Práctica de Laboratorio N.º 2

“Dispositivos Lógico Programables”

Alumno:

* Dante Agustín Cecchetti

Asignatura:

* Técnicas y Dispositivos Digitales 2

Departamento de Ingeniería Electrónica y Computación

Área Digitales

Facultad de Ingeniería

Universidad Nacional de Mar del Plata

**Fecha de realización: 18/11/2022**

**Fecha de entrega: 25/11/2022**

**1)Identifique qué elementos constituyen los LEs de la FPGA Cyclone III a emplear en el laboratorio y qué estructura tienen las LABs.**

***L****ogic* ***E****lements* (LEs):

Los LEs están constituidos por:

* + Lookup table (LUT) de 4 entradas.
  + Registro programable.
  + Conexiones IN/OUT de carry en cadena.
  + Conexión a ‘cadena de registros’ (permite que todas las LEs de una LAB operen en cascada, y el uso de los registros como shift register).
  + Multiplexores.
  + Conexiones:
    - Locales.
    - En columna.
    - En fila.
    - Directas.

***L****ogic* ***A****rray* ***B****lock* (LABs):

Las LABs están compuestas por:

* + 16 LEs interconectadas localmente.
  + Bloque MK9 de SRAM (256x36) que puede ser configurado como
    - True dual port.
    - Dual port
    - Single port
      * 8192x1
      * 4096x1
      * 2000x4
      * 1024x9
      * 512x18
      * 256x36.

la SRAM puede ser usada como ROM, shift register, FIFO

* + Multiplicadores embebidos.

Las LABs generan conexiones entre ellas a través de una matriz de interconexión.

**2) ¿De qué se trata el Nios® II?**

El Nios® II es un procesador programado con los elementos de la FPGA (Flip flops, RAM, ROM, ALUs), con una estructura tipo RISC (reduced instruction set) de 32 bits. Al ser programado se lo denomina como un procesador softcore, una ventaja de que sea programado es la posibilidad de generar “custom cores”.

**3) ¿Qué diferencia existe entre IP Cores y los bloques embebidos (ej multiplicador embebido) disponibles en la FPGA?**

Los IPcore son bloques pre armados, preprogramados, que permiten su utilización en diseños más grandes, estos bloques son parametrizados para adaptarse al diseño que se quiere implementar.

Un ejemplo de IPcore es la función FFT de la parte de DSP IPcores, este bloque ya está preprogramado, no hay una sección de la FPGA que se encarga exclusivamente de hacer FFT, este IPcore genera las conexiones necesarias para que la FFT se calcule.

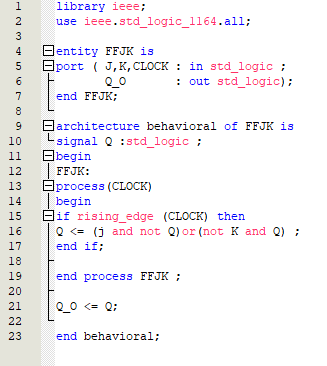
En resumen, un IPcore es un “Plano” o instrucción de cómo construir un bloque

lógico. Por otro lado los bloques embebidos están implementados y optimizados para hacer una tarea específica , pueden ser “programados”, pero con limitaciones, por ejemplo un multiplicador de 8 entradas puede ser concatenado con otro para hacer uno de 16 entradas programando las conexiones, pero va a seguir siendo un multiplicador, no se puede programar para hacer otras operaciones, estos bloques ya están implementados en el silicio, al estar ya implementados en el silicio mejoran la eficiencia y velocidad de operación.

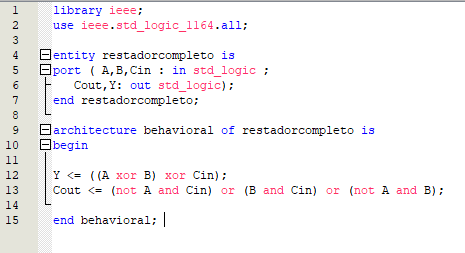
**4) ¿Qué tipo de celda de programación posee el dispositivo FPGA Cyclone III?**

La FPGA Cyclone III utiliza celdas SRAM como celdas de programación.

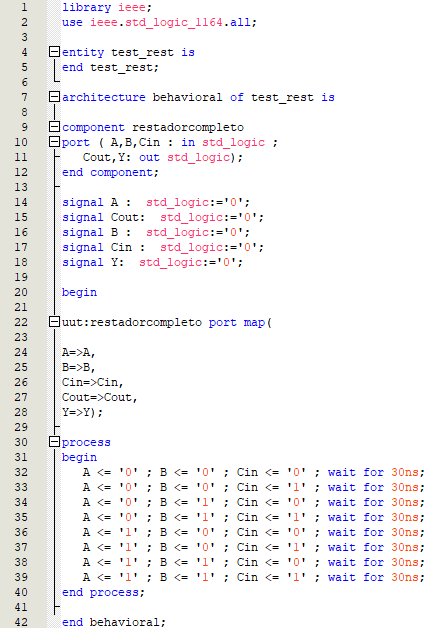
**5) Realice la descripción en VHDL de un Flip Flop JK.**



**6) Realice la descripción en VHDL de un restador completo de un bit.**



**7) Realice la descripción en VHDL del test bench del restador completo de un bit.**

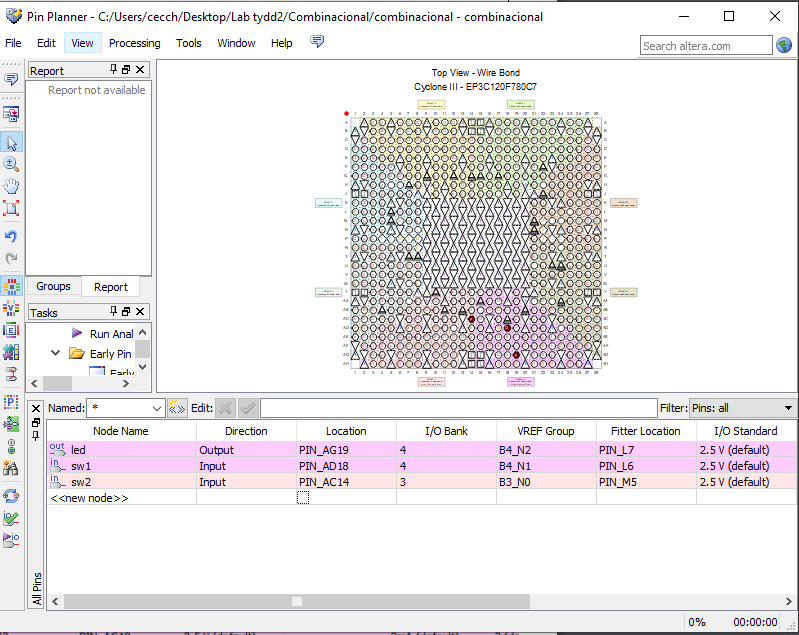


***PARTE A: Implementación de un circuito combinacional en FPGA***

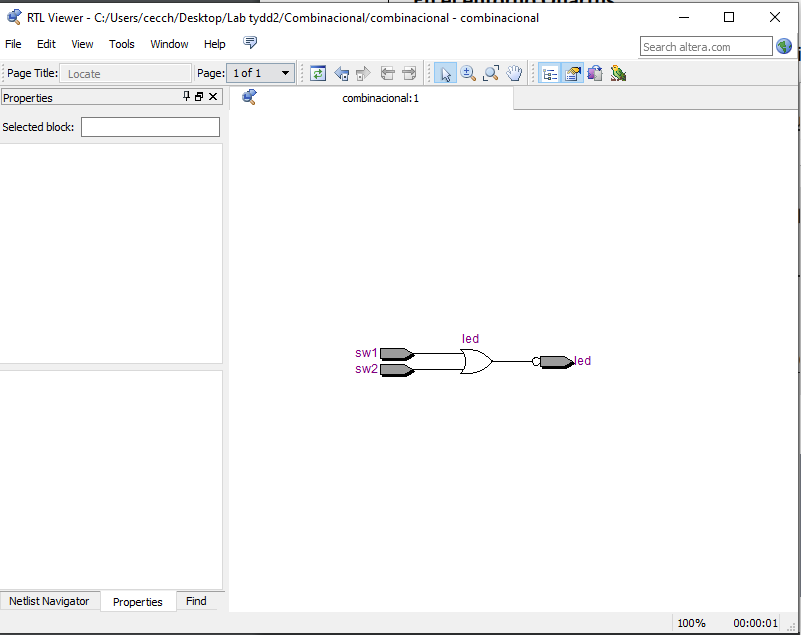
1. En un archivo VHDL describa el circuito combinacional mostrado:



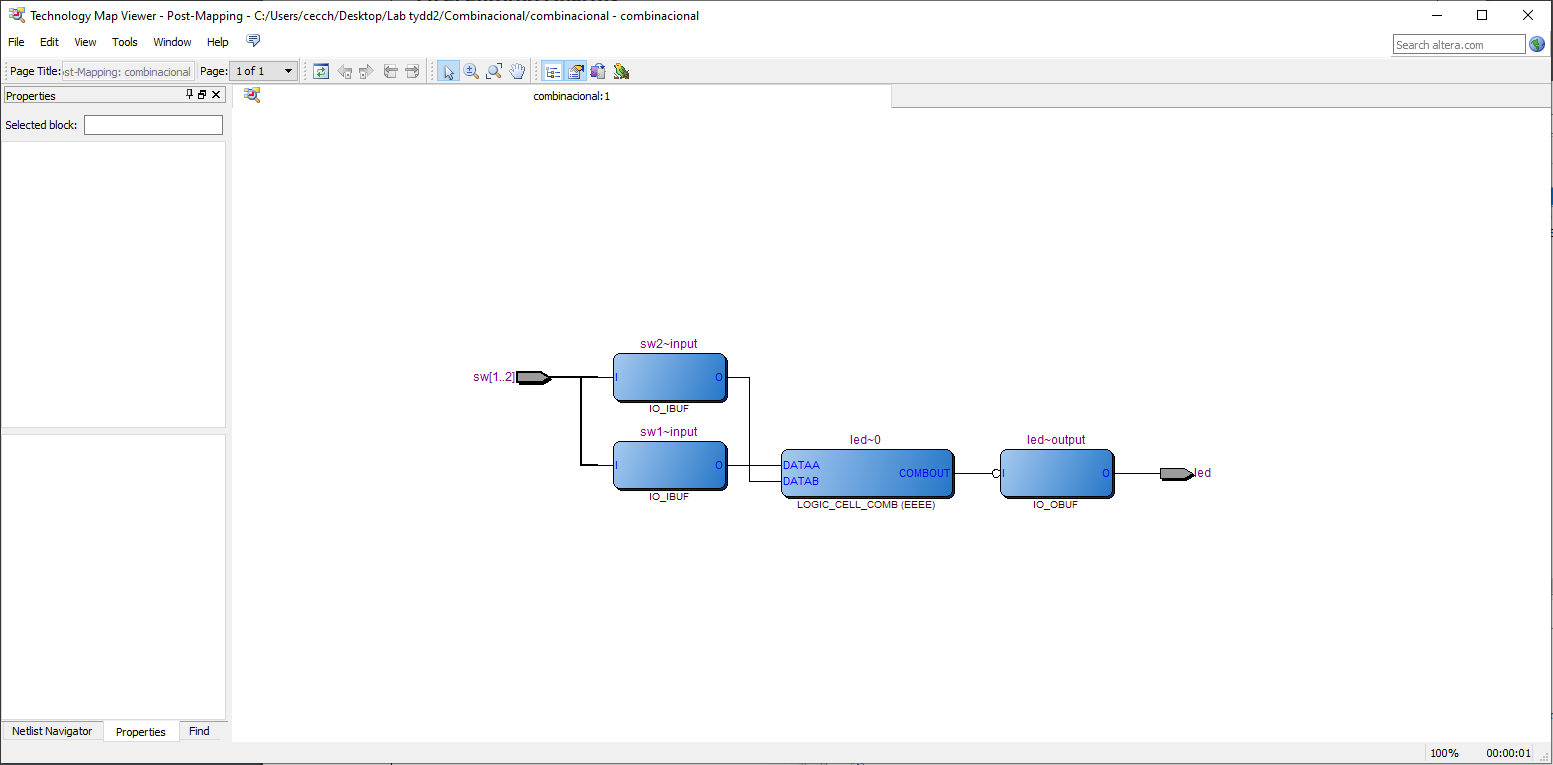
1. Asigne los pines de entrada y salida del diseño



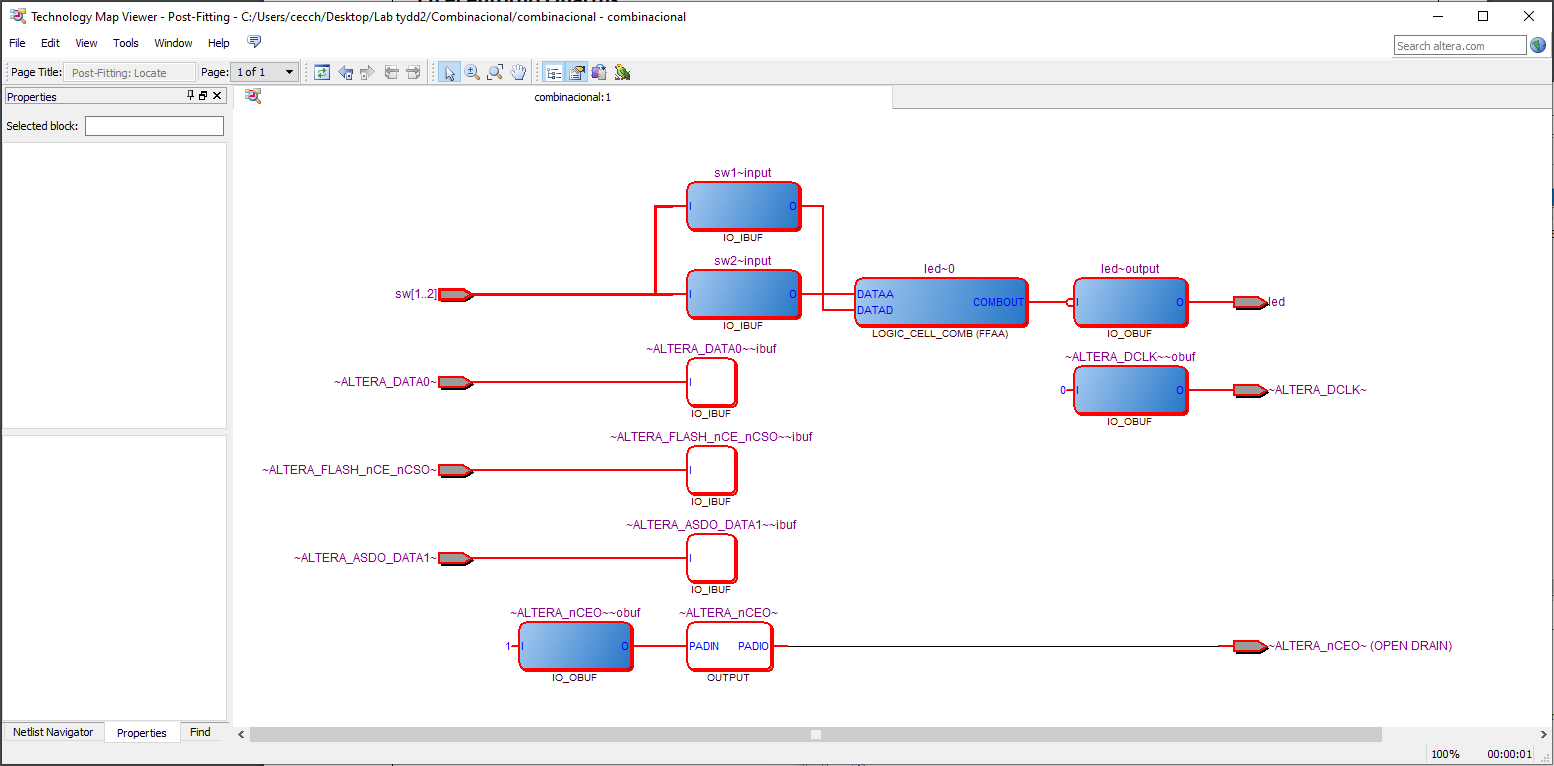
1. Verifique el circuito implementado mediante el visor de RTL



1. Verifique el circuito implementado luego del mapeo en el dispositivo.

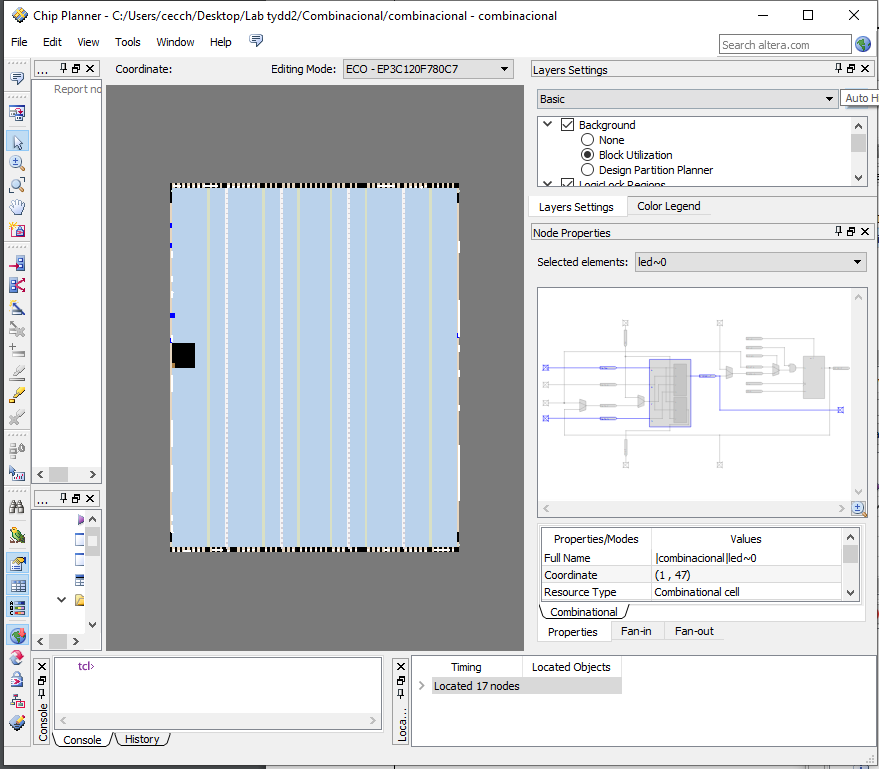


Map Viewer-Post Mapping

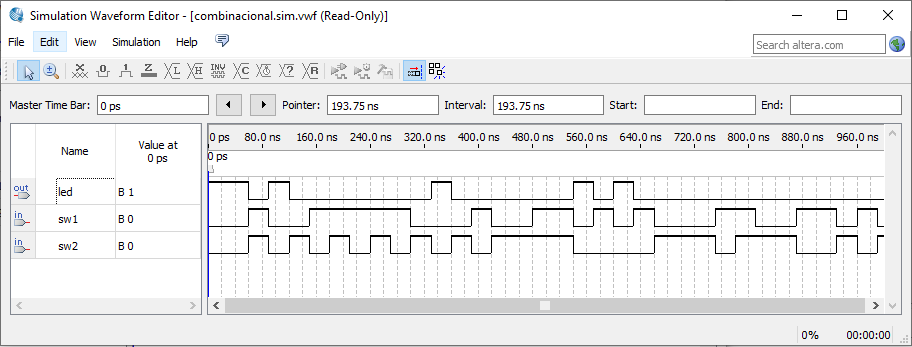


Map Viewer-Post fittiing

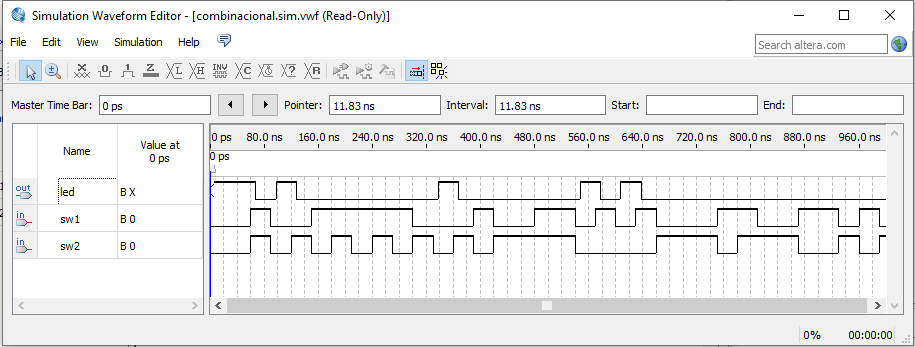
1. Verifique la implementación en el chip



1. Realice la simulación funcional y temporal del circuito mediante el simulador de Quartus.

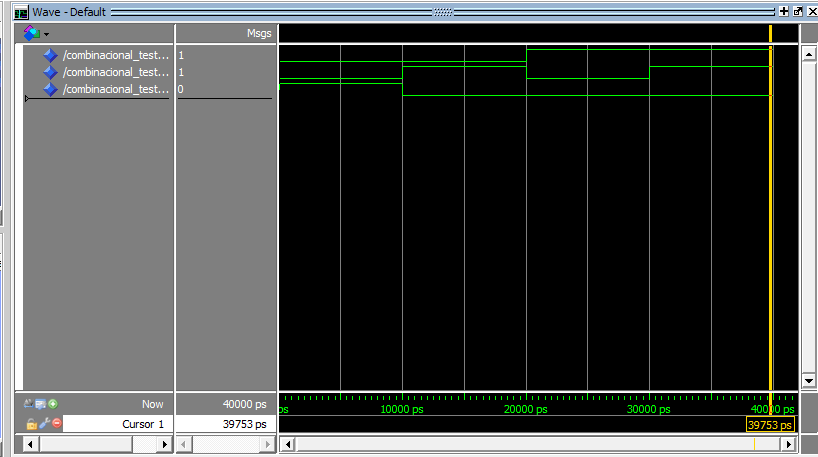


Simulación con Run Functional Simulation.

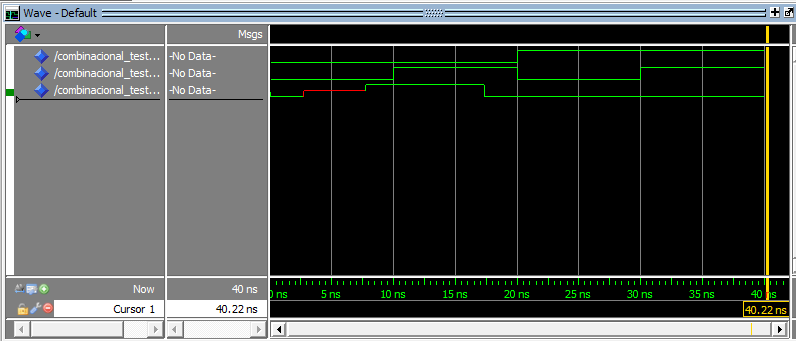


Simulación con Run Timing Simulation.

1. Realice la simulación del circuito mediante la simulación de Modelsim.

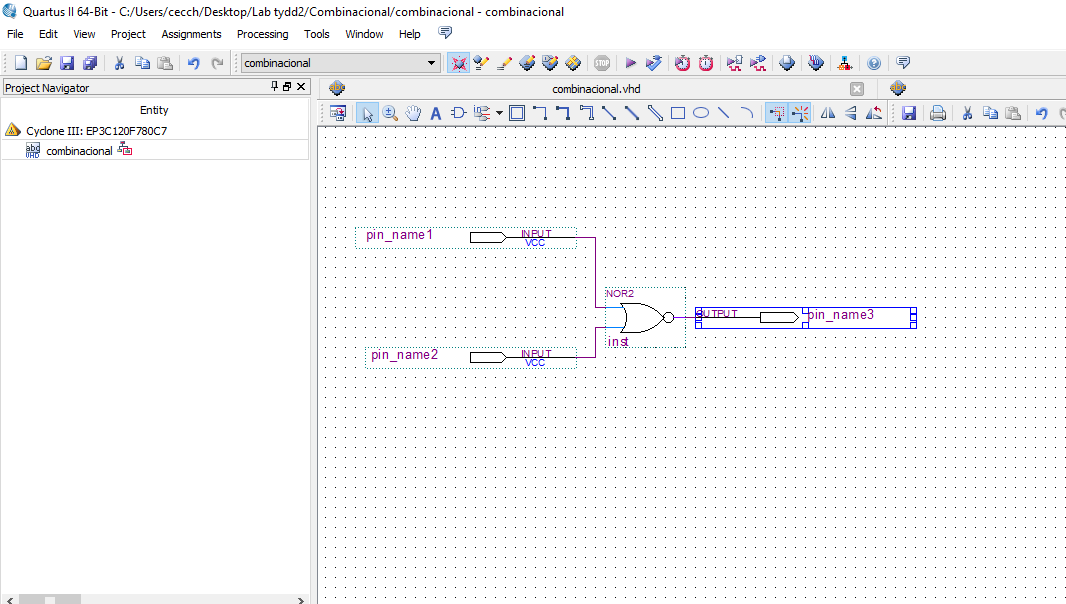


Simulación con RTL Simulation.



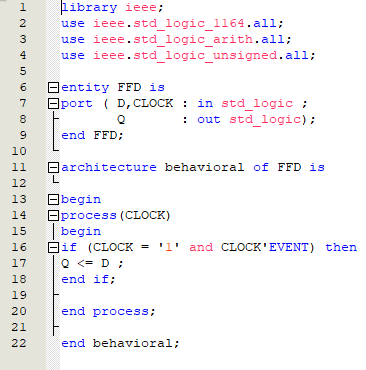
Simulación con Gate Level Simulation.

1. Genere un archivo esquemático

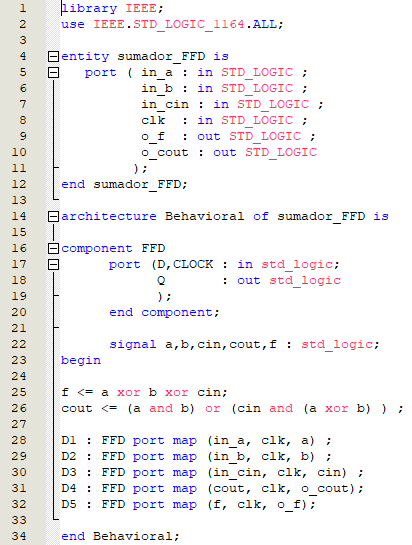


***PARTE B: Implementación de un sumador completo con salidas registradas en VHDL***

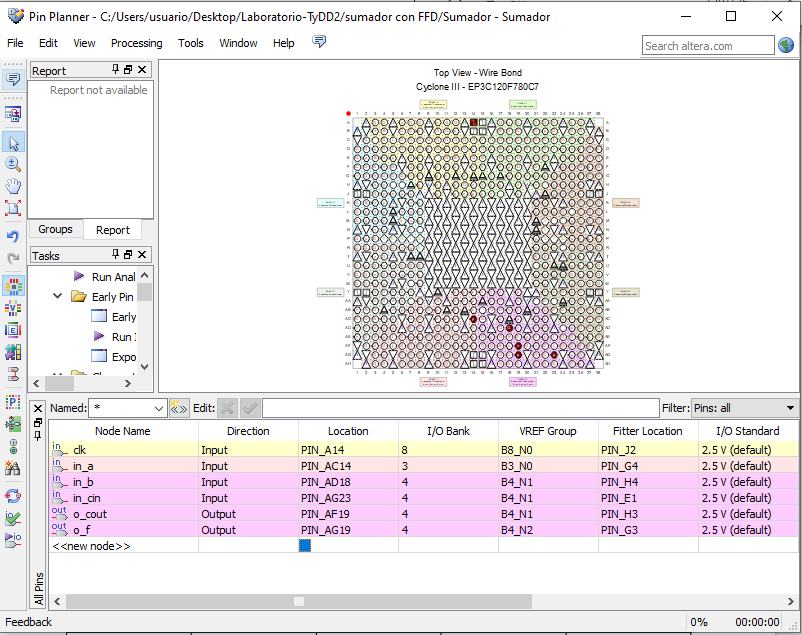
1. Genere un archivo VHDL en el cual describa un Flip Flop D



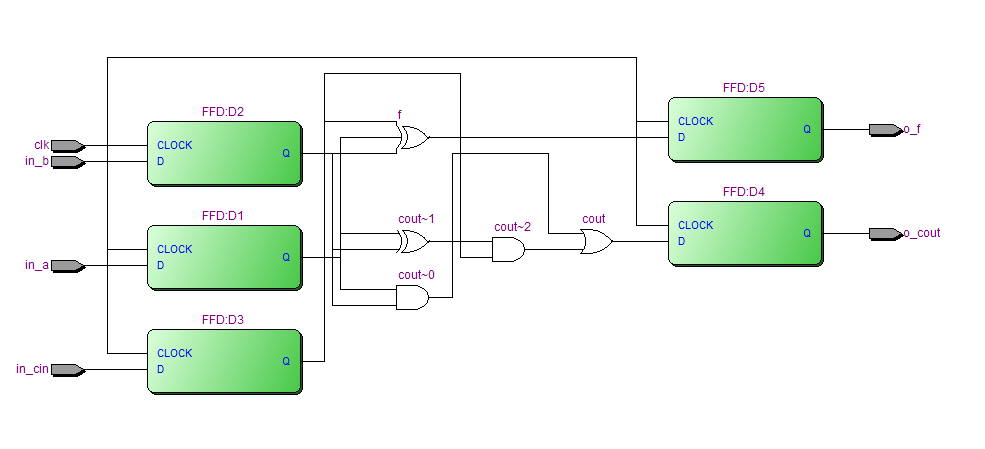
1. Genere un archivo VHDL en el cual describa el circuito sumador completo



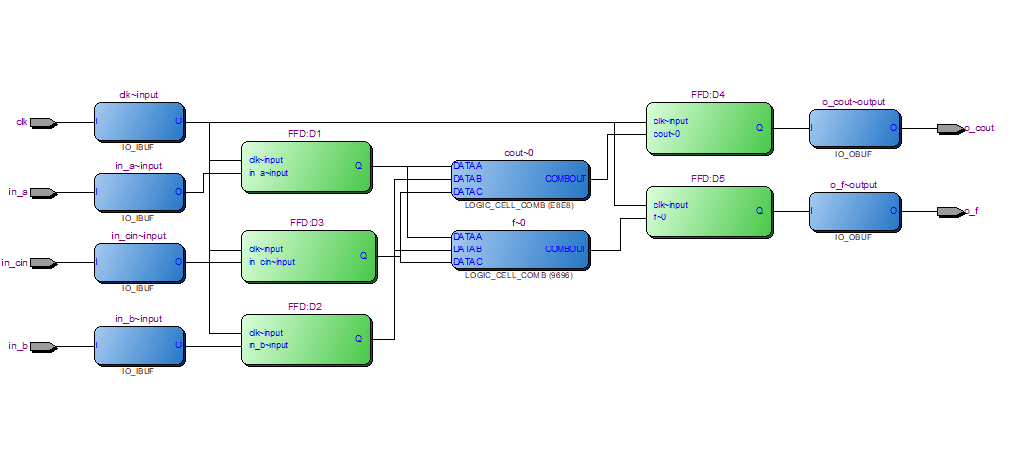
1. Asigne los pines de entrada y salida del diseño



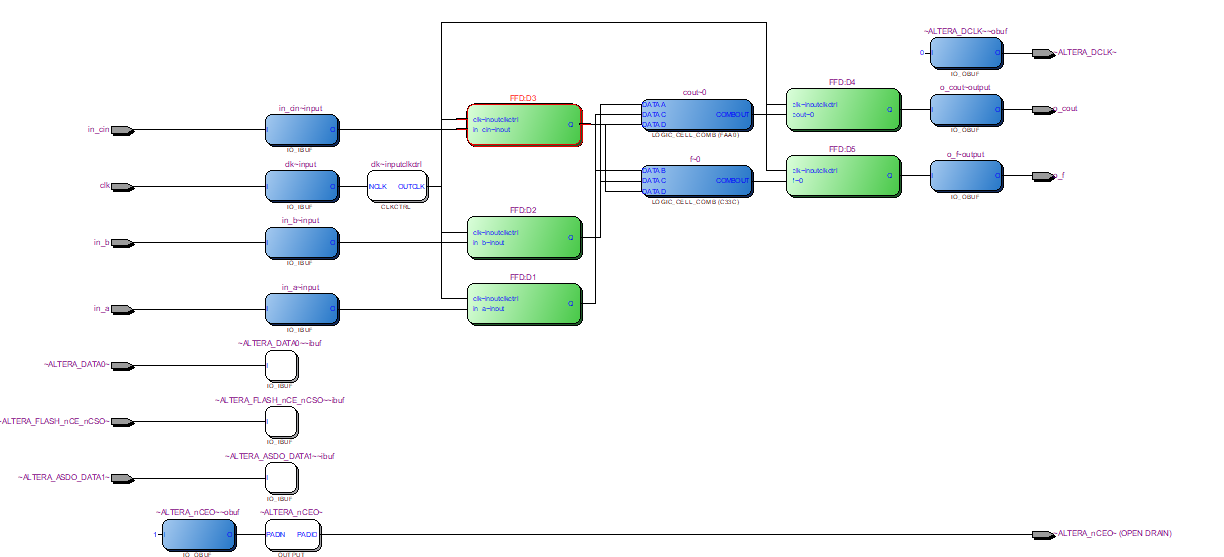
1. Verifique el circuito implementado mediante el visor de RTL



1. Verifique el circuito implementado luego del mapeo en el dispositivo

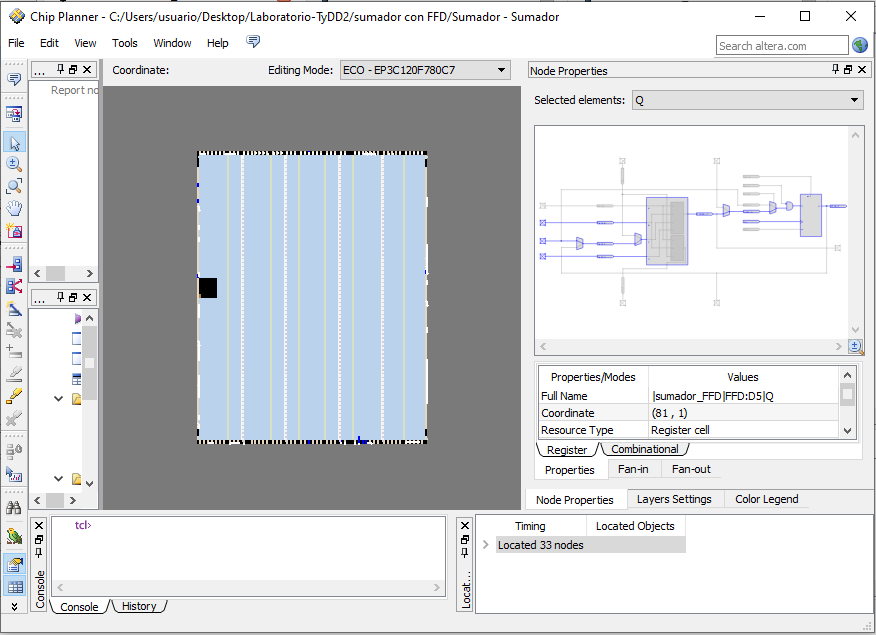


Map Viewer-Post Mapping

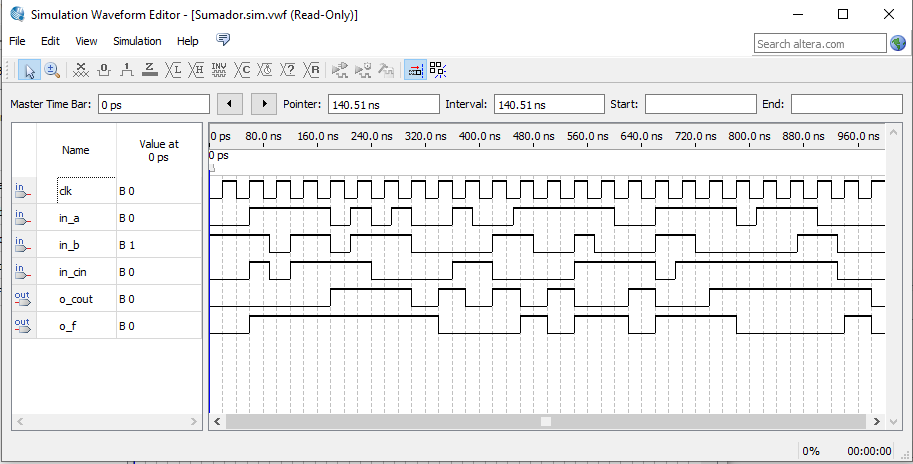


Map Viewer-Post fittiing

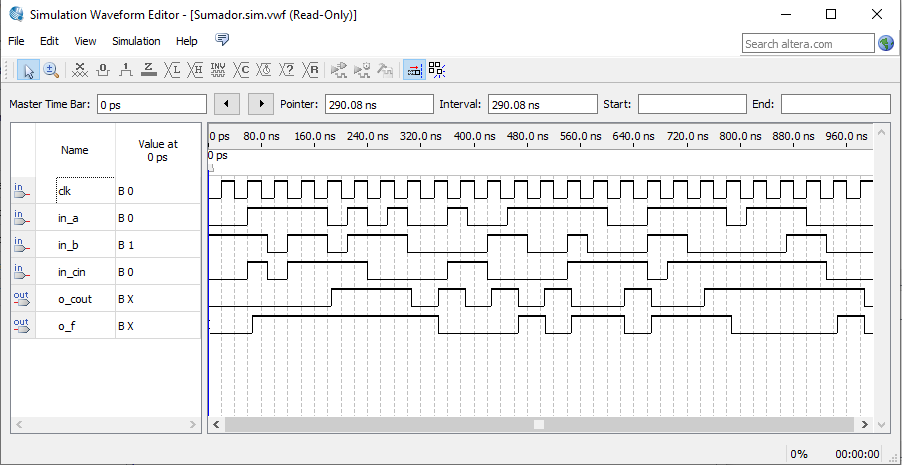
1. Verifique la implementación en el chip



1. Verifique el funcionamiento funcional y temporal del circuito mediante la simulación de Quartus

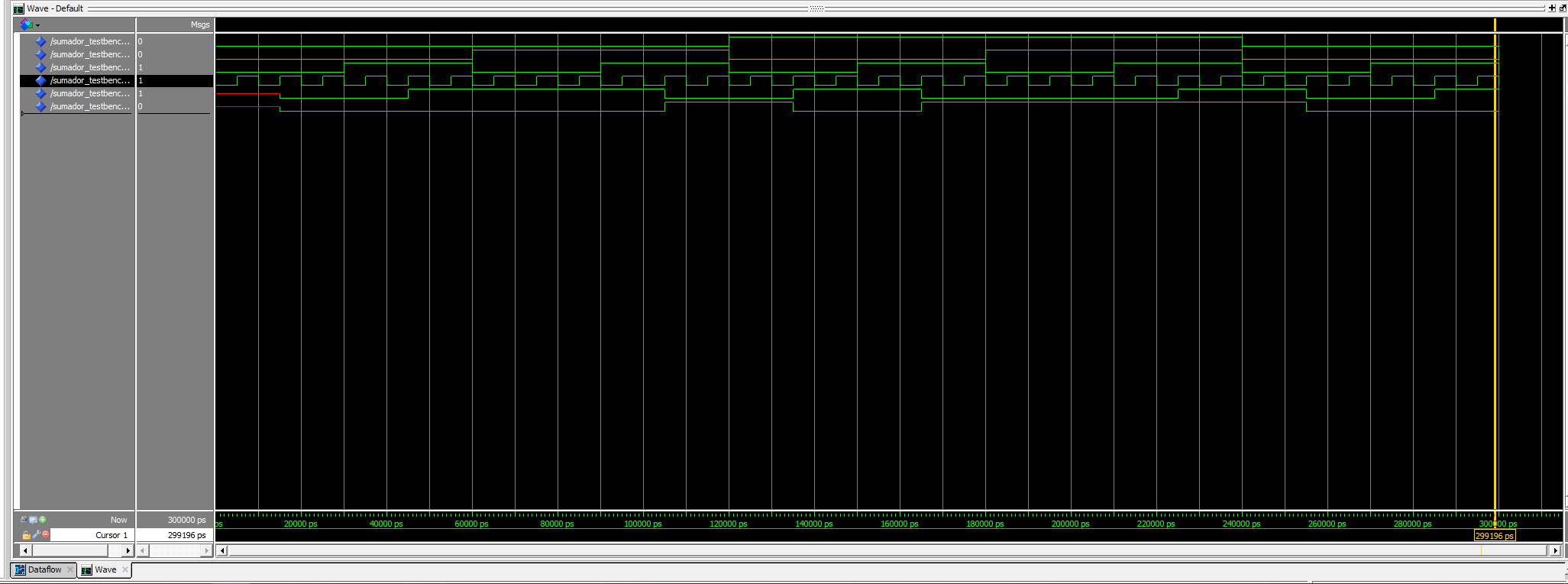


Simulación con Run Functional Simulation.

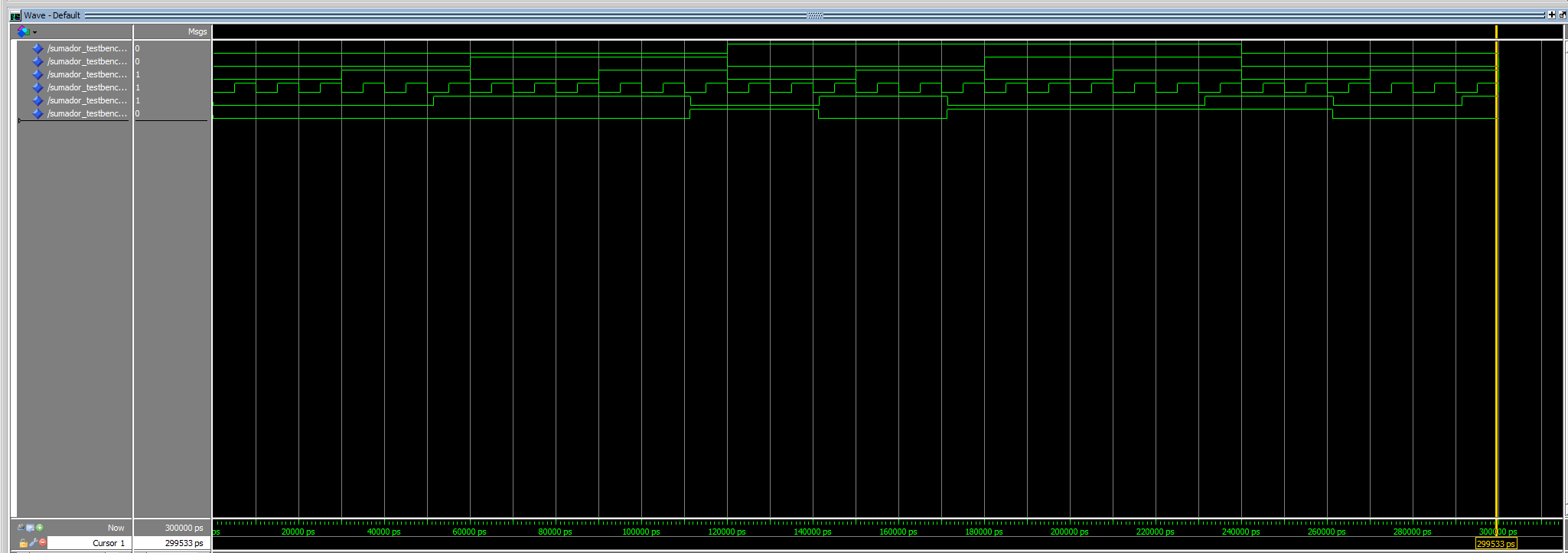


Simulación con Run Timing Simulation.

1. Verifique el funcionamiento del circuito mediante la simulación de Modelsim



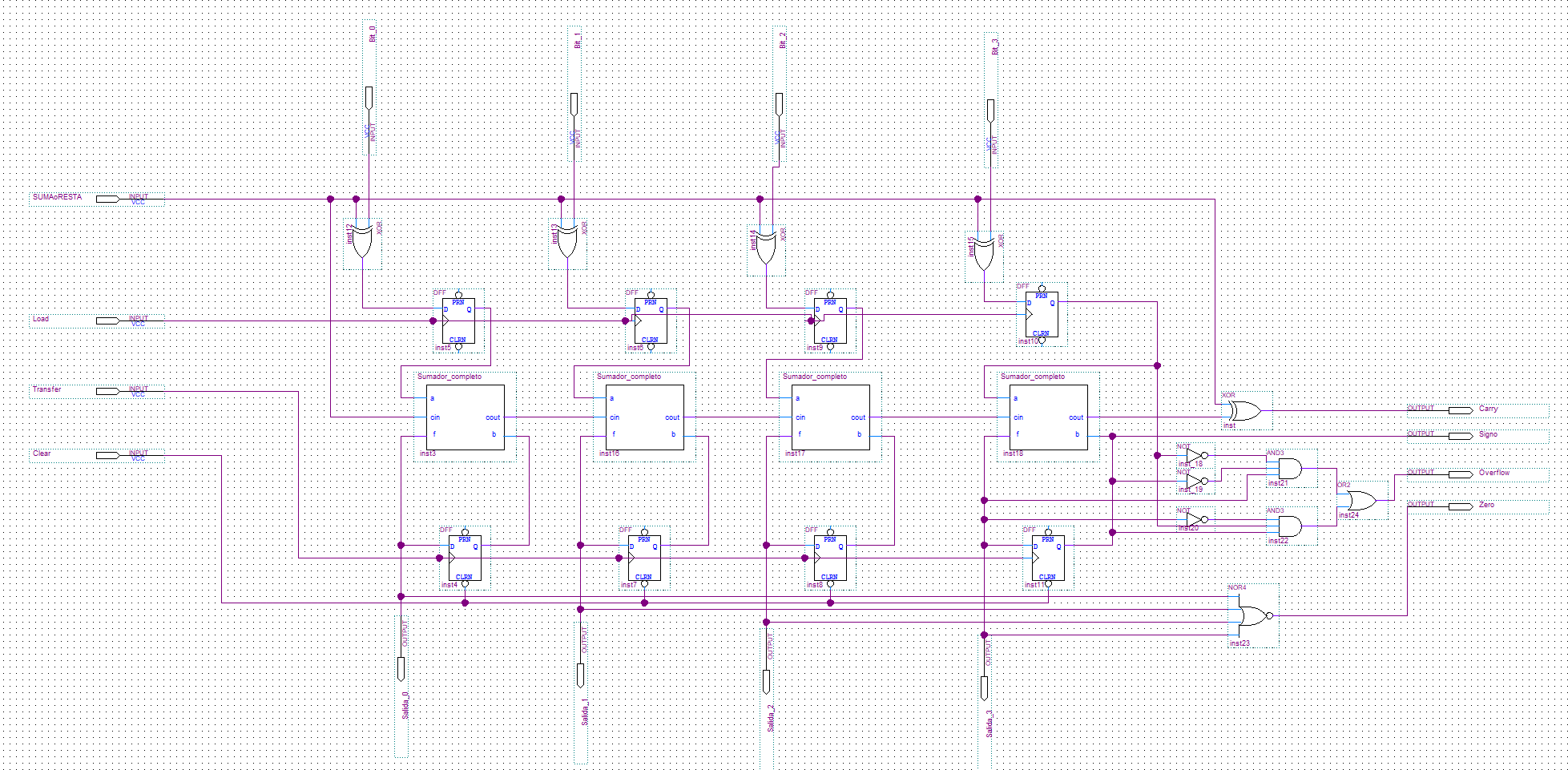
Simulación con RTL Simulation.



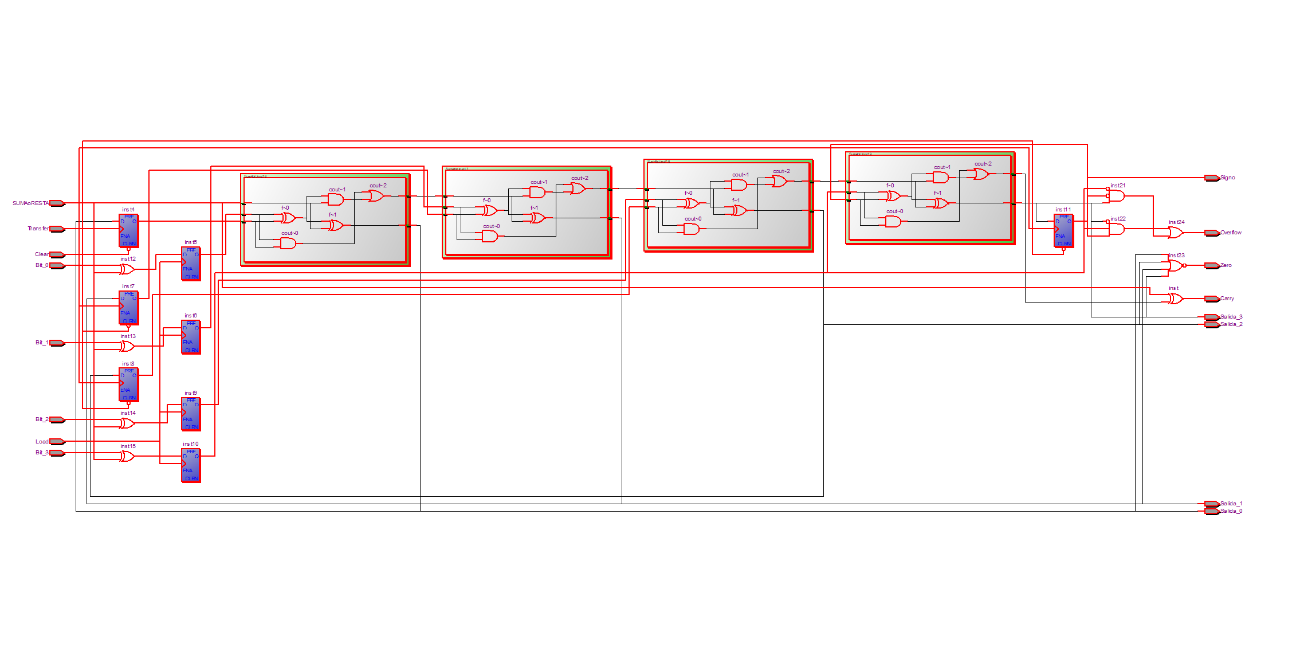
Simulación con Gate Level Simulation.

***PARTE C: Implementación de un sumador/restador de 4 bits mediante un sumador completo y los bits C, V, N y Z, empleando el entorno esquemático.***

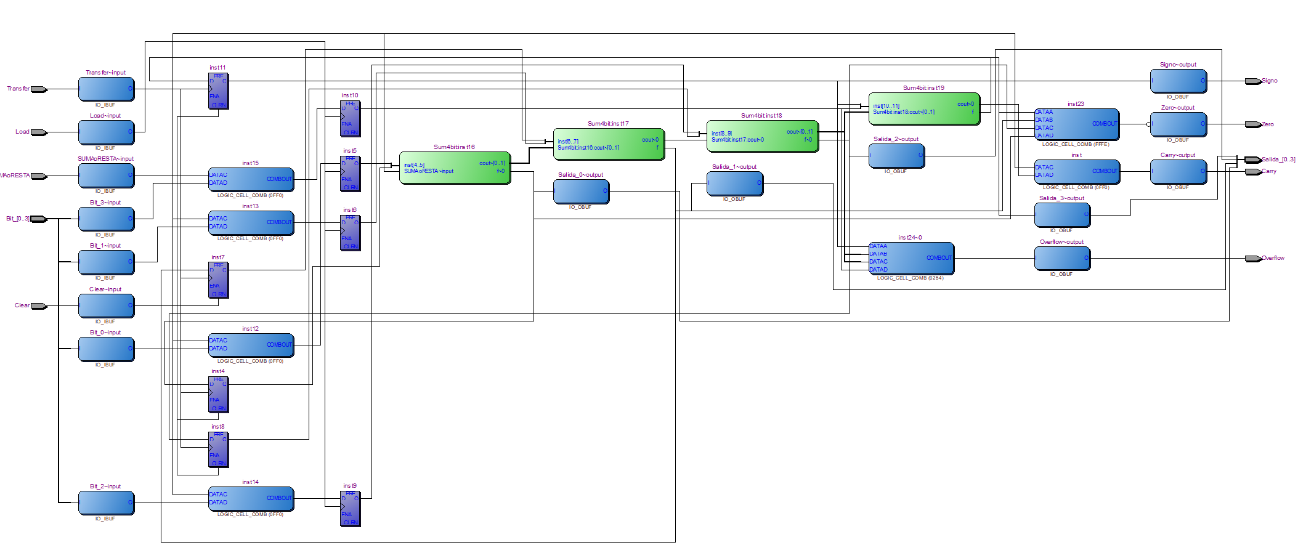
1. Genere un archivo esquemático



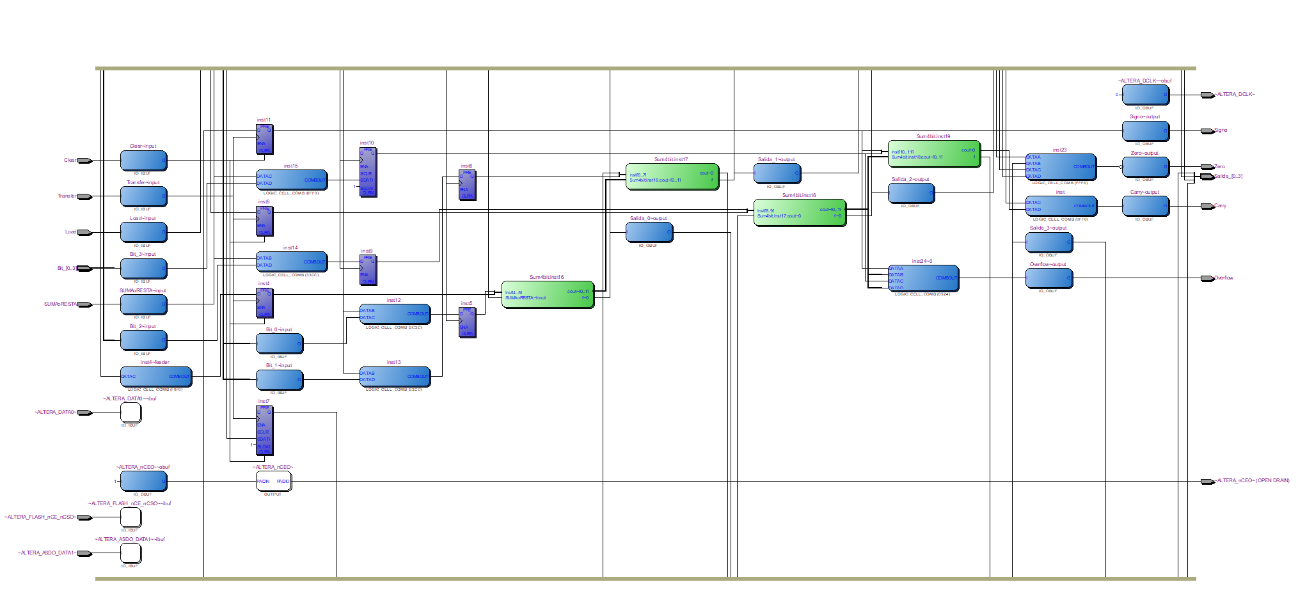
1. Verifique el circuito implementado mediante el visor de RTL.



1. Verifique el circuito implementado luego del mapeo en el dispositivo

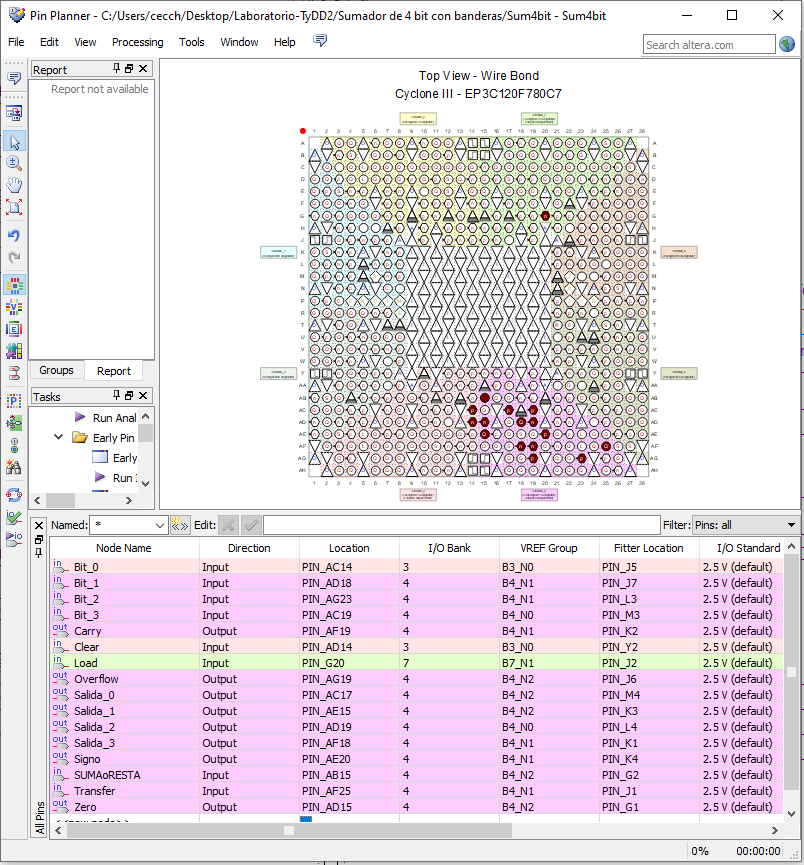


Map Viewer-Post mapping

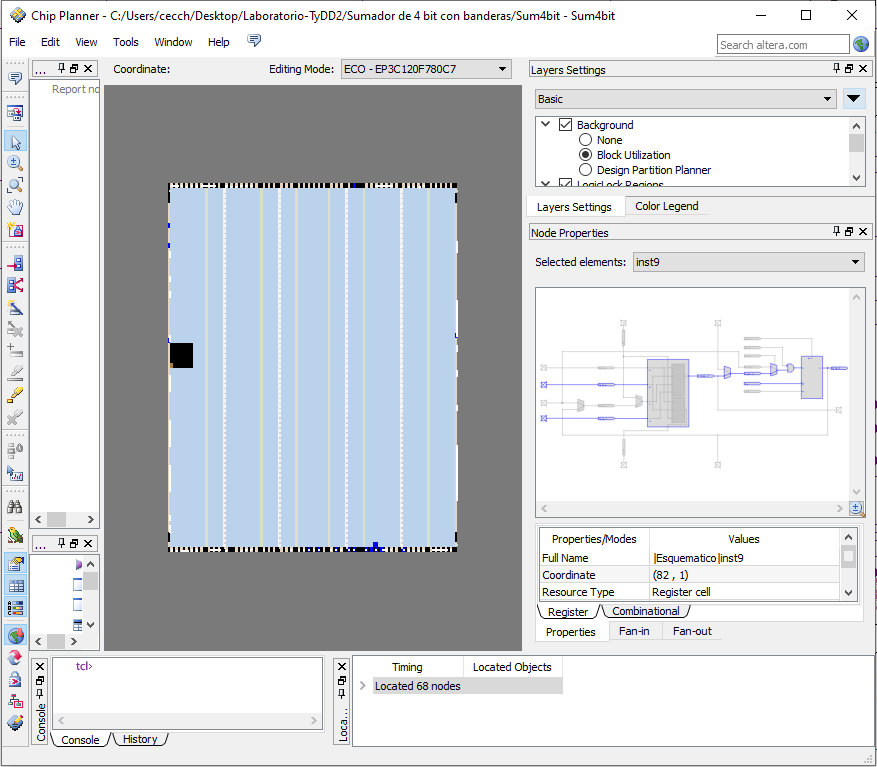


Map Viewer-Post fittiing

1. Verifique la implementación en el chip

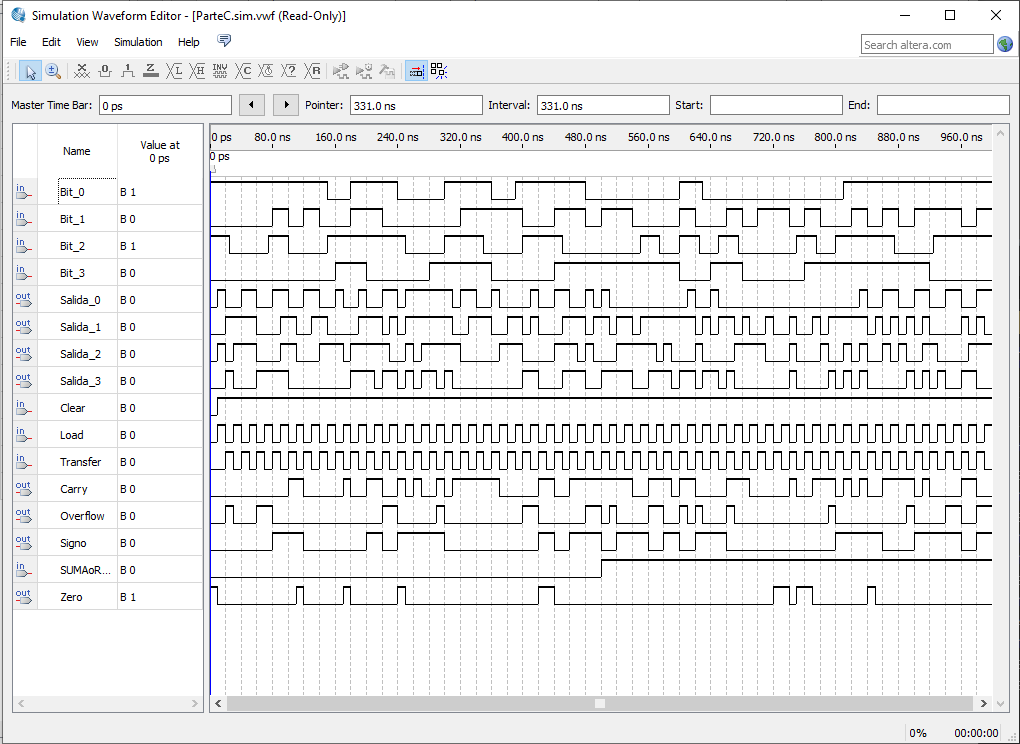


Asignación de pines

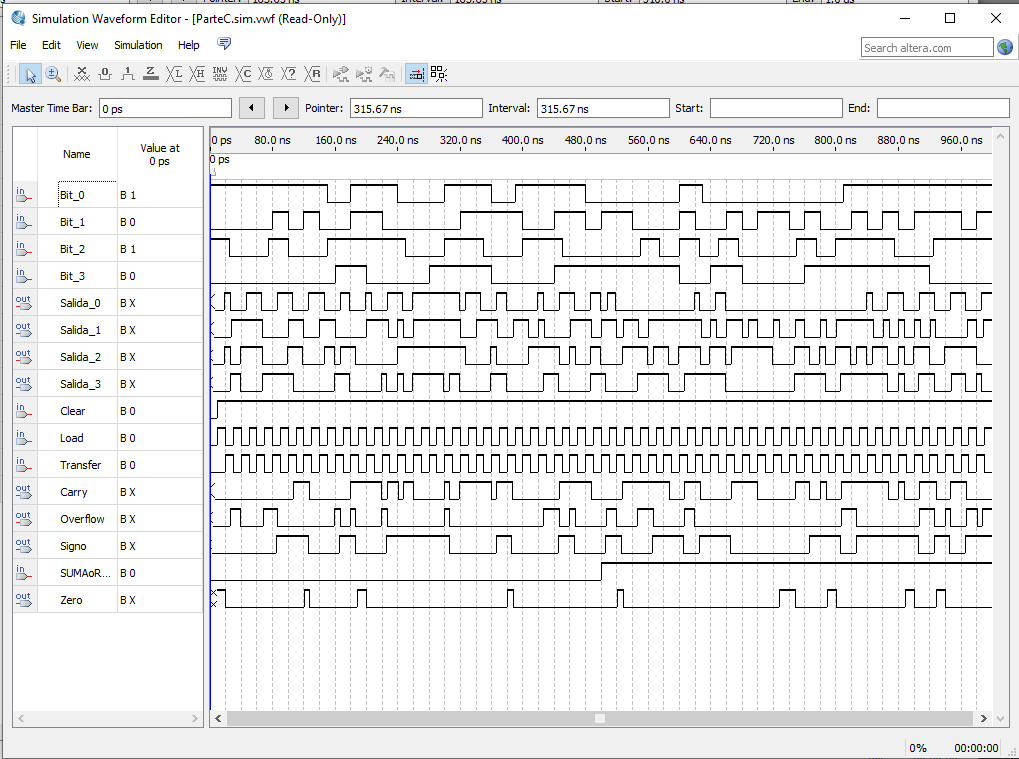


Visualización de la implementación física en el chip

1. Verifique el funcionamiento funcional y temporal del circuito mediante la simulación de Quartus

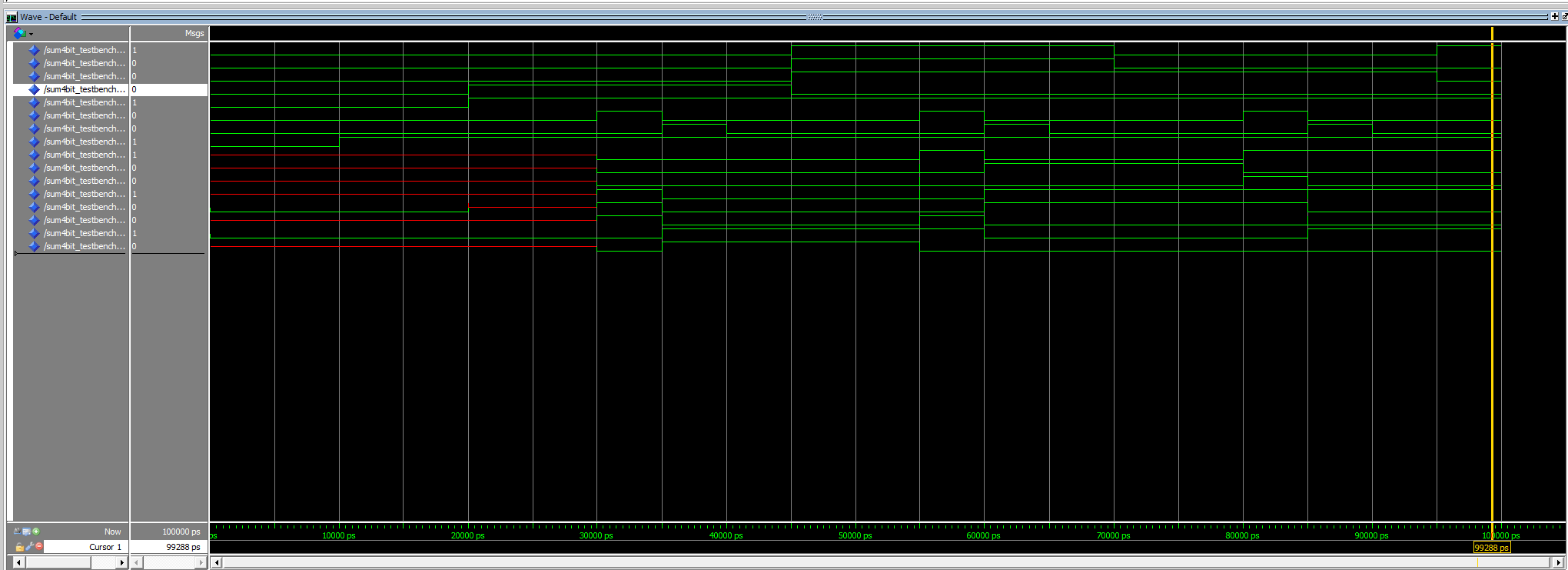


Simulación con Run Functional Simulation.

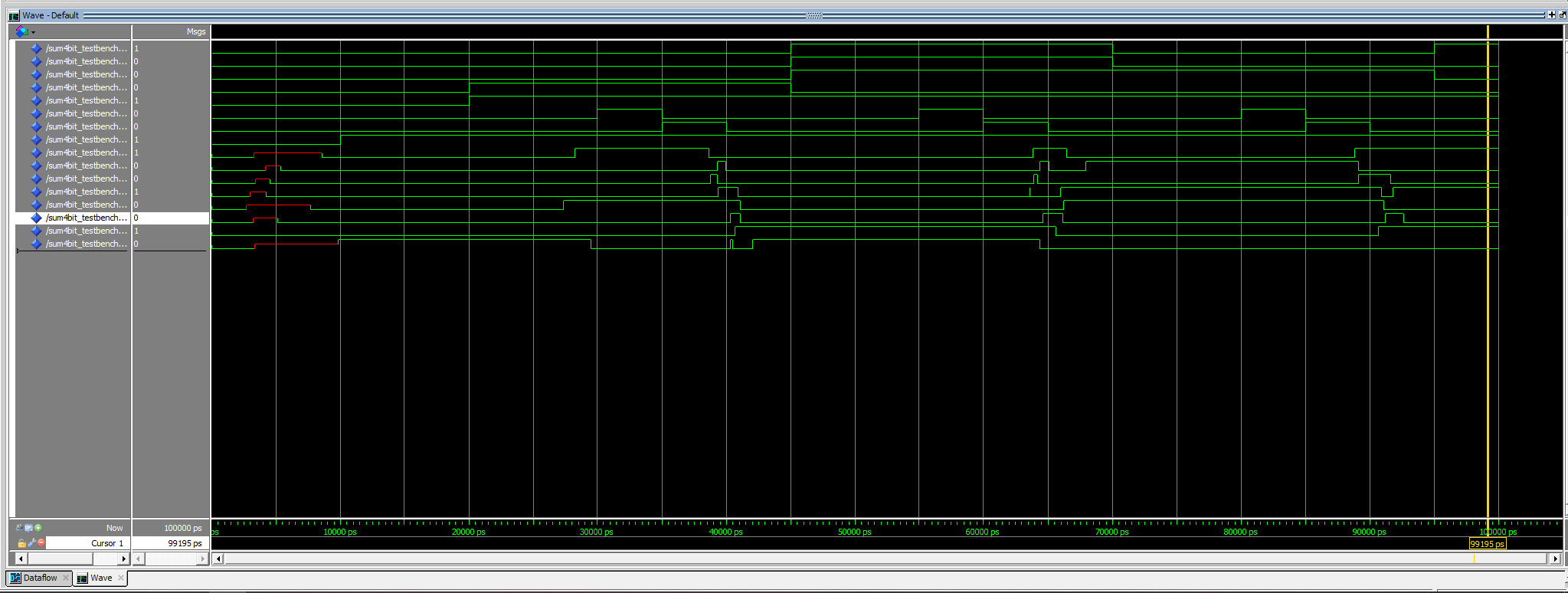


Simulación con Run Timing Simulation.

1. Verifique el funcionamiento del circuito mediante la simulación de Modelsim



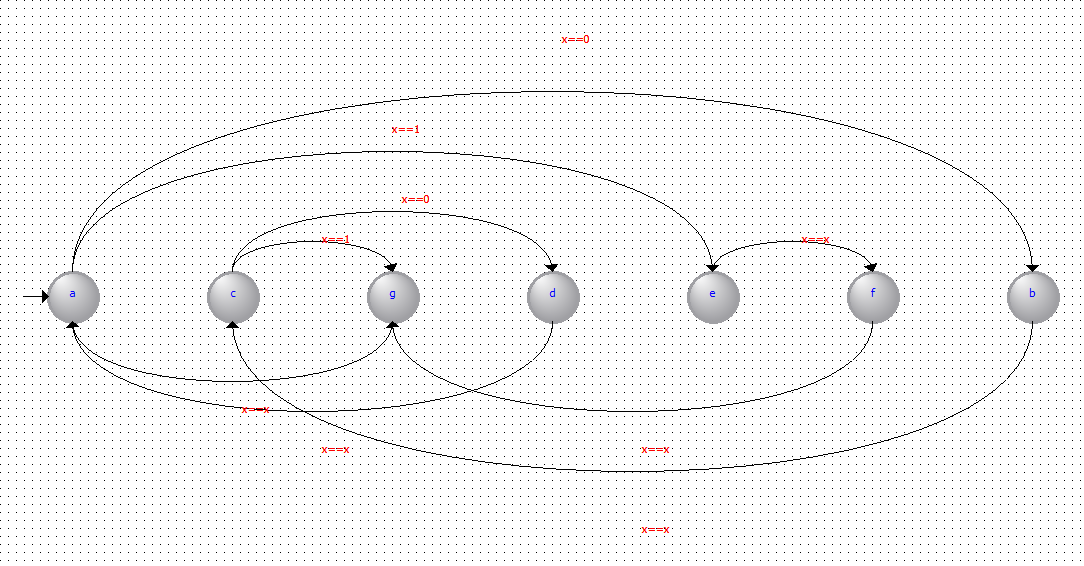
Simulación con RTL Simulation



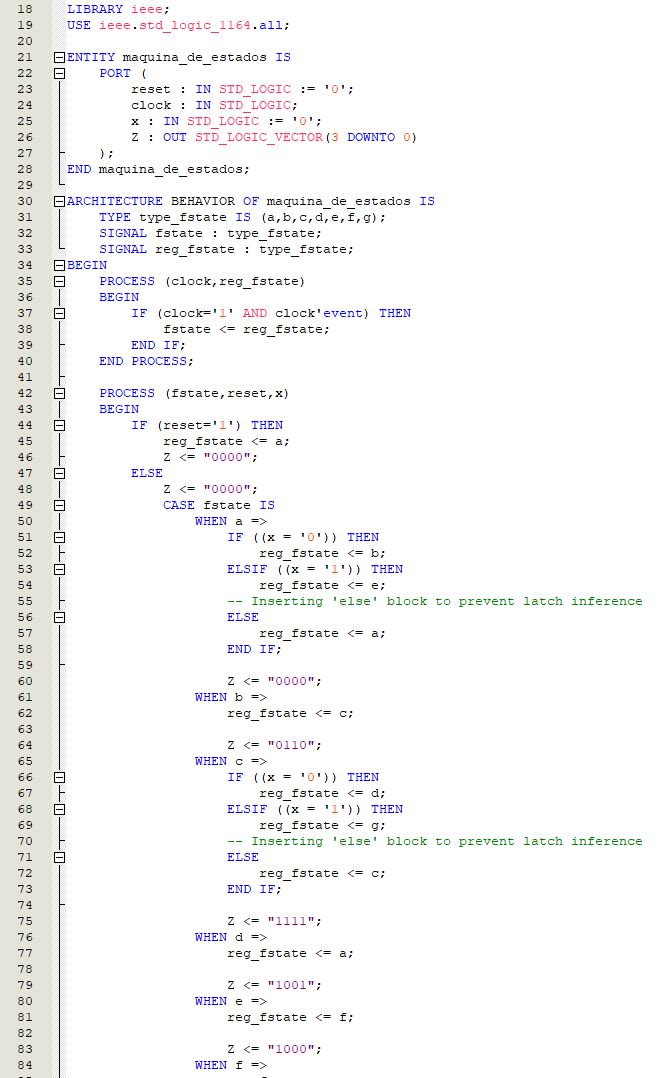
Simulación con Gate Level Simulation.

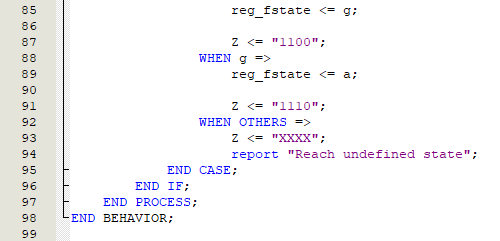
***PARTE D: Implementación de una máquina de estado.***

1. Implemente un diagrama de estados

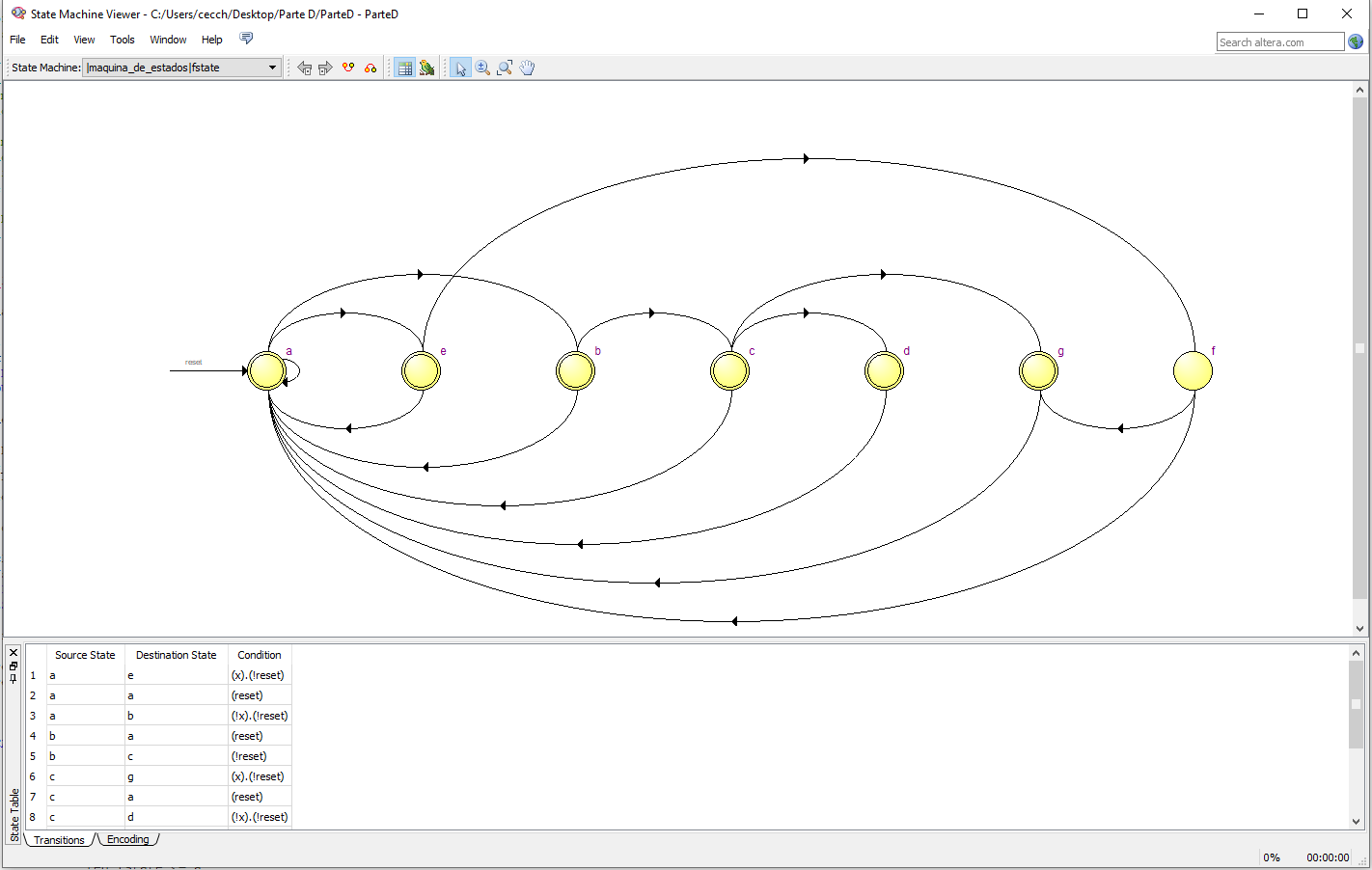


1. Genere el código VHDL

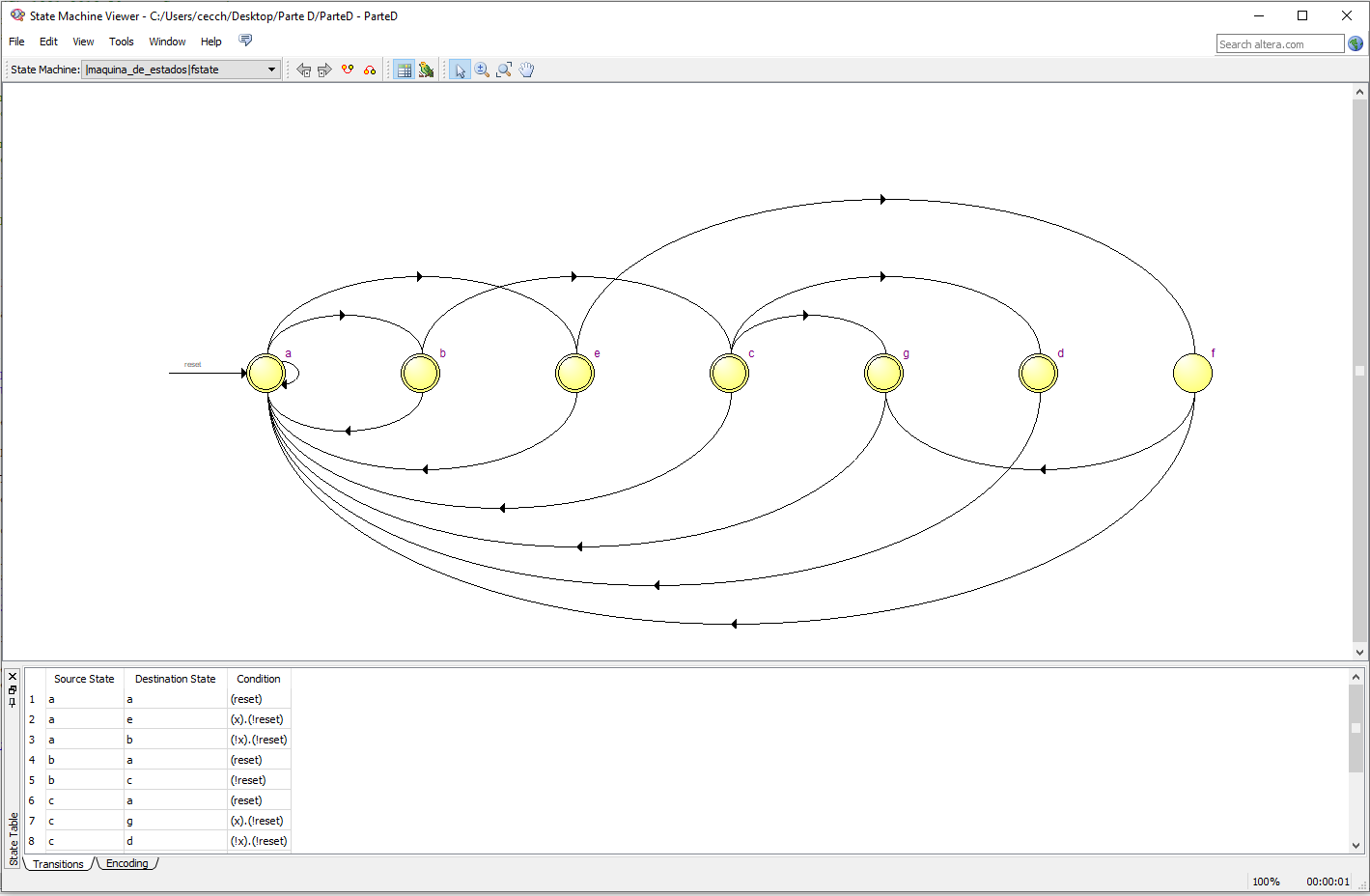




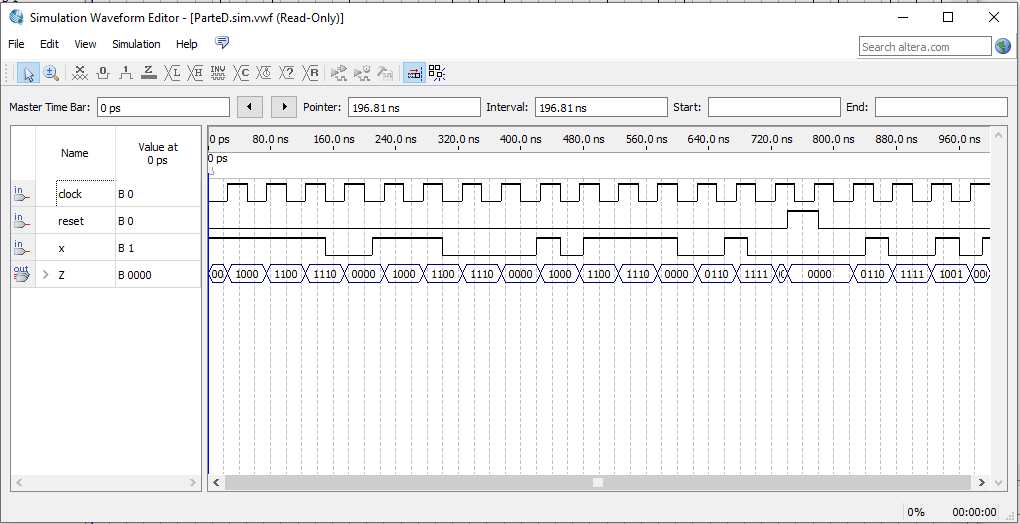
1. Vea la máquina implementada



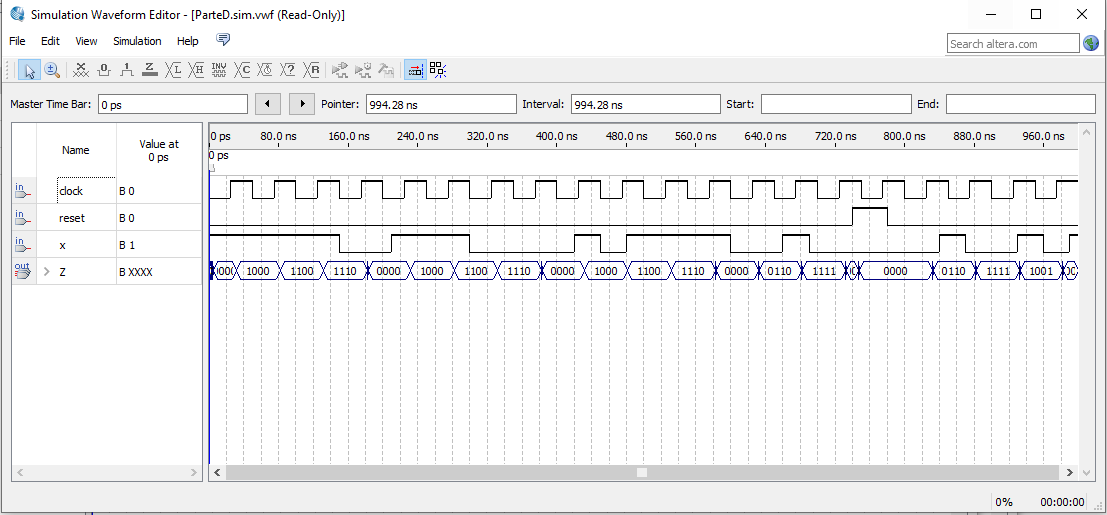
1. Cambie el estilo de procesamiento de la máquina de estado, para que el compilador asigne otra codificación. Compile nuevamente y vea el circuito implementado



1. Realice las simulaciones para verificar el correcto funcionamiento.

******

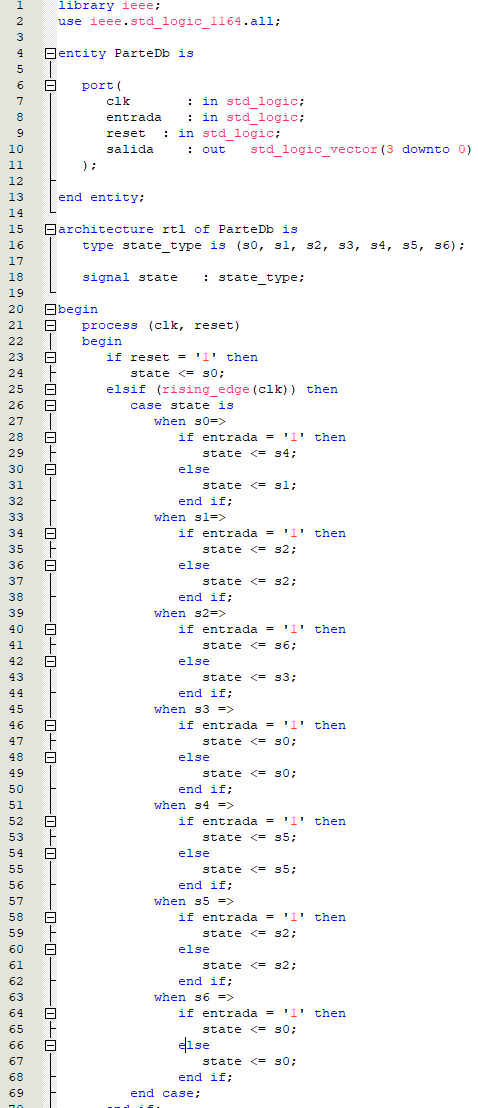
Simulación con Run Functional Simulation.

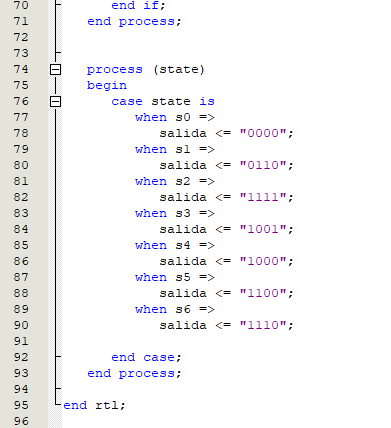
******

Simulación con Run Timing Simulation

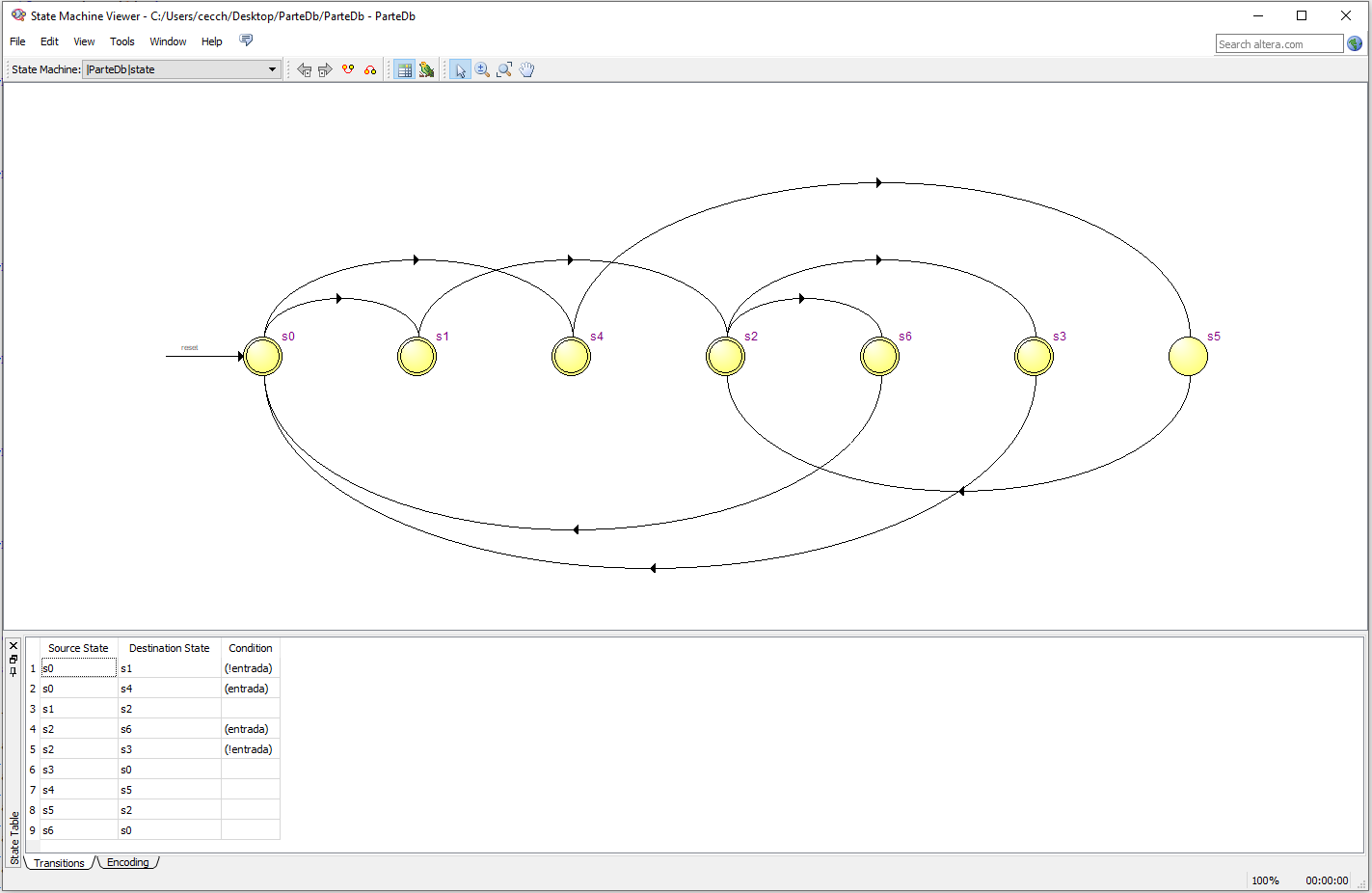
***PARTE D (Segunda parte): Implementación mediante Template de máquina de estado***

1. Seleccione la máquina de estados que desee, Moore o Mealy, note que el template es genérico de 4 estados, Ud. deberá incluir o quitar estados según la máquina que desee implementar, además de modificar la cantidad de bits de las entradas y salidas.

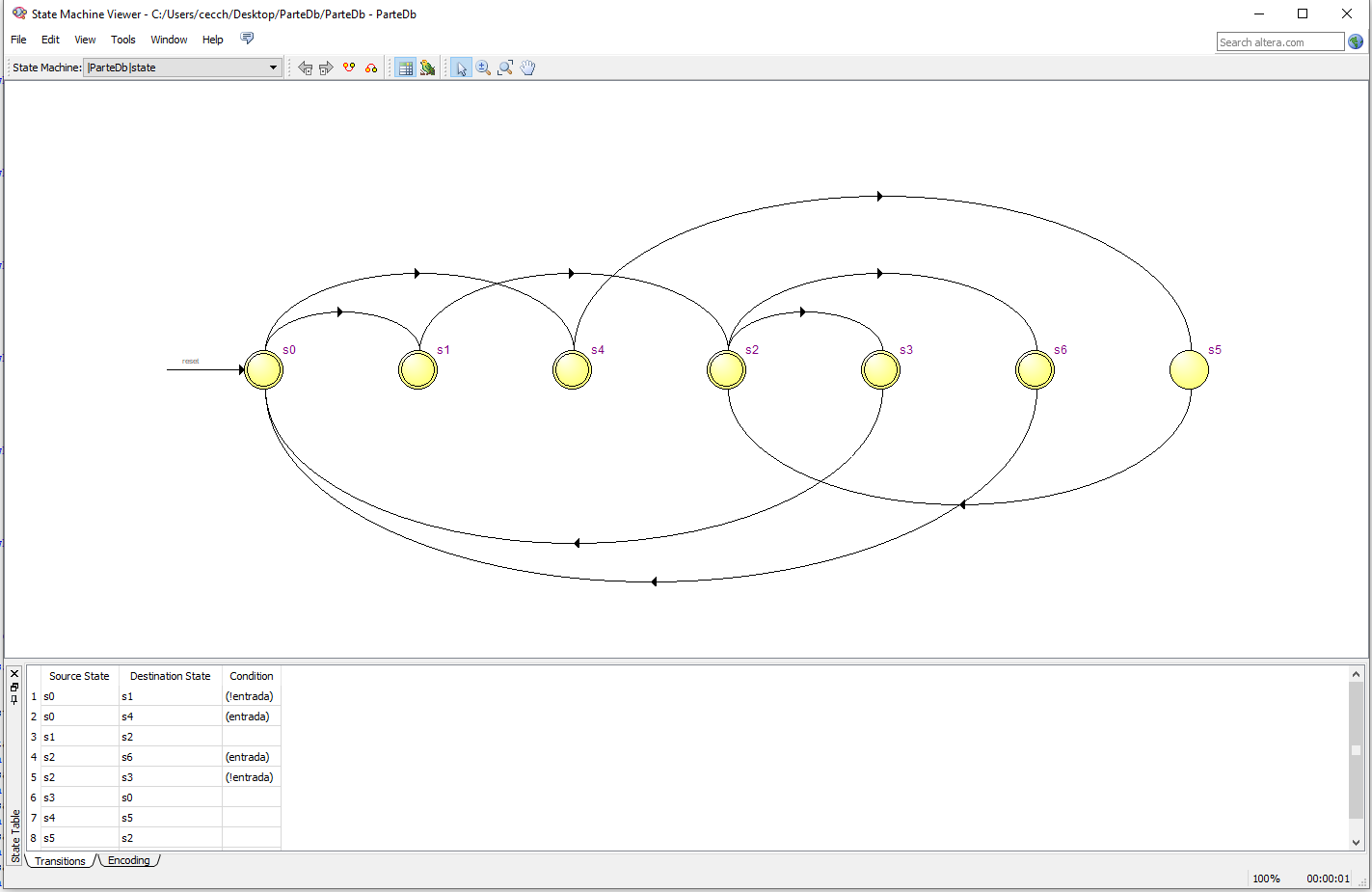




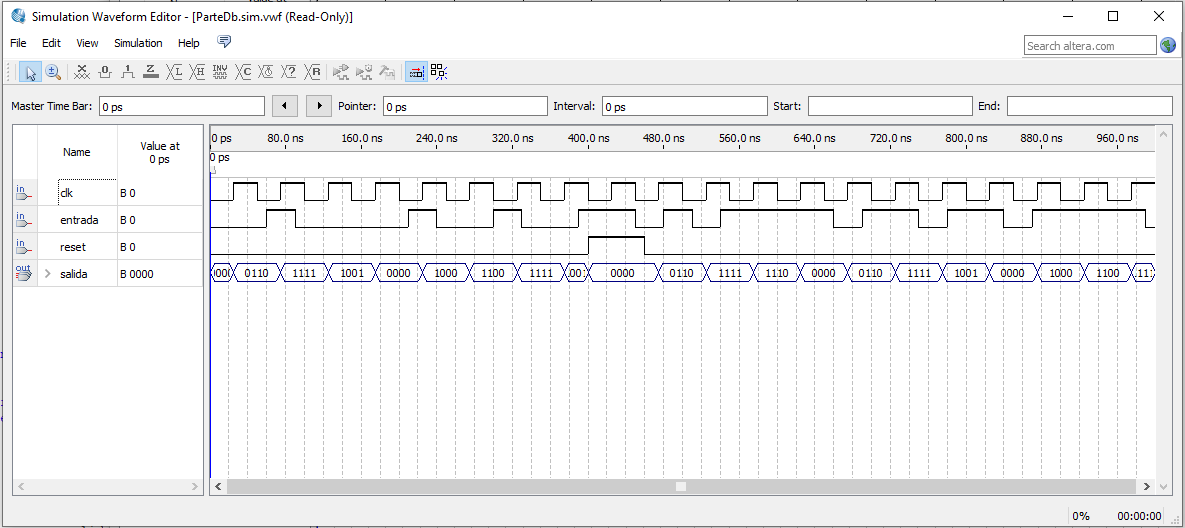
1. Vea la máquina implementada



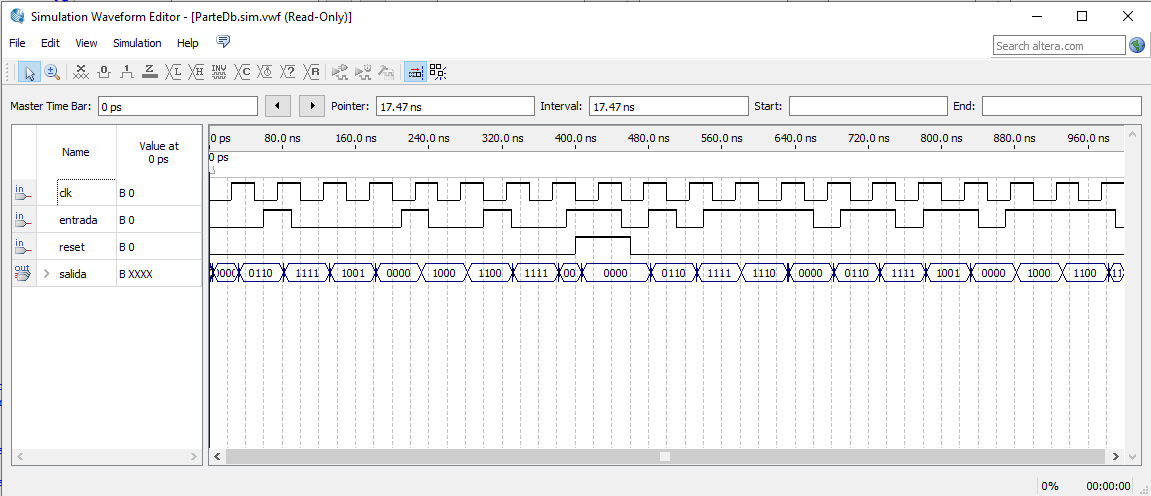
1. Cambie la codificación usada para cada estado. Compile nuevamente y vea el circuito implementado.



1. Simule para verificar el correcto funcionamiento



Simulación con Run Functional Simulation.



Simulación con Run Timing Simulation