

TÉCNICAS Y DISPOSITIVOS DIGITALES II

Laboratorio 1: FPGA

Entrega del laboratorio: En esta [planilla](#) deberá incluirse el enlace al repositorio de GitHub correspondiente al trabajo realizado.

El repositorio deberá contener seis carpetas (una por cada parte del Laboratorio 1, de la A a la F) y un archivo .doc con las respuestas a las preguntas iniciales.

Cada carpeta deberá incluir el proyecto desarrollado para resolver la parte correspondiente del laboratorio.

Fecha límite de entrega del laboratorio: viernes 31 de octubre.

Temas: Introducción en el empleo de programación digital en FPGA.

Software: Quartus II versión 13.0

LINK descarga:

<https://drive.google.com/file/d/1TP38yhMcN5oDTF68ga1Au9g96NaeENuL/view?usp=sharing>

Hojas de datos de Cyclone III: "Cyclone III Device Handbook.pdf"

LINK: https://www.intel.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/cyc3/cyclone3_handbook.pdf

Responda:

- 1) Identifique qué elementos constituyen los LEs de la FPGA Cyclone III y qué estructura tienen las LABs
- 2) ¿De qué se trata el Nios® II?
- 3) ¿Qué diferencia existe entre IP cores y los bloques embebidos (ej multiplicador embebido) disponibles en la FPGA?
- 4) ¿Qué tipo de celda de programación posee el dispositivo FPGA Cyclone III?
- 5) Realice la descripción en VHDL de un Flip Flop JK.
- 6) Realice la descripción en VHDL de un sumador completo de un bit.
- 7) Realice la descripción en VHDL del test bench del sumador completo de un bit.

PARTE A: Implementación de un circuito combinacional en FPGA.

1) Utilice el entorno Quartus II para crear un nuevo proyecto.

AYUDA:

En el entorno Quartus:

File → New Project Wizard → Next

Elija el directorio del proyecto. (Conviene tener ya creada una carpeta donde se va a guardar todo lo de **ESTE** proyecto, ya que crea muchos archivos por proyecto)

Elija el nombre del proyecto.

En Add files → next (no agregamos ningún archivo)(Esto se utiliza cuando quiero levantar archivos ya creados)

Family & device settings: Family: Cyclone III , device: EP3C120F780C7 → next

→ next

→ Finish

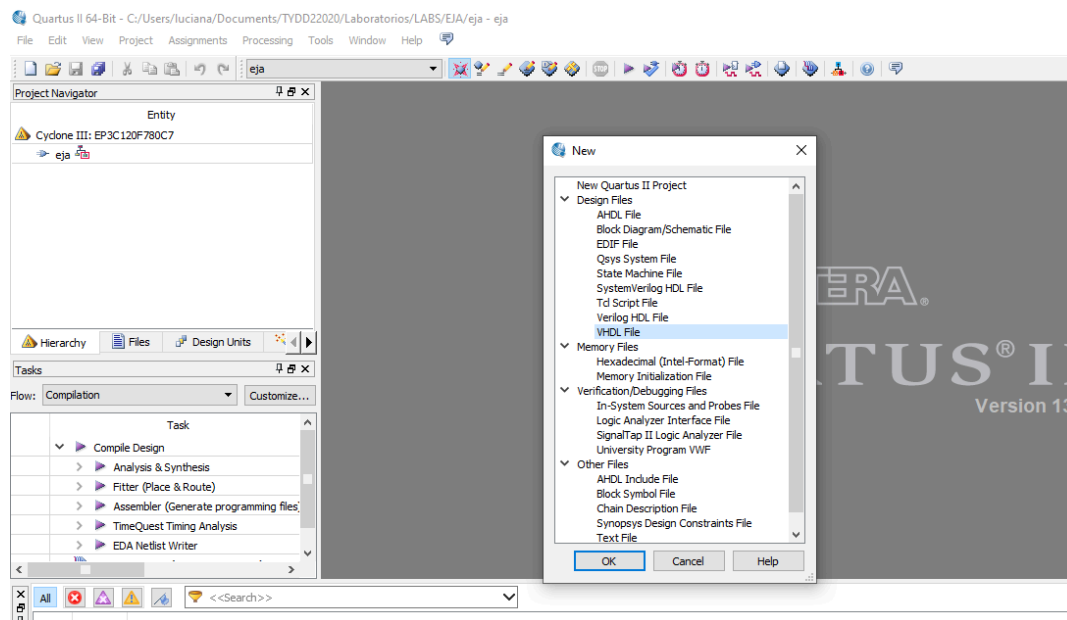
Name	Core Voltage	LES	User I/Os	Memory Bits	Embedded multiplier 9-bit elements	PLL	Clk
EP3C120F780C7	1.2V	119088	532	3981312	576	4	20

2) En el proyecto genere un archivo VHDL.

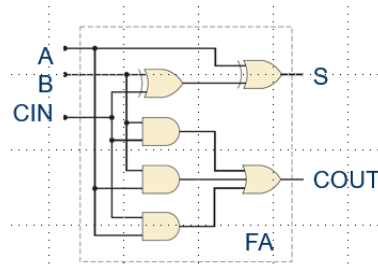
AYUDA:

En el entorno Quartus:

File → new → VHDL file → OK



3) En el archivo VHDL describa el siguiente circuito combinacional, guárdelo, seteelo como Top Level entity y compile.



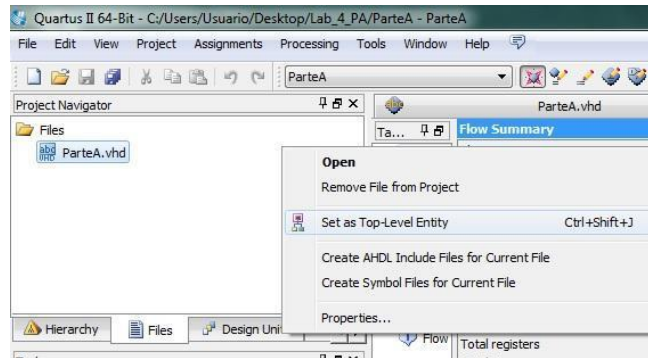
AYUDA 1:

En el archivo VHDL deberá agregar el paquete IEEE.STD_LOGIC_1164.ALL de la librería IEEE, el cual contiene definiciones de tipos, subtipos y funciones.

AYUDA 2:

Para definir al archivo VHDL creado como Top Level entity:

En el Project Navigator vaya a la solapa Files, donde podrá ver el archivo VHDL creado. Seleccionando el archivo con el botón derecho del mouse podrá setearlo como "Set as Top-Level Entity".



AYUDA 3:

Para compilar, primero guarde el archivo VHDL y luego en Processing Start Compilation. (o CTR+L)

IMPORTANTE!!! El nombre de la entidad y el nombre del archivo vhd deben ser iguales y sin espacios.

Ejemplo archivo ej_combinacional.vhd
(Atención, es otro circuito solo para ejemplificar).

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ej_combinacional is
    Port ( SW1 : in  STD_LOGIC;
          SW2 : in  STD_LOGIC;
          LED : out STD_LOGIC);
end ej_combinacional;

architecture Behavioral of ej_combinacional is
begin
    LED <= not (SW1 and SW2);
end Behavioral;
```

4) Asigne los pines de entrada y salida del diseño. Compile nuevamente.

Asigne en este caso a las entradas A, B y CIN un switch a cada una y las salidas S y COUT a leds.

AYUDA:

Busque los números de pines (necesitará tres switches de entrada y dos leds de salida) en el Manual de referencia "Cyclone III 3C120 Development Board", págs. 37 y 38.

(https://www.intel.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/rm_cycloneiii_dev_kit_host_board.pdf)

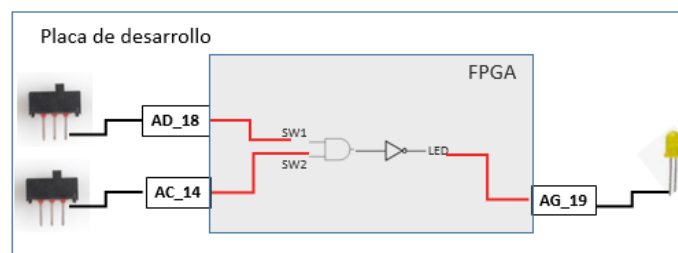
También está en la pág. de la cátedra archivo rm_cycloneiii_dev_kit_host_board.pdf.

Luego, en el entorno Quartus: Assignments Pin Planner Location

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength
LED	Output	PIN_AG19	4	B4_N2	PIN_AG19	2.5 V (default)		8mA (default)
SW1	Input	PIN_AD18	4	B4_N1	PIN_AD18	2.5 V (default)		8mA (default)
SW2	Input	PIN_AC14	3	B3_N0	PIN_AC14	2.5 V (default)		8mA (default)
<<new node>>								

NOTA: No olvide compilar nuevamente. La ventana emergente en la que asigna los pines puede cerrarse sin problemas, lo que se compila nuevamente es el archivo que está seteado como Top-level. O sea, el principal. No hay opción de guardado en la ventana de asignación de pines.

Este paso genera las conexiones rojas que se ven en el siguiente diagrama:



5) Verifique el circuito implementado mediante el visor de RTL.

AYUDA:

En el entorno Quartus:

Tools Netlist Viewer RTL view

6) Verifique el circuito implementado luego del mapeo en el dispositivo.

AYUDA:

En el entorno Quartus:

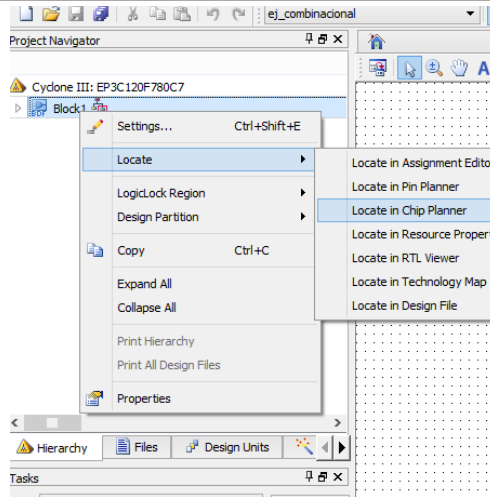
Tools Netlist Viewer Technology Map Viewer (past- Mapping y post-Fitting)

7) Verifique la implementación en el chip.

AYUDA:

En el entorno Quartus:

En la ventana: Project Navigator, solapa: Hierarchy, click derecho Locate Locate in Chip Planner.



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

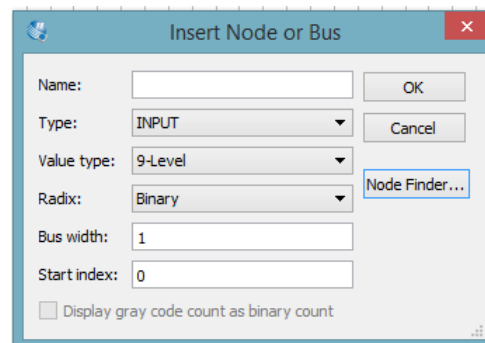
AYUDA:

En el entorno Quartus:

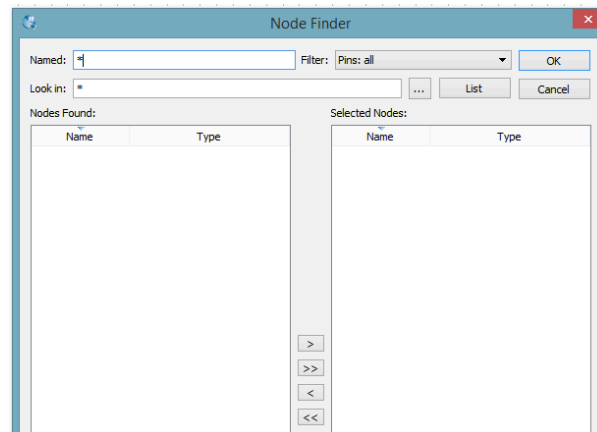
Cree un archivo de Estímulos (.vwf): File → New → University Program VWF

Guarde el archivo en el mismo directorio.

Agregue las señales de entrada y salida a simular: Edit → Insert → Insert Node or Bus

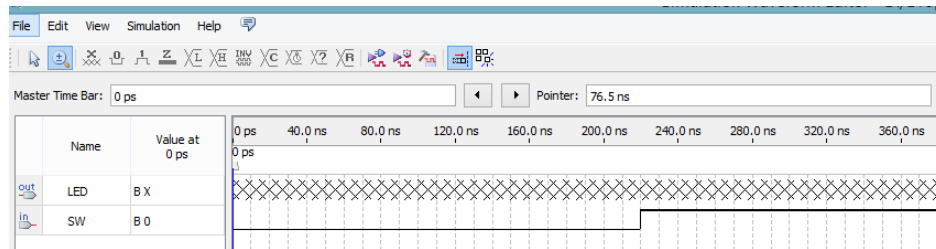


Node Finder → List



Agregue las señales de entrada y salida que se deseen evaluar, luego OK.

En el Editor de forma de onda de Simulación (Simulation Waveform Editor):
Asigne valores a las entradas.



Realice la simulación funcional: Simulation → Run Functional Simulation.

Realice la simulación temporal: Simulation → Run Timing Simulation.

9) Realice la simulación del circuito mediante la simulación de Modelsim.

Escriba un archivo testbench VHDL, guardelo y simúlelo en ModelSim.

Para simular en Modelsim ⇒ CONFIGURE EL QUARTUS SEGÚN EL ARCHIVO AYUDA_CONFIGURACION_MODELSIM.PDF (2 configuraciones).

- Configuración del Quartus para simular con ModelSim **1**, tiene en cuenta nombres de archivos y de entidad de los archivos de testbench, asique primero haga el testbench, guardelo y luego realice el paso de configuracion 1. **Esta configuración se debe realizar en cada proyecto que desee simular!**
- Configuración del Quartus para simular con ModelSim **2** se hace una sola vez y queda seteado. Es decir, es una configuración general para que funcione la herramienta.

AYUDA:

En el entorno Quartus:

Cree archivo VHDL:

File → new → VHDL file

y escriba el archivo testbench.

Ejemplo (sólo para ejemplificar, es otro circuito):

```

ARCHITECTURE behavior OF ej_combinacional_testbench IS

    COMPONENT ej_combinacional --component declaration
    PORT(
        SW1 : in  STD_LOGIC;
        SW2 : in  STD_LOGIC;
        LED : out STD_LOGIC
    );

    END COMPONENT;

    --Inputs
    signal SW1 : std_logic := '0';
    signal SW2 : std_logic := '0';
    --Outputs
    signal LED : std_logic;

    BEGIN
        -- Instantiate the Unit Under Test (UUT)
        uut: ej_combinacional PORT MAP (
            SW1 => SW1,
            SW2 => SW2,
            LED => LED
        );

        stim_proc: process -- Stimulus process
        begin
            --stimulus
            SW1 <= '0'; SW2 <= '0'; wait for 10ns;
            SW1 <= '0'; SW2 <= '1'; wait for 10ns;
            SW1 <= '1'; SW2 <= '0'; wait for 10ns;
            SW1 <= '1'; SW2 <= '1'; wait for 10ns;

            wait;
        end process;
    END;

```

Setee el archivo testbench a emplear por ModelSim ⇒ Este paso de seteo tiene que ver con los nombres que debe colocar en los pasos de configuración Modelsim 1.

Luego realice la simulación funcional:

Desde Quartus: Tools⇒Run Simulation Tool⇒RTL Simulation

Luego realice la simulación temporal:

Desde Quartus: Tools⇒Run Simulation Tool⇒ Gate Level Simulation

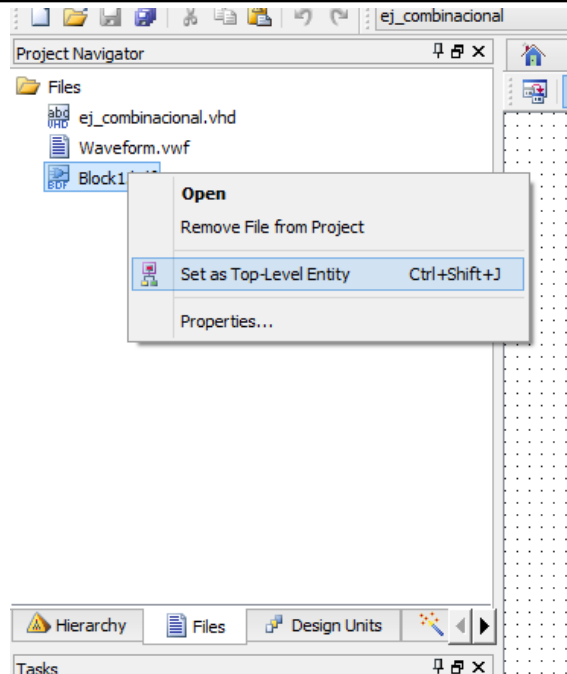
10) En el mismo proyecto genere un archivo esquemático.

AYUDA:

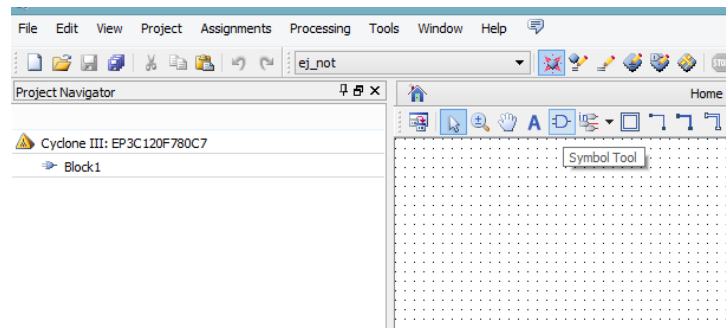
En el entorno Quartus:

File ⇒ new ⇒ Block Diagram/Schematic File

Setee este archivo como entidad Top-Level: Click derecho sobre el archivo y seleccione "Set as Top Level-Entity".



10.a En el archivo esquemático agregue los símbolos de los componentes necesarios (Symbol Tool), realice las conexiones necesarias.




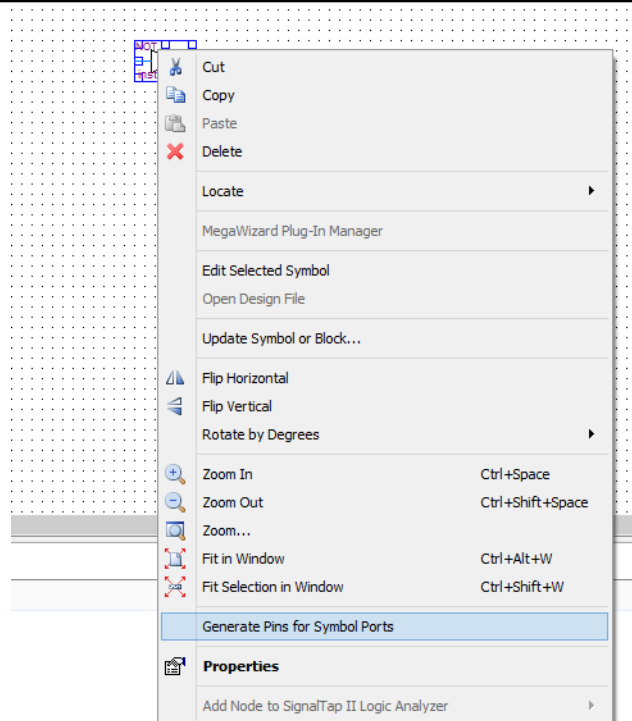
Genere los puertos de entrada y salida.

AYUDA:

En el entorno Quartus:

Automáticamente Quartus genera los puertos, luego haciendo doble click sobre cada uno es posible modificar el nombre asignado.

Click derecho sobre el componente  Generate Pins for Symbol Ports.



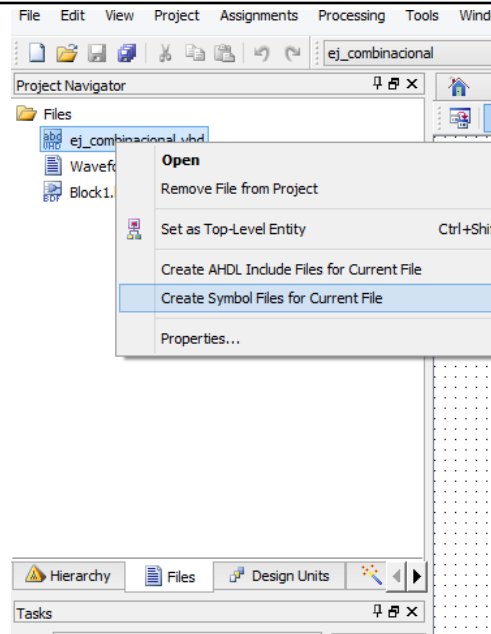
Asigne los pines de entrada y salida del diseño. Compile nuevamente.
Verifique.
Simule.

10.b Genere un símbolo esquemático a partir del archivo ej_combinacional.vhd del punto anterior:

AYUDA:

En el entorno Quartus:

En la ventana: Project Navigator, solapa: Files, click derecho sobre el archivo → Create Symbol Files for Current File.



Agregue el componente simbólico al archivo esquemático.
Compile.
Verifique.
Simule.

PARTE B: Implementación de un multiplicador de 2 bits sin signo empleando el entorno esquemático.

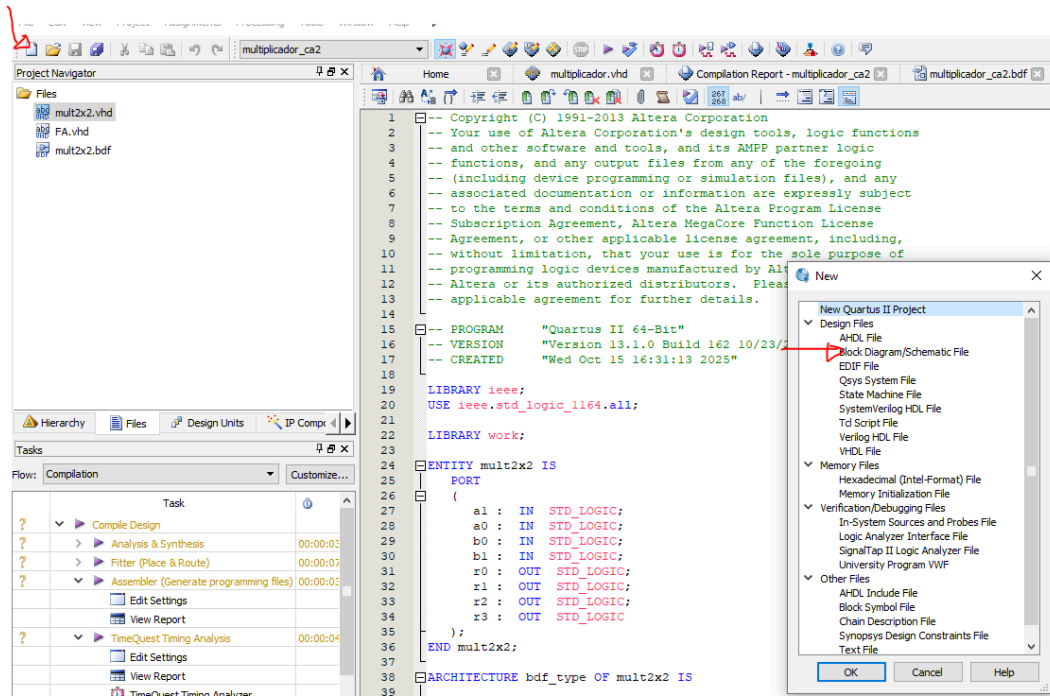
A partir del funcionamiento de un multiplicador de números sin signo, cree en Quartus II un proyecto en el cual implemente un multiplicador sin signo.

		a_1	a_0
	\times	b_1	b_0
<hr/>			
		a_1b_0	a_0b_0
	a_1b_1	a_0b_1	-
<hr/>			
	a_1b_1	$a_1b_0 + a_0b_1$	a_0b_0
<hr/>			
r_3	r_2	r_1	r_0

Entradas 2 bits del operando a, 2 bits del operando b. Salida 4 bits del resultado r.

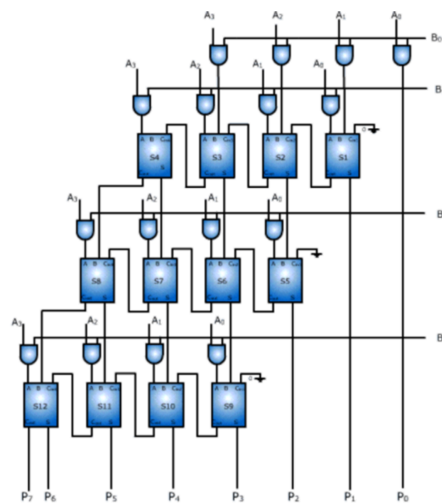
Realice los siguientes pasos:

- 1) Utilice el entorno Quartus II para crear un nuevo proyecto.
- 2) En el proyecto genere un archivo esquemático.



3) Ingrese las compuertas necesarias para implementar el circuito.

AYUDA: Ejemplo para multiplicaciones de operandos de 4 bits sin signo.



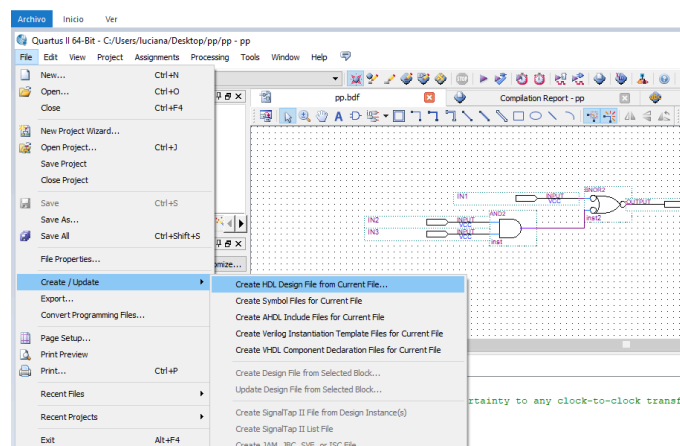
Requerirá compuertas AND y sumadores completos. Para ello utilice el sumador completo desarrollado en el inciso A.

Agregue el vhdl del sumador completo en el proyecto. Luego genere un símbolo esquemático a partir del archivo. Una vez generado aparecerá en Symbol tool para poder agregarlo al circuito esquemático.

- 4) Defina como Top Level al archivo esquemático. Compile.
- 5) Asigne los pines de entrada y salida del diseño. Compile nuevamente.
- 6) Verifique el circuito implementado mediante el visor de RTL.
- 7) Verifique el circuito implementado luego del mapeo en el dispositivo.
- 8) Verifique la implementación en el chip.
- 9) Verifique el funcionamiento funcional y temporal del circuito mediante la simulación de Quartus.
- 10) Verifique el funcionamiento del circuito mediante la simulación de Modelsim.

AYUDA: Primero deberá generar el archivo VHDL de su esquemático ya que el Modelsim no simula esquemáticos. **Para solucionarlo:** Convertir el esquemático en VHDL pararse en el archivo esquemático (doble click) en la solapa "Files".

y elegir File=>create/update=>create HDL file from current file



Esto crea un archivo .vhdl en la carpeta donde se está trabajando con el mismo nombre del esquemático. Sacar el esquemático, agregar el archivo vhd, compilar y ahí es posible simular con Modelsim.

Ejemplo testbench para sumador completo:

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- ENTITY sumador_completo_testbench IS
-- END sumador_completo_testbench;

-- ARCHITECTURE behavior OF sumador_completo_testbench IS
--
-- COMPONENT sumador_completo --component declaration
-- PORT(
--     in_a      : in STD_LOGIC;
--     in_b      : in STD_LOGIC;
--     in_cin    : in STD_LOGIC;
--     clk       : in STD_LOGIC;
--     o_f       : out STD_LOGIC;
--     o_cout    : out STD_LOGIC
-- );
-- END COMPONENT;

--Inputs
signal in_a : std_logic := '0';
signal in_b : std_logic := '0';
signal in_cin : std_logic := '0';
signal clk : std_logic := '0';

--Outputs
signal o_f : std_logic;
signal o_cout : std_logic;

--Clock period definition
constant clock_period : time := 20ns;

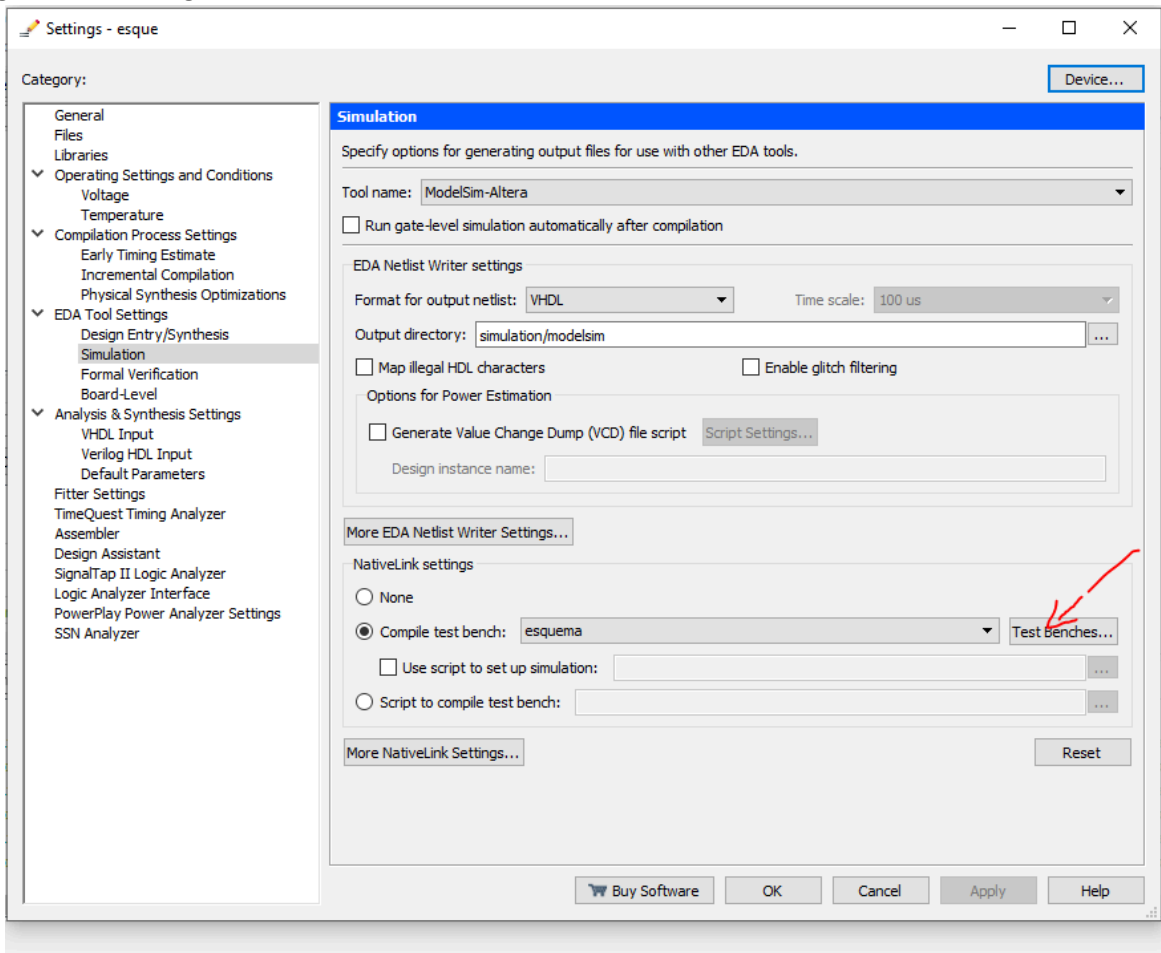
BEGIN
    -- Instantiate the Unit Under Test (UUT)
    uut: sumador_completo PORT MAP (
        in_a => in_a,
        in_b => in_b,
        in_cin => in_cin,
        clk => clk,
        o_f => o_f,
        o_cout => o_cout
    );

    --Clock process definitions
    clock_process: process
    begin
        clk<='0';
        wait for clock_period/2;
        clk<='1';
        wait for clock_period/2;
    end process;

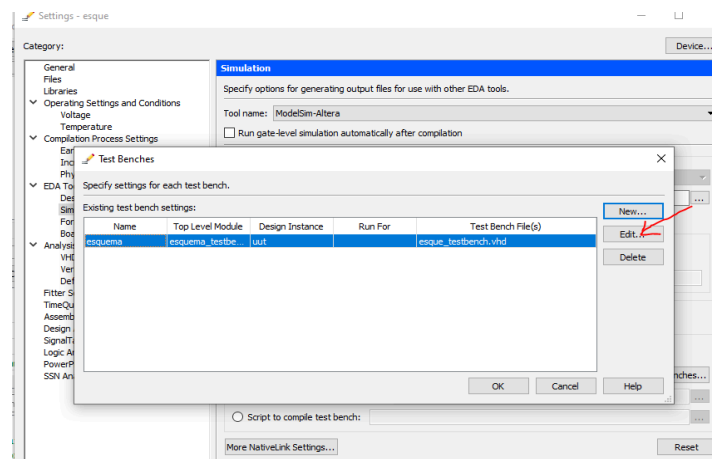
    stim_proc: process -- Stimulus process
    begin
        --stimulus
        in_a <= '0'; in_b <= '0'; in_cin <= '0'; wait for 30ns;
        in_a <= '0'; in_b <= '0'; in_cin <= '1'; wait for 30ns;
        in_a <= '0'; in_b <= '1'; in_cin <= '0'; wait for 30ns;
        in_a <= '0'; in_b <= '1'; in_cin <= '1'; wait for 30ns;
        in_a <= '1'; in_b <= '0'; in_cin <= '0'; wait for 30ns;
        in_a <= '1'; in_b <= '0'; in_cin <= '1'; wait for 30ns;
        in_a <= '1'; in_b <= '1'; in_cin <= '0'; wait for 30ns;
        in_a <= '1'; in_b <= '1'; in_cin <= '1'; wait for 30ns;
        wait;
    end process;
END;

```

Assignemts=>Settings=>simulation=>

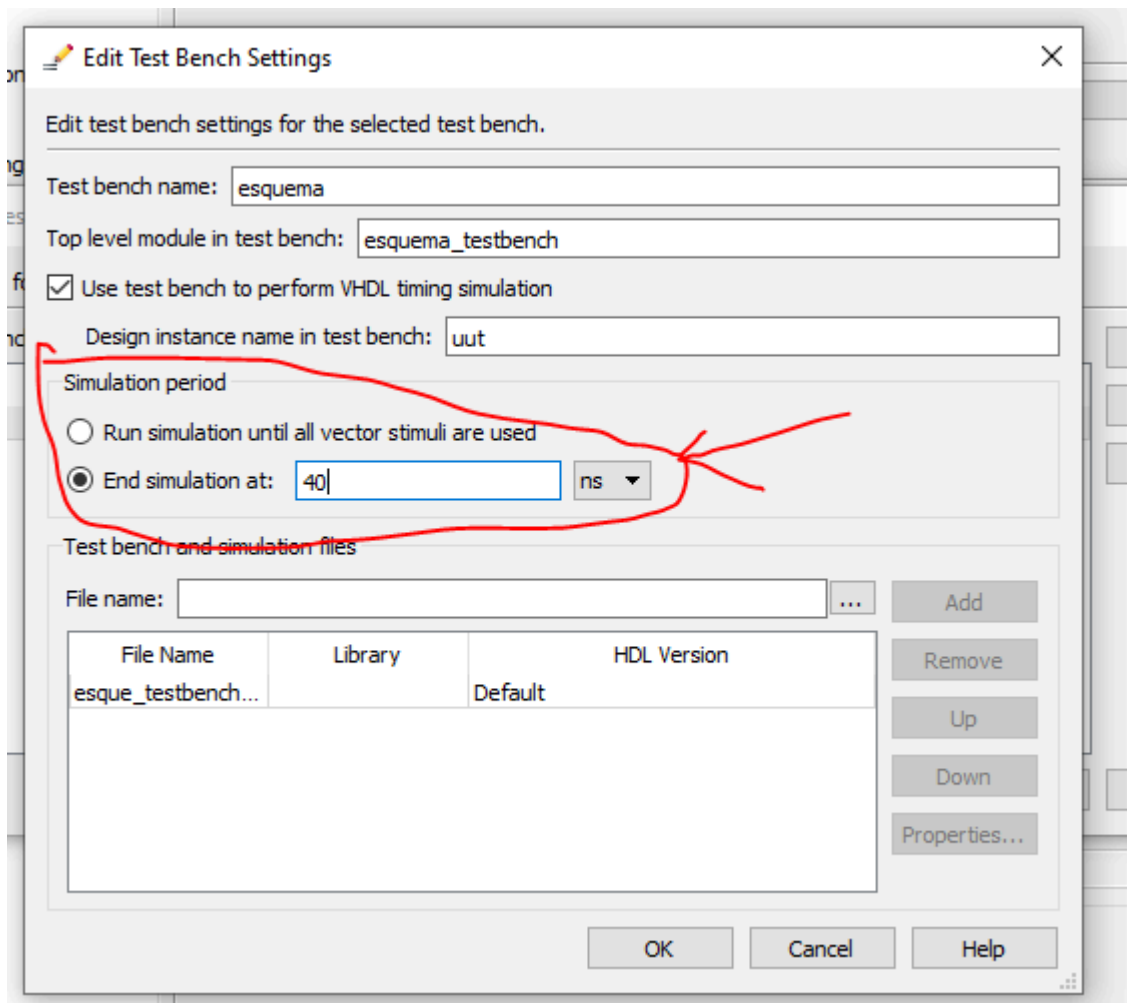


Compile testbench, boton test benches, ahi agregan el archivo testbench o editar el que ya se tenga cargado



Configuración del tiempo de simulación:

Se puede elegir un tiempo determinado o que se simule mientras haya estímulos:



PARTE C: Implementación de un multiplicador con signo

A partir del multiplicador de 2 bits del punto anterior, cree en Quartus II un proyecto esquemático en el cual se implemente un multiplicador de 2 bits en complemento a 2 y las banderas correspondientes. Latchee las señales de entrada y salida mediante Flip-Flops D.

		a_1	a_0
	\times	b_1	b_0
a_1b_0	a_1b_0	a_1b_0	a_0b_0
a_1b_1	c_1b_1	c_0b_1	-
$a_1b_0 + a_1b_1$	$a_1b_0 + c_1b_1$	$a_1b_0 + c_0b_1$	a_0b_0
r_3	r_2	r_1	r_0

Figura 2

Donde a y b son los operandos expresados en complemento a 2, r es el resultado y c es el complemento a 2 del operando a.

Note que los operandos pueden tomar valores en el rango -2 a 1, pero la salida debe ser de 4 bits por lo que el rango es mayor.

Realice los siguientes pasos:

- 1) Abra el entorno Quartus II y cree un nuevo proyecto (o bien realice una copia del proyecto anterior para conservar la configuración previa).
- 2) Dentro del proyecto, genere un nuevo archivo esquemático y defínalo como Top-Level Entity (o modifique directamente el esquemático del proyecto duplicado).
- 3) Incorpore la circuitería adicional necesaria para que la multiplicación sea correcta en complemento a dos, incluyendo la extensión de signo y el complemento del último término.

Tenga en cuenta que, si el operando a es -2 (representado como 10 en binario) y se multiplica por un número negativo ($b < 0$), al complementar se vuelve a obtener 10. En este caso, el resultado es positivo, por lo que no debe repetirse el bit más significativo de a complementado durante la extensión del signo, sino que debe extenderse con cero, tal como se muestra en la Figura 2.

- 4) Agregue FFD en las entradas y salidas del circuito, todos comandados por una entrada de clock.

AYUDA:

Puede usar el ffd que provee el Quartus en Simbol tool=> primitives=> storage=> dff

- 5) Compile.
- 6) Asigne los pines de entrada y salida del diseño. Compile nuevamente.
- 7) Verifique el circuito implementado mediante el visor de RTL.
- 8) Verifique el circuito implementado luego del mapeo en el dispositivo.
- 9) Verifique la implementación en el chip.
- 10) Verifique el funcionamiento funcional y temporal del circuito mediante la simulación de Quartus.
- 11) Verifique el funcionamiento del circuito mediante la simulación de Modelsim.

Recuerde que Modelsim no soporta circuitos esquemáticos. Convertir el esquemático en VHDL como en la parte B.

PARTE D: Implementación de multiplicador en módulo y en ca2 en VHDL.

Implemente un circuito en VHDL que sea capaz de realizar la multiplicación de dos operandos de 2 bits en ambos sistemas de representación:

- En módulo: utilizando entradas y salidas de tipo unsigned.
- En complemento a dos (CA2): utilizando entradas y salidas de tipo signed.

El circuito deberá generar, para cada caso, un resultado de 4 bits correspondiente al producto de los operandos.

Desarrolle la descripción del circuito en VHDL, implementando las operaciones en paralelo para ambos tipos de datos (unsigned y signed).

Verifique el correcto funcionamiento del diseño mediante simulación, comprobando que los resultados obtenidos coincidan con los valores esperados en cada sistema de representación.

PARTE E: Implementación de una máquina de estado.

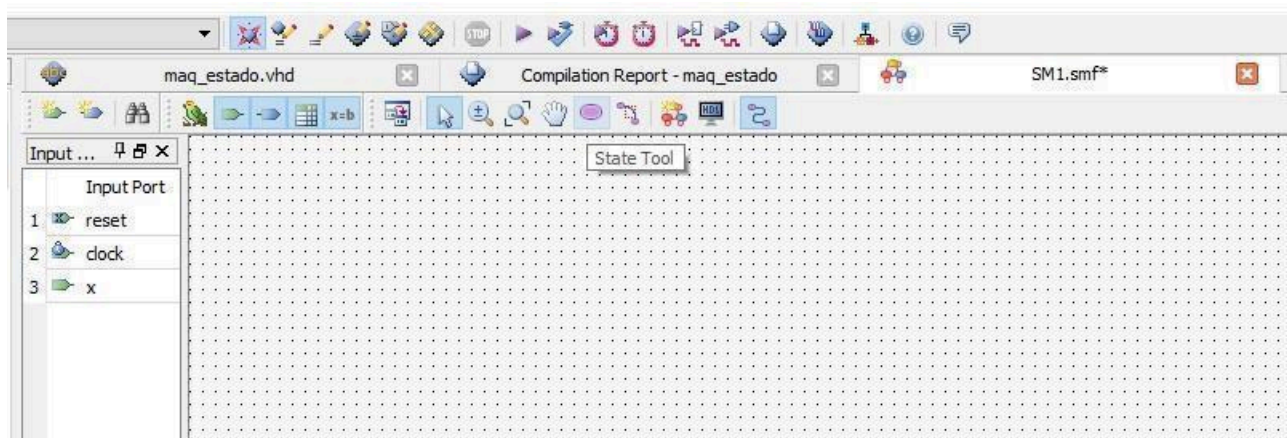
Implemente las máquinas de estados de los ejercicios 2 (secuencia de luces) de la Guía 4.

Implemente mediante la herramienta State Tool y mediante la inserción de un Template de máquina de estado:

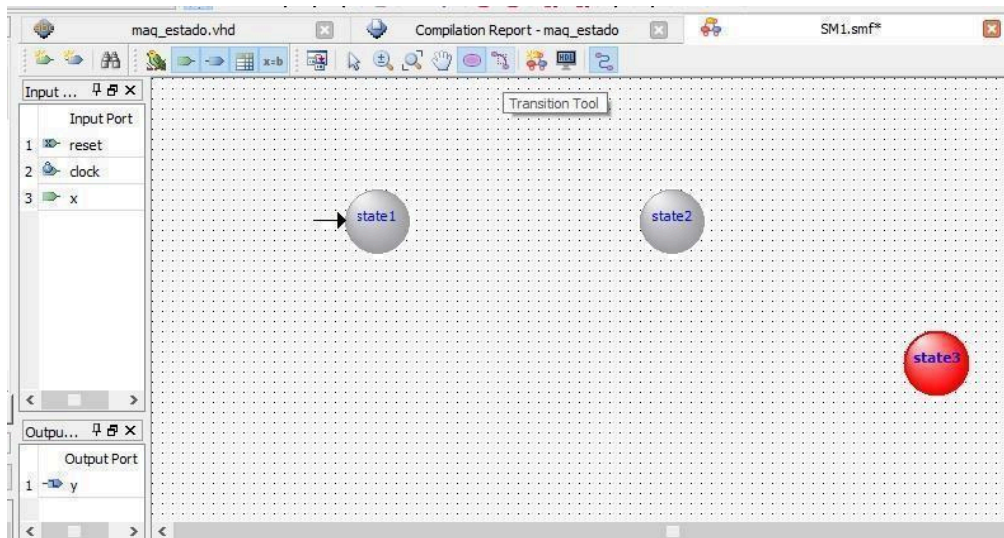
Implementación mediante State Tool:

Realice los siguientes pasos:

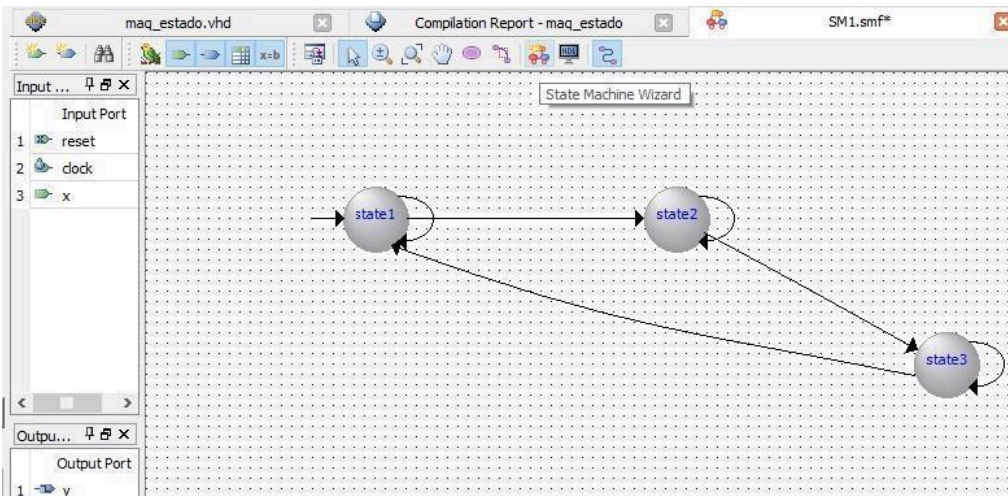
- 1) Utilice el entorno Quartus II para crear un nuevo proyecto.
- 2) En el proyecto, genere un archivo *State machine file*.
- 3) Con la herramienta *State tool* ingrese los estados.



4) Con la herramienta *Transition tool* ingrese las transiciones.



5) Con la herramienta *State machine wizard* seleccione Edit existing state machine.



6) Con la herramienta *State machine wizard* seleccione Edit:
Ingrese las entradas, salidas, las condiciones de transición:

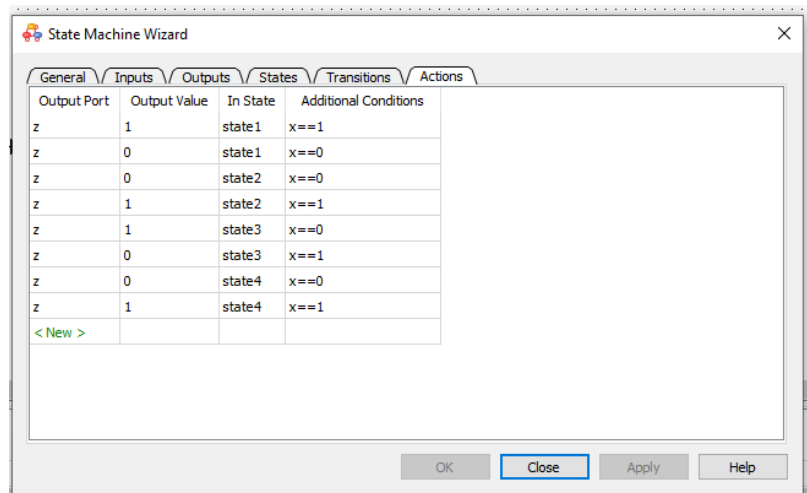
State Machine Wizard

Source State	Destination State	Transition (In Verilog or VHDL 'OTHERS')
state2	state2	x==0
state2	state1	x==1
state1	state2	x==0
state1	state3	x==1
state3	state4	x==0
state3	state2	x==1
state4	state1	x==1
state4	state4	x==0

< New >

"=="	EQUAL
"!="	INEQUAL
"<="	LESSER THAN
"<"	LESSER
">="	GREATER THAN
">"	GREATER
"&"	AND
" "	OR
"^"	XOR
"&"	NAND
" "	NOR
"^"	XNOR
"~"	NOT

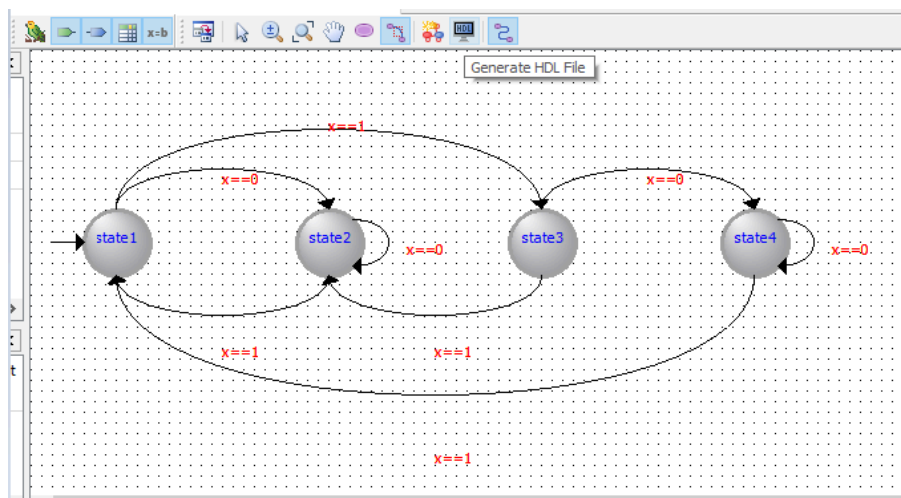
Y en la solapa *Actions* los valores de salida.



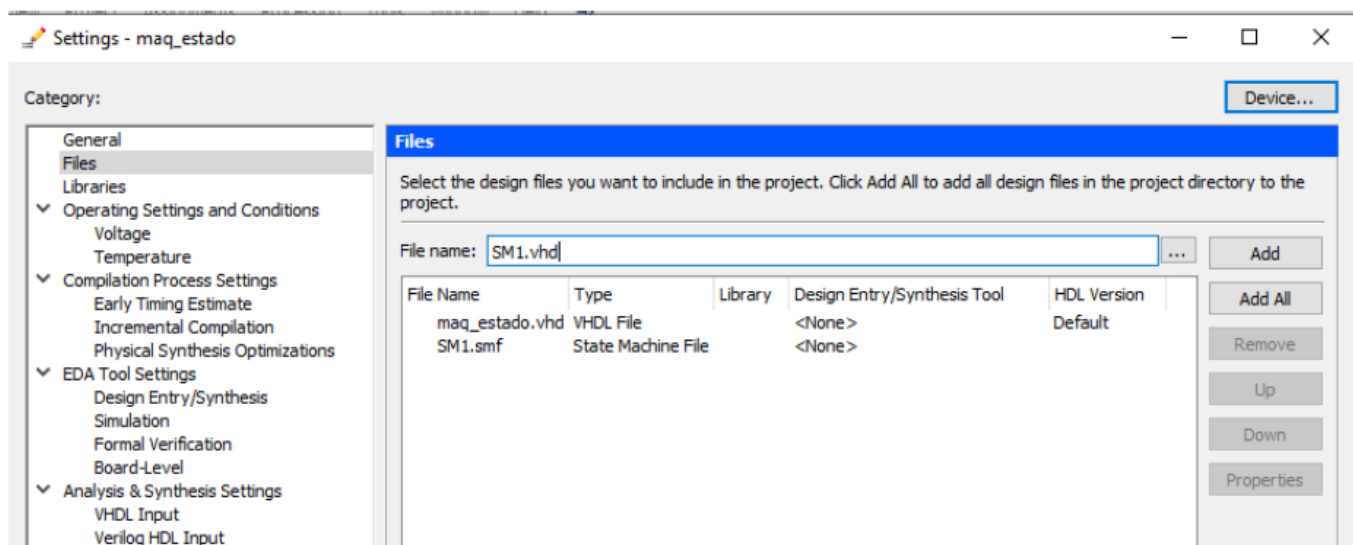
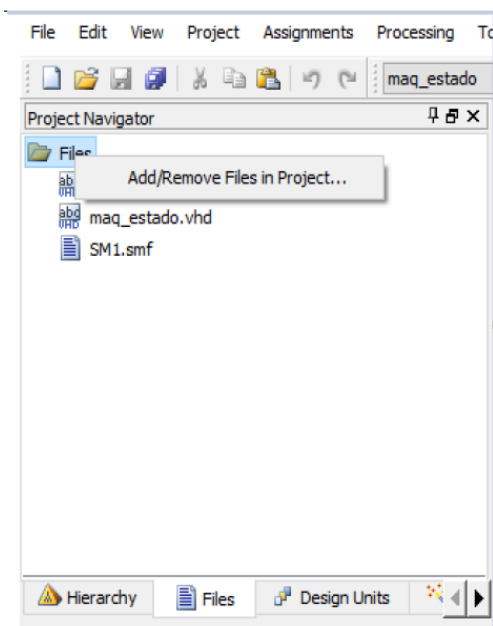
Output Port	Output Value	In State	Additional Conditions
z	1	state1	x==1
z	0	state1	x==0
z	0	state2	x==0
z	1	state2	x==1
z	1	state3	x==0
z	0	state3	x==1
z	0	state4	x==0
z	1	state4	x==1
< New >			

Note que si la máquina es Moore la columna “Additional Conditions” quedará en blanco, si es Mealy se deberá completar con la condición de la entrada.

1) Genere el código VHDL:

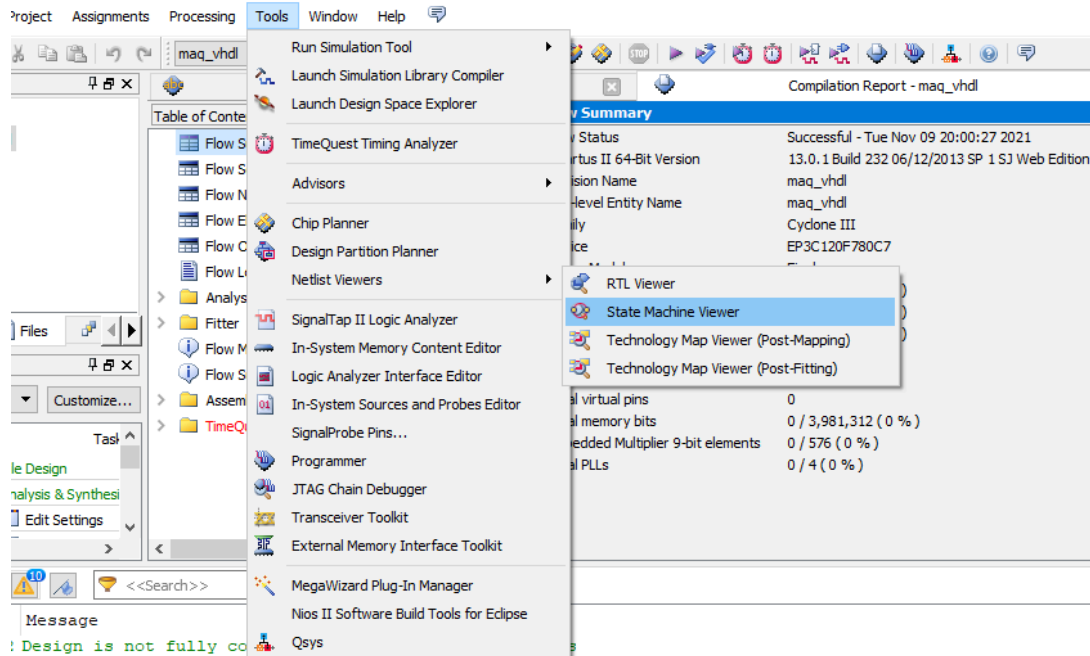


8) Guarde el archivo vhd generado y agréguelo al proyecto.

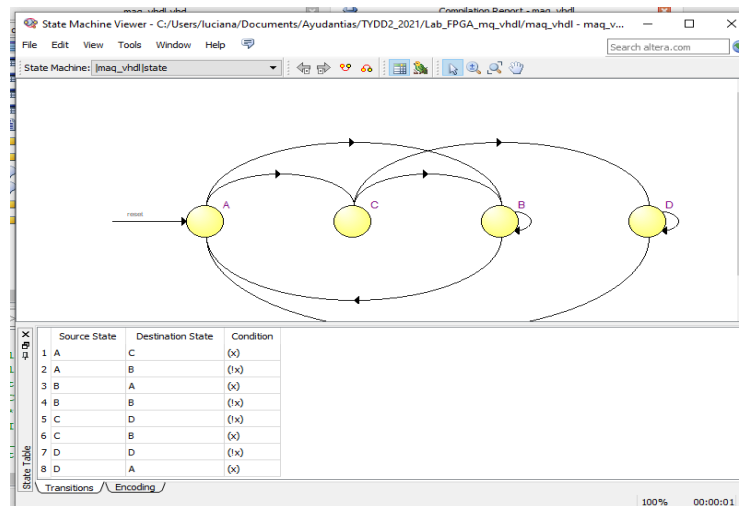


9) Compile.

10) Vea la máquina implementada seleccionando del Menu Tools=>Netlist Viewers=>State Machine Viewer

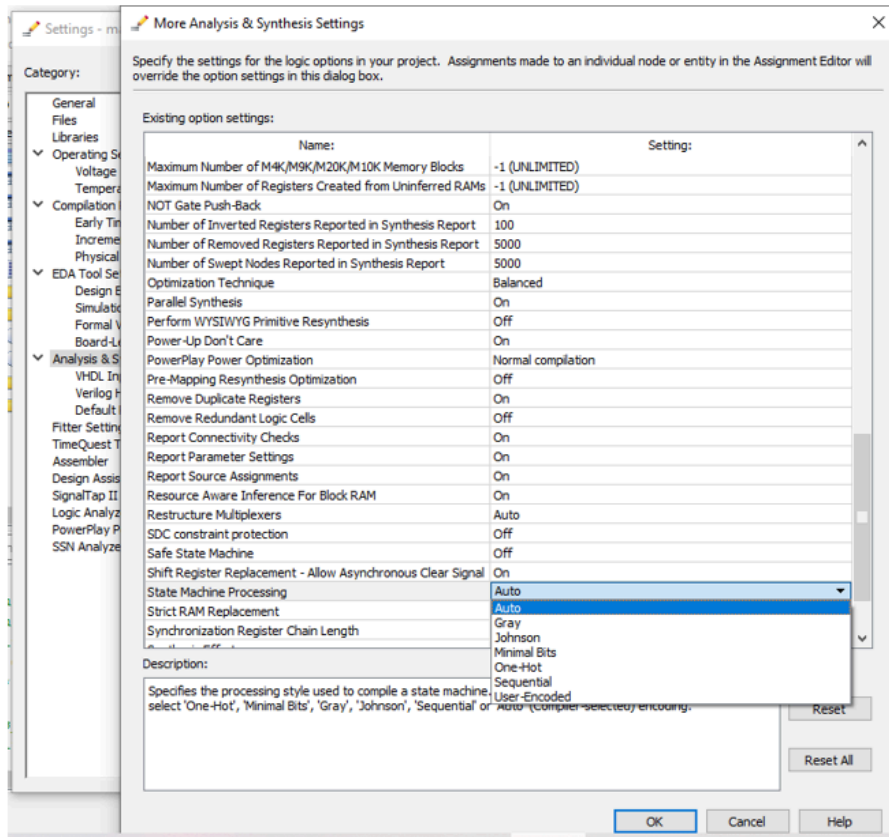


Verá algo como la siguiente figura, donde en la solapa “Encoding” podrá ver la codificación empleada.



11) Cambie el estilo de procesamiento de la máquina de estado, para que el compilador asigne otra codificación.

AYUDA: Para cambiar la codificación usada para cada estado seleccione del menú Assignments=>Settings=>Analysis & Synthesis Settings=> More Settings=> State Machine Processings podrá elegir el estilo de procesamiento entre: auto, gray, Johnson, Minimal Bits, One-Hot, Sequential y User-Encoded



Auto	Allows the Compiler to choose the best encoding for the state machine.
Gray	Uses the minimal number of bits to encode the state machine, ceiling of 2 to the log of n states. This setting is different from the Minimal Bits setting because each state has only one bit difference from its neighboring states.
Johnson	The number of bits used to encode the state machine is the ceiling of half of the states. Each state has only one bit difference from its neighboring states. Each state is generated by shifting the previous state's bits to the right by 1. The MSB of each state is the negation of the LSB of the previous state.
Minimal Bits	Uses the minimal number of bits to encode the state machine.
One-Hot	Encodes the state machine in the one-hot style. For one-hot encoding, the Quartus II software does not guarantee that each state has one bit set to one and all other bits to zero. Instead, the Quartus II software makes sure that the reset state is all-zeroes. All other states have two bits set to one and all others to zero. This is done so the state machine powers up in the reset state. This encoding has the same properties as true one-hot encoding: each state can be recognized by the value of one bit.
Sequential	Uses the minimal number of bits to encode the state machine; each state is the binary form of its state value.
User-Encoded	Encodes the state machine in the manner specified by the user.

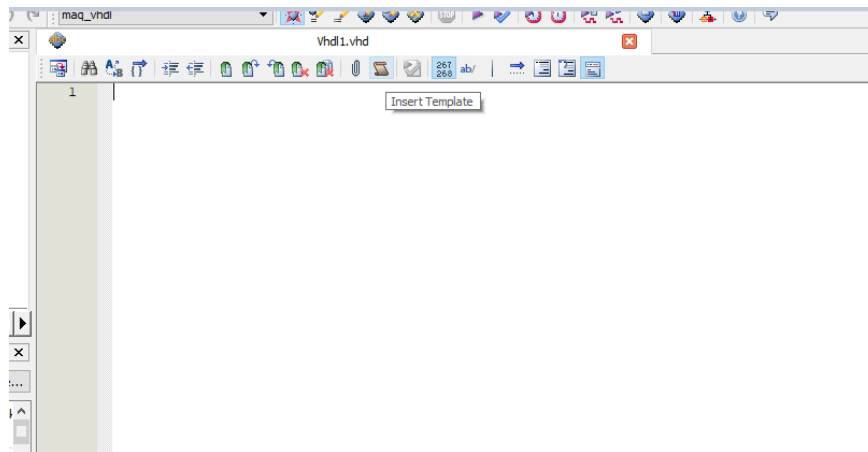
12) Compile nuevamente y vea el circuito implementado.

13) Realice las simulaciones para verificar el correcto funcionamiento.

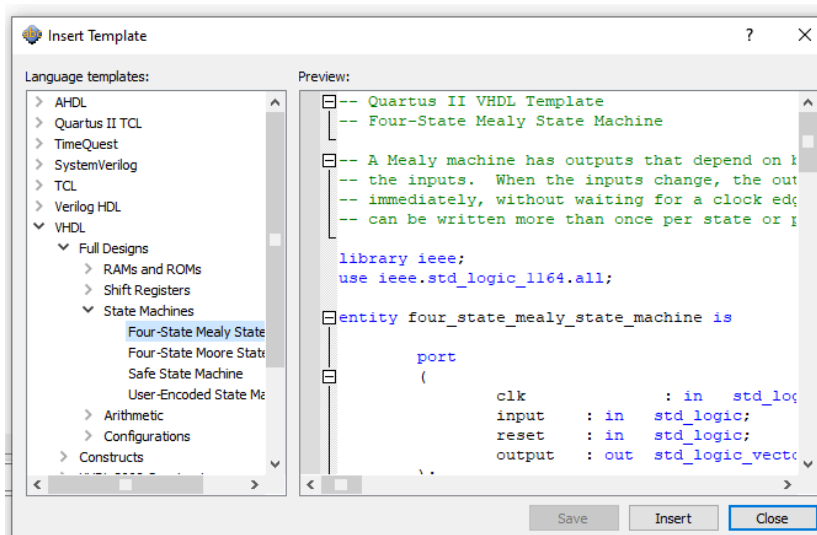
Implementación mediante Template de máquina de estado:

Realice los siguientes pasos:

- 1) Utilice el entorno Quartus II para crear un nuevo proyecto.
- 2) En el proyecto genere un archivo *VHDL file*.
- 3) Seleccione Insert Template



- 4) 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.



- 5) Compile
- 6) Vea la máquina implementada seleccionando del Menu Tools=>Netlist Viewers=>State Machine Viewer
Verifique en la solapa "Encoding" la codificación empleada.
- 7) Cambie la codificación usada para cada estado.

AYUDA:

Seleccione del menú Assignments=>Settings=> Analysis & Synthesis Settings=> More Settings=> State Machine Processings podrá elegir el estilo de procesamiento entre: auto, gray, Johnson, Minimal Bits, One-Hot, Sequential y User-Encoded

8) Compile nuevamente y vea el circuito implementado.

9) Simule para verificar el correcto funcionamiento.

Parte F: Implementación de I2C

Implemente en la FPGA el circuito de comunicación serie I²C (Inter-Integrated Circuit) explicado en la teoría.

Se deberá implementar el circuito en un esclavo que sea capaz de recibir la dirección destino, si es la propia deberá recibir 1 byte de datos del maestro y responder con un ACK y volver al estado de espera.

Verifique su funcionamiento mediante simulación.

