Universidad Nacional Autónoma de México

Facultad de Ingeniería

Arquitectura de Computadoras 2020-1

Practica 1: Circuitos Secuenciales

Aguilar Enriquez Paul Sebastian

Berdejo Arvizu Oscar

Pérez Bueno Ana Laura

Rojas Héctor

Grupo: 01

09 de Septiembre del 2019

M.I. Jose Antonio de Jesus Arredondo Garza

Objetivos

- Conocer la estructura y características de la tarjeta de los dispositivos lógicos programables que se dispone en el laboratorio, tarjeta TerAsic, el software de operación de esta, Quartus, y su programación en lenguaje VHDL y gráfico.
- Repasar el diseño de circuitos secuenciales básicos y algoritmos de máquinas de estados.

Introducción

Circuito secuencial

Es un circuito cuya salida depende no solo de la combinación de entrada, sino también de la historia de las entradas anteriores. Es decir aquellos circuitos en que el contenido de los elementos de memoria sólo puede cambiar en presencia de un pulso de reloj . Entre pulso y pulso de reloj, la información de entrada puede cambiar y realizar operaciones lógicas en el circuito combinacional, pero no hay cambio en la información contenida en las células de memoria.

El circuito secuencial debe ser capaz de mantener su estado durante algún tiempo, para ello se hace necesario el uso de dispositivos de memoria. Los dispositivos de memoria utilizados en circuitos secuenciales pueden ser tan sencillos como un simple retardador (inclusive, se puede usar el retardo natural asociado a las compuertas lógicas) o tan complejos como un circuito completo de memoria denominado multivibrador biestable o Flip Flop.

Los circuitos secuenciales se clasifican de acuerdo a la manera como manejan el tiempo:

- Circuitos secuenciales sincrónicos
- Circuitos secuenciales asíncronos.

En **un circuito secuencial asíncrono**, los cambios de estado ocurren al ritmo natural marcado por los retardos asociados a las compuertas lógicas utilizadas en su implementación, es decir, estos circuitos no usan elementos especiales de memoria, pues se sirven de los retardos propios (tiempos de propagación) de las compuertas lógicas usados en ellos. Esta manera de operar puede ocasionar algunos problemas de funcionamiento, ya que estos retardos naturales no están bajo el control del diseñador y además no son idénticos en cada compuerta lógica.

Los **circuitos secuenciales síncronos**, sólo permiten un cambio de estado en los instantes marcados por una señal de reloj. Con ésto se pueden evitar los problemas que tienen los circuitos asíncronos originados por cambios de estado no uniformes en todo el circuito.

FPGA (Field-programmable gate array)

Un FPGA es un dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada en el momento, mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip.

Las FPGA se utilizan en aplicaciones similares a los ASIC sin embargo son más lentas, tienen un mayor consumo de energía y no pueden abarcar sistemas tan complejos como ellos. A pesar de esto, las FPGA tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor.

Quartus II

Es una herramienta de software producida por Altera para el análisis y la síntesis de diseños realizados en HDL. Permite al desarrollador o desarrolladora compilar sus diseños, realizar análisis temporales, examinar diagramas RTL y configurar el dispositivo de destino con el programador.

La Edición Web es una versión gratuita de Quartus II que puede ser descargada o enviada gratuitamente por correo. Esta edición permite la compilación y la programación de un número limitado de dispositivos Altera.

La familia de FPGAs de bajo coste Cyclone, está soportada por esta edición, por lo que los pequeños desarrolladores y desarrolladoras no tendrán problemas por el coste del desarrollo de software.

Se requiere un registro de licencia para utilizar la Edición Web de Quartus II, la cual es gratuita y puede ser renovada ilimitadamente o de pago.

Lenguaje de descripción de hardware

Un lenguaje de descripción de hardware (HDL, hardware description language) es un lenguaje de programación especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos, y más comúnmente, de circuitos electrónicos digitales, como el convertidor analógico-digital o cualquier antena satelital. Así, los lenguajes de descripción de hardware hacen posible una descripción formal de un circuito electrónico, y posibilitan su análisis automático y su simulación.

Los lenguajes de descripción de hardware se parecen mucho a otros lenguajes de programación de ordenadores tales como el C o Java: básicamente consisten en una descripción textual con expresiones, declaraciones y estructuras de control. Sin embargo,

una importante diferencia entre los HDL y otros lenguajes de programación está en que el HDL incluye explícitamente la noción de tiempo.

ModelSim

ModelSim es un entorno de simulación HDL multi-idioma de Mentor Graphics, para la simulación de lenguajes de descripción de hardware como VHDL, Verilog y SystemC, e incluye un depurador C integrado. ModelSim se puede utilizar de forma independiente o en combinación con Intel Quartus Prime, Xilinx ISE o Xilinx Vivado. La simulación se realiza utilizando la interfaz gráfica de usuario (GUI), o automáticamente utilizando scripts.

ModelSim PE es el simulador de nivel básico para aficionados y estudiantes.

ModelSim también se puede utilizar con MATLAB/Simulink, utilizando Link for ModelSim. Link for ModelSim es una interfaz de co-simulación bidireccional rápida entre Simulink y ModelSim. Para tales diseños, MATLAB proporciona un conjunto de herramientas de simulación numérica, mientras que ModelSim proporciona herramientas para verificar la implementación de hardware y las características de tiempo del diseño.

ModelSim utiliza un núcleo unificado para la simulación de todos los lenguajes soportados, y el método de depuración de código C embebido es el mismo que el de VHDL o Verilog.

Desarrollo

El proyecto se desarrolló para una tarjeta **Cyclone II** modelo **EP2C5T144C8**, por lo cual se utilizó Quartus 13.1.

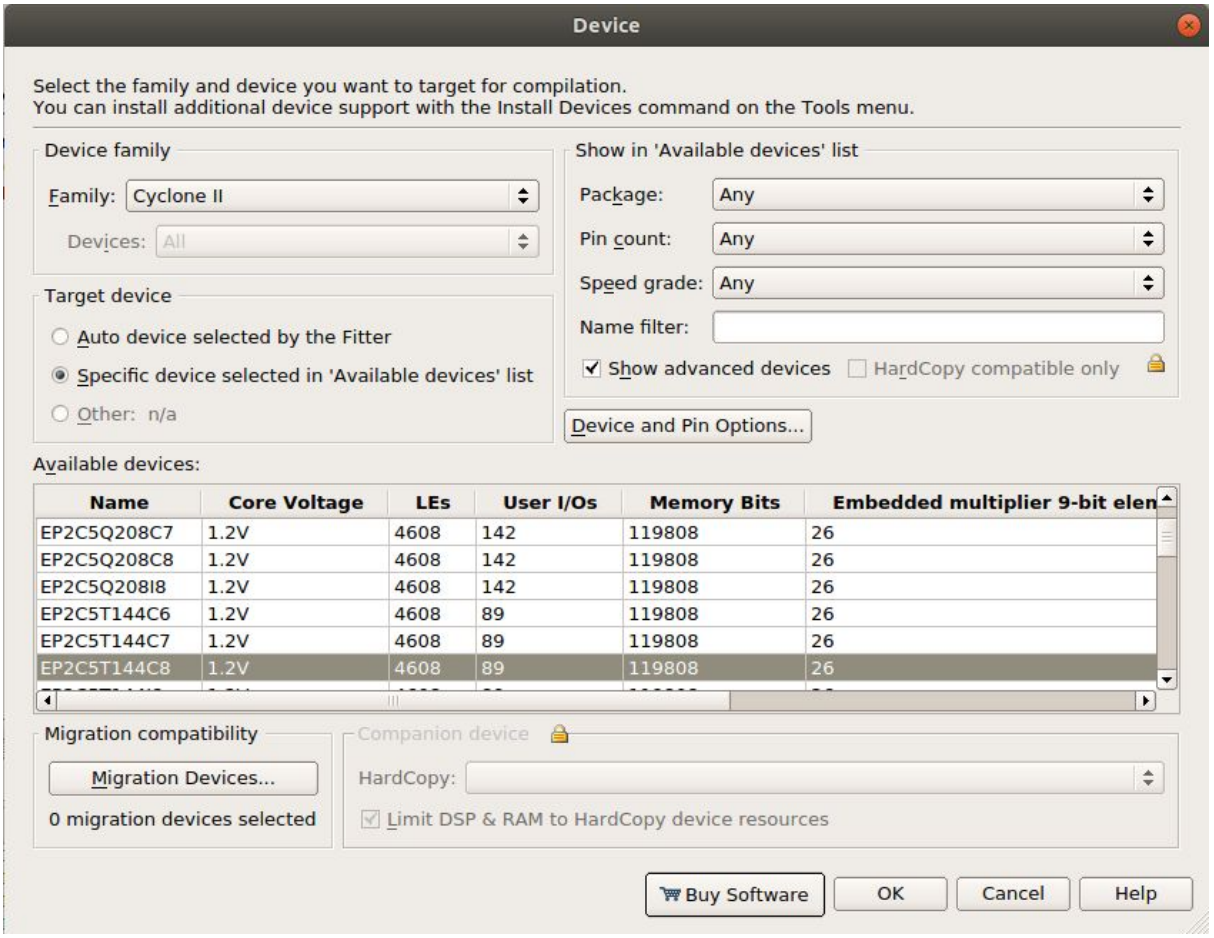

Figura 1: Configuración del proyecto y la tarjeta.

Se implementó un contador binario de cuatro bits.

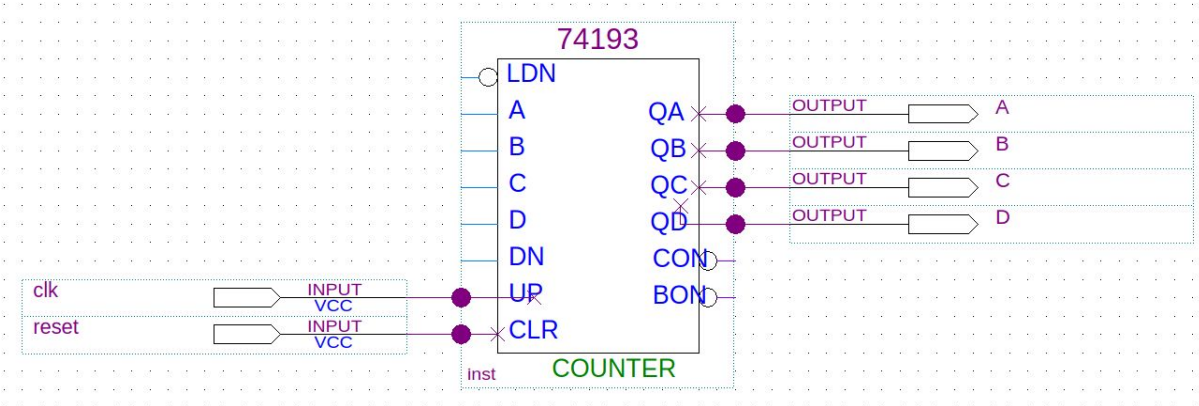

Figura 2: Implementación por bloques/símbolos del contador binario de cuatro bits.

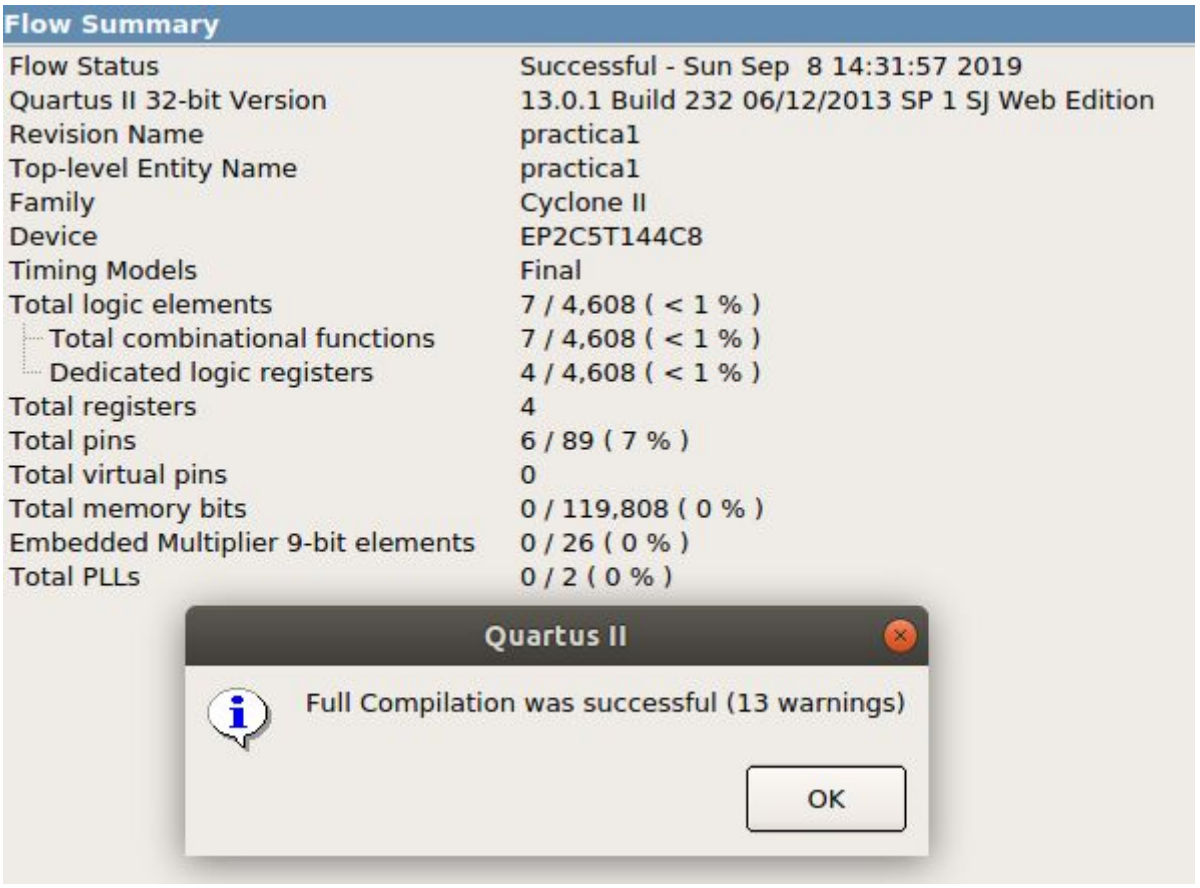

Figura 3: Compilación exitosa del contador binario.

Una vez realizado el diseño del contador, se procedió a simularlo con **ModelSim de Altera**.

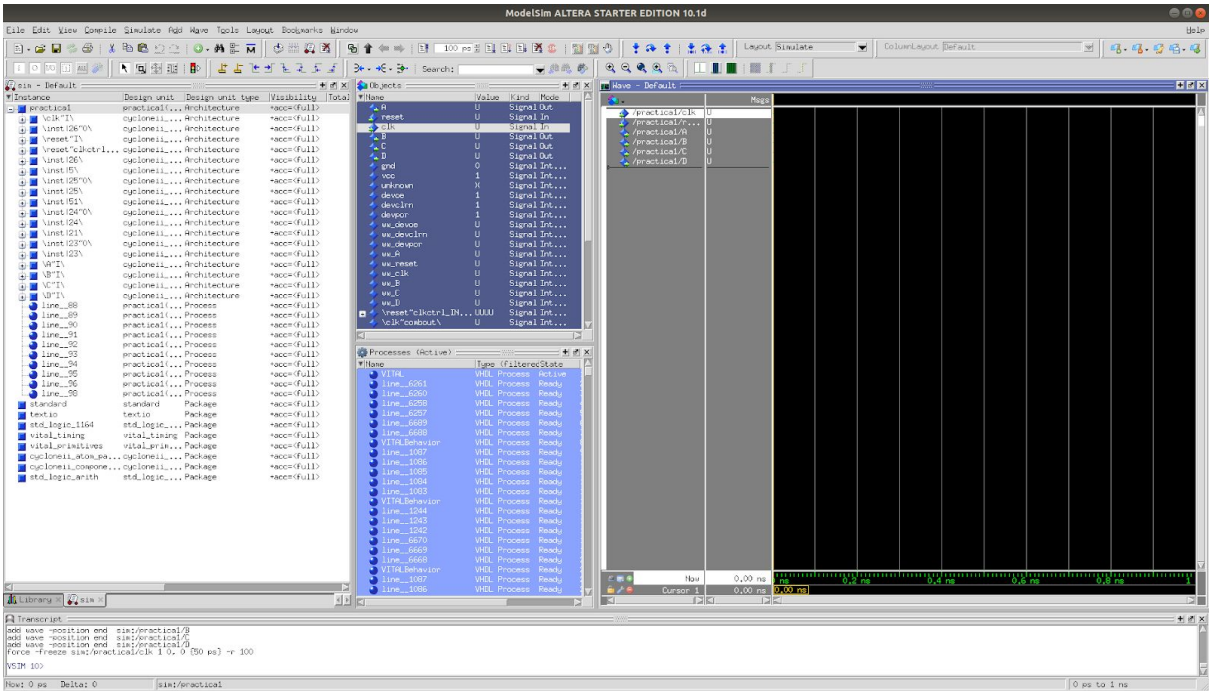

Figura 4: Entorno de simulación de ModelSim de Altera.

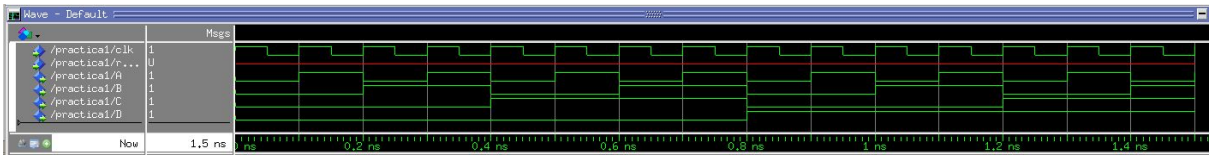

Figura 5: Simulación del contador con un reloj configurado a un un periodo de 100 ms.

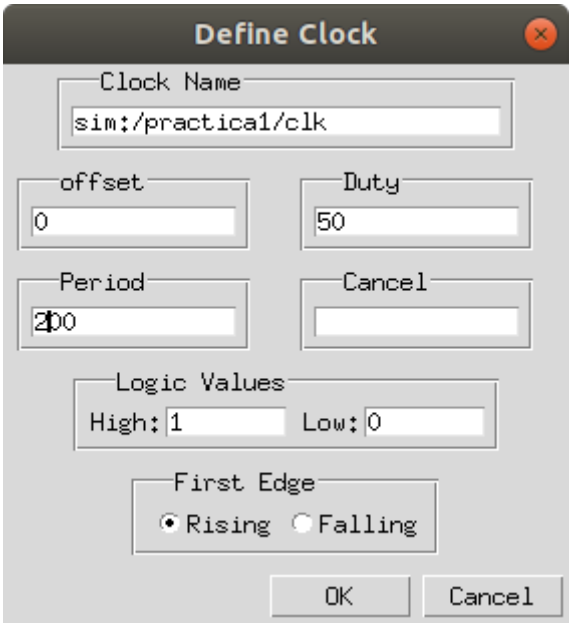

Figura 6: Configuración del reloj a un periodo de 200 ms.

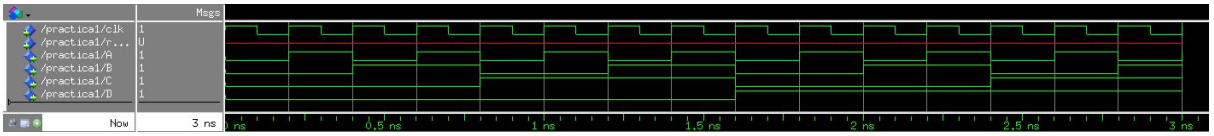

Figura 7: Simulación del contador con un reloj configurado a un un periodo de 200 ms.

Para observar físicamente el contador binario funcionando a un tiempo razonable con los LED's de salida fue necesario crear un bloque que funcionara como divisor de tiempo.

```
1   library IEEE;
2
3   use IEEE.STD_LOGIC_1164.ALL;
4   use IEEE.STD_LOGIC_ARITH.ALL;
5   use IEEE.STD_LOGIC_UNSIGNED.ALL;
6
7   entity divider is
8       Port (
9           reloj : in std_logic;
10          div_clk : out std_logic
11      );
12  end divider;
13
14  architecture Behavioral of divider is
15      begin process (reloj)
16          variable cuenta : std_logic_vector (27 downto 0) := X"0000000";
17          begin
18          if rising_edge (reloj) then
19              if cuenta = X"4000000" then
20                  cuenta := X"0000000";
21              else
22                  cuenta := cuenta+1;
23              end if;
24          end if;
25          div_clk <= cuenta (25);
26      end process;
27  end Behavioral;
```

Figura 8: Código del bloque de divisor de tiempo.

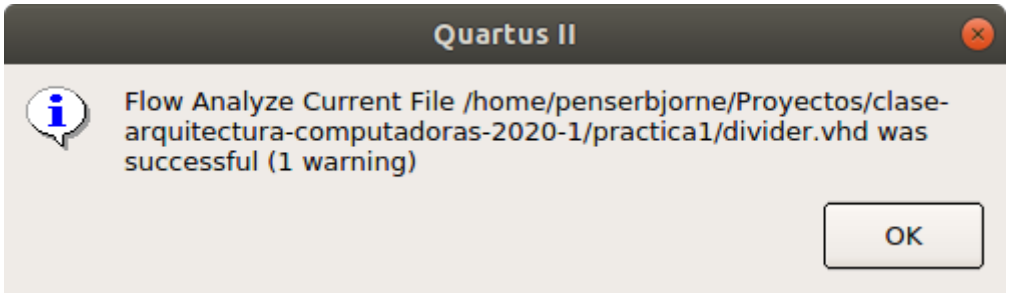

Figura 9: Aviso de procesamiento correcto del código del divisor de tiempo.

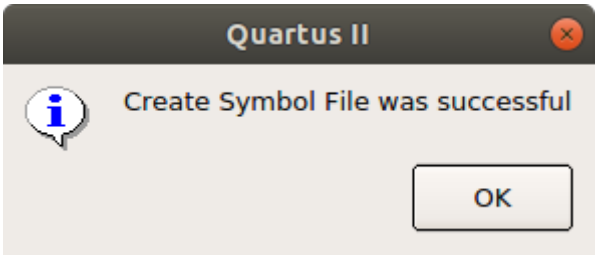

Figura 10: Aviso de que el símbolo del divisor de tiempo se creó correctamente para ser utilizado en el esquemático.

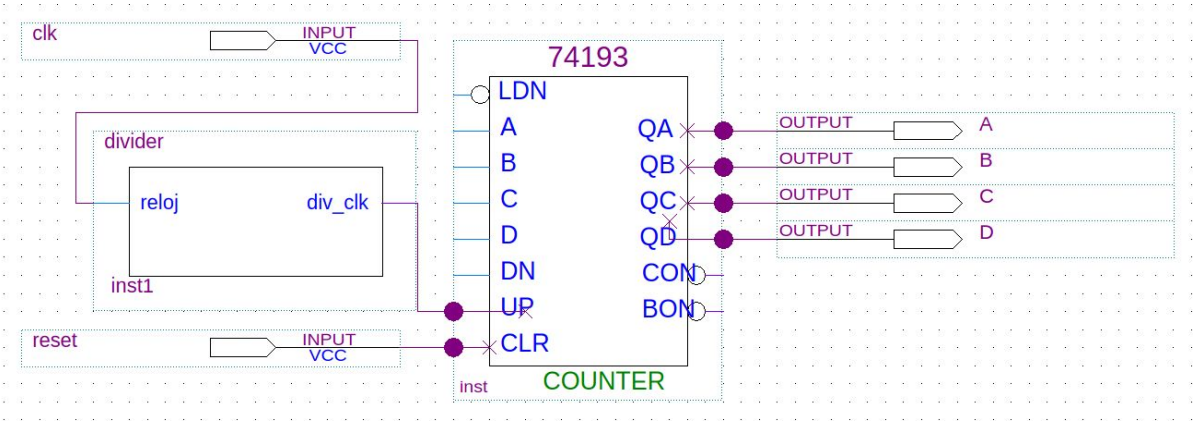

Figura 11: Implementación por bloques/símbolos del contador binario de cuatro bits con el divisor de tiempo incluido.

Debido al manejo de los valores lógicos en los pines físicos de entrada y salida de la tarjeta, es necesario incluir unas compuertas **NOT** en los pines para poder interactuar con los elementos físicos en la tarjeta, en este caso los LEDs para visualizar el conteo y el botón de reset.

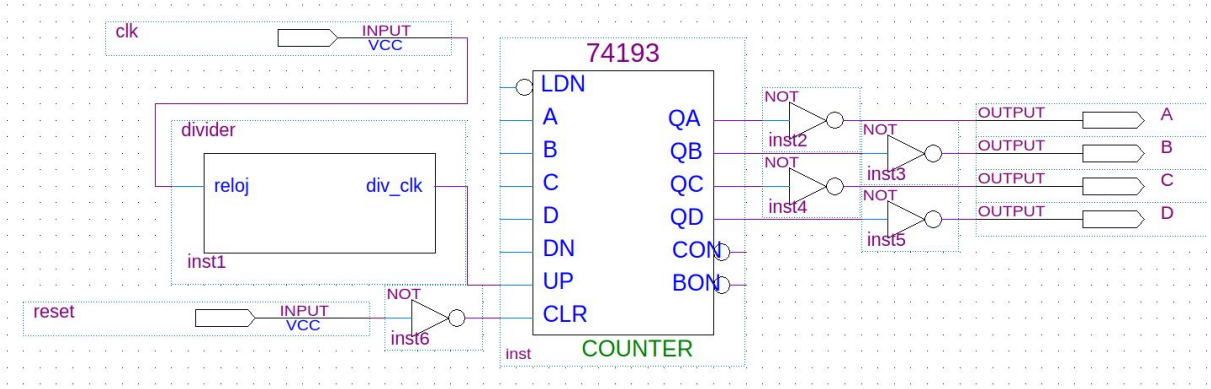

Figura 12: Implementación por bloques/símbolos del contador binario de cuatro bits con el divisor de tiempo incluido y las compuertas NOT en las entradas y salidas.

Para la interacción física con los elementos de entrada y salida es necesario asignar los pines en la tarjeta.

Los pines quedaron asignados como:

- **clk:** Al reloj interno del PIN 17, al ser una señal interna no se conecta con ningún elemento externo ya que esta es producida por los relojes propios del FPGA.
- **A:** PIN I/O 96, este bit es el bit menos significativo del contador.
- **B:** PIN I/O 93.
- **C:** PIN I/O 81.
- **D:** PIN I/O 75, este bit es el bit más significativo del contador.
- **reset:** PIN I/O 73.

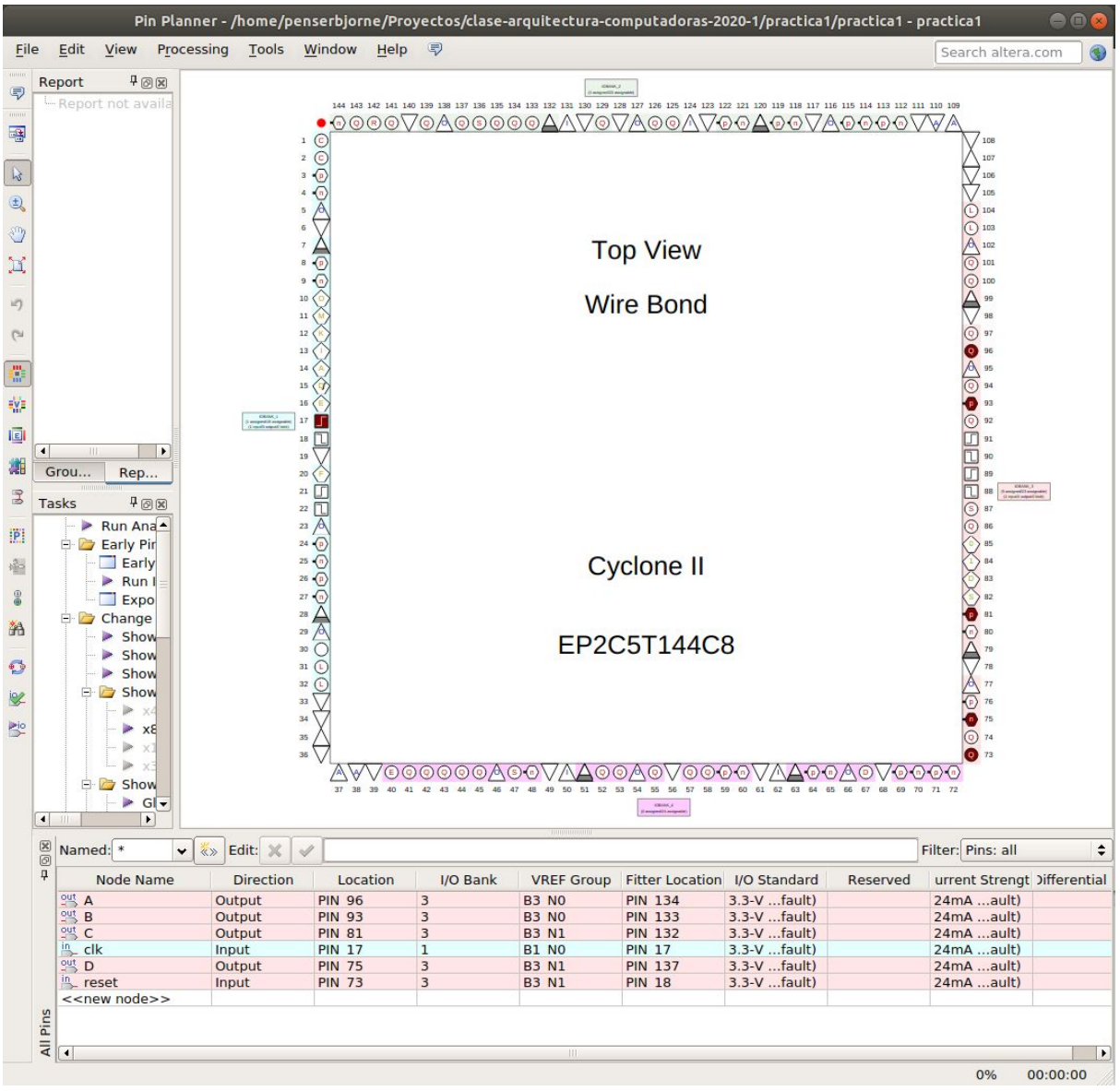

Figura 13: Asignación de los pines en la sección de PINOUT de Quartus.

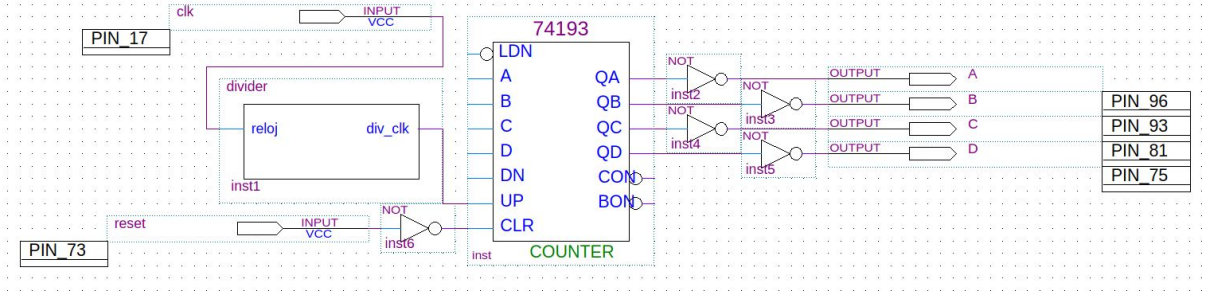

Figura 14: Asignación de los pines en el esquemático del contador binario.

Conclusiones

Los lenguajes de descripción de hardware junto a los FPGA son una excelente alternativa para el diseño e implementación de circuitos, sobre todo de circuitos digitales ya que reducen costos y tiempos en estas actividades, y permiten trabajar de manera modular y extensible.

En cuanto a la práctica, esta sirvió como tema introductorio al acercamiento de los lenguajes de descripción de hardware y los FPGA. La práctica fue sencilla por lo que seguir las intrucciones fue suficiente para poder realizarla sin mayor complicación.

Consideramos de gran utilidad estas tecnologías ya que aplicándolas a la ingeniería permiten extender y aplicar conocimiento que se encuentra limitado por cuestiones físicas, como es el uso de de CI o compuertas lógicas, ya sea por tamaños o escala de integración.

Referencias

- https://www.ecured.cu/Circuito_secuencial
- https://es.wikipedia.org/wiki/Field-programmable_gate_array
- https://es.wikipedia.org/wiki/Lenguaje_de_descripci%C3%B3n_de_hardware
- https://es.wikipedia.org/wiki/Quartus_II
- https://en.wikipedia.org/wiki/ModelSim