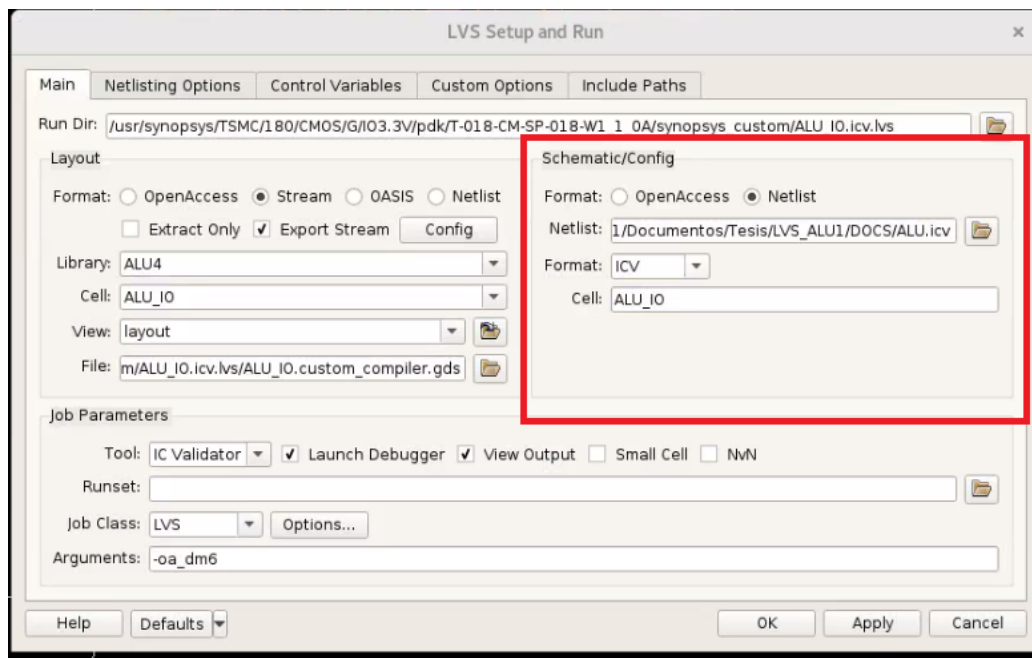


Figura 38: Configuración de LVS, *Schematic*

Finalmente, se debe modificar el apartado de *Job Parameters* tal como se muestra resaltado en la Figura #39. Tal como se ha realizado anteriormente, se debe buscar la ruta específica del archivo de *Runset* de cada circuito. Se debe asegurar que el tipo de archivo que se está buscando sea cualquiera, tal como se muestra en la Figura #40. Por último, se debe dejar la opción de argumentos libre. Esto se menciona ya que por defecto este espacio tiene un texto.

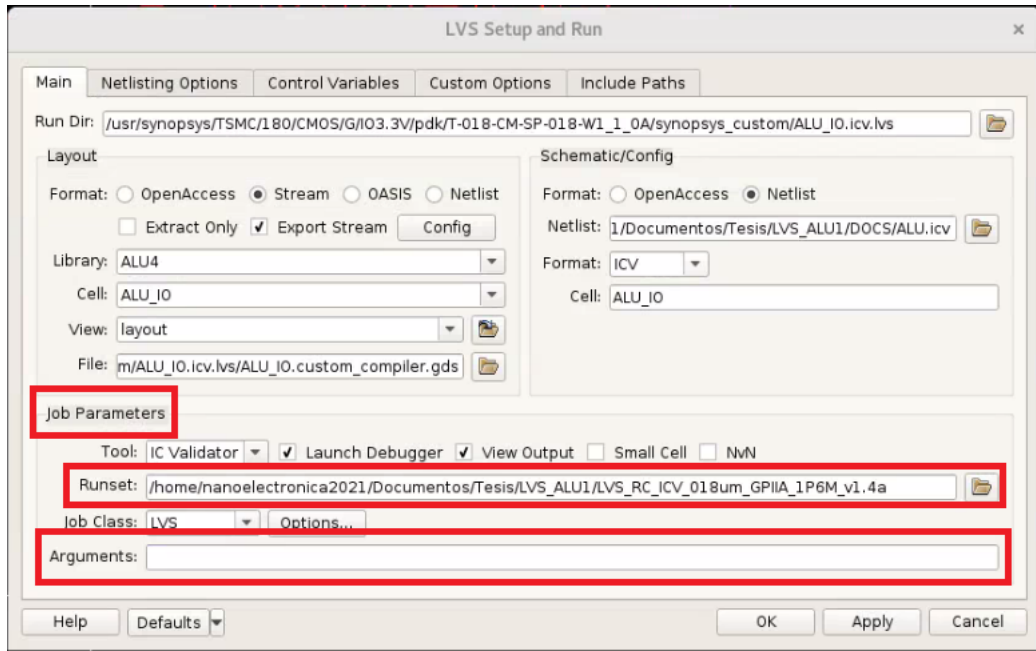


Figura 39: Configuración de LVS, *Runset*

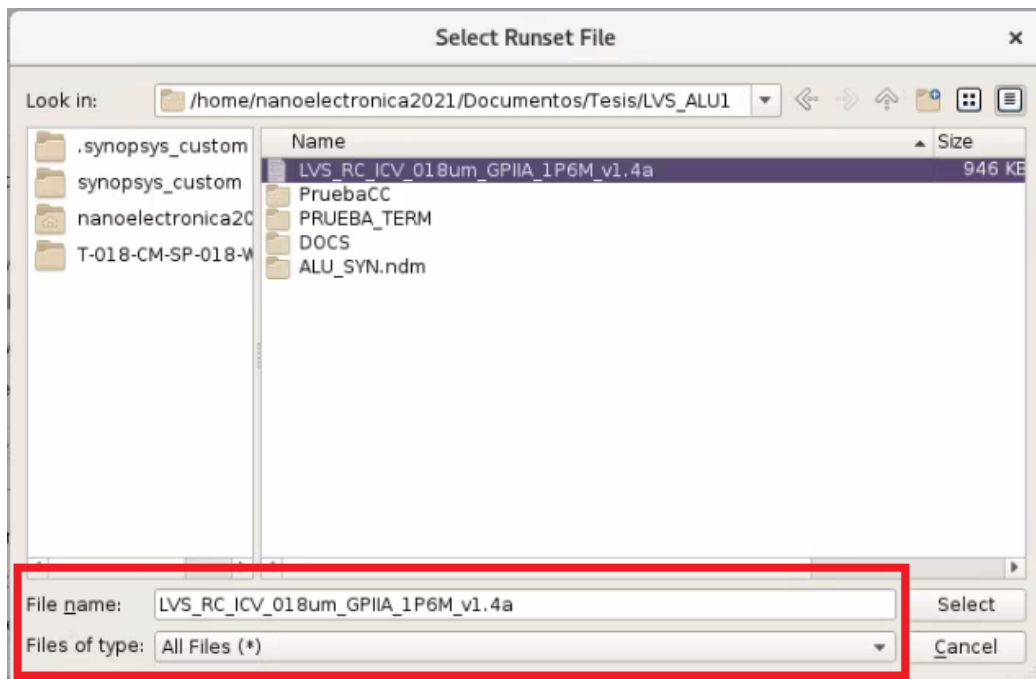


Figura 40: Ubicación de archivo *Runset* para LVS

Al realizar estos pasos y hacer click en *OK* se obtendrán todos los archivos de salida que se muestran en la Figura #41.

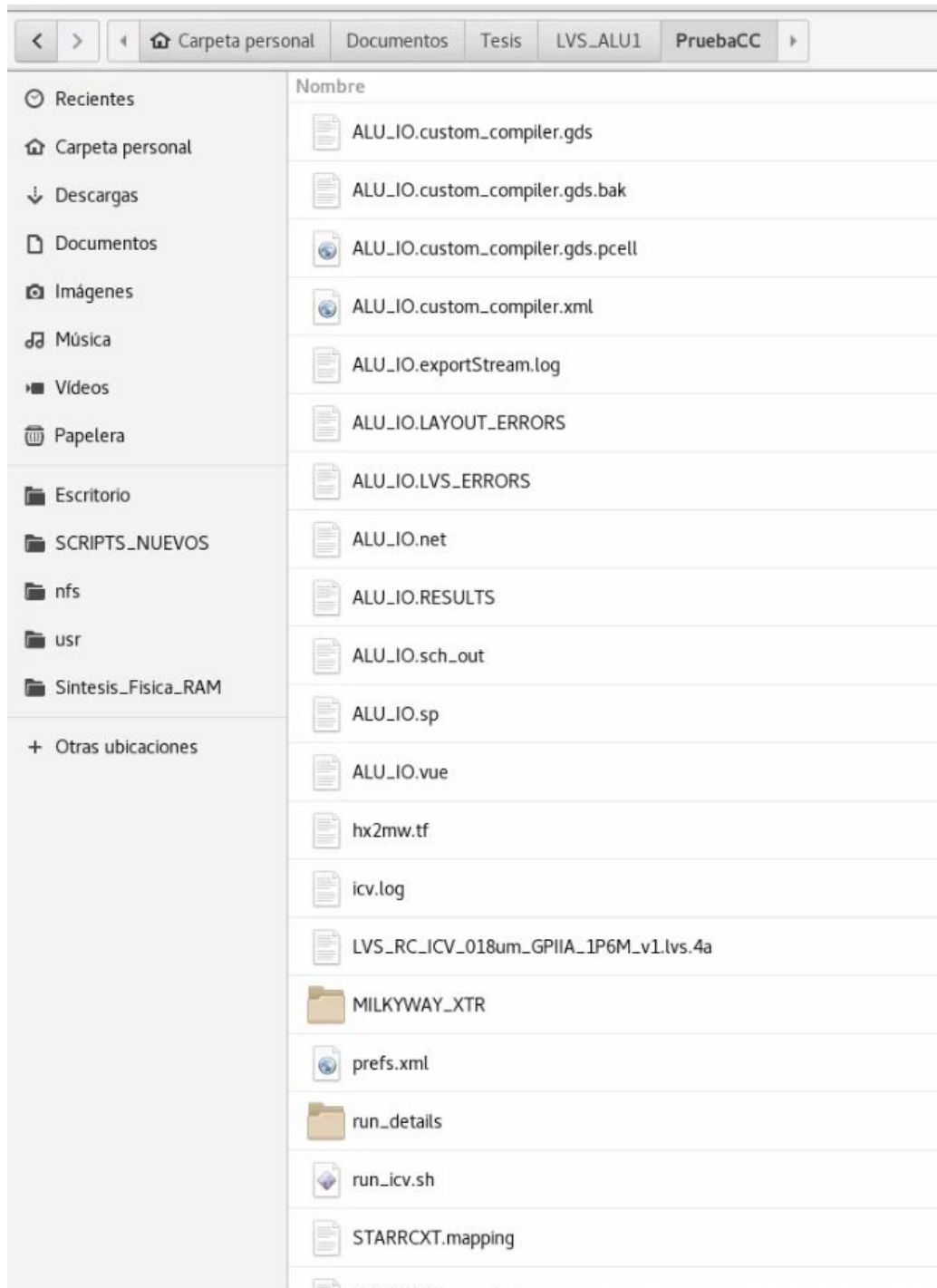


Figura 41: Archivos de salida de LVS desde *Custom Compiler*

9.2. Comandos en terminal

La ejecución de LVS mediante líneas de comandos en una terminal no parecía algo amigable para la ejecución del usuario. Esto según comentarios y conclusiones observadas en [4]. Ya que la ejecución mediante *Custom Compiler* tiene una interfaz que permite determinar

con mayor facilidad el origen de los errores y así corregirlos de una manera más precisa y rápida. Sin embargo, debido a los problemas que se expusieron en la sección anterior y con la intención de obtener resultados lo más pronto posible en la primeras etapas de diseño, se optó por realizar este método de ejecución, que, aún no siendo tan amigable para el usuario, resultó ser sumamente eficiente y fácil de comprender. Lo primero que se requirió para este método fue tener todos los comandos necesarios para poder ejecutar cada una de las etapas que se describirán a continuación.

Además de esto, se tuvo que corroborar que se contaba con todos los archivos necesarios, siendo estos los expuestos en el capítulo 7 de este documento. Para tener mayor agilidad en este método, se generó un archivo con todos los comandos necesarios, no solo para ejecutar el LVS sino los necesarios para todo el proceso que se ha mencionado con anterioridad, en el cual se concatenan archivos existentes, se traducen a otros formatos o bien, se generan archivos totalmente nuevos. Para una referencia rápida, podemos observar la lista de estos comandos en el orden que se deben ejecutar para la obtención de los archivos en los formatos adecuados para estar completamente listos para ejecutar el LVS en el Cuadro #2.

Función	Comando	Input	Output
Concatenar	<code>cat CORE.v IO.v > headers.v</code>	CORE.v IO.v	headers.v
Traducción .v a .sp	<code>icv_nettran -verilog doc.v -outType SPICE -outName doc.sp</code>	doc.v	doc.sp
Generar <i>ICV</i>	<code>icv_nettran -verilog doc.v -sp doc.sp -outType ICV -outName out.icv</code>	doc.v doc.sp	out.icv
Ejecutar <i>Custom Compiler</i>	<code>custom_compiler</code>	NA	NA
Ejecutar LVS	<code>icv -i doc.gds -c top_cell -s doc.icv -sf ICV -vue runset_file</code>	doc.gds top_cell doc.icv runset_file	Varios

Cuadro 2: Lista de comandos

1. `icv_nettran`: Es el comando utilizado para invocar la herramienta de traducción *Nettran*.
2. `-verilog`: Se utiliza cuando se emplea un archivo en formato *verilog*.
3. `-sp`: Se utiliza cuando se emplea un archivo en formato *spice*.
4. `icv`: Es el comando utilizado para invocar la herramienta de *IC Validator*, la cual ejecuta el LVS.
5. `-i`: Comando para definir la librería que se utilizará en el LVS, para este caso se define el archivo en formato *GDS*. Colocar su ruta exacta.
6. `-c`: Comando para definir el nombre de la *Top Cell* que se utilizará.
7. `-s`: Comando para definir el *netlist* en formato *ICV*. Colocar su ruta exacta.
8. `-sf`: Comando para definir el formato del *netlist*, en este caso es *ICV*.
9. `-vue`: Comando para definir el archivo *runset* que se utilizará. Colocar su ruta exacta.

Con esta lista de comandos y la explicación de cada uno, se puede definir el proceso paso a paso para ejecutar LVS a través de comandos en una terminal. Se debe tomar en cuenta que los pasos para obtener los archivos previos han sido mostrados a detalle y en el orden en que estos de deben obtener para poder ejecutar el LVS.

1. Asegurarse de contar con todos los documentos descritos con anterioridad. De no ser así, regresar algunas secciones para seguir los pasos y obtenerlos. Los archivos necesarios se muestran en la Figura #42 y Figura #43. En este caso son los archivos para el circuito *Full Adder*, pero todos los circuitos deben tener los mismos formatos de archivos.

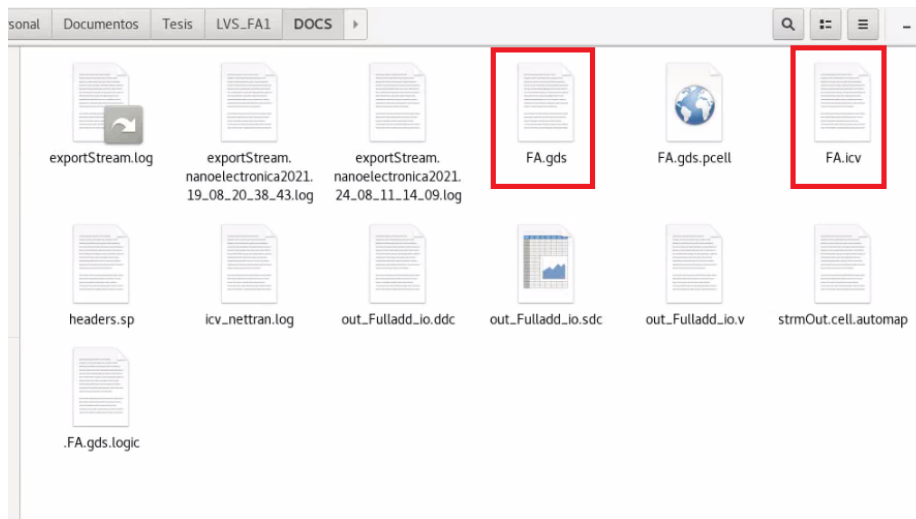


Figura 42: Archivos necesarios 1

2. Modificar el archivo *Runset*, tal como se describe en la sección 7.5 de este documento.

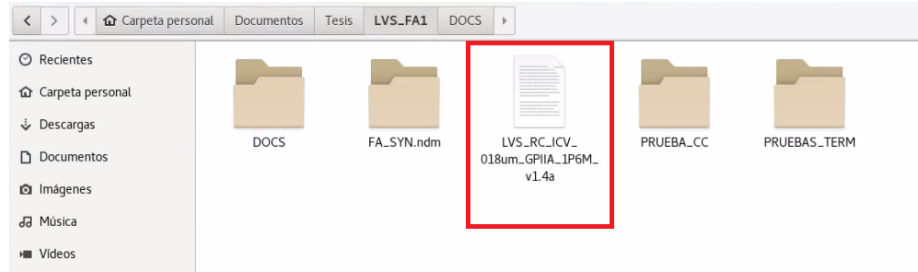


Figura 43: Archivos necesarios 2

- Desde una carpeta específica abrir una ventana de terminal. En este caso se ejecutó desde una carpeta con el nombre de `PRUEBAS_TERM` dentro de la carpeta del circuito trabajado, este nombre se repite en todos los circuitos para seguir un orden, puede observarse en la Figura #44.

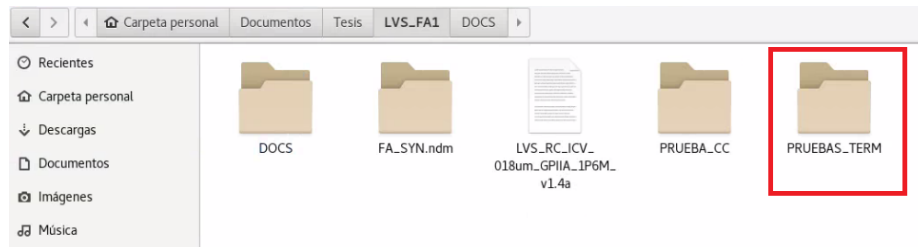


Figura 44: Carpeta de pruebas en terminal

- Desde esta terminal, se debe ejecutar el último comando descrito en el Cuadro #2. Para esto se deben aplicar ciertos cambios al comando, tomando los *inputs* que se definen en el Cuadro #2, se deben introducir las rutas específicas de dichos archivos para que el comando se ejecute correctamente. En el caso del *input* de `top_cell` se debe escribir el nombre de la misma tal y como está definido en el documento ICV.
- Una vez ejecutado este comando, se obtendrán todos los archivos de salida dentro de la carpeta desde donde se ejecuto la terminal, tal y como se ve en la Figura #45.

9.3. Archivos generados

Al momento de ejecutar LVS, ya fuese a través de *Custom Compiler* o mediante líneas de comando en una terminal, se obtienen básicamente los mismos archivos de salida. Esto dependerá siempre, de los archivos que se hayan definido para que este proceso exporte en el archivo *Runset*, tal como se describió en capítulos anteriores (ver Figura #26). En la Figura #41 y Figura #45 se muestran los distintos archivos y carpetas de salida. Los archivos más importantes con el fin de revisar los errores, de existir, para poder reportarlos a los equipos de trabajo de la síntesis física y síntesis lógica y que estos fueran corregidos a la brevedad posible y poder continuar con el flujo de diseño. Se puede observar una diferencia en la cantidad de archivos generados por cada método, sin embargo, esto no afecta los archivos que se deseaban obtener. Adicional a esto, como se puede observar en la Figura #46 y

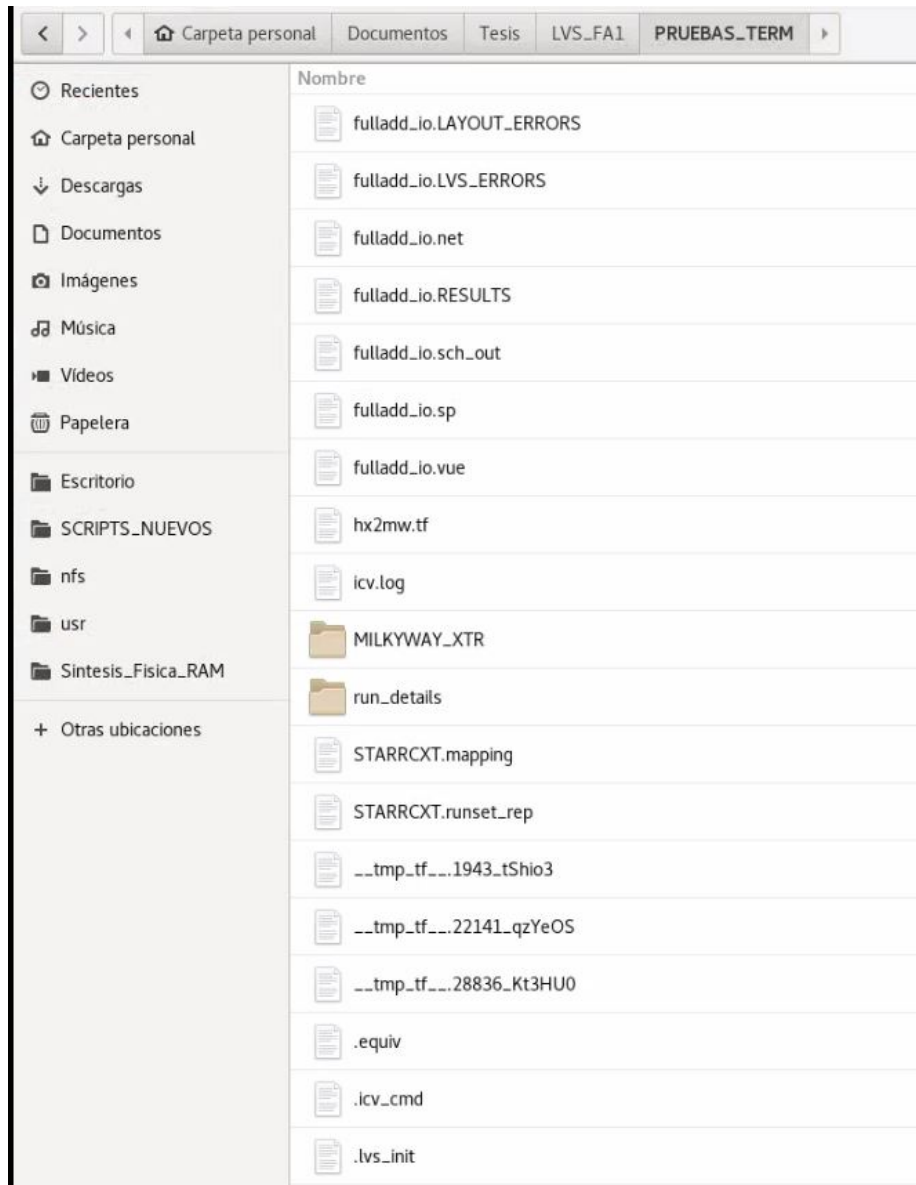


Figura 45: Archivos de salida de LVS desde terminal

en la Figura #47, los archivos y carpetas resaltados en rojo, son los archivos que fueron trasladados al equipo de trabajo de extracción de parásitos o *LPE*. Adicional a estos, se compartieron los archivos ICV, GDS, y *Runset*.

Cabe mencionar que a pesar que se exponen en primera instancia, todos los resultados deseados, el proceso no fue así de sencillo. Como se mencionó en secciones anteriores, durante el proceso se encontraron varios retos en diferentes etapas del LVS. Estas pudieron haber sido desde los equipos de trabajo anteriores y las complicaciones o errores que tuvieran o bien se generaron a raíz de la migración de las herramientas utilizadas. En el siguiente capítulo se describen los errores más comunes, los más complicados, y los más tardados de solucionar, con el fin que el siguiente equipo interesado en replicar este proyecto pueda solucionarlos de la manera más eficiente o bien evitarlos a toda costa.

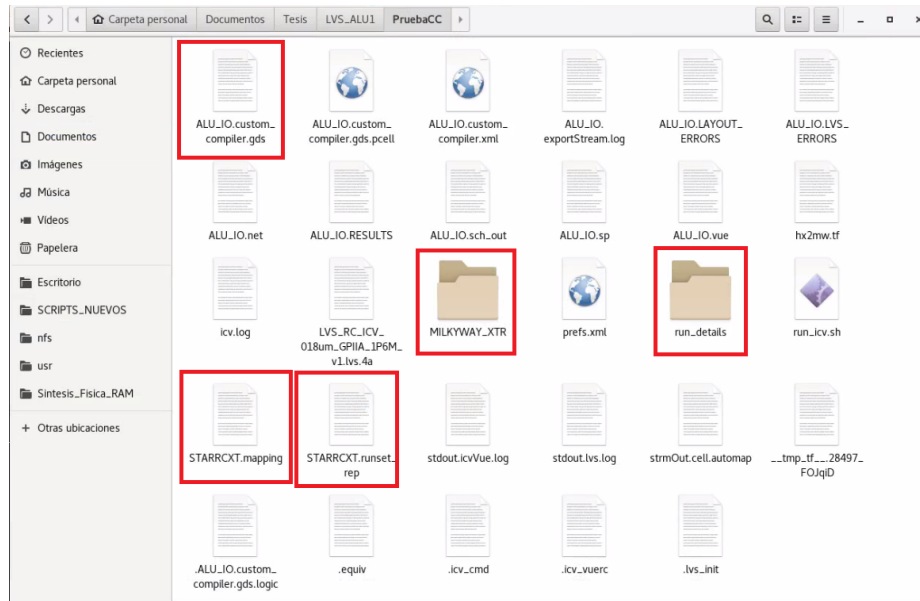


Figura 46: Archivos compartidos de LVS desde *Custom Compiler*

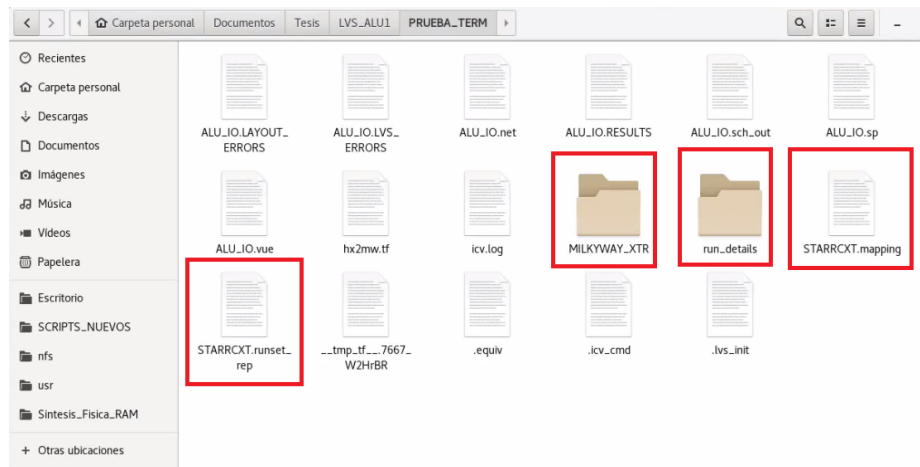


Figura 47: Archivos compartidos de LVS desde terminal

En este capítulo se describen algunos de los problemas que se encontraron a lo largo de todo el proceso. En cada sección se define un problema específico, ya sea con un archivo o con un proceso puntual dentro de alguna herramienta. Adicionalmente, se explica la manera más eficiente de solucionarlos.

10.1. Comandos en terminal

Los errores más comunes y recurrentes en este proceso de LVS, es escribir erróneamente un comando dentro de la terminal. Esto puede ocurrir cuando falta un guión, la dirección de los archivos no es la correcta, o simplemente que algo esté mal escrito. Tanto para traducir archivos de un formato a otro, extraer archivos en un formato específico o bien ejecutar finalmente el LVS, se debe asegurar que los comandos estén correctamente escritos. En el Cuadro #2 se muestran todos los comandos empleados para corregir este error de la manera más eficiente posible.

10.2. Archivo *Runset*

Antes de ejecutar el LVS se debe modificar el archivo *Runset* para que este tenga las ubicaciones adecuadas, el nombre de las *top cells* que se van a utilizar y las instancias de las *black boxes* que corresponden a las celdas implementadas en el *layout* y *elschematic*.

En la Figura #48 se puede observar resaltado en recuadros rojos, los mensajes de errores que se encontraron al momento de ejecutar LVS. También se indica la dirección del archivo donde se pueden encontrar con mayor detalle la razón de estos errores.

```

nanoelectronica2021@uvjemtbmj31305:~/Documentos/Tesis/LVS_GRAN_JAGUAR/PRUEBAS_TERM
Archivo Editar Ver Buscar Terminal Ayuda
ICV_Engine run is 80% complete. Elapsed Time=0:00:07
ICV_Engine run is 85% complete. Elapsed Time=0:00:07
ICV_Engine run is 90% complete. Elapsed Time=0:00:07
ICV_Engine run is 95% complete. Elapsed Time=0:00:10
Comparing schematic and layout netlists.
LVS compare start time : 2021-11-10 22:50:41
LVS Compare run is 0% complete. Elapsed Time=0:00:04
LVS Compare run is 5% complete. Elapsed Time=0:00:04

ERROR: The program ICV_Compare did not complete.
Please refer to /home/nanoelectronica2021/Documentos/Tesis/LVS_GRAN_JAGUAR/PRUEBAS_TERM/run_details/chip_IO_lvs.log for
error message.

Check Time=0:00:06 User=0.01 Sys=0.01 Mem=0.036 GB

ICV_Engine run is 100% complete. Elapsed Time=0:00:19

Completing error storage...
Overall error storage time: None

Generating chip_IO_LAYOUT_ERRORS...
Generation Time=0:00:00 User=0.00 Sys=0.01 Mem=0.001 GB

Check Time=0:00:00 User=0.02 Sys=0.00 Mem=0.009 GB

icv_compare exit with error status 33
IC Validator Run: Time=0:00:53
IC Validator did not complete.
nanoelectronica2021@uvjemtbmj31305:~/Documentos/Tesis/LVS_GRAN_JAGUAR/PRUEBAS_TERM$

```

Figura 48: Error en LVS mediante terminal

En la Figura #49 se resalta la ubicación del archivo que se describió anteriormente. Este, servirá para identificar en dónde radica el problema para poder solucionarlo de manera efectiva.

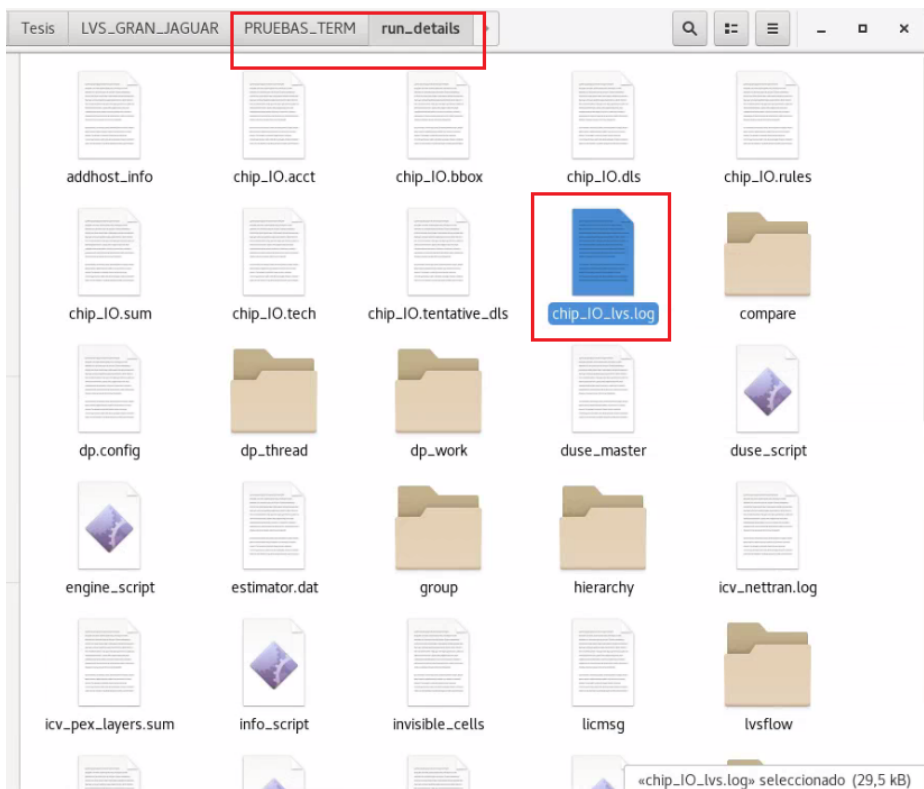
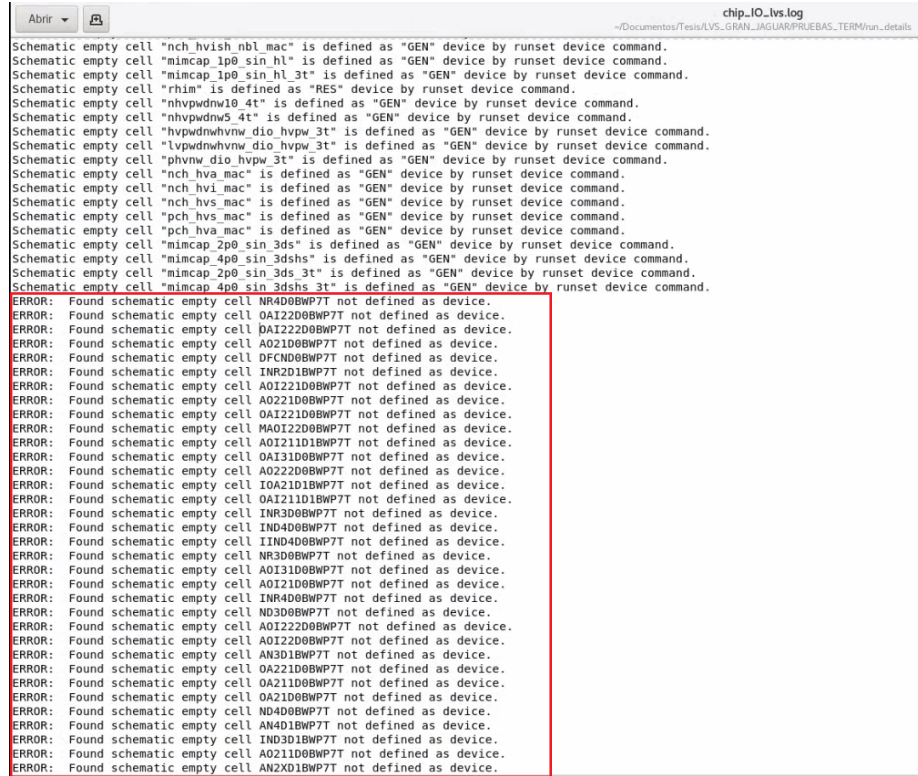


Figura 49: Error en archivo *Runset 1*

Al abrir este archivo se puede encontrar una serie de mensajes de errores al final del mismo, tal como se muestra en la Figura #50. El error que este describe es que se encuentran celdas vacías del archivo *schematic* y que no están definidas como dispositivos a utilizar. Cada

línea de error responde a una celda particular no instanciada dentro del archivo *Runset*.



```
chip_IO_lvs.log
~/Documentos/Tesis/LVS_GRAN_JAGUAR/PROJEBAS_TERM/run_details

Schematic empty cell "nch hvish nbl mac" is defined as "GEN" device by runset device command.
Schematic empty cell "mimcap 1p0 sin hl" is defined as "GEN" device by runset device command.
Schematic empty cell "mimcap 1p0 sin hl 3t" is defined as "GEN" device by runset device command.
Schematic empty cell "rhim" is defined as "RES" device by runset device command.
Schematic empty cell "nhvpwdnw10 4t" is defined as "GEN" device by runset device command.
Schematic empty cell "nhvpwdnw5 4t" is defined as "GEN" device by runset device command.
Schematic empty cell "hvpwdnwhvwnv dio hvpw 3t" is defined as "GEN" device by runset device command.
Schematic empty cell "lvpwdnwhvwnv dio hvpw 3t" is defined as "GEN" device by runset device command.
Schematic empty cell "phvwnv dio hvpw 3t" is defined as "GEN" device by runset device command.
Schematic empty cell "nch hva_mac" is defined as "GEN" device by runset device command.
Schematic empty cell "nch hvi_mac" is defined as "GEN" device by runset device command.
Schematic empty cell "nch hvs_mac" is defined as "GEN" device by runset device command.
Schematic empty cell "pch hvs_mac" is defined as "GEN" device by runset device command.
Schematic empty cell "pch hva_mac" is defined as "GEN" device by runset device command.
Schematic empty cell "mimcap 2p0 sin 3ds" is defined as "GEN" device by runset device command.
Schematic empty cell "mimcap 4p0 sin 3ds" is defined as "GEN" device by runset device command.
Schematic empty cell "mimcap 2p0 sin 3ds 3t" is defined as "GEN" device by runset device command.
Schematic empty cell "mimcap 4p0 sin 3ds 3t" is defined as "GEN" device by runset device command.
ERROR: Found schematic empty cell NR4D08BWP7T not defined as device.
ERROR: Found schematic empty cell DA122D08BWP7T not defined as device.
ERROR: Found schematic empty cell DA122D08BWP7T not defined as device.
ERROR: Found schematic empty cell AO21D08BWP7T not defined as device.
ERROR: Found schematic empty cell DFCND08BWP7T not defined as device.
ERROR: Found schematic empty cell INR2D1BWP7T not defined as device.
ERROR: Found schematic empty cell AO1221D08BWP7T not defined as device.
ERROR: Found schematic empty cell AO221D08BWP7T not defined as device.
ERROR: Found schematic empty cell AO1221D08BWP7T not defined as device.
ERROR: Found schematic empty cell MA0122D08BWP7T not defined as device.
ERROR: Found schematic empty cell AO1211D1BWP7T not defined as device.
ERROR: Found schematic empty cell AO131D08BWP7T not defined as device.
ERROR: Found schematic empty cell AO222D08BWP7T not defined as device.
ERROR: Found schematic empty cell IOA21D1BWP7T not defined as device.
ERROR: Found schematic empty cell AO1211D1BWP7T not defined as device.
ERROR: Found schematic empty cell INR3D08BWP7T not defined as device.
ERROR: Found schematic empty cell IND4D08BWP7T not defined as device.
ERROR: Found schematic empty cell IIND4D08BWP7T not defined as device.
ERROR: Found schematic empty cell NR3D08BWP7T not defined as device.
ERROR: Found schematic empty cell AO131D08BWP7T not defined as device.
ERROR: Found schematic empty cell AO121D08BWP7T not defined as device.
ERROR: Found schematic empty cell INR4D08BWP7T not defined as device.
ERROR: Found schematic empty cell ND3D08BWP7T not defined as device.
ERROR: Found schematic empty cell AO1222D08BWP7T not defined as device.
ERROR: Found schematic empty cell AO122D08BWP7T not defined as device.
ERROR: Found schematic empty cell AN3D1BWP7T not defined as device.
ERROR: Found schematic empty cell OA221D08BWP7T not defined as device.
ERROR: Found schematic empty cell OA211D08BWP7T not defined as device.
ERROR: Found schematic empty cell OA21D08BWP7T not defined as device.
ERROR: Found schematic empty cell ND4D08BWP7T not defined as device.
ERROR: Found schematic empty cell AN4D1BWP7T not defined as device.
ERROR: Found schematic empty cell IND3D1BWP7T not defined as device.
ERROR: Found schematic empty cell AO211D08BWP7T not defined as device.
ERROR: Found schematic empty cell AN2XD1BWP7T not defined as device.
```

Figura 50: Error en archivo *Runset* 1

Dentro del archivo *Runset*, si se dirige a la sección de *ICV OPTIONS* dentro del documento, se pueden observar las instancias de las *black boxes* de las celdas que se están utilizando para cada circuito, tal como se resalta en un cuadro rojo en la Figura #51.

Las celdas que se mencionan en la Figura #50 pueden ser encontradas en el archivo *headers.sp* el cual se ha mencionado con anterioridad. Este archivo contiene todas las celdas necesarias para cualquier circuito a implementar. En la Figura #52 se muestra resaltada una celda de las que se menciona en el archivo de errores.

La manera de resolver este problema es crear instancias dentro del *Runset* para cada una de las celdas que no estan definidas, tal como se especificó en la Figura #50. Estas instancias deben verse exactamente como la que se muestra resaltada en la Figura #51 y responder a la sintaxis que se muestra en las instancias de las celdas contenidas en el archivo *headers.sp*, tal como se muestra en la Figura #53.

Una vez solventado este proceso de instanciar todas las celdas faltantes, el LVS se podrá ejecutar con normalidad, siempre y cuando el resto de archivos y comandos se encuentren bien. Este proceso es sumamente importante, ya que a medida que se experimenta con compuertas o circuitos más complicados, la cantidad de celdas que deben ser instanciadas es mayor. Por lo cual, cada vez que se cambia de circuito, es muy probable que aparezca este error debido a que solo se han instanciado las celdas del circuito menos complejo trabajado con anterioridad.

```

*V533SUB*, "V5S", "V5SESD", "V55G", "V55M", "V5SPST", "V5SUB"};

/// ICV OPTIONS ///

lvs_options(
  #ifdef USER_EQUIV_FILE
    generate_system_equivs = false,
  #endif
  device_extraction_preserved_cells = {"cfmom**", "cfmom_mx**", "cfmom_rf**", "crtmom**", "crtmom_rf**", "lcesd1_rf**", "lcesd2_rf**", "lowcapac",
  "micap_rf**", "mos_var**", "moscap_rf**", "mos_var33**", "moscap_rf33**", "moscap_rf33_nw**", "moscap_rf_nw**", "radio_hia_rf**", "radio_sbd_mac**", "ri",
  "pdio_hia_rf**", "rfpmos2v**", "rfpmos3v**", "rfpmos3v_5t**", "rfpmos3v_nw**", "rfpmos3v_nw_5t**", "rfpmos2v_5t**", "rfpmos2v_nw**", "rfpmos2v_nw_5t**",
  "sbd_rf**", "sbd_rf_nw**", "ind_std_40k**", "spiral_std_mu_x_40k**", "ind_sym_40k**", "ind_sym_ct_40k**", "spiral_sym_ct_mu_x_40k**", "spiral_sym_mu",
  };
resolution_options(
  internal_resolution = 0.0001
);
);
layout_grid_options(
  resolution = 0.001
);
);
error_options(
  error_limit_per_check = 10000
);
);
run_options(
  instance_prefix = "I_"
);
);
net_options(
  schematic_power = POWER_NAME,
  schematic_ground = GROUND_NAME
);
);
text_options(
  colon_text = TRUNCATE,
  semicolon_text = REGULAR_TEXT,
  net_prefix = "B ",
  layout_power = POWER_NAME,
  layout_ground = GROUND_NAME
);
);
lvs_black_box_options(
  equiv_cells = {"schematic_cell = "CKXOR2D4BWP7T", layout_cell = "CKXOR2D4BWP7T"}},
  remove_schematic_ports = {"A1", "A2", "Z"}
);
);
lvs_black_box_options(

```

Figura 51: Error en archivo *Runset 1*

10.3. Exportar archivo GDS

Uno de los problemas o retos ya mencionados en este trabajo, fue la extracción del archivo GDS. Ya que en las etapas iniciales del proyecto, este era compartido por el equipo de síntesis física, sin embargo, no era compatible para la ejecución de LVS. Esta sección solo hace referencia al proceso correcto para extraer el GDS. Esto ya fue explicado en el capítulo 7, pero se debe resaltar la importancia de este paso dentro del proceso. Uno de los errores que se encontraron al momento de exportar el GDS fue el no especificar el formato del archivo. En la sección de *Output* que esta resaltada en rojo en la Figura #25, en el apartado *Stream File* se debe agregar el formato al nombre del archivo. En los primeros intentos se obtuvo un archivo que no correspondía al formato deseado, a pesar de haber especificado el proceso de exportar un *Stream File*. Esto se resuelve al escribir `.gds` al final del nombre del archivo. Este y todas las otras complicaciones que se puedan llegar a tener en el proceso de extraer el GDS pueden solventarse si se sigue el proceso descrito paso a paso en la sección 7.4 del documento.

10.4. Ejecutar LVS en *Custom Compiler*

Esta etapa fue sin duda la más complicada, debido a la falta de compatibilidad al usar las herramientas actualizadas y los procesos que se habían descrito en trabajos anteriores. No se extenderá en la explicación de todos los errores encontrados, debido a que la solución a estos es seguir paso a paso el proceso descrito en el capítulo 9 de este documento. Sin embargo, se

```
headers.sp
~/Documentos/Tesis/LVS_RAM...
Guardar

.SUBCKT 0AI31D1BWP7T A1 A2 A3 B ZN
.ENDS

.SUBCKT 0AI31D0BWP7T A1 A2 A3 B ZN
.ENDS

.SUBCKT 0AI22D2BWP7T A1 A2 B1 B2 ZN
.ENDS

.SUBCKT 0AI22D1BWP7T A1 A2 B1 B2 ZN
.ENDS

.SUBCKT 0AI22D0BWP7T A1 A2 B1 B2 ZN
.ENDS

.SUBCKT 0AI222D2BWP7T A1 A2 B1 B2 C1 C2 ZN
.ENDS

.SUBCKT 0AI222D1BWP7T A1 A2 B1 B2 C1 C2 ZN
.ENDS

.SUBCKT 0AI222D0BWP7T A1 A2 B1 B2 C1 C2 ZN
.ENDS

.SUBCKT 0AI221D2BWP7T A1 A2 B1 B2 C ZN
.ENDS

.SUBCKT 0AI221D1BWP7T A1 A2 B1 B2 C ZN
.ENDS

.SUBCKT 0AI221D0BWP7T A1 A2 B1 B2 C ZN
.ENDS

.SUBCKT 0AI21D2BWP7T A1 A2 B ZN
.ENDS

.SUBCKT 0AI21D1BWP7T A1 A2 B ZN
.ENDS

.SUBCKT 0AI21D0BWP7T A1 A2 B ZN
.ENDS

.SUBCKT 0AI211D2BWP7T A1 A2 B C ZN
.ENDS

.SUBCKT 0AI211D1BWP7T A1 A2 B C ZN
.ENDS
```

Texto plano ▾ Anchura del tabulador: 8 ▾ Ln 519, Col 9 ▾ INS

Figura 52: Error en archivo *Runset 1*

debe resaltar que los errores repentinos se deben a no tener todos los documentos adecuados que se han descrito a lo largo del documento. Empezando por no tener los archivos correctos en la salidas de los equipos de síntesis física y síntesis lógica, que como resultado puede llevar

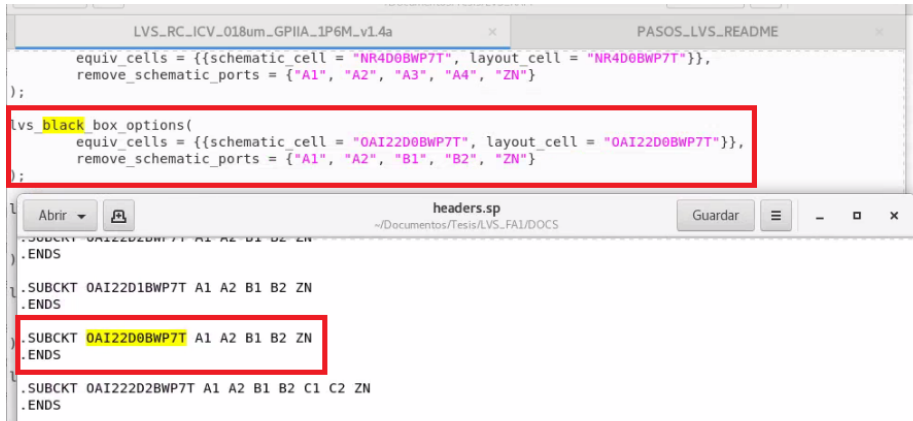
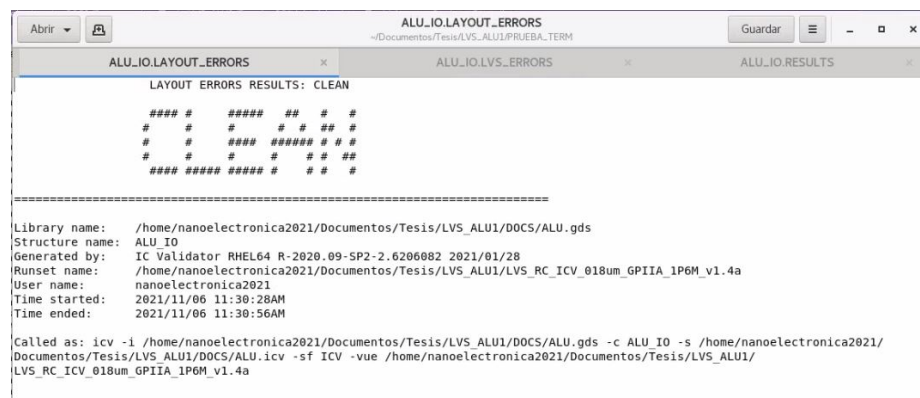


Figura 53: Error en archivo *Runset* 1

a exportar un archivo ICV que tiene errores, como también importar una librería en *Custom Compiler* que contiene errores en el *layout*. Cabe mencionar las repetidas ocasiones en las que el archivo *Runset* no se encontraba correcto, al tener diferencias en las *top cells* o no contar con todas las celdas referenciadas como *black boxes*. Luego de pasar por estos errores y solventarlos, solo se deben seguir las instrucciones descritas en la sección 9.1 y ayudarse de las imágenes para no tener complicaciones.

Como apoyo general, pueden consultarse todas las guías de usuarios de los diferentes programas y herramientas empleadas encontradas en [8], [9], [10], [11], y [12].

Luego de solventar los problemas descritos en el capítulo anterior, se obtuvieron los resultados esperados en todos los circuitos empleados. A continuación se muestran los archivos de salida que se revisaron luego de ejecutar LVS a través de la terminal, seguido por los resultados que se obtienen de *Custom Compiler*.



```
ALU_IO.LAYOUT_ERRORS
~/Documentos/Tesis/LVS_ALU1/PRUEBA_TERM

ALU_IO.LAYOUT_ERRORS x ALU_IO.LVS_ERRORS x ALU_IO.RESULTS x
LAYOUT ERRORS RESULTS: CLEAN
#####
# # # # #
# # # # #
# # # # #
# # # # #
#####

-----
Library name: /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/DOCS/ALU.gds
Structure name: ALU_IO
Generated by: IC Validator RHEL64 R-2020.09-SP2-2.6206082 2021/01/28
Runset name: /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a
User name: nanoelectronica2021
Time started: 2021/11/06 11:30:28AM
Time ended: 2021/11/06 11:30:56AM

Called as: icv -i /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/DOCS/ALU.gds -c ALU_IO -s /home/nanoelectronica2021/
Documentos/Tesis/LVS_ALU1/DOCS/ALU.icv -sf ICV -vue /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/
LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a
```

Figura 54: *Layout Clean* desde terminal

En la Figura #54 se muestra el resultado del *layout* del circuito. Como se puede observar, el resultado es *CLEAN*, lo que significa que el diseño del *layout* no contiene errores o está limpio.

En la Figura #55 se muestra el resultado de la verificación LVS. Se puede observar un *PASS*, lo que significa que el LVS fue satisfactorio y paso los requerimientos definidos.

En la Figura #56 se pueden observar ambos resultados anteriormente descritos. Esto con el fin de tener un consolidado de ambos resultados.

```

ALU_IO.LVS_ERRORS
~/Documentos/Tesis/LVS_ALU1/PRUEBA_TERM
Guardar

ALU_IO.LAYOUT_ERRORS | ALU_IO.LVS_ERRORS | ALU_IO.RESULTS
+-----+-----+-----+
| ICV_Compare LVS Comparison Report |
+-----+-----+-----+

ICV_Compare (R) Hierarchical Layout Vs. Schematic
RHEL64 R-2020.09-SP2-2.6206082 2021/01/28
Copyright (C) Synopsys, Inc. All rights reserved.

-----
LVS error file = ALU_IO.LVS_ERRORS
Layout error file = ALU_IO.LAYOUT_ERRORS
Schematic netlist = /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/PRUEBA_TERM/ALU_IO.sch.out
Layout netlist = /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/PRUEBA_TERM/ALU_IO.net
Runset file = /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a
Working directory = /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/PRUEBA_TERM
Compare directory = run_details/compare
Compare start time = 2021-11-06 11:30:41

-----
Final comparison result:PASS

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

TOP equivalence point:
[ALU_IO, ALU_ID]

Comparison summary
37 Successful blackbox cells
0 Failed blackbox cells

1 Successful equivalence points
0 Failed equivalence points

TOP-level Post-compare summary (* = unmatched devices, nets or ports):

Matched Unmatched Unmatched Instance types
schematic layout [schematic, layout]
-----
1 0 0 BLACK_BOX[AN2D0BWP7T, AN2D0BWP7T]
1 0 0 BLACK_BOX[A0I221D0BWP7T, A0I221D0BWP7T]
-----

```

Figura 55: LVS Pass desde terminal

```

ALU_IO.RESULTS
~/Documentos/Tesis/LVS_ALU1/PRUEBA_TERM
Guardar

ALU_IO.LAYOUT_ERRORS | ALU_IO.LVS_ERRORS | ALU_IO.RESULTS
LVS Compare Results: PASS

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

DRC and Extraction Results: CLEAN

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

=====

ICV Execution

IC Validator

Version R-2020.09-SP2-2 for linux64 - Jan 28, 2021 cl#6206082

Copyright (c) 1996 - 2021 Synopsys, Inc.
This software and the associated documentation are proprietary to Synopsys,
Inc. This software may only be used in accordance with the terms and conditions
of a written license agreement with Synopsys, Inc. All other use, reproduction,
or distribution of this software is strictly prohibited.

Called as: icv -i /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/DOCS/ALU.gds -c ALU_IO -s /home/nanoelectronica2021/
Documentos/Tesis/LVS_ALU1/DOCS/ALU.icv -sf ICV -vue /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/
LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a

-----

User name: nanoelectronica2021
Layout format: GDSII
Input file name: /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/DOCS/ALU.gds
Top cell name: ALU_IO
Time started: 2021/11/06 11:29:52AM
Time ended: 2021/11/06 11:30:56AM
-----

```

Figura 56: Resultado completo de LVS desde terminal

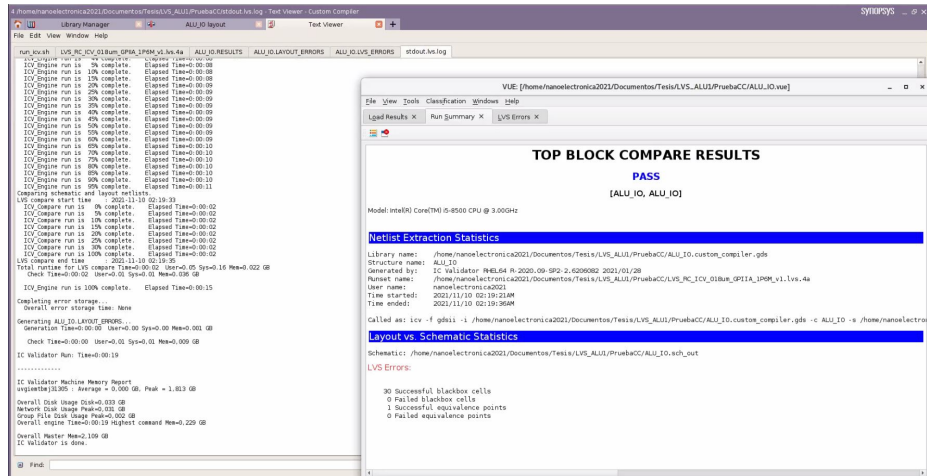


Figura 57: LVS Pass desde Custom Compiler

En la Figura #57 se puede observar la interfaz gráfica que utiliza Custom Compiler para presentar los resultados de LVS. Este representa el resultado satisfactorio de LVS, tal como se mostró en la Figura #55.

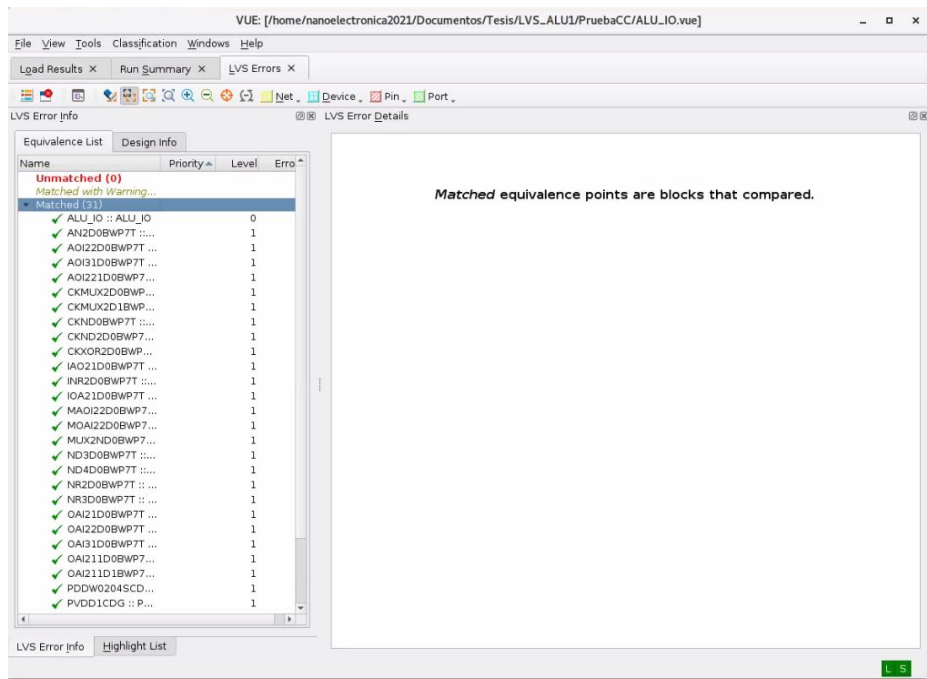


Figura 58: Resultado de LVS desde Custom Compiler

Finalmente, en la Figura #58 se puede observar con mayor detalle, la compatibilidad de las celdas utilizadas en este circuito a través de la interfaz gráfica de Custom Compiler, siendo esto muy parecido al resultado mostrado en la Figura #56. Cabe mencionar que los tres archivos que se mencionan, fueron generados al ejecutar LVS en una terminal, también se generan al ejecutar LVS en Custom Compiler. A continuación se presentan los resultados de todos los circuitos, tanto los generados desde la terminal como por Custom Compiler.

11.1. NOT

En las Figuras #59 y #60 se muestran los resultados de la compuerta *NOT*, al ejecutar LVS en la terminal de comandos y a través de *Custom Compiler* respectivamente.

```
Abrir ~Documentos/Tesis/LVS_NOT1/PRUEBAS_TERM Guardar
PASOS_LVS_README Not_IO.RESULTS
LVS Compare Results: PASS
#### ### ### ###
# # # # #
#### ##### #####
# # # # #
# # # ### ###

-----

DRC and Extraction Results: CLEAN

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

-----

ICV Execution

-----

IC Validator

Version R-2020.09-SP2-2 for linux64 - Jan 28, 2021 cl#6206082

Copyright (c) 1996 - 2021 Synopsys, Inc.
This software and the associated documentation are proprietary to Synopsys,
Inc. This software may only be used in accordance with the terms and conditions
of a written license agreement with Synopsys, Inc. All other use, reproduction,
or distribution of this software is strictly prohibited.

Called as: icv -i /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/DOCS/Not.gds -c Not_IO -s /home/
nanoelectronica2021/Documentos/Tesis/LVS_NOT1/DOCS/NOT.icv -sf ICV -vue /home/nanoelectronica2021/
Documentos/Tesis/LVS_NOT1/LVS_RC_ICV_018um_GPIIA_1PGM.v1.4a

-----

User name: nanoelectronica2021
Layout format: GDSII
Input file name: /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/DOCS/Not.gds
Top cell name: Not_IO
Time started: 2021/10/30 02:31:12PM
Time ended: 2021/10/30 02:31:24PM

-----

Texto plano Anchura del tabulador: 8 Ln 1, Col 1 INS
```

Figura 59: LVS de compuerta *NOT* a través de terminal

```
VUE: [/home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.vue]
File View Tools Classification Windows Help
Load Results X Run Summary X LVS Errors X

TOP BLOCK COMPARE RESULTS

PASS

[Not_IO, Not_IO]

Model: intel(R) Core(TM) i5-8500 CPU @ 3.00GHz

Netlist Extraction Statistics
Library name: /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.custom_compiler.gds
Structure name: Not_IO
Generated by: IC Validator RHEL64 R-2020.09-SP2-2.6206082 2021/01/28
Runset name: /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/PRUEBAS_CC/LVS_RC_ICV_018um_GPIIA_1PGM.v1.Lvs.4a
User name: nanoelectronica2021
Time started: 2021/11/11 12:25:57AM
Time ended: 2021/11/11 12:26:06AM

Called as: icv -f gdsii -i /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.custom_compiler.gds -c Not_IO -s /home/nanoelect

Layout vs. Schematic Statistics
Schematic: /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.sch_out

LVS Errors:

6 Successful blackbox cells
0 Failed blackbox cells
1 Successful equivalence points
0 Failed equivalence points
```

Figura 60: LVS de compuerta *NOT* a través de *Custom Compiler*

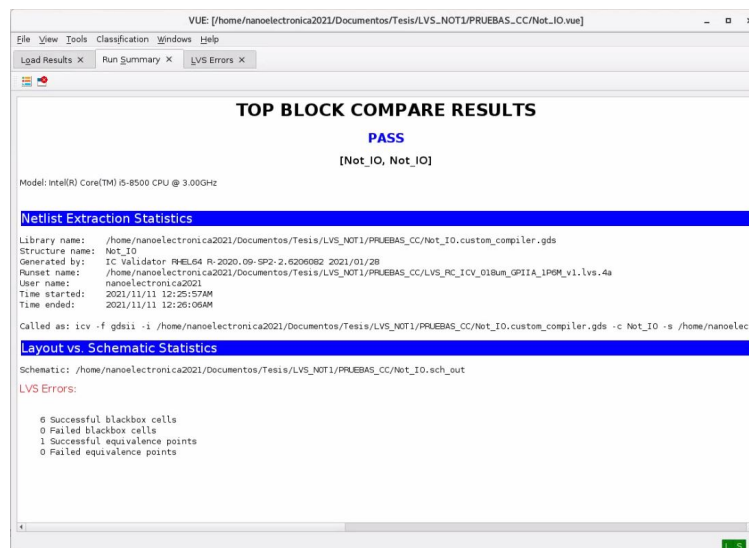
11.2. XOR

En las Figuras #61 y #62 se muestran los resultados de la compuerta *XOR*, al ejecutar LVS en la terminal de comandos y a través de *Custom Compiler* respectivamente.



```
Abrir ~ Documentos/Tesis/LVS_NOT1/PRUEBAS_TERM Guardar
PASOS_LVS_README Not_IO.RESULTS
LVS Compare Results: PASS
#### ## ## ## ##
# # # # #
#### ##### #####
# # # # #
# # # ## ##
-----
DRC and Extraction Results: CLEAN
#### # ##### ## # #
# # # # # # # #
# # ##### ##### # #
# # # # # # # #
#### ##### ##### # # #
-----
ICV Execution
-----
IC Validator
Version R-2020.09-SP2-2 for linux64 - Jan 28, 2021 cl#6206082
Copyright (c) 1996 - 2021 Synopsys, Inc.
This software and the associated documentation are proprietary to Synopsys,
Inc. This software may only be used in accordance with the terms and conditions
of a written license agreement with Synopsys, Inc. All other use, reproduction,
or distribution of this software is strictly prohibited.
Called as: icv -i /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/DOCS/Not.gds -c Not_IO -s /home/
nanoelectronica2021/Documentos/Tesis/LVS_NOT1/DOCS/NOT.icv -sf ICV -vue /home/nanoelectronica2021/
Documentos/Tesis/LVS_NOT1/LVS_RC_ICV_018um_GPIIA_1PGM_v1.4a
-----
User name: nanoelectronica2021
Layout format: GDSII
Input file name: /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/DOCS/Not.gds
Top cell name: Not_IO
Time started: 2021/10/30 02:31:12PM
Time ended: 2021/10/30 02:31:24PM
-----
Texto plano Anchura del tabulador: 8 Ln 1, Col 1 INS
```

Figura 61: LVS de compuerta *NOT* a través de terminal



```
VUE: [/home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.vue]
File View Tools Classification Windows Help
Load Results X Run Summary X LVS Errors X
TOP BLOCK COMPARE RESULTS
PASS
[Not_IO, Not_IO]
Model: intel(R) Core(TM) i5-8500 CPU @ 3.00GHz
Netlist Extraction Statistics
Library name: /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.custom_compiler.gds
Structure name: Not_IO
Generated by: IC Validator RHEL64 R-2020.09-SP2-2.6206082 2021/01/28
Runset name: /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/PRUEBAS_CC/LVS_RC_ICV_018um_GPIIA_1PGM_v1.Lvs.4a
User name: nanoelectronica2021
Time started: 2021/11/11 12:25:57AM
Time ended: 2021/11/11 12:26:06AM
Called as: icv -f gdsii -i /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.custom_compiler.gds -c Not_IO -s /home/nanolect
Layout vs. Schematic Statistics
Schematic: /home/nanoelectronica2021/Documentos/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.sch_out
LVS Errors:
6 Successful blackbox cells
0 Failed blackbox cells
1 Successful equivalence points
0 Failed equivalence points
```

Figura 62: LVS de compuerta *NOT* a través de *Custom Compiler*

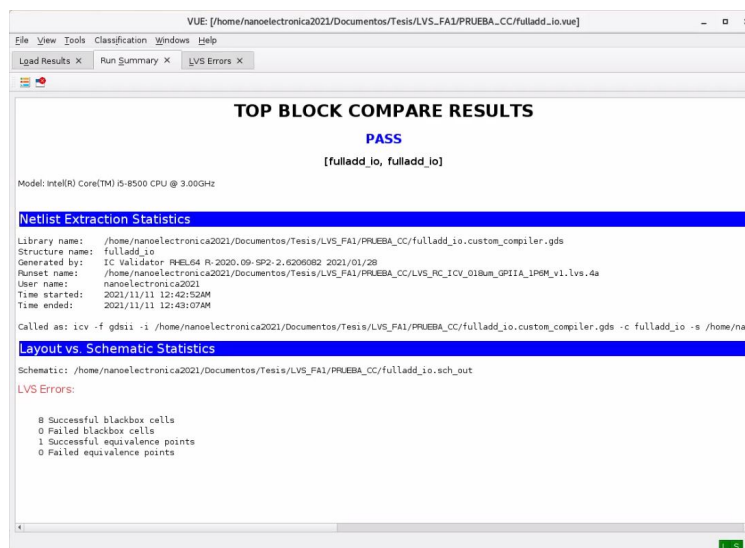
11.3. Full Adder

En las Figuras #63 y #64 se muestran los resultados del circuito *Full Adder*, al ejecutar LVS en la terminal de comandos y a través de *Custom Compiler* respectivamente.



```
fulladd_io.RESULTS
PASOS_LVS_README
fulladd_io.RESULTS
LVS Compare Results: PASS
#### ## ## ## ##
# # # # #
#### ##### #####
# # # # #
# # # ## ##
-----
DRC and Extraction Results: CLEAN
#### # ##### ## # #
# # # # # # #
# # ##### ##### # #
# # # # # # #
#### ##### ##### # # #
-----
ICV Execution
-----
IC Validator
Version R-2020.09-SP2-2 for linux64 - Jan 28, 2021 cl#6206082
Copyright (c) 1996 - 2021 Synopsys, Inc.
This software and the associated documentation are proprietary to Synopsys, Inc. This software may only be used in accordance with the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, or distribution of this software is strictly prohibited.
Called as: icv -i /home/nanoelectronica2021/Documentos/Tesis/LVS_FA1/DOCS/FA.gds -c fulladd_io -s /home/nanoelectronica2021/Documentos/Tesis/LVS_FA1/DOCS/FA.icv -sf ICV -vue /home/nanoelectronica2021/Documentos/Tesis/LVS_FA1/LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a
-----
User name: nanoelectronica2021
Layout format: GDSII
Input file name: /home/nanoelectronica2021/Documentos/Tesis/LVS_FA1/DOCS/FA.gds
Top cell name: fulladd_io
Time started: 2021/10/27 09:12:11PM
Time ended: 2021/10/27 09:12:56PM
-----
Texto plano | Anchura del tabulador: 8 | Ln 10, Col 1 | INS
```

Figura 63: LVS de *Full Adder* a través de terminal



```
VUE: [/home/nanoelectronica2021/Documentos/Tesis/LVS_FA1/PRUEBA_CC/fulladd_io.vue]
File View Tools Classification Windows Help
Load Results X Run Summary X LVS Errors X
TOP BLOCK COMPARE RESULTS
PASS
[fulladd_io, fulladd_io]
Model: intel(R) Core(TM) i5-8500 CPU @ 3.00GHz
Netlist Extraction Statistics
Library name: /home/nanoelectronica2021/Documentos/Tesis/LVS_FA1/PRUEBA_CC/fulladd_io.custom_compiler.gds
Structure name: fulladd_io
Generated by: IC Validator PHEL64 R-2020.09-SP2-2.6206082 2021/01/28
Runset name: /home/nanoelectronica2021/Documentos/Tesis/LVS_FA1/PRUEBA_CC/LVS_RC_ICV_018um_GPIIA_1P6M_v1.lvs.4a
User name: nanoelectronica2021
Time started: 2021/11/11 12:42:52AM
Time ended: 2021/11/11 12:43:07AM
Called as: icv -f gdsii -i /home/nanoelectronica2021/Documentos/Tesis/LVS_FA1/PRUEBA_CC/fulladd_io.custom_compiler.gds -c fulladd_io -s /home/nanoelectronica2021/Documentos/Tesis/LVS_FA1/PRUEBA_CC/fulladd_io.sch_out
Layout vs. Schematic Statistics
Schematic: /home/nanoelectronica2021/Documentos/Tesis/LVS_FA1/PRUEBA_CC/fulladd_io.sch_out
LVS Errors:
8 Successful blackbox cells
0 Failed blackbox cells
1 Successful equivalence points
0 Failed equivalence points
```

Figura 64: LVS de *Full Adder* a través de *Custom Compiler*

11.4. ALU

En las Figuras #65 y #66 se muestran los resultados de una *ALU*, al ejecutar LVS en la terminal de comandos y a través de *Custom Compiler* respectivamente.

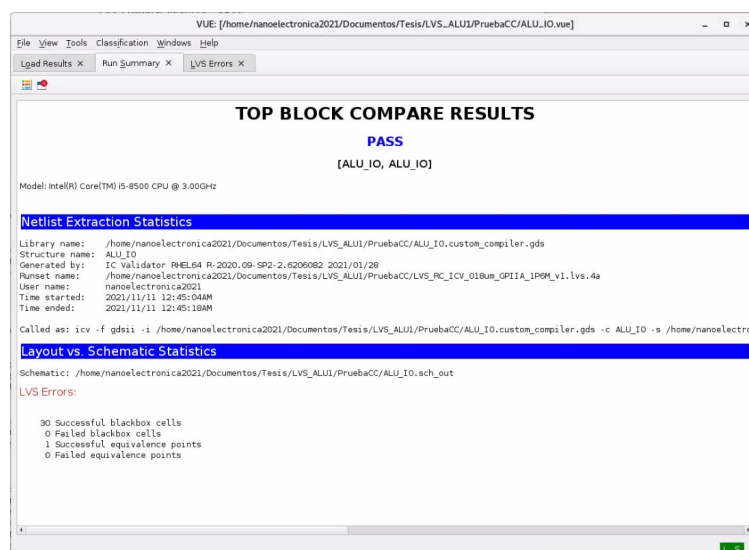


```
Abrir Guardar
~/Documentos/Tesis/LVS_ALU/PRUEBA_TERM
PASOS_LVS_README fulladd_io.RESULTS ALU.IO.RESULTS
LVS Compare Results: PASS
#### ## ## ##
# # # # #
#### ##### #####
# # # # #
# # # ## ##

-----
DRC and Extraction Results: CLEAN
#### # ##### ## # #
# # # # # # #
# # ##### ##### # #
# # # # # # #
#### ##### ##### # # #

-----
ICV Execution
-----
IC Validator
Version R-2020.09-SP2-2 for linux64 - Jan 28, 2021 cl#6206082
Copyright (c) 1996 - 2021 Synopsys, Inc.
This software and the associated documentation are proprietary to Synopsys, Inc. This software may only be used in accordance with the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, or distribution of this software is strictly prohibited.
Called as: icv -i /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/DOCS/ALU.gds -c ALU_IO -s /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/DOCS/ALU.icv -sf ICV -vue /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/LVS_RC_ICV_018um_6PIIA_1P6M_v1.4a
-----
User name: nanoelectronica2021
Layout format: GDSII
Input file name: /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/DOCS/ALU.gds
Top cell name: ALU_IO
Time started: 2021/11/06 11:29:52AM
Time ended: 2021/11/06 11:30:56AM
-----
Texto plano Anchura del tabulador: 8 Ln 1, Col 1 INS
```

Figura 65: LVS de *ALU* a través de terminal



```
VUE: [/home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/PruebaCC/ALU_IO.vue]
File View Tools Classification Windows Help
Lvsd Results X Run Summary X Lvs Errors X
TOP BLOCK COMPARE RESULTS
PASS
[ALU_IO, ALU_IO]
Model: Intel(R) Core(TM) i5-8500 CPU @ 3.00GHz
Netlist Extraction Statistics
Library name: /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/PruebaCC/ALU_IO.custom_compiler.gds
Structure name: ALU_IO
Generated by: IC Validator RHEL64 R-2020.09-SP2-2.6206082 2021/01/28
Runset name: /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/PruebaCC/LVS_RC_ICV_018um_6PIIA_1P6M_v1.lvs.4a
User name: nanoelectronica2021
Time started: 2021/11/11 12:45:04AM
Time ended: 2021/11/11 12:45:18AM
Called as: icv -f gdsii -i /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/PruebaCC/ALU_IO.custom_compiler.gds -c ALU_IO -s /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/PruebaCC/ALU_IO.sch_out
Layout vs. Schematic Statistics
Schematic: /home/nanoelectronica2021/Documentos/Tesis/LVS_ALU1/PruebaCC/ALU_IO.sch_out
LVS Errors:
30 Successful blackbox cells
0 Failed blackbox cells
1 Successful equivalence points
0 Failed equivalence points
```

Figura 66: LVS de *ALU* a través de *Custom Compiler*

11.5. 4 Bit Counter

En las Figuras #67 y #68 se muestran los resultados de un 4 bit Counter, al ejecutar LVS en la terminal de comandos y a través de *Custom Compiler* respectivamente.



```
Abrir [icon] counter4IO.RESULTS [icon] Guardar [icon] [icon] [icon] x
~/Documentos/Tesis/LVS_COUNTER/PRUEBA_TERM

COUNTER.icv x counter4IO.RESULTS x

LVS Compare Results: PASS

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

-----

DRC and Extraction Results: CLEAN

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

-----

ICV Execution

-----

IC Validator

Version R-2020.09-SP2-2 for linux64 - Jan 28, 2021 cl#6206082

Copyright (c) 1996 - 2021 Synopsys, Inc.
This software and the associated documentation are proprietary to Synopsys,
Inc. This software may only be used in accordance with the terms and conditions
of a written license agreement with Synopsys, Inc. All other use, reproduction,
or distribution of this software is strictly prohibited.

Called as: icv -i /home/nanoelectronica2021/Documentos/Tesis/LVS_COUNTER/DOCs/counter.gds -c
counter4IO -s /home/nanoelectronica2021/Documentos/Tesis/LVS_COUNTER/DOCs/COUNTER.icv -sf ICV -vue /
home/nanoelectronica2021/Documentos/Tesis/LVS_COUNTER/LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a

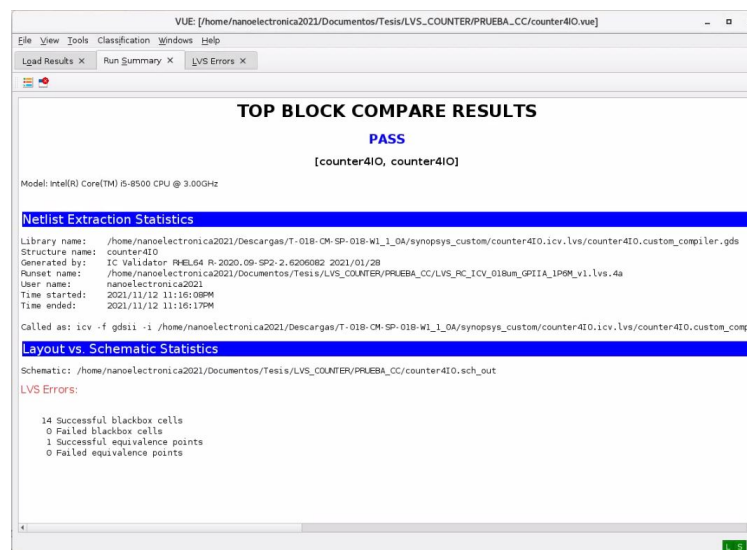
-----

User name: nanoelectronica2021
Layout format: GDSII
Input file name: /home/nanoelectronica2021/Documentos/Tesis/LVS_COUNTER/DOCs/counter.gds
Top cell name: counter4IO
Time started: 2021/09/13 10:17:38PM
Time ended: 2021/09/13 10:17:48PM

-----

Texto plano | Anchura del tabulador: 8 | Ln 1, Col 1 | INS
```

Figura 67: LVS de Counter de 4 bits a través de terminal



```
VUE: [home/nanoelectronica2021/Documentos/Tesis/LVS_COUNTER/PRUEBA_CC/counter4IO.vue]
File View Tools Classification Windows Help
Load Results x Run Summary x LVS Errors x

TOP BLOCK COMPARE RESULTS
PASS
[counter4IO, counter4IO]

Model: Intel(R) Core(TM) i5-8500 CPU @ 3.00GHz

Netlist Extraction Statistics
Library name: /home/nanoelectronica2021/Descargas/T-018-0M-SP-018-W1_1_0A/synopsys_custom/counter4IO.icv.lvs/counter4IO.custom_compiler.gds
Structure name: counter4IO
Generated by: IC Validator RHEL64 R-2020.09-SP2-2.6206082 2021/01/28
Runset name: /home/nanoelectronica2021/Documentos/Tesis/LVS_COUNTER/PRUEBA_CC/LVS_RC_ICV_018um_GPIIA_1P6M_v1.lvs.4a
User name: nanoelectronica2021
Time started: 2021/11/12 11:16:08PM
Time ended: 2021/11/12 11:16:17PM
Called as: icv -f gdsii -i /home/nanoelectronica2021/Descargas/T-018-0M-SP-018-W1_1_0A/synopsys_custom/counter4IO.icv.lvs/counter4IO.custom_comp

Layout vs. Schematic Statistics
Schematic: /home/nanoelectronica2021/Documentos/Tesis/LVS_COUNTER/PRUEBA_CC/counter4IO.sch_out

LVS Errors:
14 Successful blackbox cells
0 Failed blackbox cells
1 Successful equivalence points
0 Failed equivalence points
```

Figura 68: LVS de Counter de 4 bits a través de Custom Compiler

11.6. RAM Memmory

En las Figuras #69 y #70 se muestran los resultados de una memoria *RAM*, al ejecutar LVS en la terminal de comandos y a través de *Custom Compiler* respectivamente.

```
ram_IO.RESULTS
~|Documents/Tesis/LVS_RAM/PRUEBAS_TERM

LVS Compare Results: PASS

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

-----

DRC and Extraction Results: CLEAN

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

=====

ICV Execution

-----

IC Validator

Version R-2020.09-SP2-2 for linux64 - Jan 28, 2021 cl#6206082

Copyright (c) 1996 - 2021 Synopsys, Inc.
This software and the associated documentation are proprietary to Synopsys,
Inc. This software may only be used in accordance with the terms and conditions
of a written license agreement with Synopsys, Inc. All other use, reproduction,
or distribution of this software is strictly prohibited.

Called as: icv -i /home/nanoelectronica2021/Documents/Tesis/LVS_RAM/DOCS/RAM.gds -c ram_IO -s /home/
nanoelectronica2021/Documents/Tesis/LVS_RAM/DOCS/RAM.icv -sf ICV -vue /home/nanoelectronica2021/
Documents/Tesis/LVS_RAM/LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a

-----

User name: nanoelectronica2021
Layout format: GDSII
Input file name: /home/nanoelectronica2021/Documents/Tesis/LVS_RAM/DOCS/RAM.gds
Top cell name: ram_IO
Time started: 2021/11/11 01:31:39AM
Time ended: 2021/11/11 01:31:52AM

-----

Texto plano | Anchura del tabulador: 8 | Ln 1, Col 1 | INS
```

Figura 69: LVS de memoria *RAM* a través de terminal

```
VUE: [/home/nanoelectronica2021/Documents/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.vue]

File View Tools Classification Windows Help
Load Results X Run Summary X LVS Errors X

TOP BLOCK COMPARE RESULTS

PASS

[Not_IO, Not_IO]

Model: intel(R) Core(TM) i5-8500 CPU @ 3.00GHz

Netlist Extraction Statistics
Library name: /home/nanoelectronica2021/Documents/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.custom_compiler.gds
Structure name: Not_IO
Generated by: IC Validator RHEL64 R-2020.09-SP2-2.6206082 2021/01/28
Runset name: /home/nanoelectronica2021/Documents/Tesis/LVS_NOT1/PRUEBAS_CC/LVS_RC_ICV_018um_GPIIA_1P6M_v1.lv4.a
User name: nanoelectronica2021
Time started: 2021/11/11 12:26:57AM
Time ended: 2021/11/11 12:26:06AM

Called as: icv -f gdsii -i /home/nanoelectronica2021/Documents/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.custom_compiler.gds -c Not_IO -s /home/nanoelect

Layout vs. Schematic Statistics
Schematic: /home/nanoelectronica2021/Documents/Tesis/LVS_NOT1/PRUEBAS_CC/Not_IO.sch_out

LVS Errors:

6 Successful blackbox cells
0 Failed blackbox cells
1 Successful equivalence points
0 Failed equivalence points
```

Figura 70: LVS de memoria *RAM* a través de *Custom Compiler*

Proyecto final: El Gran Jaguar

Como etapa final, el grupo de diseño definió que el reto final, sería diseñar un circuito que cumpliera una función más compleja y que este cumpliera todas las etapas en el flujo de diseño. Iniciando en la síntesis lógica y física y pasando satisfactoriamente todas las verificaciones y pruebas siguientes hasta estar listo para ser enviado a su producción. Al proyecto se le llamó *El Gran Jaguar*. El circuito está diseñado para cumplir dos procesos. El primero es un *Ring Oscillator*, el cual es un circuito adicional agregado al *chip* completo como un extra al funcionamiento principal. El segundo y el más completo, es un circuito que tiene 8 salidas, las cuales, controladas mediante un contador, para que a medida que el contador aumentara se enviaran configuraciones distintas en los pines de salida. Cada *set* de bits es específico para una letra del abecedario. El contador llegaría al número de caracteres necesarios para completar una frase completa, la cual llegaría a un programa en un arduino que traduciría esta configuración de los 8 *bits* a los caracteres *ASCII* para que fuesen reproducidos como un audio completo de la frase definida.

A continuación se muestran los resultados del proceso de LVS ejecutado para *El Gran Jaguar*, desde los documentos generados por los equipos de síntesis física y síntesis lógica, hasta los resultados finales al completar LVS.

12.1. Documentos de Síntesis Lógica

En la Figura #71 se pueden observar los 3 archivos generados por el equipo de Síntesis Lógica. Estos son el formato de archivos descritos con anterioridad.

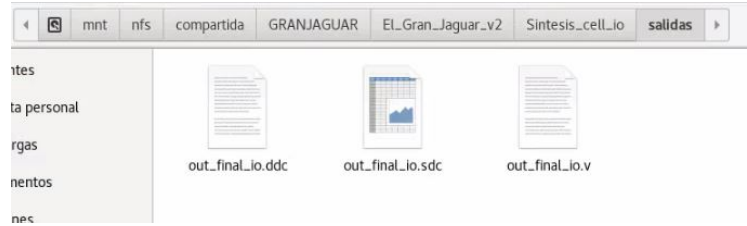


Figura 71: Documentos de salida generados en Síntesis Lógica

12.2. Documentos de Síntesis Física

En la Figura #72 se pueden observar los diferentes archivos generados por el equipo de Síntesis Física. El más relevante para el proceso de LVS es la carpeta en formato NDM.

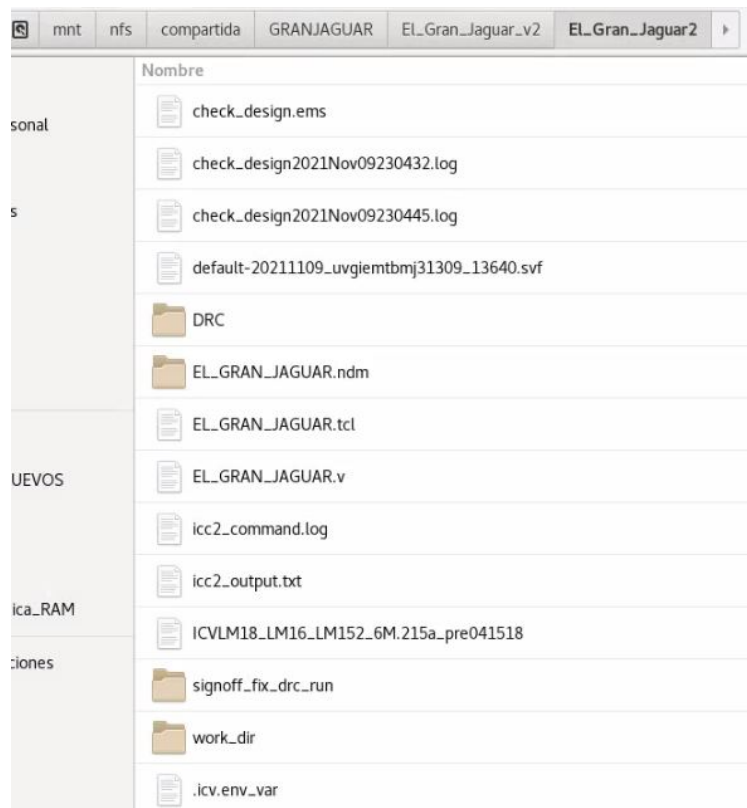


Figura 72: Documentos de salida generados en Síntesis Física

12.3. Documentos generados

En la Figura #73, resaltados en rojo, se encuentran los archivos en formato ICV y GDS, que son los primeros archivos generados en el proceso de LVS.

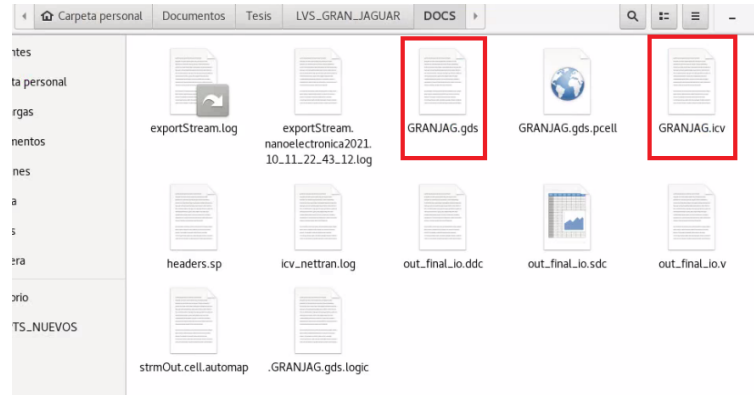


Figura 73: Archivos de ICV y GDS generados

En la Figura #74 se resalta en rojo el archivo *Runset*, ya que este se debe modificar específicamente para cada circuito.

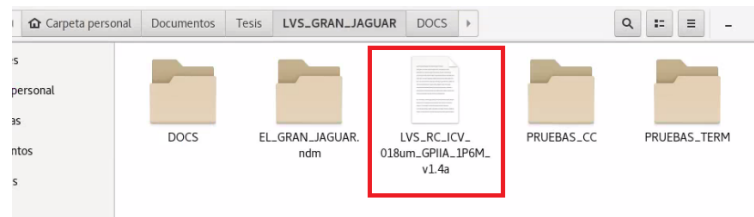


Figura 74: Archivo de *Runset* modificado

En la Figura #75 se pueden observar los diferentes archivos de salida generados al ejecutar LVS a través de líneas de comandos en una terminal.

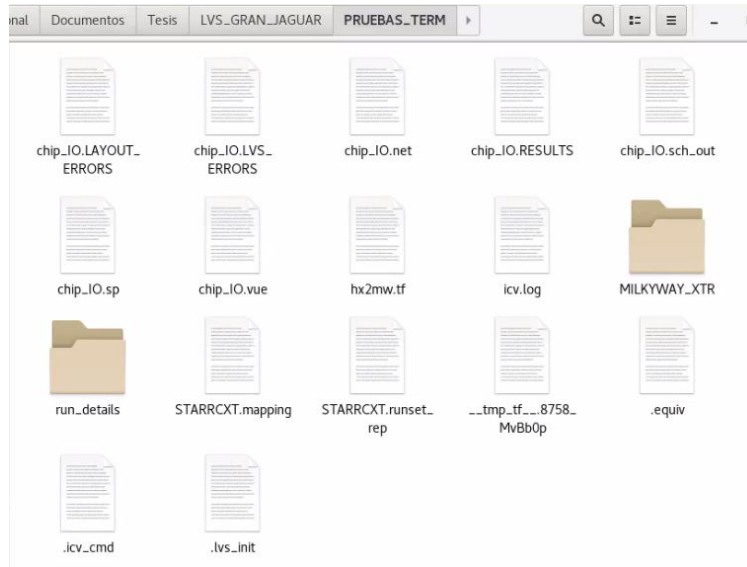


Figura 75: Archivos de salida al ejecutar LVS en terminal

En la Figura #76 se pueden observar los diferentes archivos de salida generados al ejecutar LVS mediante *Custom Compiler*.

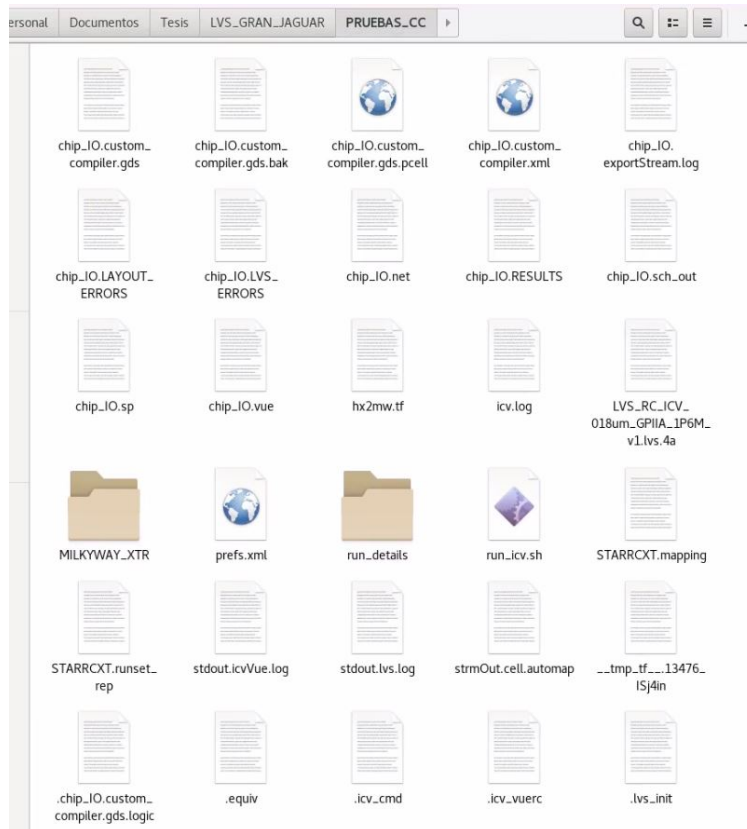


Figura 76: Archivos de salida al ejecutar LVS en *Custom Compiler*

12.4. Vista de *Layout*

En la Figura #77 se muestra la vista de *Layout* en *Custom Compiler* del circuito completo del proyecto final.

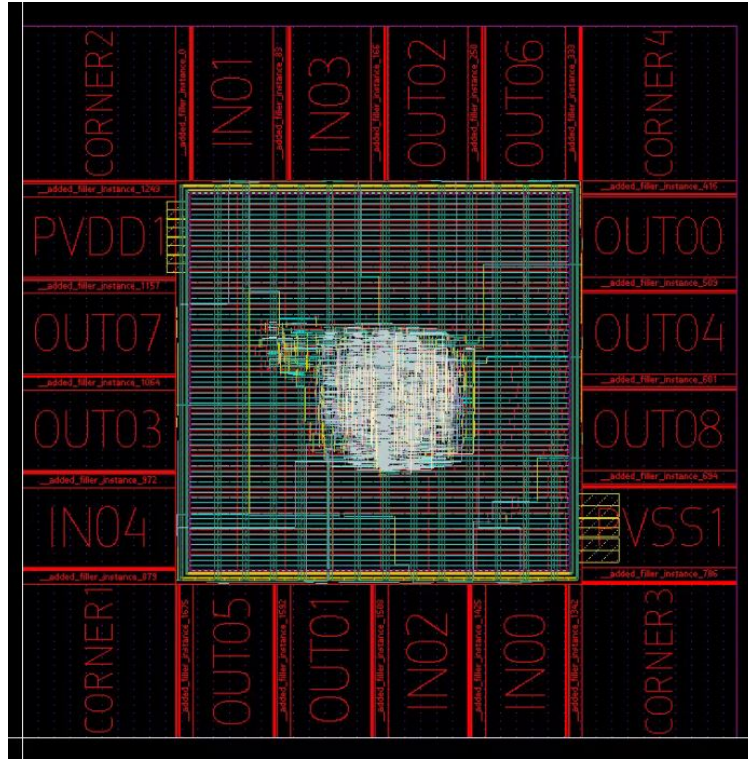


Figura 77: LVS de memoria *RAM* a través de *Custom Compiler*

12.5. Resultados de LVS

En la Figura #78 se marcan en rojo las ubicaciones en donde se guardaron los archivos de salida al ejecutar LVS, mediante *Custom Compiler* y a través de terminal respectivamente.

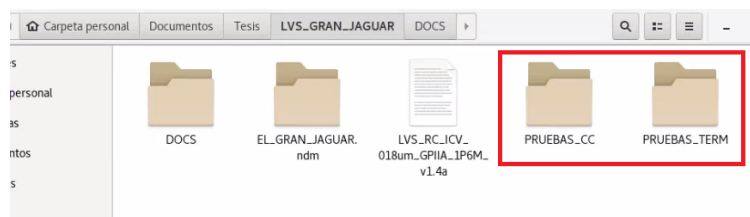


Figura 78: Ubicaciones para guardar archivos generados por LVS

En la Figura #79 se muestra el resultado final de la ejecución de LVS a través de la terminal. Como se puede observar, el resultado es satisfactorio.

```

chip_IO.RESULTS
~\Documentos\Tesis\LVS_GRAN_JAGUAR\PRUEBAS_TERM Guardar
PASOS_LVS_README chip_IO.RESULTS
LVS Compare Results: PASS
#####
# # # # #
#####
# # # # #
#####
# # # # #
#####

-----

DRC and Extraction Results: CLEAN
#####
# # # # #
# # # # #
# # # # #
#####
#####

-----

ICV Execution
-----

IC Validator
Version R-2020.09-SP2-2 for linux64 - Jan 28, 2021 cl#6206082
Copyright (c) 1996 - 2021 Synopsys, Inc.
This software and the associated documentation are proprietary to Synopsys,
Inc. This software may only be used in accordance with the terms and conditions
of a written license agreement with Synopsys, Inc. All other use, reproduction,
or distribution of this software is strictly prohibited.

Called as: icv -i /home/nanoelectronica2021/Documentos/Tesis/LVS_GRAN_JAGUAR/DOCS/GRANJAG.gds -c
chip_IO -s /home/nanoelectronica2021/Documentos/Tesis/LVS_GRAN_JAGUAR/DOCS/GRANJAG.icv -sf ICV -vue /
home/nanoelectronica2021/Documentos/Tesis/LVS_GRAN_JAGUAR/LVS_RC_ICV_018um_GPIIA_1P6M_v1.4a

-----

User name: nanoelectronica2021
Layout format: GDSII
Input file name: /home/nanoelectronica2021/Documentos/Tesis/LVS_GRAN_JAGUAR/DOCS/GRANJAG.gds
Top cell name: chip_IO
Time started: 2021/11/11 12:02:39AM
Time ended: 2021/11/11 12:02:51AM

```

Figura 79: Resultado de LVS a través de Terminal

En la Figura #80 se muestra el resultado final de la ejecución de LVS mediante *Custom Compiler*. Este resultado se presenta con la herramienta *VUE Tool* y al igual que la figura anterior, se observa un resultado satisfactorio.

```

VUE: [home/nanoelectronica2021/Documentos/Tesis/LVS_GRAN_JAGUAR/PRUEBAS_CC/chip_IO.vue]
File View Tools Classification Windows Help
Load Results X Run Summary X LVS Errors X

TOP BLOCK COMPARE RESULTS
PASS
[chip_IO, chip_IO]

Model: intel(R) Core(TM) i5-8500 CPU @ 3.00GHz

Netlist Extraction Statistics
Library name: /home/nanoelectronica2021/Documentos/Tesis/LVS_GRAN_JAGUAR/PRUEBAS_CC/chip_IO.custom_compiler.gds
Structure name: chip_IO
Generated by: IC Validator RHEL64 R-2020.09-SP2-2.6206082 2021/01/28
Punset name: /home/nanoelectronica2021/Documentos/Tesis/LVS_GRAN_JAGUAR/PRUEBAS_CC/LVS_RC_ICV_018um_GPIIA_1P6M_v1.lvs.4a
User name: nanoelectronica2021
Time started: 2021/11/11 12:05:50AM
Time ended: 2021/11/11 12:06:00AM

Called as: icv -f gdsii -i /home/nanoelectronica2021/Documentos/Tesis/LVS_GRAN_JAGUAR/PRUEBAS_CC/chip_IO.custom_compiler.gds -c chip_IO -s /home/

Layout vs. Schematic Statistics
Schematic: /home/nanoelectronica2021/Documentos/Tesis/LVS_GRAN_JAGUAR/PRUEBAS_CC/chip_IO.sch_out

LVS Errors:
47 Successful blackbox cells
0 Failed blackbox cells
1 Successful equivalence points
0 Failed equivalence points

```

Figura 80: Resultado de LVS a través de *Custom Compiler*

- Se logró realizar el proceso de LVS en su totalidad, resolviendo los errores y desafíos encontrados en la marcha, en conjunto con los equipos encargados de las etapas anteriores en el flujo de diseño.
- Se definió paso a paso el proceso para la ejecución de LVS mediante líneas de comando en una terminal para poder implementarlo en un proceso de automatización en un futuro.
- Se realizó una guía para la ejecución adecuada de LVS con las herramientas y programas actuales, con el fin de implementarlo en cualquier circuito a desarrollar en un futuro.
- Gracias a la comunicación efectiva con el equipo de trabajo, se logró trasladar todos los archivos necesarios para continuar las etapas del flujo de diseño.

Para poder llevar a cabo un experimento similar a este o replicarlo, se recomienda de sobremanera revisar toda la documentación anterior que sirvió de base a este trabajo; principalmente [4], en conjunto con los documentos, manuales de usuario, guías, y *release notes* que este incluye, tales como [8], [9], [10], entre otros. Una vez revisada esta documentación, es recomendable leer en su totalidad este trabajo, para lograr obtener la idea general y global del proceso que se debe llevar a cabo y comprender en que parte del flujo de diseño se desarrolla LVS y su función en el.

Dentro de todo el trabajo se hizo mención constante al orden que se debe tener para realizar el proceso de LVS. Esto va de la mano con la facilidad de acceder a los archivos con los que se este trabjando, ya que es más sencillo ubicar archivos en carpetas que mantengan un mismo formato de trabajo y en dónde se puede acceder rápidamente. Con esto en mente, también se recomienda mantener una línea definida para nombrar los distintos archivos que se generan en cada etapa de este proceso, ya que, al tener una forma parecida o similar, es más fácil saber que archivos se necesitan en cada sección, apartado o etapa de trabajo. También ayuda a identificar si algún archivo o carpeta se ha perdido o simplemente no se ha generado, para corregir estos errores de forma precisa.

Es de suma importancia actualizarse en cuanto a novedades que puedan surgir en este proceso. Esto parte de actualizaciones que sufran las herramientas y/o los programas utilizados como también los archivos como tal que son brindados por empresas como TSMC, que apoyan este flujo de diseño y su proceso. Adicionalmente, se recomienda mantener actualizado el software del equipo que se este utilizando, para no tener complicaciones en la ejecución de las librerías y con ello los programas y herramientas.

Finalmente, es altamente recomendado mantener una comunicación efectiva con el resto del equipo de trabajo. Recordar que este proyecto se genera de manera grupal y ninguna etapa es más importante que otra, pero ninguna puede cumplir su función total si no se ejecutan todas a la perfección, ya que, en conjunto, buscan completar el proceso de diseño de un chip y con ello llegar a su fabricación en silicio.

Disfrute del proceso, los errores que parecen incorregibles, que con tan solo comentar una línea se solucionan, y las horas de esfuerzo y dedicación que puede invertir en este y otros trabajos de diseño, porque cuando se consigue obtener el resultado deseado, todo esto pasa a un segundo plano, y la felicidad de haberlo logrado y la satisfacción personal de conseguir el objetivo final es lo que vale.

-
- [1] J. De los Santos, “Diseño de un sumador/restador completo de 32 bits con tecnología CMOS en un proceso de 28 nanómetros usando aplicaciones de diseño de la empresa Synopsys,” UVG, 2014.
 - [2] S. Rubio, “Definición del Flujo de Diseño para Fabricación de un Chip con Tecnología VLSI CMOS,” UVG, 2019, págs. 11-32.
 - [3] L. Nájera, “Implementación de circuitos sintetizados a nivel netlist a partir de un diseño en lenguaje descriptivo de hardware como primer paso en el flujo de diseño de un circuito integrado.,” UVG, 2019.
 - [4] R. Girón, “Etapa de verificación física de Diseño en Silicio vs Esquemático (LVS) en el flujo de diseño para un chip a nanoescala,” UVG, 2020, págs. 15-30.
 - [5] D. Weste N. y Harris, “CMOS VLSI Design : A Circuits and Systems Perspective,” PEARSON, 2015, págs. 634-646.
 - [6] H. Kommuru H. y Mahmoodi, “ASIC Design Flow Tutorial Using Synopsys Tools,” Nano-Electronics 'I&' Computing Research Lab School of Engineering San Francisco State University, 2009, págs. 10-12.
 - [7] I. M. y. J. H. Andrew Khang Jens Lienig, “VLSI Physical Design: From Graph Partitioning to Timing Closure,” Springer, 2011, págs. 7-11.
 - [8] Synopsys, “IC Validator Reference Manual Q-2019.12,” SolvNet, 2020.
 - [9] —, “IC Validator LVS User Guide Q-2019.12,” SolvNet, 2020.
 - [10] —, “IC Validator User Guide Q-2019.12,” SolvNet, 2020.
 - [11] —, “Custom Compiler Environment User Guide S-2021.09,” SolvNet, 2021.
 - [12] —, “IC Compiler II Data Model User Guide S-2021.06,” SolvNet, 2021.

