Universidad Autónoma de Baja California

Ingeniería en Computación

Facultad de Ciencias Químicas e Ingeniería

Diseño Digital

Práctica 9: Implementación de circuitos combinacionales en VHDL

Zavala Román Irvin Eduardo

Grupo: 551

31/10/2021

Periodo 2021-2

Instrucciones

Implementar circuitos secuenciales, utilizando lenguaje de descripción de hardware, para su modelado, síntesis y simulación, de manera que puedan ser aplicados y obtener la visualización de la solución, con actitud ordenada y proactiva.
El docente plantea ejercicios con solicitudes de diseño de circuitos secuenciales, en donde se identifican componentes combinacionales.
Construye circuitos combinacionales que formen parte de un sistema secuencial utilizando lenguaje de descripción de hardware:

- AND de 3 entradas
- AND de 2 entradas
- Medio sumador
- Sumador completo
- Multiplexor 8 a 1

Procedimiento

La implementación de las compuertas AND de 3 entradas en VHDL es bastante sencilla, siendo solo el siguiente código:

```
-- AND (3 input) gate design
--ZAVALA ROMAN IRVIN EDUARDO
library IEEE;
use IEEE.std_logic_1164.all;
entity and_3 is
port(
    a: in std_logic;
    b: in std_logic;
    c: in std_logic;
    q: out std_logic);
end and_3;
architecture arch_and_3 of and_3 is
begin
    process(a, b, c) is
    begin
      q <= a and b and c;
    end process;
end arch_and_3;
```

Para hacer la de 2 entradas solo quito la c y simulo con a y b.

Para hacer un medio sumador debemos recordar su funcionamiento, el resultado de la suma de 2 entradas se obtiene haciendo a ⊕ b y el carry haciendo ab. Representado en un circuito combinacional queda de la siguiente manera:

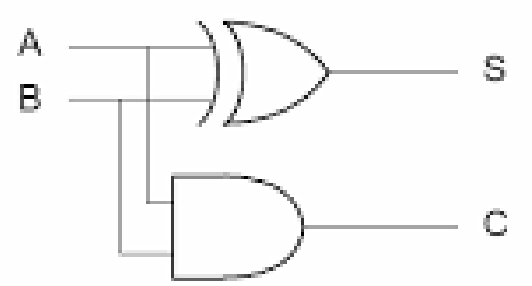

Figura 1. Circuito medio sumador

```
--MEDIO SUMADOR
--ZAVALA ROMAN IRVIN EDUARDO
library IEEE;
use IEEE.std_logic_1164.all;
entity halfAdder is
port(
   a: in std_logic;
   b: in std_logic;
   s: out std_logic;
   cout: out std_logic);
end halfAdder;
architecture arch_halfAdder of halfAdder is
begin
   process(a, b) is
   begin
     s <= a xor b;
     cout <= a and b;
     end process;
end arch_halfAdder;
```

El sumador completo a diferencia del medio sumador usa una entrada extra, el carry in, por lo que el circuito cambia. Ahora la salida se obtiene con a ⊕ b ⊕ cin, el carry se obtiene con (a ⊕ b) cin + ab.

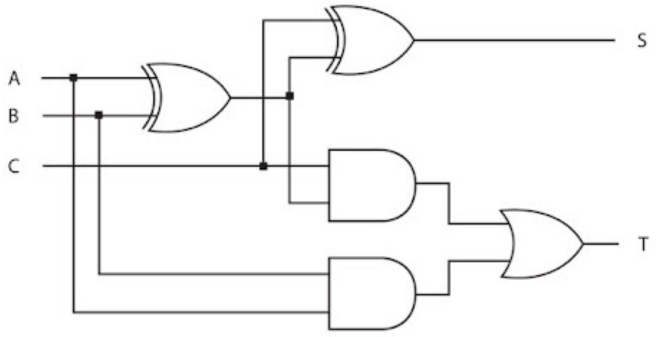

Figura 2. Circuito sumador completo

```
-- SUMADOR COMPLETO
--ZAVALA ROMAN IRVIN EDUARDO
library IEEE;
use IEEE.std_logic_1164.all;
entity adder is
port(
   a: in std_logic;
   b: in std_logic;
   cin: in std_logic;
   s: out std_logic;
   cout: out std_logic);
end adder;
architecture arch_adder of adder is
begin
   s <= a xor b xor cin;
   cout <=  (a and b) or ((a xor b) and cin);
end arch_adder;
```

Los multiplexores son circuitos combinacionales donde se tienen múltiples entradas y solo 1 salida. Esto nos permite conmutar entre las varias entradas y tener una sola salida, las entradas suelen ser en cantidades de 2^n donde n es mayor que 1. Para implementar esto en VHDL ocupamos crear un vector de 8 elementos donde estarían las entradas del multiplexor y un vector de 3 elementos en el cual se escoge el elemento del vector de 8 elementos.

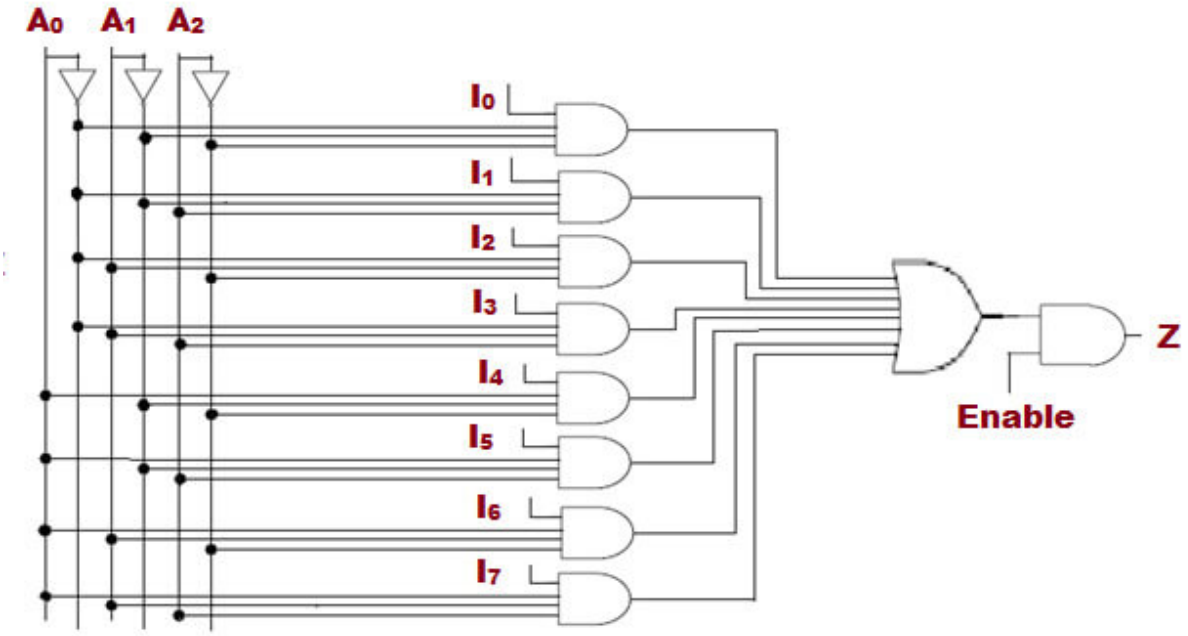

Figura 3. Circuito multiplexor 8 a 1, obtenido de https://electronicaonline.net/electronica-digital/multiplexor-y-demultiplexor-tipos-y-sus-diferencias/

```
-- Multiplexer 8 to 1 design
--ZAVALA ROMAN IRVIN EDUARDO
library IEEE;
use IEEE.std_logic_1164.all;
entity multiplexor_8_1 is
port(
    posibles_salidas: in std_logic_vector(7 downto 0);
    input: in std_logic_vector(2 downto 0);
    q: out std_logic);
end multiplexor_8_1;
architecture arch_multiplexor_8_1 of multiplexor_8_1 is
begin
        q <=    posibles_salidas(7) when (input = "000") else
                posibles_salidas(6) when (input = "001") else
                posibles_salidas(5) when (input = "010") else
                posibles_salidas(4) when (input = "011") else
                posibles_salidas(3) when (input = "100") else
                posibles_salidas(2) when (input = "101") else
                posibles_salidas(1) when (input = "110") else
                posibles_salidas(0) when (input = "111");

end arch_multiplexor_8_1;
```

Los resultados de las simulaciones de todos los circuitos son los siguientes:

```
testbench.vhd:34:9:@1ns:(assertion error): 000 funciona bn
testbench.vhd:37:9:@2ns:(assertion error): 001 funciona bn
testbench.vhd:40:9:@3ns:(assertion error): 010 funciona bn
testbench.vhd:43:9:@4ns:(assertion error): 011 funciona bn
testbench.vhd:46:9:@5ns:(assertion error): 100 funciona bn
testbench.vhd:49:9:@6ns:(assertion error): 101 funciona bn
testbench.vhd:52:9:@7ns:(assertion error): 110 funciona bn
testbench.vhd:55:9:@8ns:(assertion error): 111 funciona bn
testbench.vhd:58:8:@8ns:(assertion note): SIMULACION TERMINADA
```

Figura 4. Resultado multiplexor

```
testbench.vhd:27:6:@1ns:(assertion error): 0+0 funciona bn
testbench.vhd:33:6:@2ns:(assertion error): 0+1 funciona bn
testbench.vhd:39:6:@3ns:(assertion error): 1+0 funciona bn
testbench.vhd:45:6:@4ns:(assertion error): 1+1 funciona bn
testbench.vhd:51:6:@5ns:(assertion error): c=1 0+0 funciona bn
testbench.vhd:57:6:@6ns:(assertion error): c=1 0+1 funciona bn
testbench.vhd:63:6:@7ns:(assertion error): c=1 1+0 funciona bn
testbench.vhd:69:6:@8ns:(assertion error): c=1 1+1 funciona bn
testbench.vhd:76:6:@8ns:(assertion note): SIMULACION TERMINADA
```

Figura 5. Resultado sumador completo

```
testbench.vhd:25:6:@1ns:(assertion error): 0+0 funciona bn
testbench.vhd:30:6:@2ns:(assertion error): 0+1 funciona bn
testbench.vhd:35:6:@3ns:(assertion error): 1+0 funciona bn
testbench.vhd:40:6:@4ns:(assertion error): 1+1 funciona bn
testbench.vhd:47:6:@4ns:(assertion note): SIMULACION TERMINADA
```

Figura 3. Resultado medio sumador

```
testbench.vhd:28:7:@1ns:(assertion error): 0XX ta bn
testbench.vhd:34:7:@2ns:(assertion error): X0X ta bn
testbench.vhd:40:7:@3ns:(assertion error): XX0 ta bn
testbench.vhd:46:7:@4ns:(assertion error): 111 ta bn
testbench.vhd:54:7:@4ns:(assertion note): SIMULACION TERMINADA
```

Figura 3. Resultado AND 3 entradas

Conclusión

En VHDL es bastante sencillo programar elementos que usábamos para crear circuitos, aunque no sé el alcance de esta herramienta, con crear compuertas lógicas, flip flops, sumadores y multiplexores se me hace una herramienta bastante completa. Lo único complicado se me hace la parte de las pruebas ya que es donde más me perdía en el código, pero con un poco de práctica será solucionado.

Anexos

AND 2 entradas: https://www.edaplayground.com/x/Mp5s

AND 3 entradas: https://www.edaplayground.com/x/XX6p

Medio sumador: https://www.edaplayground.com/x/wjaQ

Sumador completo: https://www.edaplayground.com/x/pEkS

Multiplexor: https://www.edaplayground.com/x/UPaN