Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Manual de las herramientas Synopsys utilizadas en el diseño ASIC

Ejemplo de diseño para explicar el flujo de diseño de un sistema digital utilizando las herramientas de Synopsys.

Jairo Mauricio Valverde Cruz

Modificado por Dave Porras Alvarado

Cartago, 2 de septiembre de 2011

Índice general

Índice de figuras

Lista de símbolos y abreviaciones

Abreviaciones

ASIC	Circuitos Integrados de Aplicación Específica
HDL	Lenguaje de Descripción de Hardware
RTL	Nivel de Registros de Transferencia
VCS	Compilador y Simulador de Verilog

Capítulo 1

Introducción

En este manual se describe la configuración y utilización de las herramientas Synopsys utilizadas en el flujo para el diseño ASIC.

En el capítulo 2 se detalla el flujo para el diseño ASIC implementado utilizando las herramientas comerciales de *Synopsys*. En el capítulo 4 se describe la herramienta *Design Compiler* que permite realizar la *síntesis RTL*. En el capítulo 7 se explica cómo obtener la simulación lógica por medio de la herramienta *VCS*. La implementación física utilizando en la herramienta *IC Compiler* se detalla en el capítulo 6. Para finalizar, en el capítulo 8 se muestra la utilización de la herramienta *Primetime* para obtener la verificación de temporizado del layout.

Los *comandos* expuestos en este documento se tomaron de [1] y de los manuales de la herramientas de *Synopsys*. Además para la parte de la *síntesis RTL* fue de gran ayuda contar con el trabajo de [2].

Capítulo 2

Flujo para el diseño de circuitos integrados de aplicación específica (ASIC)

La *integración física "correcta por construcción (CBC)"* es el proceso por el que se obtienen los archivos necesarios para la fabricación de un circuito integrado a partir de la descripción en HDL de un sistema digital. Durante este proceso la estructura interna del sistema digital puede variar pero sin alterar su funcionamiento lógico.

Para realizar la *integración física "correcta por construcción (CBC)"* de un sistema digital descrito en alto nivel se utiliza un flujo para el *diseño ASIC*. Las etapas del flujo para el *diseño ASIC* son descritas a continuación [1]:

1. **Idea**:
   Es el origen del flujo, corresponde a lo que se desea diseñar.

2. **Especificaciones *ASIC***:
   Como especificaciones *ASIC* se consideran los siguientes aspectos:
   - Objetivos y limitaciones del diseño.
   - Especificaciones lógicas:
     La función lógica que debe realizar el sistema digital.
   - Especificaciones físicas:
     El consumo de potencia y temporizado del sistema digital.

3. **RTL (Register Transfer Level)**:
   La idea y las especificaciones lógicas son plasmadas en un diseño a nivel de registros de transferencia utilizando un (*HDL*).

4. **Simulación lógica**:
   Permite verificar el funcionamiento lógico del *RTL*. Esta simulación se realiza en las siguientes tres etapas del flujo para el *diseño de circuitos integrados de aplicación específica (ASIC)*:

- Durante la creación del *RTL*.
- Después de realizar la *síntesis RTL*.
- Después de la *implementación física*.

5. **Síntesis RTL**:
   Es el proceso en el cual el diseño a nivel de registros de transferencia es mapeado a una lista de nodos a nivel de compuertas lógicas (Gate Level Netlist) que contiene sólo bloques lógicos incluidos en la biblioteca de celdas estándar.

   La biblioteca de celdas estándar posee bloques lógicos tales como compuertas lógicas, registros, y arreglos de compuertas lógicas.

   Cada bloque lógico incluido en la biblioteca de celdas estándar contiene tres formas de representación, las cuales se describen a continuación:

   - Representación CELL:
     Contiene el trazado físico.
   - Representación FRAM:
     Posee el trazado físico en forma más simple utilizado durante la implementación física.
   - Representación LM:
     Incluye la información de temporizado y del consumo de potencia.

6. **Implementación física**:
   En esta etapa se obtiene el trazado físico (layout) del sistema digital a partir de la lista de nodos a nivel de compuertas (Gate Level Netlist) obtenida en la *Síntesis RTL*. Además se genera la lista de nodos a nivel de compuertas con la información de las parásitas (resistencias y capacitancias).

   La *implementación física* consta de las siguientes tres etapas:

   - *Descripción física*:
     Se transforma la descripción lógica de la lista de nodos a nivel de compuertas en una descripción física y se busca minimizar el área y el retardo de sistema digital. Durante esta etapa de la *implementación física* se hace una estimación del área del circuito integrado, se asignan los pines, y se crean los anillos de alimentación del circuito integrado.
   - *Colocación física*:
     En esta etapa se coloca en un lugar específico los diferentes bloques lógicos que conforma la lista de nodos a nivel de compuestas. Se trata de minimizar al área (colocando los bloques lo más cerca posible) y se busca minimizar el largo del los cables que unen los diferentes bloques lógicos (colocando en forma adyacente los bloques necesarios para realizar determinada función).
   - *Enrutado físico*:
     Consiste en unir entre sí los diferentes bloques lógicos que conforman el circuito integrado. Al finalizar esta etapa, el trazado físico del sistema digital ha sido creado y se guarda como archivo *GDSII*.

7. **Verificación de temporizado**:
   Se determinan las rutas críticas del diseño y se verifica que no existan problemas de temporizado.

8. **Simulación eléctrica**:
   Permite determinar la existencia de errores en el *trazado físico* (circuito abierto o corto circuito).

9. **GDSII**:
   Es el archivo que contiene toda la información del trazado físico y se utiliza para fabricar el circuito integrado.

El flujo para el *diseño ASIC* se muestra en la figura 2.1, en donde se observan las etapas necesarias para originar los archivos utilizados en la fabricación de un circuito integrado a partir de una idea y especificaciones de un sistema digital.

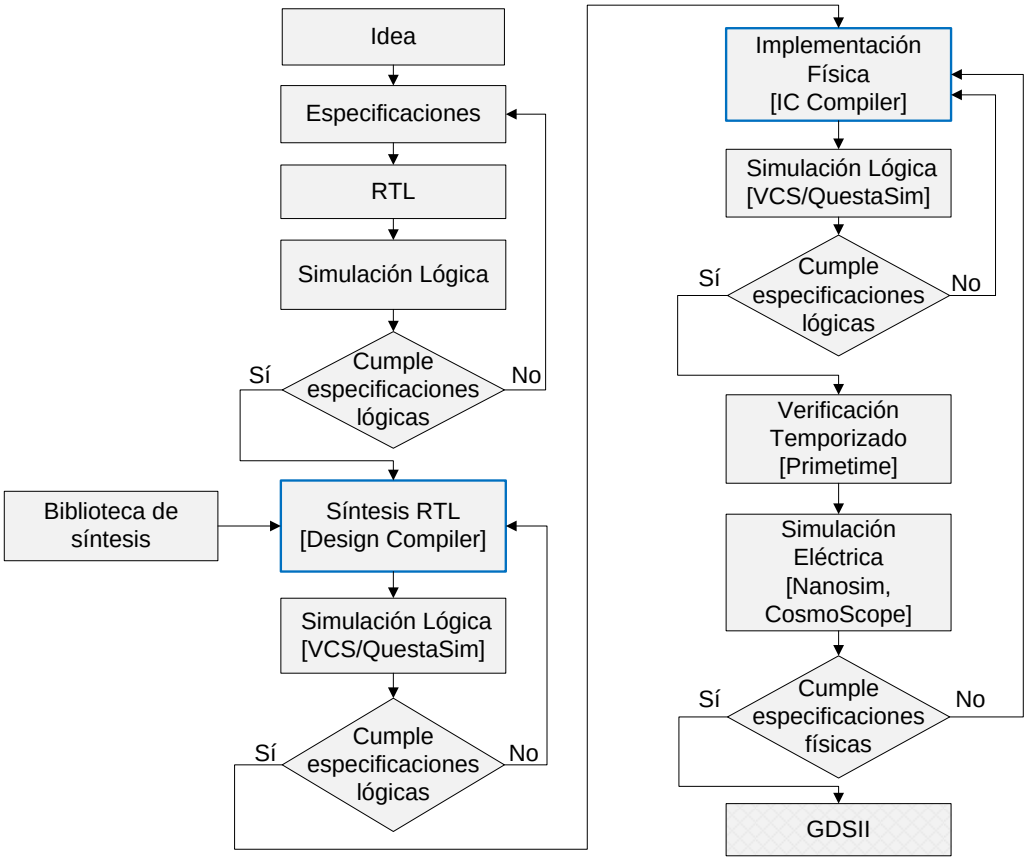

Figura 2.1: Flujo para el diseño de circuitos integrados de aplicación específica (ASIC).

Capítulo 3

Ejemplo de flujo de diseño Synopsys

3.1 Diseño de una máquina de estados

3.1.1 Idea

Como ejercicio para comprender el flujo de diseño de un circuito digital con las herramientas de *synopsys*, se procederá a implementar un detector de secuencia en una entrada del sistema.

El sistema debe detectar la secuencia *1101* en la señal *dato* de entrada, el sistema también cuenta con una entrada de reloj (*clk*) y una señal de restablecimiento (*reset*). Al detectarse la secuencia se debe acertar la salida (*detectada*).

3.1.2 Especificaciones Lógicas

La figura 3.1 muestra el diagrama de estados que interpreta el comportamiento deseado del detector de secuencia. Para este diagrama los datos en las transiciones indican el comportamiento de *entrada/salida*.

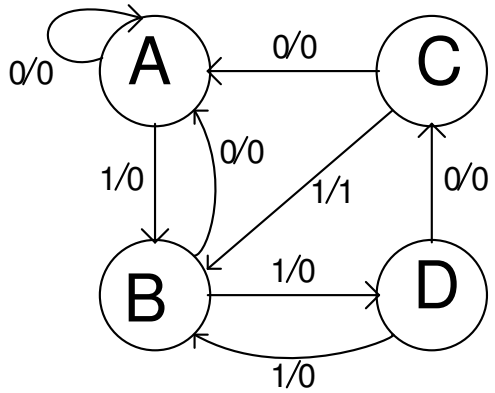

Figura 3.1: Diagrama de estados para el detector de secuencia.

Para probar el correcto comportamiento lógico se diseña un *testbench* con una secuencia en la señal de entrada de *1101101011101* de manera que la salida sea *0001001000001* tal y como se muestra en la figura 3.2.

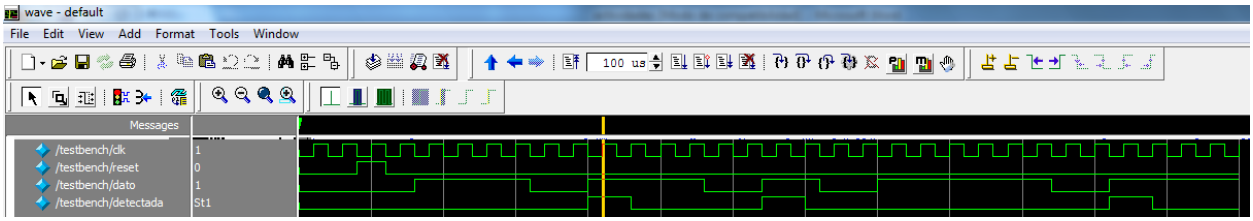

Figura 3.2: Formas de onda obtenidas de la simulación del sistema detector de secuencia.

3.1.3 Especificaciones físicas

Las especificaciones de temporizado son las siguientes:

1. Frecuencia del reloj de entrada clk = 50 kHz, con clock skew y jetter de 30 ns.
2. El reloj de la máquina de estados trabaja a la mitad de la frecuencia del reloj de entrada.
3. Retardo en las señales de entrada de máximo 200 ns y mínimo 100 ns.
4. Retardo en las señales de salida máximo de 400 ns y mínimo de 200 ns.
5. Fanout de 10.
6. La señales de entrada son manejadas por la celda estándar *INVX8*.

Capítulo 4

Design Compiler para la síntesis RTL

4.1 Estructura del directorio front_end

La estructura del directorio *front_end* se muestra en la figura 4.1 y contiene todos los archivos necesarios para realizar la *Síntesis RTL* así como los archivos generados durante este proceso.

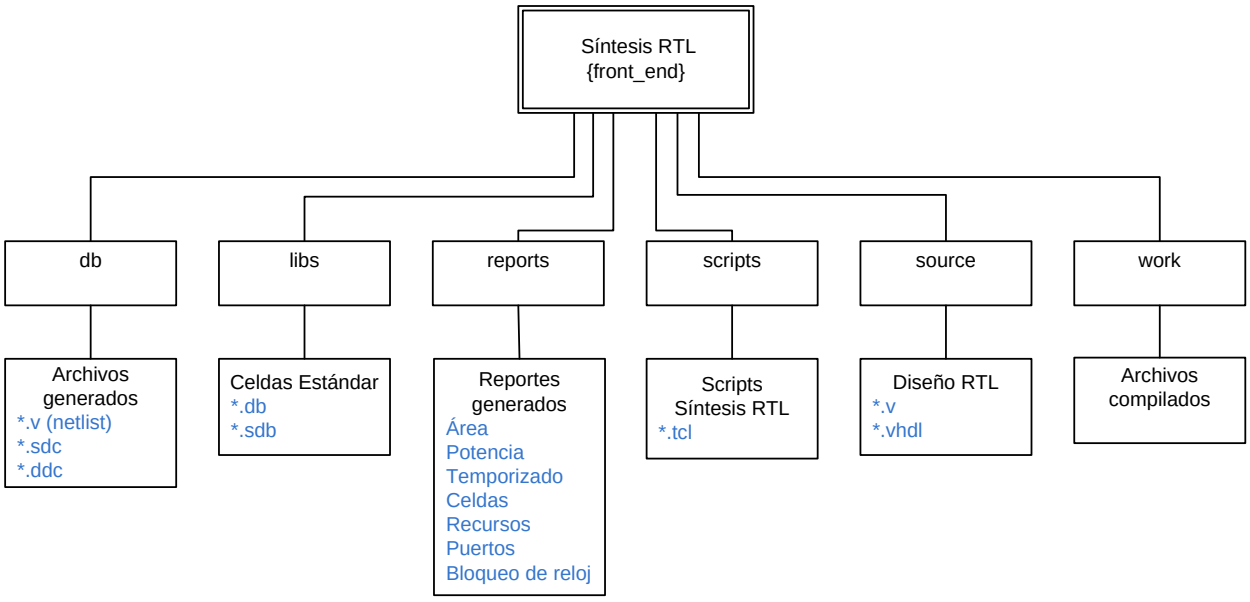

Figura 4.1: Estructura del directorio front_end.

Los subdirectorios pertenecientes a *front_end* se describe a continuación:

- **db:**
  Los dos archivos principales son el archivo *.v (gate level netlist) y el archivo *.sdc que contiene las especificaciones de temporizado. Ambos archivos son generados durante la *síntesis RTL* y son los que se utilizan en las siguientes etapas del flujo para el diseño ASIC.

- **libs:**
  Contiene la biblioteca de celdas estándar así como su representación gráfica (*.sdb).
- **reports:**
  Durante el proceso de *síntesis RTL* se generan reportes de diferentes aspectos del diseño y se almacenan en archivos de texto.
- **scripts:**
  Contiene los archivos de síntesis, son dos *.tcl en donde se encuentran todos los comandos para realizar la síntesis.
- **source:**
  Los diseños en RTL pueden ser en *.v ó *.vhdl.
- **work:**
  Contiene los archivos que genera *Design Compiler* durante el proceso de *síntesis RTL*, éstos archivos no se utilizan en las siguientes etapas del diseño.

4.1.1 Procedimiento para la sintesis RTL

- Se debe copiar el archivo *.bashrc* ubicado en */home/tutorial_synopsys/* en el directorio personal de cada cuenta (*home*). Estando ubicado en el directorio */home/tutorial_synopsys* se ejecuta: *cp .bashrc /home/**user**/*

- Se debe copiar el directorio *detector/* ubicado en */home/tutorial_synopsys/* en el directorio personal de cada cuenta (*home*). Estando ubicado en el directorio */home/tutorial_synopsys* se ejecuta: *scp -r detector /home/**user**/*

- Reiniciar sesión *ssh*

- En la ruta */home/**user**/detector/integracion_fisica/front_end/* modificar la linea 16 del archivo *.synopsys_dc.setup* sustituyendo *jvalverde* por su nombre de usuario, para abrir este archivo ejecute *gedit .synopsys_dc.setup*.

- *Script de síntesis con el software Design Compiler*: El archivo *detector_syn.tcl* localizado en el directorio */home/**user**/detector/integracion_fisica/front_end/scripts/* debe revisarse para corroborar que se encuentre configurado de la siguiente manera:

  1. Eliminar diseños previos:
     Comando: *remove_design -designs*
  2. En caso de utilizar la técnica de bloqueo del reloj (Clock Gating) utilizar:
     Comando: *set_clock_gating_style -sequential_cell latch -positive_edge_logic and -negative_edge_logic or*
  3. Analizar el módulo principal del diseño en *HDL*:
     Comando: *analyze -library WORK -format* **extensión {módulo_principal.extensión}**
  4. Analizar los demás módulos del diseño en *HDL*:
     Comando: *analyze -format* **extensión {módulo.extensión}**

5. Elaborar el módulo principal del diseño en *HDL*:
   Comando: *elaborate* **módulo_principal** *-architecture behavioral -library WORK*
6. Enlazar los demás módulos al módulo principal:
   Comando: *link*
7. Escribir el archivo de base de datos sin sintetizar:
   Comando: *write -hierarchy -format ddc -output ./db/****.ddc**
8. Leer el script (*.tcl ubicado en el mismo directorio) que contiene las especificaciones de temporizado:
   Comando: *source* ***_constraints.tcl**
   Algunos de los aspectos a especificar en éste archivo son:
   - 8.1. Frecuencia del reloj de entrada:
     Comando: *create_clock -period* **periodo** [*get_ports* **nombre_reloj**]
   - 8.2. No colocar buffers en la red del reloj:
     Comando: *set_dont_touch_network* [*get_clocks* **nombre_reloj**]
   - 8.3. Retardo entre las diferentes ramificaciones del reloj (Clock Skew):
     Comando: *set_clock_uncertainty -setup* **tiempo(***ns***)** [*get_clocks* **nombre_reloj**]
     Comando: *set_clock_uncertainty -hold* **tiempo(***ns***)** [*get_clocks* **nombre_reloj**]
   - 8.4. Retardo de transición del reloj:
     Comando: *set_clock_transition* **tiempo(***ns***)** [*get_clocks* **nombre_reloj**]
   - 8.5. Retardo de la señal de reloj en la entrada:
     Comando: *set_clock_latency -source* **tiempo(***ns***)** [*get_clocks* **nombre_reloj**]
     Comando: *set_clock_latency* **tiempo(***ns***)** [*get_clocks* **nombre_reloj**]
   - 8.6. Retardo (máximo y mínimo) de todas las señales de entrada, excepto la del reloj:
     Comando: *set_input_delay -max* **tiempo(***ns***)** -clock **nombre_reloj** [*remove_from_collection* [*all_inputs*] [*get_clocks* **nombre_reloj**]]
     Comando: *set_input_delay -min* **tiempo(***ns***)** -clock **nombre_reloj** [*remove_from_collection* [*all_inputs*] [*get_clocks* **nombre_reloj**]]
   - 8.7. Retardo (máximo y mínimo) de todas las señales de salida:
     Comando: *set_output_delay -max* **tiempo(***ns***)** *-clock* **nombre_reloj** [*get_ports* **puertos_salida**]
     Comando: *set_output_delay -min* **tiempo(***ns***)** *-clock* **nombre_reloj** [*get_ports* **puertos_salida**]
   - 8.8. Fanout:
     Comando: *set_max_fanout* **fanout** *$current_design*
   - 8.9. Especificar la celda que maneja todas las entradas (cell driving inputs):
     Comando: *set_driving_cell -lib_cell* **celda_entrada** [*remove_from_collection* [*all_inputs*] [*get_clocks* **nombre_reloj**]]
   - 8.10. Configuración del reloj interno: Comando: *create_generated_clock -name* **nombre_reloj_generado** -divide_by_2 -source [get_ports **nombre_reloj**] [gets_pins **nombre_reloj_generado**]

8.11. No colocar buffers en la red del reloj interno:
Comando: *set_dont_touch_network* [*get_clocks* **nombre_reloj_generado**]

9. Propagar las especificaciones de temporizado a todo el diseño a través de los niveles de jerarquía:
Comando: *propagate_constraints*

10. Revisar el diseño:
Comando: *check_design*

11. Compilar el diseño:
Comando: *compile_ultra*
En caso de utilizar la técnica del bloqueo del reloj (Clock Gating) utilizar:
Comando: *compile_ultra -gate_clock*

12. Escribir el archivo de la lista de nodos a nivel de compuertas (Gate Level Netlist) *.v con los siguientes tres comandos:
Comando: *set verilogout_no_tri true*
Comando: *change_names -hierarchy -rules verilog*
Comando: *write -hierarchy -format verilog -output ./db/***.v**

13. Generar los reportes:
Comando: *report_***opción** *>reports/***.txt**
Donde **opción** puede ser:

13.1. **power**: Estimación del consumo de potencia estático y dinámico.

13.2. **area**: Estimación del área en μm^2.

13.3. **cell**: Contiene las celdas estándar utilizadas en el diseño.

13.4. **qor**: Resumen del área, de las celdas estándar y de las rutas críticas de temporizado.

13.5. **timing**: Rutas críticas de temporizado.

13.6. **port**: Incluye todos los puertos de entrada y salida.

14. Escribir el archivo de base de datos sintetizado:
Comando: *write -hierarchy -format ddc -output ./db/***.ddc**

15. Escribir el archivo con las especificaciones de temporizado:
Comando: *write_sdc ./db/***.sdc**

16. Revisar la configuración de temporizado: Comando: *check_timing*

17. Dentro del directorio *front_end* se ejecuta los siguientes comandos:
    - *dc_shell*:
      Una vez ejecutado este comando, muestra en la consola la biblioteca *link*, biblioteca *target* y la biblioteca *symbol* configuradas.
      Desde este modo es posible ejecutar los comandos necesarios para realizar la *Síntesis RTL*.
    - *start_gui*:
      Levanta la interfaz gráfica del software.

En la figura 4.2 se muestra el software Design Compiler en modo gráfico.

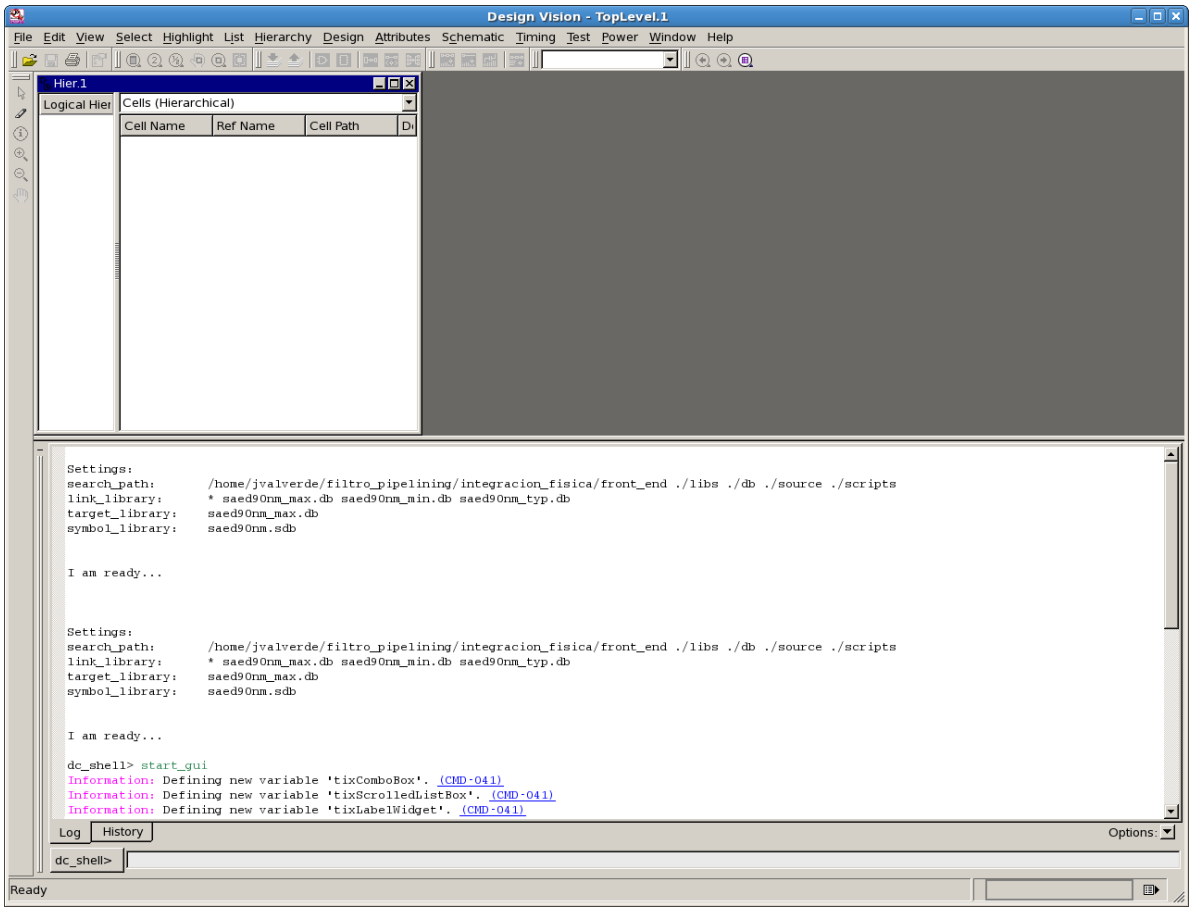

Figura 4.2: Software Design Compiler.

18. Desde la interfaz gráfica de *Design Compiler* se ejecuta el script que contiene todos los anteriores pasos de la siguiente manera: *File/Execute Script* y buscar en el directorio scripts el archivo *detector_syn.tcl*

Cuando la síntesis ha terminado se puede examinar el diagrama esquemático generado, para esto selecciona la celda *detector_secuencia* y luego seleccionar el botón *Create Design Schematic* tal y como se muestra en la figura 4.3 y el circuito generado se aprecia en la figura 4.4

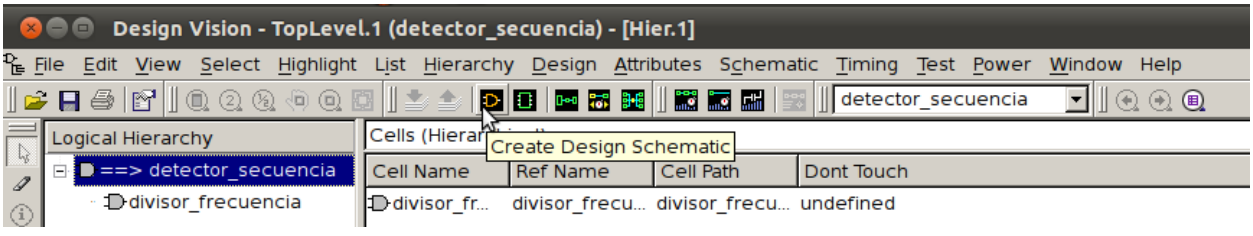

Figura 4.3: Botón para crear esquemático.

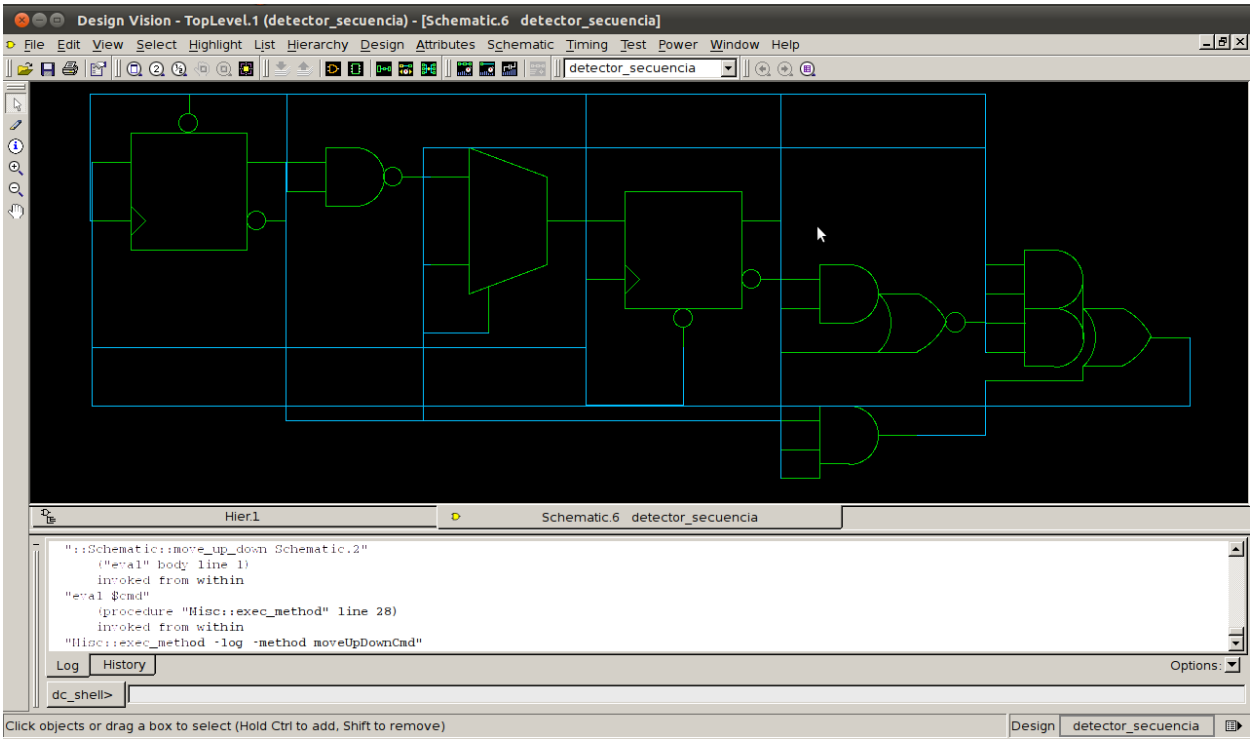

Figura 4.4: Esquemático generado por la síntesis.

Capítulo 5

VCS para la simulación lógica post síntesis RTL

Con la herramienta VCS se realiza la simulación lógica de la lista de nodos a nivel de compuertas (Gate Level Netlist) en las siguientes dos ocasiones:

- Después de realizar la *Síntesis RTL*:
  Se realiza en el directorio *simulacion_logica_sintesis*.
- Después de realizar la *Implementación Física*:
  Se realiza en el directorio *simulacion_logica_layout*.

Para realizar la simulación lógica se requiere de:

1. Las primitivas de la biblioteca de síntesis (*.v): Ya se encuentran en el directorio *simulacion_logica_sintesis* y se llaman *celdas_reducidas.v*

2. La lista de nodos a nivel de compuertas (*.v). Se generó con la síntesis RTL y se encuentran en el directorio */home/**user**/detector/integracion_fisica/front_end/db/* y hay que copiar el archivo *detector_syn.v* en el directorio *simulacion_logica_sintesis*

3. El archivo de simulación (testbench.v). Ya presente en */home/**user**/detector/integracion_fisica/simulacion_logica_sintesis/*

4. Ejecutar el siguiente comando:
   vcs -R -gui **primitivas.v lista de nodos a nivel de compuertas.v testbench.v**

La herramienta posee un ambiente gráfico (ver figura 7.1), en el que se seleccionan las señales a mostrar en la simulación por un tiempo que el usuario crea conveniente para verificar el funcionamiento lógico de la lista de nodos a nivel de compuertas.

- Para correr la simulación lógica del detector, seleccione el módulo testbench en la ventana *Hierarchy* a la izquierda.

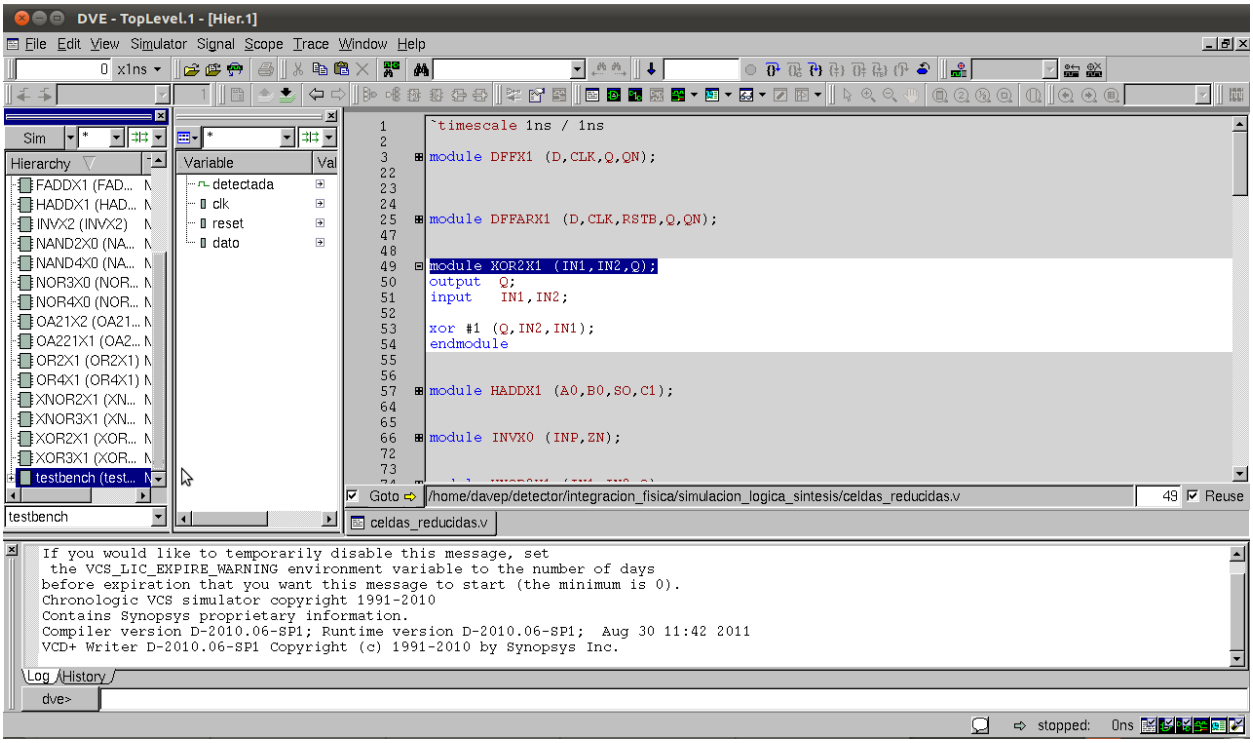

Figura 5.1: Software VCS.

- Se marcan con la ayuda de la tecla control, las señales *detectada, clk, reset y dato* en la ventana *Variable*.
- Hay que agregar estas señales a la simulación para esto se debe hacer click derecho sobre las señales seleccionadas el menú *Add To Waves* y luego *New Wave View*.
- Se abre una nueva ventana donde se presiona el botón *Start/Continue* (botón con flecha azul apuntando hacia abajo). La forma de onda a obtener se muestra en la figura 7.2.

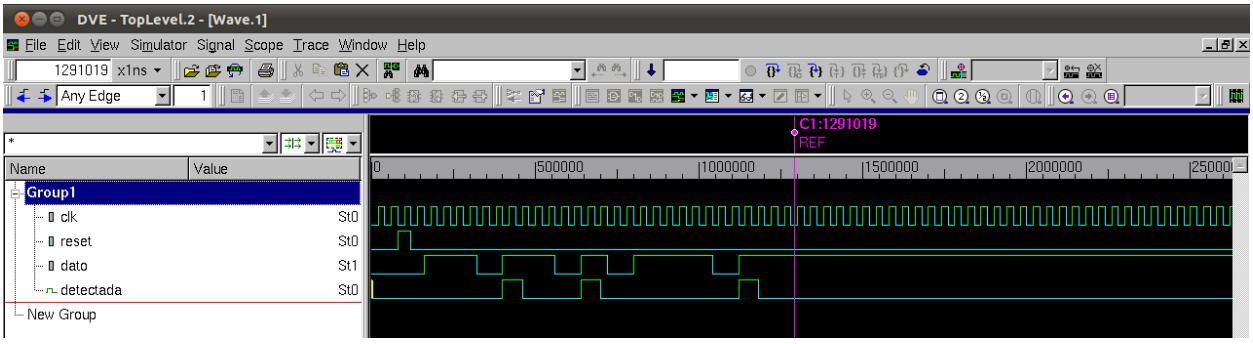

Figura 5.2: Formas de onda de la simulación lógica.

Capítulo 6

IC Compiler para la implementación física

6.1 Estructura del directorio back_end

La estructura del directorio *back_end* se muestra en la figura 6.1 y contiene todos los archivos necesarios para realizar la *Implementación Física* así como los archivos generados durante esta etapa.

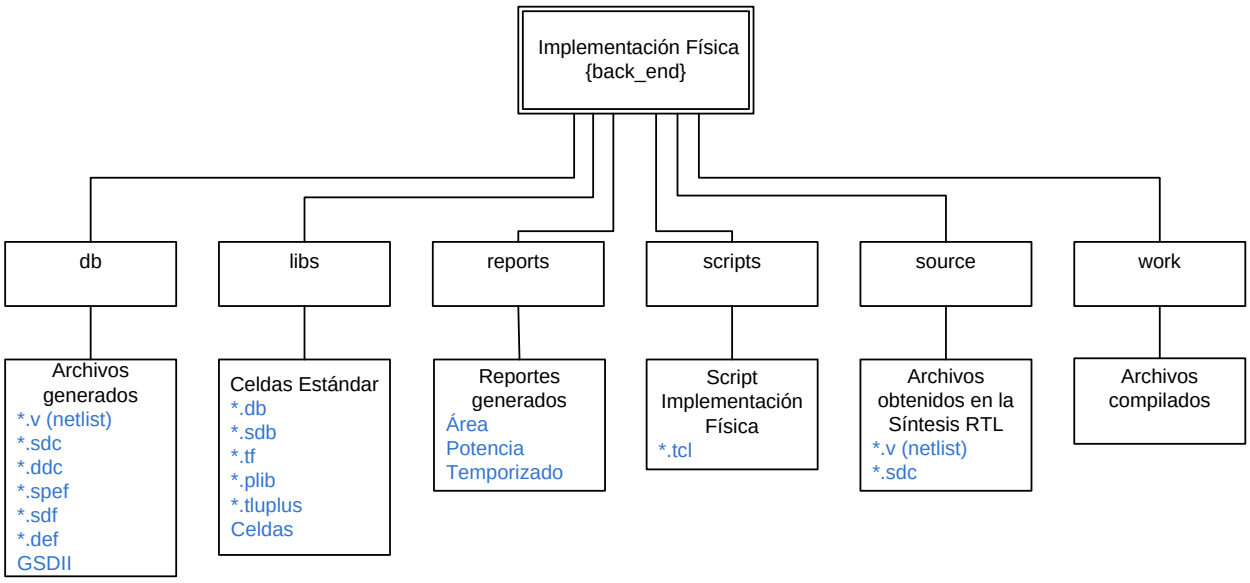

Figura 6.1: Estructura del directorio back_end.

- **db:**
  Los archivos principales son el archivo *.v (gate level netlist), el archivo *.sdc (especificaciones de temporizado) y el archivo *.spef (capacitancias parásitas). Éstos archivos son utilizados en las siguientes etapas del flujo ASIC.
- **libs:**

Contiene la biblioteca de celdas estándar así como su representación gráfica (*.sdb). Además incluye la biblioteca de tecnología (*.tf), la biblioteca física (*.plib) y los archivos *TLUPlus* con la información de las capacitancias parásitas. La configuración de éstas tres bibliotecas se realizan en la sección 6.2.

- **reports:**
  Durante la *Implementación Física* se generan reportes de diferentes aspectos del diseño y se almacenan en archivos de texto.
- **scripts:**
  Contiene el scripts para realizar la Implementación Física (*.tcl).
- **source:**
  Contiene los archivos generados durante la *Síntesis RTL*.
- **work:**
  Contiene los archivos que genera *IC Compiler* durante el proceso de *Implementación*, éstos archivos no se utilizan en las siguientes etapas del diseño.

6.2 Procedimiento para realizar la Implementación Física

El procedimiento descrito en ésta sección corresponde a una forma básica de realizar la *Implementación Física*, los pasos a seguir son los siguientes:

6.2.1 Copia de archivos generados en la síntesis RTL

Se deben copiar los archivos *detector_syn.sdc* y *detector_syn.v* ubicados en el directorio */home/**user**/detector/integracion_fisica/front_end/db/*, a la siguiente ruta */home/**user**/detector/integracion_fisica/back_end/source/*

6.2.2 Revisión de script

Inicialmente se debe corroborar que el archivo *detector_phy.tcl* contenido en el directorio */home/**user**/detector/integracion_fisica/back_end/scripts* esté configurado de la siguiente manera (no olvide cambiar en todas las direcciones dentro del script *jvalverde* por su nombre de usuario):

1. Eliminar diseños previos:
   Comando: *remove_design -designs*

2. Definir VSS y VDD:
   Comando: *set mw_logic0_net VSS*
   Comando: *set mw_logic1_net VDD*

3. Especificar los archivos *TLUplus*:
   Comando: *set_tlu_plus_files -max_tluplus ./libs/tluplus/***Cmax.tluplus** -min_tluplus*

./libs/tluplus/ ***Cmin.tluplus** -tech2itf_map ./libs/tech/**.map**

4. Crear una nueva base de datos Milkway, sólo se hace una vez para cada diseño:
   Comando: *create_mw_lib -technology ./libs/tech/**.tf** -mw_reference_library {./libs/saed90nm_fr/} -bus_naming_style [%d] ./libs/saed90nm_fr/**.mw**

5. Abrir la base de datos Milkway:
   Comando: *open_mw_lib ./libs/saed90nm_fr/**.mw***

6. Importa el Netlist obtenido en la *Síntesis RTL*:
   Comando: *import_designs -format verilog* {***.v**} *-top* **módulo_principal**

7. Resolver múltiples instancias:
   Comando: *uniquify_fp_mw_cel*
   Comando: *link*

8. Leer las especificaciones de temporizado:
   Comando: *read_sdc* {***.sdc**}

9. Iniciar la etapa de descripción física (floorplan):
   Comando: *initialize_floorplan*

10. Conectar los pines de alimentación del circuito integrado:
    Comando: *derive_pg_connection -power_net "VDD" -ground_net "VSS"*
    Comando: *derive_pg_connection -power_net "VDD" -ground_net "VSS" -tie*

11. Crear los anillos de alimentación para VSS y VDD:
    Comando: *create_rectangular_rings -nets {VSS}*
    Comando: *create_rectangular_rings -nets {VDD}*
    Comando: *create_power_strap -nets {VDD}*
    Comando: *create_power_strap -nets {VSS}*

12. Reducir la cantidad de buffers e inversores, sin afectar la calidad del resultado:
    Comando: *set_buffer_opt_strategy -effort low*

13. Iniciar la etapa de colocación física (placement):
    Comando: *create_fp_placement*

14. Guardar el diseño de la colocación física:
    Comando: *save_mw_cel -as* **nombre**

15. Iniciar la etapa del enrutado físico (routing):
    Comando: *route_zrt_auto -max_detail_route_iterations* **10**

16. Verificar el enrutado físico:
    Comando: *verify_zrt_route*

17. Extraer el netlist:
    Comando: *write_verilog ./db/***.v**

18. Extraer las capacitancias parásitas:
    Comando: *write_parasitics -output* {*./db/***.spef**}

19. Escribir el archivo con los datos de retado del layout:
    Comando: *write_sdf ./db/***.sdf**

20. Extraer el archivo con las especificaciones de temporizado:
    Comando: *write_sdc ./db/***.sdc**

21. Escribir el archivo .ddc y el archivo .def:
    Comando: *write -format ddc -output* {*./db/***.ddc**}
    Comando: *write_def -output* **"*.def"**

22. Crear el archivo GDSII:
    Comando: *write_stream -format gds -lib_name ./libs/saed90nm_fr/***.mw** *-cells* {*detector_routing*} *./db/***nombre_GDSII**

23. Guardar el diseño después del enrutado físico:
    Comando: *save_mw_cel -as* **nombre**

24. Generar los reportes:
    Comando: *report_***opción** *>reports/nombre_archivo.txt*
    Donde **opción** puede ser:

    24.1. **power**: Estimación del consumo de potencia estático y dinámico.

    24.2. **qor**: Resumen del área, de las celdas estándar y de las rutas críticas de temporizado.

25. Se debe verificar que para correr el script por primera vez la linea 44 (*create_mw_lib...*) NO debe estar comentada. Y debe comentarse si se va a ejecutar este por segunda vez.

6.3 Ejecutar IC Compiler

Dentro del directorio *back_end* se ejecuta los siguientes comandos:

- *icc_shell*:
  Una vez ejecutado este comando, muestra en la consola la biblioteca *link*, biblioteca *target* y la biblioteca *symbol* configuradas. Desde este modo es posible ejecutar los comandos necesarios para realizar la *Implementación*.
- *start_gui*:
  Levanta la interfaz gráfica del software (ver figura 6.2).

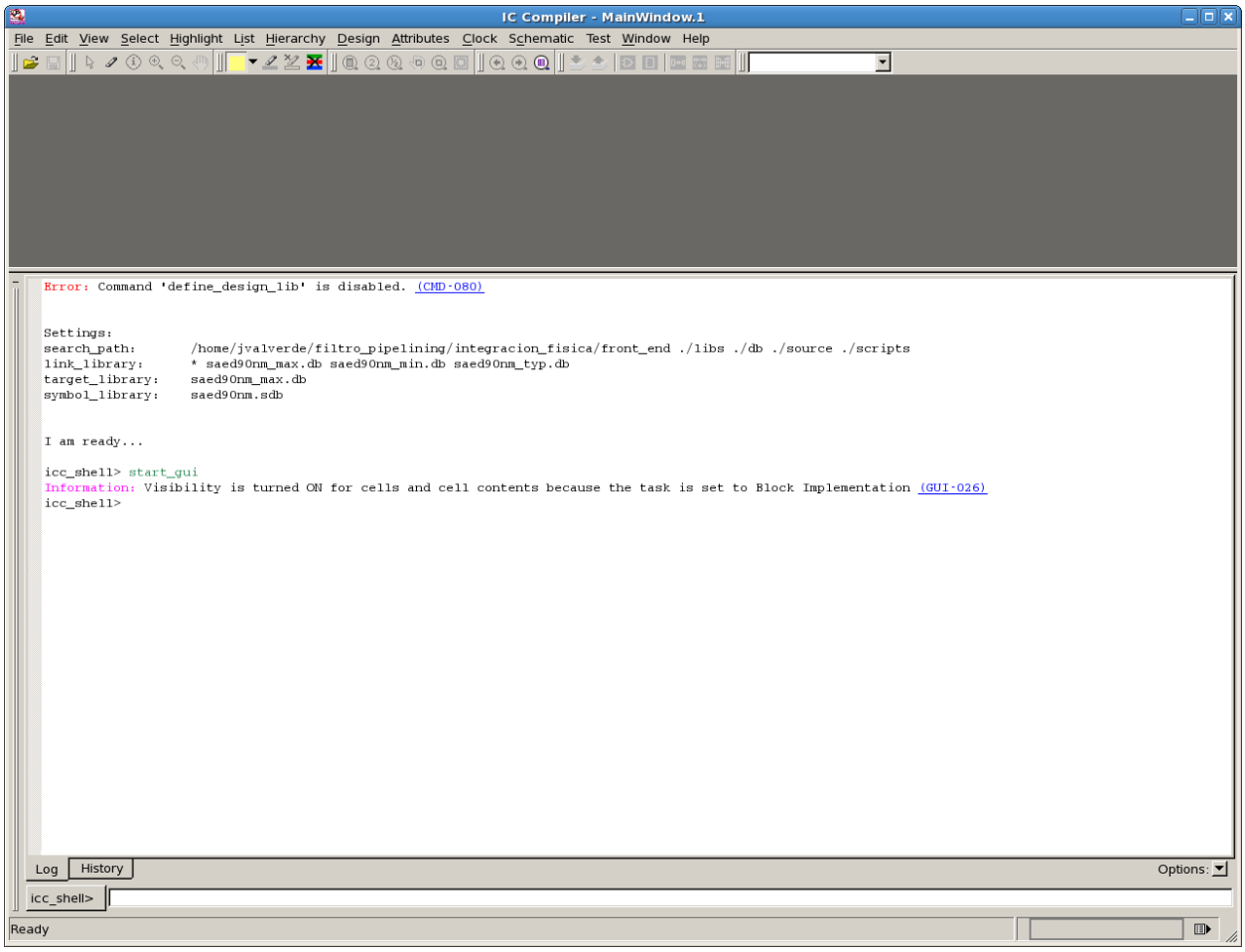

Figura 6.2: Software IC Compiler.

Desde la interfaz gráfica de *IC Compiler* se ejecuta el script que contiene todos los anteriores pasos yendo a *File/Execute Script...*.

Si todo es correcto se debe abrir una ventana con la implementación física con celdas estandar según la figura 6.3

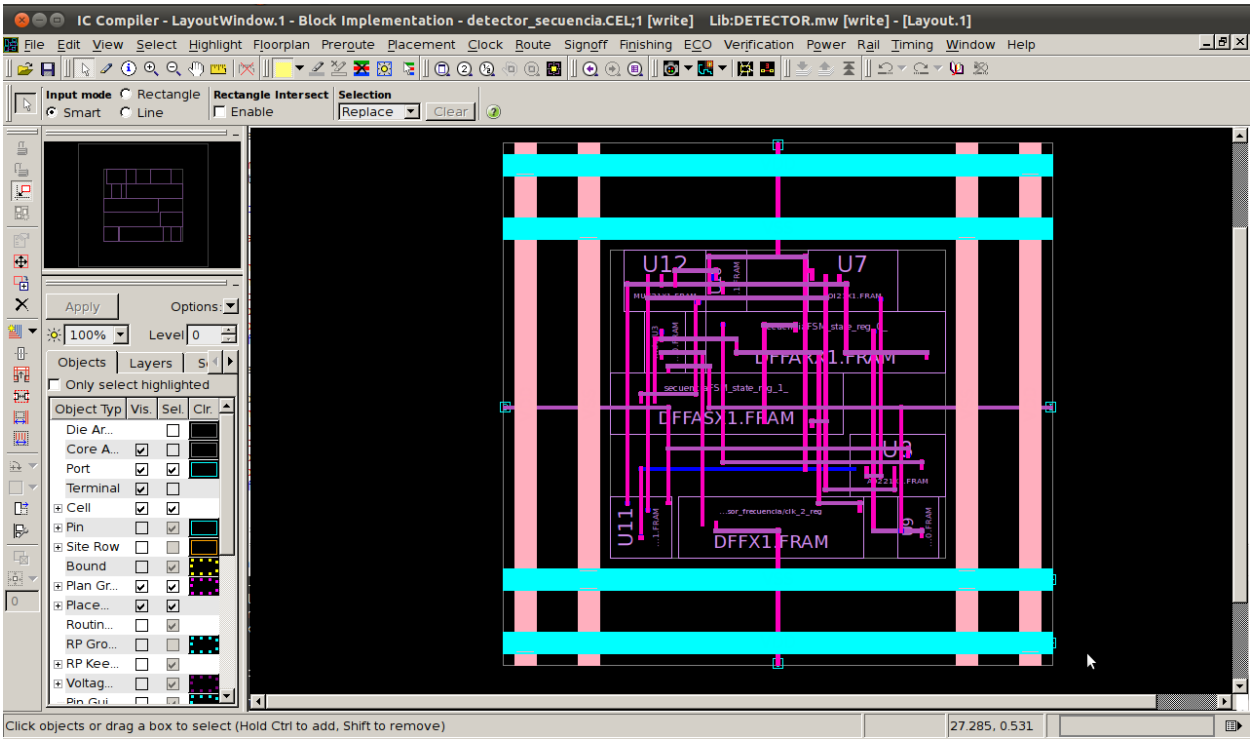

Figura 6.3: Implementación física obtenida con el IC COmpiler.

Capítulo 7

VCS para la simulación lógica post implementación física

Con la herramienta VCS se realiza la simulación lógica de la lista de nodos a nivel de compuertas (Gate Level Netlist) en las siguientes dos ocasiones:

- Después de realizar la *Síntesis RTL*:
  Se realiza en el directorio *simulacion_logica_sintesis*.
- Después de realizar la *Implementación Física*:
  Se realiza en el directorio *simulacion_logica_layout*.

Para realizar la simulación lógica se requiere de:

1. Las primitivas de la biblioteca de síntesis (*.v): Ya se encuentran en el directorio *simulacion_logica_layout* y se llaman *celdas_reducidas.v*

2. La lista de nodos a nivel de compuertas (*.v). Se generó con la implementación física y se encuentran en el directorio */home/**user**/detector/integracion_fisica/back_end/db/* y hay que copiar el archivo *detector_phy.v* en el directorio *simulacion_logica_layout*

3. El archivo de simulación (testbench.v). Ya presente en */home/**user**/detector/integracion_fisica/simulacion_logica_layout/*

4. Ejecutar el siguiente comando:
   vcs -R -gui **primitivas.v lista de nodos a nivel de compuertas.v testbench.v**

La herramienta posee un ambiente gráfico (ver figura 7.1), en el que se seleccionan las señales a mostrar en la simulación por un tiempo que el usuario crea conveniente para verificar el funcionamiento lógico de la lista de nodos a nivel de compuertas.

- Para correr la simulación lógica del detector, seleccione el módulo testbench en la ventana *Hierarchy* a la izquierda.

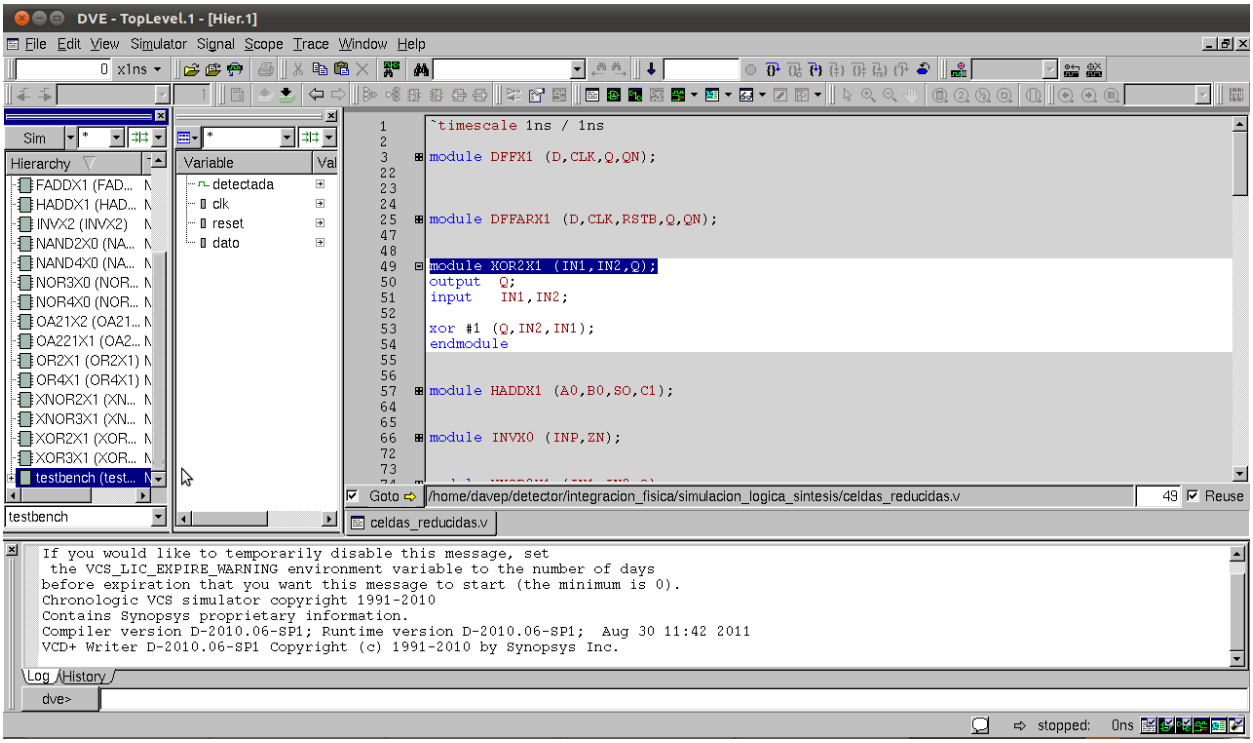

Figura 7.1: Software VCS.

- Se marcan con la ayuda de la tecla control, las señales *detectada, clk, reset y dato* en la ventana *Variable*.
- Hay que agregar estas señales a la simulación para esto se debe hacer click derecho sobre las señales seleccionadas el menú *Add To Waves* y luego *New Wave View*.
- Se abre una nueva ventana donde se presiona el botón *Start/Continue* (botón con flecha azul apuntando hacia abajo). La forma de onda a obtener se muestra en la figura 7.2.

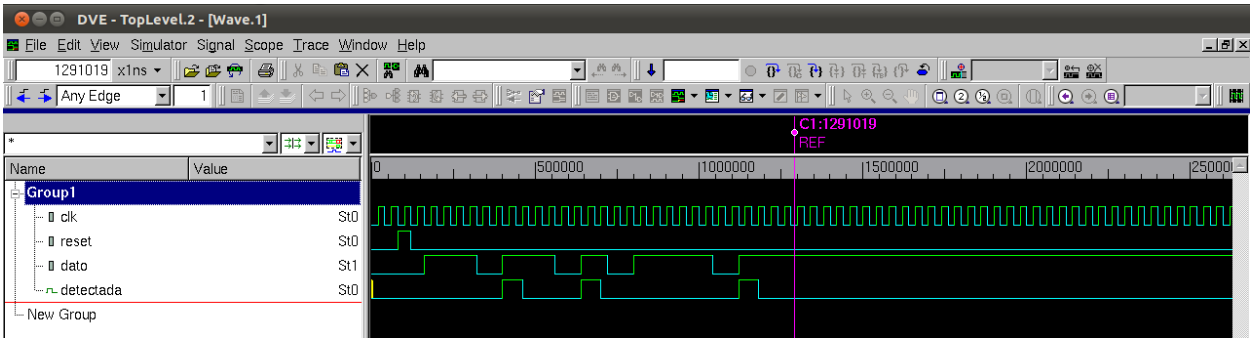

Figura 7.2: Formas de onda de la simulación lógica.

Capítulo 8

Primetime para verificación de temporizado

8.1 Estructura del directorio temporizado_layout

La estructura del directorio *temporizado_layout* se muestra en la figura 8.1 y contiene todos los archivos necesarios para realizar la *Verificación de Temporizado del Layout* así como los reportes generados.

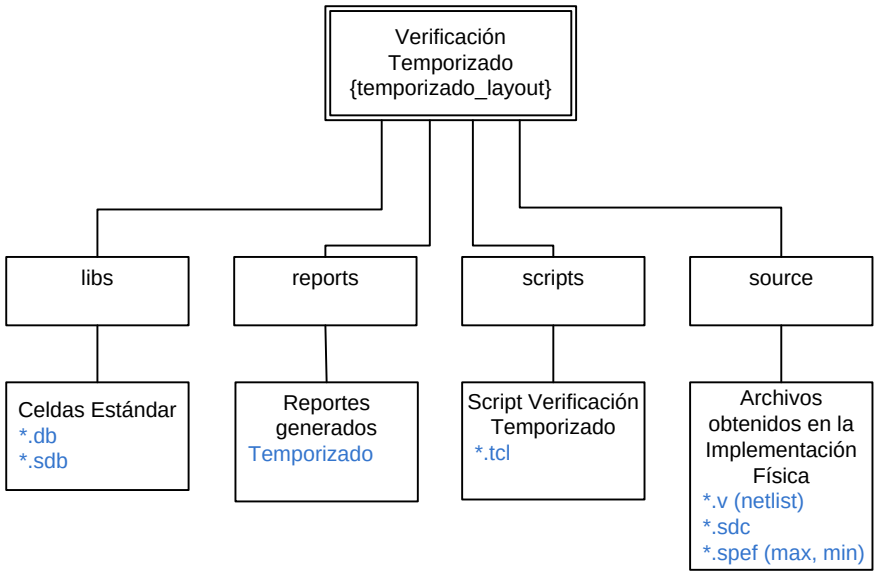

Figura 8.1: Estructura del directorio temporizado_layout.

8.2 Procedimiento para realizar la Verificación de Temporizado

El procedimiento descrito en ésta sección corresponde a una forma básica de realizar la *Verificación de Temporizado*, inicialmente se debe corroborar que el archivo *detector_tim.tcl* contenido en el directorio */home/**user**/detector/integracion_fisica/temporizado_layout/scripts/* esté configurado de la siguiente manera (no olvide cambiar en todas las direcciones dentro del script *jvalverde* por su nombre de usuario):

1. Definir la ruta del directorio *temporizado_layout* con el comando *set_search_path* {**ruta**}.
2. Agregar a la ruta los directorios pertenecientes a *temporizado_layout* con el comando *set_search_path* {**"$search_path ./libs ./scripts ./source"**}.
3. Establecer la biblioteca *link* con el comando *set link_library* {***.db**}.
4. Establecer la biblioteca *target* con el comando *set target_library* {***.db**}.
5. Leer el Netlist obtenido en la *Implementación Física*:
   Comando: *read_verilog* ***.v**
6. Definir el módulo principal:
   Comando: *current_design* **nombre_módulo**
7. Leer las capacitancias parásitas máximas:
   Comando: *read_parasitics -format SPEF* ***.spef.max**
8. Leer las especificaciones de temporizado:
   Comando: *read_sdc* ***.sdc**
9. Generar reporte de temporizado desde las entradas hasta los registros:
   Comando: *report_timing -from [all_inputs] -to [all_registers -data_pins] -max_paths* **40** *> reports/****.txt**
10. Generar reporte de temporizado de registros a registros:
    Comando: *report_timing -from [all_register -clock_pins] -to [all_registers -data_pins] -max_paths* **40** *> reports/****.txt**
11. Generar reporte de temporizado desde los registros hasta las salidas:
    Comando: *report_timing -from [all_register -clock_pins] -to [all_outputs] -max_paths* **40** *> reports/****.txt**
12. Leer las capacitancias parásitas mínimas:
    Comando: *read_parasitics -format SPEF* ***.spef.min**
13. Generar el reporte de temporizado de registro a registro con el tiempo de transición y la capacitancia:
    Comando: *report_timing -transition_time -capacitance -nets -input_pins -from [all_registers -clock_pins] -to [all_registers -data_pins] > reports/* ***.txt**

8.3 Ejecutar Primetime

Se deben copiar los archivos *detector_phy.sdc detector_phy.spef.max detector_phy.spef.min detector_phy.v* ubicados en el directorio */home/**user**/detector/integracion_fisica/back_end/db/*, a la siguiente ruta */home/**user**/detector/integracion_fisica/temporizado_layout/source/*

Dentro del directorio *temporizado_layout* se ejecuta el siguiente comando:

- *primetime*:
  Una vez ejecutado este comando, muestra en la consola la biblioteca *link* y la biblioteca *target* configuradas. El software se ejecuta en modo gráfico, tal como se muestra en la figura 8.2.

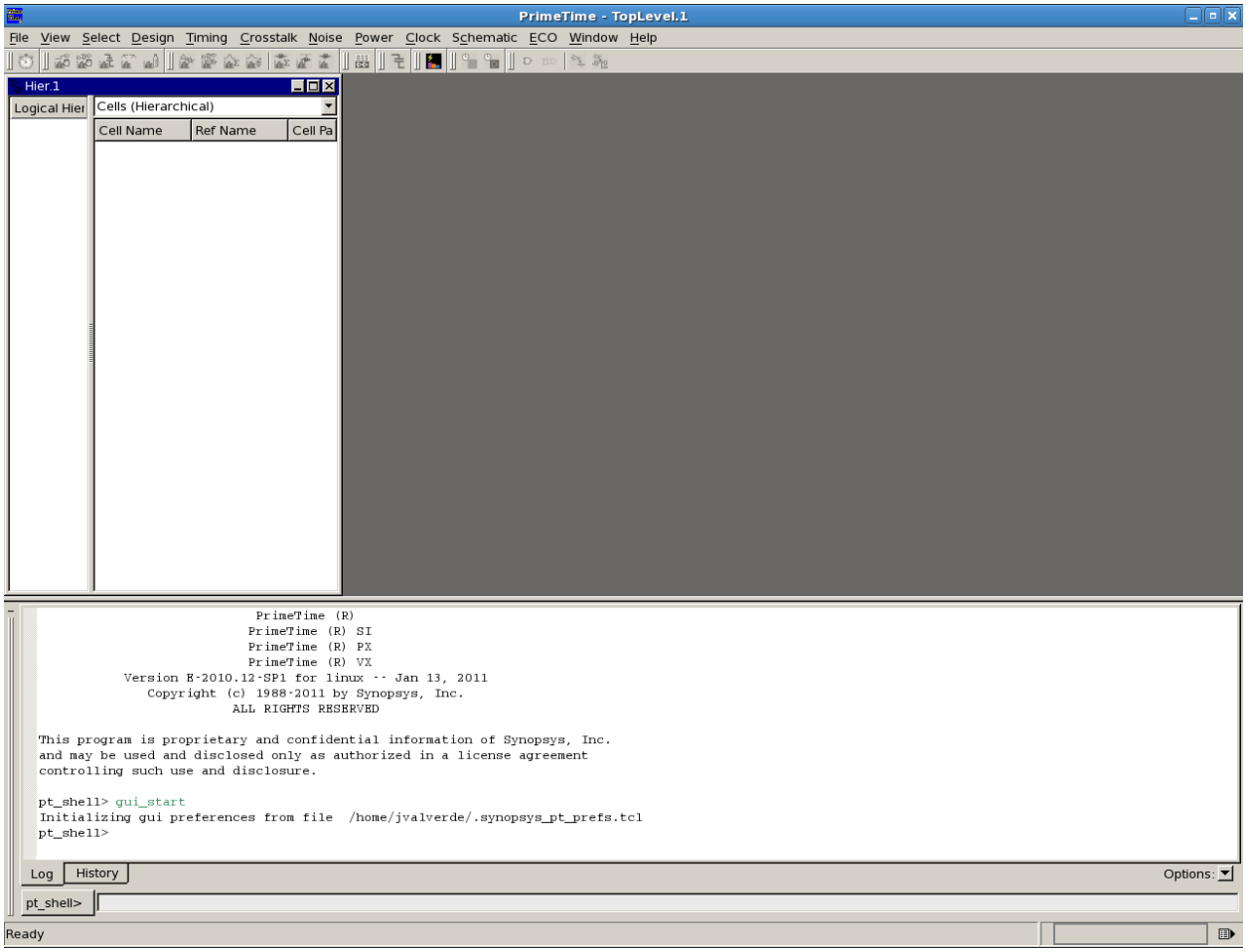

Figura 8.2: Software Primetime.

Desde *Primetime* ejecutar el script que contiene todos los anteriores pasos yendo a *File/ Execute Script*. Esto genera los reportes en el directorio */home/**user**/detector/integracion_fisica/temporizado_layout/reports/*, especificamente el archivo *criticas_tran_cap.txt*

Recordar que si los valores de *slack (MET)* son positivos, se cumplen las especificaciones de temporizado.

Capítulo 9

Ejercicio de Diseño

9.1 Ejercicio del Sistema Estimador de delay entre dos señales de entrada

El sistema brindado es un estimador de delay entre dos señales de entrada x1 y x2 para mostrar el valor estimado en la salida *Y_output_le*, el ejercicio consta en realizar la integración física del sistema digital comenzando desde la similación lógica pre-síntesis (en caso de existir un error, debe corregirlo) para ello se suministra:

1. La descripción de hardware del sistema (archivos *.v) así como el testbench utilizado en la simulación lógica ubicados en */home/tutorial_synopsys/estimador/*

2. Las siguientes especificaciones de temporizado:

- Periodo del reloj (clk) de 5000 ns, con un clock skew de 0.4 ns, un tiempo de transición de 0.1 ns, y un retardo de 0.1 ns entre la fuente del reloj y la entrada al chip.

- Retardo máximo de 200 ns y mínimo de 100 ns en las señales de entrada (excepto en la señal de reloj).

- Retardo máximo de 400 ns y mínimo de 200 ns en las señales de salida.

3. Fanout de 10.

Para las simulación lógica pre-síntesis se debe seleccionar la base de tiempo de 100 ns dentro del software Questasim, para ejecutarlo use el comando *qhsim*.

Para las simulaciones seleccione únicamente las señales *clk, reset_L, x1, x2, Y_output_L, tof_count, retardo y ret_teor*. El resultado de la simulación se muestra en la figura 9.1

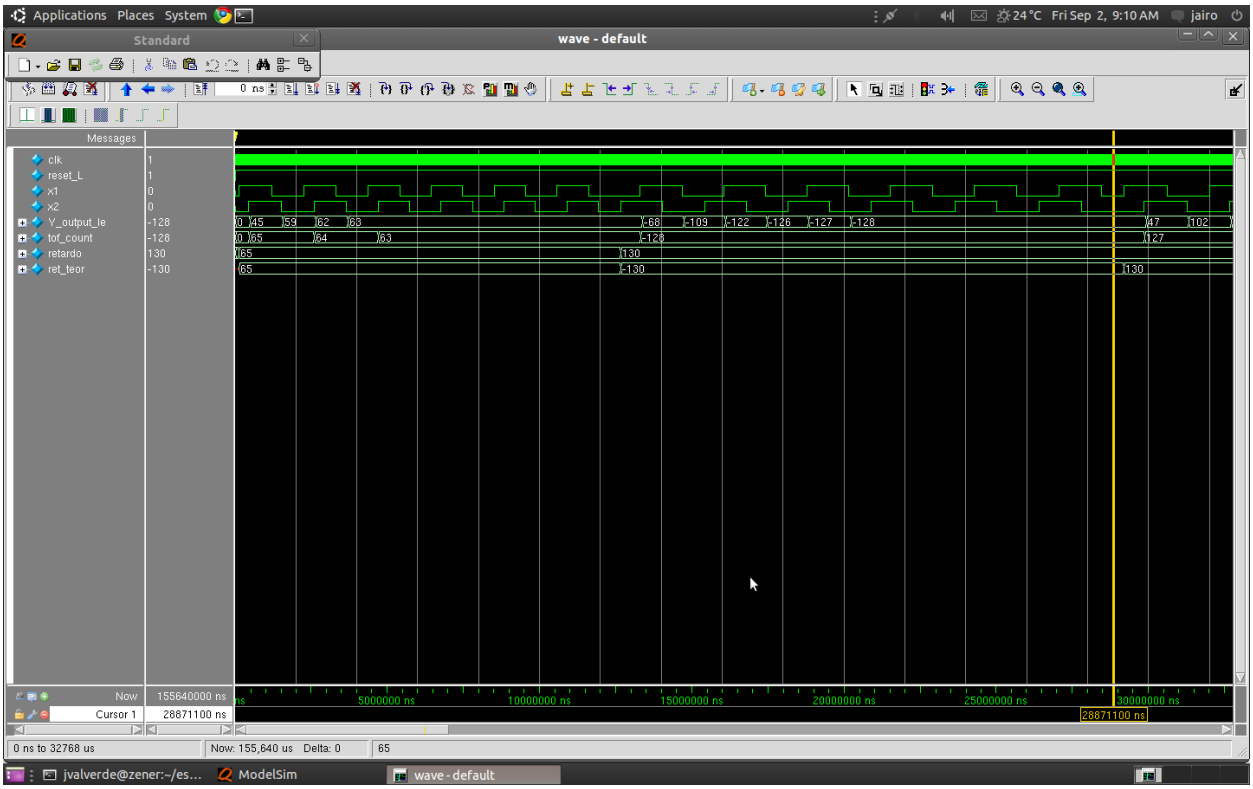

Figura 9.1: Resultados de la simulación lógica pre-síntesis del sistema estimador de delay.

Bibliografía

[1] H. Bindu and H. Mahmoodi. *ASIC Design Flow Tutorial using Synopsys Tools*. San Francisco, CA, 2009. 1, 3

[2] A. Chacón. *Circuitos Integrados de bajo consumo para detección y localización de disparos de armas de fuego*. Mar del Plata, Argentina, 2009. 1

Índice alfabético