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 y simulación para la etapa de síntesis lógica

Trabajo de graduación presentado por Elmer Otoniel Torres Garza para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

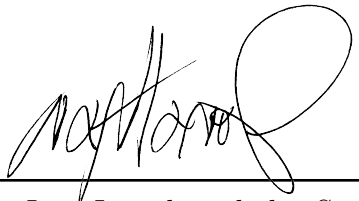
Diseño de un circuito integrado con tecnología de 180 nm usando librerías de diseño de TSMC: Ejecución y simulación para la etapa de síntesis lógica

Trabajo de graduación presentado por Elmer Otoniel Torres Garza para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2022

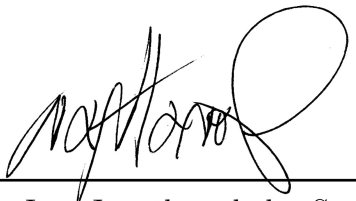
Vo.Bo.:



(f) _____

Ing. Jonathan de los Santos

Tribunal

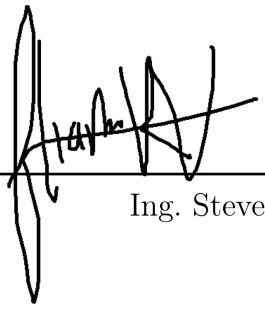


(f) _____

Ing. Jonathan de los Santos

(f) _____

MSc. Carlos Esquit



(f) _____

Ing. Steven Rubio

Fecha de aprobación: Guatemala, 5 de enero de 2022.

Índice

Lista de figuras

Lista de cuadros

Resumen

El presente proyecto se basa en el desarrollo de la etapa de síntesis lógica para llevar a cabo el diseño y fabricación del primer chip elaborado en Guatemala. Esta etapa es la primera de varias que conforman el flujo de diseño con el que se está trabajando el circuito integrado, por lo que es de vital importancia que los resultados obtenidos sean satisfactorios y el flujo no sea interrumpido. El objetivo principal de este proyecto es realizar y verificar que la síntesis lógica cumpla con todos los requisitos necesarios de las siguientes etapas libre de errores.

Para poder realizar la síntesis lógica se utilizó la herramienta de *Design Vision* perteneciente al grupo de herramientas de *Synopsys*. Con esta herramienta somos capaces de realizar la lectura de un archivo HDL y sintetizarlo utilizando librerías que nos provee el fabricante para obtener un diseño que utilice celdas reales fabrícables. Los circuitos que se emplearon para corroborar este proceso fueron las compuertas *Not, Xor, Synchronous Counter, Fulladder, Rom/Ram Memory* y una *ALU*.

Por último se utilizó la herramienta de VCS y el complemento DVE para realizar la verificación del funcionamiento de los diferentes circuitos empleados para el proceso de síntesis, para poder ejecutar simulaciones de los circuitos, se desarrolló un *test bench* el cual simula el comportamiento de las entradas y salidas de los circuitos.

Abstract

This project is based on the development of the logical synthesis stage to carry out the design and manufacture of the first chip made in Guatemala. This stage is the first of several that make up the design flow with which the integrated circuit is being worked, so it is vitally important that the results obtained are satisfactory and that the flow is not interrupted. The main objective of this project is to carry out and verify that the logical synthesis complies with all the necessary requirements of the following stages free of errors.

In order to carry out the logical synthesis, the *Design Vision* tool belonging to the *Synopsys* group of tools was used. With this tool we are able to read an HDL file and synthesize it using libraries provided by the manufacturer to obtain a design that uses real, manufacturable cells. The circuits used to corroborate this process were a *Not, Xor, Synchronous Counter, Fulladder, Rom/Ram Memory* gates and a *ALU*.

Finally, the VCS tool and the DVE complement were used to verify the operation of the different circuits used for the synthesis process, in order to run simulations of the circuits, a *test bench* was developed which simulates the behavior of the inputs and outputs of the circuits.

CAPÍTULO 1

Introducción

Existe un flujo de diseño para la creación del circuito integrado. La primera etapa que se debe elaborar es la de síntesis lógica[1], la cual consiste en la elaboración del circuito en lenguaje HDL para poder llevar a cabo la síntesis de este utilizando las librerías de fabricación TSMC. Para asegurarse que el proceso de síntesis lógica fue correcto es necesario la verificación de funcionamiento del circuito original, antes de introducirlo en el flujo de diseño, así mismo es necesario realizar la verificación para el archivo de salida que contiene el diseño con las celdas sintetizadas y los pines I/O. Esto es necesario para asegurar que el diseño se encuentra libre de errores antes proceder con las otras etapas del flujo.

El presente trabajo está organizado en dos capítulos principales. En el capítulo 7 se encuentra el proceso de síntesis lógica descrito paso a paso con todas las configuraciones necesarias y el porqué de estas, es necesario la revisión de estas debido que las configuraciones pueden variar dependiendo el tipo y tamaño de circuito que tengamos, al final de este se encuentran los *netlist* obtenidos del proceso de síntesis. En el capítulo 8 se encuentra descrito del proceso para realizar la verificación de funcionamiento para ambos circuitos, pre y post síntesis, así mismo se describen las configuraciones necesarias para la creación de los archivos de *test bench* y los resultados para las diferentes compuertas realizadas.

CAPÍTULO 2

Antecedentes

La Universidad del Valle de Guatemala se ha dedicado a invertir recursos en avanzar distintos sectores como la educación, tecnología e innovación en nuestra región del mundo. Prueba de esto son los diferentes proyectos en los cuales la universidad ha sido partícipe todos los años, uno de los últimos proyectos que se realizaron fue la creación de un Cubesat el cual fue capaz de orbitar la tierra durante 211 días. Una de las áreas que se está desarrollando gracias al apoyo de la universidad es la micro y nano electrónica gracias al Ing. Carlos Esquit, el cual cuenta con una maestría en esta rama de la electrónica por lo que es la persona indicada para guiar el avance de estas tecnologías.

Los primeros avances hechos por la universidad en esta área de tecnología empezaron en 2013 cuando se agregaron cursos de VLSI a la malla curricular de estudios, posteriormente en 2014 se logró crear un acuerdo con la empresa Synopsys para poder obtener acceso a una gran variedad de herramientas para el desarrollo de las nuevas tecnologías, estos avances logrados por la universidad permitieron que los alumnos pudieran incursionar en esta rama de la electrónica. Logrando que en 2014 se presentara el primer trabajo de graduación en el área de VLSI por parte de la UVG [2], este trabajo ha servido de guía para que otros estudiantes puedan experimentar con las herramientas que nos brinda Synopsys y así incursionar en estas nuevas tecnologías.

En los últimos años se consiguió el apoyo para realizar la fabricación de un chip a escala nanométrica por la empresa TSMC, la cual es una empresa líder y pionera en la fabricación de nuevos chips a nivel mundial, este apoyo se logró a través del centro de investigación y desarrollo de tecnología nanométrica y digital IMEC, esta tiene como propósito lograr crear equipos talentosos al rededor del mundo que sean capaces de destacar en el sector tecnológico, por lo que la universidad ha invertido recursos para que los estudiantes de electrónica puedan realizar el desarrollo de este chip. Los primeros avances que se dieron para este proyecto se enfocaron en la definición del flujo de diseño[3] y la implementación de circuitos sintetizados a nivel de *netlist* a partir de un diseño en HDL[4], estos avances fueron de gran ayuda para poder a realizar el trabajo presentado en este documento.

CAPÍTULO 3

Justificación

La fabricación de chips utilizando silicio es un proceso de fabricación costoso y complejo, teniendo un costo de varios millones de dólares y el desarrollo toma cerca de un año, por lo que es necesario que los diseños que se desean materializar cumplan con las características que el fabricante provee. El diseño de estos chips suele ser dividido en distintas etapas para poder verificar constantemente si el diseño cumple los requisitos necesarios, este proceso se denomina *Design Flow* (Flujo de Diseño). Compañías como Intel o AMD cuentan con su propio flujo de diseño debido a la gran complejidad de sus diseños, esto les permite dividir el trabajo en distintos módulos para que los ingenieros encargados sean capaces de crear e integrar las distintas partes de los procesadores que se encuentran dentro de todas las computadoras al rededor del mundo.

El objetivo de este trabajo es establecer los parámetros necesarios para el desarrollo de la etapa de *Logic Synthesis*. Esta etapa es el primer paso para poder llevar a cabo el diseño completo de un chip, donde partimos de un código desarrollado en *HDL* hasta la verificación de la síntesis lógica del circuito. Con este trabajo se desea motivar a las personas a indagar sobre el avance de la nanoelectrónica en Guatemala y hacer crecer esta área de trabajo en nuestra región.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Realizar y verificar la etapa de síntesis lógica en el flujo de diseño, cumpliendo con todos los requisitos necesarios para que las siguientes etapas puedan desarrollarse libre de errores.

4.2. Objetivos específicos

- Construir el código HDL con todos los parámetros necesarios para poder ser sintetizado de manera correcta.
- Realizar las simulaciones necesarias empleando la herramienta de VCS para comprobar el funcionamiento correcto del código descriptor del diseño y la síntesis lógica.
- Obtener el diseño del circuito a nivel de *Register Transfer level* (RTL) que cumpla con todos los requisitos necesarios para llegar a la siguiente etapa.
- Poder localizar y mitigar los posibles errores que puedan surgir en la etapa del equipo de LVS, para solucionarlos lo más rápido posible.
- Proveer los archivos e información necesaria al equipo encargado de realizar la síntesis física del diseño.
- Proveer todos los archivos e información necesaria al equipo encargado de realizar la automatización del proceso completo.
- Realizar un manual que contenga instrucciones del uso correcto de todas las herramientas empleadas para la etapa de *Logic Synthesis*.

CAPÍTULO 5

Alcance

El objetivo de esta investigación es poder establecer la etapa de síntesis lógica implementando librerías de fabricación reales de tecnología de 180 nanómetros. Esto se llevará a cabo utilizando los conocimientos previos de los estudiantes que estuvieron en dicho proyecto. Para llevar a cabo la síntesis lógica se utilizará el script de comandos que nos proporcionó la ingeniera Sophia Cardona[1], para el cual únicamente se cambiaran las librerías que este utiliza para llevar a cabo la síntesis.

También se modificará y corregirán los comandos en base a lo que las demás etapas del flujo del diseño nos soliciten, debido que todas las etapas tienen requisitos diferentes es necesario tener una buena comunicación sobre las necesidades de cada equipo. Si se llegase a encontrar alguna forma de optimizar el proceso de síntesis en alguna de las otras etapas se realizarán los cambios necesarios para tener un flujo de diseño óptimo.

Se desea completar la etapa de síntesis lógica satisfactoriamente, para lograr que las demás etapas no tengan ningún inconveniente con los archivos que les brindemos. También se pretende establecer el proceso adecuado para llevar a cabo las pruebas del *netlist* final y confirmar que su funcionamiento lógico es correcto.

CAPÍTULO 6

Marco teórico

6.1. Transistores MOS

El silicio (Si), un material semiconductor, es uno de los componentes esenciales para la creación circuitos integrados. A este material es posible introducirle pequeñas cantidades de impurezas, llamados dopantes, para poder alterar las propiedades del silicio. existen dos tipos de semiconductores que se crean en el silicio, para el primer tipo se utiliza Arsénico como dopante para poder crear semiconductores de *tipo n* estos contiene un electrón libre capaz de trasmitir la corriente, por lo tanto su conductividad es alta. El segundo tipo utiliza Boto como dopante, a diferencia del anterior a este le falta un electrón que le permite actuar como un portador positivo dando lugar a los semiconductores de *tipo p*. Un transistor es creado apilando distintas capas de materiales conductores y aislantes en forma de sándwich, estas estructuras son creadas utilizando procesos químicos que involucran oxidar el silicio, introducir dopantes e introducir metales para las interconexiones. La tecnología CMOS cuenta con dos tipos de transistores, los *tipo n*(nMOS) y también los *tipo p*(pMOS) en la Figura 1 se puede observar ambos tipos de transistores. [5]

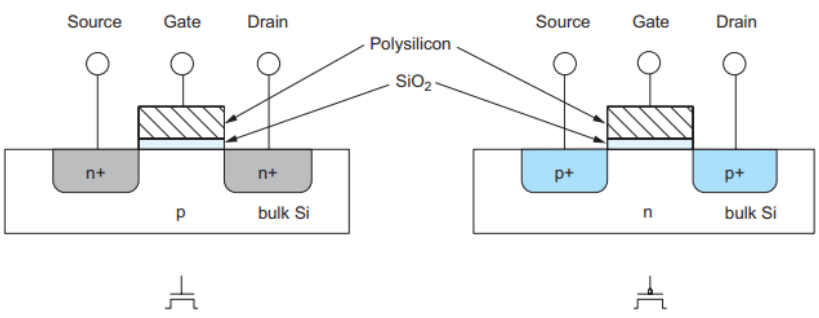


Figura 1: Transistores nMOS y pMOS.

6.2. VLSI

Cuando ya se tiene el diseño del circuito para una tarea en específico se debe tener en consideración el proceso de fabricación debido que tiene un gran impacto en el desempeño, potencia y costo de los circuitos integrados. El tamaño de los transistores e interconexiones está limitado a la resolución del proceso de manufactura, avances continuos en este campo han permitido que la resolución del proceso sea mayor y esta industria pueda tener un crecimiento exponencial. A pesar de la gran complejidad de los chips estos son relativamente económicos, debido que estos pueden ser impresos a gran escala de forma similar a los libros. El proceso de fabricación consiste en una serie pasos en los cuales se van formando las diferentes capas del chip por medio de un proceso llamado *Photolithoghaphy*, el cual consiste en mascaras que se sobreponen a una oblea circular fabricada de silicio denominada *wafer* antes de exponerlo a diferentes tipos de luz. El espacio del *wafer* se trata de optimizar lo más posible para lograr fabricar el mayor número de chips posibles en un único *wafer*, por lo tanto el costo individual de cada chip es proporcional al área que este ocupa y no por el número de transistores que este contiene.[5]

6.3. Flujo de diseño

El flujo de diseño es un conjunto de procedimientos que se deben seguir para poder pasar de una fase de especificaciones técnicas a una implementación final del chip sin errores. El flujo se divide en dos partes, *front-end* se encarga de la capa de comportamiento lógico y *back-end* se encarga de la capa física y de estructura. El proceso empieza en la etapa de *front-end*, donde se realiza un análisis de los requerimientos del diseño para poder realizar las especificaciones de comportamiento a las cuales se les realiza una síntesis lógica para obtener un diseño a nivel de *Register Transfer level* (RTL) en formato de HDL. Luego pasa a la etapa de *back-end* en la cual se establecen las especificaciones físicas del sistema, luego se realiza la síntesis física de lo obtenido a partir de la síntesis lógica de la etapa de *front-end*, obtenemos una descripción física del sistema(*layout*) la cual puede ser fabricada. En la Figura 2 se puede observar el flujo de diseño visualmente.

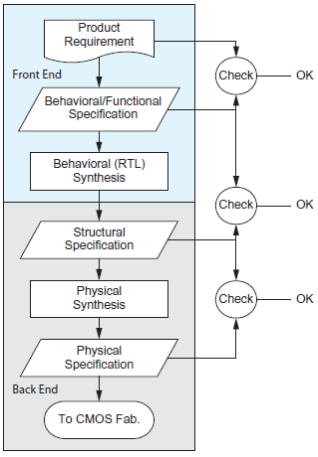


Figura 2: Design Flow

6.4. Front end

El diseño *Front-end* tiene más etapas que las vistas previamente en el diseño general, en esta etapa del diseño se plantea el diseño a través de un lenguaje descriptor de Hardware (HDL). Esta etapa de diseño se puede observar en la figura3.

El proceso consta de los siguientes pasos:

1. La definición de la tarea y especificaciones que se desea cumplir con la implementación del chip.

2. Diseño lógico del sistema, se puede realizar utilizando el lenguaje Verilog de forma *structural* o *behavioral*, a este diseño se le conoce como RTL.

3. Se realiza la síntesis lógica, la cual nos permite convertir el RTL en un *netlist* de compuertas lógicas.

4. El netlist se obtiene de manera *structural* y con componentes de las librerías de la tecnología que se trabajara.

5. La parte final antes de proceder con el *Back-end* consta de la verificación de funcionalidad de la síntesis lógica, esta verificación se realiza con las herramientas de Formality o VCS.

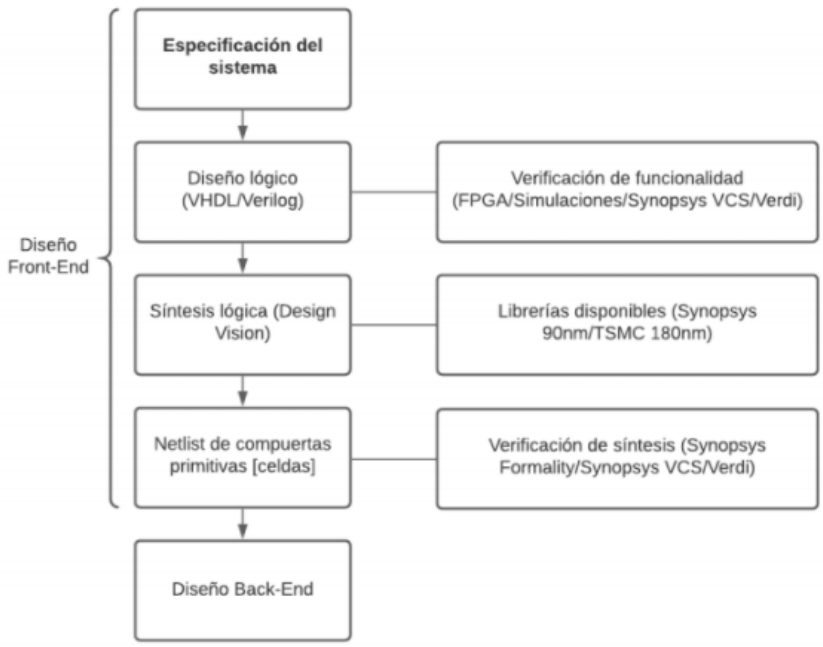


Figura 3: Diseño Front-end

6.5. Back end

La etapa de *Back-end* Consiste en la implementación física del diseño elaborado en la parte de *Front-end*, en esta etapa se toma el *netlist* que ya ha sido verificado y se convierte en un *layout* compuesto de *frames* o celdas por medio de la síntesis física. Esta etapa de diseño se puede observar en la Figura 4

El proceso consta de los siguientes pasos:

1. Se realiza la síntesis física, la cual toma el *netlist* verificado y lo convierte en una descripción física, conocida como *layout*. El *layout* obtenido contiene una topología del circuito que ya podría ser fabricable.
2. Se procede a realizar el *Placement*, el cual consiste en el diseño del *floorplan* que contiene las restricciones que se seguirán para organizar los componentes de manera estratégica en la planta.
3. Procedemos a realizar la parte de *Routing*, en la cual se realiza la interconexión de las celdas.
4. Se realiza el *Layout Versus Schematic*(LVS), el cual consiste en la verificación de la equivalencia entre el *layout* y el *netlist* del sistema.
5. Por último se realiza *Parasitic Extraction* y simulaciones, para obtener todas las propiedades físicas (resistividad, capacitancia, *delay*) de cada uno de los nodos que conforman nuestro chip.

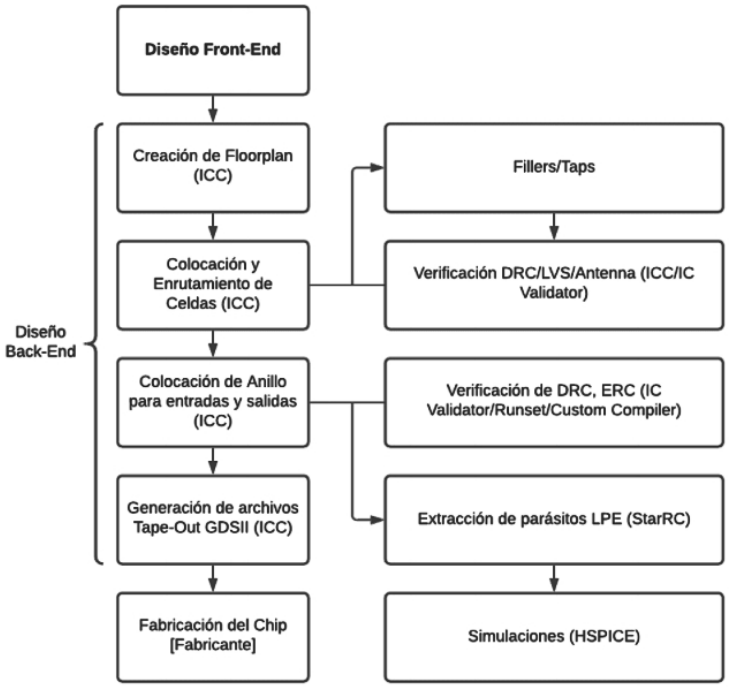


Figura 4: Diseño Back-end

6.6. HDL

Para la construcción y pruebas de circuitos que tengan un nivel de complejidad alto se utilizan los lenguajes descriptores de Hardware, denominados HDL (*Hardware description Languaje*), en los cuales es posible describir por texto conexiones, funciones y módulos electrónicos. Estos lenguajes permiten construir circuitos de manera rápida y sin la necesidad de invertir recursos físicos o tiempo en el ensamblaje de los circuitos, también nos brinda la capacidad de realizar cambios y simulaciones de una manera sencilla y ordenada. Algunos de los lenguajes descriptores de hardware más utilizados son Verilog y VHDL.[6]

6.7. Verilog

Desde sus inicios en 1984, Verilog ha sido de los lenguajes descriptores de Hardware preferidos debido a su amplio uso en el diseño de circuitos integrados y sistemas digitales. Una de las características que hacen destacar a verilog entre otros lenguajes, es su capacidad de poder especificar los sistemas digitales en una variedad de niveles de abstracción. Los sistemas digitales desarrollados utilizando verilog pueden ser simulados para verificar el funcionamiento del circuito descrito. Debido a que verilog fue diseñado con el objetivo de ser utilizado en todas las fases que conforman la creación de un sistema electrónico, es capaz de soportar el desarrollo, síntesis, verificación y simulaciones de diseños de hardware.[7]

6.8. Netlist

Un *netlist* está conformado por la representación real del diseño de un circuito electrónico. Este se conforma de instancias de los elementos básicos que conforman el circuito, sus interconexiones y sus atributos. Los elementos que contiene son referencias de las librerías que contienen los comportamiento y características reales de cada uno de los componentes. Las *netlist* están un paso más cerca de la implementación real de un circuito en Silicio.[8]

6.9. Synopsys

La compañía Synopsys se especializa en la creación de herramientas para el desarrollo y verificación de circuitos de nanométricos fabricados en silicio. Muchas empresas dedicadas al desarrollo de nuevas tecnologías como Intel, AMD, o Nvidia utilizan las herramientas que Synopsys para llevar a cabo sus proyectos. Synopsys cuenta con un amplio repertorio de herramientas especializadas para cada una de las diferentes etapas del desarrollo de un circuito a escala manométrica, con las cuales es posible optimizar el rendimiento, potencia, y área de un Chip. [9]

6.10. Design Vision

Dsesign Vision es una herramienta utilizada para realizar la síntesis lógica de circuitos. Toma archivos de diseño HDL y los sintetiza para obtener archivos *netlist* a nivel compuerta, soporta archivos Verilog y VHDL. Es capaz de realizar la síntesis utilizando librerías estándar y también librerías personalizadas como las de TSMC. La herramienta cuenta con gui(*Graphic User Interfaze*) y también se puede utilizar desde la línea de comandos.[10]

6.11. VCS y DVE

Synopsys cuenta con la herramienta de VCS, esta herramienta es la más empleada por las compañías de semiconductores alrededor del mundo para realizar las verificaciones de sus diseños, esta herramienta cuenta con un simulador de alto desempeño y sistema para resolver las limitaciones del sistema. En adicción contamos con DVE el cual es una complemento para la herramienta de VCS, este complemento nos permite observar los resultados de nuestra simulación de forma amigable y fácil de comprender.[11] [12]

6.12. Síntesis lógica empleando librerías de Synopsys

La etapa de síntesis lógica ya se ha desarrollado previamente por la UVG, esta se llevó a cabo empleando las librerías que Synopsys nos brinda para la utilización de su software, sin embargo, estas librerías no nos permiten realizar la fabricación del diseño y es necesario volver a realizar el proceso del flujo de diseño, pero empleando las librerías que nos brinda el fabricante TSMC.

Previamente se estableció como se debe realizar el diseño del circuito en lenguaje HDL, las configuraciones necesarias para utilizar la herramienta de Design Vision, los comandos que se deben realizar para llevar acabo la síntesis, las librerías y archivos que se le deben proporcionar a la herramienta, como establecer las restricciones de diseño, la revisión del diseño y generación de archivos de salida, la utilización de pines de entrada y salida, los procesos anteriores para llevar acabo la síntesis lógica fueron desarrollados previamente en [1].

CAPÍTULO 7

Síntesis lógica

7.1. Flujo de diseño

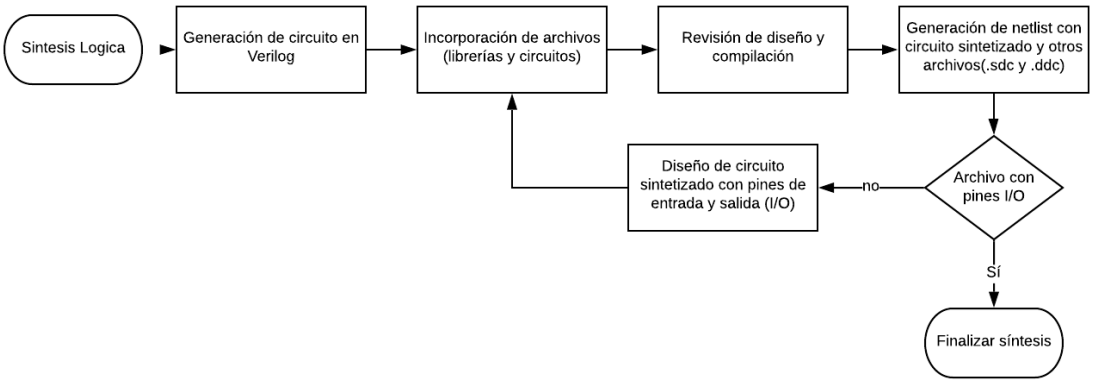


Figura 5: Diagrama del flujo para la síntesis lógica

Para poder realizar la etapa de síntesis lógica es necesario que se siga una serie de pasos, los cuales permiten llegar de un archivo en lenguaje HDL hasta un *netlist* final. El flujo que se seguirá para esta se muestra en la Figura 5, este flujo fue establecido por los estudiantes que trabajaron previamente en este proyecto [1]. Más adelante se encuentran detalladas cada una de las etapas con la descripción de lo que se realizó, los problemas que se encontraron, las herramientas que se emplearon y los resultados que se obtuvieron durante la síntesis de los diferentes circuitos empleando las librerías proveídas por TSMC.

Aparte de las etapas que se encuentran en la Figura 5, en necesario la implementación

de dos etapas extras que realizaran el mismo proceso pero en diferentes momentos del flujo, esta etapa es la simulación y verificación del funcionamiento de ambos archivos *verilog*, por lo que una etapa se agregara luego de haber generado el circuito en *verilog* y la otra etapa de simulación se realizara luego de haber verificado que el *netlist* sintetizado cuente con los pines I/O. La etapa de simulación se verá con mayor detalle en el siguiente capitulo.

7.2. Diseño del circuito en lenguaje HDL

La primera etapa para llevar a cabo la síntesis lógica es generar un circuito empleando un lenguaje descriptivo de *hardware*, para esto se empleó el lenguaje *verilog* que nos permite poder construir el circuito en distintos niveles de abstracción (nivel de compuerta, nivel RTL y a nivel de comportamiento lógico) en este caso se construyó el circuito a nivel de comportamiento y se deja que la herramienta *Design Vision* construya el circuito a nivel *RTL* como crea conveniente. Este primer diseño que se realice debe ser verificado por medio de un simulador, en el **capítulo 8** se describe este proceso, debido que para llevar a cabo el proceso de síntesis de forma exitosa es necesario tener la correcta descripción del circuito en lenguaje HDL.

El primer circuito que se desarrollo fue una simple *NOT*, esto con el objetivo de poder depurar los posibles errores que se obtuvieran e ir aumentando el nivel de complejidad conforme se lograra completar la síntesis.

```
1 module Not (A, Y);
2     input A;
3     output B;
4     assign Y = ~A;
5 endmodule
```

Con estas pruebas preliminares que se realizaron utilizando circuitos simples se pretendía lograr que los encargados de cada una de las etapas fueran capaces de completar su proceso y de esta manera poder establecer los requerimientos que tiene cada una de las etapas para poder llevarlas a cabo de forma exitosa.

7.3. Uso de Design Vision

La herramienta que se utilizó para realizar la síntesis de los circuitos es Design Vision, esta puede ser utilizada por medio de comandos y por medio de la interfaz gráfica. La empresa Synopsys nos provee una serie de manuales en donde se describe todas las funciones y comandos que se pueden implementar. Se sugiere la lectura de las guías para tener un mayor entendimiento de la herramienta, ya que en los manuales se explica el funcionamiento, los posibles errores y soluciones durante el proceso de diseño.

Para el desarrollo de este proyecto se empleó un uso híbrido de la herramienta, se emplearán comandos y también se utilizará la interfaz gráfica para ciertos procesos, esto con el fin de apoyar a la etapa de automatización estableciendo los comandos necesarios para llevar a cabo la síntesis. Para ejecutar la aplicación de Design Vision desde cualquier lugar y

lograr ser más eficiente, es necesario el agregar la variable de entorno de la carpeta donde se encuentra instalado nuestra aplicación (en nuestro caso $usr > synopsys > vcs$) la variable de entorno seria la siguiente.

```
PATH = /usr/synopsys/vcs/Q-2020.03-SP2-6/nin:$PATH
```

Luego de haber creado nuestra variable de entorno ya se podrá invocar a la aplicación desde la terminar estando en cualquier ubicación con este comando. es importante no correr la aplicación en segundo plano utilizando el símbolo & esto genera que la aplicación no funcione correctamente al seguir utilizando la terminal donde se ejecutó.

```
design_vision
```

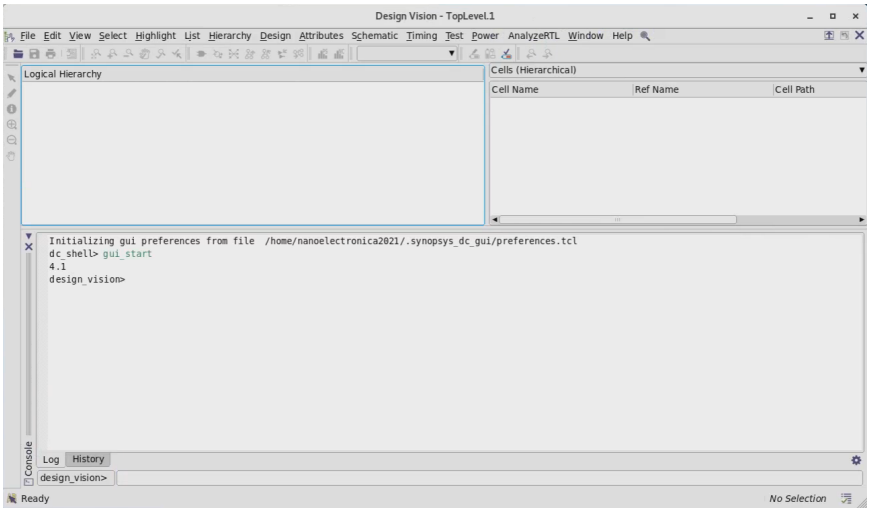


Figura 6: Interfaz gráfica de Design Vision

Luego de haber ejecutado nuestra herramienta de manera correcta se debe obtener una ventana como la que se observa en la Figura 6, los comandos para controlar el software pueden ser ingresados desde la terminal original del sistema operativo o en la parte inferior de la herramienta también se cuenta con una línea de comandos. Para concluir por completo la aplicación se debe ejecutar el comando **exit** y confirmar en la interfaz gráfica que se cerrara la aplicación.

Lo primero que se debe realizar para poder empezar a trabajar con la herramienta es el establecer las librerías necesarias, luego de esto se procede a cargar el circuito en lenguaje HDL y se tienen que definir las restricciones de entorno que se desea tener en el circuito ya sintetizado, este proceso puede llegar a complicarse dependiendo la precisión y complejidad que pueda con llevar cada diseño, sin embargo, para diseños simples no es necesario la definición te tantos parámetros.

La herramienta de Synopsys tiene la capacidad de poder configurar el rendimiento que se desea utilizar pare llevar acabo ciertas tareas, esto se debe a que para ciertos circuitos se puede llegar a volver complicado el procesamiento lógico de la computadora para obtener los comportamientos especificados, por lo tanto se puede configurar que le herramienta no esfuerce demasiado al CPU y aun así cumpla con los parámetros especificados, a cambio de no obtener un comportamiento completamente preciso pero aceptable.

7.3.1. Análisis del Script mejorado

La exalumna de Ingeniería electrónica Sophia Cardona que trabajó en este proyecto en años anteriores nos proveyó un script que contenía todos los comandos utilizados por ellos para llevar a cabo la síntesis lógica[1]. Este archivo contiene los pasos necesarios para ejecutar la importación de librerías, la lectura de archivos en verilog, definición de algunos parámetros de diseño y la generación de archivos de salida '.sdc', '.ddc'y '.v'

A continuación, está el *script* que se nos proveyó junto con la descripción de los que realiza cada uno de los comandos. En secciones más adelante del documento se mostrará a profundidad los cambios que se realizaron para que el proceso de síntesis se cumpliera sin errores.

```
1
2  ##########################################################################
3  ####                                                                  ####
4  ####  Copyright (C) 2019 UVG.                                         ####
5  ####                                                                  ####
6  ##########################################################################
7
8  #     SINTETIZACION DE CIRCUITO     #
9
10 ./design_vision
11
12 # Inclusion de librerias
13 lappend search_path /home/administrador/Escritorio/Proyecto/Libs/
       tcb018gbwp7t_290a_FE/tcb018gbwp7t/LM
14 set link_library " * tcb018gbwp7ttc.db tcb018gbwp7twc.db tcb018gbwp7tbc.db
       tpd018nvtc.db"
15 set target_library "tcb018gbwp7ttc.db"
16
17 # Inclusion de archivo descriptivo de circuito
18 read_file -format verilog {/home/administrador/Escritorio/Proyecto/FA.v}
19
20 # Verificacion de archivo
21 check_design
22
23 # Configuracion de restricciones
24 reset_design
25 set_max_area 0
26
27 # Compilacion de cicuito
28 compile -map_effort high -area_effort high
29
30 # Generacion de reportes
31 report_area
32 report_design
33 report_power
34
35 # Generacion de archivos
36 write -format ddc -h -o /home/administrador/Escritorio/Proyecto/Salida/FA_p.
       ddc
37 write -hierarchy -format verilog -output /home/administrador/Escritorio/
       Proyecto/Salida/FA_p.v
38 write_sdc /home/administrador/Escritorio/Proyecto/Salida/FA_sdc.sdc
39
40 # Salida de la herramienta
```

```
exit
```

El primer comando que se utilizó sirve para indicar en qué dirección se encuentran agrupadas las librerías que se emplearan durante el proceso de síntesis. en este caso la instrucción es utilizada para establecer las librerías: *link library* y también *target library*

```
lappend search_path
```

luego se procede a realizar la lectura del archivo que contiene la descripción lógica del circuito que emplearon para realizar la síntesis lógica. La opción -format se utiliza para especificar el tipo de formato del archivo que se introducirá, en este caso verilog, *Design Vision* tiene la capacidad de leer diferentes tipos de archivos. Por último es necesario establecer la ubicación del archivo en nuestra computadora, es recomendado usar la ubicación completa del archivo para evitar confusiones al momento de estar cargando los archivos.

```
read_file -format verilog **path**
```

El siguiente comando se utiliza para verificar si el diseño es correcto, concretamente verifica la estructura con la que se construyó el circuito. Esto para evitar que se sintetice algún circuito que no tenga sus pines bien conectados, referencias de celdas inexistentes entre las librerías y el diseño, etc.

```
chek_design
```

Luego se procede a realizar las configuraciones de restricciones, lo cual consiste en dos comandos. El primer comando se encarga de limpiar todas las configuraciones que el software tuviera automáticas y lo deja todo por defecto, con el segundo comando se define que el circuito ocupe la menor área posible al momento de sintetizarlo

```
reset_design
set_max_area 0
```

El siguiente comando procede a efectuar la síntesis y la optimización del circuito en base a las restricciones que se le hayan definido, a este comando se le definen dos configuraciones. La primera opción map_effort es para establecer el nivel de rendimiento que se desea utilizar al momento de realizar el mapeo de *netlist*, el segundo comando *area_ effort* es para establecer el rendimiento que utilizara para llevar a cabo la etapa de área del circuito sintetizado, un mayor rendimiento implica una mayor precisión por cumplir las restricciones.

```
compile -map_effort high -area_effort high
```

La generación de reportes es una parte vital para saber distintos factores del circuito sintetizado, el primer tipo de reporte *report_ area* nos muestra las características físicas del diseño, como el número de celdas y el área total que ocupa, el segundo reporte *report_ design* nos muestra todas las restricciones que se le definieron al diseño, y por el ultimo *report_ power* nos muestra los valores físicos de potencia que tendría el circuito ya sintetizado.

```
report_area
report_design
report_power
```

Por último se realiza la generación de archivos de salida, estos comandos realizan la función de almacenar el diseño sintetizado en distintos formatos para las diferentes etapas de diseño siguientes[13]. Los formatos que se generan en este caso son '.ddc', '.sdc' y '.v'

```
write -format ddc -h -o **PATH**
write -hierarchy -format verilog -output **PATH**
write_sdc **PATH**
```

7.3.2. Librerías TSMC

Esta es una de las etapas con mayor importancia para poder ejecutar la síntesis lógica sin ningún tipo de error. Al momento de seleccionar las librerías que se utilizaran se deben tener múltiples aspectos en cuenta, como la tecnología que se utilizara (en nuestro caso 180nm), el tipo de transistores que emplea, el nivel de voltaje correcto para alimentar el circuito (para la tecnología de 180nm se recomienda utilizar un voltaje de 1.8V).

Una diferencia principal con lo desarrollado previamente por los exalumnos que participaron en el proyecto en años anteriores es la utilización de las librerías de fabricación que TSMC nos proveyó para todo el flujo de diseño, debido que en años anteriores se utilizaron las librerías educaciones de Synopsys, por lo que era necesario volver a validar el flujo con los nuevos cambios.

Como se mencionó, se utilizan las librerías que nos suministró TSMC, a continuación, se encuentra las diferentes librerías empleadas y donde se pueden ubicar los diferentes archivos dentro de las carpetas comprimidas que nos brindó TSMC.

1. *Target Library*:
   a) TSMC > 180 > CMOS > G > stclib > 7-track > tcp018gbwp7t_290a_FE > tcb018gbwp7t_270a_nldm > TSMCHOME > digital > Front_End > timing_-power_noise > NLDM > tcb018gbwp7t_270a > tcb018gbwp7ttc.db
2. *Link Library*:
   a) TSMC > 180 > CMOS > G > stclib > 7-track > tcp018gbwp7t_290a_FE > tcb018gbwp7t_270a_nldm > TSMCHOME > digital > Front_End > timing_-power_noise > NLDM > tcb018gbwp7t_270a > tcb018gbwp7ttc.db
   b) TSMC > 180 > CMOS > G > stclib > 7-track > tcp018gbwp7t_290a_FE > tcb018gbwp7t_270a_nldm > TSMCHOME > digital > Front_End > timing_-power_noise > NLDM > tcb018gbwp7t_270a > tcb018gbwp7twc.db
   c) TSMC > 180 > CMOS > G > stclib > 7-track > tcp018gbwp7t_290a_FE > tcb018gbwp7t_270a_nldm > TSMCHOME > digital > Front_End > timing_-power_noise > NLDM > tcb018gbwp7t_270a > tcb018gbwp7tbc.db
   d) TSMC > 180 > CMOS > G > IO3.3V > iolib > LINEAR > tpd018nv_280a_FE > tpd018nv_280a_nldm > TSMCHOME > digital > Front_End > timing_-power_noise > NLDM > tpd018nv_280a_270a > tpd018nvtc.db

Luego de haber podido localizar todas las librerías que se emplearan, se recomienda copiar y ubicar todos los archivos dentro de un mismo directorio, ya que algunas librerías se encuentran separadas, esto con el fin de facilitar la incorporación de los archivos dentro de la herramienta de synopsys.

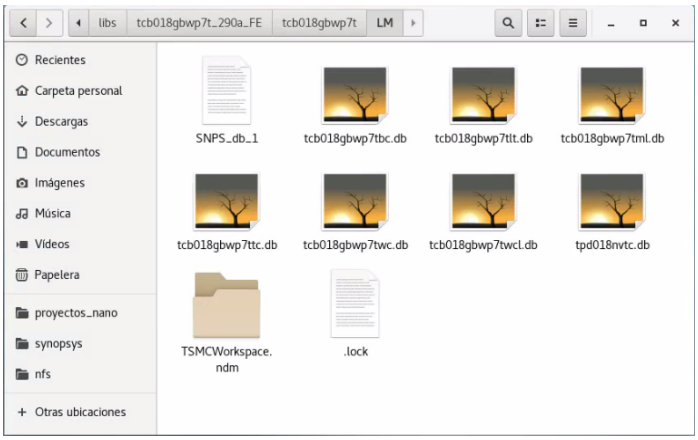


Figura 7: Librerías agrupadas en un mismo directorio

Los archivos anteriores son librerías estándar, estas contienen la descripción de las celdas que utilizara Design Vision para poder ensamblar el circuito que cumpla con el comportamiento que se desea obtener, entre ellas se encuentran compuertas como *not*, *and*, *mux*, *flip-flops* y muchas otras celdas. Aparte de las librerías estándar también se cuenta con una librería que contiene el comportamiento y descripción de los pines de I/O utilizados para hacer las conexiones al exterior.

Dentro de Design Vision se cuenta con dos formas para realizar la instancia de las librerías, en este proyecto se optó por realizarlo empleando la línea de comandos de la aplicación para poder automatizar este proceso en otra etapa del flujo de diseño. Sin embargo, es posible realizar la configuración por medio de la interfaz gráfica de la siguiente manera:

1. En la pestaña *file* se debe hacer clic en *Setup*:

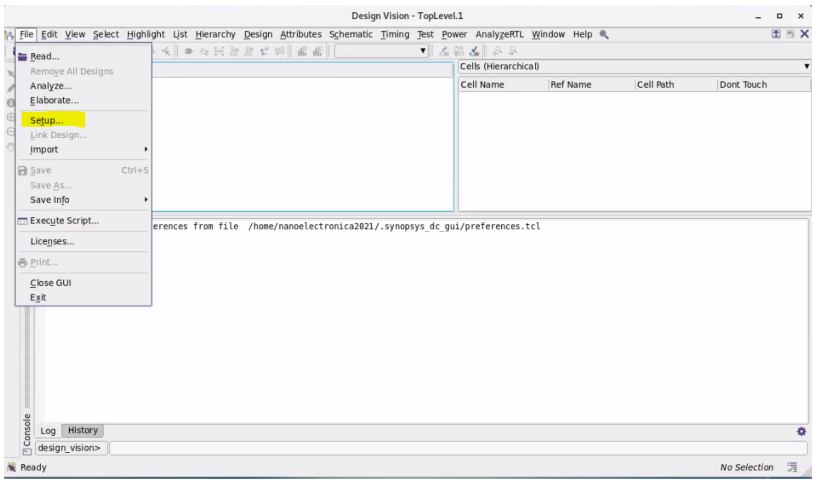


Figura 8: Configuración de Design Vision empleando GUI

2. Luego se debe abrir la ventana de variables de aplicación, en la cual se debe buscar las siguientes variables para modificarlas:

   a) search_path: en esta variable se debe especificar el directorio en el que se encuentran ubicadas nuestras librerías. todas las librerías a utilizar deben estar en el mismo directorio, de caso contrario la herramienta no sera capaz de ubicar los archivos.

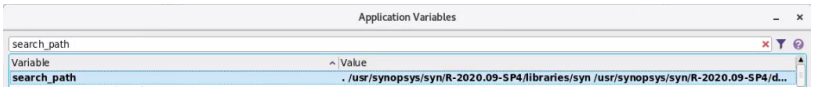


Figura 9: Variable search_path de Design Vision

   b) target_library: en esta variable se tiene que especificar nuestra librería objetivo, en nuestro caso es 'tcb018gbwp7ttc.db'.

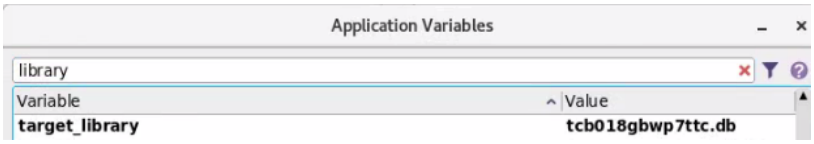


Figura 10: Variable target_library de Design Vision

   c) link_library: Se tiene que especificar las librerías que se utilizaran como enlace, en nuestro caso son cuatro librerías: 'tcb018gbwp7ttc.db', 'tcb018gbwp7twc.db', 'tcb018gbwp7tbc.db' y 'tpd018nvtc.db'.

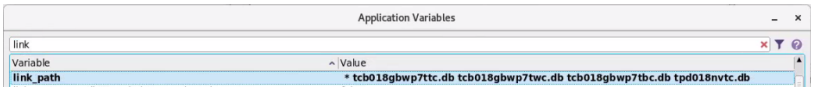


Figura 11: Variable link_library de Design Vision

Este proceso puede ser tedioso en comparación a hacerlo por medio de la línea de comandos, en lo cual se reduce a utilizar las siguientes tres líneas.

```
1 lappend search_path  *path directorio de librerias*
2 set link_library " * tcb018gbwp7ttc.db tcb018gbwp7twc.db tcb018gbwp7tbc.db
      tpd018nvtc.db"
3 set target_library "tcb018gbwp7ttc.db"
```

Luego de introducir estos comandos en Design Vision ya se tendrían las librerías instanciadas y estará todo listo para poder empezar a trabajar.

7.3.3. Importación de archivos

El siguiente paso luego de incorporar las librerías es la importación del archivo que contiene la descripción del circuito en lenguaje HDL. Los formatos de archivos aceptados por defecto en *Design Vision* son *Verilog*, *System Varilog* y VHDL. Esta herramienta también soporta los siguientes tipos de archivos:

1. ddc - *Synopsis internal database format*
2. db - formato para librerías
3. equation - *Synopsus equation format*
4. pla - *Berkeley PLA format*
5. st - *Synopsys state table format*

Para realizar la importación del archivo se utiliza el siguiente comando, para esta instrucción es necesario definir el formato del archivo que se está utilizando y también el *path* donde se encuentra ubicado.

```
read_file -format # *path*
```

Antes de realizar cualquier intento de síntesis es necesario verificar que el código *verilog* que se cargara a la aplicación compila de manera correcta (para esto se debe utilizar un simulador de *verilog* como VCS) y se tiene que confirmar que su *waveform* es de la forma que se espera. Únicamente luego de que el código haya pasado la parte de simulaciones se podrá realizar la síntesis del circuito.

Para realizar la importación del circuito por medio de la interfaz gráfica se deben seguir los siguientes pasos:

1. Se debe hacer clic en la pestaña de *File* y luego en la opción de *Analyze*.

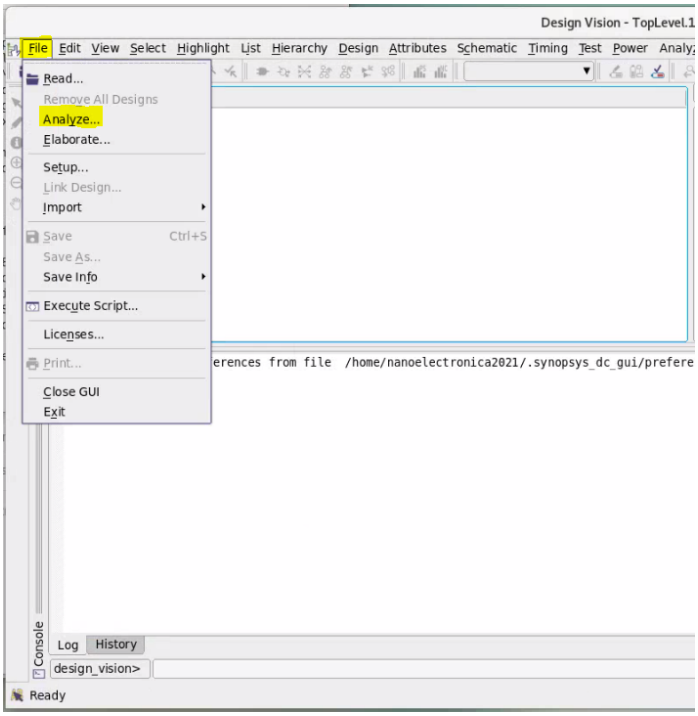


Figura 12: Inclusión de archivo a sintetizar

2. Luego se nos abrirá una venta como la de la Figura 13 en la cual se debe hacer clic al botón resaltado para acceder a una ventana como la de la Figura 14, en esta ventana se tendrá que presionar el botón de *add* para poder acceder al botón de búsqueda, este nos permitirá localizar el archivo que se desea cargar, por último se debe hacer clic en el botón *Ok*.

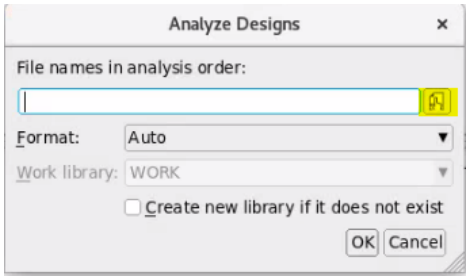


Figura 13: Preparación para cargar archivos

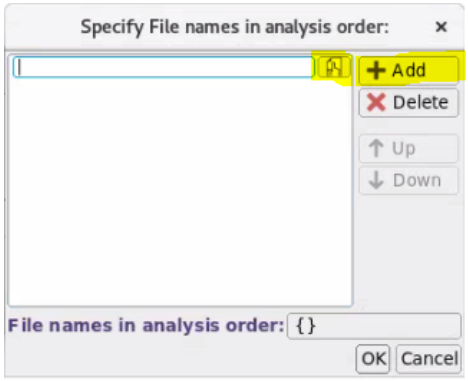


Figura 14: Selección de los archivos a cargar

3. Luego se tiene que volver a la pestaña de *file* para hacer clic en la opción de *Elaborate*.

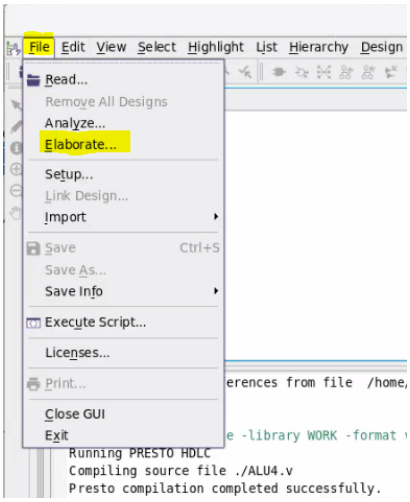


Figura 15: Elaboración del diseño

4. Se nos abrirá una ventana como la siguiente, en la cual se debe seleccionar la librería *work* y se tiene que seleccionar el diseño con el nombre que se le haya definido a el módulo previamente. Por último se tiene que hacer clic en *ok* y se tendrá el diseño cargado en la herramienta.

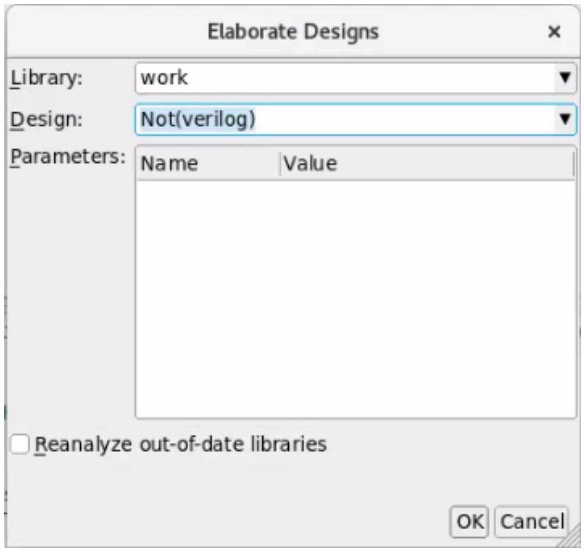


Figura 16: Elaboración del diseño

5. Para verificar que el diseño fue cargado de manera correcta, este nos debería aparecer de la siguiente manera.

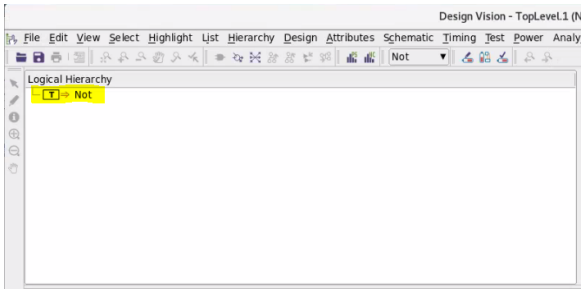


Figura 17: Archivo cargado correctamente

Una de las cualidades de la herramienta *Design Vision* es poder analizar de forma visual y amigable el diseño del circuito. Para acceder a esta vista es necesario hacer clic derecho sobre el diseño en la parte de *Logical Hierarchy* y luego seleccionar *Schematic View* como se muestra en la Figura 18. Esta opción nos permitirá acceder a una vista previa del diseño del circuito como la de la Figura 19.

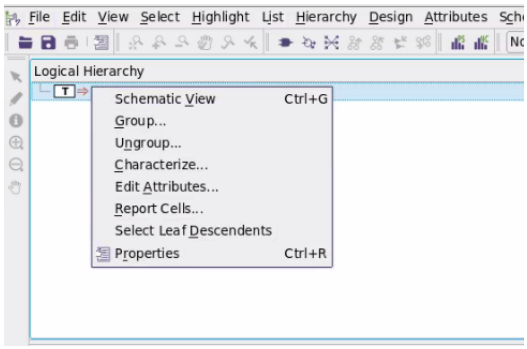


Figura 18: Acceder a la vista de esquematico

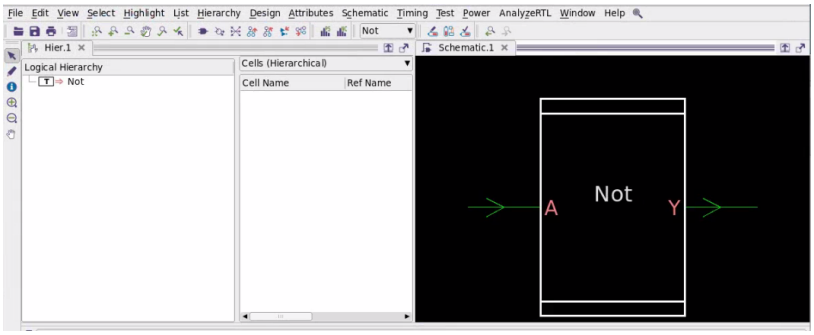


Figura 19: Esquemático de el circuito

Esta función es bastante útil debido que nos muestra el circuito como una *black box* para colaborar que el diseño cuente con las entradas y salidas correctas. Sin embargo esta función es bastante potente, con solo hacer doble clic a nuestra diseño en la pestaña de *schematic* nos permite observarlo en distintas jerarquías y poder corroborar que todo se encuentre bien, en la Figura 20 se puede observar la celda que conformaría el circuito. Esta opción puede ser de gran utilidad con circuitos más complejos, pero también aumenta la cantidad de interconexiones y celdas que se tendrían.

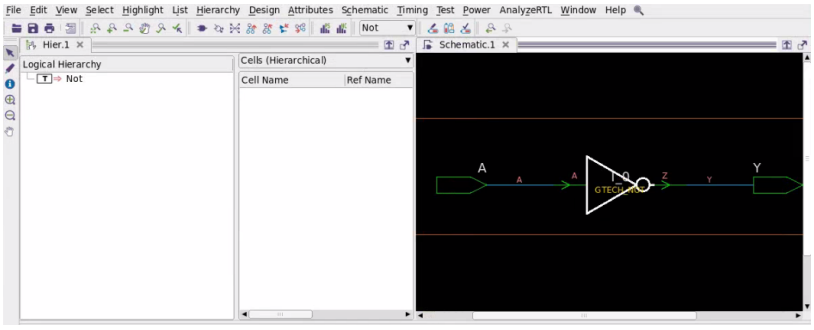


Figura 20: Distintas jerarquias del esquematico

7.3.4. Restricciones de diseño

Agregar las restricciones de diseño es un paso crucial para lograr que el diseño sea más realista que simplemente tener compuertas. Estas restricciones nos permitirán optimizar ciertos aspectos del circuito como los relojes que utiliza, los retrasos en las señales de entrada y salida, el área de diseño que utiliza, etc. Hay que tener en cuenta que estos parámetros deben ser configurados con valores realistas y no con valores aleatorios [10].

Antes de proceder a realizar cualquier cambio en las restricciones del diseño se deben realizar dos pasos de preparación. El primero consiste en realizar el "LINK"del diseño, esto se realiza en la pestaña de *File* y haciendo clic en la opción de *Link Design* como se muestra en la figura , también se puede hacer uso del comando 'link' en la línea de comandos, esta instrucción chequea que todos los elementos en el diseño actual están disponibles en las librerías que se le proporcionaron.

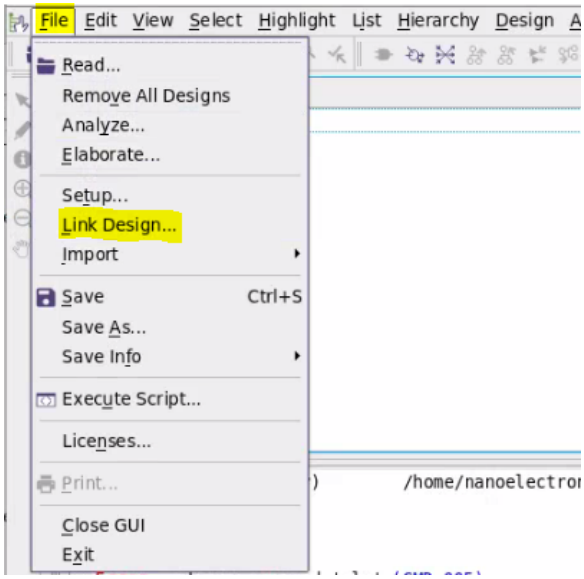


Figura 21: Realizar el enlace del diseño con las librerías

El segundo paso de preparación que se debe realizar es la unificación del diseño, este proceso es necesario para evitar que se tengan múltiples instancias de un mismo módulo en el diseño, esto se realiza en la pestaña *Hierarchy* y la opción de *Uniquify* > *Hierarchy* como se muestra en la Figura 22. También se puede realizar por de medio de la línea de comandos usando el comando 'uniquify'.

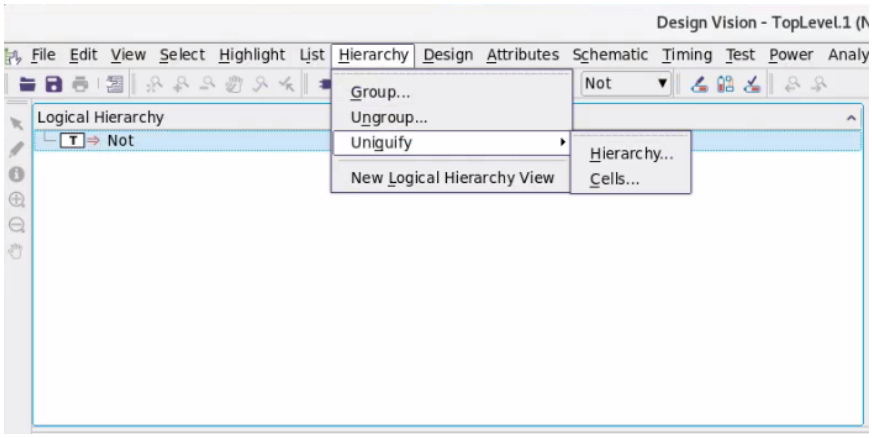


Figura 22: Unificación de el diseño

Restricciones de *Timing*

Para este tipo de restricciones se tendrán dos opciones, esto dependerá si el circuito es síncrono o puramente combinacional. Si el diseño no cuenta con un reloj se debe crear un *virtual clock*, este reloj será análogo a un reloj de sistema con el cual se compararan las señales del diseño. Para crear un *virtual clock* llamado clk se tiene que ejecutar el siguiente comando:

```
1 create_clock -period 40 -waveform {0 20} -name clk
```

Esto crea un reloj virtual con un periodo de 40ns y un ciclo de trabajo del 50 %, lo cual es igual a un reloj de 25MHz.

El caso que se tenga un circuito síncrono que tenga un pin de reloj se debe realizar el siguiente comando, el nombre del reloj debe ser el mismo que el nombre utilizado para nuestro pin de reloj en el código de *verilog*.

```
1 create_clock clk -period 40 -waveform {0 20}
```

Con este comando se crea un reloj de sistema con periodo de 40ns y ciclo de trabajo de 50 %, este será utilizado para crear las restricciones de las rutas de datos entre los *flip-flops* del diseño.

Estas configuraciones también se pueden realizar por medio de la interfaz gráfica, para esto se necesita ir a la pestaña de *Attributes* y hacer clic en la opción de *Specify Clock*, esto nos abrirá una ventana como la dela Figura 23, en la cual se podrá definir el nombre, periodo deseado y el ciclo de trabajo para el reloj que se desea implementar.

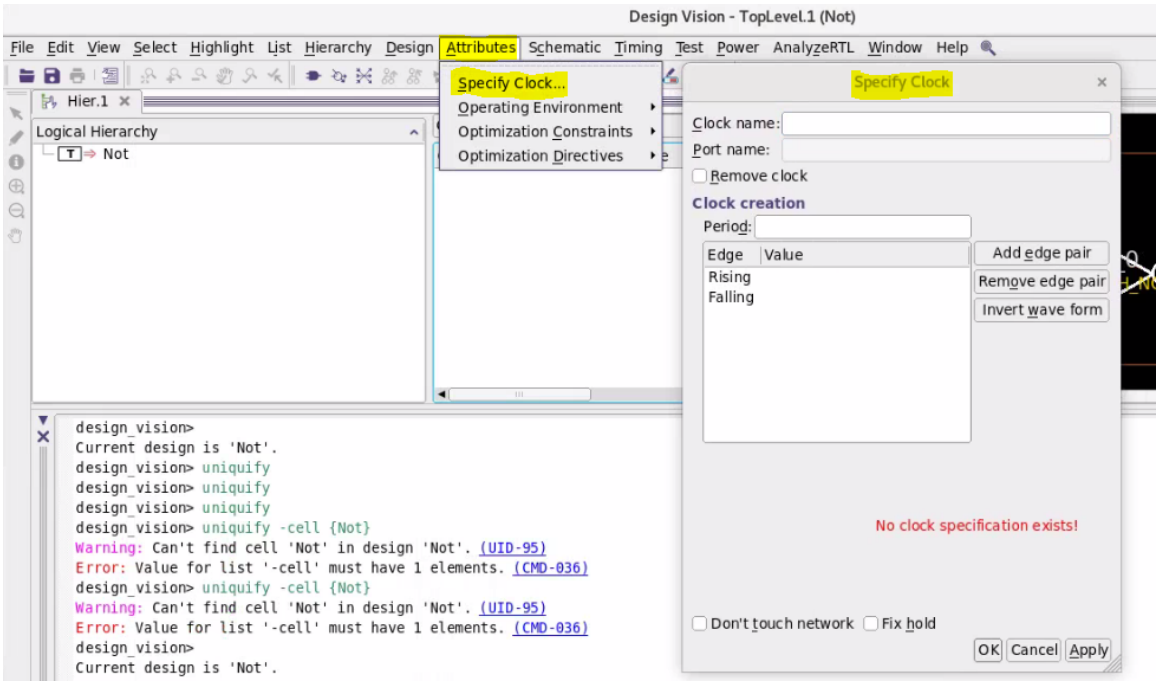


Figura 23: Creación de relojes utilizando la interfaz gráfica

Luego de haber creado los relojes dependiendo lo que el diseño necesite, se procede a crear la restricción de *delay* del reloj, esto es necesario debido que *Design Vision* asume por defecto que los relojes son ideales y en la realidad dependiendo del tamaño que tenga el diseño se tendrá que el reloj sufre de un sesgo para propagarse a lo largo del sistema. Para modelar este sesgo se tiene que ejecutar el siguiente comando (esta instrucción generara un sesgo de 3ns para el reloj en el diseño):

```
set_clock_latency 0.3 clk
```

Restricciones para pines de entrada y salida

Por defecto *Design Vision* modelara que las señales del sistema llegan hacia los puertos de entrada en el tiempo 0, de forma ideal, sin embargo en la mayoría de casos reales las señales toman tiempo para propagarse a lo largo del sistema, por lo que es necesario establecer un retardo para estas. Con el siguiente comando se podrá establecer un retardo, en este caso utilizamos un valor arbitrario de 2ns en todas las entradas para mostrar la forma correcta de realizar dicha configuración, relativo al reloj que se haya creado previamente.

```
set_input_delay 2.0 -clock clk [all_inputs]
```

De la misma forma que sucede en las entradas, las salidas también sufren de un retardo para propagarse por el sistema, por lo que es necesario establecer esta restricción en las salidas y lograr que el diseño sea más apegado a la realidad. Para esto se debe utilizar el siguiente comando:

```
set_output_delay 2.0 -clock clk [all_outputs]
```

Se deben establecer cargas reales para los pines de salida y entrada, esto con el fin de lograr que el modelo se vuelva más realista al simular que sus terminales no son ideales. Usando *set_load* es posible especificar el valor de carga capacitiva total que se desea en los puertos, este valor debe estar expresado en la misma unidad que la utilizada por la librería de tecnología. El comando *set_fanout_load* es utilizado para establecer el valor de *fanout_load* que se desea tener en los puertos de salida, este valor es necesario para realizar el cálculo de *fanout* en los puertos de entrada. La opción de *set_max_fanout* permite asegurarse que la suma del atributo *fanout_load* de todas las conexiones que reciben los pines de entradas no sea mayor al establecido. En este caso utilizamos valores arbitrarios para mostrar cómo se realiza la configuración, debido a que no se cuenta con un diseño final establecido al cual se le puedan hacer los cálculos de cargas reales necesarias.

```
set_load 0.1 [all_outputs]
set_fanout_load 8 [all_output]
set_max_fanout 1 [all_input]
```

Para verificar las restricciones que se han establecido se utiliza el comando *report_port*, el cual nos mostrara todas las entradas y salidas con sus respectivas configuraciones. Luego de ejecutar este comando se debe obtener lo siguiente:

```
design_vision> report_port

****************************************
Report : port
Design : Not
Version: R-2020.09-SP4
Date   : Mon Sep 20 13:06:22 2021
****************************************


                      Pin      Wire     Max     Max     Connection
Port          Dir     Load     Load     Trans   Cap     Class      Attrs
--------------------------------------------------------------------------------
A             in      0.0000   0.0000   --      --      --
Y             out     0.1000   0.0000   --      --      --

1
```

Figura 24: Restricciones configuradas para los pines de entrada y salida

Restricciones de área y velocidad

Al momento de utilizar *Design Vision* para realizar la síntesis del circuito, este puede tratar de minimizar el área que utiliza a cambio de sacrificar velocidad, o también puede minimizar la velocidad y sacrificar área[10]. Esto dependerá de lo que se desee obtener y los recursos que se puedan utilizar (el área suele ser limitada). *Design Vision* tratara de optimizar la síntesis con respecto al tiempo por defecto, si se desea que este optimice el área se debe ejecutar el siguiente comando:

```
set_max_area 0
```

Esto forzará que la herramienta optimice el área a lo mínimo posible, para optimizar la velocidad no es necesario realizar ningún cambio.

Si se desea realizar estas configuraciones por medio de la interfaz gráfica se debe dirigir hacia la pestaña de *Attributes* y en la sección de *Optimization Constraints* se podrán observar las siguientes opciones:

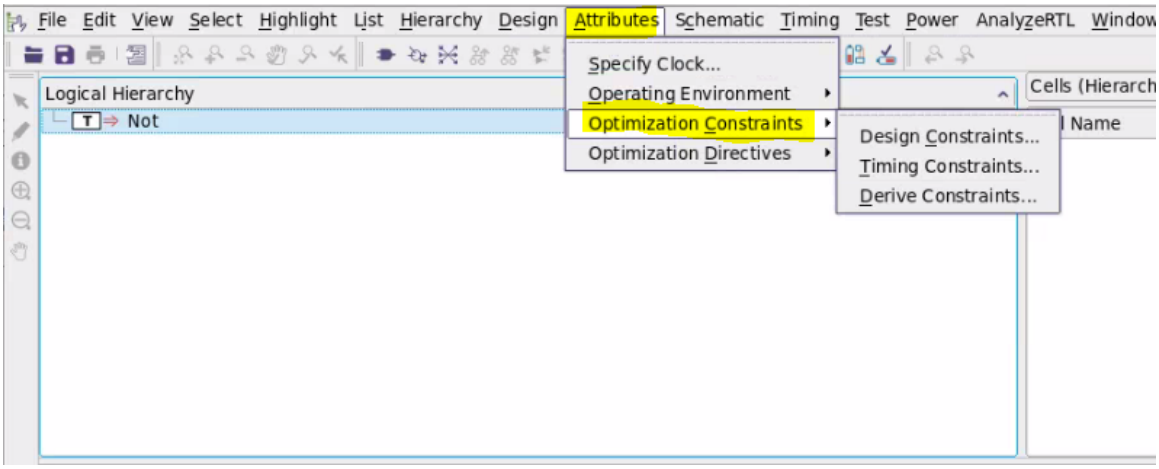


Figura 25: Restricciones en interfaz gráfica

De estas opciones se pueden seleccionar las que se deseen para agregar restricciones que vuelvan a el sistema más cercano a un comportamiento real.

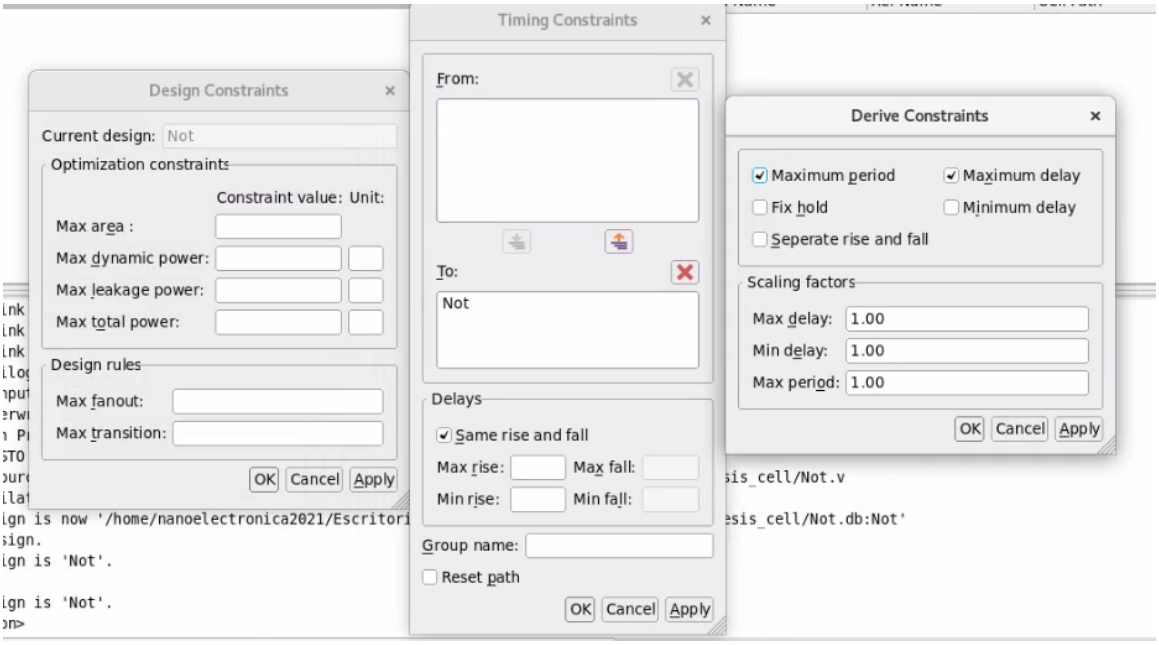


Figura 26: Configuración de restricciones en interfaz gráfica

7.3.5. Verificación de diseño y compilación

Luego de haber realizado todas las configuraciones que se deseaban en el diseño, se tiene que realizar un chequeo para corroborar que no existan problemas de conectividad, cortos,

múltiples instancias de un mismo modulo, etc. Esto se hace por medio del siguiente comando:

```
check_design
```

si todo está configurado correctamente *Design Vision* debería devolvernos un '1', esto significa que no encontró ningún problema al verificar el sistema. Luego de esto se puede proceder con la compilación del diseño.

Al ejecutar la compilación del diseño la herramienta se encarga de sintetizar el código *verilog* original en base a las celdas de las librerías de TSMC. En este proceso la herramienta realiza la combinación de múltiples celdas que logren el comportamiento que se le haya descrito por medio de HDL. Previo a realizar la síntesis se recomienda verificar lo siguiente:

- Verificar que el circuito descrito en *verilog* cuenta con la mejor estructura posible. Esto nos facilitara el solucionar posibles problemas.
- Corroborar que el diseño si es funcional por medio un simulador.
- Verificación visual del esquemático para corroborar que todas las entradas y salidas existen.
- Verificar que el diseño cuentes solo con las restricciones necesarias para evitar restringir el diseño.
- Verificar la creación de relojes dependiendo que tipo de circuito se tenga, síncrono o combinacional.
- Establecer los parámetros para la compilación, estos dependerán de la complejidad del diseño y la calidad de los resultados que se desea obtener.

Para realizar la síntesis únicamente se debe ejecutar el comando *'compile'*, sin embargo, esta instrucción cuenta con distintas opciones para configurar su comportamiento, en este caso son tres valores que se establecerán: *map_ effort* es utilizado para establecer el esfuerzo que realizara para la parte de mapeo(tiene dos estados posibles *medium* y *high*), el siguiente es *area_ effort* el cual define el esfuerzo que realizara para optimizar el área del diseño (cuenta con tres posibles estados *none, medium y high*), y por último se tiene *power_ effort* el cual controla el esfuerzo que realiza para la parte de optimización de energía(este parámetros cuenta con cuatro estados *none, low, medium y high*). El comando para ejecutar la síntesis tendría la siguiente forma:

```
compile -map_effort high -area_effort high -power_effort high
```

La compilación también se puede realizar por medio de interfaz gráfica, para ello se debe ir a la pestaña de *Design* y luego hacer clic en la opción de *Compile Design* esto abrirá una ventana como la de la Figura 27. En esta venta se podrá establecer el nivel de esfuerzo que se utilizará, al igual que con el comando anterior.

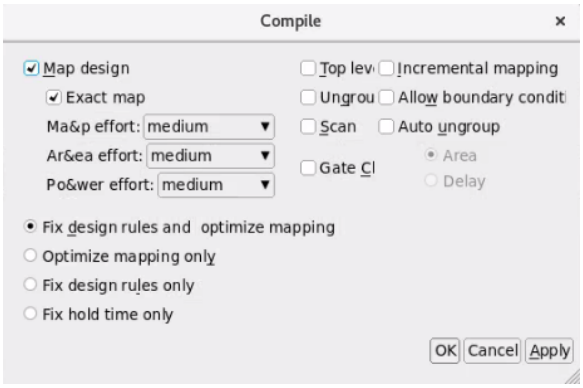


Figura 27: Síntesis por medio de interfaz gráfica

Luego de haber realiza la síntesis se puede hacer una segunda inspección del esquemático, para poder verificar si todos los módulos fueron compilados de manera correcta. En el caso de la compuerta *NOT* se tiene el siguiente esquemático:

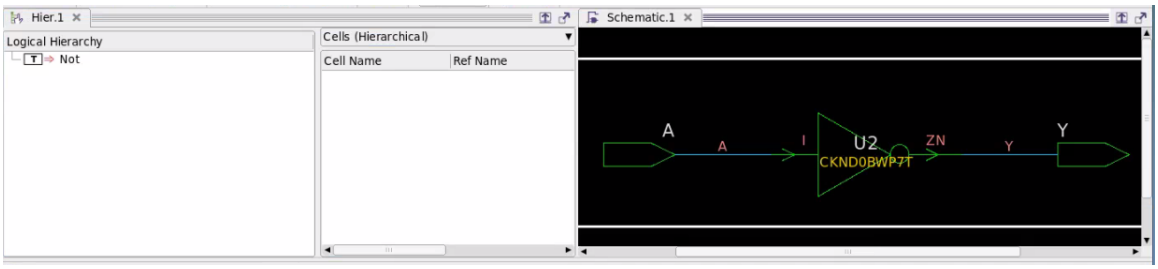


Figura 28: Esquemático sintetizado

Al comparar el esquemático de la Figura 28 y el previamente obtenido de la Figura 18, se puede observar que la celda utilizada como inversor no es la misma, esto es porque al sintetizarlo se reemplazan las celdas genéricas por celdas de las librerías. En este caso la celda CKND0BWP7T ya es una celda fabricable de TSMC.

7.3.6. Reportes y archivos de salida

Luego de realizar todos los pasos anteriores ya se podrá proceder con la última parte de la síntesis, en la cual se deben generar los reportes de las características del diseño y también los archivos de salida que serán utilizados en las siguientes etapas. Los archivos que se deben generar incluyen un archivo *verilog* que contiene el circuito sintetizado a nivel compuerta en formato *netlist*, otro archivo en formado '.sdc' que incluye las restricciones del diseño y por ultimo un archivo '.ddc' el cual contiene la información general del circuito. Para realizar la escritura de estos archivo se utilizan los comandos *write_file y write_sdc*.

Para realizar la escritura del archivo *verilog* de salida, se parte del comando *write_file* el cual almacena el diseño en un archivo físico. A este comando es necesario especificarle el formato que se desea generar por medio de la opción '-format', también se tiene que utilizar la opción '-hierarchy' debido que el sistema es jerárquico y por último se utiliza la opción

'-output' para establecer la ubicación y nombre con el cual se desea almacenar el resultado. El comando tendría la siguiente forma:

```
write_file -hierarchy -format verilog -output *path*/nombre.v
```

Para generar el archivo 'ddc', el cual contiene todas nuestras restricciones de diseño, se utiliza un formato parecido al anterior pero esta vez no se empleará la opción de '-hierarchy'. El nuevo comando tendría la siguiente estructura:

```
write_file -format ddc -h -o -output *path*/nombre.ddc
```

Por último se tiene que generar el archivo '.sdc' que contiene todas las restricciones que se definieron previamente. El comando para generar este archivo tendría esta estructura:

```
write_sdc *path*/nombre.sdc
```

Aparte de generar los archivos de salida que se les brindaran a las siguientes etapas de diseño, también se debe generar los distintos reportes que nos permitan confirmar que el diseño es correcto, los reportes más importantes que se deben generar son de área, diseño y potencia. Sin embargo existe una gran variedad de reportes dependiendo que se desee analizar, estos pueden encontrarse en la pestaña de *Design*:

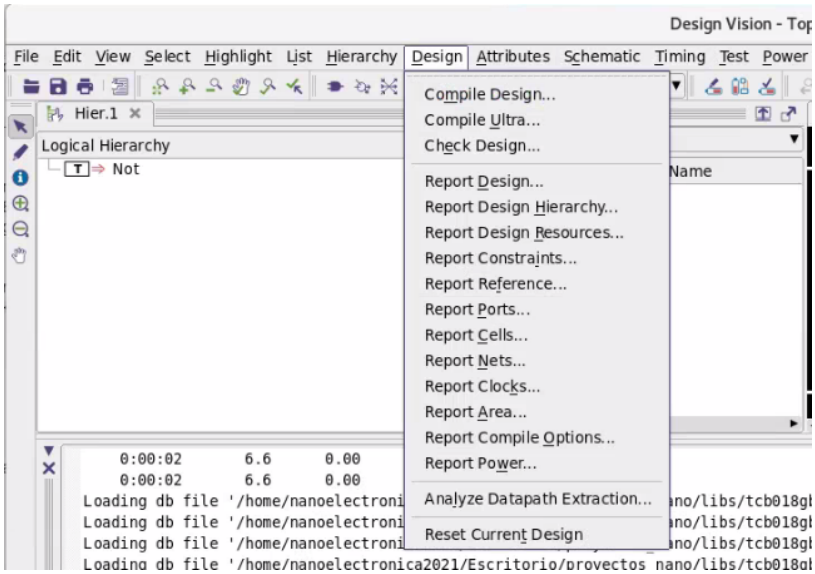


Figura 29: Reportes de Design Vision

Para generar los reportes de diseño, potencia y área se pueden utilizar los siguientes comandos desde la terminar:

```
report_design
report_power
report_area
```

En la Figura 30a se puede observar nuestra compuerta *NOT* sin sintetizar en lenguaje HDL, en la Figura 30b se tiene el código sintetizado que contiene el circuito ya implementado con celdas reales de TSMC.

```
module Not (A,Y);
        input A;
        output Y;

        assign Y=~A;
endmodule
```

(a) Compuerta *NOT* original

```
module Not ( A, Y );
  input A;
  output Y;


  CKND0BWP7T U2 ( .I(A), .ZN(Y) );
endmodule
```

(b) Compuerta *NOT* sintetizada

Figura 30: Archivos *verilog* de entrada y salida

Haciendo uso de los comandos mencionados anteriormente para la obtención de los reportes, se obtendrían los siguientes los resultados para la compuerta *NOT*:

- Reporte de área, este nos mostrará todos los aspectos físicos del diseño como el numero puertos, celdas utilizadas, área relativa, etc.

```
design_vision> report_area

****************************************
Report : area
Design : Not
Version: R-2020.09-SP4
Date   : Wed Sep 22 10:25:36 2021
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    tcb018gbwp7ttc (File: /home/nanoelectronica2021/Escritorio/proyectos_nano/libs/tcb018gbwp7t_290a_FE/tcb018gbwp7t/LM/tcb018gbwp7ttc.db)

Number of ports:                          2
Number of nets:                           2
Number of cells:                          1
Number of combinational cells:            1
Number of sequential cells:               0
Number of macros/black boxes:             0
Number of buf/inv:                        1
Number of references:                     1

Combinational area:                6.585600
Buf/Inv area:                      6.585600
Noncombinational area:             0.000000
Macro/Black Box area:              0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                   6.585600
Total area:                 undefined
1
```

Figura 31: Reporte de área

- Reporte de diseño, este tipo de reporte nos mostrara todas las restricciones y configuraciones con las que cuenta el diseño.

```
design_vision> report_design

****************************************
Report : design
Design : Not
Version: R-2020.09-SP4
Date   : Wed Sep 22 10:27:44 2021
****************************************

Design allows ideal nets on clock nets.

Library(s) Used:

    tcb018gbwp7ttc (File: /home/nanoelectronica2021/Escritorio/proyectos_nano/libs/tcb018gbwp7t_290a_FE/tcb018gbwp7t/LM/tcb018gbwp7ttc.db)

Local Link Library:

    {tcb018gbwp7ttc.db}

Flip-Flop Types:

    No flip-flop types specified.

Latch Types:

    No latch types specified.

Operating Conditions:


    Operating Condition Name : NCCOM
    Library : tcb018gbwp7ttc
    Process :   1.00
    Temperature :  25.00
    Voltage :   1.80
    Interconnect Model : balanced_tree
```

Figura 32: Reporte de diseño

- Reporte de potencia, como su nombre lo indica este tipo de reporte nos muestra todo lo relacionado al consumo eléctrico del diseño.

```
Operating Conditions: NCCOM   Library: tcb018gbwp7ttc
Wire Load Model Mode: segmented

Design        Wire Load Model            Library
------------------------------------------------
Not                   ZeroWireload      tcb018gbwp7ttc


Global Operating Voltage = 1.8
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  = 705.3370 nW  (100%)
  Net Switching Power  =   0.0000 mW    (0%)
                         ---------
Total Dynamic Power    = 705.3370 nW  (100%)

Cell Leakage Power     =  18.8771 pW

Information: report_power power group summary does not include estimated clock tree power. (PWR-789)

                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000             0.0000  (   0.00%)
register           0.0000           0.0000            0.0000             0.0000  (   0.00%)
sequential         0.0000           0.0000            0.0000             0.0000  (   0.00%)
combinational  7.0534e-04           0.0000        1.8877e-02         7.0536e-04  ( 100.00%)
--------------------------------------------------------------------------------------------------
Total          7.0534e-04 mW        0.0000 mW     1.8877e-02 nW      7.0536e-04 mW
1
```

Figura 33: Reporte de potencia

7.4. Implementación de pines de entrada y salida

En la sección anterior se llevó a cabo el proceso de síntesis para un circuito desarrollado en lenguaje HDL, sin embargo, todos los componentes de este diseño son nanométricos y no se puede interactuar con ellos de manera sencilla desde el exterior, debido a esto es necesario la implementación de pines de entradas y salidas que nos permitan interactuar con el circuito desde el exterior.

Para llevar a cabo la implementación de estos pines es necesario definir la librería de TSMC que se estará utilizando, esta librería ya fue definida previamente por los predecesores de este proyecto en [1] donde definen las configuraciones necesarias que se deben hacer para lograr una implementación exitosa en la tecnología de 180nm. Esta librería puede encontrarse en la siguiente ubicación:

- TSMC > 180 > CMOS > G > IO3.3V > iolib > tpd018nv_280a_FE > tpd018nv_-280a_nldm > TSMCHOME > digital > Front_End > timing_power_noise > NLDM > tpd018nv_280a_270a > tpd018nvtc.db

Para utilizar esta librería se debe incluir junto a nuestras librerías *link* en *Design Vision*. También es de vital importancia conocer la forma correcta de realizar la conexión de estos pines, de lo contrario no se podrán trasmitir las señales correctamente hacia dentro y fuera del circuito. Para esto es posible guiarnos del documento PDF que nos brinda el fabricante, el cual contiene la documentación de los pines.

El pin que se estará utilizando para las entradas y salidas es el PDDW0204SCDG, el comportamiento del pin está definido por tabla de verdad de la Figura 34, este módulo cuenta con la característica de poder ser configurado como entrada o salida dependiendo el estado de sus pines.

Truth Table

INPUT						OUTPUT	
DS	OEN	I	PAD	PE	IE	PAD	C
0/1	0	0	-	0/1	0	0	0
0/1	0	0	-	0/1	1	0	0
0/1	0	1	-	0/1	0	1	0
0/1	0	1	-	0/1	1	1	1
0/1	1	0/1	0	0/1	0	-	0
0/1	1	0/1	0	0/1	1	-	0
0/1	1	0/1	1	0/1	0	-	0
0/1	1	0/1	1	0/1	1	-	1
0/1	1	0/1	Z	0	0	-	0
0/1	1	0/1	Z	0	1	-	X
0/1	1	0/1	Z	1	0	L	0
0/1	1	0/1	Z	1	1	L	L

Figura 34: Pin de entrada y salida

Para realizar la inicialización del pin como entrada se debe realizar las configuraciones de la tabla de verdad de la Figura 35a, el realizar esta configuración hará que el pin tenga una forma como la de la Figura 35b donde *PAD* es la entra del mundo exterior y C es la conexión hacia el circuito nanométrico. Para incorporar este módulo en el diseño HDL se debe incluir la siguiente línea al código *verilog*:

```
PDDW0204SCDG *NAME* (.I(1'b0), .DS(1'b0), .OEN(1'b1), .PAD(*PIN_IN*), .C(*
    PIN_IN_WIRE*), .PE(1'b0), .IE(1'b1))
```

0/1	1	0/1	0	0/1	0	-	0
0/1	1	0/1	0	0/1	1	-	0
0/1	1	0/1	1	0/1	0	-	0
0/1	1	0/1	1	0/1	1	-	1

(a) Tabla de verdad para configuración como entrada

(b) Representación de PDDW0204SCDG como entrada

Figura 35: Configuración para PDDW0204SCDG como entrada

Para el caso en que se desee inicializar el pin como una salida se deben establecer las configuraciones acorde a la tabla de verdad de la Figura 36a, esta configuración hará que el pin de salida funcione como el de la 36b donde I es la conexión que recibe la salida del circuito nanométrico y *PAD* seria la conexión hacia el exterior. Para incorporar este módulo a el diseño se debe agregar la siguiente línea a el código:

```
PDDW0204SCDG *NAME* (.I(*PIN_OUT_WIRE*), .DS(1'b0), .OEN(1'b0), .PAD(*
    PIN_OUT*), .C(1'bx), .PE(1'b0), .IE(1'b1))
```

0/1	0	0	-	0/1	0	0	0
0/1	0	0	-	0/1	1	0	0
0/1	0	1	-	0/1	0	1	0
0/1	0	1	-	0/1	1	1	1

(a) Tabla de verdad para configuración como salida

(b) Representación de PDDW0204SCDG como salida

Figura 36: Configuración para PDDW0204SCDG como salida

Antes de realizar la implementación de estos pines se debe verificar que el archivo donde se hará ya se encuentre sintetizado, de no estar sintetizado *Design Vision* puede que falle debido a los distintos niveles de abstracción que contendría el archivo HDL. Los módulos para los pines de entrada y salida deben ser agregados en un nuevo nivel de jerarquía del sistema sintetizado, esto se debe a que se tiene que realizar la interconexión de todos los pines y nos ayudará a llevar un mejor orden del sistema. Lo primero que se debe realizar es la creación de un nuevo módulo de verilog al cual se le definen las mismas entradas y salidas que el circuito principal, también se debe crear un *wire* por cada entrada y salida del diseño, luego se tienen que agregar una instancia del módulo original utilizando los *wire* anteriores como en la siguiente figura:

```
module Not_IO (A, Y);
input A;
output Y;
wire A_w, Y_w;

Not compuerta (A_w, Y_w);

endmodule

module Not ( A, Y);
  input A;
  output Y;

  CKND0BWP7T U2 ( .I(A), .ZN(Y));
endmodule
```

Figura 37: Modulo *NOT* instanciado usando *wires*

Luego de haber instanciado el circuito empleando los *wire* como entradas y salidas, se puede pasar a realizar la conexión de estos con las entradas y salidas reales por medio del módulo PDDW0204SCDG, para realizarlo se deben utilizar las líneas de código que se establecieron previamente para entradas y salidas. Esto hará que el diseño sea como el siguiente:

```
module Not_IO (A, Y);
input A;
output Y;
wire A_w, Y_w;

Not compuerta (A_w, Y_w);


//Entradas:
PDDW0204SCDG U10(.I(1'b0), .DS(1'b0), .OEN(1'b1), .PAD(A), .C(A_w), .PE(1'b0), .IE(1'b1));

//Salidas:
PDDW0204SCDG U11(.I(Y_w), .DS(1'b0), .OEN(1'b0), .PAD(Y), .C(1'bx), .PE(1'b0), .IE(1'b1));

endmodule

module Not ( A, Y);
  input A;
  output Y;

  CKND0BWP7T U2 ( .I(A), .ZN(Y));
endmodule
```

Figura 38: Instancia de módulos para entradas y salidas

Luego de haber realizado todos los cambios a el código de *verilog* se debe replicar el procedimiento de la sección 7.3 con este archivo nuevo, al observar el esquemático del nuevo diseño en *Design Vision* Figura 39, se puede ver como existen más *black box* en comparación al diseño anterior de la Figura 28.

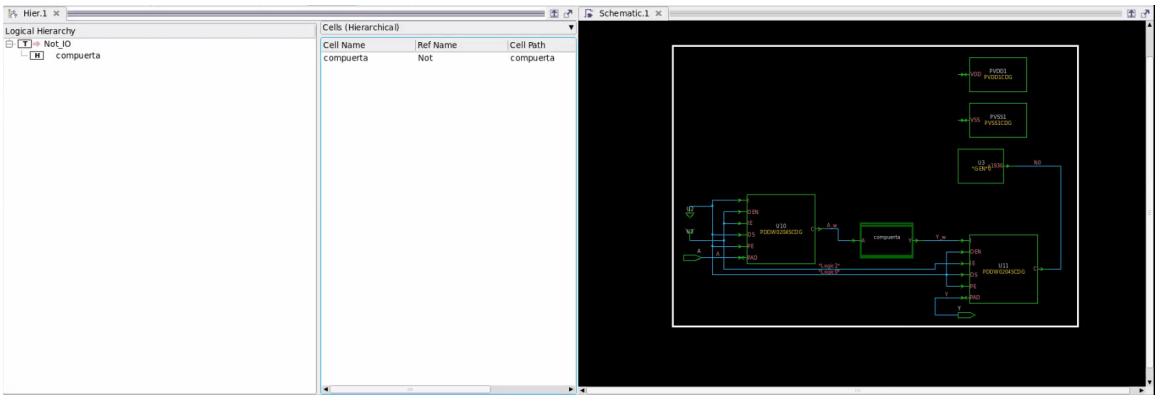


Figura 39: Esquematico circuito sintetizado con pines de entada y salida

Por último, se tiene que realizar la instancia de los pines de alimentación PVDD1CDG y PVSS1CDG, si no se agregan estos pines en esta etapa los encargados de la etapa de síntesis física necesitarán realizar la instancia de estos manualmente en su diseño, para realizar la instancia de estos solo se necesita agregar las siguientes líneas de código a el diseño:

```
PVDD1CDG PVDD1();
PVSS1CDG PVSS1();
```

7.5. *Script* y *Netlist* final

Para realizar la presente investigación se utilizó la información, instrucciones y resultados brindados por la ingeniera Karol Sophia Cardona Polanco, la cual era parte del equipo encargado del proyecto durante el 2020[1]. El archivo principal que se nos brindó consta de un *script* con el cual se había logrado ejecutar la síntesis lógica de manera exitosa, a este *script* se le incorporaron los comandos para la creación de los relojes, configuraciones de *delay* para el reloj, cargas para los pines y el enlace de las celdas con el diseño pare evitar módulos duplicados.

El *script* que se desarrolló, se divide en dos partes principales y que se desarrollan en distintos momentos del flujo de diseño. En la primera parte se realiza el proceso de síntesis lógica para el circuito que se encuentra en HDL, este nos brindara como salida el circuito como *netlist* implementado usando celdas fabricables de TSMC. En la segunda parte se ejecuta la síntesis de los pines de entrada y salida junto al *netlist* creado en la primera parte del *script*.

A continuación se pueden observar los dos *script* finales empleados para llevar a cabo la síntesis lógica y también se muestran los *netlist* sintetizados con pines de entrada y salida de las siguientes compuertas: *NOT*, *XOR*(utilizando una celda), *XOR*(usando múltiples celdas para ensamblarlo), *Full-Adder 4bits*, *Counter 4bits*, *ALU* y una memoria RAM. Empleando estos *netlist* y el archivo '.sdc' para cada uno se puede dar inicio a la etapa de síntesis física.

```
-----------------Sintesis Circuito HDL-----------------------
-------Creacion de directorios de trabajo--------------------
mkdir compuerta
```

```
cd compuerta

mkdir sintesis_cell
mkdir sintesis_cell_io

cd sintesis_cell
mkdir salidas
mkdir simulaciones

cd ..
cd sintesis_cell_io
mkdir salidas
mkdir simulaciones

------Librerias------
lappend search_path /home/nanoelectronica2021/Escritorio/proyectos_nano/libs
    /tcb018gbwp7t_290a_FE/tcb018gbwp7t/LM
set link_library " * tcb018gbwp7ttc.db tcb018gbwp7twc.db tcb018gbwp7tbc.db
    tpd018nvtc.db"
set target_library "tcb018gbwp7ttc.db"

----incluir verilog----
read_file -format verilog {*PATH*/compuerta/sintesis_cell/name.v}

----- Creacion de Relojes----------
create_clock - period 40 - waveform {0 20} - name clk //Caso cuando el
    circuito es puramente combinacional
create_clock clk - period 40 - waveform {0 20} // Caso cuando nuestro
    circuito es sincrono y cuenta con un pin de clock

---set de restricciones----
reset_design
set_max_area 0
set_clock_latency 0.3 clk
set_input_delay 2.0 - clock clk [ all_inputs ]
set_output_delay 2.0 - clock clk [ all_outputs ]
set_load 0.1 [ all_outputs ]
set_max_fanout 1 [ all_input ]
set_fanout_load 8 [ all_output ]

---Verificacion-----
check_design

---- compilacion -----
compile -map_effort high -area_effort high

------ Reportes ------
report_area
report_design
report_power

---- generar salidas -----
write -format ddc -h -o *PATH*/compuerta/sintesis_cell/salidas/name.ddc
write -hierarchy -format verilog -output *PATH*/compuerta/sintesis_cell/
    salidas/name.v
write_sdc *PATH*/compuerta/sintesis_cell/salidas/name.sdc
```

```
------------------Sintesis Circuito con IO-------------------------

```

```
------Librerias------
lappend search_path /home/nanoelectronica2021/Escritorio/proyectos_nano/libs
    /tcb018gbwp7t_290a_FE/tcb018gbwp7t/LM
set link_library " * tcb018gbwp7ttc.db tcb018gbwp7twc.db tcb018gbwp7tbc.db
    tpd018nvtc.db"
set target_library "tcb018gbwp7ttc.db"

----incluir verilog----
read_file -format verilog {*PATH*/compuerta/sintesis_cell_io/name.v}

---Verificacion-----
check_design

---- compilacion -----
compile -map_effort high -area_effort high

------ Reportes ------
report_area
report_design
report_power

---- generar salidas -----
write -format ddc -h -o *PATH*/compuerta/sintesis_cell_io/salidas/name.ddc
write -hierarchy -format verilog -output *PATH*/compuerta/sintesis_cell_io/
    salidas/name.v
write_sdc *PATH*/compuerta/sintesis_cell_io/salidas/name.sdc
```

```
module Not ( A, Y );
  input A;
  output Y;


  CKND0BWP7T U1 ( .I(A), .ZN(Y) );
endmodule


module Not_IO ( A, Y );
  input A;
  output Y;
  wire   A_w, Y_w, n1, n2;
  tri   A;
  tri   Y;

  Not compuerta ( .A(A_w), .Y(Y_w) );
  PDDW0204SCDG U10 ( .I(n1), .OEN(n2), .IE(n2), .PAD(A), .DS(n1), .PE(n1), .C(A_w) );
  PDDW0204SCDG U11 ( .I(Y_w), .OEN(n1), .IE(n2), .PAD(Y), .DS(n1), .PE(n1) );
  TIELBWP7T U6 ( .ZN(n1) );
  TIEHBWP7T U7 ( .Z(n2) );
endmodule
```

Figura 40: *Netlist* final de la síntesis lógica de la compuerta *NOT*

```
module Xor_2 ( A, B, Y );
  input A, B;
  output Y;


  CKXOR2D0BWP7T U1 ( .A1(B), .A2(A), .Z(Y) );
endmodule


module Xor_IO ( A, B, Y );
  input A, B;
  output Y;
  wire   A_w, B_w, Y_w, n1, n2;
  tri   A;
  tri   B;
  tri   Y;

  Xor_2 compuerta ( .A(A_w), .B(B_w), .Y(Y_w) );
  PDDW0204SCDG U10 ( .I(n1), .OEN(n2), .IE(n2), .PAD(A), .DS(n1), .PE(n1), .C(
        A_w) );
  PDDW0204SCDG U11 ( .I(n1), .OEN(n2), .IE(n2), .PAD(B), .DS(n1), .PE(n1), .C(
        B_w) );
  PDDW0204SCDG U12 ( .I(Y_w), .OEN(n1), .IE(n1), .PAD(Y), .DS(n1), .PE(n1) );
  TIELBWP7T U6 ( .ZN(n1) );
  TIEHBWP7T U7 ( .Z(n2) );
endmodule
```

Figura 41: *Netlist* final de la síntesis lógica de la compuerta *XOR* con una única celda

En el apartado de anexos se pueden encontrar los *netlist* finales para las siguientes compuertas: *XOR*(con múltiples celdas), *Full-Adder* de 4 bits y memoria RAM de 4 bits.

CAPÍTULO 8

Simulación y pruebas

El proceso de simulaciones es de suma importancia para confirmar que el *netlist* final que se obtuvo al realizar la síntesis lógica concuerda con el funcionamiento del circuito original en HDL, si no se realiza la verificación se corre el riesgo que las siguientes etapas del flujo de diseño no puedan completarse exitosamente. Este proceso se debe llevar a acabo dos veces, la primera comprobación se efectúa previo a realizar la síntesis lógica para confirmar que el código funciona correctamente y la segunda comprobación se hace luego de haber sintetizado el circuito junto a los pines de entrada y salida, ambas comprobaciones deberían tener el mismo comportamiento para saber que la síntesis se completó sin ningún error.

8.1. Software y librerías

Para poder verificar el funcionamiento del diseño se debe utilizar un programa capaz de efectuar simulaciones de archivos *verilog*, Synopsys cuenta con una herramienta llamada VCS la cual es una de las soluciones para verificación más utilizadas por empresas dedicadas a semiconductores, esta herramienta nos provee con una alta eficiencia en simulaciones y también un motor para resolución de restricciones. [11]

A parte de utilizar VCS para llevar a cabo la simulación también se necesita otra herramienta que nos permita analizar el waveform del circuito, para esto se empleó el complemento de VCS llamado DVE el cual es una interfaz gráfica que se puede utilizar para verificar el comportamiento del diseño de forma amigable. Algo importante que resaltar, es que ambos programas deben ser instalados en conjunto, de lo contrario se darán múltiples errores al instalar DVE a un VCS ya existente, también cabe mencionar que ambas herramientas deben tener la misma versión de lo contrario puede generarnos problemas de compatibilidad.[12]

Otro elemento clave necesario para poder realizar las simulaciones son los archivos *verilog*

que contienen el comportamiento de las celdas fabricables de TSMC, si no se cuenta con estos archivos no será posible realizar la simulación del *netlist* sintetizado debido que al introducir el código al simulador este no será capaz de saber cómo se comportan las distintas celdas de TSMC. Son dos librerías que se deben localizar tcb018gbwp7t.v y tpd018nv.v, una contiene el comportamiento de las celdas y la otra cuenta con el comportamiento de los pines de entrada y salida, estas pueden ser ubicadas en las siguientes ubicaciones:

- TSMC > 180 > CMOS > G > stclib > 7-track > tcp018gbwp7t_290a_FE > tcb018gbwp7t_270a_nldm > TSMCHOME > digital > Front_End > verilog > tcb018gbwp7t_270a > tcb018gbwp7t.v
- TSMC > 180 > CMOS > G > IO3.3V > iolib > LINEAR >tpd018nv_280a_FE > tpd018nv_260a_vlg > TSMCHOME > digital > Front_End > verilog > tpd018nv_-260a > tpd018nv.v

8.2. Proceso de simulación

Lo primero que se debe realizar antes de empezar a realizar la simulación es un archivo de *test bench* en formato *verilog*, en el cual se definen los estados de las entradas en distintos tiempos, en la Figura 42 se puede observar como el valor de la entrada alterna para la compuerta not. Cabe remarcar que el archivo de *test bench* debe incluir las siguientes líneas para poder generar el archivo '.vcd' que contiene el *waveform* de la simulación y también se tiene generar el archivo '.fsdb' el cual es necesario para una de las siguientes etapas del flujo de diseño.

```
$fsdbDumpfile("not.fsdb");
$fsdbDumpvars(0, Not_tb);
$dumpfile("not.vcd");
$dumpvars(0, Not_tb);
```

```
module Not_tb;
wire Y;
reg A;

Not compuerta(.A(A), .Y(Y));

initial
begin
        $fsdbDumpfile("not.fsdb");
        $fsdbDumpvars(0, Not_tb);

        $dumpfile("not.vcd");
        $dumpvars(0, Not_tb);

        A = 1'b0;
        #1;
        A = 1'b1;
        #1;
        A = 1'b0;
        #1;
        A = 1'b1;
end
endmodule
```

Figura 42: *Test Bench* para la compuerta *NOT*

El proceso para realizar la simulación se compone de dos pasos, primero utilizar VCS para crear un ejecutable de nuestra simulación, la segunda parte es ejecutar la simulación para obtener el *waveform*. Por lo tanto, se empieza del primer comando para poder realizar la compilación del diseño:

```
vcs -v tpd018nv.v tcb018gbwp7t.v not.v not_tb.v -o simulacion -full64 -
    debug_access+all
```

A este comando se le deben especificar las librerías que se localizaron previamente en este capítulo y también se tiene que agregar el archivo *verilog* del diseño y su respectivo *test bench*, aparte de introducirle estos archivos también se le deben establecer las siguientes configuraciones:

- -v: Establece el formato del archivo que se cargará a la aplicación.
- -o: Sirve para especificar el nombre del ejecutable producto de la compilación.
- -full64: compila el diseño en modo de 64 bits y crea un ejecutable de 64 bits.
- -debug_access+all: Activa el *dumping* de los archivos FSDB/VPD, también habilita todas las capacidades de depuración.

Luego de ejecutar VCS para realizar la compilación se obtendrán múltiples archivos, de estos nos interesa el ejecutable llamado 'simulación' este archivo es nuestra simulación prácticamente, al ejecutarlo este procederá a realizar la simulación de el diseño y creará los archivos '.vcd' y '.fsdb'. Para ejecutar nuestra simulación se utiliza el siguiente comando en la terminar de la carpeta que lo contiene:

```
./simulacion
```

8.3. Análisis de resultados

Posteriormente a haber realizado la ejecución nuestra simulación se puede proceder a observar los resultados empleando DVE, para este paso es necesario verificar que la simulación haya creado del archivo '.vcd' de lo contrario se debe verificar que se haya realizado el *dumpfile* dentro del *test bench*, para ejecutar DVE se tiene que utilizar el comando 'dve' dentro de una terminal y se mostrara una ventana como la de la Figura 43

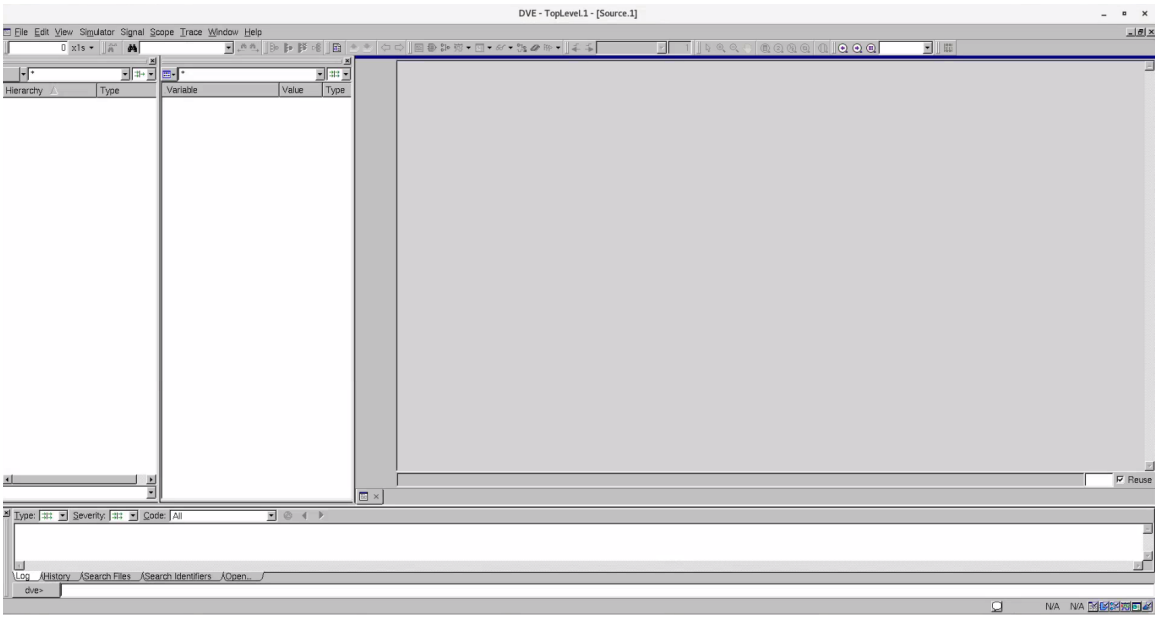
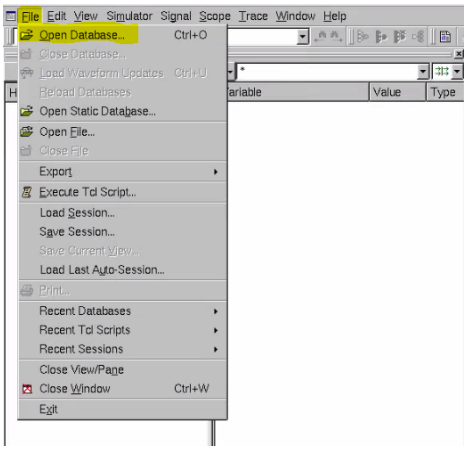
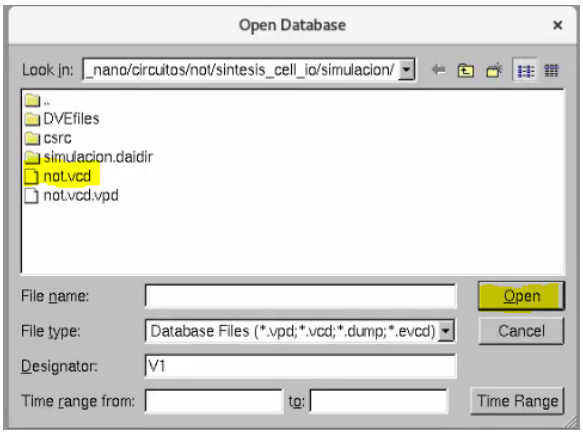


Figura 43: Interfaz grafica de DVE

Luego de haber ejecutado la herramienta de DVE se debe ir a la pestaña de file y hacer clic en la opción *Open Database* como se muestra en la Figura 44a, esto nos abrirá una ventana como la de la Figura 44b en la cual se debe ubicar el archivo '.vcd' y hacer clic en *Open*.



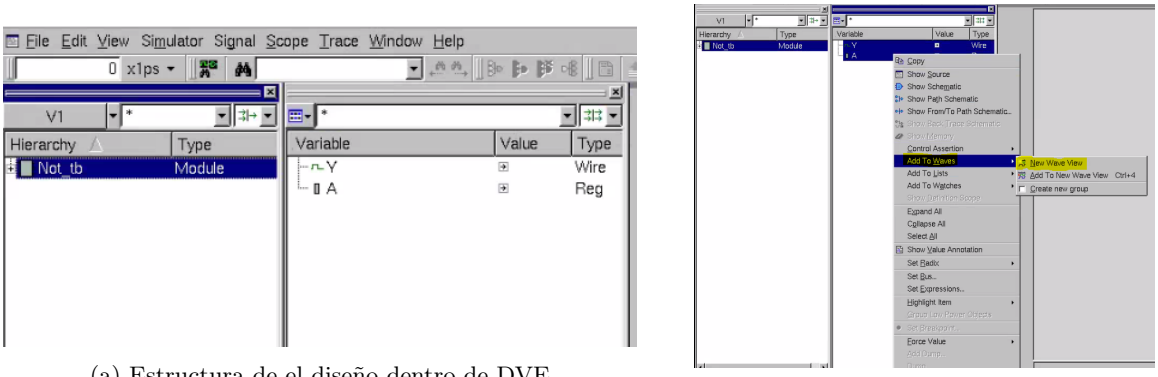
(a) Pestaña *File* DVE



(b) Ventana de *Open Database*

Figura 44: Apertura del archivo '.vcd' en DVE

Al haber realizado los pasos anteriores, el diseño debería aparecernos en el apartado de *Hierarchy* como en la Figura 45a, se debe hacer clic en el diseño y esto nos mostrará las entradas y salidas que lo conforman, se tienen que seleccionar los puertos de interés para analizar y hacer clic derecho sobre ellos para que se nos abran las opciones de la Figura 45b, en donde se debe ir al apartado de *Add To Waves* y hacer clic en *New Wave View*.



(a) Estructura de el diseño dentro de DVE

(b) Selección de las entradas de el diseño

Figura 45: Selección de el modelo y salidas a analizar

8.3.1. Not

Si todos los pasos fueron realizados correctamente se nos abrirá una venta como la de la Figura 46, en la cual se puede observar si los resultados tienen el comportamiento que se desea. Al abrir esta ventana probablemente no se observará el comportamiento del circuito, esto se debe a la escala de la gráfica, para corregirlo se debe ir a la barra de herramienta superior y localizar el icono de lupa (este está subrayado en la figura) para poder ajustar la escala automáticamente.

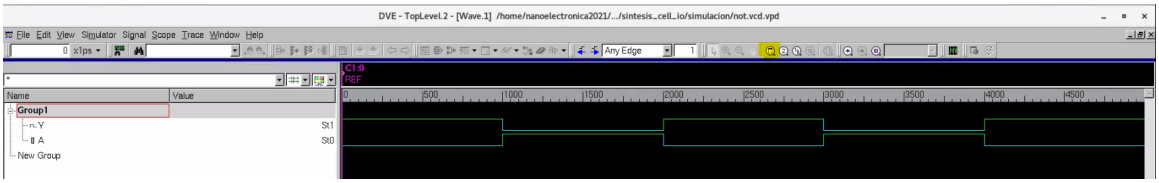


Figura 46: Resultados de simulación para la compuerta *NOT*

8.3.2. Xor

Para el caso de la *XOR* se cuenta con dos simulaciones finales, debido que se construyeron dos circuitos distintos, pero con el mismo propósito.

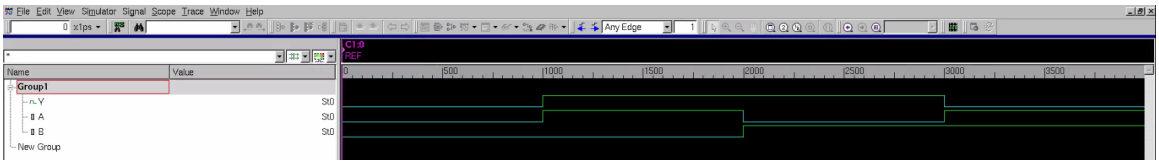


Figura 47: Resultados de simulación para la compuerta *Xor* con una única celda

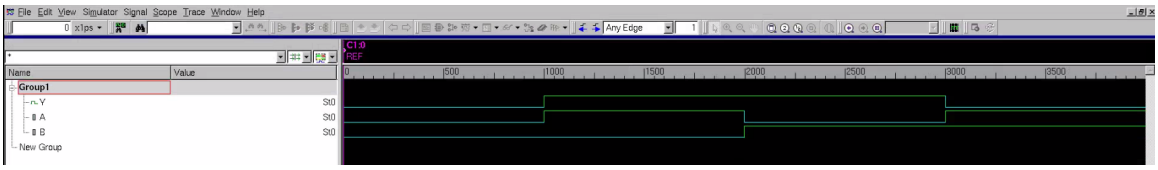


Figura 48: Resultados de simulación para la compuerta *Xor* con múltiples celdas

8.3.3. Counter 4bits

Se creó un contador síncrono para corroborar que la síntesis lógica funcionara con circuitos que no son puramente combinacionales.

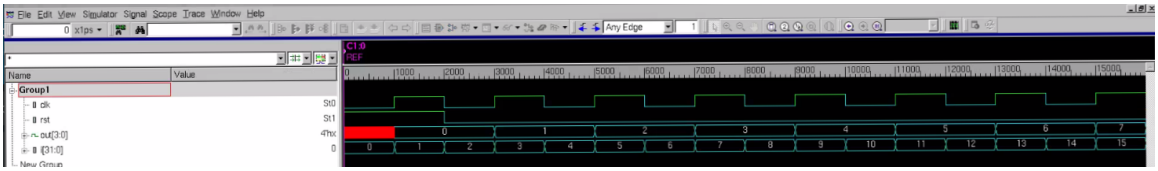


Figura 49: Resultados de simulación para un contador síncrono de 4 bits

8.3.4. FullAdder, ALU y RAM de 4bits

Luego de haber terminado de realizar diferentes pruebas con circuitos relativamente simples, se procedió a sintetizar circuitos que tuvieran una mayor complejidad como una ALU o un *FullAdder*, en las siguientes imágenes se muestra como las compuertas cuentan con un comportamiento deseado, por lo tanto, la síntesis se llevó a cabo de forma exitosa.

Para el caso del *fulladder* se realizaron múltiples sumas para comprobar su funcionamiento, en la siguiente tabla se puede observar cómo en T1 realiza la suma de A, B y C_in dando como resultado 6, para el caso de T2 se realiza la misma suma con los nuevos valores, pero el total de esta suma es mayor a 4 bits por lo que el C_out que es el *carry out* se activa, así sucesivamente para los tiempos T3, T4 y T5. (Los valores se encuentran en Hexadecimal)

Pin	T0	T1	T2	T3	T4
A (dato 1)	0	4	3	5	D
B (dato 2)	0	1	D	2	6
C_in	0	1	1	1	1
sum	0	6	1	8	4
C_out	0	0	1	0	1

Cuadro 1: Tabla de comportamiento para Full-Adder de 4 bits

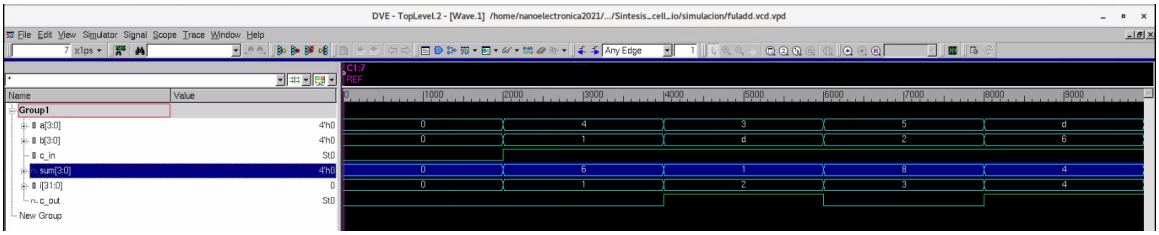


Figura 50: Resultados de simulación para un *FullAdder* de 4 bits

Para el caso de la ALU se realizaron múltiples pruebas para confirmar su funcionamiento, este modulo tiene la capacidad de seleccionar que tipo de operación se desea efectuar de la siguiente tabla. Para realizar las pruebas se usaron los valores 4'h7 y 4'h8 como entradas para todas las operaciones, en el caso de T2 se tiene el operador de suma y nos da como resultado 8'h0F, para T3 se realiza el operador de resta y nos da como resultados 8'h01, en T4 se realiza la multiplicación de ambos valores y nos da como resultados 8'h38, así sucesivamente para todas las operaciones y resultados .

Selección de Operador	operación
000	a + b
001	a - b
010	a * b
011	a & b
100	a | b
101	a , b
110	a ∧ b

Cuadro 2: Tabla de operaciones realizables por la ALU

Pin	T0	T1	T2	T3	T4	T5	T6	T7	T8
a (dato 1)	0	0	8	8	8	8	8	8	8
b (dato 2)	0	0	7	7	7	7	7	7	7
oper	0	0	0	1	2	3	4	5	6
EN	0	1	1	1	1	1	1	1	1
data	0	0	0F	01	38	00	0F	78	0F

Cuadro 3: Tabla de comportamiento para una ALU de 4 bits

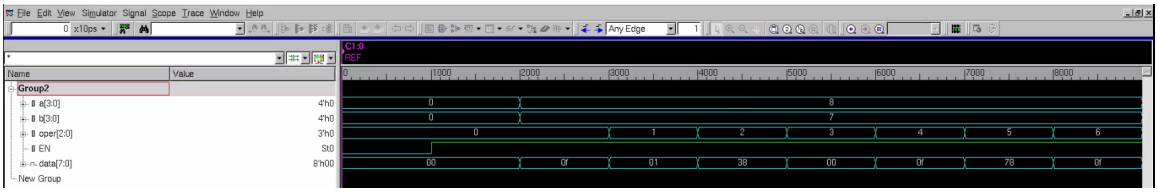


Figura 51: Resultados de simulación para una ALU con registros de 4 bits

Para realizar las pruebas de funcionamiento de nuestro módulo de RAM, se realizó la

carga de un valor hacia uno de los registros de la memoria, para realizar la escritura del valor se debe establecer la dirección a escribir en el pin 'a' y activar el pin 'we', esto permitirá almacenar el valor de 'di' en la ubicación que se desee, luego de activar la escritura se debe esperar al siguiente *posedge* del reloj para tener el valor almacenado, luego de esto se puede deshabilitar la escritura y leer la dirección de memoria que se visualizara en los pines 'do'. Al momento de realizar la lectura de una dirección que aún no ha sido establecida, esta se mostrara como una línea roja que significa alta impedancia.

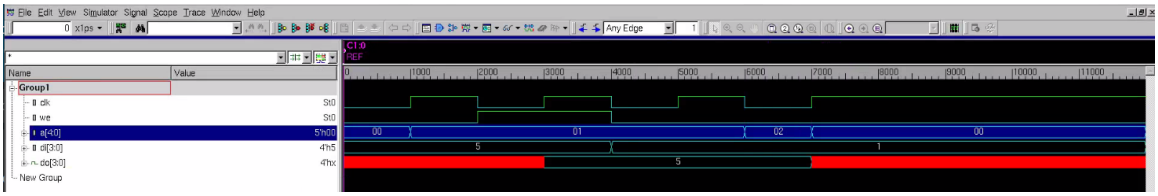


Figura 52: Resultados de simulación para una memoria RAM de 32 registros

CAPÍTULO 9

Diseño de circuito a fabricar

Para el diseño del último circuito, al cual denominamos "El Gran Jaguar", que se analizara en este documento, y el cual puede llegar a ser fabricado en silicio, se decidió crear una máquina de estados finitos que contenga dos textos en lenguaje hexadecimal (los cuales representan caracteres ASCII). El primer texto que se desea construir con la máquina de estados finitos es una pequeña introducción del chip y donde fue desarrollado, el segundo texto es la lista de participantes que contribuyeron al desarrollo de este. La construcción de este circuito se realizó en verilog con ayuda de Python, debido que la estructura que tiene el circuito es repetitiva y se debe escribir cada uno de los caracteres del texto en bits, por lo que hacerlo a mano se volvería algo tedioso y poco práctico.

9.1. Automatización con Python

Se empleo Python para realizar la escritura del circuito en verilog, debido que la forma en la que se decidió describir el comportamiento del circuito se basa en una cadena de If y Else que muestra un valor distinto en la salida dependiendo del valor de un contador interno. Lo que hace en esencia el código de Python en crear un archivo. v(Verilog), luego procede a escribir todo el código de verilog que no es repetitivo, pero cuando llega a la parte que corresponde a los textos que deseamos desplegar empieza a realizar la conversión de los caracteres a cadenas de 8bits, estos valores son colocados en una estructura condicional que avanza uno por uno de los caracteres y los muestra en los pines de salida de nuestro chip. El código utilizado puede encontrarse en la sección 13.4

9.2. Síntesis lógica

Luego de haber obtenido el código para El Gran Jaguar procedemos a replicar el proceso que se ha hecho en capítulos anteriores, lo primero que se debe realizar es la simulación del circuito para comprobar que lo que se sintetizara efectivamente cuenta con el comportamiento que se desea, utilizando las herramientas de VCS y DVE obtenemos el comportamiento de la Figura 53. En los resultados podemos observar múltiples pines sin embargo los importantes son la entrada select, la cual nos permite seleccionar cuál de los dos textos deseamos mostrar, y el otro es la salida q_out en el cual se muestran los 8 bits que representan cada uno de los caracteres del texto que se seleccionara con la entrada, al traducir los pares hexadecimales de cada ciclo obtenemos que se arma el texto 'Hola, El eq' en donde Hola seria hasta donde el valor de select cambia de 0 a 3 en la simulación y se estaba mostrando el primer texto luego del cambio los caracteres corresponden al segundo texto. Con esto comprobamos que el circuito si se comporta como deseamos.

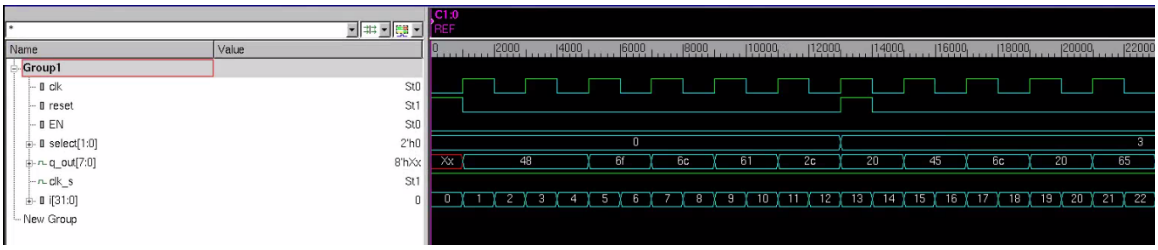


Figura 53: Resultados de simulación para el circuito del Gran Jaguar

Luego se procede a realizar la síntesis lógica del circuito sin los pines de entrada y salida, este proceso se encuentra descrito en capítulos anteriores, al introducir nuestro circuito en la herramienta de Design Vision obtenemos el diseño *BlackBox* de la Figura 54 correspondiente al Gran Jaguar, en este se puede comprobar que todos los pines que se definieron se hayan creado correctamente

Figura 54: BlackBox para el circuito del Gran Jaguar

Luego se proceder a realizar la síntesis del diseño, para esto es necesario realizar la definición del reloj para el sistema de acuerdo con los capítulos anteriores y el resto de las restricciones necesarias para el sistema. luego de realizar la síntesis se puede observar el esquemático del diseño de la Figura 55, querer analizar el diseño visualmente ya no es posible debido a la gran complejidad de este por lo que es necesario efectuar la simulación del diseño nuevamente para comprobar que el diseño es correcto. Al volver a ejecutar la simulación obtenemos el comportamiento de la Figura 56 en donde podemos apreciar que el comportamiento sigue siendo el mismo que el obtenido antes de la síntesis.

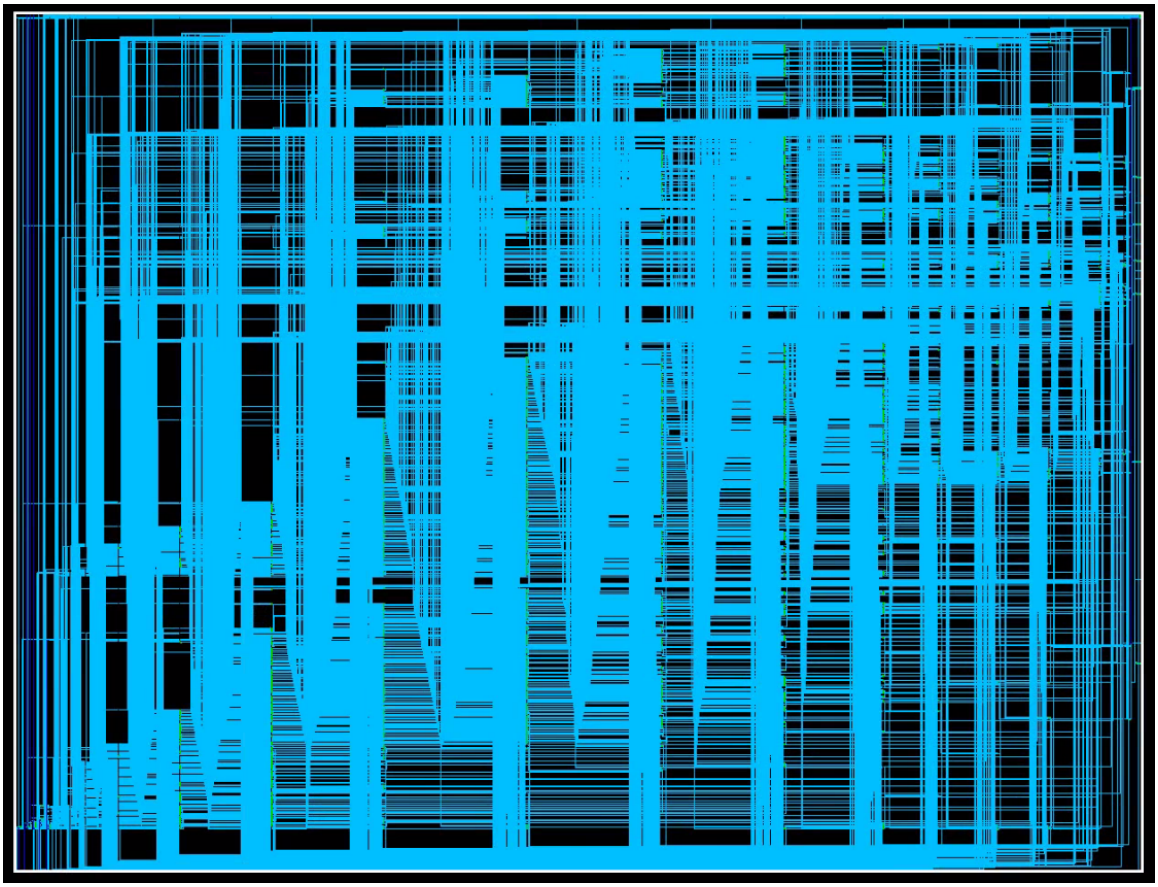


Figura 55: Síntesis lógica para el circuito del Gran Jaguar

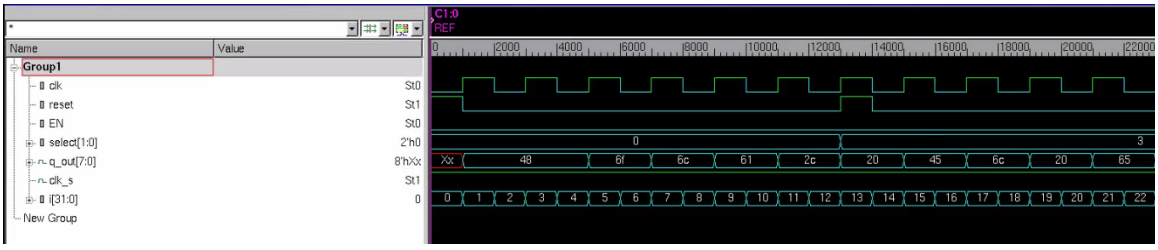


Figura 56: Resultados de simulación para el circuito del Gran Jaguar sintetizado

La última etapa por la que debe pasar el diseño del Gran Jaguar consiste en realizar la

síntesis de los pines de entrada y salida junto con el diseño final que obtuvimos previamente, el módulo que se utiliza para realizar las interconexiones de los pines I/O con el módulo sintetizado es el siguiente:

```
module chip_IO ( q, reset, clk, EN, clk_s, select);
input reset, clk, EN;
input [1:0] select;
output [7:0] q;
output clk_s;
wire [7:0] q_w;
wire [1:0] select_w;
wire reset_w, clk_w, EN_w, clk_s_w;

chip_SP chip (.q_out(q_w), .reset(reset_w), .clk(clk_w), .EN(EN_w), .clk_s(
    clk_s_w), .select(select_w));


//Entradas (C):
PDDW0204SCDG IN00 (1'b0,1'b0,1'b1,clk,clk_w,1'b0,1'b1);
PDDW0204SCDG IN01 (1'b0,1'b0,1'b1,reset,reset_w,1'b0,1'b1);
PDDW0204SCDG IN02 (1'b0,1'b0,1'b1,EN,EN_w,1'b0,1'b1);
PDDW0204SCDG IN03 (1'b0,1'b0,1'b1,select[0],select_w[0],1'b0,1'b1);
PDDW0204SCDG IN04 (1'b0,1'b0,1'b1,select[1],select_w[1],1'b0,1'b1);


//Salidas (I):
 //Prueba con C igual a 1'bx
PDDW0204SCDG OUT00 (q_w[0],1'b0,1'b0,q[0],1'bx,1'b0,1'b0);
PDDW0204SCDG OUT01 (q_w[1],1'b0,1'b0,q[1],1'bx,1'b0,1'b0);
PDDW0204SCDG OUT02 (q_w[2],1'b0,1'b0,q[2],1'bx,1'b0,1'b0);
PDDW0204SCDG OUT03 (q_w[3],1'b0,1'b0,q[3],1'bx,1'b0,1'b0);
PDDW0204SCDG OUT04 (q_w[4],1'b0,1'b0,q[4],1'bx,1'b0,1'b0);
PDDW0204SCDG OUT05 (q_w[5],1'b0,1'b0,q[5],1'bx,1'b0,1'b0);
PDDW0204SCDG OUT06 (q_w[6],1'b0,1'b0,q[6],1'bx,1'b0,1'b0);
PDDW0204SCDG OUT07 (q_w[7],1'b0,1'b0,q[7],1'bx,1'b0,1'b0);
PDDW0204SCDG OUT08 (clk_s_w,1'b0,1'b0,clk_s,1'bx,1'b0,1'b0);

// Pines alimentacion
PVDD1CDG PVDD();
PVSS1CDG PVSS();
endmodule
```

Luego de realizar la implementación de los pines de entrada y salida se debe realizar la síntesis nuevamente del nuevo código, al observar el esquemático de Design Vision con nuestro nuevo código obtenemos el diseño de la Figura 57 en el cual se pueden observar los módulos utilizados como pines I/O y en el centro existe un bloque denominado 'chip' el cual contiene el diseño de nuestro circuito sintetizado previamente. Luego de haber realizado la síntesis del nuevo diseño se debe realizar la simulación del sistema nuevo para comprobar que su funcionamiento es correcto, al analizar los resultados de las simulación en la Figura 58 se puede concluir que el funcionamiento del diseño que se obtuvo de la síntesis es el correcto. El *netlist* final de este circuito se puede encontrar en la sección 13.5

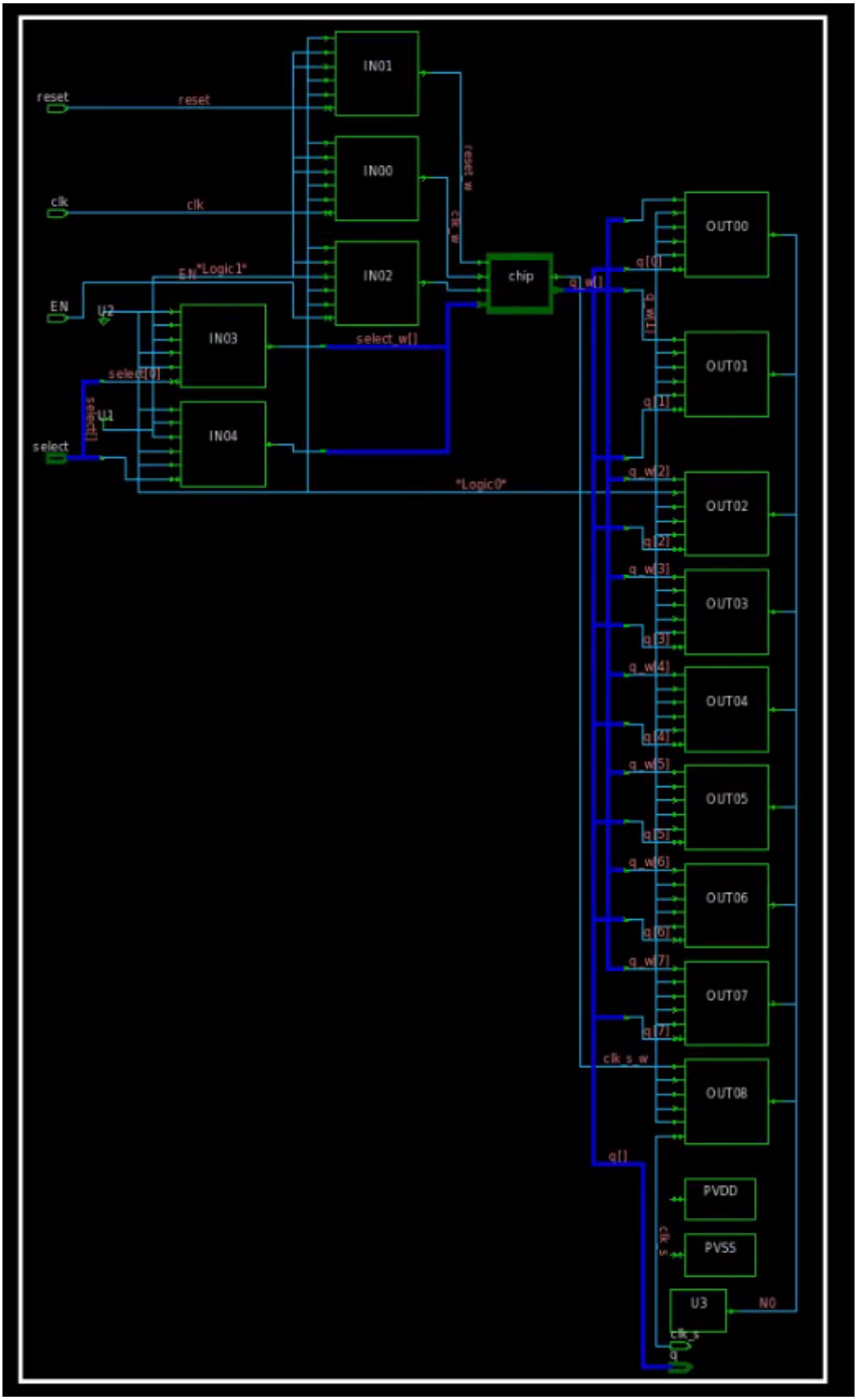


Figura 57: Esquemático final del Gran Jaguar sintetizado

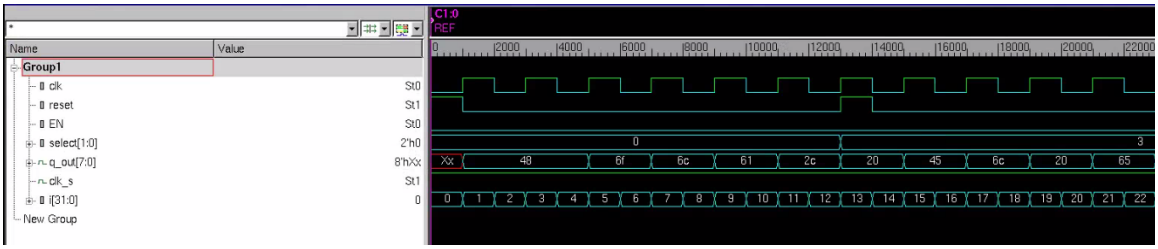


Figura 58: Resultados de simulación para el circuito del Gran Jaguar sintetizado

CAPÍTULO 10

Conclusiones

- Realizar las pruebas de funcionamiento al circuito original, al circuito sintetizado y al circuito sintetizado incluyendo los pines I/O, garantiza que el proceso de síntesis lógica se efectuó correctamente.
- Los cambios que se realizaron al *script* original utilizado para la síntesis lógica, permiten lograr que el diseño no contenga redundancias y la síntesis del circuito no se realice asumiendo comportamientos ideales.
- El proceso de síntesis lógica fue correcto tanto para circuitos puramente combinacionales y circuitos que requieren sincronía.
- El módulo de *Ring Oscillator* del "Gran Jaguar" fue el único que no se logró simular exitosamente, debido que el funcionamiento de este depende del comportamiento no ideal de los componentes, y la herramienta de VCS que se encarga de realizar las simulaciones asume que los componentes son ideales.
- Fue posible utilizar Python para automatizar y facilitar el proceso de creación de una máquina de estados finitos compleja en lenguaje verilog.

CAPÍTULO 11

Recomendaciones

- Realizar una exploración profunda de la herramienta DVE debido que esta cuenta con otras funciones a parte de realizar la visualización del *wave form* de las simulaciones, una de ellas es el uso de *Breakpoints* en simulaciones interactivas lo cual sería de gran ayuda al momento de hacer *debugging* al código.
- Explorar el comportamiento de circuitos síncronos con mayor complejidad que implementen un mayor número conexiones, celdas y *flip-flops*, para confirmar que el proceso de síntesis lógica es replicable independientemente de la complejidad.
- Llevar a cabo la automatización del proceso para la implementación de los pines de entrada y salida a la primera síntesis que se realiza, debido que este proceso puede llegar a ser una fuente de error al tener un gran número de pines.
- Ejecutar la simulación del *Ring oscillator* utilizando un FPGA, el cual debería ser capaz de implementarlo exitosamente y obtener el comportamiento correcto debido a que este utiliza compuertas no ideales.

CAPÍTULO 12

Bibliografía

[1] K. S. Cardona, «Mejoramiento del proceso de síntesis lógica llevada a cabo para la elaboración de un circuito integrado a escala nanométrica,» en *Trabajo de graduación en modalidad de Tesis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2021.

[2] J. d. l. Santos, «Diseño de un sumador/restador de 32 bits con tecnología CMOS en un proceso de 28 nanómetros usando aplicaciones de diseño de la empresa Synopsys,» en *Trabajo de graduación en modalidad de Tesis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2014.

[3] S. H. Rubio, «Definición del Flujo de Diseño para Fabricación de un Chip con Tecnologia VLSI CMOS,» en *Trabajo de graduación en modalidad de Tesis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2019.

[4] L. A. Najera, «Implementación de circuitos sintetizados a nivel netlist a partir de un diseño en lenguaje descriptivo de hardware como primer paso en el flujo de diseño de un circuito integrado,» en *Trabajo de graduación en modalidad de Tesis*, Facultad de Ingeniería Universidad del Valle de Guatemala, 2019.

[5] N. Weste y D. M. Harris, «A Circuits and Systems Perspective,» en *CMOS VLSI Design*, Pearson Education, Inc., 2011.

[6] H. Paul y H. Winfield, «The Art Of Electronics,» Cambridge University, 2015.

[7] IEEE, «Hardware Description Language,» en *Standard Verilog*, The Institute of Electrical y Electronics Engineers, 2001.

[8] X. Liming, «VLSI Circuit Design Mathodology Demystified,» The Institute of Electrical y Electronics Engineers, 2008.

[9] Solvnet.com, «Synopsys Corporate Backgrounder Spring 2018,» Synopsys, 2008.

[10] Synopsys, «Design Vision User Guide R-2020.09,» Solvnet, 2020.

[11] ——, «VCS User Guide Q-2020.03,» Solvnet, 2020.

[12] ——, «DVE (Discovery Visualization Environment) User Guide Q-2020.03,» Solvnet, 2020.

[13] ——, «Synthesis Tool Commands R-2020.09,» Solvnet, 2020.

CAPÍTULO 13

Anexos

13.1. *Netlist* final de la síntesis lógica de la compuerta *XOR* utilizando múltiples celdas:

```
module Not_0 ( A, Y );
  input A;
  output Y;


  CKND0BWP7T U1 ( .I(A), .ZN(Y) );
endmodule


module Not_2 ( A, Y );
  input A;
  output Y;


  CKND0BWP7T U1 ( .I(A), .ZN(Y) );
endmodule


module And_0 ( A, B, Y );
  input A, B;
  output Y;


  AN2D0BWP7T U1 ( .A1(B), .A2(A), .Z(Y) );
endmodule


module And_1 ( A, B, Y );
  input A, B;
  output Y;


```

```
  AN2D0BWP7T U1 ( .A1(B), .A2(A), .Z(Y) );
endmodule


module Or ( A, B, Y );
  input A, B;
  output Y;


  OR2D0BWP7T U1 ( .A1(A), .A2(B), .Z(Y) );
endmodule


module Not_1 ( A, Y );
  input A;
  output Y;


  CKND0BWP7T U1 ( .I(A), .ZN(Y) );
endmodule


module Xor ( A, B, Y, Y_neg );
  input A, B;
  output Y, Y_neg;
  wire   Aneg, Bneg, And1, And2;

  Not_0 U10 ( .A(A), .Y(Aneg) );
  Not_2 U11 ( .A(B), .Y(Bneg) );
  And_0 U12 ( .A(A), .B(Bneg), .Y(And1) );
  And_1 U13 ( .A(B), .B(Aneg), .Y(And2) );
  Or U14 ( .A(And1), .B(And2), .Y(Y) );
  Not_1 U15 ( .A(Y), .Y(Y_neg) );
endmodule


module Xor_IO ( A, B, Y, Y_neg );
  input A, B;
  output Y, Y_neg;
  wire   A_w, B_w, Y_w, Y_negw, n1, n2;
  tri   A;
  tri   B;
  tri   Y;
  tri   Y_neg;

  Xor compuerta ( .A(A_w), .B(B_w), .Y(Y_w), .Y_neg(Y_negw) );
  PDDW0204SCDG U20 ( .I(n1), .OEN(n2), .IE(n2), .PAD(A), .DS(n1), .PE(n1), .
    C(A_w) );
  PDDW0204SCDG U21 ( .I(n1), .OEN(n2), .IE(n2), .PAD(B), .DS(n1), .PE(n1), .
    C(B_w) );
  PDDW0204SCDG U22 ( .I(Y_w), .OEN(n1), .IE(n1), .PAD(Y), .DS(n1), .PE(n1) )
    ;
  PDDW0204SCDG U23 ( .I(Y_negw), .OEN(n1), .IE(n1), .PAD(Y_neg), .DS(n1), .
    PE(n1) );
  TIELBWP7T U6 ( .ZN(n1) );
  TIEHBWP7T U7 ( .Z(n2) );
endmodule
```

Netlist final para la síntesis lógica de un *Full-Adder* de *4bits*:

```
module fulladd ( a, b, c_in, c_out, sum );
  input [3:0] a;
  input [3:0] b;
  output [3:0] sum;
  input c_in;
  output c_out;
  wire   n6, n3, n5, n4, n7, n8, n9;

  MAOI222D1BWP7T U3 ( .A(n6), .B(a[2]), .C(b[2]), .ZN(n3) );
  INVD1BWP7T U4 ( .I(n3), .ZN(n5) );
  MAOI222D1BWP7T U5 ( .A(a[3]), .B(n5), .C(b[3]), .ZN(n4) );
  INVD1BWP7T U6 ( .I(n4), .ZN(c_out) );
  XOR3D0BWP7T U7 ( .A1(b[3]), .A2(a[3]), .A3(n5), .Z(sum[3]) );
  XOR3D0BWP7T U8 ( .A1(b[2]), .A2(a[2]), .A3(n6), .Z(sum[2]) );
  XOR3D0BWP7T U9 ( .A1(b[1]), .A2(a[1]), .A3(n7), .Z(sum[1]) );
  XOR3D0BWP7T U10 ( .A1(c_in), .A2(b[0]), .A3(a[0]), .Z(sum[0]) );
  IOA21D0BWP7T U11 ( .A1(n7), .A2(a[1]), .B(n8), .ZN(n6) );
  OAI21D0BWP7T U12 ( .A1(a[1]), .A2(n7), .B(b[1]), .ZN(n8) );
  IOA21D0BWP7T U13 ( .A1(a[0]), .A2(b[0]), .B(n9), .ZN(n7) );
  OAI21D0BWP7T U14 ( .A1(a[0]), .A2(b[0]), .B(c_in), .ZN(n9) );
endmodule


module fulladd_io ( a, b, c_in, c_out, sum );
  input [3:0] a;
  input [3:0] b;
  output [3:0] sum;
  input c_in;
  output c_out;
  wire   N0, c_in_w, c_out_w;
  wire   [3:0] a_w;
  wire   [3:0] b_w;
  wire   [3:0] sum_w;
  tri   [3:0] a;
  tri   [3:0] b;
  tri   c_in;
  tri   c_out;
  tri   [3:0] sum;

  fulladd celda ( .a(a_w), .b(b_w), .c_in(c_in_w), .c_out(c_out_w), .sum(
    sum_w) );
  PDDW0204SCDG IN00 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(a[0]), .DS(1'b0)
    ,
       .PE(1'b0), .C(a_w[0]) );
  PDDW0204SCDG IN01 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(a[1]), .DS(1'b0)
    ,
       .PE(1'b0), .C(a_w[1]) );
  PDDW0204SCDG IN02 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(a[2]), .DS(1'b0)
    ,
       .PE(1'b0), .C(a_w[2]) );
  PDDW0204SCDG IN03 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(a[3]), .DS(1'b0)
    ,
       .PE(1'b0), .C(a_w[3]) );
  PDDW0204SCDG IN04 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(b[0]), .DS(1'b0)
    ,
       .PE(1'b0), .C(b_w[0]) );
  PDDW0204SCDG IN05 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(b[1]), .DS(1'b0)
```

```
    ,
52        .PE(1'b0), .C(b_w[1]) );
53  PDDW0204SCDG IN06 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(b[2]), .DS(1'b0)
    ,
54        .PE(1'b0), .C(b_w[2]) );
55  PDDW0204SCDG IN07 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(b[3]), .DS(1'b0)
    ,
56        .PE(1'b0), .C(b_w[3]) );
57  PDDW0204SCDG IN08 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(c_in), .DS(1'b0)
    ,
58        .PE(1'b0), .C(c_in_w) );
59  PDDW0204SCDG OUT00 ( .I(sum_w[0]), .OEN(1'b0), .IE(1'b0), .PAD(sum[0]), .
    DS(
60        1'b0), .PE(1'b0), .C(N0) );
61  PDDW0204SCDG OUT01 ( .I(sum_w[1]), .OEN(1'b0), .IE(1'b0), .PAD(sum[1]), .
    DS(
62        1'b0), .PE(1'b0), .C(N0) );
63  PDDW0204SCDG OUT02 ( .I(sum_w[2]), .OEN(1'b0), .IE(1'b0), .PAD(sum[2]), .
    DS(
64        1'b0), .PE(1'b0), .C(N0) );
65  PDDW0204SCDG OUT03 ( .I(sum_w[3]), .OEN(1'b0), .IE(1'b0), .PAD(sum[3]), .
    DS(
66        1'b0), .PE(1'b0), .C(N0) );
67  PDDW0204SCDG OUT04 ( .I(c_out_w), .OEN(1'b0), .IE(1'b0), .PAD(c_out), .DS(
68        1'b0), .PE(1'b0), .C(N0) );
69  PVDD1CDG VDDP (  );
70  PVSS1CDG VSSP (  );  assign N0 = 1'b0;
71
72  endmodule
```

13.2. *Netlist* final para la síntesis lógica de una ALU de *4bits*:

```
1   module ALU_4bit ( data, a, b, oper, EN );
2     output [7:0] data;
3     input [3:0] a;
4     input [3:0] b;
5     input [2:0] oper;
6     input EN;
7     wire   n134, n148, n78, n90, n72, n69, n70, n71, n73, n74, n75, n76, n80,
8           n81, n82, n83, n84, n85, n86, n87, n88, n89, n77, n79, n91, n92,
      n93,
9           n94, n95, n96, n97, n98, n99, n100, n101, n102, n103, n104, n105,
10          n106, n107, n108, n109, n110, n111, n112, n113, n114, n115, n116,
11          n117, n118, n119, n120, n121, n122, n123, n124, n125, n126, n127,
12          n128, n129, n130, n131, n132, n133, n135, n136, n137, n138, n139,
13          n140, n141, n142, n143, n144, n145, n146, n147, n149, n150, n151,
14          n152, n153, n154, n155, n156, n157, n158, n159, n160, n161, n163,
15          n164, n165, n166, n167, n168, n169, n170, n171, n172, n162, n173,
16          n174, n175;
17
18    CKMUX2D1BWP7T U70 ( .I0(n134), .I1(a[0]), .S(b[0]), .Z(n148) );
19    OAI211D1BWP7T U71 ( .A1(n78), .A2(n90), .B(a[3]), .C(b[3]), .ZN(n72) );
20    CKND2D0BWP7T U72 ( .A1(n69), .A2(n70), .ZN(data[7]) );
21    ND4D0BWP7T U73 ( .A1(oper[2]), .A2(oper[0]), .A3(EN), .A4(n71), .ZN(n70) )
      ;
22    OAI211D0BWP7T U74 ( .A1(n72), .A2(n73), .B(n74), .C(n69), .ZN(data[6]) );
```

```
  ND3D0BWP7T U75 ( .A1(EN), .A2(n75), .A3(n76), .ZN(n74) );
  OAI211D0BWP7T U76 ( .A1(n73), .A2(n80), .B(n81), .C(n69), .ZN(data[5]) );
  CKND2D0BWP7T U77 ( .A1(EN), .A2(n82), .ZN(n69) );
  ND3D0BWP7T U78 ( .A1(EN), .A2(n83), .A3(n76), .ZN(n81) );
  MUX2ND0BWP7T U79 ( .I0(n84), .I1(n85), .S(n78), .ZN(n80) );
  OAI21D0BWP7T U80 ( .A1(n86), .A2(n87), .B(n88), .ZN(n78) );
  IOA21D0BWP7T U81 ( .A1(n87), .A2(n86), .B(n89), .ZN(n88) );
  CKND2D0BWP7T U82 ( .A1(n77), .A2(n79), .ZN(n85) );
  XNR2D1BWP7T U83 ( .A1(n77), .A2(n79), .ZN(n84) );
  CKND2D0BWP7T U84 ( .A1(b[3]), .A2(n90), .ZN(n79) );
  OAI22D0BWP7T U85 ( .A1(n91), .A2(n92), .B1(n93), .B2(n73), .ZN(data[4]) );
  XNR3D0BWP7T U86 ( .A1(n87), .A2(n86), .A3(n89), .ZN(n93) );
  OAI21D0BWP7T U87 ( .A1(n94), .A2(n95), .B(n96), .ZN(n89) );
  IOA21D0BWP7T U88 ( .A1(n95), .A2(n94), .B(n97), .ZN(n96) );
  XNR2D1BWP7T U89 ( .A1(n98), .A2(n90), .ZN(n86) );
  MOAI22D0BWP7T U90 ( .A1(n99), .A2(n100), .B1(n101), .B2(n102), .ZN(n90) );
  NR2D0BWP7T U91 ( .A1(n102), .A2(n101), .ZN(n100) );
  NR2D0BWP7T U92 ( .A1(n75), .A2(n103), .ZN(n98) );
  CKND2D0BWP7T U93 ( .A1(a[3]), .A2(b[2]), .ZN(n87) );
  AOI221D0BWP7T U94 ( .A1(n76), .A2(n104), .B1(n105), .B2(n106), .C(n82), .
    ZN(
        n91) );
  OAI21D0BWP7T U95 ( .A1(n107), .A2(n106), .B(n108), .ZN(n82) );
  MOAI22D0BWP7T U96 ( .A1(n109), .A2(n71), .B1(n110), .B2(n111), .ZN(n106) )
    ;
  NR2D0BWP7T U97 ( .A1(n111), .A2(n110), .ZN(n109) );
  AOI31D0BWP7T U98 ( .A1(n112), .A2(n113), .A3(n114), .B(n92), .ZN(data[3])
    );
  AOI22D0BWP7T U99 ( .A1(n77), .A2(n115), .B1(n116), .B2(n117), .ZN(n114) );
  XOR3D0BWP7T U100 ( .A1(n97), .A2(n94), .A3(n95), .Z(n116) );
  ND3D0BWP7T U101 ( .A1(b[0]), .A2(n118), .A3(a[3]), .ZN(n95) );
  XOR3D0BWP7T U102 ( .A1(n101), .A2(n99), .A3(n102), .Z(n94) );
  IAO21D0BWP7T U103 ( .A1(n119), .A2(n120), .B(n121), .ZN(n99) );
  NR2D0BWP7T U104 ( .A1(n103), .A2(n83), .ZN(n101) );
  AN2D0BWP7T U105 ( .A1(b[1]), .A2(a[3]), .Z(n97) );
  NR2D0BWP7T U106 ( .A1(n71), .A2(n103), .ZN(n77) );
  MUX2ND0BWP7T U107 ( .I0(n122), .I1(n123), .S(n71), .ZN(n113) );
  CKND0BWP7T U108 ( .I(a[3]), .ZN(n71) );
  OAI21D0BWP7T U109 ( .A1(n124), .A2(n125), .B(n126), .ZN(n123) );
  MUX2ND0BWP7T U110 ( .I0(n127), .I1(n128), .S(n103), .ZN(n126) );
  XNR2D1BWP7T U111 ( .A1(n111), .A2(n110), .ZN(n125) );
  OAI22D0BWP7T U112 ( .A1(b[3]), .A2(n129), .B1(n130), .B2(n124), .ZN(n122)
    );
  CKXOR2D0BWP7T U113 ( .A1(n111), .A2(n110), .Z(n130) );
  IOA21D0BWP7T U114 ( .A1(n131), .A2(n132), .B(n133), .ZN(n110) );
  OAI21D0BWP7T U115 ( .A1(n132), .A2(n131), .B(a[2]), .ZN(n133) );
  XNR2D1BWP7T U116 ( .A1(n103), .A2(n134), .ZN(n111) );
  MUX2ND0BWP7T U117 ( .I0(n135), .I1(n76), .S(n103), .ZN(n112) );
  CKND0BWP7T U118 ( .I(b[3]), .ZN(n103) );
  AOI31D0BWP7T U119 ( .A1(n136), .A2(n137), .A3(n138), .B(n92), .ZN(data[2])
        );
  AOI22D0BWP7T U120 ( .A1(n102), .A2(n115), .B1(n117), .B2(n139), .ZN(n138)
    );
  XNR2D1BWP7T U121 ( .A1(n118), .A2(n140), .ZN(n139) );
  CKND2D0BWP7T U122 ( .A1(a[3]), .A2(b[0]), .ZN(n140) );
  XOR3D0BWP7T U123 ( .A1(n119), .A2(n121), .A3(n120), .Z(n118) );
  CKND2D0BWP7T U124 ( .A1(a[2]), .A2(b[1]), .ZN(n120) );
  NR3D0BWP7T U125 ( .A1(n75), .A2(n141), .A3(n142), .ZN(n121) );
  CKND2D0BWP7T U126 ( .A1(b[2]), .A2(a[1]), .ZN(n119) );
```

```
77  INR2D0BWP7T U127 ( .A1(b[2]), .B1(n75), .ZN(n102) );
78  MUX2ND0BWP7T U128 ( .I0(n143), .I1(n144), .S(n75), .ZN(n137) );
79  CKND0BWP7T U129 ( .I(a[2]), .ZN(n75) );
80  OAI21D0BWP7T U130 ( .A1(n124), .A2(n145), .B(n146), .ZN(n144) );
81  MUX2ND0BWP7T U131 ( .I0(n128), .I1(n127), .S(b[2]), .ZN(n146) );
82  XNR2D1BWP7T U132 ( .A1(n132), .A2(n131), .ZN(n145) );
83  OAI22D0BWP7T U133 ( .A1(b[2]), .A2(n129), .B1(n147), .B2(n124), .ZN(n143)
     );
84  CKXOR2D0BWP7T U134 ( .A1(n132), .A2(n131), .Z(n147) );
85  IOA21D0BWP7T U135 ( .A1(n148), .A2(n149), .B(n150), .ZN(n131) );
86  OAI21D0BWP7T U136 ( .A1(n149), .A2(n148), .B(a[1]), .ZN(n150) );
87  CKXOR2D0BWP7T U137 ( .A1(b[2]), .A2(n134), .Z(n132) );
88  MUX2ND0BWP7T U138 ( .I0(n76), .I1(n135), .S(b[2]), .ZN(n136) );
89  AOI31D0BWP7T U139 ( .A1(n151), .A2(n152), .A3(n153), .B(n92), .ZN(data[1])
90        );
91  CKND0BWP7T U140 ( .I(EN), .ZN(n92) );
92  MAOI22D0BWP7T U141 ( .A1(n117), .A2(n154), .B1(n142), .B2(n155), .ZN(n153)
93        );
94  CKXOR2D0BWP7T U142 ( .A1(n156), .A2(n142), .Z(n154) );
95  CKND2D0BWP7T U143 ( .A1(a[1]), .A2(b[1]), .ZN(n142) );
96  CKND2D0BWP7T U144 ( .A1(a[2]), .A2(b[0]), .ZN(n156) );
97  MUX2ND0BWP7T U145 ( .I0(n157), .I1(n158), .S(n83), .ZN(n152) );
98  OAI21D0BWP7T U146 ( .A1(n124), .A2(n159), .B(n160), .ZN(n158) );
99  MUX2ND0BWP7T U147 ( .I0(n128), .I1(n127), .S(b[1]), .ZN(n160) );
100 XNR2D1BWP7T U148 ( .A1(n149), .A2(n148), .ZN(n159) );
101 OAI22D0BWP7T U149 ( .A1(b[1]), .A2(n129), .B1(n161), .B2(n124), .ZN(n157)
     );
102 CKXOR2D0BWP7T U150 ( .A1(n149), .A2(n148), .Z(n161) );
103 CKXOR2D0BWP7T U151 ( .A1(b[1]), .A2(n134), .Z(n149) );
104 MUX2ND0BWP7T U152 ( .I0(n76), .I1(n135), .S(b[1]), .ZN(n151) );
105 OAI31D0BWP7T U153 ( .A1(n73), .A2(n141), .A3(n83), .B(n163), .ZN(data[0])
     );
106 OAI21D0BWP7T U154 ( .A1(n164), .A2(n165), .B(EN), .ZN(n163) );
107 CKMUX2D0BWP7T U155 ( .I0(n76), .I1(n135), .S(b[0]), .Z(n165) );
108 NR2D0BWP7T U156 ( .A1(n129), .A2(oper[1]), .ZN(n135) );
109 NR3D0BWP7T U157 ( .A1(n166), .A2(oper[1]), .A3(n167), .ZN(n76) );
110 MUX2ND0BWP7T U158 ( .I0(n168), .I1(n169), .S(n104), .ZN(n164) );
111 CKND0BWP7T U159 ( .I(a[0]), .ZN(n104) );
112 NR2D0BWP7T U160 ( .A1(n170), .A2(n171), .ZN(n169) );
113 MUX2ND0BWP7T U161 ( .I0(n129), .I1(n108), .S(n141), .ZN(n171) );
114 CKND0BWP7T U162 ( .I(n128), .ZN(n108) );
115 NR2D0BWP7T U163 ( .A1(n155), .A2(n167), .ZN(n128) );
116 MUX2ND0BWP7T U164 ( .I0(n107), .I1(n172), .S(n162), .ZN(n170) );
117 NR2D0BWP7T U165 ( .A1(n173), .A2(n174), .ZN(n168) );
118 MUX2ND0BWP7T U166 ( .I0(n155), .I1(n129), .S(n141), .ZN(n174) );
119 CKND0BWP7T U167 ( .I(n127), .ZN(n129) );
120 NR2D0BWP7T U168 ( .A1(n167), .A2(oper[0]), .ZN(n127) );
121 CKND0BWP7T U169 ( .I(n115), .ZN(n155) );
122 NR2D0BWP7T U170 ( .A1(n175), .A2(n166), .ZN(n115) );
123 MUX2ND0BWP7T U171 ( .I0(n172), .I1(n107), .S(n162), .ZN(n173) );
124 XNR2D1BWP7T U172 ( .A1(n141), .A2(n134), .ZN(n162) );
125 CKND0BWP7T U173 ( .I(n134), .ZN(n107) );
126 CKND0BWP7T U174 ( .I(n105), .ZN(n172) );
127 NR2D0BWP7T U175 ( .A1(n124), .A2(n134), .ZN(n105) );
128 NR2D0BWP7T U176 ( .A1(n124), .A2(n166), .ZN(n134) );
129 CKND0BWP7T U177 ( .I(oper[0]), .ZN(n166) );
130 CKND2D0BWP7T U178 ( .A1(n175), .A2(n167), .ZN(n124) );
131 CKND0BWP7T U179 ( .I(oper[2]), .ZN(n167) );
132 CKND0BWP7T U180 ( .I(a[1]), .ZN(n83) );
```

```
  CKND0BWP7T U181 ( .I(b[0]), .ZN(n141) );
  CKND2D0BWP7T U182 ( .A1(n117), .A2(EN), .ZN(n73) );
  NR3D0BWP7T U183 ( .A1(oper[0]), .A2(oper[2]), .A3(n175), .ZN(n117) );
  CKND0BWP7T U184 ( .I(oper[1]), .ZN(n175) );
endmodule


module ALU_IO ( data, a, b, oper, EN );
  output [7:0] data;
  input [3:0] a;
  input [3:0] b;
  input [2:0] oper;
  input EN;
  wire   N0, EN_w;
  wire   [7:0] data_w;
  wire   [3:0] a_w;
  wire   [3:0] b_w;
  wire   [2:0] oper_w;
  tri   [7:0] data;
  tri   [3:0] a;
  tri   [3:0] b;
  tri   [2:0] oper;
  tri   EN;

  ALU_4bit cell  ( .data(data_w), .a(a_w), .b(b_w), .oper(oper_w), .EN(EN_w)
       );
  PDDW0204SCDG IN00 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(a[0]), .DS(1'b0)
    ,
       .PE(1'b0), .C(a_w[0]) );
  PDDW0204SCDG IN01 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(a[1]), .DS(1'b0)
    ,
       .PE(1'b0), .C(a_w[1]) );
  PDDW0204SCDG IN02 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(a[2]), .DS(1'b0)
    ,
       .PE(1'b0), .C(a_w[2]) );
  PDDW0204SCDG IN03 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(a[3]), .DS(1'b0)
    ,
       .PE(1'b0), .C(a_w[3]) );
  PDDW0204SCDG IN04 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(b[0]), .DS(1'b0)
    ,
       .PE(1'b0), .C(b_w[0]) );
  PDDW0204SCDG IN05 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(b[1]), .DS(1'b0)
    ,
       .PE(1'b0), .C(b_w[1]) );
  PDDW0204SCDG IN06 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(b[2]), .DS(1'b0)
    ,
       .PE(1'b0), .C(b_w[2]) );
  PDDW0204SCDG IN07 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(b[3]), .DS(1'b0)
    ,
       .PE(1'b0), .C(b_w[3]) );
  PDDW0204SCDG IN08 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(oper[0]), .DS(1'
    b0), .PE(1'b0), .C(oper_w[0]) );
  PDDW0204SCDG IN09 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(oper[1]), .DS(1'
    b0), .PE(1'b0), .C(oper_w[1]) );
  PDDW0204SCDG IN10 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(oper[2]), .DS(1'
    b0), .PE(1'b0), .C(oper_w[2]) );
  PDDW0204SCDG IN11 ( .I(1'b0), .OEN(1'b1), .IE(1'b1), .PAD(EN), .DS(1'b0),
       .PE(1'b0), .C(EN_w) );
  PDDW0204SCDG OUT00 ( .I(data_w[0]), .OEN(1'b0), .IE(1'b0), .PAD(data[0]),
```

```
        .DS(1'b0), .PE(1'b0), .C(N0) );
  PDDW0204SCDG OUT01 ( .I(data_w[1]), .OEN(1'b0), .IE(1'b0), .PAD(data[1]),
        .DS(1'b0), .PE(1'b0), .C(N0) );
  PDDW0204SCDG OUT02 ( .I(data_w[2]), .OEN(1'b0), .IE(1'b0), .PAD(data[2]),
        .DS(1'b0), .PE(1'b0), .C(N0) );
  PDDW0204SCDG OUT03 ( .I(data_w[3]), .OEN(1'b0), .IE(1'b0), .PAD(data[3]),
        .DS(1'b0), .PE(1'b0), .C(N0) );
  PDDW0204SCDG OUT04 ( .I(data_w[4]), .OEN(1'b0), .IE(1'b0), .PAD(data[4]),
        .DS(1'b0), .PE(1'b0), .C(N0) );
  PDDW0204SCDG OUT05 ( .I(data_w[5]), .OEN(1'b0), .IE(1'b0), .PAD(data[5]),
        .DS(1'b0), .PE(1'b0), .C(N0) );
  PDDW0204SCDG OUT06 ( .I(data_w[6]), .OEN(1'b0), .IE(1'b0), .PAD(data[6]),
        .DS(1'b0), .PE(1'b0), .C(N0) );
  PDDW0204SCDG OUT07 ( .I(data_w[7]), .OEN(1'b0), .IE(1'b0), .PAD(data[7]),
        .DS(1'b0), .PE(1'b0), .C(N0) );
  PVDD1CDG VDDP (  );
  PVSS1CDG VSSP (  );  assign N0 = 1'b0;

endmodule
```

13.3. *Netlist* final para la síntesis de una RAM de 4bits con 32 posiciones:

```
module raminfr ( clk, we, a, di, do );
  input [4:0] a;
  input [3:0] di;
  output [3:0] do;
  input clk, we;
  wire   N14, N13, N12, N11, N10, n215, \ram[31][3] , \ram[31][2] ,
        \ram[31][1] , \ram[31][0] , n216, \ram[30][3] , \ram[30][2] ,
        \ram[30][1] , \ram[30][0] , n217, \ram[29][3] , \ram[29][2] ,
        \ram[29][1] , \ram[29][0] , n218, \ram[28][3] , \ram[28][2] ,
        \ram[28][1] , \ram[28][0] , n219, \ram[27][3] , \ram[27][2] ,
        \ram[27][1] , \ram[27][0] , n220, \ram[26][3] , \ram[26][2] ,
        \ram[26][1] , \ram[26][0] , n221, \ram[25][3] , \ram[25][2] ,
        \ram[25][1] , \ram[25][0] , n222, \ram[24][3] , \ram[24][2] ,
        \ram[24][1] , \ram[24][0] , n223, \ram[23][3] , \ram[23][2] ,
        \ram[23][1] , \ram[23][0] , n224, \ram[22][3] , \ram[22][2] ,
        \ram[22][1] , \ram[22][0] , n225, \ram[21][3] , \ram[21][2] ,
        \ram[21][1] , \ram[21][0] , n226, \ram[20][3] , \ram[20][2] ,
        \ram[20][1] , \ram[20][0] , n227, \ram[19][3] , \ram[19][2] ,
        \ram[19][1] , \ram[19][0] , n228, \ram[18][3] , \ram[18][2] ,
        \ram[18][1] , \ram[18][0] , n229, \ram[17][3] , \ram[17][2] ,
        \ram[17][1] , \ram[17][0] , n230, \ram[16][3] , \ram[16][2] ,
        \ram[16][1] , \ram[16][0] , n231, \ram[15][3] , \ram[15][2] ,
        \ram[15][1] , \ram[15][0] , n232, \ram[14][3] , \ram[14][2] ,
        \ram[14][1] , \ram[14][0] , n233, \ram[13][3] , \ram[13][2] ,
        \ram[13][1] , \ram[13][0] , n234, \ram[12][3] , \ram[12][2] ,
        \ram[12][1] , \ram[12][0] , n235, \ram[11][3] , \ram[11][2] ,
        \ram[11][1] , \ram[11][0] , n236, \ram[10][3] , \ram[10][2] ,
        \ram[10][1] , \ram[10][0] , n237, \ram[9][3] , \ram[9][2] ,
        \ram[9][1] , \ram[9][0] , n238, \ram[8][3] , \ram[8][2] , \ram
    [8][1] ,
        \ram[8][0] , n239, \ram[7][3] , \ram[7][2] , \ram[7][1] , \ram
    [7][0] ,
```

```
        n240, \ram[6][3] , \ram[6][2] , \ram[6][1] , \ram[6][0] , n241,
        \ram[5][3] , \ram[5][2] , \ram[5][1] , \ram[5][0] , n242, \ram
    [4][3] ,
        \ram[4][2] , \ram[4][1] , \ram[4][0] , n243, \ram[3][3] , \ram
    [3][2] ,
        \ram[3][1] , \ram[3][0] , n244, \ram[2][3] , \ram[2][2] , \ram
    [2][1] ,
        \ram[2][0] , n245, \ram[1][3] , \ram[1][2] , \ram[1][1] , \ram
    [1][0] ,
        n246, \ram[0][3] , \ram[0][2] , \ram[0][1] , \ram[0][0] , n1, n2,
    n3,
        n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16, n17, n18
    ,
        n19, n20, n21, n22, n23, n24, n25, n26, n27, n28, n29, n30, n31,
    n32,
        n33, n34, n35, n36, n37, n38, n39, n40, n41, n42, n43, n44, n45,
    n46,
        n47, n48, n49, n50, n51, n52, n53, n54, n55, n56, n57, n58, n59,
    n60,
        n61, n62, n63, n64, n65, n66, n67, n68, n69, n70, n71, n72, n73,
    n74,
        n75, n76, n77, n78, n79, n80, n81, n82, n83, n84, n85, n86, n87,
    n88,
        n89, n90, n91, n92, n93, n94, n95, n96, n97, n98, n99, n100, n101,
        n102, n103, n104, n105, n106, n107, n108, n109, n110, n111, n112,
        n113, n114, n115, n116, n117, n118, n119, n120, n121, n122, n123,
        n124, n125, n126, n127, n128, n129, n130, n131, n132, n133, n134;

  DFQD1BWP7T \read_a_reg[4]  ( .D(a[4]), .CP(n2), .Q(N14) );
  DFQD1BWP7T \read_a_reg[3]  ( .D(a[3]), .CP(n1), .Q(N13) );
  DFQD1BWP7T \read_a_reg[2]  ( .D(a[2]), .CP(n2), .Q(N12) );
  DFQD1BWP7T \read_a_reg[1]  ( .D(a[1]), .CP(n2), .Q(N11) );
  DFQD1BWP7T \read_a_reg[0]  ( .D(a[0]), .CP(n1), .Q(N10) );
  EDFQD1BWP7T \ram_reg[31][3]  ( .D(di[3]), .E(n215), .CP(n2), .Q(\ram
    [31][3] ) );
  EDFQD1BWP7T \ram_reg[31][2]  ( .D(di[2]), .E(n215), .CP(n2), .Q(\ram
    [31][2] ) );
  EDFQD1BWP7T \ram_reg[31][1]  ( .D(di[1]), .E(n215), .CP(n1), .Q(\ram
    [31][1] ) );
  EDFQD1BWP7T \ram_reg[31][0]  ( .D(di[0]), .E(n215), .CP(n1), .Q(\ram
    [31][0] ) );
  EDFQD1BWP7T \ram_reg[30][3]  ( .D(di[3]), .E(n216), .CP(n1), .Q(\ram
    [30][3] ) );
  EDFQD1BWP7T \ram_reg[30][2]  ( .D(di[2]), .E(n216), .CP(n2), .Q(\ram
    [30][2] ) );
  EDFQD1BWP7T \ram_reg[30][1]  ( .D(di[1]), .E(n216), .CP(n1), .Q(\ram
    [30][1] ) );
  EDFQD1BWP7T \ram_reg[30][0]  ( .D(di[0]), .E(n216), .CP(n2), .Q(\ram
    [30][0] ) );
  EDFQD1BWP7T \ram_reg[29][3]  ( .D(di[3]), .E(n217), .CP(n1), .Q(\ram
    [29][3] ) );
  EDFQD1BWP7T \ram_reg[29][2]  ( .D(di[2]), .E(n217), .CP(n1), .Q(\ram
    [29][2] ) );
  EDFQD1BWP7T \ram_reg[29][1]  ( .D(di[1]), .E(n217), .CP(n2), .Q(\ram
    [29][1] ) );
  EDFQD1BWP7T \ram_reg[29][0]  ( .D(di[0]), .E(n217), .CP(clk), .Q(
        \ram[29][0] ) );
  EDFQD1BWP7T \ram_reg[28][3]  ( .D(di[3]), .E(n218), .CP(n1), .Q(\ram
    [28][3] ) );
```

```
67  EDFQD1BWP7T \ram_reg[28][2]  ( .D(di[2]), .E(n218), .CP(n1), .Q(\ram
      [28][2] ) );
68  EDFQD1BWP7T \ram_reg[28][1]  ( .D(di[1]), .E(n218), .CP(n1), .Q(\ram
      [28][1] ) );
69  EDFQD1BWP7T \ram_reg[28][0]  ( .D(di[0]), .E(n218), .CP(n2), .Q(\ram
      [28][0] ) );
70  EDFQD1BWP7T \ram_reg[27][3]  ( .D(di[3]), .E(n219), .CP(n1), .Q(\ram
      [27][3] ) );
71  EDFQD1BWP7T \ram_reg[27][2]  ( .D(di[2]), .E(n219), .CP(n2), .Q(\ram
      [27][2] ) );
72  EDFQD1BWP7T \ram_reg[27][1]  ( .D(di[1]), .E(n219), .CP(clk), .Q(
73        \ram[27][1] ) );
74  EDFQD1BWP7T \ram_reg[27][0]  ( .D(di[0]), .E(n219), .CP(n2), .Q(\ram
      [27][0] ) );
75  EDFQD1BWP7T \ram_reg[26][3]  ( .D(di[3]), .E(n220), .CP(n1), .Q(\ram
      [26][3] ) );
76  EDFQD1BWP7T \ram_reg[26][2]  ( .D(di[2]), .E(n220), .CP(n2), .Q(\ram
      [26][2] ) );
77  EDFQD1BWP7T \ram_reg[26][1]  ( .D(di[1]), .E(n220), .CP(n1), .Q(\ram
      [26][1] ) );
78  EDFQD1BWP7T \ram_reg[26][0]  ( .D(di[0]), .E(n220), .CP(n2), .Q(\ram
      [26][0] ) );
79  EDFQD1BWP7T \ram_reg[25][3]  ( .D(di[3]), .E(n221), .CP(n2), .Q(\ram
      [25][3] ) );
80  EDFQD1BWP7T \ram_reg[25][2]  ( .D(di[2]), .E(n221), .CP(n1), .Q(\ram
      [25][2] ) );
81  EDFQD1BWP7T \ram_reg[25][1]  ( .D(di[1]), .E(n221), .CP(n1), .Q(\ram
      [25][1] ) );
82  EDFQD1BWP7T \ram_reg[25][0]  ( .D(di[0]), .E(n221), .CP(n2), .Q(\ram
      [25][0] ) );
83  EDFQD1BWP7T \ram_reg[24][3]  ( .D(di[3]), .E(n222), .CP(n1), .Q(\ram
      [24][3] ) );
84  EDFQD1BWP7T \ram_reg[24][2]  ( .D(di[2]), .E(n222), .CP(n2), .Q(\ram
      [24][2] ) );
85  EDFQD1BWP7T \ram_reg[24][1]  ( .D(di[1]), .E(n222), .CP(n2), .Q(\ram
      [24][1] ) );
86  EDFQD1BWP7T \ram_reg[24][0]  ( .D(di[0]), .E(n222), .CP(n1), .Q(\ram
      [24][0] ) );
87  EDFQD1BWP7T \ram_reg[23][3]  ( .D(di[3]), .E(n223), .CP(n2), .Q(\ram
      [23][3] ) );
88  EDFQD1BWP7T \ram_reg[23][2]  ( .D(di[2]), .E(n223), .CP(n1), .Q(\ram
      [23][2] ) );
89  EDFQD1BWP7T \ram_reg[23][1]  ( .D(di[1]), .E(n223), .CP(n2), .Q(\ram
      [23][1] ) );
90  EDFQD1BWP7T \ram_reg[23][0]  ( .D(di[0]), .E(n223), .CP(n1), .Q(\ram
      [23][0] ) );
91  EDFQD1BWP7T \ram_reg[22][3]  ( .D(di[3]), .E(n224), .CP(n2), .Q(\ram
      [22][3] ) );
92  EDFQD1BWP7T \ram_reg[22][2]  ( .D(di[2]), .E(n224), .CP(n1), .Q(\ram
      [22][2] ) );
93  EDFQD1BWP7T \ram_reg[22][1]  ( .D(di[1]), .E(n224), .CP(n2), .Q(\ram
      [22][1] ) );
94  EDFQD1BWP7T \ram_reg[22][0]  ( .D(di[0]), .E(n224), .CP(n1), .Q(\ram
      [22][0] ) );
95  EDFQD1BWP7T \ram_reg[21][3]  ( .D(di[3]), .E(n225), .CP(n2), .Q(\ram
      [21][3] ) );
96  EDFQD1BWP7T \ram_reg[21][2]  ( .D(di[2]), .E(n225), .CP(n1), .Q(\ram
      [21][2] ) );
97  EDFQD1BWP7T \ram_reg[21][1]  ( .D(di[1]), .E(n225), .CP(n2), .Q(\ram
```

```
  [21][1] ) );
EDFQD1BWP7T \ram_reg[21][0]  ( .D(di[0]), .E(n225), .CP(n1), .Q(\ram
  [21][0] ) );
EDFQD1BWP7T \ram_reg[20][3]  ( .D(di[3]), .E(n226), .CP(n2), .Q(\ram
  [20][3] ) );
EDFQD1BWP7T \ram_reg[20][2]  ( .D(di[2]), .E(n226), .CP(n2), .Q(\ram
  [20][2] ) );
EDFQD1BWP7T \ram_reg[20][1]  ( .D(di[1]), .E(n226), .CP(n2), .Q(\ram
  [20][1] ) );
EDFQD1BWP7T \ram_reg[20][0]  ( .D(di[0]), .E(n226), .CP(n2), .Q(\ram
  [20][0] ) );
EDFQD1BWP7T \ram_reg[19][3]  ( .D(di[3]), .E(n227), .CP(n2), .Q(\ram
  [19][3] ) );
EDFQD1BWP7T \ram_reg[19][2]  ( .D(di[2]), .E(n227), .CP(n2), .Q(\ram
  [19][2] ) );
EDFQD1BWP7T \ram_reg[19][1]  ( .D(di[1]), .E(n227), .CP(n2), .Q(\ram
  [19][1] ) );
EDFQD1BWP7T \ram_reg[19][0]  ( .D(di[0]), .E(n227), .CP(n2), .Q(\ram
  [19][0] ) );
EDFQD1BWP7T \ram_reg[18][3]  ( .D(di[3]), .E(n228), .CP(n2), .Q(\ram
  [18][3] ) );
EDFQD1BWP7T \ram_reg[18][2]  ( .D(di[2]), .E(n228), .CP(n2), .Q(\ram
  [18][2] ) );
EDFQD1BWP7T \ram_reg[18][1]  ( .D(di[1]), .E(n228), .CP(n2), .Q(\ram
  [18][1] ) );
EDFQD1BWP7T \ram_reg[18][0]  ( .D(di[0]), .E(n228), .CP(n2), .Q(\ram
  [18][0] ) );
EDFQD1BWP7T \ram_reg[17][3]  ( .D(di[3]), .E(n229), .CP(n2), .Q(\ram
  [17][3] ) );
EDFQD1BWP7T \ram_reg[17][2]  ( .D(di[2]), .E(n229), .CP(n1), .Q(\ram
  [17][2] ) );
EDFQD1BWP7T \ram_reg[17][1]  ( .D(di[1]), .E(n229), .CP(clk), .Q(
      \ram[17][1] ) );
EDFQD1BWP7T \ram_reg[17][0]  ( .D(di[0]), .E(n229), .CP(n1), .Q(\ram
  [17][0] ) );
EDFQD1BWP7T \ram_reg[16][3]  ( .D(di[3]), .E(n230), .CP(clk), .Q(
      \ram[16][3] ) );
EDFQD1BWP7T \ram_reg[16][2]  ( .D(di[2]), .E(n230), .CP(n1), .Q(\ram
  [16][2] ) );
EDFQD1BWP7T \ram_reg[16][1]  ( .D(di[1]), .E(n230), .CP(n2), .Q(\ram
  [16][1] ) );
EDFQD1BWP7T \ram_reg[16][0]  ( .D(di[0]), .E(n230), .CP(n1), .Q(\ram
  [16][0] ) );
EDFQD1BWP7T \ram_reg[15][3]  ( .D(di[3]), .E(n231), .CP(n2), .Q(\ram
  [15][3] ) );
EDFQD1BWP7T \ram_reg[15][2]  ( .D(di[2]), .E(n231), .CP(clk), .Q(
      \ram[15][2] ) );
EDFQD1BWP7T \ram_reg[15][1]  ( .D(di[1]), .E(n231), .CP(n2), .Q(\ram
  [15][1] ) );
EDFQD1BWP7T \ram_reg[15][0]  ( .D(di[0]), .E(n231), .CP(n1), .Q(\ram
  [15][0] ) );
EDFQD1BWP7T \ram_reg[14][3]  ( .D(di[3]), .E(n232), .CP(n2), .Q(\ram
  [14][3] ) );
EDFQD1BWP7T \ram_reg[14][2]  ( .D(di[2]), .E(n232), .CP(n1), .Q(\ram
  [14][2] ) );
EDFQD1BWP7T \ram_reg[14][1]  ( .D(di[1]), .E(n232), .CP(clk), .Q(
      \ram[14][1] ) );
EDFQD1BWP7T \ram_reg[14][0]  ( .D(di[0]), .E(n232), .CP(n2), .Q(\ram
  [14][0] ) );
```

```
131  EDFQD1BWP7T \ram_reg[13][3]  ( .D(di[3]), .E(n233), .CP(clk), .Q(
132        \ram[13][3] ) );
133  EDFQD1BWP7T \ram_reg[13][2]  ( .D(di[2]), .E(n233), .CP(clk), .Q(
134        \ram[13][2] ) );
135  EDFQD1BWP7T \ram_reg[13][1]  ( .D(di[1]), .E(n233), .CP(n2), .Q(\ram
       [13][1] ) );
136  EDFQD1BWP7T \ram_reg[13][0]  ( .D(di[0]), .E(n233), .CP(n1), .Q(\ram
       [13][0] ) );
137  EDFQD1BWP7T \ram_reg[12][3]  ( .D(di[3]), .E(n234), .CP(n2), .Q(\ram
       [12][3] ) );
138  EDFQD1BWP7T \ram_reg[12][2]  ( .D(di[2]), .E(n234), .CP(n1), .Q(\ram
       [12][2] ) );
139  EDFQD1BWP7T \ram_reg[12][1]  ( .D(di[1]), .E(n234), .CP(n1), .Q(\ram
       [12][1] ) );
140  EDFQD1BWP7T \ram_reg[12][0]  ( .D(di[0]), .E(n234), .CP(n1), .Q(\ram
       [12][0] ) );
141  EDFQD1BWP7T \ram_reg[11][3]  ( .D(di[3]), .E(n235), .CP(n1), .Q(\ram
       [11][3] ) );
142  EDFQD1BWP7T \ram_reg[11][2]  ( .D(di[2]), .E(n235), .CP(clk), .Q(
143        \ram[11][2] ) );
144  EDFQD1BWP7T \ram_reg[11][1]  ( .D(di[1]), .E(n235), .CP(n2), .Q(\ram
       [11][1] ) );
145  EDFQD1BWP7T \ram_reg[11][0]  ( .D(di[0]), .E(n235), .CP(n1), .Q(\ram
       [11][0] ) );
146  EDFQD1BWP7T \ram_reg[10][3]  ( .D(di[3]), .E(n236), .CP(n2), .Q(\ram
       [10][3] ) );
147  EDFQD1BWP7T \ram_reg[10][2]  ( .D(di[2]), .E(n236), .CP(n1), .Q(\ram
       [10][2] ) );
148  EDFQD1BWP7T \ram_reg[10][1]  ( .D(di[1]), .E(n236), .CP(n1), .Q(\ram
       [10][1] ) );
149  EDFQD1BWP7T \ram_reg[10][0]  ( .D(di[0]), .E(n236), .CP(clk), .Q(
150        \ram[10][0] ) );
151  EDFQD1BWP7T \ram_reg[9][3]  ( .D(di[3]), .E(n237), .CP(n2), .Q(\ram[9][3]
       )
152        );
153  EDFQD1BWP7T \ram_reg[9][2]  ( .D(di[2]), .E(n237), .CP(n1), .Q(\ram[9][2]
       )
154        );
155  EDFQD1BWP7T \ram_reg[9][1]  ( .D(di[1]), .E(n237), .CP(n1), .Q(\ram[9][1]
       )
156        );
157  EDFQD1BWP7T \ram_reg[9][0]  ( .D(di[0]), .E(n237), .CP(n2), .Q(\ram[9][0]
       )
158        );
159  EDFQD1BWP7T \ram_reg[8][3]  ( .D(di[3]), .E(n238), .CP(clk), .Q(\ram[8][3]
        )
160        );
161  EDFQD1BWP7T \ram_reg[8][2]  ( .D(di[2]), .E(n238), .CP(n1), .Q(\ram[8][2]
       )
162        );
163  EDFQD1BWP7T \ram_reg[8][1]  ( .D(di[1]), .E(n238), .CP(n2), .Q(\ram[8][1]
       )
164        );
165  EDFQD1BWP7T \ram_reg[8][0]  ( .D(di[0]), .E(n238), .CP(n1), .Q(\ram[8][0]
       )
166        );
167  EDFQD1BWP7T \ram_reg[7][3]  ( .D(di[3]), .E(n239), .CP(n1), .Q(\ram[7][3]
       )
168        );
```

```
169   EDFQD1BWP7T \ram_reg[7][2]  ( .D(di[2]), .E(n239), .CP(clk), .Q(\ram[7][2]
        )
170         );
171   EDFQD1BWP7T \ram_reg[7][1]  ( .D(di[1]), .E(n239), .CP(n2), .Q(\ram[7][1]
        )
172         );
173   EDFQD1BWP7T \ram_reg[7][0]  ( .D(di[0]), .E(n239), .CP(n1), .Q(\ram[7][0]
        )
174         );
175   EDFQD1BWP7T \ram_reg[6][3]  ( .D(di[3]), .E(n240), .CP(n2), .Q(\ram[6][3]
        )
176         );
177   EDFQD1BWP7T \ram_reg[6][2]  ( .D(di[2]), .E(n240), .CP(n1), .Q(\ram[6][2]
        )
178         );
179   EDFQD1BWP7T \ram_reg[6][1]  ( .D(di[1]), .E(n240), .CP(n2), .Q(\ram[6][1]
        )
180         );
181   EDFQD1BWP7T \ram_reg[6][0]  ( .D(di[0]), .E(n240), .CP(n1), .Q(\ram[6][0]
        )
182         );
183   EDFQD1BWP7T \ram_reg[5][3]  ( .D(di[3]), .E(n241), .CP(n2), .Q(\ram[5][3]
        )
184         );
185   EDFQD1BWP7T \ram_reg[5][2]  ( .D(di[2]), .E(n241), .CP(clk), .Q(\ram[5][2]
        )
186         );
187   EDFQD1BWP7T \ram_reg[5][1]  ( .D(di[1]), .E(n241), .CP(n1), .Q(\ram[5][1]
        )
188         );
189   EDFQD1BWP7T \ram_reg[5][0]  ( .D(di[0]), .E(n241), .CP(n1), .Q(\ram[5][0]
        )
190         );
191   EDFQD1BWP7T \ram_reg[4][3]  ( .D(di[3]), .E(n242), .CP(n2), .Q(\ram[4][3]
        )
192         );
193   EDFQD1BWP7T \ram_reg[4][2]  ( .D(di[2]), .E(n242), .CP(clk), .Q(\ram[4][2]
        )
194         );
195   EDFQD1BWP7T \ram_reg[4][1]  ( .D(di[1]), .E(n242), .CP(n1), .Q(\ram[4][1]
        )
196         );
197   EDFQD1BWP7T \ram_reg[4][0]  ( .D(di[0]), .E(n242), .CP(n2), .Q(\ram[4][0]
        )
198         );
199   EDFQD1BWP7T \ram_reg[3][3]  ( .D(di[3]), .E(n243), .CP(clk), .Q(\ram[3][3]
        )
200         );
201   EDFQD1BWP7T \ram_reg[3][2]  ( .D(di[2]), .E(n243), .CP(n1), .Q(\ram[3][2]
        )
202         );
203   EDFQD1BWP7T \ram_reg[3][1]  ( .D(di[1]), .E(n243), .CP(n1), .Q(\ram[3][1]
        )
204         );
205   EDFQD1BWP7T \ram_reg[3][0]  ( .D(di[0]), .E(n243), .CP(n1), .Q(\ram[3][0]
        )
206         );
207   EDFQD1BWP7T \ram_reg[2][3]  ( .D(di[3]), .E(n244), .CP(n2), .Q(\ram[2][3]
        )
```

```
208         );
209   EDFQD1BWP7T \ram_reg[2][2]  ( .D(di[2]), .E(n244), .CP(n2), .Q(\ram[2][2]
        )
210         );
211   EDFQD1BWP7T \ram_reg[2][1]  ( .D(di[1]), .E(n244), .CP(n2), .Q(\ram[2][1]
        )
212         );
213   EDFQD1BWP7T \ram_reg[2][0]  ( .D(di[0]), .E(n244), .CP(clk), .Q(\ram[2][0]
        )
214         );
215   EDFQD1BWP7T \ram_reg[1][3]  ( .D(di[3]), .E(n245), .CP(clk), .Q(\ram[1][3]
        )
216         );
217   EDFQD1BWP7T \ram_reg[1][2]  ( .D(di[2]), .E(n245), .CP(n1), .Q(\ram[1][2]
        )
218         );
219   EDFQD1BWP7T \ram_reg[1][1]  ( .D(di[1]), .E(n245), .CP(n1), .Q(\ram[1][1]
        )
220         );
221   EDFQD1BWP7T \ram_reg[1][0]  ( .D(di[0]), .E(n245), .CP(n1), .Q(\ram[1][0]
        )
222         );
223   EDFQD1BWP7T \ram_reg[0][3]  ( .D(di[3]), .E(n246), .CP(clk), .Q(\ram[0][3]
        )
224         );
225   EDFQD1BWP7T \ram_reg[0][2]  ( .D(di[2]), .E(n246), .CP(n2), .Q(\ram[0][2]
        )
226         );
227   EDFQD1BWP7T \ram_reg[0][1]  ( .D(di[1]), .E(n246), .CP(n1), .Q(\ram[0][1]
        )
228         );
229   EDFQD1BWP7T \ram_reg[0][0]  ( .D(di[0]), .E(n246), .CP(clk), .Q(\ram[0][0]
        )
230         );
231   NR2D1BWP7T U1 ( .A1(n134), .A2(N11), .ZN(n32) );
232   INVD1BWP7T U2 ( .I(N10), .ZN(n134) );
233   NR2D1BWP7T U3 ( .A1(N11), .A2(N10), .ZN(n31) );
234   CKBD1BWP7T U4 ( .I(n1), .Z(n2) );
235   CKBD1BWP7T U5 ( .I(clk), .Z(n1) );
236   NR2D0BWP7T U6 ( .A1(n3), .A2(n4), .ZN(n246) );
237   NR2D0BWP7T U7 ( .A1(n3), .A2(n5), .ZN(n245) );
238   NR2D0BWP7T U8 ( .A1(n3), .A2(n6), .ZN(n244) );
239   NR2D0BWP7T U9 ( .A1(n3), .A2(n7), .ZN(n243) );
240   NR2D0BWP7T U10 ( .A1(n3), .A2(n8), .ZN(n242) );
241   NR2D0BWP7T U11 ( .A1(n3), .A2(n9), .ZN(n241) );
242   NR2D0BWP7T U12 ( .A1(n3), .A2(n10), .ZN(n240) );
243   NR2D0BWP7T U13 ( .A1(n3), .A2(n11), .ZN(n239) );
244   ND3D0BWP7T U14 ( .A1(n12), .A2(n13), .A3(we), .ZN(n3) );
245   NR2D0BWP7T U15 ( .A1(n4), .A2(n14), .ZN(n238) );
246   NR2D0BWP7T U16 ( .A1(n5), .A2(n14), .ZN(n237) );
247   NR2D0BWP7T U17 ( .A1(n6), .A2(n14), .ZN(n236) );
248   NR2D0BWP7T U18 ( .A1(n7), .A2(n14), .ZN(n235) );
249   NR2D0BWP7T U19 ( .A1(n8), .A2(n14), .ZN(n234) );
250   NR2D0BWP7T U20 ( .A1(n9), .A2(n14), .ZN(n233) );
251   NR2D0BWP7T U21 ( .A1(n10), .A2(n14), .ZN(n232) );
252   NR2D0BWP7T U22 ( .A1(n11), .A2(n14), .ZN(n231) );
253   ND3D0BWP7T U23 ( .A1(we), .A2(n13), .A3(a[3]), .ZN(n14) );
254   CKND0BWP7T U24 ( .I(a[4]), .ZN(n13) );
255   NR2D0BWP7T U25 ( .A1(n4), .A2(n15), .ZN(n230) );
```

```
256   NR2D0BWP7T U26 ( .A1(n5), .A2(n15), .ZN(n229) );
257   NR2D0BWP7T U27 ( .A1(n6), .A2(n15), .ZN(n228) );
258   NR2D0BWP7T U28 ( .A1(n7), .A2(n15), .ZN(n227) );
259   NR2D0BWP7T U29 ( .A1(n8), .A2(n15), .ZN(n226) );
260   NR2D0BWP7T U30 ( .A1(n9), .A2(n15), .ZN(n225) );
261   NR2D0BWP7T U31 ( .A1(n10), .A2(n15), .ZN(n224) );
262   NR2D0BWP7T U32 ( .A1(n11), .A2(n15), .ZN(n223) );
263   ND3D0BWP7T U33 ( .A1(we), .A2(n12), .A3(a[4]), .ZN(n15) );
264   CKND0BWP7T U34 ( .I(a[3]), .ZN(n12) );
265   NR2D0BWP7T U35 ( .A1(n4), .A2(n16), .ZN(n222) );
266   ND3D0BWP7T U36 ( .A1(n17), .A2(n18), .A3(n19), .ZN(n4) );
267   NR2D0BWP7T U37 ( .A1(n5), .A2(n16), .ZN(n221) );
268   ND3D0BWP7T U38 ( .A1(n17), .A2(n18), .A3(a[0]), .ZN(n5) );
269   NR2D0BWP7T U39 ( .A1(n6), .A2(n16), .ZN(n220) );
270   ND3D0BWP7T U40 ( .A1(n19), .A2(n18), .A3(a[1]), .ZN(n6) );
271   NR2D0BWP7T U41 ( .A1(n7), .A2(n16), .ZN(n219) );
272   ND3D0BWP7T U42 ( .A1(a[0]), .A2(n18), .A3(a[1]), .ZN(n7) );
273   CKND0BWP7T U43 ( .I(a[2]), .ZN(n18) );
274   NR2D0BWP7T U44 ( .A1(n8), .A2(n16), .ZN(n218) );
275   ND3D0BWP7T U45 ( .A1(n19), .A2(n17), .A3(a[2]), .ZN(n8) );
276   NR2D0BWP7T U46 ( .A1(n9), .A2(n16), .ZN(n217) );
277   ND3D0BWP7T U47 ( .A1(a[0]), .A2(n17), .A3(a[2]), .ZN(n9) );
278   CKND0BWP7T U48 ( .I(a[1]), .ZN(n17) );
279   NR2D0BWP7T U49 ( .A1(n10), .A2(n16), .ZN(n216) );
280   ND3D0BWP7T U50 ( .A1(a[1]), .A2(n19), .A3(a[2]), .ZN(n10) );
281   CKND0BWP7T U51 ( .I(a[0]), .ZN(n19) );
282   NR2D0BWP7T U52 ( .A1(n11), .A2(n16), .ZN(n215) );
283   ND3D0BWP7T U53 ( .A1(a[3]), .A2(we), .A3(a[4]), .ZN(n16) );
284   ND3D0BWP7T U54 ( .A1(a[1]), .A2(a[0]), .A3(a[2]), .ZN(n11) );
285   MUX2ND0BWP7T U55 ( .I0(n20), .I1(n21), .S(N14), .ZN(do[3]) );
286   NR4D0BWP7T U56 ( .A1(n22), .A2(n23), .A3(n24), .A4(n25), .ZN(n21) );
287   AOI21D0BWP7T U57 ( .A1(n26), .A2(n27), .B(n28), .ZN(n25) );
288   AOI22D0BWP7T U58 ( .A1(\ram[22][3] ), .A2(n29), .B1(\ram[23][3] ), .B2(n30
        ),
289        .ZN(n27) );
290   AOI22D0BWP7T U59 ( .A1(\ram[20][3] ), .A2(n31), .B1(\ram[21][3] ), .B2(n32
        ),
291        .ZN(n26) );
292   AOI21D0BWP7T U60 ( .A1(n33), .A2(n34), .B(n35), .ZN(n24) );
293   AOI22D0BWP7T U61 ( .A1(\ram[18][3] ), .A2(n29), .B1(\ram[19][3] ), .B2(n30
        ),
294        .ZN(n34) );
295   AOI22D0BWP7T U62 ( .A1(\ram[16][3] ), .A2(n31), .B1(\ram[17][3] ), .B2(n32
        ),
296        .ZN(n33) );
297   AOI21D0BWP7T U63 ( .A1(n36), .A2(n37), .B(n38), .ZN(n23) );
298   AOI22D0BWP7T U64 ( .A1(\ram[30][3] ), .A2(n29), .B1(\ram[31][3] ), .B2(n30
        ),
299        .ZN(n37) );
300   AOI22D0BWP7T U65 ( .A1(\ram[28][3] ), .A2(n31), .B1(\ram[29][3] ), .B2(n32
        ),
301        .ZN(n36) );
302   AOI21D0BWP7T U66 ( .A1(n39), .A2(n40), .B(n41), .ZN(n22) );
303   AOI22D0BWP7T U67 ( .A1(\ram[26][3] ), .A2(n29), .B1(\ram[27][3] ), .B2(n30
        ),
304        .ZN(n40) );
305   AOI22D0BWP7T U68 ( .A1(\ram[24][3] ), .A2(n31), .B1(\ram[25][3] ), .B2(n32
        ),
306        .ZN(n39) );
```

```
307  NR4D0BWP7T U69 ( .A1(n42), .A2(n43), .A3(n44), .A4(n45), .ZN(n20) );
308  AOI21D0BWP7T U70 ( .A1(n46), .A2(n47), .B(n28), .ZN(n45) );
309  AOI22D0BWP7T U71 ( .A1(\ram[6][3] ), .A2(n29), .B1(\ram[7][3] ), .B2(n30),
310        .ZN(n47) );
311  AOI22D0BWP7T U72 ( .A1(\ram[4][3] ), .A2(n31), .B1(\ram[5][3] ), .B2(n32),
312        .ZN(n46) );
313  AOI21D0BWP7T U73 ( .A1(n48), .A2(n49), .B(n35), .ZN(n44) );
314  AOI22D0BWP7T U74 ( .A1(\ram[2][3] ), .A2(n29), .B1(\ram[3][3] ), .B2(n30),
315        .ZN(n49) );
316  AOI22D0BWP7T U75 ( .A1(\ram[0][3] ), .A2(n31), .B1(\ram[1][3] ), .B2(n32),
317        .ZN(n48) );
318  AOI21D0BWP7T U76 ( .A1(n50), .A2(n51), .B(n38), .ZN(n43) );
319  AOI22D0BWP7T U77 ( .A1(\ram[14][3] ), .A2(n29), .B1(\ram[15][3] ), .B2(n30
       ),
320        .ZN(n51) );
321  AOI22D0BWP7T U78 ( .A1(\ram[12][3] ), .A2(n31), .B1(\ram[13][3] ), .B2(n32
       ),
322        .ZN(n50) );
323  AOI21D0BWP7T U79 ( .A1(n52), .A2(n53), .B(n41), .ZN(n42) );
324  AOI22D0BWP7T U80 ( .A1(\ram[10][3] ), .A2(n29), .B1(\ram[11][3] ), .B2(n30
       ),
325        .ZN(n53) );
326  AOI22D0BWP7T U81 ( .A1(\ram[8][3] ), .A2(n31), .B1(\ram[9][3] ), .B2(n32),
327        .ZN(n52) );
328  MUX2ND0BWP7T U82 ( .I0(n54), .I1(n55), .S(N14), .ZN(do[2]) );
329  NR4D0BWP7T U83 ( .A1(n56), .A2(n57), .A3(n58), .A4(n59), .ZN(n55) );
330  AOI21D0BWP7T U84 ( .A1(n60), .A2(n61), .B(n28), .ZN(n59) );
331  AOI22D0BWP7T U85 ( .A1(\ram[22][2] ), .A2(n29), .B1(\ram[23][2] ), .B2(n30
       ),
332        .ZN(n61) );
333  AOI22D0BWP7T U86 ( .A1(\ram[20][2] ), .A2(n31), .B1(\ram[21][2] ), .B2(n32
       ),
334        .ZN(n60) );
335  AOI21D0BWP7T U87 ( .A1(n62), .A2(n63), .B(n35), .ZN(n58) );
336  AOI22D0BWP7T U88 ( .A1(\ram[18][2] ), .A2(n29), .B1(\ram[19][2] ), .B2(n30
       ),
337        .ZN(n63) );
338  AOI22D0BWP7T U89 ( .A1(\ram[16][2] ), .A2(n31), .B1(\ram[17][2] ), .B2(n32
       ),
339        .ZN(n62) );
340  AOI21D0BWP7T U90 ( .A1(n64), .A2(n65), .B(n38), .ZN(n57) );
341  AOI22D0BWP7T U91 ( .A1(\ram[30][2] ), .A2(n29), .B1(\ram[31][2] ), .B2(n30
       ),
342        .ZN(n65) );
343  AOI22D0BWP7T U92 ( .A1(\ram[28][2] ), .A2(n31), .B1(\ram[29][2] ), .B2(n32
       ),
344        .ZN(n64) );
345  AOI21D0BWP7T U93 ( .A1(n66), .A2(n67), .B(n41), .ZN(n56) );
346  AOI22D0BWP7T U94 ( .A1(\ram[26][2] ), .A2(n29), .B1(\ram[27][2] ), .B2(n30
       ),
347        .ZN(n67) );
348  AOI22D0BWP7T U95 ( .A1(\ram[24][2] ), .A2(n31), .B1(\ram[25][2] ), .B2(n32
       ),
349        .ZN(n66) );
350  NR4D0BWP7T U96 ( .A1(n68), .A2(n69), .A3(n70), .A4(n71), .ZN(n54) );
351  AOI21D0BWP7T U97 ( .A1(n72), .A2(n73), .B(n28), .ZN(n71) );
352  AOI22D0BWP7T U98 ( .A1(\ram[6][2] ), .A2(n29), .B1(\ram[7][2] ), .B2(n30),
353        .ZN(n73) );
354  AOI22D0BWP7T U99 ( .A1(\ram[4][2] ), .A2(n31), .B1(\ram[5][2] ), .B2(n32),
```

```
355         .ZN(n72) );
356   AOI21D0BWP7T U100 ( .A1(n74), .A2(n75), .B(n35), .ZN(n70) );
357   AOI22D0BWP7T U101 ( .A1(\ram[2][2] ), .A2(n29), .B1(\ram[3][2] ), .B2(n30)
        ,
358         .ZN(n75) );
359   AOI22D0BWP7T U102 ( .A1(\ram[0][2] ), .A2(n31), .B1(\ram[1][2] ), .B2(n32)
        ,
360         .ZN(n74) );
361   AOI21D0BWP7T U103 ( .A1(n76), .A2(n77), .B(n38), .ZN(n69) );
362   AOI22D0BWP7T U104 ( .A1(\ram[14][2] ), .A2(n29), .B1(\ram[15][2] ), .B2(
        n30),
363         .ZN(n77) );
364   AOI22D0BWP7T U105 ( .A1(\ram[12][2] ), .A2(n31), .B1(\ram[13][2] ), .B2(
        n32),
365         .ZN(n76) );
366   AOI21D0BWP7T U106 ( .A1(n78), .A2(n79), .B(n41), .ZN(n68) );
367   AOI22D0BWP7T U107 ( .A1(\ram[10][2] ), .A2(n29), .B1(\ram[11][2] ), .B2(
        n30),
368         .ZN(n79) );
369   AOI22D0BWP7T U108 ( .A1(\ram[8][2] ), .A2(n31), .B1(\ram[9][2] ), .B2(n32)
        ,
370         .ZN(n78) );
371   MUX2ND0BWP7T U109 ( .I0(n80), .I1(n81), .S(N14), .ZN(do[1]) );
372   NR4D0BWP7T U110 ( .A1(n82), .A2(n83), .A3(n84), .A4(n85), .ZN(n81) );
373   AOI21D0BWP7T U111 ( .A1(n86), .A2(n87), .B(n28), .ZN(n85) );
374   AOI22D0BWP7T U112 ( .A1(\ram[22][1] ), .A2(n29), .B1(\ram[23][1] ), .B2(
        n30),
375         .ZN(n87) );
376   AOI22D0BWP7T U113 ( .A1(\ram[20][1] ), .A2(n31), .B1(\ram[21][1] ), .B2(
        n32),
377         .ZN(n86) );
378   AOI21D0BWP7T U114 ( .A1(n88), .A2(n89), .B(n35), .ZN(n84) );
379   AOI22D0BWP7T U115 ( .A1(\ram[18][1] ), .A2(n29), .B1(\ram[19][1] ), .B2(
        n30),
380         .ZN(n89) );
381   AOI22D0BWP7T U116 ( .A1(\ram[16][1] ), .A2(n31), .B1(\ram[17][1] ), .B2(
        n32),
382         .ZN(n88) );
383   AOI21D0BWP7T U117 ( .A1(n90), .A2(n91), .B(n38), .ZN(n83) );
384   AOI22D0BWP7T U118 ( .A1(\ram[30][1] ), .A2(n29), .B1(\ram[31][1] ), .B2(
        n30),
385         .ZN(n91) );
386   AOI22D0BWP7T U119 ( .A1(\ram[28][1] ), .A2(n31), .B1(\ram[29][1] ), .B2(
        n32),
387         .ZN(n90) );
388   AOI21D0BWP7T U120 ( .A1(n92), .A2(n93), .B(n41), .ZN(n82) );
389   AOI22D0BWP7T U121 ( .A1(\ram[26][1] ), .A2(n29), .B1(\ram[27][1] ), .B2(
        n30),
390         .ZN(n93) );
391   AOI22D0BWP7T U122 ( .A1(\ram[24][1] ), .A2(n31), .B1(\ram[25][1] ), .B2(
        n32),
392         .ZN(n92) );
393   NR4D0BWP7T U123 ( .A1(n94), .A2(n95), .A3(n96), .A4(n97), .ZN(n80) );
394   AOI21D0BWP7T U124 ( .A1(n98), .A2(n99), .B(n28), .ZN(n97) );
395   AOI22D0BWP7T U125 ( .A1(\ram[6][1] ), .A2(n29), .B1(\ram[7][1] ), .B2(n30)
        ,
396         .ZN(n99) );
397   AOI22D0BWP7T U126 ( .A1(\ram[4][1] ), .A2(n31), .B1(\ram[5][1] ), .B2(n32)
        ,
```

```
398         .ZN(n98) );
399   AOI21D0BWP7T U127 ( .A1(n100), .A2(n101), .B(n35), .ZN(n96) );
400   AOI22D0BWP7T U128 ( .A1(\ram[2][1] ), .A2(n29), .B1(\ram[3][1] ), .B2(n30)
        ,
401         .ZN(n101) );
402   AOI22D0BWP7T U129 ( .A1(\ram[0][1] ), .A2(n31), .B1(\ram[1][1] ), .B2(n32)
        ,
403         .ZN(n100) );
404   AOI21D0BWP7T U130 ( .A1(n102), .A2(n103), .B(n38), .ZN(n95) );
405   AOI22D0BWP7T U131 ( .A1(\ram[14][1] ), .A2(n29), .B1(\ram[15][1] ), .B2(
        n30),
406         .ZN(n103) );
407   AOI22D0BWP7T U132 ( .A1(\ram[12][1] ), .A2(n31), .B1(\ram[13][1] ), .B2(
        n32),
408         .ZN(n102) );
409   AOI21D0BWP7T U133 ( .A1(n104), .A2(n105), .B(n41), .ZN(n94) );
410   AOI22D0BWP7T U134 ( .A1(\ram[10][1] ), .A2(n29), .B1(\ram[11][1] ), .B2(
        n30),
411         .ZN(n105) );
412   AOI22D0BWP7T U135 ( .A1(\ram[8][1] ), .A2(n31), .B1(\ram[9][1] ), .B2(n32)
        ,
413         .ZN(n104) );
414   MUX2ND0BWP7T U136 ( .I0(n106), .I1(n107), .S(N14), .ZN(do[0]) );
415   NR4D0BWP7T U137 ( .A1(n108), .A2(n109), .A3(n110), .A4(n111), .ZN(n107) );
416   AOI21D0BWP7T U138 ( .A1(n112), .A2(n113), .B(n28), .ZN(n111) );
417   AOI22D0BWP7T U139 ( .A1(\ram[22][0] ), .A2(n29), .B1(\ram[23][0] ), .B2(
        n30),
418         .ZN(n113) );
419   AOI22D0BWP7T U140 ( .A1(\ram[20][0] ), .A2(n31), .B1(\ram[21][0] ), .B2(
        n32),
420         .ZN(n112) );
421   AOI21D0BWP7T U141 ( .A1(n114), .A2(n115), .B(n35), .ZN(n110) );
422   AOI22D0BWP7T U142 ( .A1(\ram[18][0] ), .A2(n29), .B1(\ram[19][0] ), .B2(
        n30),
423         .ZN(n115) );
424   AOI22D0BWP7T U143 ( .A1(\ram[16][0] ), .A2(n31), .B1(\ram[17][0] ), .B2(
        n32),
425         .ZN(n114) );
426   AOI21D0BWP7T U144 ( .A1(n116), .A2(n117), .B(n38), .ZN(n109) );
427   AOI22D0BWP7T U145 ( .A1(\ram[30][0] ), .A2(n29), .B1(\ram[31][0] ), .B2(
        n30),
428         .ZN(n117) );
429   AOI22D0BWP7T U146 ( .A1(\ram[28][0] ), .A2(n31), .B1(\ram[29][0] ), .B2(
        n32),
430         .ZN(n116) );
431   AOI21D0BWP7T U147 ( .A1(n118), .A2(n119), .B(n41), .ZN(n108) );
432   AOI22D0BWP7T U148 ( .A1(\ram[26][0] ), .A2(n29), .B1(\ram[27][0] ), .B2(
        n30),
433         .ZN(n119) );
434   AOI22D0BWP7T U149 ( .A1(\ram[24][0] ), .A2(n31), .B1(\ram[25][0] ), .B2(
        n32),
435         .ZN(n118) );
436   NR4D0BWP7T U150 ( .A1(n120), .A2(n121), .A3(n122), .A4(n123), .ZN(n106) );
437   AOI21D0BWP7T U151 ( .A1(n124), .A2(n125), .B(n28), .ZN(n123) );
438   CKND2D0BWP7T U152 ( .A1(N12), .A2(n126), .ZN(n28) );
439   AOI22D0BWP7T U153 ( .A1(\ram[6][0] ), .A2(n29), .B1(\ram[7][0] ), .B2(n30)
        ,
440         .ZN(n125) );
441   AOI22D0BWP7T U154 ( .A1(\ram[4][0] ), .A2(n31), .B1(\ram[5][0] ), .B2(n32)
```

```
    ,
442         .ZN(n124) );
443   AOI21D0BWP7T U155 ( .A1(n127), .A2(n128), .B(n35), .ZN(n122) );
444   CKND2D0BWP7T U156 ( .A1(n129), .A2(n126), .ZN(n35) );
445   CKND0BWP7T U157 ( .I(N13), .ZN(n126) );
446   AOI22D0BWP7T U158 ( .A1(\ram[2][0] ), .A2(n29), .B1(\ram[3][0] ), .B2(n30)
    ,
447         .ZN(n128) );
448   AOI22D0BWP7T U159 ( .A1(\ram[0][0] ), .A2(n31), .B1(\ram[1][0] ), .B2(n32)
    ,
449         .ZN(n127) );
450   AOI21D0BWP7T U160 ( .A1(n130), .A2(n131), .B(n38), .ZN(n121) );
451   CKND2D0BWP7T U161 ( .A1(N12), .A2(N13), .ZN(n38) );
452   AOI22D0BWP7T U162 ( .A1(\ram[14][0] ), .A2(n29), .B1(\ram[15][0] ), .B2(
    n30),
453         .ZN(n131) );
454   AOI22D0BWP7T U163 ( .A1(\ram[12][0] ), .A2(n31), .B1(\ram[13][0] ), .B2(
    n32),
455         .ZN(n130) );
456   AOI21D0BWP7T U164 ( .A1(n132), .A2(n133), .B(n41), .ZN(n120) );
457   CKND2D0BWP7T U165 ( .A1(N13), .A2(n129), .ZN(n41) );
458   CKND0BWP7T U166 ( .I(N12), .ZN(n129) );
459   AOI22D0BWP7T U167 ( .A1(\ram[10][0] ), .A2(n29), .B1(\ram[11][0] ), .B2(
    n30),
460         .ZN(n133) );
461   AN2D0BWP7T U168 ( .A1(N11), .A2(N10), .Z(n30) );
462   AN2D0BWP7T U169 ( .A1(N11), .A2(n134), .Z(n29) );
463   AOI22D0BWP7T U170 ( .A1(\ram[8][0] ), .A2(n31), .B1(\ram[9][0] ), .B2(n32)
    ,
464         .ZN(n132) );
465 endmodule
466
467
468 module ram_IO ( clk, we, a, di, do );
469   input [4:0] a;
470   input [3:0] di;
471   output [3:0] do;
472   input clk, we;
473   wire   clk_w, we_w, n1, n2, n3, n4, n5, n6, n7, n8, n9, n10;
474   wire   [4:0] a_w;
475   wire   [3:0] di_w;
476   wire   [3:0] do_w;
477   tri   clk;
478   tri   we;
479   tri   [4:0] a;
480   tri   [3:0] di;
481   tri   [3:0] do;
482
483   raminfr \cell  ( .clk(clk_w), .we(we_w), .a(a_w), .di({n4, n6, n8, n10}),
484         .do(do_w) );
485   PDDW0204SCDG IN00 ( .I(n1), .OEN(n2), .IE(n2), .PAD(a[0]), .DS(n1), .PE(n1
    ),
486         .C(a_w[0]) );
487   PDDW0204SCDG IN01 ( .I(n1), .OEN(n2), .IE(n2), .PAD(a[1]), .DS(n1), .PE(n1
    ),
488         .C(a_w[1]) );
489   PDDW0204SCDG IN02 ( .I(n1), .OEN(n2), .IE(n2), .PAD(a[2]), .DS(n1), .PE(n1
    ),
490         .C(a_w[2]) );
```

```
  PDDW0204SCDG IN03 ( .I(n1), .OEN(n2), .IE(n2), .PAD(a[3]), .DS(n1), .PE(n1
    ),
       .C(a_w[3]) );
  PDDW0204SCDG IN04 ( .I(n1), .OEN(n2), .IE(n2), .PAD(a[4]), .DS(n1), .PE(n1
    ),
       .C(a_w[4]) );
  PDDW0204SCDG IN05 ( .I(n1), .OEN(n2), .IE(n2), .PAD(di[0]), .DS(n1), .PE(
    n1),
       .C(di_w[0]) );
  PDDW0204SCDG IN06 ( .I(n1), .OEN(n2), .IE(n2), .PAD(di[1]), .DS(n1), .PE(
    n1),
       .C(di_w[1]) );
  PDDW0204SCDG IN07 ( .I(n1), .OEN(n2), .IE(n2), .PAD(di[2]), .DS(n1), .PE(
    n1),
       .C(di_w[2]) );
  PDDW0204SCDG IN08 ( .I(n1), .OEN(n2), .IE(n2), .PAD(di[3]), .DS(n1), .PE(
    n1),
       .C(di_w[3]) );
  PDDW0204SCDG IN09 ( .I(n1), .OEN(n2), .IE(n2), .PAD(clk), .DS(n1), .PE(n1)
    ,
       .C(clk_w) );
  PDDW0204SCDG IN10 ( .I(n1), .OEN(n2), .IE(n2), .PAD(we), .DS(n1), .PE(n1),
       .C(we_w) );
  PDDW0204SCDG OUT00 ( .I(do_w[0]), .OEN(n1), .IE(n1), .PAD(do[0]), .DS(n1),
       .PE(n1) );
  PDDW0204SCDG OUT01 ( .I(do_w[1]), .OEN(n1), .IE(n1), .PAD(do[1]), .DS(n1),
       .PE(n1) );
  PDDW0204SCDG OUT02 ( .I(do_w[2]), .OEN(n1), .IE(n1), .PAD(do[2]), .DS(n1),
       .PE(n1) );
  PDDW0204SCDG OUT03 ( .I(do_w[3]), .OEN(n1), .IE(n1), .PAD(do[3]), .DS(n1),
       .PE(n1) );
  INVD1BWP7T U6 ( .I(di_w[3]), .ZN(n3) );
  INVD1BWP7T U7 ( .I(n3), .ZN(n4) );
  INVD1BWP7T U8 ( .I(di_w[2]), .ZN(n5) );
  INVD1BWP7T U9 ( .I(n5), .ZN(n6) );
  INVD1BWP7T U10 ( .I(di_w[1]), .ZN(n7) );
  INVD1BWP7T U11 ( .I(n7), .ZN(n8) );
  INVD1BWP7T U12 ( .I(di_w[0]), .ZN(n9) );
  INVD1BWP7T U13 ( .I(n9), .ZN(n10) );
  TIELBWP7T U14 ( .ZN(n1) );
  TIEHBWP7T U15 ( .Z(n2) );
endmodule
```

13.4. Código Python para escritura del circuito Gran Jaguar

```
#Libreria para converion de texto a bits
import binascii

#definir los textos que se van a traducir
texto = "Hola, soy El Gran Jaguar, el primer nanochip elaborado en
    Centroamerica por una Universidad. Desarrollado completamente en la
    Universidad del Valle de Guatemala por alumnos graduandos de Ingenieria
    en Electronica entre 2019 y 2021"
texto2 = "El equipo encargado de mi creacion se conformo por: Geovanni
    Flores, Matthias Sibrian, Steven Rubio, Julio Shin, Karol Cardona, Luis
    Najera, Luis Abadia, Joel Gonzalez, Jose Ayala, Jose Ruiz, Ricardo Giron
```

```
    , Carlos Esquit, Charlie Ayenci, Elmer Torres, Gerardo Cardoza, Jonathan
     de los Santos, Antonio Altuna, Kurt Kellner y Luis Rivera"
7
8 #cantidad de caracteres para cada texto
9 print len(texto)
10 print len(texto2)
11
12 #crear el archivo tipo verilog
13 archivo = open("circuito.v", "w")
14
15 #escritura del codigo no repetitivo
16 archivo.writelines("module chip_SP(q_out, reset, clk, EN, clk_s, select);\n"
    )
17
18 archivo.writelines("output [7:0] q_out;\n")
19 archivo.writelines("input reset;\n")
20 archivo.writelines("input clk;\n")
21 archivo.writelines("input [1:0]select;\n")
22
23 archivo.writelines("reg [8:0] contador;\n")
24 archivo.writelines("reg [7:0] q;\n")
25
26 archivo.writelines("input EN;\n")
27 archivo.writelines("output clk_s;\n")
28
29 archivo.writelines("wire W_1;\n")
30 archivo.writelines("wire W_2;\n")
31 archivo.writelines("wire W_3;\n")
32 archivo.writelines("wire W_4;\n")
33 archivo.writelines("wire W_5;\n")
34 archivo.writelines("wire W_6;\n")
35 archivo.writelines("wire W_7;\n")
36 archivo.writelines("wire W_8;\n")
37 archivo.writelines("wire W_9;\n")
38 archivo.writelines("wire W_10;\n")
39 archivo.writelines("wire W_11;\n")
40 archivo.writelines("wire W_12;\n")
41 archivo.writelines("wire W_13;\n")
42 archivo.writelines("wire W_14;\n")
43 archivo.writelines("wire W_15;\n")
44 archivo.writelines("wire W_16;\n")
45 archivo.writelines("wire W_17;\n")
46 archivo.writelines("wire W_18;\n")
47 archivo.writelines("wire W_19;\n")
48 archivo.writelines("wire clk_G;\n")
49
50 archivo.writelines("assign clk_s = clk_G;\n")
51
52 archivo.writelines("AND2 U1(EN,clk_G,W_1);\n")
53 archivo.writelines("INV  U2(W_1,W_2);\n")
54 archivo.writelines("INV  U3(W_2,W_3);\n")
55 archivo.writelines("INV  U4(W_3,W_4);\n")
56 archivo.writelines("INV  U5(W_4,W_5);\n")
57 archivo.writelines("INV  U6(W_5,W_6);\n")
58 archivo.writelines("INV  U7(W_6,W_7);\n")
59 archivo.writelines("INV  U8(W_7,W_8);\n")
60 archivo.writelines("INV  U9(W_8,W_9);\n")
61 archivo.writelines("INV  U10(W_9,W_10);\n")
62 archivo.writelines("INV  U11(W_10,W_11);\n")
```

```
63 archivo.writelines("INV  U12(W_11,W_12);\n")
64 archivo.writelines("INV  U13(W_12,W_13);\n")
65 archivo.writelines("INV  U14(W_13,W_14);\n")
66 archivo.writelines("INV  U15(W_14,W_15);\n")
67 archivo.writelines("INV  U16(W_15,W_16);\n")
68 archivo.writelines("INV  U17(W_16,W_17);\n")
69 archivo.writelines("INV  U18(W_17,W_18);\n")
70 archivo.writelines("INV  U19(W_18,W_19);\n")
71 archivo.writelines("INV  U20(W_19,clk_G);\n")
72
73
74 archivo.writelines("always @ (posedge reset or posedge clk)\n")
75 archivo.writelines("if (reset)\n")
76 archivo.writelines("contador<=9'b00000000;\n")
77 archivo.writelines("else if(select==2'b00 || select==2'b01) begin\n")
78 archivo.writelines("if (contador <"+str(len(texto)-1)+ ")\n")
79 archivo.writelines("contador <= contador + 1;\n")
80 archivo.writelines("else\n")
81 archivo.writelines("contador <= 0;\n")
82 archivo.writelines("end\n")
83 archivo.writelines("else if(select==2'b10 || select==2'b11) begin\n")
84 archivo.writelines("if (contador <"+str(len(texto2)-1)+ ")\n")
85 archivo.writelines("contador <= contador + 1;\n")
86 archivo.writelines("else\n")
87 archivo.writelines("contador <= 0;\n")
88 archivo.writelines("end\n")
89
90
91 archivo.writelines("always @ (posedge clk)\n")
92 archivo.writelines("if(select==2'b00 || select==2'b01)\n")
93 archivo.writelines("begin\n")
94 archivo.writelines("if(contador == 'd0)\n")
95 archivo.writelines("begin\n")
96 archivo.writelines("q<=8'b01001000;\n")
97 archivo.writelines("end\n")
98
99 #Escritura del primer Texto
100 for i in range(len(texto)-1):
101   archivo.writelines("else if(contador == 'd"+str(i+1)+")\n")
102   archivo.writelines("begin\n")
103   archivo.writelines("q<=8'b")
104   binario = bin(int(binascii.hexlify(texto[i+1]),16))
105   for x in range(10-len(binario)):
106     archivo.writelines("0")
107   for j in range(len(binario)-2):
108     archivo.writelines(str(binario[j+2]))
109   archivo.writelines(";\n")
110   archivo.writelines("end\n")
111 archivo.writelines("end\n")
112
113 archivo.writelines("else if(select==2'b10 || select==2'b11)\n")
114 archivo.writelines("begin\n")
115 archivo.writelines("if(contador == 'd0)\n")
116 archivo.writelines("begin\n")
117 archivo.writelines("q<=8'b01000101;\n")
118 archivo.writelines("end\n")
119
120 #Escritura del segundo texto
121 for i in range(len(texto2)-1):
```

```
  archivo.writelines("else if(contador == 'd"+str(i+1)+")\n")
  archivo.writelines("begin\n")
  archivo.writelines("q<=8'b")
  binario = bin(int(binascii.hexlify(texto2[i+1]),16))
  for x in range(10-len(binario)):
    archivo.writelines("0")
  for j in range(len(binario)-2):
    archivo.writelines(str(binario[j+2]))
  archivo.writelines(";\n")
  archivo.writelines("end\n")
archivo.writelines("end\n")
archivo.writelines("assign q_out = q;\n")
archivo.writelines("endmodule\n")


#Definion de los modulos para el ring oscilator
archivo.writelines("module INV(A, B);\n")
archivo.writelines("input A;\n")
archivo.writelines("output B;\n")
archivo.writelines("assign B = ~A;\n")
archivo.writelines("endmodule\n")

archivo.writelines("module AND2(in1, in2, out);\n")
archivo.writelines("input in1, in2;\n")
archivo.writelines("output out;\n")
archivo.writelines("assign out = in1 & in2;\n")
archivo.writelines("endmodule\n")

#Guardar el archivo que escribimos
archivo.close()
```

13.5. *Netlist* final para la síntesis del circuito el Gran Jaguar

```
////////////////////////////////////////////////////////////
// Created by: Synopsys DC Expert(TM) in wire load mode
// Version   : R-2020.09-SP4
// Date      : Tue Nov  2 21:18:56 2021
////////////////////////////////////////////////////////////


module AND2 ( in1, in2, out );
  input in1, in2;
  output out;


  AN2XD1BWP7T U1 ( .A1(in2), .A2(in1), .Z(out) );
endmodule


module INV_18 ( A, B );
  input A;
  output B;


  INVD1BWP7T U1 ( .I(A), .ZN(B) );
endmodule
```

```
24
25
26 module INV_17 ( A, B );
27   input A;
28   output B;
29
30
31   INVD1BWP7T U1 ( .I(A), .ZN(B) );
32 endmodule
33
34
35 module INV_16 ( A, B );
36   input A;
37   output B;
38
39
40   INVD1BWP7T U1 ( .I(A), .ZN(B) );
41 endmodule
42
43
44 module INV_15 ( A, B );
45   input A;
46   output B;
47
48
49   INVD1BWP7T U1 ( .I(A), .ZN(B) );
50 endmodule
51
52
53 module INV_14 ( A, B );
54   input A;
55   output B;
56
57
58   INVD1BWP7T U1 ( .I(A), .ZN(B) );
59 endmodule
60
61
62 module INV_13 ( A, B );
63   input A;
64   output B;
65
66
67   INVD1BWP7T U1 ( .I(A), .ZN(B) );
68 endmodule
69
70
71 module INV_12 ( A, B );
72   input A;
73   output B;
74
75
76   INVD1BWP7T U1 ( .I(A), .ZN(B) );
77 endmodule
78
79
80 module INV_11 ( A, B );
81   input A;
82   output B;
```

```
83
84
85    INVD1BWP7T U1 ( .I(A), .ZN(B) );
86  endmodule
87
88
89  module INV_10 ( A, B );
90    input A;
91    output B;
92
93
94    INVD1BWP7T U1 ( .I(A), .ZN(B) );
95  endmodule
96
97
98  module INV_9 ( A, B );
99    input A;
100   output B;
101
102
103   INVD1BWP7T U1 ( .I(A), .ZN(B) );
104 endmodule
105
106
107 module INV_8 ( A, B );
108   input A;
109   output B;
110
111
112   INVD1BWP7T U1 ( .I(A), .ZN(B) );
113 endmodule
114
115
116 module INV_7 ( A, B );
117   input A;
118   output B;
119
120
121   INVD1BWP7T U1 ( .I(A), .ZN(B) );
122 endmodule
123
124
125 module INV_6 ( A, B );
126   input A;
127   output B;
128
129
130   INVD1BWP7T U1 ( .I(A), .ZN(B) );
131 endmodule
132
133
134 module INV_5 ( A, B );
135   input A;
136   output B;
137
138
139   INVD1BWP7T U1 ( .I(A), .ZN(B) );
140 endmodule
141
```

```
142
143 module INV_4 ( A, B );
144   input A;
145   output B;
146
147
148   INVD1BWP7T U1 ( .I(A), .ZN(B) );
149 endmodule
150
151
152 module INV_3 ( A, B );
153   input A;
154   output B;
155
156
157   INVD1BWP7T U1 ( .I(A), .ZN(B) );
158 endmodule
159
160
161 module INV_2 ( A, B );
162   input A;
163   output B;
164
165
166   INVD1BWP7T U1 ( .I(A), .ZN(B) );
167 endmodule
168
169
170 module INV_1 ( A, B );
171   input A;
172   output B;
173
174
175   INVD1BWP7T U1 ( .I(A), .ZN(B) );
176 endmodule
177
178
179 module INV_0 ( A, B );
180   input A;
181   output B;
182
183
184   INVD1BWP7T U1 ( .I(A), .ZN(B) );
185 endmodule
186
187
188 module chip_SP_DW01_inc_0 ( A, SUM );
189   input [8:0] A;
190   output [8:0] SUM;
191   wire   n1, n3, n4, n5, n6, n7, n8, n9;
192
193   CKXOR2D1BWP7T U1 ( .A1(n7), .A2(n1), .Z(SUM[3]) );
194   CKXOR2D1BWP7T U2 ( .A1(A[8]), .A2(n3), .Z(SUM[8]) );
195   INR2D1BWP7T U3 ( .A1(A[7]), .B1(n4), .ZN(n3) );
196   XNR2D1BWP7T U4 ( .A1(n4), .A2(A[7]), .ZN(SUM[7]) );
197   XNR2D1BWP7T U5 ( .A1(n6), .A2(A[6]), .ZN(SUM[6]) );
198   ND2D1BWP7T U6 ( .A1(A[5]), .A2(n5), .ZN(n6) );
199   INVD1BWP7T U7 ( .I(A[0]), .ZN(SUM[0]) );
200   CKXOR2D1BWP7T U8 ( .A1(A[1]), .A2(A[0]), .Z(SUM[1]) );
```

```
  XNR2D1BWP7T U9 ( .A1(n9), .A2(A[2]), .ZN(SUM[2]) );
  ND2D1BWP7T U10 ( .A1(A[1]), .A2(A[0]), .ZN(n9) );
  CKXOR2D1BWP7T U11 ( .A1(A[4]), .A2(n8), .Z(SUM[4]) );
  NR2D1BWP7T U12 ( .A1(n7), .A2(n1), .ZN(n8) );
  CKXOR2D1BWP7T U13 ( .A1(A[5]), .A2(n5), .Z(SUM[5]) );
  INR3D0BWP7T U14 ( .A1(A[4]), .B1(n1), .B2(n7), .ZN(n5) );
  ND3D0BWP7T U15 ( .A1(A[1]), .A2(A[0]), .A3(A[2]), .ZN(n7) );
  ND3D0BWP7T U16 ( .A1(A[5]), .A2(n5), .A3(A[6]), .ZN(n4) );
  INVD1BWP7T U17 ( .I(A[3]), .ZN(n1) );
endmodule


module chip_SP ( q_out, reset, clk, EN, clk_s, select );
  output [7:0] q_out;
  input [1:0] select;
  input reset, clk, EN;
  output clk_s;
  wire   n614, n613, n612, n611, n610, n609, n608, W_1, W_2, W_3, W_4, W_5,
         W_6, W_7, W_8, W_9, W_10, W_11, W_12, W_13, W_14, W_15, W_16, W_17,
         W_18, W_19, N584, N583, N582, N581, N580, N579, N578, N577, N576,
         n618, n1159, n619, n635, n620, n637, n621, n1160, n622, n625, n623,
         n1158, n617, n636, n616, n1161, n615, n629, n1, n2, n3, n4, n5, n6,
         n7, n9, n10, n11, n12, n13, n14, n15, n16, n17, n18, n19, n20, n21,
         n22, n23, n24, n25, n26, n27, n28, n29, n30, n31, n32, n33, n34,
    n35,
         n37, n38, n39, n40, n41, n42, n43, n44, n46, n47, n48, n49, n50,
    n51,
         n52, n53, n54, n55, n58, n59, n60, n61, n62, n63, n64, n65, n66,
    n67,
         n68, n69, n70, n71, n72, n73, n74, n75, n76, n77, n78, n83, n84,
    n86,
         n87, n88, n89, n90, n91, n92, n93, n94, n95, n96, n97, n98, n99,
    n100,
         n101, n102, n103, n104, n105, n106, n107, n108, n109, n110, n111,
         n112, n113, n114, n115, n116, n117, n118, n119, n120, n121, n122,
         n123, n124, n125, n126, n127, n128, n129, n130, n131, n132, n133,
         n134, n135, n136, n137, n138, n139, n140, n141, n142, n143, n144,
         n145, n146, n147, n148, n149, n150, n151, n152, n153, n154, n155,
         n156, n157, n158, n159, n160, n161, n162, n163, n164, n165, n166,
         n167, n168, n169, n170, n171, n172, n173, n174, n175, n176, n177,
         n178, n179, n180, n181, n182, n183, n184, n185, n186, n187, n188,
         n189, n190, n191, n192, n193, n194, n195, n196, n197, n198, n199,
         n200, n201, n202, n203, n204, n205, n206, n207, n208, n209, n210,
         n211, n212, n213, n214, n215, n216, n217, n218, n219, n220, n221,
         n222, n223, n224, n225, n226, n227, n228, n229, n230, n231, n232,
         n233, n234, n235, n236, n237, n238, n239, n240, n241, n242, n243,
         n244, n245, n246, n247, n248, n249, n250, n251, n252, n253, n254,
         n255, n256, n257, n258, n259, n260, n261, n262, n263, n264, n265,
         n266, n267, n268, n269, n270, n271, n272, n273, n274, n275, n276,
         n277, n278, n279, n280, n281, n282, n283, n284, n285, n286, n287,
         n288, n289, n290, n291, n292, n293, n294, n297, n298, n299, n300,
         n301, n302, n303, n304, n305, n306, n307, n308, n309, n310, n311,
         n312, n313, n314, n315, n316, n317, n318, n319, n320, n321, n322,
         n323, n324, n325, n326, n327, n328, n329, n330, n331, n332, n333,
         n334, n335, n336, n337, n338, n339, n340, n341, n342, n343, n344,
         n345, n346, n347, n348, n349, n350, n351, n352, n353, n354, n355,
         n356, n357, n358, n359, n360, n361, n362, n363, n364, n365, n366,
         n367, n368, n369, n370, n371, n372, n373, n374, n375, n376, n377,
         n378, n379, n380, n381, n382, n383, n384, n385, n386, n387, n388,
```

```
        n389, n390, n391, n392, n393, n394, n395, n396, n397, n398, n399,
        n400, n401, n402, n403, n404, n405, n406, n407, n408, n409, n410,
        n411, n412, n413, n414, n415, n416, n417, n418, n419, n420, n421,
        n422, n423, n424, n425, n426, n427, n428, n429, n430, n431, n432,
        n433, n434, n435, n436, n437, n438, n439, n440, n441, n442, n443,
        n444, n445, n446, n447, n448, n449, n450, n451, n452, n453, n454,
        n455, n456, n457, n458, n459, n460, n461, n462, n463, n464, n465,
        n466, n467, n468, n469, n470, n471, n472, n473, n474, n475, n476,
        n477, n478, n479, n480, n481, n482, n483, n484, n485, n486, n487,
        n488, n489, n490, n491, n492, n493, n494, n495, n496, n497, n498,
        n499, n500, n501, n502, n503, n504, n505, n506, n507, n508, n509,
        n510, n511, n512, n513, n514, n515, n516, n517, n518, n519, n520,
        n521, n522, n523, n524, n525, n526, n527, n528, n529, n530, n8, n36
  ;
wire   [8:0] contador;

DFQD1BWP7T \q_reg[6]  ( .D(n614), .CP(clk), .Q(q_out[6]) );
DFQD1BWP7T \q_reg[5]  ( .D(n613), .CP(clk), .Q(q_out[5]) );
DFQD1BWP7T \q_reg[4]  ( .D(n612), .CP(clk), .Q(q_out[4]) );
DFQD1BWP7T \q_reg[3]  ( .D(n611), .CP(clk), .Q(q_out[3]) );
DFQD1BWP7T \q_reg[2]  ( .D(n610), .CP(clk), .Q(q_out[2]) );
DFQD1BWP7T \q_reg[1]  ( .D(n609), .CP(clk), .Q(q_out[1]) );
DFQD1BWP7T \q_reg[0]  ( .D(n608), .CP(clk), .Q(q_out[0]) );
AND2 U1 ( .in1(EN), .in2(clk_s), .out(W_1) );
INV_18 U2 ( .A(W_1), .B(W_2) );
INV_17 U3 ( .A(W_2), .B(W_3) );
INV_16 U4 ( .A(W_3), .B(W_4) );
INV_15 U5 ( .A(W_4), .B(W_5) );
INV_14 U6 ( .A(W_5), .B(W_6) );
INV_13 U7 ( .A(W_6), .B(W_7) );
INV_12 U8 ( .A(W_7), .B(W_8) );
INV_11 U9 ( .A(W_8), .B(W_9) );
INV_10 U10 ( .A(W_9), .B(W_10) );
INV_9 U11 ( .A(W_10), .B(W_11) );
INV_8 U12 ( .A(W_11), .B(W_12) );
INV_7 U13 ( .A(W_12), .B(W_13) );
INV_6 U14 ( .A(W_13), .B(W_14) );
INV_5 U15 ( .A(W_14), .B(W_15) );
INV_4 U16 ( .A(W_15), .B(W_16) );
INV_3 U17 ( .A(W_16), .B(W_17) );
INV_2 U18 ( .A(W_17), .B(W_18) );
INV_1 U19 ( .A(W_18), .B(W_19) );
INV_0 U20 ( .A(W_19), .B(clk_s) );
chip_SP_DW01_inc_0 r80 ( .A(contador), .SUM({N584, N583, N582, N581, N580,
      N579, N578, N577, N576}) );
DFCND0BWP7T \contador_reg[8]  ( .D(n615), .CP(clk), .CDN(n83), .Q(
      contador[8]), .QN(n629) );
DFCND0BWP7T \contador_reg[7]  ( .D(n616), .CP(clk), .CDN(n83), .Q(
      contador[7]), .QN(n1161) );
DFCND0BWP7T \contador_reg[6]  ( .D(n617), .CP(clk), .CDN(n83), .Q(
      contador[6]), .QN(n636) );
DFCND0BWP7T \contador_reg[5]  ( .D(n618), .CP(clk), .CDN(n83), .Q(
      contador[5]), .QN(n1159) );
DFCND0BWP7T \contador_reg[4]  ( .D(n619), .CP(clk), .CDN(n83), .Q(
      contador[4]), .QN(n635) );
DFCND0BWP7T \contador_reg[3]  ( .D(n620), .CP(clk), .CDN(n83), .Q(
      contador[3]), .QN(n637) );
DFCND0BWP7T \contador_reg[2]  ( .D(n621), .CP(clk), .CDN(n83), .Q(
      contador[2]), .QN(n1160) );
```

```
313  DFCND0BWP7T \contador_reg[1]  ( .D(n622), .CP(clk), .CDN(n83), .Q(
314        contador[1]), .QN(n625) );
315  DFCND0BWP7T \contador_reg[0]  ( .D(n623), .CP(clk), .CDN(n83), .Q(
316        contador[0]), .QN(n1158) );
317  AOI211D1BWP7T U21 ( .A1(n32), .A2(n64), .B(n199), .C(n188), .ZN(n313) );
318  OAI22D0BWP7T U22 ( .A1(n330), .A2(n275), .B1(n256), .B2(n253), .ZN(n8) );
319  NR4D0BWP7T U23 ( .A1(n106), .A2(n107), .A3(n108), .A4(n109), .ZN(n105) );
320  INVD1BWP7T U24 ( .I(n179), .ZN(n2) );
321  INVD1BWP7T U25 ( .I(n152), .ZN(n39) );
322  INVD1BWP7T U26 ( .I(n141), .ZN(n20) );
323  NR4D0BWP7T U27 ( .A1(n345), .A2(n175), .A3(n102), .A4(n292), .ZN(n316) );
324  ND4D0BWP7T U28 ( .A1(n129), .A2(n383), .A3(n267), .A4(n384), .ZN(n345) );
325  AOI222D0BWP7T U29 ( .A1(n40), .A2(n76), .B1(n54), .B2(n42), .C1(n49), .C2(
326        n33), .ZN(n488) );
327  INR4D0BWP7T U30 ( .A1(n488), .B1(n489), .B2(n490), .B3(n291), .ZN(n482) );
328  ND3D0BWP7T U31 ( .A1(n103), .A2(n152), .A3(n158), .ZN(n175) );
329  NR3D0BWP7T U32 ( .A1(n109), .A2(n221), .A3(n357), .ZN(n352) );
330  AN4D1BWP7T U33 ( .A1(n5), .A2(n351), .A3(n352), .A4(n353), .Z(n158) );
331  AOI22D0BWP7T U34 ( .A1(n48), .A2(n30), .B1(n52), .B2(n9), .ZN(n351) );
332  NR4D0BWP7T U35 ( .A1(n354), .A2(n355), .A3(n287), .A4(n356), .ZN(n353) );
333  INVD1BWP7T U36 ( .I(n376), .ZN(n5) );
334  ND3D0BWP7T U37 ( .A1(n228), .A2(n229), .A3(n230), .ZN(n110) );
335  NR4D0BWP7T U38 ( .A1(n231), .A2(n232), .A3(n233), .A4(n234), .ZN(n230) );
336  INR4D0BWP7T U39 ( .A1(n290), .B1(n291), .B2(n292), .B3(n131), .ZN(n228) );
337  NR4D0BWP7T U40 ( .A1(n101), .A2(n242), .A3(n243), .A4(n244), .ZN(n229) );
338  ND3D0BWP7T U41 ( .A1(n245), .A2(n190), .A3(n246), .ZN(n101) );
339  AOI21D0BWP7T U42 ( .A1(n31), .A2(n58), .B(n39), .ZN(n246) );
340  AN4D1BWP7T U43 ( .A1(n493), .A2(n494), .A3(n495), .A4(n496), .Z(n247) );
341  AOI22D0BWP7T U44 ( .A1(n29), .A2(n46), .B1(n71), .B2(n41), .ZN(n493) );
342  AOI22D0BWP7T U45 ( .A1(n6), .A2(n75), .B1(n17), .B2(n235), .ZN(n494) );
343  NR4D0BWP7T U46 ( .A1(n509), .A2(n510), .A3(n198), .A4(n511), .ZN(n495) );
344  ND2D1BWP7T U47 ( .A1(n76), .A2(n41), .ZN(n405) );
345  ND2D1BWP7T U48 ( .A1(n32), .A2(n64), .ZN(n153) );
346  INVD1BWP7T U49 ( .I(n342), .ZN(n33) );
347  ND2D1BWP7T U50 ( .A1(n21), .A2(n48), .ZN(n141) );
348  IND3D1BWP7T U51 ( .A1(n127), .B1(n462), .B2(n463), .ZN(n219) );
349  AOI221D0BWP7T U52 ( .A1(n6), .A2(n75), .B1(n74), .B2(n464), .C(n286), .ZN(
350        n463) );
351  AOI22D0BWP7T U53 ( .A1(n50), .A2(n19), .B1(n12), .B2(n48), .ZN(n462) );
352  INVD1BWP7T U54 ( .I(n203), .ZN(n58) );
353  INVD1BWP7T U55 ( .I(n138), .ZN(n31) );
354  INVD1BWP7T U56 ( .I(n239), .ZN(n19) );
355  AOI22D0BWP7T U57 ( .A1(n17), .A2(n68), .B1(n13), .B2(n50), .ZN(n450) );
356  INVD1BWP7T U58 ( .I(n398), .ZN(n17) );
357  INVD1BWP7T U59 ( .I(n262), .ZN(n14) );
358  AOI22D0BWP7T U60 ( .A1(n54), .A2(n38), .B1(n6), .B2(n60), .ZN(n142) );
359  AOI222D0BWP7T U61 ( .A1(n37), .A2(n49), .B1(n71), .B2(n17), .C1(n55), .C2(
360        n38), .ZN(n370) );
361  AOI22D0BWP7T U62 ( .A1(n22), .A2(n50), .B1(n58), .B2(n40), .ZN(n368) );
362  ND4D0BWP7T U63 ( .A1(n369), .A2(n370), .A3(n371), .A4(n372), .ZN(n109) );
363  AOI22D0BWP7T U64 ( .A1(n52), .A2(n30), .B1(n34), .B2(n50), .ZN(n369) );
364  AOI211D1BWP7T U65 ( .A1(n74), .A2(n224), .B(n373), .C(n374), .ZN(n372) );
365  AOI222D0BWP7T U66 ( .A1(n31), .A2(n70), .B1(n43), .B2(n54), .C1(n72), .C2(
       n6), .ZN(n371) );
366  ND4D0BWP7T U67 ( .A1(n358), .A2(n359), .A3(n360), .A4(n361), .ZN(n221) );
367  NR4D0BWP7T U68 ( .A1(n199), .A2(n362), .A3(n363), .A4(n364), .ZN(n361) );
368  AOI222D0BWP7T U69 ( .A1(n44), .A2(n53), .B1(n70), .B2(n41), .C1(n63), .C2(
369        n24), .ZN(n359) );
370  AOI222D0BWP7T U70 ( .A1(n72), .A2(n31), .B1(n48), .B2(n366), .C1(n9), .C2(
```

```
      n186), .ZN(n360) );
IND3D1BWP7T U71 ( .A1(n357), .B1(n383), .B2(n410), .ZN(n107) );
INVD1BWP7T U72 ( .I(n149), .ZN(n7) );
ND2D1BWP7T U73 ( .A1(n54), .A2(n19), .ZN(n389) );
INVD1BWP7T U74 ( .I(n220), .ZN(n75) );
NR3D0BWP7T U75 ( .A1(n452), .A2(n205), .A3(n154), .ZN(n513) );
IND3D1BWP7T U76 ( .A1(n144), .B1(n368), .B2(n528), .ZN(n205) );
AOI221D0BWP7T U77 ( .A1(n40), .A2(n73), .B1(n74), .B2(n10), .C(n163), .ZN(
      n528) );
AN4D1BWP7T U78 ( .A1(n513), .A2(n514), .A3(n515), .A4(n516), .Z(n245) );
AOI222D0BWP7T U79 ( .A1(n13), .A2(n46), .B1(n23), .B2(n403), .C1(n76), .C2
  (
      n31), .ZN(n515) );
AOI211D1BWP7T U80 ( .A1(n61), .A2(n307), .B(n517), .C(n518), .ZN(n516) );
NR3D0BWP7T U81 ( .A1(n520), .A2(n421), .A3(n339), .ZN(n514) );
ND4D0BWP7T U82 ( .A1(n299), .A2(n300), .A3(n301), .A4(n302), .ZN(n206) );
AOI22D0BWP7T U83 ( .A1(n52), .A2(n9), .B1(n66), .B2(n17), .ZN(n299) );
AOI22D0BWP7T U84 ( .A1(n73), .A2(n24), .B1(n12), .B2(n54), .ZN(n300) );
AOI22D0BWP7T U85 ( .A1(n64), .A2(n307), .B1(n75), .B2(n308), .ZN(n301) );
NR3D0BWP7T U86 ( .A1(n206), .A2(n8), .A3(n36), .ZN(n99) );
AO221D0BWP7T U87 ( .A1(n70), .A2(n41), .B1(n74), .B2(n17), .C(n297), .Z(
  n36)
       );
NR2D1BWP7T U88 ( .A1(n66), .A2(n70), .ZN(n290) );
IND3D1BWP7T U89 ( .A1(n282), .B1(n284), .B2(n399), .ZN(n395) );
OAI21D0BWP7T U90 ( .A1(n6), .A2(n40), .B(n63), .ZN(n399) );
ND2D1BWP7T U91 ( .A1(n68), .A2(n10), .ZN(n284) );
AOI22D0BWP7T U92 ( .A1(n21), .A2(n55), .B1(n9), .B2(n53), .ZN(n337) );
NR2D1BWP7T U93 ( .A1(n50), .A2(n52), .ZN(n238) );
ND3D0BWP7T U94 ( .A1(n523), .A2(n524), .A3(n525), .ZN(n154) );
AOI22D0BWP7T U95 ( .A1(n59), .A2(n307), .B1(n65), .B2(n17), .ZN(n523) );
AOI211D1BWP7T U96 ( .A1(n10), .A2(n442), .B(n184), .C(n409), .ZN(n524) );
AOI211D1BWP7T U97 ( .A1(n9), .A2(n54), .B(n526), .C(n489), .ZN(n525) );
NR2D1BWP7T U98 ( .A1(n24), .A2(n6), .ZN(n274) );
AO211D0BWP7T U99 ( .A1(n32), .A2(n73), .B(n233), .C(n198), .Z(n379) );
AN3D1BWP7T U100 ( .A1(n276), .A2(n277), .A3(n278), .Z(n210) );
AOI211D1BWP7T U101 ( .A1(n40), .A2(n70), .B(n11), .C(n288), .ZN(n276) );
NR4D0BWP7T U102 ( .A1(n279), .A2(n280), .A3(n281), .A4(n282), .ZN(n278) );
INR4D0BWP7T U103 ( .A1(n284), .B1(n285), .B2(n286), .B3(n287), .ZN(n277) )
  ;
ND4D0BWP7T U104 ( .A1(n263), .A2(n207), .A3(n264), .A4(n265), .ZN(n132) );
AOI22D0BWP7T U105 ( .A1(n22), .A2(n46), .B1(n52), .B2(n30), .ZN(n263) );
AOI22D0BWP7T U106 ( .A1(n31), .A2(n61), .B1(n4), .B2(n54), .ZN(n264) );
AOI21D0BWP7T U107 ( .A1(n76), .A2(n6), .B(n183), .ZN(n265) );
AOI22D0BWP7T U108 ( .A1(n13), .A2(n54), .B1(n19), .B2(n46), .ZN(n470) );
IND3D1BWP7T U109 ( .A1(n119), .B1(n289), .B2(n476), .ZN(n472) );
OAI21D0BWP7T U110 ( .A1(n64), .A2(n59), .B(n41), .ZN(n476) );
AO21D0BWP7T U111 ( .A1(n46), .A2(n30), .B(n118), .Z(n244) );
AO221D0BWP7T U112 ( .A1(n53), .A2(n4), .B1(n30), .B2(n55), .C(n283), .Z(
  n279) );
AOI22D0BWP7T U113 ( .A1(n10), .A2(n66), .B1(n17), .B2(n70), .ZN(n207) );
INVD1BWP7T U114 ( .I(n289), .ZN(n11) );
INVD1BWP7T U115 ( .I(n307), .ZN(n16) );
ND2D1BWP7T U116 ( .A1(n65), .A2(n6), .ZN(n312) );
OA21D0BWP7T U117 ( .A1(n308), .A2(n24), .B(n67), .Z(n356) );
NR3D0BWP7T U118 ( .A1(n106), .A2(n127), .A3(n128), .ZN(n114) );
ND2D1BWP7T U119 ( .A1(n129), .A2(n130), .ZN(n106) );
AOI221D0BWP7T U120 ( .A1(n58), .A2(n17), .B1(n53), .B2(n4), .C(n163), .ZN(
      n179) );
```

```
  AOI22D0BWP7T U121 ( .A1(n32), .A2(n77), .B1(n52), .B2(n34), .ZN(n166) );
  ND2D1BWP7T U122 ( .A1(n71), .A2(n40), .ZN(n152) );
  NR4D0BWP7T U123 ( .A1(n154), .A2(n155), .A3(n156), .A4(n157), .ZN(n147) );
  IIND4D0BWP7T U124 ( .A1(n187), .A2(n209), .B1(n210), .B2(n211), .ZN(n157)
    );
  AN3D1BWP7T U125 ( .A1(n313), .A2(n189), .A3(n314), .Z(n100) );
  AOI21D0BWP7T U126 ( .A1(n17), .A2(n58), .B(n209), .ZN(n314) );
  INR3D0BWP7T U127 ( .A1(n207), .B1(n157), .B2(n208), .ZN(n194) );
  OAI211D1BWP7T U128 ( .A1(n238), .A2(n239), .B(n240), .C(n241), .ZN(n231) )
    ;
  AOI22D0BWP7T U129 ( .A1(n69), .A2(n41), .B1(n22), .B2(n52), .ZN(n241) );
  OAI31D0BWP7T U130 ( .A1(n42), .A2(n9), .A3(n12), .B(n50), .ZN(n240) );
  AOI221D0BWP7T U131 ( .A1(n52), .A2(n37), .B1(n46), .B2(n4), .C(n151), .ZN(
        n293) );
  AOI21D0BWP7T U132 ( .A1(n46), .A2(n9), .B(n20), .ZN(n189) );
  IND3D1BWP7T U133 ( .A1(n221), .B1(n222), .B2(n223), .ZN(n108) );
  AOI211D1BWP7T U134 ( .A1(n61), .A2(n224), .B(n39), .C(n162), .ZN(n223) );
  AOI21D0BWP7T U135 ( .A1(n64), .A2(n10), .B(n225), .ZN(n222) );
  IND3D1BWP7T U136 ( .A1(n136), .B1(n492), .B2(n470), .ZN(n188) );
  OAI21D0BWP7T U137 ( .A1(n75), .A2(n69), .B(n24), .ZN(n492) );
  OA21D0BWP7T U138 ( .A1(n235), .A2(n71), .B(n32), .Z(n234) );
  INVD1BWP7T U139 ( .I(n104), .ZN(n18) );
  NR2D1BWP7T U140 ( .A1(n46), .A2(n52), .ZN(n331) );
  AOI211D1BWP7T U141 ( .A1(n54), .A2(n13), .B(n205), .C(n206), .ZN(n195) );
  NR3D0BWP7T U142 ( .A1(n102), .A2(n167), .A3(n168), .ZN(n159) );
  IND3D1BWP7T U143 ( .A1(n151), .B1(n152), .B2(n153), .ZN(n150) );
  NR4D0BWP7T U144 ( .A1(n117), .A2(n118), .A3(n119), .A4(n120), .ZN(n116) );
  IND3D1BWP7T U145 ( .A1(n121), .B1(n122), .B2(n123), .ZN(n117) );
  NR4D0BWP7T U146 ( .A1(n181), .A2(n182), .A3(n183), .A4(n184), .ZN(n172) );
  NR2D1BWP7T U147 ( .A1(n138), .A2(n220), .ZN(n120) );
  AO21D0BWP7T U148 ( .A1(n19), .A2(n49), .B(n120), .Z(n217) );
  INVD1BWP7T U149 ( .I(n92), .ZN(n35) );
  ND2D1BWP7T U150 ( .A1(n35), .A2(n91), .ZN(n138) );
  NR4D0BWP7T U151 ( .A1(n430), .A2(n431), .A3(n167), .A4(n244), .ZN(n429) );
  OAI211D1BWP7T U152 ( .A1(n272), .A2(n367), .B(n470), .C(n471), .ZN(n430) )
    ;
  IND3D1BWP7T U153 ( .A1(n219), .B1(n14), .B2(n180), .ZN(n431) );
  AOI222D0BWP7T U154 ( .A1(n185), .A2(n49), .B1(n61), .B2(n32), .C1(n53), .
    C2(
        n23), .ZN(n471) );
  ND4D0BWP7T U155 ( .A1(n153), .A2(n141), .A3(n453), .A4(n454), .ZN(n128) );
  AOI22D0BWP7T U156 ( .A1(n52), .A2(n440), .B1(n53), .B2(n366), .ZN(n453) );
  AOI221D0BWP7T U157 ( .A1(n70), .A2(n17), .B1(n49), .B2(n9), .C(n455), .ZN(
        n454) );
  OAI22D0BWP7T U158 ( .A1(n138), .A2(n343), .B1(n213), .B2(n365), .ZN(n455)
    );
  INR4D0BWP7T U159 ( .A1(n450), .B1(n451), .B2(n128), .B3(n452), .ZN(n449) )
    ;
  OAI221D0BWP7T U160 ( .A1(n178), .A2(n375), .B1(n272), .B2(n342), .C(n456),
        .ZN(n451) );
  AOI22D0BWP7T U161 ( .A1(n15), .A2(n54), .B1(n185), .B2(n46), .ZN(n456) );
  ND4D0BWP7T U162 ( .A1(n316), .A2(n317), .A3(n318), .A4(n319), .ZN(n111) );
  NR4D0BWP7T U163 ( .A1(n320), .A2(n321), .A3(n283), .A4(n67), .ZN(n319) );
  NR4D0BWP7T U164 ( .A1(n325), .A2(n326), .A3(n327), .A4(n328), .ZN(n318) );
  INR4D0BWP7T U165 ( .A1(n337), .B1(n338), .B2(n280), .B3(n339), .ZN(n317) )
    ;
  AN4D1BWP7T U166 ( .A1(n446), .A2(n447), .A3(n448), .A4(n449), .Z(n180) );
  AOI22D0BWP7T U167 ( .A1(n37), .A2(n55), .B1(n40), .B2(n59), .ZN(n446) );
  MAOI22D0BWP7T U168 ( .A1(n69), .A2(n31), .B1(n298), .B2(n238), .ZN(n447) )
```

```
  ;
NR4D0BWP7T U169 ( .A1(n281), .A2(n457), .A3(n458), .A4(n459), .ZN(n448) );
AN4D1BWP7T U170 ( .A1(n426), .A2(n427), .A3(n428), .A4(n429), .Z(n383) );
AOI222D0BWP7T U171 ( .A1(n31), .A2(n235), .B1(n10), .B2(n478), .C1(n77),
      .C2(n402), .ZN(n427) );
AOI22D0BWP7T U172 ( .A1(n65), .A2(n17), .B1(n12), .B2(n46), .ZN(n426) );
NR3D0BWP7T U173 ( .A1(n472), .A2(n473), .A3(n474), .ZN(n428) );
INVD1BWP7T U174 ( .I(n329), .ZN(n9) );
INVD1BWP7T U175 ( .I(n439), .ZN(n76) );
OAI211D1BWP7T U176 ( .A1(n139), .A2(n138), .B(n215), .C(n479), .ZN(n357) )
  ;
AOI22D0BWP7T U177 ( .A1(n42), .A2(n50), .B1(n12), .B2(n54), .ZN(n479) );
AN4D1BWP7T U178 ( .A1(n480), .A2(n481), .A3(n482), .A4(n483), .Z(n215) );
AOI22D0BWP7T U179 ( .A1(n25), .A2(n48), .B1(n55), .B2(n30), .ZN(n481) );
AOI22D0BWP7T U180 ( .A1(n185), .A2(n52), .B1(n66), .B2(n32), .ZN(n480) );
NR4D0BWP7T U181 ( .A1(n484), .A2(n485), .A3(n209), .A4(n486), .ZN(n483) );
NR4D0BWP7T U182 ( .A1(n497), .A2(n498), .A3(n143), .A4(n376), .ZN(n496) );
OAI222D0BWP7T U183 ( .A1(n177), .A2(n272), .B1(n346), .B2(n439), .C1(n260)
  ,
      .C2(n342), .ZN(n498) );
IIND4D0BWP7T U184 ( .A1(n155), .A2(n490), .B1(n211), .B2(n384), .ZN(n497)
  );
AN4D1BWP7T U185 ( .A1(n210), .A2(n247), .A3(n248), .A4(n249), .Z(n190) );
NR4D0BWP7T U186 ( .A1(n250), .A2(n251), .A3(n181), .A4(n252), .ZN(n249) );
NR3D0BWP7T U187 ( .A1(n156), .A2(n262), .A3(n132), .ZN(n248) );
AOI21D0BWP7T U188 ( .A1(n253), .A2(n254), .B(n213), .ZN(n252) );
AN4D1BWP7T U189 ( .A1(n488), .A2(n389), .A3(n501), .A4(n502), .Z(n211) );
AOI211D1BWP7T U190 ( .A1(n68), .A2(n24), .B(n420), .C(n505), .ZN(n501) );
INR4D0BWP7T U191 ( .A1(n405), .B1(n503), .B2(n504), .B3(n422), .ZN(n502) )
  ;
AOI21D0BWP7T U192 ( .A1(n259), .A2(n342), .B(n272), .ZN(n505) );
INVD1BWP7T U193 ( .I(n347), .ZN(n64) );
OAI221D0BWP7T U194 ( .A1(n256), .A2(n310), .B1(n347), .B2(n213), .C(n507),
      .ZN(n490) );
AOI22D0BWP7T U195 ( .A1(n42), .A2(n46), .B1(n22), .B2(n53), .ZN(n507) );
INVD1BWP7T U196 ( .I(n305), .ZN(n22) );
INVD1BWP7T U197 ( .I(n178), .ZN(n49) );
ND2D1BWP7T U198 ( .A1(n35), .A2(n506), .ZN(n342) );
INVD1BWP7T U199 ( .I(n388), .ZN(n42) );
INVD1BWP7T U200 ( .I(n385), .ZN(n21) );
OAI222D0BWP7T U201 ( .A1(n346), .A2(n257), .B1(n140), .B2(n349), .C1(n258)
  ,
      .C2(n236), .ZN(n468) );
ND3D0BWP7T U202 ( .A1(n465), .A2(n466), .A3(n467), .ZN(n127) );
AOI22D0BWP7T U203 ( .A1(n69), .A2(n32), .B1(n73), .B2(n17), .ZN(n465) );
AOI221D0BWP7T U204 ( .A1(n67), .A2(n6), .B1(n64), .B2(n464), .C(n469), .ZN
  (
      n466) );
AOI211D1BWP7T U205 ( .A1(n50), .A2(n23), .B(n468), .C(n243), .ZN(n467) );
INVD1BWP7T U206 ( .I(n237), .ZN(n38) );
OA221D0BWP7T U207 ( .A1(n324), .A2(n273), .B1(n272), .B2(n236), .C(n500),
      .Z(n384) );
AOI22D0BWP7T U208 ( .A1(n70), .A2(n32), .B1(n38), .B2(n46), .ZN(n500) );
INVD1BWP7T U209 ( .I(n137), .ZN(n70) );
ND2D1BWP7T U210 ( .A1(n375), .A2(n261), .ZN(n366) );
OAI22D0BWP7T U211 ( .A1(n140), .A2(n275), .B1(n260), .B2(n259), .ZN(n243)
  );
OAI221D0BWP7T U212 ( .A1(n346), .A2(n310), .B1(n256), .B2(n343), .C(n487),
      .ZN(n484) );
```

```
  AOI22D0BWP7T U213 ( .A1(n58), .A2(n308), .B1(n29), .B2(n53), .ZN(n487) );
  INVD1BWP7T U214 ( .I(n259), .ZN(n29) );
  ND2D1BWP7T U215 ( .A1(n506), .A2(n62), .ZN(n203) );
  INVD1BWP7T U216 ( .I(n273), .ZN(n53) );
  ND2D1BWP7T U217 ( .A1(n138), .A2(n140), .ZN(n308) );
  OAI222D0BWP7T U218 ( .A1(n177), .A2(n273), .B1(n445), .B2(n213), .C1(n178)
    ,
       .C2(n340), .ZN(n242) );
  NR2D1BWP7T U219 ( .A1(n74), .A2(n58), .ZN(n445) );
  ND4D0BWP7T U220 ( .A1(n432), .A2(n433), .A3(n434), .A4(n435), .ZN(n167) );
  AOI222D0BWP7T U221 ( .A1(n53), .A2(n440), .B1(n40), .B2(n441), .C1(n24),
       .C2(n442), .ZN(n434) );
  NR4D0BWP7T U222 ( .A1(n285), .A2(n436), .A3(n437), .A4(n438), .ZN(n435) );
  AOI222D0BWP7T U223 ( .A1(n73), .A2(n10), .B1(n43), .B2(n46), .C1(n27), .C2
    (
       n78), .ZN(n433) );
  AOI211D1BWP7T U224 ( .A1(n63), .A2(n32), .B(n242), .C(n126), .ZN(n432) );
  OAI221D0BWP7T U225 ( .A1(n398), .A2(n275), .B1(n272), .B2(n237), .C(n530),
       .ZN(n452) );
  AOI22D0BWP7T U226 ( .A1(n49), .A2(n19), .B1(n68), .B2(n6), .ZN(n530) );
  ND2D1BWP7T U227 ( .A1(n78), .A2(n506), .ZN(n220) );
  OAI22D0BWP7T U228 ( .A1(n256), .A2(n203), .B1(n220), .B2(n330), .ZN(n489)
    );
  INVD1BWP7T U229 ( .I(n164), .ZN(n68) );
  OAI222D0BWP7T U230 ( .A1(n273), .A2(n342), .B1(n140), .B2(n257), .C1(n330)
    ,
       .C2(n164), .ZN(n291) );
  OAI211D1BWP7T U231 ( .A1(n330), .A2(n257), .B(n450), .C(n508), .ZN(n155) )
    ;
  AOI222D0BWP7T U232 ( .A1(n41), .A2(n58), .B1(n43), .B2(n50), .C1(n31), .C2
    (
       n73), .ZN(n508) );
  OAI211D1BWP7T U233 ( .A1(n202), .A2(n259), .B(n405), .C(n406), .ZN(n168) )
    ;
  AOI22D0BWP7T U234 ( .A1(n72), .A2(n24), .B1(n53), .B2(n38), .ZN(n406) );
  AN4D1BWP7T U235 ( .A1(n390), .A2(n391), .A3(n392), .A4(n393), .Z(n214) );
  AOI22D0BWP7T U236 ( .A1(n25), .A2(n53), .B1(n22), .B2(n55), .ZN(n390) );
  AOI22D0BWP7T U237 ( .A1(n48), .A2(n19), .B1(n70), .B2(n24), .ZN(n391) );
  NR4D0BWP7T U238 ( .A1(n394), .A2(n395), .A3(n396), .A4(n397), .ZN(n393) );
  OA211D0BWP7T U239 ( .A1(n178), .A2(n388), .B(n389), .C(n214), .Z(n129) );
  ND2D1BWP7T U240 ( .A1(n506), .A2(n95), .ZN(n239) );
  INVD1BWP7T U241 ( .I(n177), .ZN(n13) );
  INVD1BWP7T U242 ( .I(n260), .ZN(n54) );
  INVD1BWP7T U243 ( .I(n202), .ZN(n46) );
  INVD1BWP7T U244 ( .I(n200), .ZN(n74) );
  INVD1BWP7T U245 ( .I(n254), .ZN(n73) );
  INVD1BWP7T U246 ( .I(n140), .ZN(n41) );
  ND2D1BWP7T U247 ( .A1(n91), .A2(n95), .ZN(n398) );
  INVD1BWP7T U248 ( .I(n340), .ZN(n23) );
  INVD1BWP7T U249 ( .I(n255), .ZN(n32) );
  INVD1BWP7T U250 ( .I(n261), .ZN(n43) );
  OAI22D0BWP7T U251 ( .A1(n253), .A2(n330), .B1(n140), .B2(n212), .ZN(n503)
    );
  OAI221D0BWP7T U252 ( .A1(n177), .A2(n258), .B1(n257), .B2(n138), .C(n461),
       .ZN(n262) );
  AOI22D0BWP7T U253 ( .A1(n53), .A2(n21), .B1(n76), .B2(n24), .ZN(n461) );
  NR2D1BWP7T U254 ( .A1(n346), .A2(n220), .ZN(n209) );
  OAI211D1BWP7T U255 ( .A1(n347), .A2(n256), .B(n142), .C(n499), .ZN(n376) )
    ;
```

```
  AOI22D0BWP7T U256 ( .A1(n43), .A2(n49), .B1(n29), .B2(n48), .ZN(n499) );
  MOAI22D0BWP7T U257 ( .A1(n7), .A2(n349), .B1(n478), .B2(n40), .ZN(n485) );
  ND2D1BWP7T U258 ( .A1(n343), .A2(n460), .ZN(n478) );
  AOI21D0BWP7T U259 ( .A1(n138), .A2(n398), .B(n348), .ZN(n504) );
  INVD1BWP7T U260 ( .I(n272), .ZN(n52) );
  AOI21D0BWP7T U261 ( .A1(n322), .A2(n254), .B(n140), .ZN(n486) );
  OA221D0BWP7T U262 ( .A1(n367), .A2(n178), .B1(n341), .B2(n324), .C(n368),
        .Z(n358) );
  INVD1BWP7T U263 ( .I(n333), .ZN(n15) );
  NR4D0BWP7T U264 ( .A1(n174), .A2(n175), .A3(n176), .A4(n18), .ZN(n173) );
  OAI211D1BWP7T U265 ( .A1(n177), .A2(n178), .B(n179), .C(n180), .ZN(n174) )
    ;
  INVD1BWP7T U266 ( .I(n365), .ZN(n69) );
  INVD1BWP7T U267 ( .I(n212), .ZN(n66) );
  ND2D1BWP7T U268 ( .A1(n398), .A2(n213), .ZN(n149) );
  INVD1BWP7T U269 ( .I(n271), .ZN(n48) );
  ND2D1BWP7T U270 ( .A1(n398), .A2(n330), .ZN(n464) );
  INVD1BWP7T U271 ( .I(n298), .ZN(n12) );
  INVD1BWP7T U272 ( .I(n324), .ZN(n30) );
  ND2D1BWP7T U273 ( .A1(n28), .A2(n91), .ZN(n323) );
  INVD1BWP7T U274 ( .I(n213), .ZN(n6) );
  ND3D0BWP7T U275 ( .A1(n266), .A2(n267), .A3(n268), .ZN(n156) );
  AOI22D0BWP7T U276 ( .A1(n60), .A2(n10), .B1(n66), .B2(n32), .ZN(n266) );
  AOI211D1BWP7T U277 ( .A1(n74), .A2(n24), .B(n269), .C(n270), .ZN(n268) );
  AOI21D0BWP7T U278 ( .A1(n271), .A2(n272), .B(n261), .ZN(n270) );
  NR2D1BWP7T U279 ( .A1(n330), .A2(n212), .ZN(n422) );
  OA221D0BWP7T U280 ( .A1(n346), .A2(n137), .B1(n385), .B2(n341), .C(n386),
        .Z(n267) );
  AOI221D0BWP7T U281 ( .A1(n43), .A2(n55), .B1(n17), .B2(n75), .C(n387), .ZN
    (
        n386) );
  AOI21D0BWP7T U282 ( .A1(n177), .A2(n388), .B(n271), .ZN(n387) );
  INVD1BWP7T U283 ( .I(n336), .ZN(n27) );
  AOI31D0BWP7T U284 ( .A1(n139), .A2(n257), .A3(n460), .B(n255), .ZN(n459) )
    ;
  OAI211D1BWP7T U285 ( .A1(n329), .A2(n178), .B(n337), .C(n529), .ZN(n144) )
    ;
  AOI22D0BWP7T U286 ( .A1(n53), .A2(n204), .B1(n40), .B2(n64), .ZN(n529) );
  NR2D1BWP7T U287 ( .A1(n305), .A2(n178), .ZN(n420) );
  AOI21D0BWP7T U288 ( .A1(n257), .A2(n220), .B(n256), .ZN(n457) );
  INVD1BWP7T U289 ( .I(n330), .ZN(n40) );
  OAI22D0BWP7T U290 ( .A1(n7), .A2(n365), .B1(n290), .B2(n138), .ZN(n509) );
  AOI211D1BWP7T U291 ( .A1(n48), .A2(n303), .B(n304), .C(n119), .ZN(n302) );
  AOI21D0BWP7T U292 ( .A1(n164), .A2(n254), .B(n140), .ZN(n304) );
  ND4D0BWP7T U293 ( .A1(n305), .A2(n239), .A3(n237), .A4(n306), .ZN(n303) );
  NR2D1BWP7T U294 ( .A1(n33), .A2(n12), .ZN(n306) );
  IND4D0BWP7T U295 ( .A1(n208), .B1(n293), .B2(n294), .B3(n99), .ZN(n131) );
  AOI22D0BWP7T U296 ( .A1(n41), .A2(n309), .B1(n4), .B2(n50), .ZN(n294) );
  ND2D1BWP7T U297 ( .A1(n310), .A2(n200), .ZN(n309) );
  OAI22D0BWP7T U298 ( .A1(n237), .A2(n273), .B1(n274), .B2(n275), .ZN(n269)
    );
  AOI21D0BWP7T U299 ( .A1(n385), .A2(n298), .B(n178), .ZN(n437) );
  INVD1BWP7T U300 ( .I(n341), .ZN(n50) );
  NR2D1BWP7T U301 ( .A1(n388), .A2(n273), .ZN(n286) );
  INVD1BWP7T U302 ( .I(n275), .ZN(n63) );
  OAI221D0BWP7T U303 ( .A1(n272), .A2(n388), .B1(n298), .B2(n273), .C(n521),
        .ZN(n339) );
  AOI211D1BWP7T U304 ( .A1(n70), .A2(n6), .B(n162), .C(n522), .ZN(n521) );
  AOI21D0BWP7T U305 ( .A1(n138), .A2(n346), .B(n275), .ZN(n522) );
```

```
  NR2D1BWP7T U306 ( .A1(n385), .A2(n260), .ZN(n162) );
  NR2D1BWP7T U307 ( .A1(n388), .A2(n258), .ZN(n281) );
  AOI21D0BWP7T U308 ( .A1(n255), .A2(n140), .B(n348), .ZN(n362) );
  AOI21D0BWP7T U309 ( .A1(n375), .A2(n177), .B(n202), .ZN(n374) );
  NR2D1BWP7T U310 ( .A1(n255), .A2(n220), .ZN(n199) );
  ND2D1BWP7T U311 ( .A1(n254), .A2(n365), .ZN(n442) );
  INVD1BWP7T U312 ( .I(n257), .ZN(n60) );
  INVD1BWP7T U313 ( .I(n139), .ZN(n59) );
  AOI21D0BWP7T U314 ( .A1(n439), .A2(n365), .B(n398), .ZN(n438) );
  NR2D1BWP7T U315 ( .A1(n164), .A2(n255), .ZN(n282) );
  INVD1BWP7T U316 ( .I(n460), .ZN(n67) );
  NR2D1BWP7T U317 ( .A1(n258), .A2(n298), .ZN(n198) );
  AN4D1BWP7T U318 ( .A1(n123), .A2(n122), .A3(n377), .A4(n378), .Z(n103) );
  AOI211D1BWP7T U319 ( .A1(n15), .A2(n53), .B(n380), .C(n121), .ZN(n377) );
  AOI221D0BWP7T U320 ( .A1(n72), .A2(n17), .B1(n25), .B2(n54), .C(n379), .ZN
    (
       n378) );
  AOI21D0BWP7T U321 ( .A1(n305), .A2(n177), .B(n272), .ZN(n380) );
  OAI22D0BWP7T U322 ( .A1(n258), .A2(n340), .B1(n298), .B2(n178), .ZN(n526)
    );
  AOI21D0BWP7T U323 ( .A1(n322), .A2(n365), .B(n330), .ZN(n364) );
  OAI22D0BWP7T U324 ( .A1(n340), .A2(n341), .B1(n298), .B2(n272), .ZN(n143)
    );
  INVD1BWP7T U325 ( .I(n322), .ZN(n72) );
  INVD1BWP7T U326 ( .I(n348), .ZN(n71) );
  NR2D1BWP7T U327 ( .A1(n272), .A2(n385), .ZN(n184) );
  INVD1BWP7T U328 ( .I(n367), .ZN(n25) );
  AOI21D0BWP7T U329 ( .A1(n164), .A2(n310), .B(n140), .ZN(n436) );
  INVD1BWP7T U330 ( .I(n311), .ZN(n37) );
  NR2D1BWP7T U331 ( .A1(n348), .A2(n346), .ZN(n233) );
  NR2D1BWP7T U332 ( .A1(n259), .A2(n341), .ZN(n285) );
  NR2D1BWP7T U333 ( .A1(n258), .A2(n239), .ZN(n118) );
  ND3D0BWP7T U334 ( .A1(n310), .A2(n253), .A3(n349), .ZN(n441) );
  INVD1BWP7T U335 ( .I(n236), .ZN(n4) );
  ND2D1BWP7T U336 ( .A1(n258), .A2(n341), .ZN(n186) );
  ND2D1BWP7T U337 ( .A1(n346), .A2(n140), .ZN(n224) );
  AOI21D0BWP7T U338 ( .A1(n212), .A2(n348), .B(n138), .ZN(n396) );
  OAI21D0BWP7T U339 ( .A1(n58), .A2(n59), .B(n477), .ZN(n289) );
  OAI21D0BWP7T U340 ( .A1(n255), .A2(n203), .B(n346), .ZN(n477) );
  ND2D1BWP7T U341 ( .A1(n239), .A2(n259), .ZN(n204) );
  ND2D1BWP7T U342 ( .A1(n138), .A2(n256), .ZN(n402) );
  AOI21D0BWP7T U343 ( .A1(n258), .A2(n178), .B(n333), .ZN(n363) );
  INVD1BWP7T U344 ( .I(n256), .ZN(n24) );
  AOI21D0BWP7T U345 ( .A1(n342), .A2(n334), .B(n271), .ZN(n373) );
  OAI222D0BWP7T U346 ( .A1(n255), .A2(n200), .B1(n16), .B2(n212), .C1(n202),
       .C2(n305), .ZN(n354) );
  ND2D1BWP7T U347 ( .A1(n398), .A2(n140), .ZN(n307) );
  OAI22D0BWP7T U348 ( .A1(n164), .A2(n138), .B1(n177), .B2(n178), .ZN(n421)
    );
  ND4D0BWP7T U349 ( .A1(n125), .A2(n166), .A3(n417), .A4(n418), .ZN(n218) );
  NR4D0BWP7T U350 ( .A1(n419), .A2(n151), .A3(n181), .A4(n182), .ZN(n418) );
  AOI221D0BWP7T U351 ( .A1(n30), .A2(n54), .B1(n185), .B2(n50), .C(n421), .
    ZN(
       n417) );
  AO21D0BWP7T U352 ( .A1(n55), .A2(n25), .B(n420), .Z(n419) );
  OA211D0BWP7T U353 ( .A1(n47), .A2(n311), .B(n415), .C(n416), .Z(n410) );
  INVD1BWP7T U354 ( .I(n403), .ZN(n47) );
  MAOI22D0BWP7T U355 ( .A1(n38), .A2(n48), .B1(n334), .B2(n331), .ZN(n415) )
    ;
```

```
  AOI211D1BWP7T U356 ( .A1(n76), .A2(n10), .B(n218), .C(n2), .ZN(n416) );
  OA21D0BWP7T U357 ( .A1(n381), .A2(n260), .B(n382), .Z(n123) );
  AO21D0BWP7T U358 ( .A1(n255), .A2(n140), .B(n343), .Z(n382) );
  NR2D1BWP7T U359 ( .A1(n23), .A2(n44), .ZN(n381) );
  INVD1BWP7T U360 ( .I(n346), .ZN(n10) );
  NR2D1BWP7T U361 ( .A1(n257), .A2(n398), .ZN(n409) );
  OA221D0BWP7T U362 ( .A1(n137), .A2(n330), .B1(n202), .B2(n367), .C(n413),
        .Z(n165) );
  AOI211D1BWP7T U363 ( .A1(n76), .A2(n6), .B(n184), .C(n414), .ZN(n413) );
  AOI21D0BWP7T U364 ( .A1(n259), .A2(n261), .B(n272), .ZN(n414) );
  NR2D1BWP7T U365 ( .A1(n138), .A2(n347), .ZN(n287) );
  IND3D1BWP7T U366 ( .A1(n183), .B1(n165), .B2(n124), .ZN(n225) );
  AOI31D0BWP7T U367 ( .A1(n237), .A2(n385), .A3(n388), .B(n178), .ZN(n518) )
    ;
  NR2D1BWP7T U368 ( .A1(n305), .A2(n260), .ZN(n283) );
  NR2D1BWP7T U369 ( .A1(n253), .A2(n274), .ZN(n121) );
  AOI21D0BWP7T U370 ( .A1(n253), .A2(n275), .B(n255), .ZN(n511) );
  AOI21D0BWP7T U371 ( .A1(n253), .A2(n343), .B(n398), .ZN(n397) );
  AOI211D1BWP7T U372 ( .A1(n46), .A2(n15), .B(n422), .C(n423), .ZN(n125) );
  OAI22D0BWP7T U373 ( .A1(n340), .A2(n271), .B1(n329), .B2(n260), .ZN(n423)
    );
  AOI21D0BWP7T U374 ( .A1(n324), .A2(n259), .B(n178), .ZN(n288) );
  NR2D1BWP7T U375 ( .A1(n340), .A2(n272), .ZN(n163) );
  NR2D1BWP7T U376 ( .A1(n213), .A2(n139), .ZN(n119) );
  ND2D1BWP7T U377 ( .A1(n271), .A2(n273), .ZN(n403) );
  INVD1BWP7T U378 ( .I(n375), .ZN(n34) );
  OAI22D0BWP7T U379 ( .A1(n202), .A2(n340), .B1(n341), .B2(n342), .ZN(n280)
    );
  INVD1BWP7T U380 ( .I(n253), .ZN(n61) );
  ND2D1BWP7T U381 ( .A1(n439), .A2(n254), .ZN(n235) );
  AOI21D0BWP7T U382 ( .A1(n258), .A2(n271), .B(n329), .ZN(n510) );
  OA221D0BWP7T U383 ( .A1(n178), .A2(n334), .B1(n346), .B2(n322), .C(n312),
        .Z(n122) );
  INVD1BWP7T U384 ( .I(n310), .ZN(n65) );
  NR2D1BWP7T U385 ( .A1(n236), .A2(n271), .ZN(n151) );
  INVD1BWP7T U386 ( .I(n334), .ZN(n44) );
  OAI22D0BWP7T U387 ( .A1(n346), .A2(n310), .B1(n273), .B2(n261), .ZN(n517)
    );
  NR2D1BWP7T U388 ( .A1(n139), .A2(n256), .ZN(n182) );
  AOI21D0BWP7T U389 ( .A1(n236), .A2(n311), .B(n341), .ZN(n355) );
  AOI21D0BWP7T U390 ( .A1(n274), .A2(n346), .B(n212), .ZN(n474) );
  NR2D1BWP7T U391 ( .A1(n342), .A2(n202), .ZN(n183) );
  INVD1BWP7T U392 ( .I(n258), .ZN(n55) );
  OAI22D0BWP7T U393 ( .A1(n258), .A2(n259), .B1(n260), .B2(n261), .ZN(n250)
    );
  OAI221D0BWP7T U394 ( .A1(n341), .A2(n261), .B1(n346), .B2(n349), .C(n350),
        .ZN(n102) );
  AOI22D0BWP7T U395 ( .A1(n68), .A2(n24), .B1(n4), .B2(n49), .ZN(n350) );
  AOI21D0BWP7T U396 ( .A1(n255), .A2(n256), .B(n257), .ZN(n251) );
  AOI21D0BWP7T U397 ( .A1(n298), .A2(n261), .B(n202), .ZN(n297) );
  NR2D1BWP7T U398 ( .A1(n342), .A2(n258), .ZN(n181) );
  OAI221D0BWP7T U399 ( .A1(n310), .A2(n138), .B1(n213), .B2(n343), .C(n344),
        .ZN(n338) );
  AOI22D0BWP7T U400 ( .A1(n22), .A2(n48), .B1(n25), .B2(n50), .ZN(n344) );
  OAI22D0BWP7T U401 ( .A1(n331), .A2(n311), .B1(n332), .B2(n258), .ZN(n326)
    );
  NR2D1BWP7T U402 ( .A1(n29), .A2(n34), .ZN(n332) );
  OAI22D0BWP7T U403 ( .A1(n324), .A2(n341), .B1(n258), .B2(n305), .ZN(n520)
    );
```

```
738   IND4D0BWP7T U404 ( .A1(n288), .B1(n220), .B2(n253), .B3(n323), .ZN(n320) )
        ;
739   INVD1BWP7T U405 ( .I(n349), .ZN(n77) );
740   OAI22D0BWP7T U406 ( .A1(n346), .A2(n347), .B1(n274), .B2(n348), .ZN(n292)
        );
741   OAI21D0BWP7T U407 ( .A1(n258), .A2(n311), .B(n312), .ZN(n208) );
742   AOI221D0BWP7T U408 ( .A1(n55), .A2(n28), .B1(n46), .B2(n21), .C(n409), .ZN
        (
743         n104) );
744   AOI21D0BWP7T U409 ( .A1(n257), .A2(n254), .B(n330), .ZN(n327) );
745   IND3D1BWP7T U410 ( .A1(n182), .B1(n236), .B2(n237), .ZN(n232) );
746   AOI21D0BWP7T U411 ( .A1(n329), .A2(n236), .B(n202), .ZN(n328) );
747   OAI22D0BWP7T U412 ( .A1(n324), .A2(n271), .B1(n346), .B2(n253), .ZN(n136)
        );
748   ND2D1BWP7T U413 ( .A1(n185), .A2(n55), .ZN(n130) );
749   AOI21D0BWP7T U414 ( .A1(n310), .A2(n322), .B(n255), .ZN(n321) );
750   NR4D0BWP7T U415 ( .A1(n161), .A2(n162), .A3(n120), .A4(n163), .ZN(n160) );
751   OAI211D1BWP7T U416 ( .A1(n164), .A2(n138), .B(n165), .C(n166), .ZN(n161) )
        ;
752   NR4D0BWP7T U417 ( .A1(n197), .A2(n198), .A3(n20), .A4(n199), .ZN(n196) );
753   OAI222D0BWP7T U418 ( .A1(n16), .A2(n200), .B1(n201), .B2(n202), .C1(n203),
754         .C2(n138), .ZN(n197) );
755   NR2D1BWP7T U419 ( .A1(n38), .A2(n204), .ZN(n201) );
756   AOI211D1BWP7T U420 ( .A1(n58), .A2(n24), .B(n135), .C(n136), .ZN(n134) );
757   OAI221D0BWP7T U421 ( .A1(n137), .A2(n138), .B1(n139), .B2(n140), .C(n141),
758         .ZN(n135) );
759   AN3D1BWP7T U422 ( .A1(n26), .A2(n124), .A3(n125), .Z(n115) );
760   INVD1BWP7T U423 ( .I(n126), .ZN(n26) );
761   NR2D1BWP7T U424 ( .A1(n212), .A2(n213), .ZN(n187) );
762   AOI221D0BWP7T U425 ( .A1(n37), .A2(n48), .B1(n185), .B2(n186), .C(n187),
763         .ZN(n171) );
764   OA211D0BWP7T U426 ( .A1(n227), .A2(n84), .B(n100), .C(n1), .Z(n97) );
765   INVD1BWP7T U427 ( .I(n110), .ZN(n1) );
766   NR3D0BWP7T U428 ( .A1(n315), .A2(n18), .A3(n111), .ZN(n227) );
767   IND3D1BWP7T U429 ( .A1(n225), .B1(n130), .B2(n410), .ZN(n315) );
768   ND2D1BWP7T U430 ( .A1(contador[5]), .A2(contador[4]), .ZN(n92) );
769   ND2D1BWP7T U431 ( .A1(n512), .A2(n475), .ZN(n329) );
770   ND2D1BWP7T U432 ( .A1(n519), .A2(n78), .ZN(n439) );
771   ND2D1BWP7T U433 ( .A1(n519), .A2(n407), .ZN(n347) );
772   ND2D1BWP7T U434 ( .A1(n491), .A2(n95), .ZN(n305) );
773   ND2D1BWP7T U435 ( .A1(n407), .A2(n94), .ZN(n178) );
774   ND2D1BWP7T U436 ( .A1(n491), .A2(n411), .ZN(n365) );
775   NR3D0BWP7T U437 ( .A1(contador[8]), .A2(contador[7]), .A3(contador[6]), .
        ZN(
776         n506) );
777   NR2D1BWP7T U438 ( .A1(contador[3]), .A2(contador[2]), .ZN(n91) );
778   ND2D1BWP7T U439 ( .A1(n425), .A2(n506), .ZN(n388) );
779   ND2D1BWP7T U440 ( .A1(n443), .A2(n475), .ZN(n213) );
780   ND2D1BWP7T U441 ( .A1(n519), .A2(n95), .ZN(n385) );
781   ND2D1BWP7T U442 ( .A1(n491), .A2(n475), .ZN(n236) );
782   ND2D1BWP7T U443 ( .A1(n425), .A2(n519), .ZN(n237) );
783   ND2D1BWP7T U444 ( .A1(n411), .A2(n519), .ZN(n137) );
784   ND2D1BWP7T U445 ( .A1(n424), .A2(n35), .ZN(n375) );
785   ND2D1BWP7T U446 ( .A1(n519), .A2(n35), .ZN(n259) );
786   ND2D1BWP7T U447 ( .A1(n424), .A2(n62), .ZN(n343) );
787   INVD1BWP7T U448 ( .I(n335), .ZN(n62) );
788   ND2D1BWP7T U449 ( .A1(n425), .A2(n512), .ZN(n261) );
789   INVD1BWP7T U450 ( .I(n444), .ZN(n78) );
790   ND2D1BWP7T U451 ( .A1(n527), .A2(n78), .ZN(n273) );
```

```
  INR2D1BWP7T U452 ( .A1(n491), .B1(n92), .ZN(n185) );
  ND2D1BWP7T U453 ( .A1(n407), .A2(n506), .ZN(n275) );
  ND2D1BWP7T U454 ( .A1(n512), .A2(n411), .ZN(n164) );
  ND2D1BWP7T U455 ( .A1(n519), .A2(n62), .ZN(n257) );
  NR3D0BWP7T U456 ( .A1(n404), .A2(n176), .A3(n168), .ZN(n392) );
  OAI22D0BWP7T U457 ( .A1(n202), .A2(n408), .B1(n260), .B2(n375), .ZN(n404)
    );
  ND2D1BWP7T U458 ( .A1(n527), .A2(n407), .ZN(n260) );
  ND2D1BWP7T U459 ( .A1(n519), .A2(n475), .ZN(n177) );
  ND2D1BWP7T U460 ( .A1(n411), .A2(n94), .ZN(n202) );
  ND2D1BWP7T U461 ( .A1(n408), .A2(n239), .ZN(n440) );
  ND2D1BWP7T U462 ( .A1(n491), .A2(n78), .ZN(n200) );
  ND2D1BWP7T U463 ( .A1(n512), .A2(n78), .ZN(n254) );
  ND2D1BWP7T U464 ( .A1(n425), .A2(n91), .ZN(n140) );
  ND2D1BWP7T U465 ( .A1(n512), .A2(n95), .ZN(n340) );
  AOI21D0BWP7T U466 ( .A1(n408), .A2(n236), .B(n260), .ZN(n469) );
  ND2D1BWP7T U467 ( .A1(n35), .A2(n443), .ZN(n255) );
  OAI222D0BWP7T U468 ( .A1(n178), .A2(n237), .B1(n444), .B2(n323), .C1(n260)
    ,
        .C2(n311), .ZN(n126) );
  ND2D1BWP7T U469 ( .A1(n512), .A2(n407), .ZN(n212) );
  ND2D1BWP7T U470 ( .A1(n424), .A2(n78), .ZN(n349) );
  AOI21D0BWP7T U471 ( .A1(n408), .A2(n237), .B(n341), .ZN(n458) );
  ND2D1BWP7T U472 ( .A1(n411), .A2(n506), .ZN(n348) );
  NR2D1BWP7T U473 ( .A1(contador[4]), .A2(contador[5]), .ZN(n95) );
  ND2D1BWP7T U474 ( .A1(n527), .A2(n411), .ZN(n272) );
  ND2D1BWP7T U475 ( .A1(n91), .A2(n475), .ZN(n346) );
  ND2D1BWP7T U476 ( .A1(n424), .A2(n475), .ZN(n333) );
  ND2D1BWP7T U477 ( .A1(n78), .A2(n94), .ZN(n271) );
  ND2D1BWP7T U478 ( .A1(n506), .A2(n475), .ZN(n298) );
  ND2D1BWP7T U479 ( .A1(n512), .A2(n35), .ZN(n324) );
  AO222D0BWP7T U480 ( .A1(n6), .A2(n73), .B1(n407), .B2(n27), .C1(n54), .C2(
        n33), .Z(n176) );
  ND2D1BWP7T U481 ( .A1(n425), .A2(n491), .ZN(n311) );
  ND2D1BWP7T U482 ( .A1(n424), .A2(n407), .ZN(n460) );
  INVD1BWP7T U483 ( .I(n408), .ZN(n28) );
  ND2D1BWP7T U484 ( .A1(n28), .A2(n443), .ZN(n336) );
  ND2D1BWP7T U485 ( .A1(n491), .A2(n407), .ZN(n310) );
  ND2D1BWP7T U486 ( .A1(n425), .A2(n443), .ZN(n330) );
  OAI211D1BWP7T U487 ( .A1(n335), .A2(n323), .B(n400), .C(n401), .ZN(n394) )
    ;
  OAI21D0BWP7T U488 ( .A1(n54), .A2(n403), .B(n185), .ZN(n400) );
  AOI22D0BWP7T U489 ( .A1(n74), .A2(n402), .B1(n69), .B2(n224), .ZN(n401) );
  ND2D1BWP7T U490 ( .A1(n94), .A2(n62), .ZN(n341) );
  ND2D1BWP7T U491 ( .A1(n491), .A2(n62), .ZN(n139) );
  ND2D1BWP7T U492 ( .A1(n512), .A2(n62), .ZN(n253) );
  ND2D1BWP7T U493 ( .A1(n424), .A2(n411), .ZN(n322) );
  ND2D1BWP7T U494 ( .A1(n90), .A2(contador[2]), .ZN(n258) );
  NR2D1BWP7T U495 ( .A1(n335), .A2(contador[3]), .ZN(n90) );
  ND2D1BWP7T U496 ( .A1(n443), .A2(n95), .ZN(n256) );
  ND2D1BWP7T U497 ( .A1(n424), .A2(n95), .ZN(n367) );
  ND2D1BWP7T U498 ( .A1(n424), .A2(n425), .ZN(n334) );
  AOI221D0BWP7T U499 ( .A1(n411), .A2(n27), .B1(n32), .B2(n76), .C(n412), .
    ZN(
        n124) );
  OAI22D0BWP7T U500 ( .A1(n139), .A2(n398), .B1(n329), .B2(n271), .ZN(n412)
    );
  AOI21D0BWP7T U501 ( .A1(n408), .A2(n333), .B(n271), .ZN(n473) );
  OAI222D0BWP7T U502 ( .A1(n238), .A2(n333), .B1(n51), .B2(n334), .C1(n335),
```

```
845         .C2(n336), .ZN(n325) );
846   INVD1BWP7T U503 ( .I(n186), .ZN(n51) );
847   INR2D1BWP7T U504 ( .A1(N579), .B1(n86), .ZN(n620) );
848   IOA21D1BWP7T U505 ( .A1(contador[1]), .A2(n94), .B(n95), .ZN(n93) );
849   AO222D0BWP7T U506 ( .A1(n226), .A2(n84), .B1(select[1]), .B2(n107), .C1(
850         q_out[0]), .C2(n97), .Z(n608) );
851   ND4D0BWP7T U507 ( .A1(n293), .A2(n245), .A3(n247), .A4(n313), .ZN(n226) );
852   AO222D0BWP7T U508 ( .A1(n110), .A2(n84), .B1(q_out[5]), .B2(n97), .C1(
853         select[1]), .C2(n111), .Z(n613) );
854   AO221D0BWP7T U509 ( .A1(select[1]), .A2(n192), .B1(q_out[1]), .B2(n97), .C
        (
855         n193), .Z(n609) );
856   IND4D0BWP7T U510 ( .A1(n108), .B1(n214), .B2(n215), .B3(n216), .ZN(n192) )
        ;
857   AOI31D0BWP7T U511 ( .A1(n194), .A2(n195), .A3(n196), .B(select[1]), .ZN(
        n193) );
858   NR3D0BWP7T U512 ( .A1(n217), .A2(n218), .A3(n219), .ZN(n216) );
859   AO221D0BWP7T U513 ( .A1(n169), .A2(n84), .B1(q_out[2]), .B2(n97), .C(n170)
        ,
860         .Z(n610) );
861   IND4D0BWP7T U514 ( .A1(n188), .B1(n189), .B2(n190), .B3(n191), .ZN(n169) )
        ;
862   AOI31D0BWP7T U515 ( .A1(n171), .A2(n172), .A3(n173), .B(n84), .ZN(n170) );
863   AOI211D1BWP7T U516 ( .A1(n6), .A2(n58), .B(n187), .C(n118), .ZN(n191) );
864   AO221D0BWP7T U517 ( .A1(select[1]), .A2(n145), .B1(q_out[3]), .B2(n97), .C
        (
865         n146), .Z(n611) );
866   AOI21D0BWP7T U518 ( .A1(n147), .A2(n148), .B(select[1]), .ZN(n146) );
867   ND4D0BWP7T U519 ( .A1(n158), .A2(n104), .A3(n159), .A4(n160), .ZN(n145) );
868   AOI211D1BWP7T U520 ( .A1(n58), .A2(n149), .B(n150), .C(n118), .ZN(n148) );
869   AO221D0BWP7T U521 ( .A1(n112), .A2(n84), .B1(q_out[4]), .B2(n97), .C(n113)
        ,
870         .Z(n612) );
871   IIND4D0BWP7T U522 ( .A1(n131), .A2(n132), .B1(n133), .B2(n134), .ZN(n112)
        );
872   AOI31D0BWP7T U523 ( .A1(n114), .A2(n115), .A3(n116), .B(n84), .ZN(n113) );
873   INR3D0BWP7T U524 ( .A1(n142), .B1(n143), .B2(n144), .ZN(n133) );
874   AO221D0BWP7T U525 ( .A1(select[1]), .A2(n96), .B1(q_out[6]), .B2(n97), .C(
875         n98), .Z(n614) );
876   AOI31D0BWP7T U526 ( .A1(n99), .A2(n3), .A3(n100), .B(select[1]), .ZN(n98)
        );
877   IND4D0BWP7T U527 ( .A1(n102), .B1(n103), .B2(n104), .B3(n105), .ZN(n96) );
878   INVD1BWP7T U528 ( .I(n101), .ZN(n3) );
879   NR2D1BWP7T U529 ( .A1(contador[5]), .A2(n635), .ZN(n475) );
880   NR3D0BWP7T U530 ( .A1(contador[8]), .A2(n1161), .A3(contador[6]), .ZN(n519
        )
881         );
882   NR3D0BWP7T U531 ( .A1(n1161), .A2(n636), .A3(contador[8]), .ZN(n491) );
883   NR2D1BWP7T U532 ( .A1(n1158), .A2(n625), .ZN(n407) );
884   NR2D1BWP7T U533 ( .A1(contador[1]), .A2(n1158), .ZN(n411) );
885   NR3D0BWP7T U534 ( .A1(contador[7]), .A2(n636), .A3(contador[8]), .ZN(n512)
886         );
887   NR2D1BWP7T U535 ( .A1(n1160), .A2(n637), .ZN(n94) );
888   ND2D1BWP7T U536 ( .A1(n625), .A2(n1158), .ZN(n335) );
889   ND2D1BWP7T U537 ( .A1(n1158), .A2(contador[1]), .ZN(n444) );
890   NR3D0BWP7T U538 ( .A1(contador[7]), .A2(n629), .A3(contador[6]), .ZN(n424)
891         );
892   NR2D1BWP7T U539 ( .A1(contador[2]), .A2(n637), .ZN(n443) );
893   NR2D1BWP7T U540 ( .A1(contador[4]), .A2(n1159), .ZN(n425) );
```

```
  ND4D0BWP7T U541 ( .A1(n1161), .A2(n95), .A3(contador[8]), .A4(contador[6]),
       .ZN(n408) );
  NR2D1BWP7T U542 ( .A1(contador[3]), .A2(n1160), .ZN(n527) );
  OAI31D0BWP7T U543 ( .A1(n90), .A2(n1159), .A3(n91), .B(n92), .ZN(n89) );
  INR2D1BWP7T U544 ( .A1(N584), .B1(n86), .ZN(n615) );
  INR2D1BWP7T U545 ( .A1(N583), .B1(n86), .ZN(n616) );
  INR2D1BWP7T U546 ( .A1(N582), .B1(n86), .ZN(n617) );
  INR2D1BWP7T U547 ( .A1(N576), .B1(n86), .ZN(n623) );
  INR2D1BWP7T U548 ( .A1(N577), .B1(n86), .ZN(n622) );
  INR2D1BWP7T U549 ( .A1(N578), .B1(n86), .ZN(n621) );
  INR2D1BWP7T U550 ( .A1(N580), .B1(n86), .ZN(n619) );
  INR2D1BWP7T U551 ( .A1(N581), .B1(n86), .ZN(n618) );
  OAI21D0BWP7T U552 ( .A1(n629), .A2(n87), .B(n88), .ZN(n86) );
  AOI211D1BWP7T U553 ( .A1(n93), .A2(contador[6]), .B(n84), .C(contador[7]),
       .ZN(n87) );
  ND4D0BWP7T U554 ( .A1(n89), .A2(contador[7]), .A3(contador[6]), .A4(n84),
       .ZN(n88) );
  INVD1BWP7T U555 ( .I(reset), .ZN(n83) );
  INVD1BWP7T U556 ( .I(select[1]), .ZN(n84) );
  TIELBWP7T U557 ( .ZN(q_out[7]) );
endmodule


module chip_IO ( q, reset, clk, EN, clk_s, select );
  output [7:0] q;
  input [1:0] select;
  input reset, clk, EN;
  output clk_s;
  wire   reset_w, clk_w, EN_w, clk_s_w, net1001, n2;
  wire   [7:0] q_w;
  wire   [1:0] select_w;
  tri   [7:0] q;
  tri   reset;
  tri   clk;
  tri   EN;
  tri   clk_s;
  tri   [1:0] select;

  chip_SP chip ( .q_out(q_w), .reset(reset_w), .clk(clk_w), .EN(EN_w), .
    clk_s(
       clk_s_w), .select(select_w) );
  PDDW0204SCDG IN00 ( .I(net1001), .OEN(n2), .IE(n2), .PAD(clk), .DS(net1001
    ),
       .PE(net1001), .C(clk_w) );
  PDDW0204SCDG IN01 ( .I(net1001), .OEN(n2), .IE(n2), .PAD(reset), .DS(
    net1001), .PE(net1001), .C(reset_w) );
  PDDW0204SCDG IN02 ( .I(net1001), .OEN(n2), .IE(n2), .PAD(EN), .DS(net1001),
       .PE(net1001), .C(EN_w) );
  PDDW0204SCDG IN03 ( .I(net1001), .OEN(n2), .IE(n2), .PAD(select[0]), .DS(
       net1001), .PE(net1001), .C(select_w[0]) );
  PDDW0204SCDG IN04 ( .I(net1001), .OEN(n2), .IE(n2), .PAD(select[1]), .DS(
       net1001), .PE(net1001), .C(select_w[1]) );
  PDDW0204SCDG OUT00 ( .I(q_w[0]), .OEN(net1001), .IE(net1001), .PAD(q[0]),
       .DS(net1001), .PE(net1001) );
  PDDW0204SCDG OUT01 ( .I(q_w[1]), .OEN(net1001), .IE(net1001), .PAD(q[1]),
       .DS(net1001), .PE(net1001) );
  PDDW0204SCDG OUT02 ( .I(q_w[2]), .OEN(net1001), .IE(net1001), .PAD(q[2]),
```

```
948         .DS(net1001), .PE(net1001) );
949   PDDW0204SCDG OUT03 ( .I(q_w[3]), .OEN(net1001), .IE(net1001), .PAD(q[3]),
950         .DS(net1001), .PE(net1001) );
951   PDDW0204SCDG OUT04 ( .I(q_w[4]), .OEN(net1001), .IE(net1001), .PAD(q[4]),
952         .DS(net1001), .PE(net1001) );
953   PDDW0204SCDG OUT05 ( .I(q_w[5]), .OEN(net1001), .IE(net1001), .PAD(q[5]),
954         .DS(net1001), .PE(net1001) );
955   PDDW0204SCDG OUT06 ( .I(q_w[6]), .OEN(net1001), .IE(net1001), .PAD(q[6]),
956         .DS(net1001), .PE(net1001) );
957   PDDW0204SCDG OUT07 ( .I(net1001), .OEN(net1001), .IE(net1001), .PAD(q[7]),
958         .DS(net1001), .PE(net1001) );
959   PDDW0204SCDG OUT08 ( .I(clk_s_w), .OEN(net1001), .IE(net1001), .PAD(clk_s)
        ,
960         .DS(net1001), .PE(net1001) );
961   TIELBWP7T U6 ( .ZN(net1001) );
962   TIEHBWP7T U7 ( .Z(n2) );
963 endmodule
```