Universidad Nacional Autónoma de México

Facultad de Ingeniería

Diseño Digital VLSI

Semestre 2020-1

Práctica 0

Grupo 05

Chávez Troncoso Héctor Manuel
Garciarebollo Rojas Diego Iñaki
Moreno Osuna Isaac
Rubio Carmona Alonso Rafael

Cd. Universitaria a 27 de agosto de 2019

Planteamiento del problema

Retoman los conceptos vistos en la con respecto a la programación en VHDL vistos en anteriormente en la materia de Diseño Digital Moderno, se dispuso a diseñar en el ambiente de desarrollo Quartus varios circuitos. El primero consiste en un sumador de 1 solo bit; el siguiente consta de una compuerta AND la cual recibe dos vectores, uno de 2 bits y otro de 3 bits, y los compara; un contador hasta el número 264,144 utilizando el reloj interno de la tarjeta FPGA, y por último un multiplexor el cual tiene como opciones corrimiento a la derecha y a la izquierda de bits, agregando un 1 o un 0.

Objetivo

Diseñar cuatro pequeños circuitos básicos los cuales consisten en; un sumador, una compuerta AND que recibe dos vectores, un contador hasta el número 264,144 y un multiplexor con funciones internas de corrimiento de bits, para reforzar los conocimientos básicos en lenguajes VHDL.

Metodología

Para el diseño del sumador de un bit se ocuparon tanto dos variables de entrada como dos variables de salida. El resultado de la operación XOR sobre las variables de entrada (A y B) se asigna a la variable de acarreo de bits (C). Además se asigna a la variable de salida (O) el resultado de la operación AND de las variables de entrada.

Con respecto a la compuerta AND, es un poco más sencillo con respecto a operaciones, a la variable de salida se le asigna por defecto el valor de “00” concatenado con el resultado de la operación AND de los arreglos de entrada (A y B), pero solamente tomando los dos bits menos significativos.

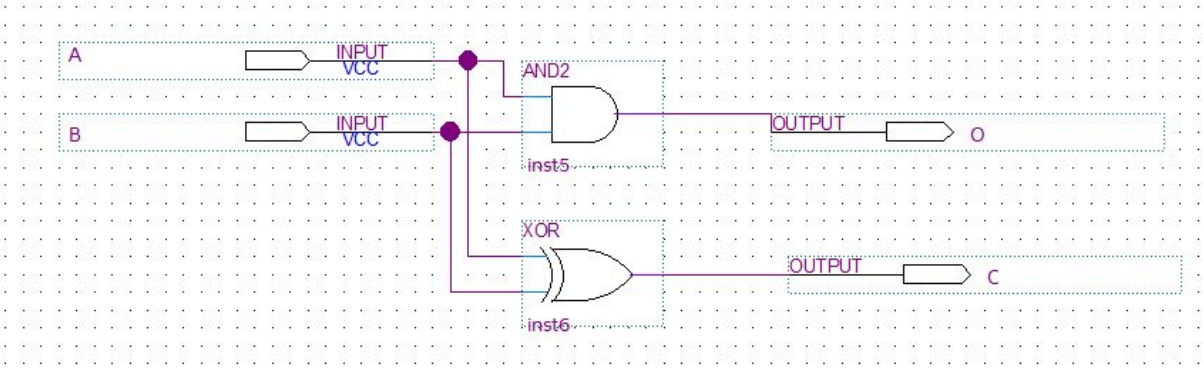
Ahora para el desarrollo del contador, se tiene como variable de entrada un reloj (dicho reloj es el que se toma de la tarjeta). A una señal (count) se le asigna el valor de 0 y dentro de un process se va a ir sumando uno a una variable cada que clk'EVENT ocurra y la variable clk sea igual a 1. Esto se hará mientras que la señal (count) no sea mayor a 264,145. Al final se pasa el valor de la señal se pasa a la variable de salida LED, para mostrar el resultado.

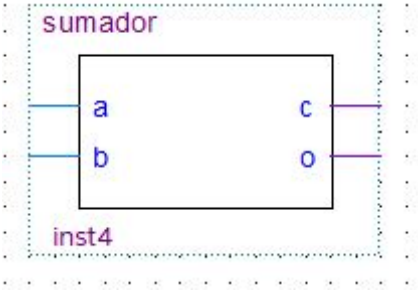
Para el caso del multiplexor se tiene como entradas al selector y a X y se tienen como salida a Y, siendo todos vectores lógicos de 2 bits para el selector y de 3 bits

para X y Y. Se utiliza un SELECT WITH en el cual a Y se le asignan los valores de las operaciones definidas según la entrada del selector haciendo el corrimiento a la izquierda y a la derecha así como agregando al principio y al final un uno y un cero según la entrada o devolviendo un 0 en caso de que se ingrese un valor diferente.

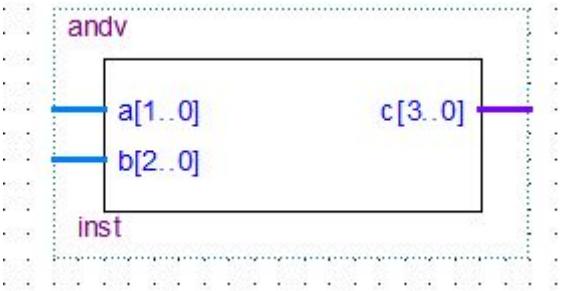
Arquitectura del Sistema

Sumador

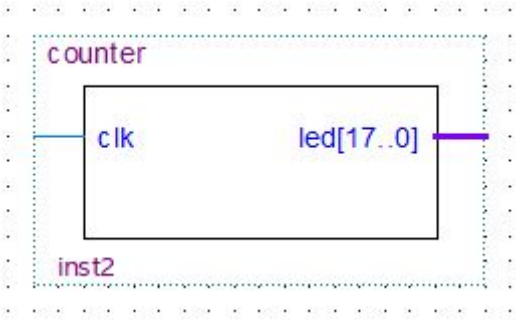




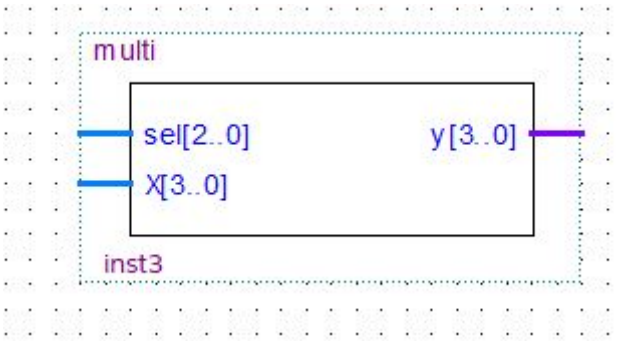
AND



Contador



Multiplexor



Código Fuente

Sumador

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY sumador IS
PORT(
    a,b : IN STD_LOGIC;
    c,o : OUT STD_LOGIC);
END sumador;

ARCHITECTURE prueba OF sumador IS
BEGIN
    c <= a XOR b;
    o <= a AND b;
END prueba;
```

AND

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY andv IS
PORT(
     a : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
     b : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
     c : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END andv;
ARCHITECTURE prueba OF andv IS
BEGIN
    c <= "00" & (a (1 DOWNTO 0) AND b (1 DOWNTO 0));
END prueba;
```

Contador

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY counter IS
     Port ( clk : in  STD_LOGIC;   -- input clock
               -- LEDs to display count
            led : out  STD_LOGIC_VECTOR (17 DOWNTO 0)
     );
END counter;

ARCHITECTURE Behavioral OF counter IS
     SIGNAL count   : STD_LOGIC_VECTOR (17 DOWNTO 0) :=
X"0000"&"00";
BEGIN
     -- counter
     PROCESS (clk)
     BEGIN
     IF (clk'EVENT AND clk = '1' AND (NOT
COUNT="11110100011011110 0")) THEN
                count <= count + '1';   -- counting up
     END IF;
     END PROCESS;

     -- display count on LEDs
     LED <= NOT count;
```

```
END Behavioral;
```

Multiplexor

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY multi IS
PORT(
    sel : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
    X   : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
    y   : OUT STD_LOGIC_VECTOR (3 DOWNTO 0)
);
END multi;
ARCHITECTURE prueba OF multi IS
BEGIN
    WITH sel SELECT
      y<=
           x(0) & x(3 DOWNTO 1) WHEN "000",
           x(2 DOWNTO 0) & x(3) WHEN "001",
           '0' & x(3 DOWNTO 1)  WHEN "010",
           x(2 DOWNTO 0) & '0' WHEN "011",

           '1' & x(3 DOWNTO 1)  WHEN "100",
           x(2 DOWNTO 0) & '1' WHEN "101",
           X"0" WHEN OTHERS;
END prueba;
```

Resultados

```
Running Quartus Prime Analysis & Synthesis
Command: quartus_map --read_settings_files=on --write_settings_files=off p1 -c p1
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 to an appropriate value for best performance.
20030 Parallel compilation is enabled and will use 4 of the 4 processors detected
12021 Found 2 design units, including 1 entities, in source file counter.vhd
12021 Found 1 design units, including 1 entities, in source file de10_lite_golden_top.v
12021 Found 2 design units, including 1 entities, in source file sumador.vhd
12021 Found 2 design units, including 1 entities, in source file andv.vhd
12021 Found 2 design units, including 1 entities, in source file multi.vhd
12127 Elaborating entity "counter" for the top level hierarchy
10492 VHDL Process Statement warning at counter.vhd(18): signal "count" is read inside the Process Statement but isn't in the Process Statement's sensitivity list
13024 Output pins are stuck at VCC or GND
16010 Generating hard_block partition "hard_block:auto_generated_inst"
21074 Design contains 1 input pin(s) that do not drive logic
21057 Implemented 19 device resources after synthesis - the final resource count might be different
Quartus Prime Analysis & Synthesis was successful. 0 errors, 23 warnings
```

Todos nuestros programas se lograron sintetizar sin problemas.

Conclusiones

El desarrollo de esta práctica nos ayudó a reforzar los conocimientos aprendidos a lo largo de Diseño Digital Moderno, programando unos sencillos módulos en VHDL

con una actividad muy específica. Se recordó el tema de las entidades y el cómo se declaran así como el uso de las variables o señales que puede llegar a manejar este lenguaje.
Los problemas realizados en la práctica no tuvieron mucha complicación, excepto por el contador que se tuvo que realizar la investigación de cómo poder representar el número entero de 250,300, en un conjunto de bits y se optó por la opción más simple que fue representarlo en la cadena de bits correspondiente.