

Facultad de ingeniería

Materia: Laboratorio de Organización y
Arquitectura de Computadoras

Título: Practica 1 Introducción a las
herramientas de desarrollo de los FPGAs

Alumno: Martínez Pérez Brian Erik

Profesor: M. C. I. Julio Cesar Cruz Estrada

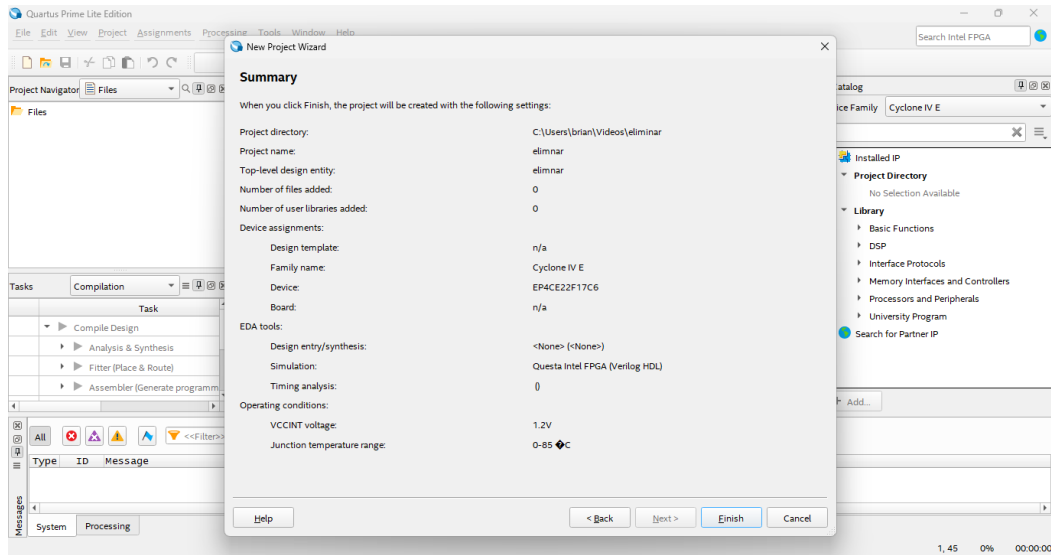
Grupo: 7

Semestre: 2026-1

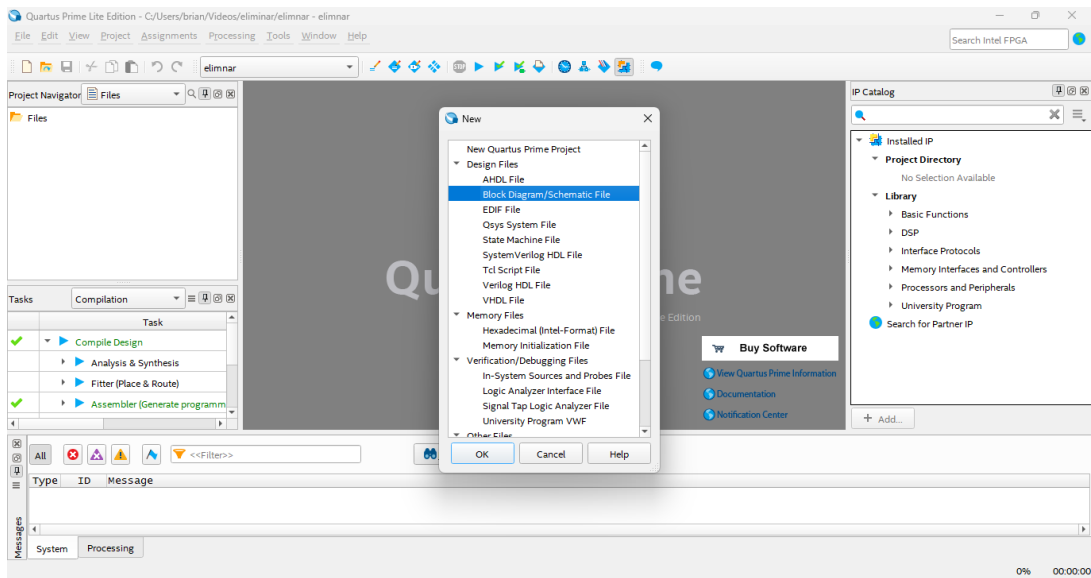


Desarrollo

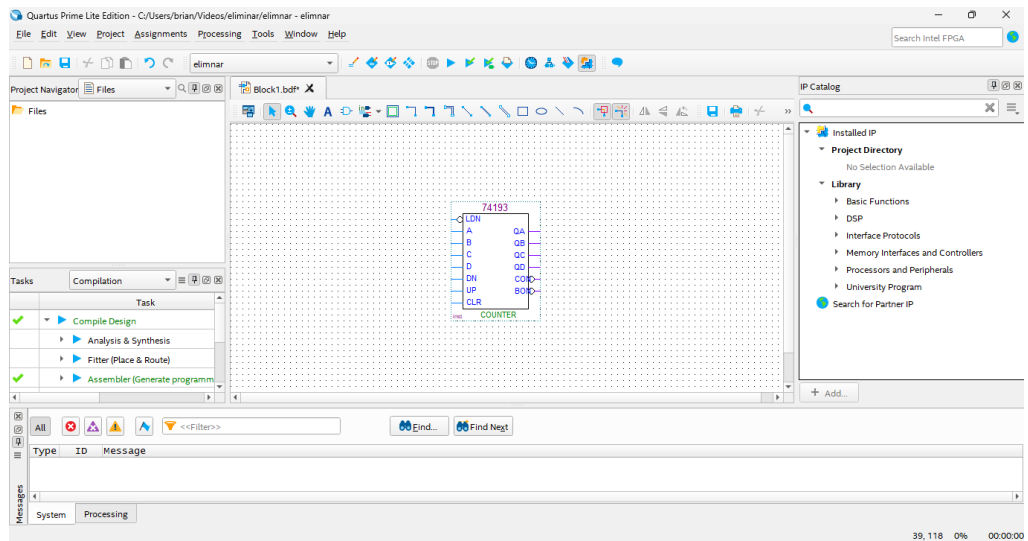
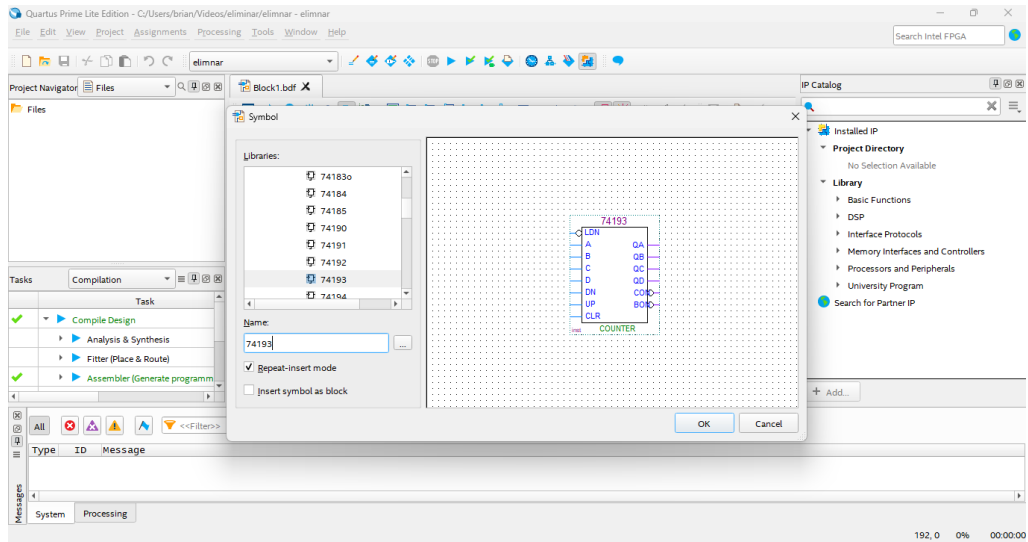
Creamos un nuevo proyecto en “FILE - New Project Wizard”, con la siguiente configuración:



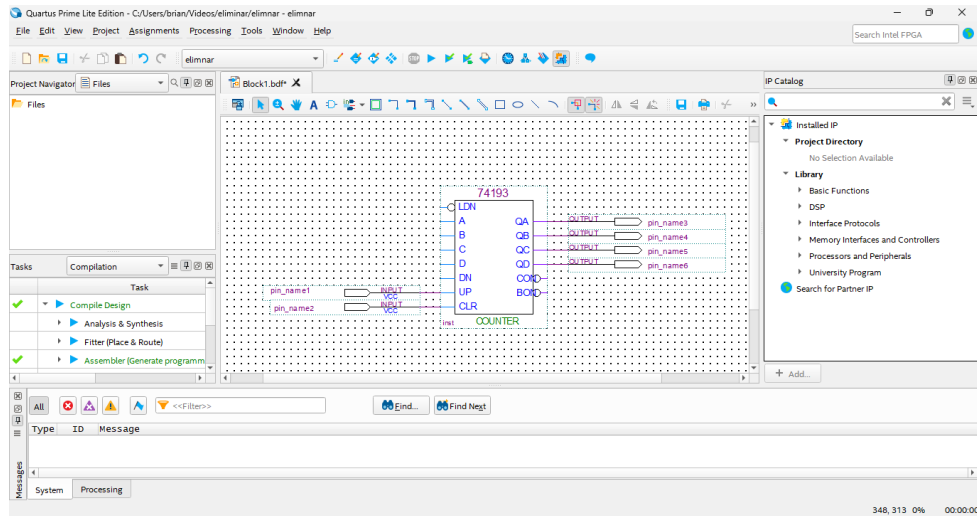
Una vez que tenemos creado el nuevo proyecto, creamos un nuevo archivo de tipo “Block diagram / Schematic file”.



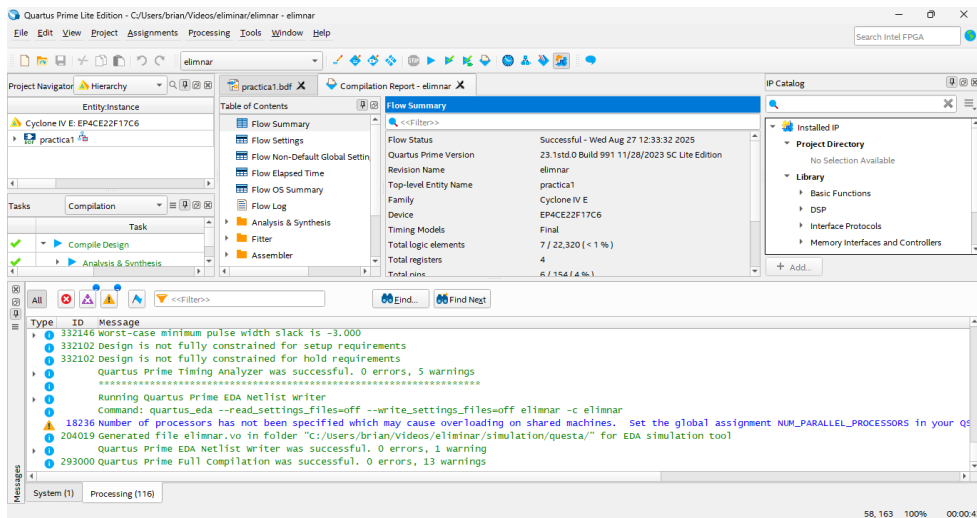
Agregamos el componente en la sección de “Simbol Tool”, ingresamos el numero “74193”, para encontrar el componente, posteriormente le damos en OK.



Agregamos las entradas y salidas del componente en la sección de “Pin Tool”, debemos de colocar 2 entradas y 4 salidas. Conectamos cada una de las entradas y de las salidas



Por último, compilamos y guardamos el programa, si tenemos errores el compilador nos lo mostrara en la sección inferior, de lo contrario la compilación llegara al 100%.



Divisor de frecuencia.

Para poder observar el encendido y apagados de los leds debemos crear un divisor de frecuencia, ya que el ciclo de reloj del CLK es muy rápido para la percepción del ojo humano.

Para poder implementar el divisor de frecuencia debemos obtener el valor del factor n para poder conseguir un cambio entre los leds de un segundo. El valor obtenido fue $n=2,500,000$.

$$F = 50 \text{ MHz}$$

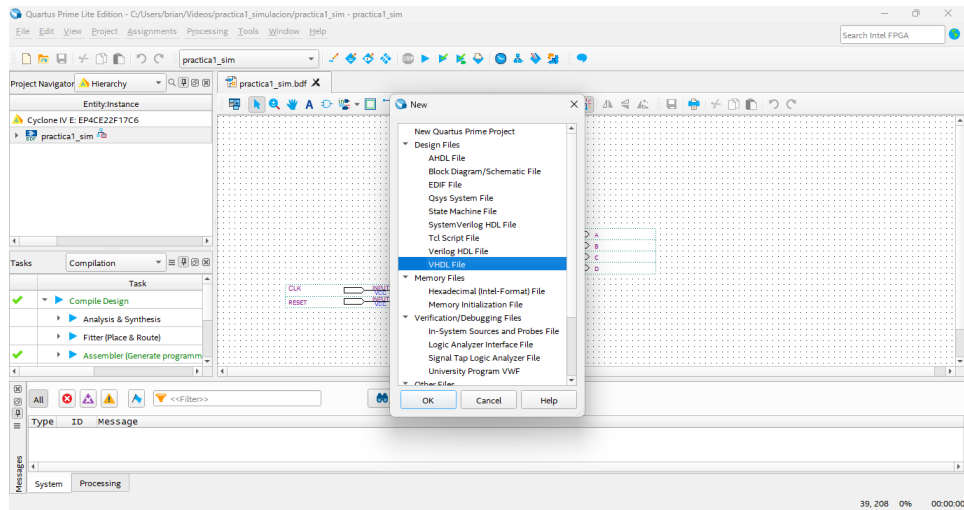
$$T = 20 \text{ ns} \Rightarrow 20 \times 10^{-8}$$

$$1 \text{ s} = n * 2 * 20^{-8}$$

$$n = 1 / (2 * 20^{-8})$$

$$n = 2,500,000$$

Ahora procederemos a crear un archivo nuevo de "VHDL File"



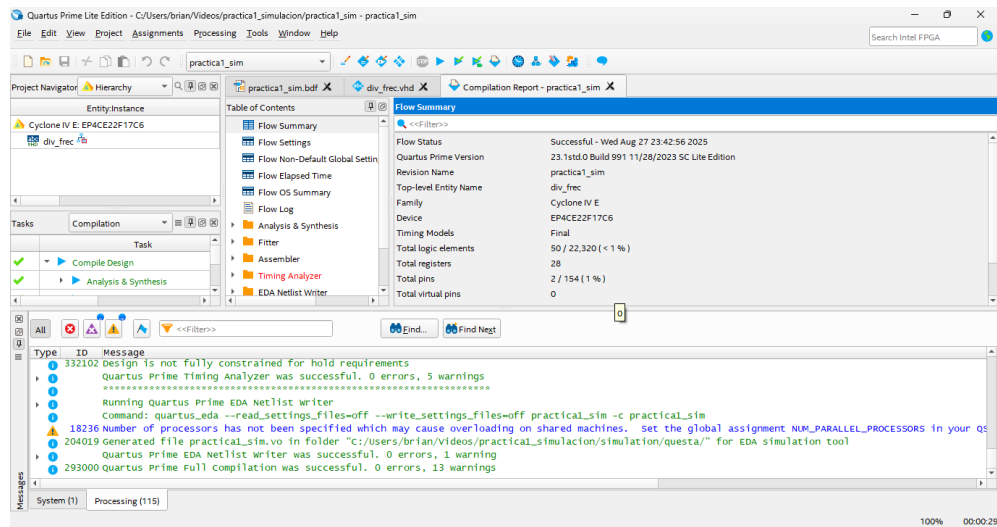
Obtenemos este código para el divisor de frecuencia solicitado.

```

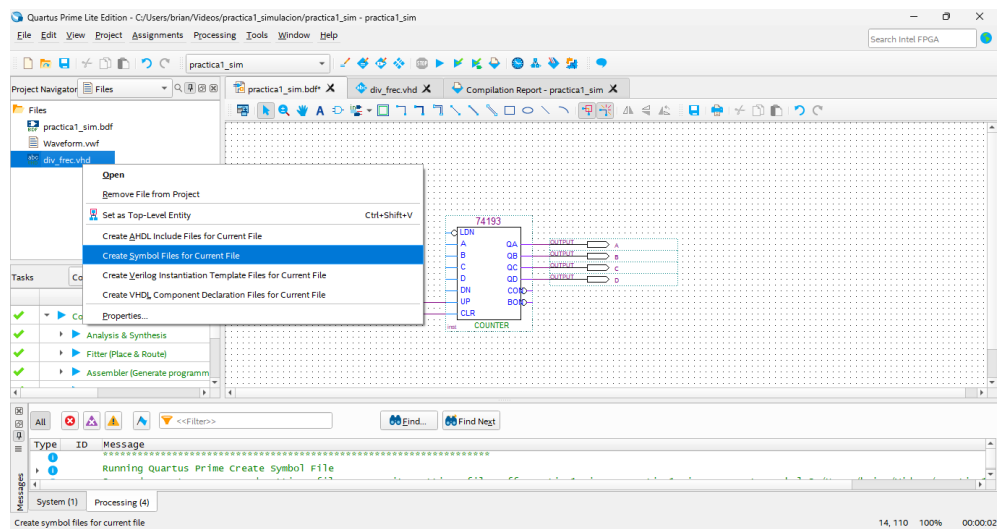
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity div_frec is
7      port ( reloj : in std_logic;
8            div_clk : out std_logic);
9  end div_frec;
10
11 architecture Behavioral of div_frec is
12 begin
13     process (reloj)
14         variable cuenta: std_logic_vector (27 downto 0):=x"0000000";
15     begin
16         if rising_edge (reloj) then
17             if cuenta = x"17D7840" then --25M pulsos
18                 cuenta:= x"0000000";
19             else
20                 cuenta:= cuenta+1;
21             end if;
22         end if;
23         div_clk <= cuenta(24);
24     end process;
25 end Behavioral;
26

```

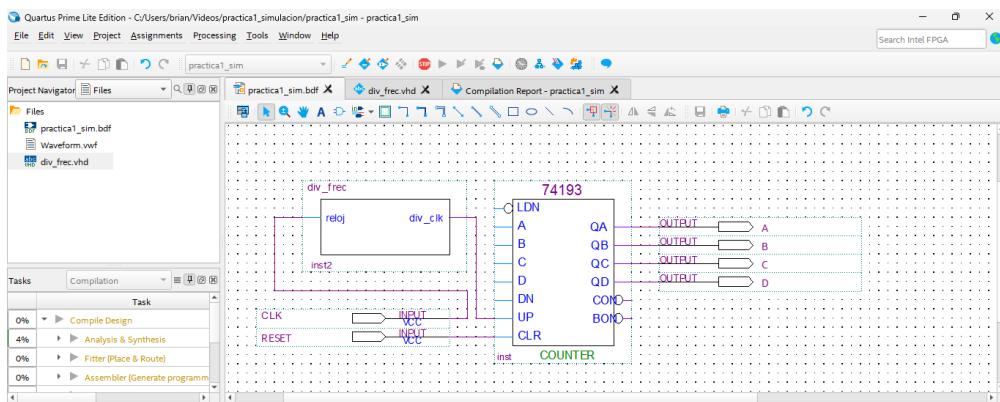
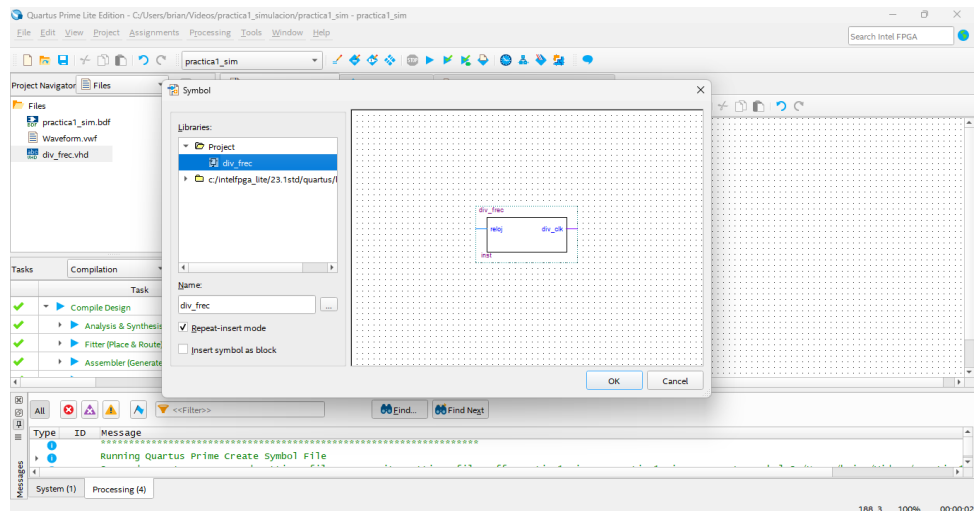
Después compilamos el código y si NO marca errores significa que podemos usar el código en nuestro contador.



Creamos el componente del bloque para poder usarlo en el contador



Ahora lo agregamos desde la sección de “Symbol Tool” y colocamos la entrada “CLK” a la entrada “reloj” del divisor de frecuencia y la salida “div_clk” a la entrada del contador “UP”.



Debemos compilar para saber que conectamos correctamente los bloques.

Flow Summary	
Flow Status	Successful - Thu Aug 28 00:04:28 2025
Quartus Prime Version	23.1std.0 Build 991 11/28/2023 SC Lite Edition
Revision Name	practical_sim
Top-level Entity Name	practical_sim
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	58 / 22,320 (< 1 %)
Total registers	32
Total pins	6 / 154 (4 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Command: quartus_eda --read_settings_files=off --write_settings_files=off practical_sim -c practical_sim

18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PARALLEL_PROCESSORS in your qsf file.

204019 Generated file practical_sim.vo in folder "C:/Users/brian/Videos/practical_simulation/simulation/questa/" for EDA simulation tool

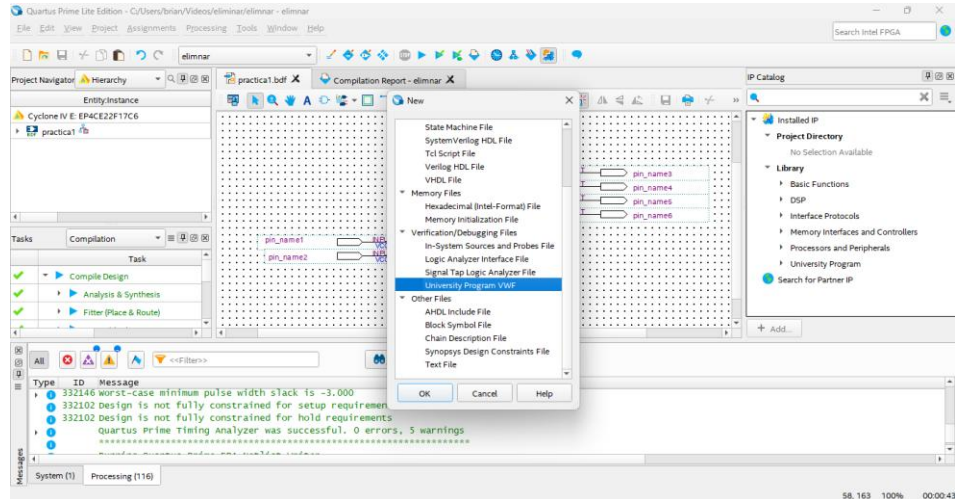
293000 Quartus Prime Full compilation was successful. 0 errors, 13 warnings

Ahora solo quedaría asignar pines y subir el archivo a la tarjeta FPGA que utilizaremos para poder ver el contador funcionando.

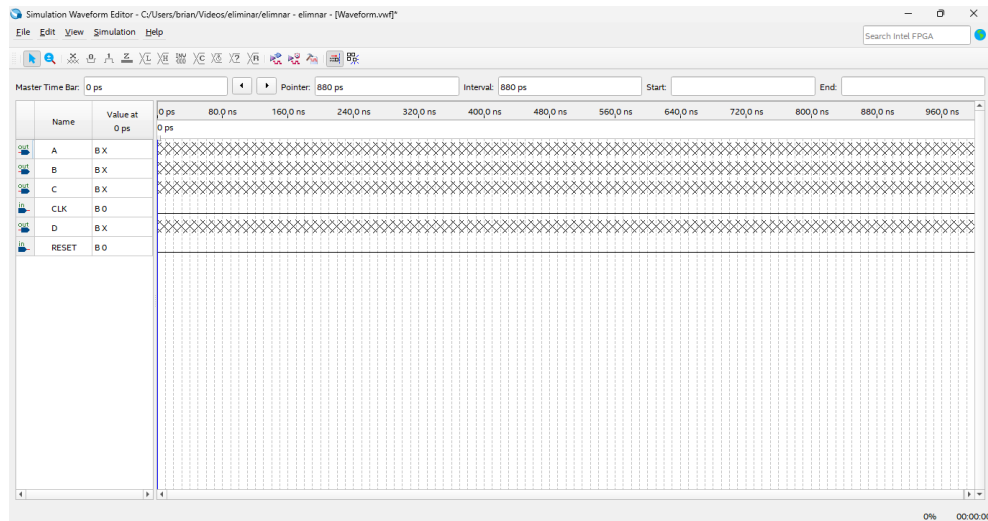
Simulaciones

Simulación con Waveform Editor

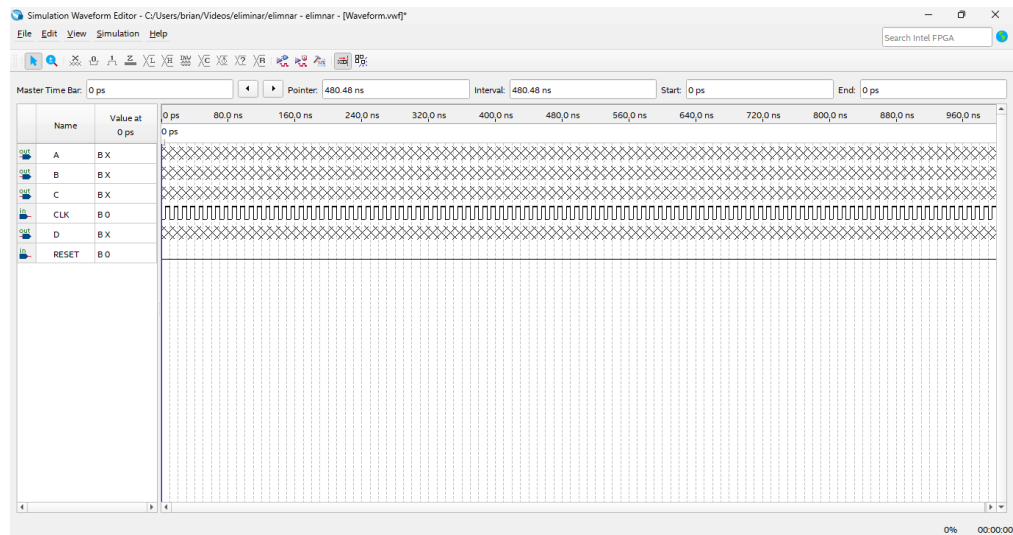
Creamos un nuevo archivo de tipo “University Program VWF”. Le damos en Ok.



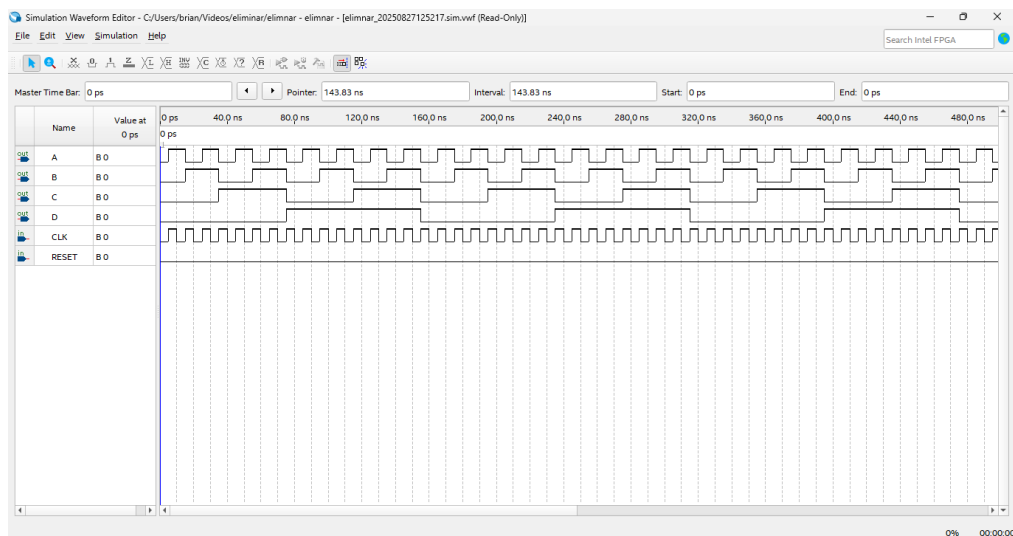
Agregamos todas nuestras variables en la barra de herramientas para que se muestren las señales de cada una de ella.



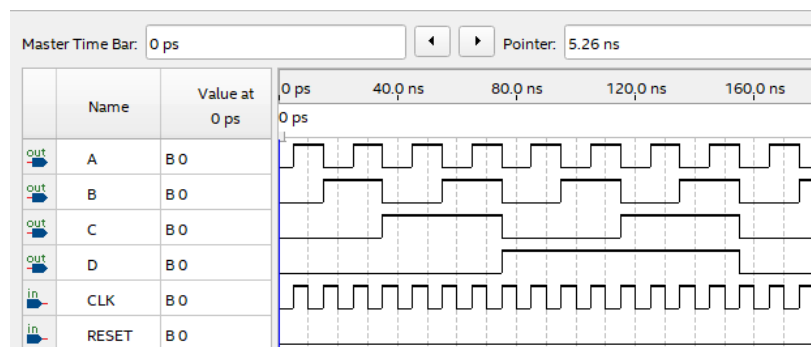
Damos una señal cuadrada al CLK y señal en “cero lógico” a RESET.



Corremos la simulación para ver como se comportan las salidas, respecto a las entradas que le dimos al contador.

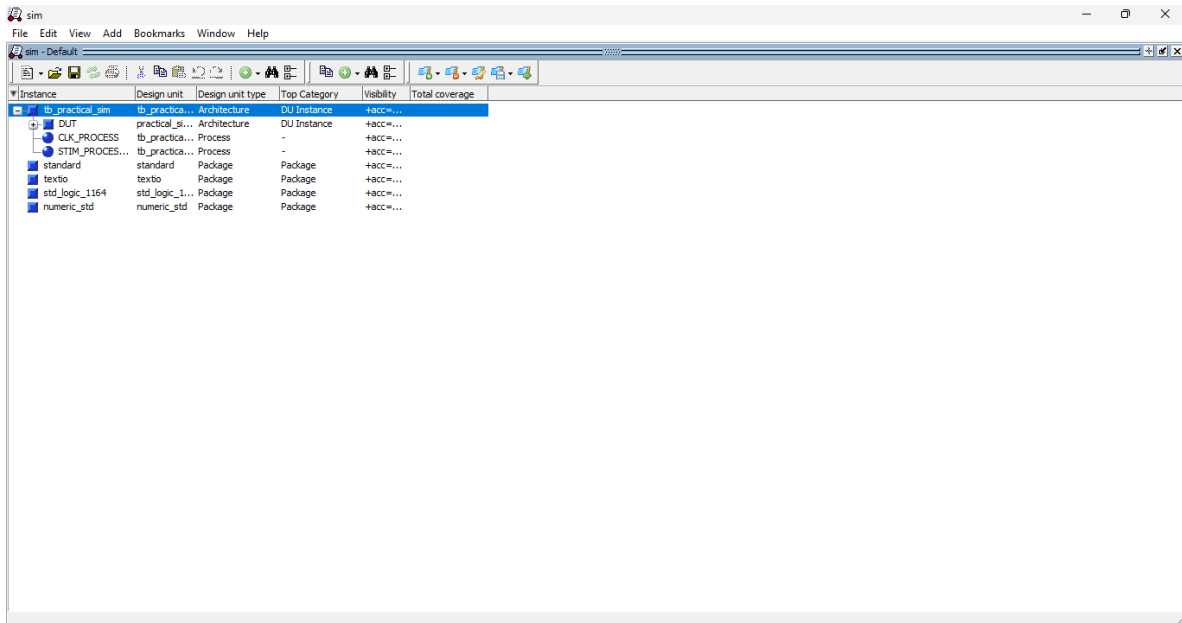


Podemos observar que realmente tenemos un contador de 4 bits, por lo que se mostrara la sucesión de numero binarios del 0 al 15.

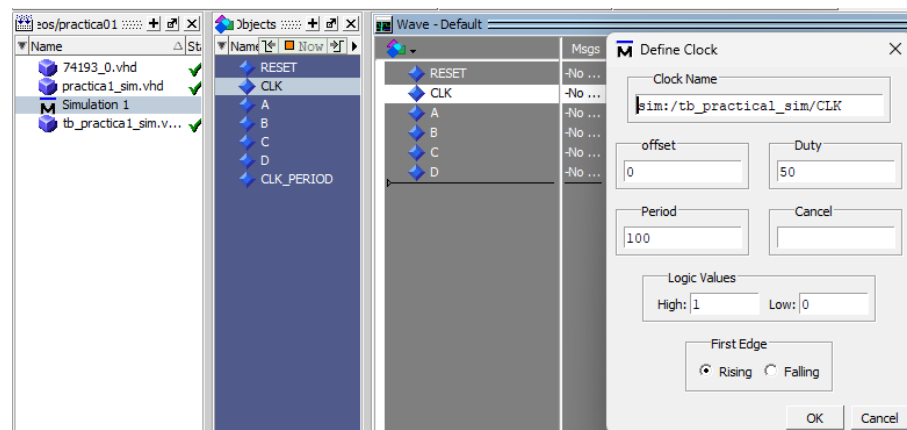
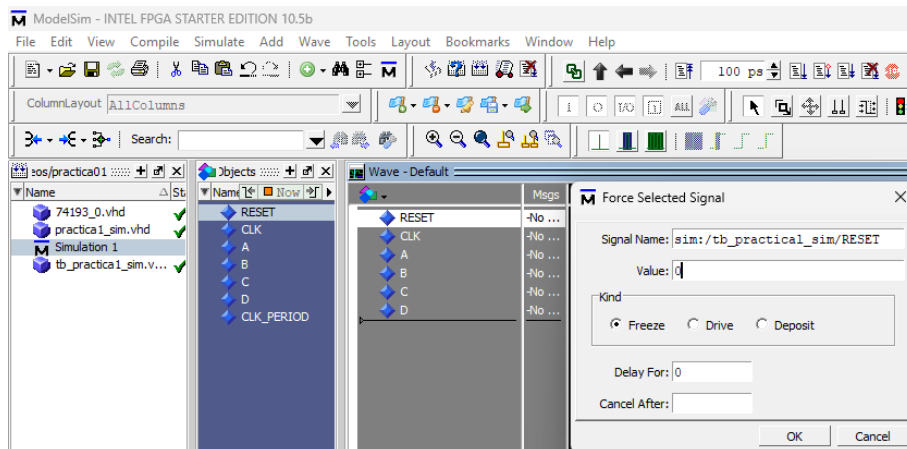


Simulación externa con ModelSim – Altera

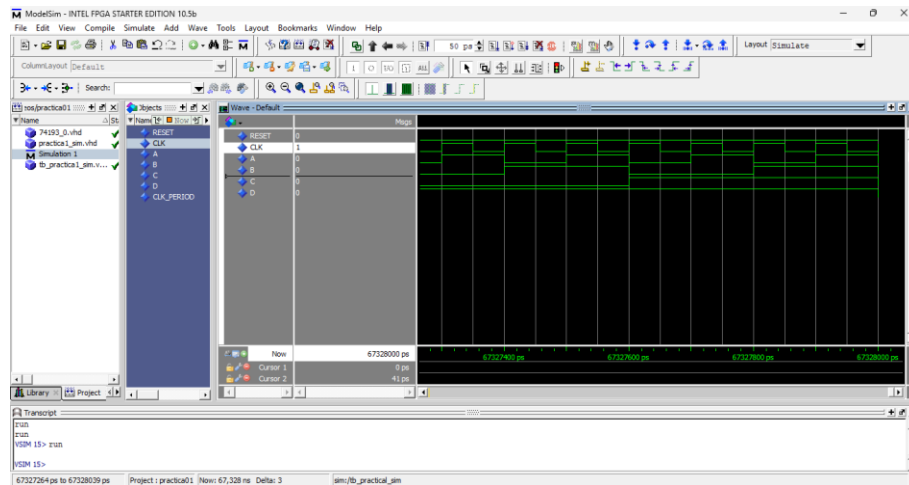
Pasamos el diagrama de bloque al simulador modelsim, una vez que la tenemos en la sección de objetos podemos asignar valores a nuestras entradas.



El valor de RESET lo forzamos a "0", y a CLK le ponemos el valor de clock.



Una vez que asignamos valores a las entradas podemos iniciar la simulación:



Podemos observar que, en el simulador externo, obtenemos los mismos resultados que en el simulador propio de Quartus. El contador realiza la sucesión de binaria del 0 al 15.

	Msgs		Msgs
RESET	0	RESET	0
CLK	1	CLK	1
A	0	A	1
B	0	B	0
C	0	C	0
D	0	D	0

	Msgs
RESET	0
CLK	1
A	1
B	1
C	1
D	1

Conclusión

La práctica introdujo el uso de Quartus Prime y ModelSim-Altera para FPGAs. Se configuró el entorno y se vinculó ambas herramientas correctamente. Se implementó un contador mediante diagramas esquemáticos. Se aprendió a simular con ModelSim y el editor de waveforms de Quartus. Se comprobó el funcionamiento del contador con reset y reloj. Adquirí habilidades esenciales en compilación y detección de errores. El proyecto quedó listo para programarse en la tarjeta FPGA. Los objetivos de familiarización con las herramientas se cumplieron exitosamente.