Sistemas digitales I

Práctica # 6

Capítulo del curso: simulación de un circuito secuencial

Objetivos de aprendizaje:

- Simular en QUARTUS un sistema digital secuencial diseñado a partir de programación en código VHDL y programación gráfica con diagramas de bloques.
- Aplicar conceptos de simulación en Quartus para convertir un código VHDL en su correspondiente símbolo.

Duración: 60 minutos

Materiales y herramientas:

- Circuitos msi estudiados en las sesiones de clase.
- Quartus prime

Descripción de la práctica:

Procedimiento:

En esta práctica se diseñará un sistema digital, combinatorial y secuencial, que al activar la botonera **INCREMENTAR,** aumente en uno un contador. Existe también una botonera **REINICIAR**, la cual permite encerar el sistema en cualquier momento.

1. Dado a continuación el código VHDL de los componentes a usarse, realice la compilación de cada uno.

- ## Sumador de 4 bits (sum)

```
library IEEE;
   use IEEE.STD_LOGIC_1164.all;
         use IEEE.STD_LOGIC_signed.all;
   use IEEE.NUMERIC_STD.all;

   ENTITY sum IS
    PORT (a : IN std_logic_vector(3 DOWNTO 0);
        b : IN std_logic_vector(3 DOWNTO 0);
        salida : OUT std_logic_vector(3 DOWNTO 0);
                                    c: OUT std_logic);
   END sum;

   ARCHITECTURE synth OF sum IS
         signal suma: std_logic_vector(4 DOWNTO 0);
```

```
BEGIN

 PROCESS (a, b) IS
 BEGIN
  suma <= (('0'&a)+('0'&b));
                        salida<= suma(3 DOWNTO 0);
                        c<= suma(4);
 END PROCESS;
END synth;
```

- **Registro universal**

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

entity registro_universal is
        Port ( D: in STD_LOGIC_VECTOR(3 downto 0);
                S : in STD_LOGIC_VECTOR(1 downto 0);
                CLK, R, L, RESET: in STD_LOGIC;
                Q: out STD_LOGIC_VECTOR(3 downto 0));
end registro_universal;

Architecture sol of registro_universal is
SIGNAL Q1: STD_LOGIC_VECTOR(3 downto 0);
Begin
        Q <= Q1;

        process (CLK, RESET)
        begin
                if reset ='0' then Q1 <= "0000";
                elsif (clk'event and clk='1') then
                case S is
                        when "00"=>  Q1<=Q1;
                        when "01"=>  Q1<=R & Q1 (3 downto 1);
                        when "10"=>  Q1<=Q1 (2 downto 0) & L;
                        when "11"=>  Q1<=D (3 downto 0);
                        when others => Q1<=Q1;
                end case;
                end if;
        end process;
end sol;
```

2. Una vez compilado y comprobado el funcionamiento de cada código, crear el símbolo del mismo, para ello dar clic derecho sobre el archivo del cual se desea generar el bloque y seleccionar la opción Create Symbol Files for Current File.

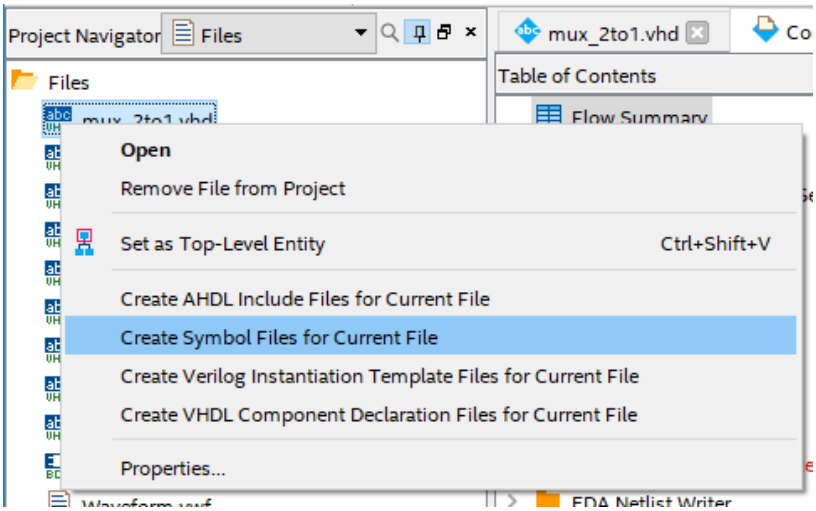


Imagen de referencia

3. Ahora puede usar este símbolo en cualquier archivo BDF del proyecto, para este fin puede buscarlo con el nombre de la entidad o en el directorio del proyecto.

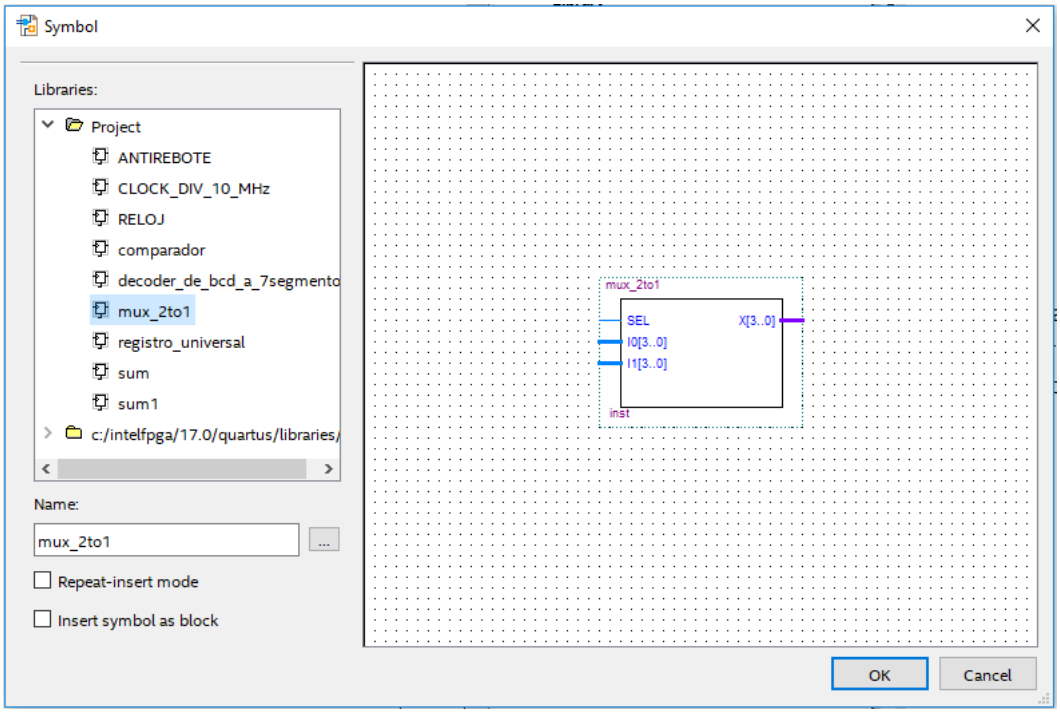
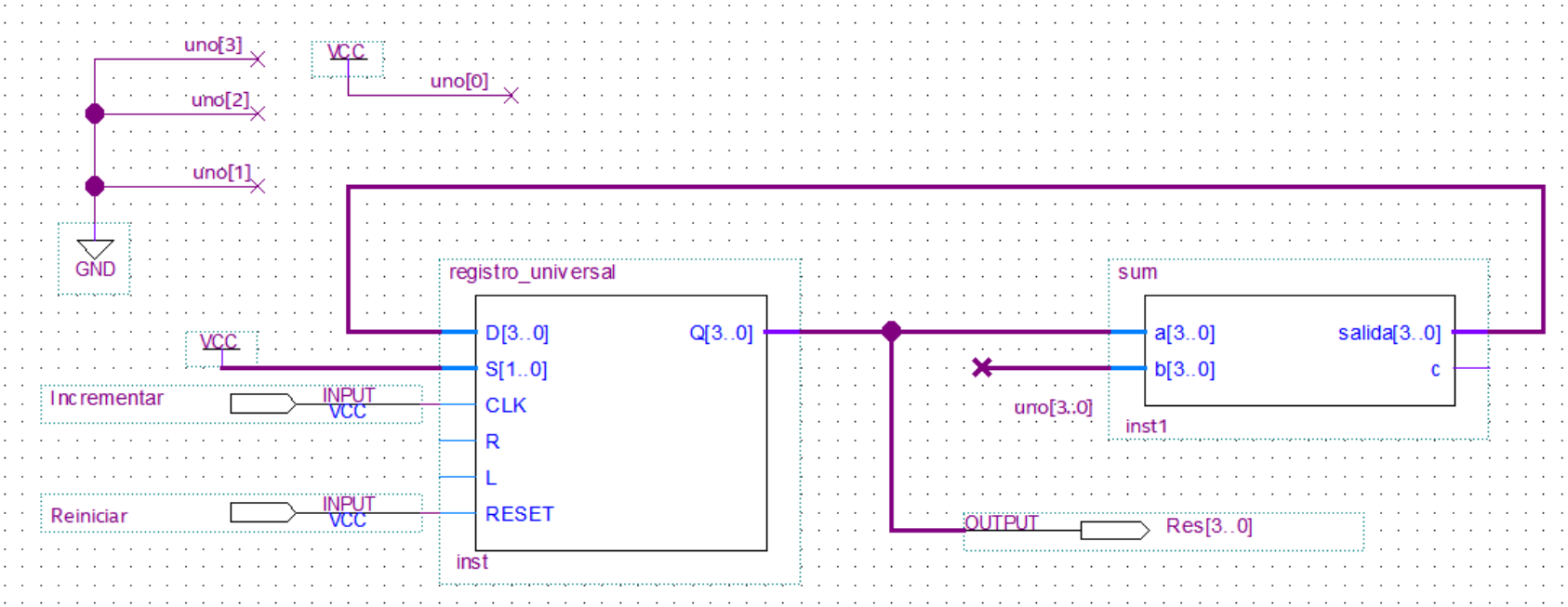


Imagen de referencia

4. Una vez realizado el bloque de cada dispositivo, de modo que estén guardados en la biblioteca creada por uno mismo, implementar el circuito mostrado a continuación, compilarlo y simularlo como se realizó en prácticas anteriores.



Bibliografía:

[1]. Fundamentos de Lógica Digital, Stephen Brown & Zvonco Vranesic, Segunda Edición, Mc.Graw Hill, 2009.

[2]. Sistemas Digitales: Principios y Aplicaciones, Ronald Tocci, Octava Edición, Prentice Hall, 2003.

[3]. Digital Design with RTL Design, Verilog and VHDL, Frank Vahid, Second Edition, John Wiley and Sons, 2010.

Por: Alexis Lema Ordóñez

23/08/2020