PARTE C: Implementación de un restador de 4 bits mediante un restador completo y los bits B y Z, empleando el entorno esquemático.

A partir del restador completo creado en VHDL en el punto anterior cree en Quartus II un proyecto esquemático en el cual se implemente un restador de 4 bits y los bits de CCR (Z y B). Latchee las señales de entrada y salida mediante Flip-Flops D.

Realice los siguientes pasos:

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

2) En el proyecto genere un archivo esquemático (defina como Top Level a este archivo).

3) Agregue en el proyecto un archivo VHDL que implemente el restador completo, como el desarrollado en el punto anterior pero SIN registrar las entradas y salidas, ya que en este caso será un componente interno de un circuito mayor, por lo que se deben registrar las entradas y salidas del circuito final.

AYUDA (Ejemplo para el caso sumador completo):

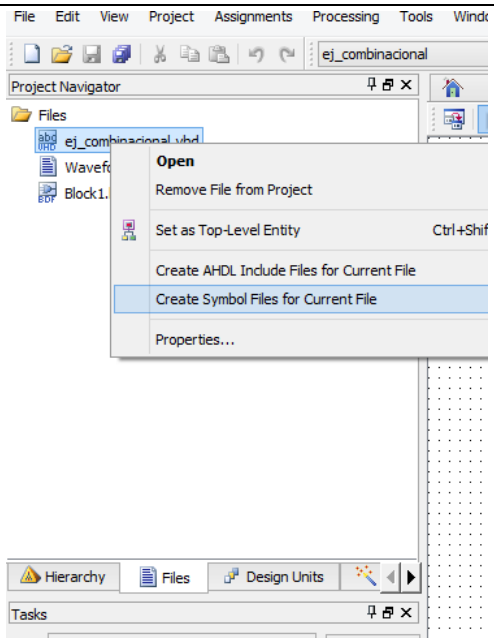
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sumador_completo is
    Port (
        a    : in STD_LOGIC;
        b    : in STD_LOGIC;
        cin  : in STD_LOGIC;
        f    : out STD_LOGIC;
        cout : out STD_LOGIC);
end sumador_completo;

architecture Behavioral of sumador_completo is

begin

    f <= a xor b xor cin;
    cout <= (a and b) or (cin and (a xor b));

end Behavioral;
```

4) Genere un símbolo esquemático a partir del archivo vhd desarrollado en el 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.



5) Agregue el componente simbólico al archivo esquemático.

6) En el archivo esquemático agregue los símbolos de los componentes necesarios (Symbol Tool), necesarias para conseguir el funcionamiento de un restador y los bits de CCR. Agregue FFD en las entradas y salidas del circuito.

AYUDA:

Puede usar el archivo VHDL DFF desarrollado en el punto anterior, incluirlo en el proyecto, generar un símbolo esquemático a partir de este arhicvo VHDL y agregarlo al circuito esquemático.

Otra opción es usar el ffd que provee el Quartus en Simbol tool=> primitives=> storage=> dff

7) Realice las conexiones genere puertos de entrada/salida y compile.

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

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

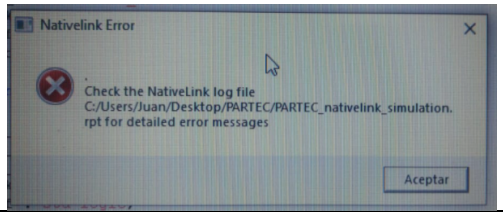
10) Verifique el circuito implementado luego del mapeo en el dispositivo.

11) Verifique la implementación en el chip.

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

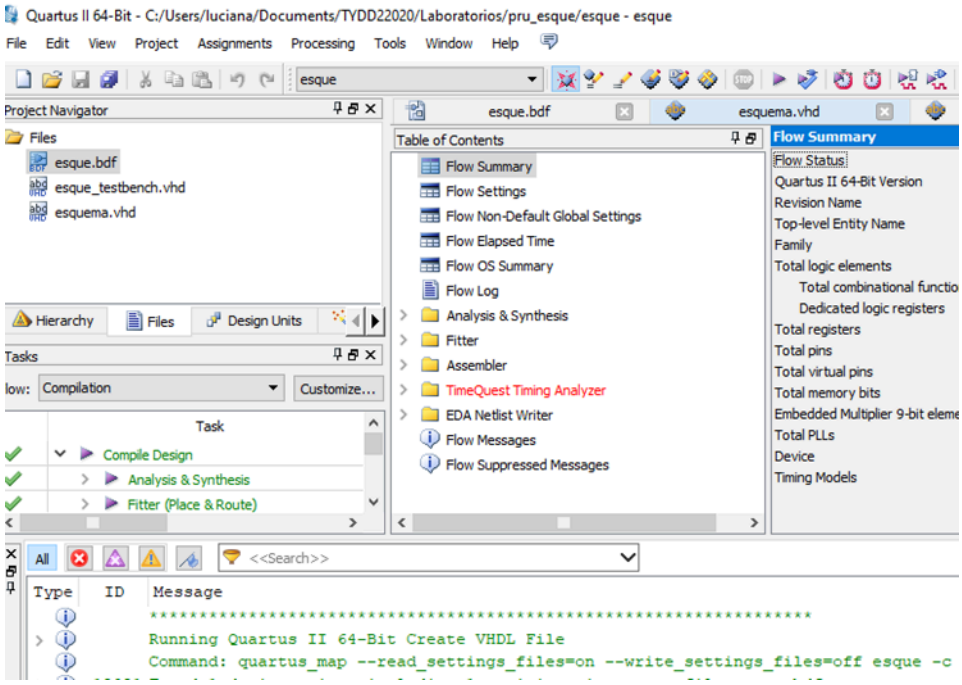
13) Verifique el funcionamiento del circuito mediante la simulación de Modelsim.

En este caso no se logra realizar la simulación con el entorno de simulación de Modelsim. El archivo testbench no logra instanciar el archivo esquemático creado. Aparece el siguiente error:

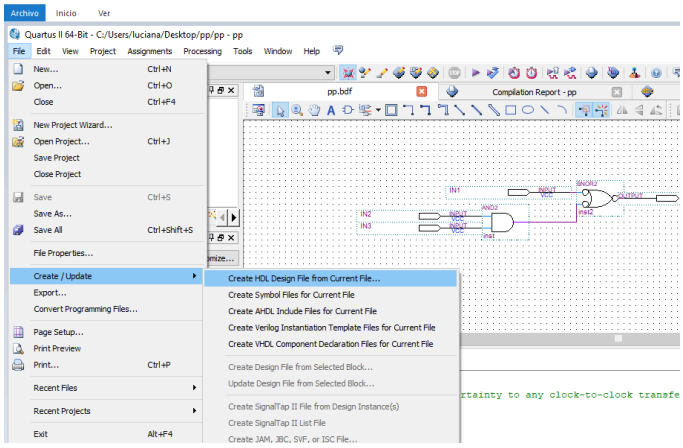


Para poder simular un archivo esquemático primero hay que pasar el esquemático a VHDL, porque Modelsim no soporta circuitos esquemáticos.

Para solucionarlo: Convertir el esquemático en VHDL➔ pararse en el archivo esquemático.



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



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

PARTE D: 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:

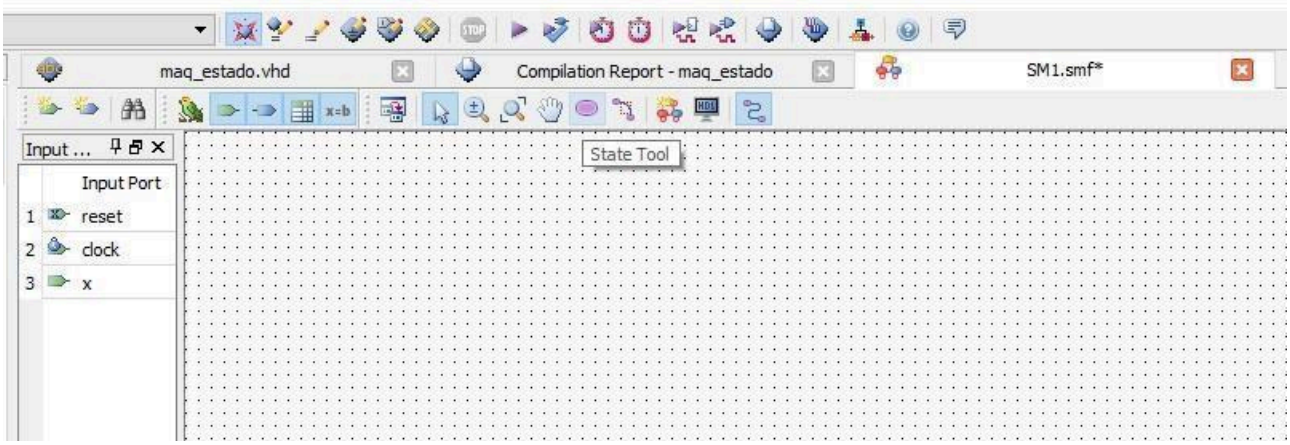
Implementación mediante State Tool:

Realice los siguientes pasos:

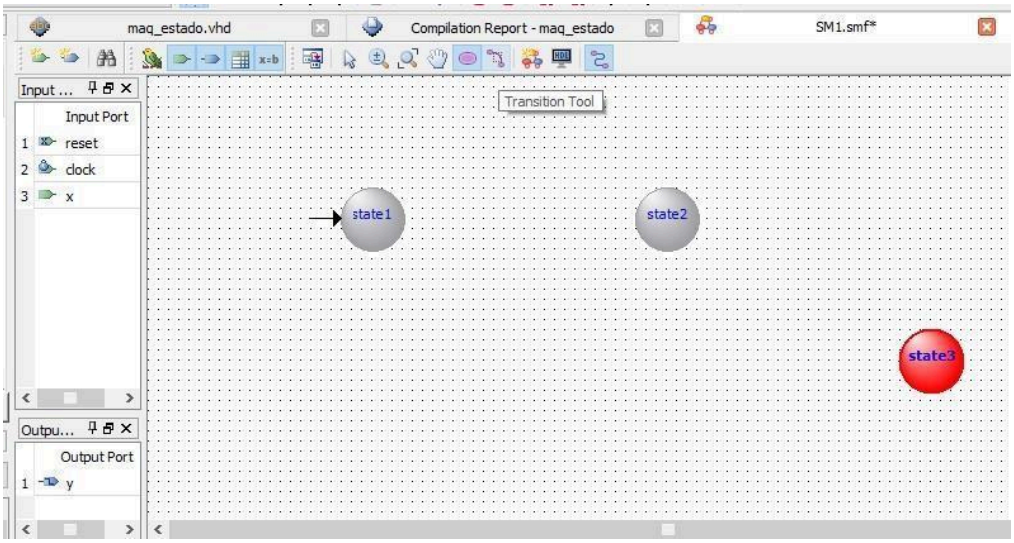
1) Utilice el entorno Quartus II para crear un nuevo proyecto.

2) En el proyecto genere un archivo *State machine file*.

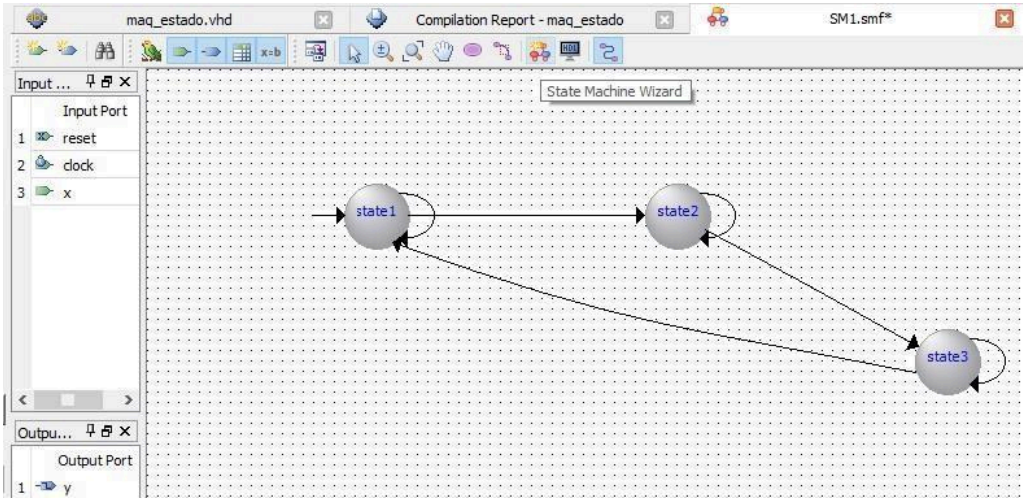
3) Con la herramienta *State tool* ingrese los estados.



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

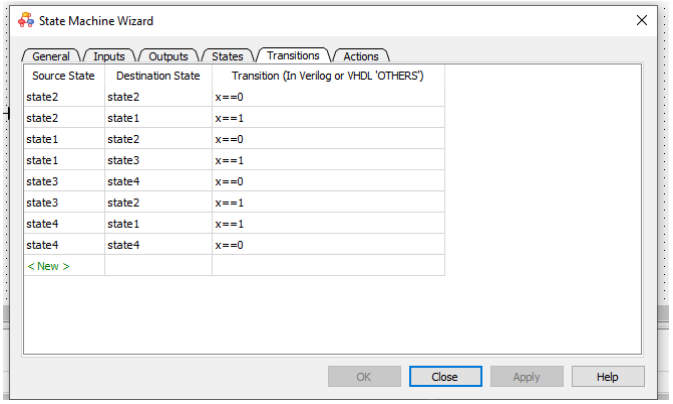


5) Con la herramienta *State machine wizard* seleccione Edit existing state machine.

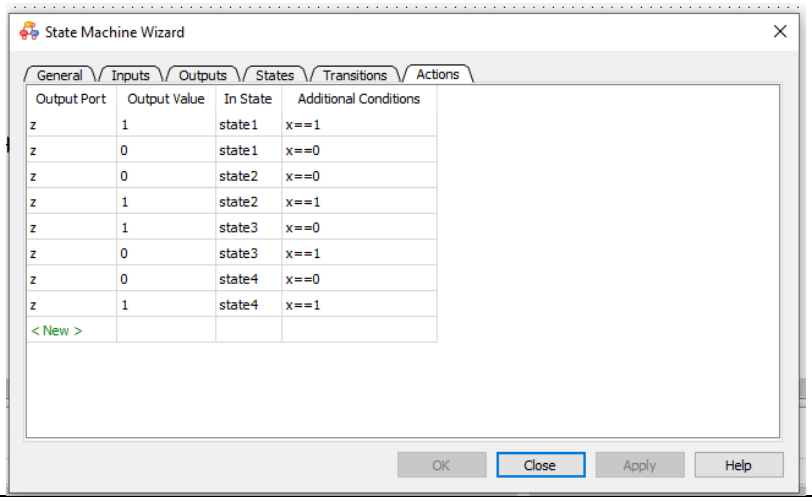


6) Con la herramienta State machine wizard seleccione Edit:

Ingrese las entradas, salidas, las condiciones de transición:

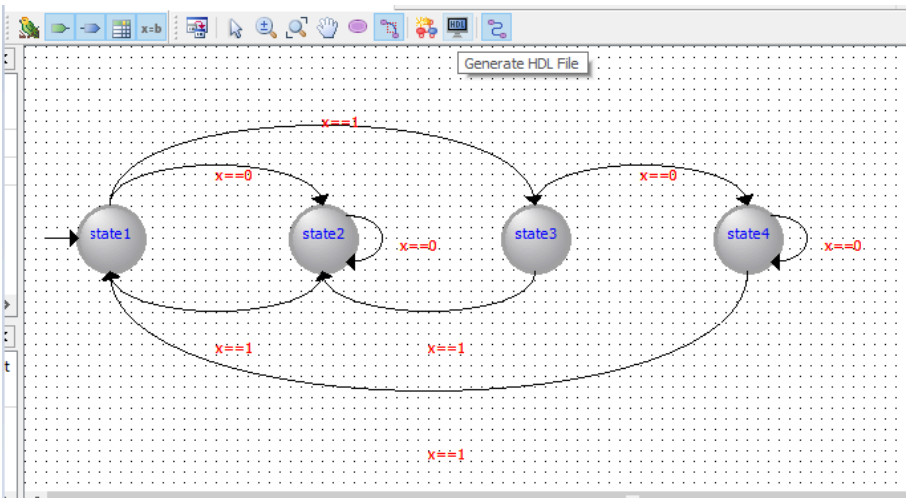


Y en la solapa *Actions* los valores de salida.

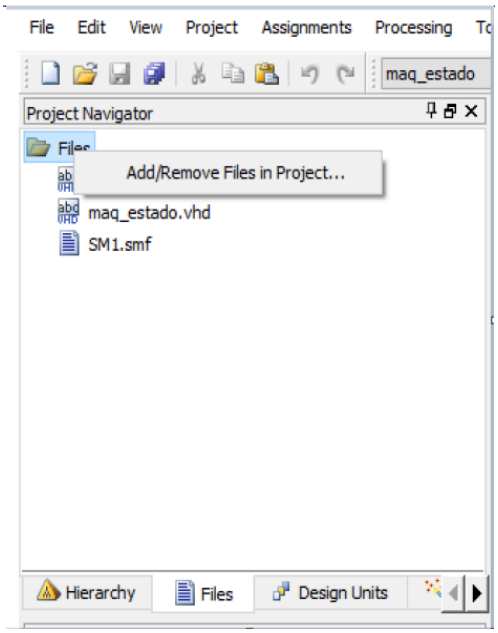


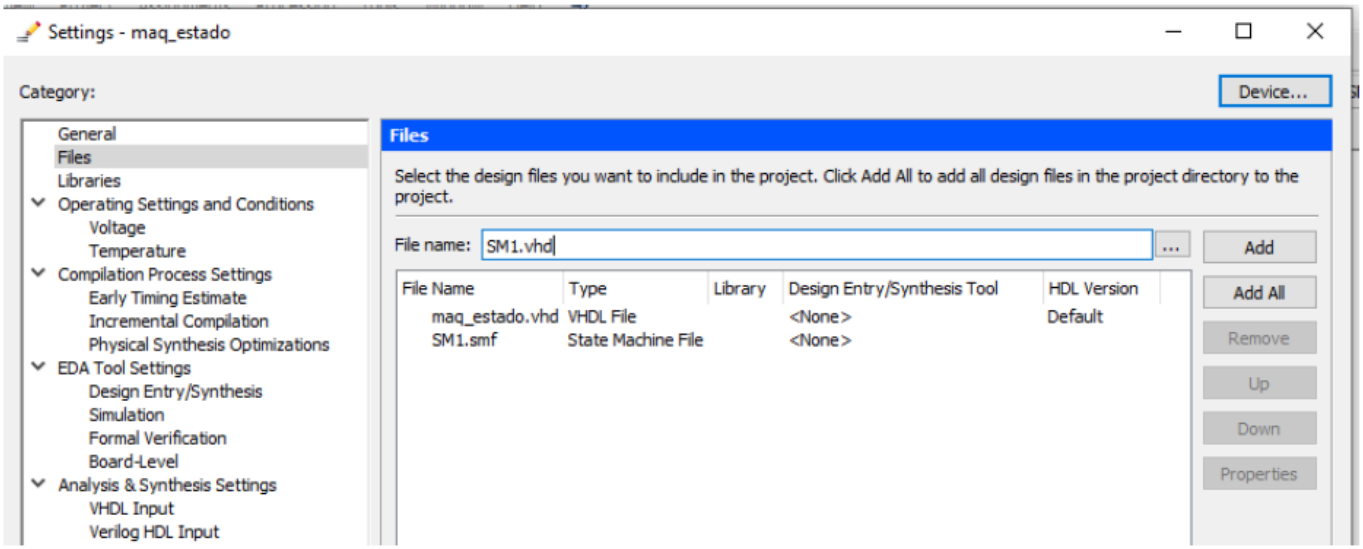
Note que si la maquina es Moore la columna “Additional Conditions” quedara en blanco, si es Mealy se deberá completar con la condición de la entrada.

1) Genere el código VHDL:



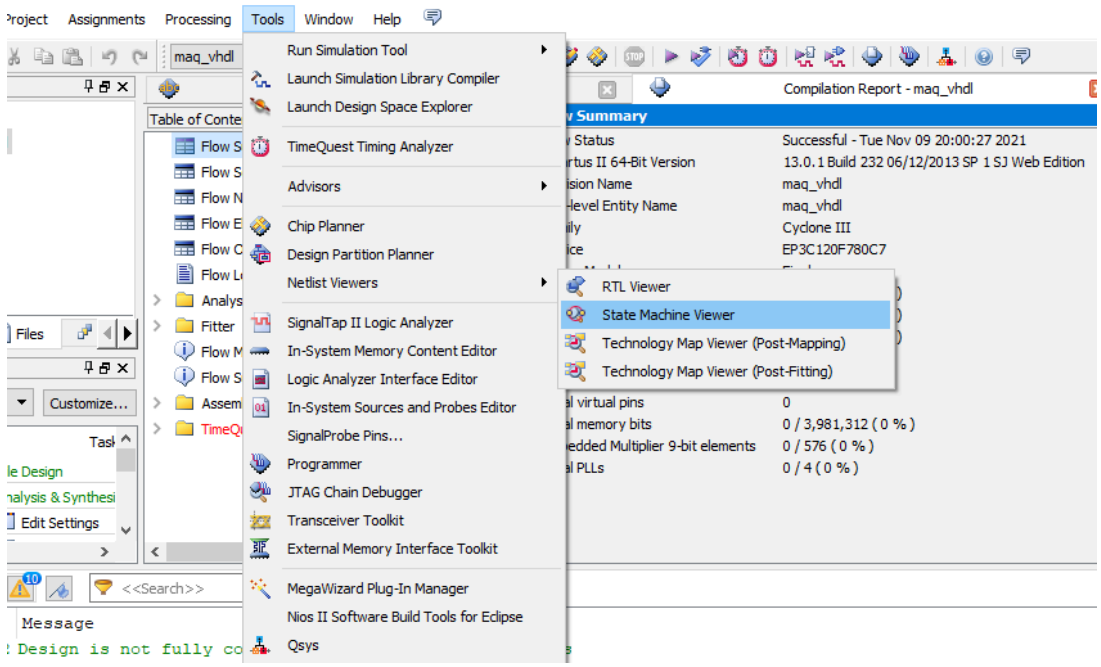
8) Guarde el archivo vhdl generado y agréguelo al proyecto.



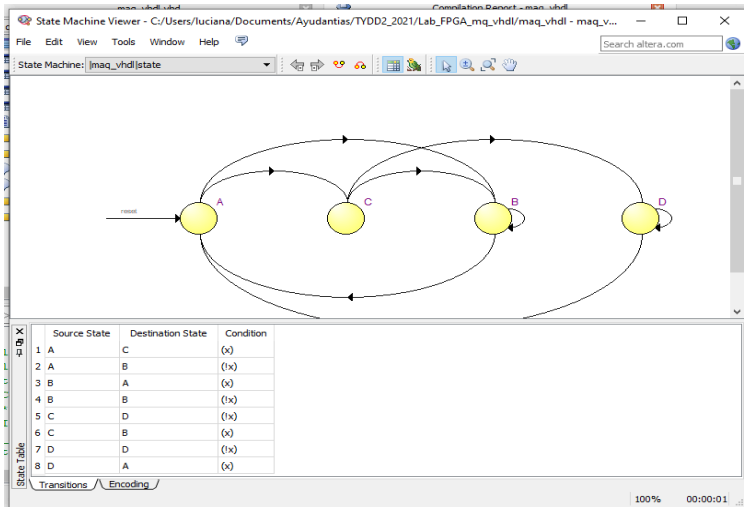


9) Compile.

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

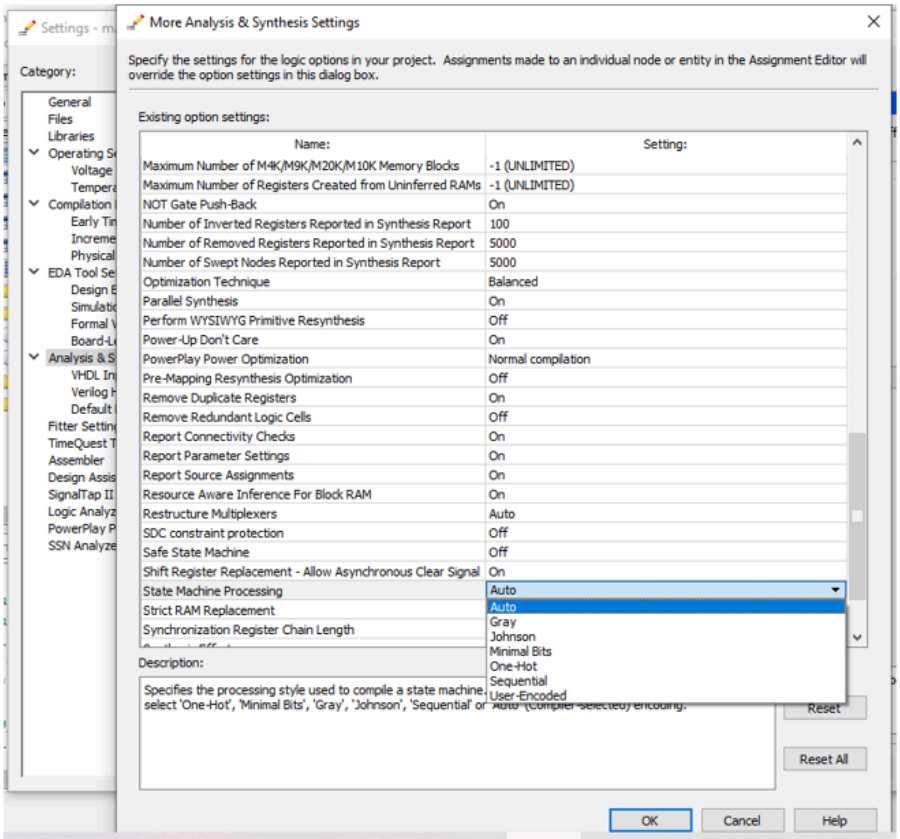


Verá algo como la siguiente figura, donde en la solapa “Encoding” podrá ver la codificación empleada.



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

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 difference 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 ceilling 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.

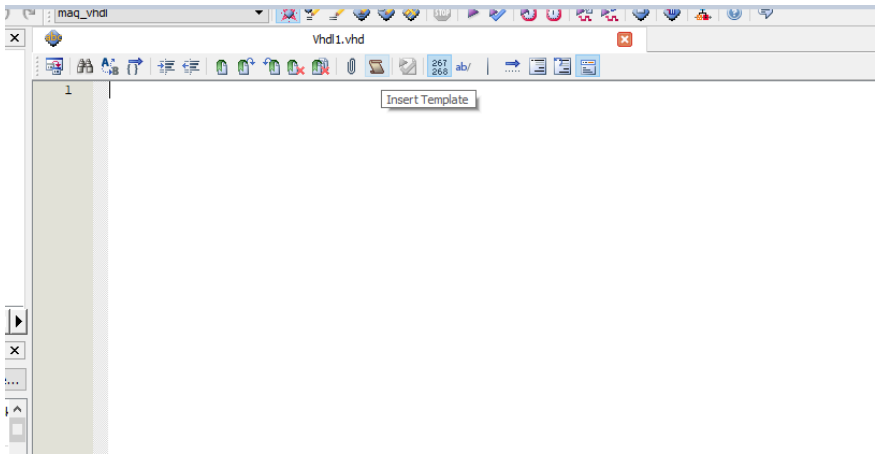
Implementación mediante Template de máquina de estado:

Realice los siguientes pasos:

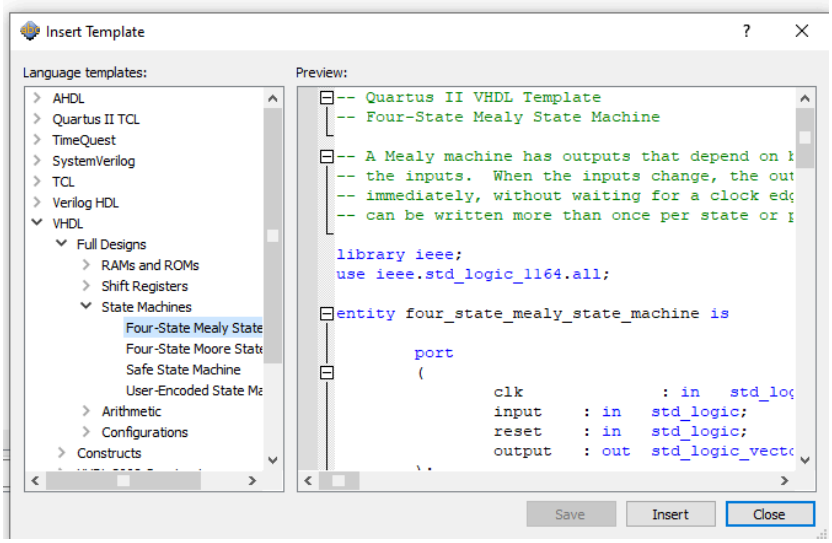
1) Utilice el entorno Quartus II para crear un nuevo proyecto.

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 E: Implementación de I2C

Implemente en la FPGA el circuito de Comunicación serie I^2C (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 a el estado de espera.

Verifique su funcionamiento mediante simulación.