TP5 Resolución

miércoles, 6 de noviembre de 2024 16:13

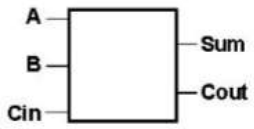
1) La principal diferencia entre una PAL y una GAL es:

- una PAL está diseñada para ecuaciones en forma de SOP mientras que una GAL está diseñada para POS.
- una GAL tiene una matriz AND y una OR, ambas reprogramables.
- una GAL puede implementar expresiones más complicadas, porque tiene mayor cantidad de términos producto que cualquier PAL.
- usa tecnología EECMOS, de modo que es reprogramable.

2) Conteste V o F:

- F Los ASICs poseen arquitecturas con conexiones fijas.
- V Un ASICs Full Custom está pensado para el desarrollo de un diseño en particular a gran escala.
- F Un ASICs Full Custom se utiliza para el diseño de prototipos.
- V Los Microprocesadores poseen arquitecturas fijas que no pueden reconfigurarse.
- F Un ASICs Full Custom es un circuito prediseñado en el cual el usuario establece las conexiones.
- V Las FPGA son configuradas por el usuario.
- V Las FPGA son muy utilizadas para pruebas y desarrollo de prototipos.
- F El consumo de potencia es mucho menor en las FPGA que en los circuitos Full-Custom.
- F Un microcontrolador (con las mismas características) implementado dentro de una FPGA es más rápido que un microcontrolador dedicado.

3) Construya en VHDL la arquitectura de un Sumador Completo a partir de la siguiente declaración de entidad:



A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

```
ENTITY SUMADOR_COMPLETO IS
    PORT ( A, B, Cin : IN BIT;
           Sum, Cout : OUT BIT );
END SUMADOR_COMPLETO;
```

a) Con procesos.
b) Sin procesos.

a)

```
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;
    S    : out std_logic;
    Cout : out std_logic
  );
end Sumador_Completo;

architecture Behavioral of Sumador_Completo is
begin

  Sumador_Completo_Proceso: process(A, B, Cin)
  begin
    case (A & B & Cin) is
      when "000" =>
```

```
            S <= '0';
            Cout <= '0';
         when "001" =>
            S <= '1';
            Cout <= '0';
         when "010" =>
            S <= '1';
            Cout <= '0';
         when "011" =>
            S <= '0';
            Cout <= '1';
         when "100" =>
            S <= '1';
            Cout <= '0';
         when "101" =>
            S <= '0';
            Cout <= '1';
         when "110" =>
            S <= '0';
            Cout <= '1';
         when "111" =>
            S <= '1';
            Cout <= '1';
         when others =>
            S <= '0';
            Cout <= '0';
      end case;
   end process Sumador_Completo_Proceso;
end Behavioral;
```

b)

```
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;
      S    : out std_logic;
      Cout  : out std_logic
   );
end Sumador_Completo;

architecture Behavioral of Sumador_Completo is
begin

   S    <= A xor B xor Cin;
   Cout  <= (A and B) or (A and Cin) or (B and Cin);
end Behavioral;
```

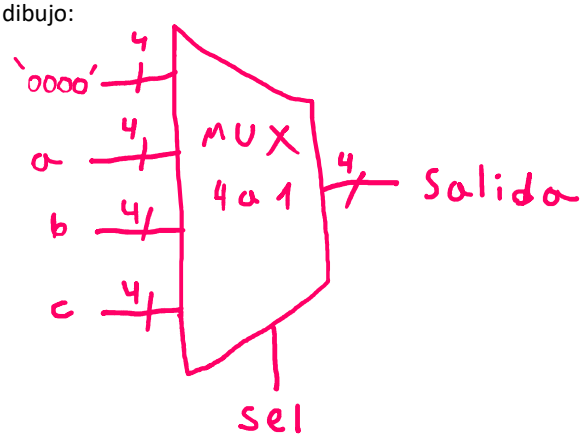
4) ¿Qué se sintetiza a partir de los siguientes códigos VHDL? construya la tabla de verdad para cada caso:

a)

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
ENTITY XX IS
    PORT(a : IN std_logic_vector(3 DOWNTO 0);
         b : IN std_logic_vector(3 DOWNTO 0);
         c : IN std_logic_vector(3 DOWNTO 0);
       sel : IN std_logic_vector(1 DOWNTO 0);
    salida : OUT std_logic_vector(3 DOWNTO 0));
END XX;

ARCHITECTURE synth OF XX IS
BEGIN
    PROCESS (sel, a, b, c) IS
    BEGIN
        CASE sel IS
            WHEN "00" => salida <= (others => '0');
            WHEN "01" => salida <= a;
            WHEN "10" => salida <= b;
            WHEN "11" => salida <= c;
            WHEN OTHERS => salida <= (others => '0');
        END CASE;
    END PROCESS;
END synth;
```

Implementa un MUX 4 a 1 de 4 canales con las conexiones del dibujo:

sel	salida
00	0
01	a
10	b
11	c

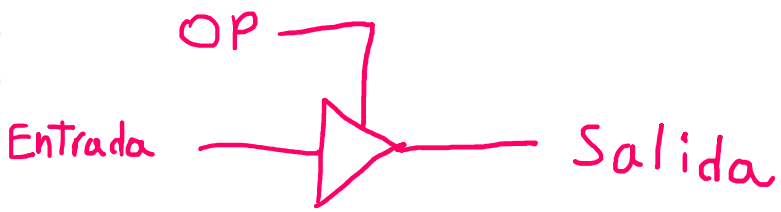


b)

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY XX IS
    PORT(op, entrada: IN std_logic;
                salida: OUT std_logic);
END XX;

ARCHITECTURE synth OF XX IS
BEGIN
    PROCESS(entrada,op)
    BEGIN
        IF op='1' THEN
            salida <= entrada;
        ELSE
            salida <= 'Z';
        END IF;
    END PROCESS;
END ARCHITECTURE synth;
```

Implementa un Buffer Tri-State

op	salida
0	Z
1	entrada



5) ¿Cuáles son los tres tipos de elementos programables que posee una FPGA?¿Qué funciones cumplen? ¿Cuáles son los componentes básicos de un bloque lógico de una FPGA?

Los Bloques Lógicos:
Son los bloques básicos, cumplen la función de realizar todas las operaciones lógicas de nuestro programa, y están compuestos (básicamente) por: –Look up Table (LUT) , Lógica de acarreo, Multiplexor y Registro.

Los Bloques de entrada/salida (IOB):
Son las interfaces entre los pines de los dispositivos FPGA y el circuito lógico. Se pueden configurar como entrada, salida, bidireccional o en estado tristate de manera independiente. Las opciones de configuración incluyen el ajuste de la velocidad de respuesta (slew rate), la habilitación de resistencias pull-up o pull-down, y otras funciones adicionales.

Las Lineas de Interconexión:
Cumplen la función de conectar las distintas partes de la FPGA entre sí, y estas con las entradas/salidas. Estas incluyen "líneas directas", que permiten la conexión inmediata entre bloques sin necesidad de pasar por la matriz de interconexión, y "líneas largas", que son conductores de gran longitud, tanto horizontales como verticales, que atraviesan el dispositivo de extremo a extremo.

6) ¿Qué es un lenguaje de descripción de hardware y en qué se diferencia con los lenguajes de programación?

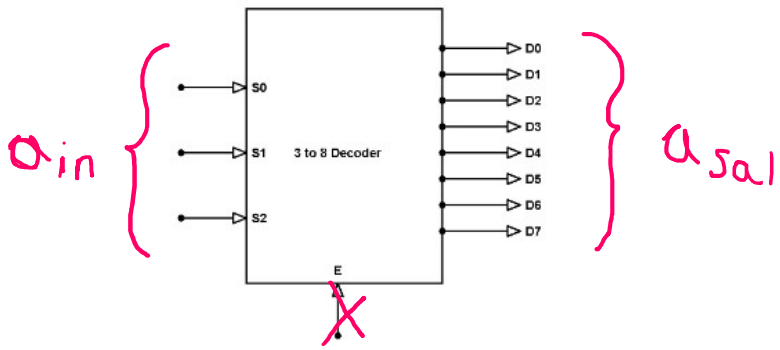
Un lenguaje de descripción de hardware es un tipo de lenguaje de programación utilizado para describir el comportamiento y la estructura de un sistema digital, como un circuito o un sistema electrónico. A diferencia de los lenguajes de programación convencionales, que se utilizan para desarrollar software que se ejecuta en procesadores, los HDL se utilizan para definir cómo interactúan los componentes de hardware, como las puertas lógicas, registros, y otros bloques de un sistema digital, y la principal diferencia funcional es que no ejecutan código de forma secuencial, sino que todo lo descripto en el código se realiza de forma paralela.

7) A partir de la siguiente arquitectura determine: cuántas entradas y salidas tendrá la entidad ¿Qué se sintetiza?

```vhdl
library ieee;
use ieee. std_logic_1164.all;
use ieee. numeric_std.all;

entity [redacted] is
PORT( a_in: in std_logic_vector(2 downto 0);
      a_sal: out std_logic_vector(7 downto 0));
end [redacted];

architecture uam of [redacted] is
begin
 [redacted] process (a_in) is
 variable i: integer;
       begin
       for i in 0 to 7 loop
              if i=to_integer(unsigned(a_in)) then
                     a_sal(i)<='1';
              else
                     a_sal(i)<='0';
              end if;
       end loop;
       end process [redacted]
end architecture uam;
```

Ayuda: La función **to_integer()** transforma una señal de tipo unsigned en un entero. La función **unsigned()** transforma una señal de tipo std_logic_vector en un unsigned.

Implementa un Decodificador 3 a 8. Tendrá 1 entrada de 3 bits y una salida de 8 bits.

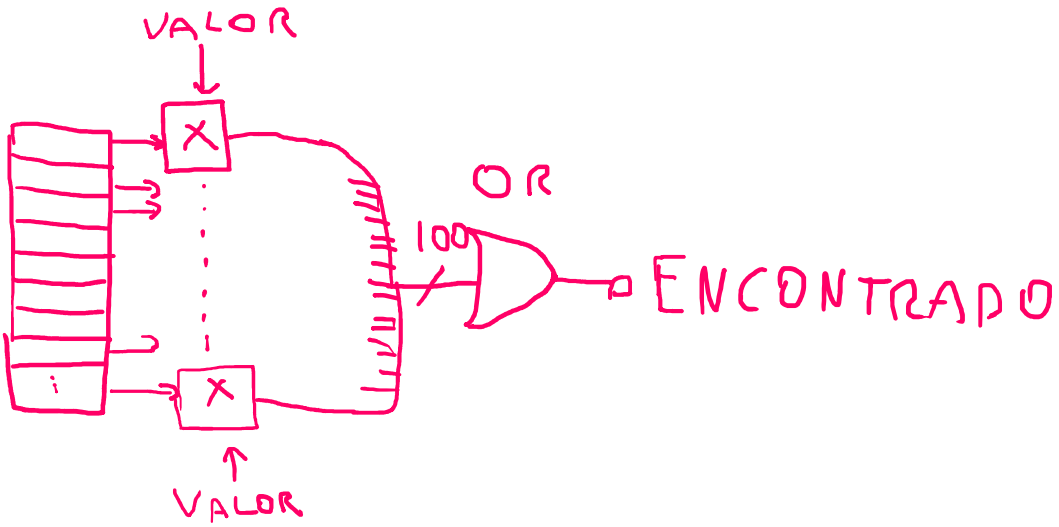


8) A partir de la siguiente entidad:

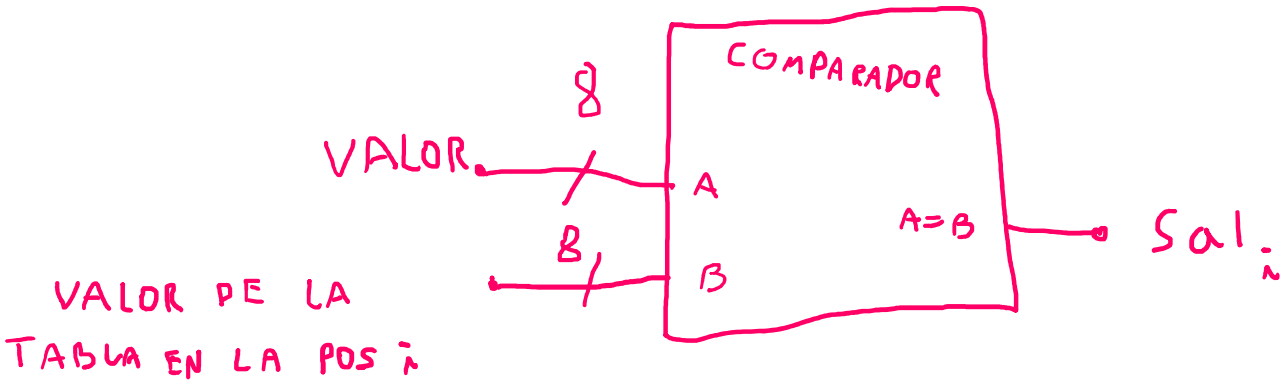
a) Realice un esquema de la entidad.
b) Conteste ¿El circuito que se sintetiza es sincrónico o combinacional? Justifique.
c) Conteste ¿A qué frecuencia máxima podrá operar el circuito?¿Cómo podría incrementarlo? Explique ventajas y desventajas de aplicar la solución sugerida.
d) Conteste ¿Qué tamaño deberá tener la memoria RAM donde se sintetiza la tabla?

```vhdl
ENTITY bucador is
    PORT(       valor: in std_logic_vector (7 downto 0);
          encontrado: out std_logic);
END bucador;

ARCHITECTURE behavioral OF bucador IS
    TYPE ram IS ARRAY (0 to 99) OF integer;
    SIGNAL tabla:ram;
BEGIN
    PROCESS(valor)
        variable pos:integer:=0;
        variable valor_i:integer;
    BEGIN
        valor_i:=to_integer(unsigned(valor));
        encontrado<='0';
        pos:=0;

        while valor_i /= tabla(pos) or pos<100 loop
            pos:=pos+1;
```

```
        pos:=0;

        while valor_i /= tabla(pos) or pos<100 loop
            pos:=pos+1;
        end loop;

        if pos<100 then
            encontrado<='1';
        end if;

    END PROCESS;
END behavioral;
```

ELEMENTO 'X':



a) Es combinacional, ya que si bien el process nos permite aplicar una lógica secuencial, esta luego al compilar se traduce a circuitos combinacionales y se ejecuta en forma paralela.

b) La frecuencia máxima dependerá de del retardo de la velocidad en que se puedan transferir los datos de la RAM a las LUT de la FPGA, del del retardo del comparador de 8 bits y luego del retardo del comparador de 100 entradas. Una forma de incrementar la misma sería cargar los datos previamente en la FPGA y luego realizar las comparaciones, reduciendo así los retardos generados al cargar los datos.
Las ventajas serían que se podría operar con esos datos de forma rápida e incrementar la frecuencia de operación

considerablemente.
La desventaja principal es que si se quisiera modificar algun dato de esta memoria, o si se produjera un cambio debido a algún otro proceso, se perdería la equivalencia de los datos, debiendo cargarlos nuevamente.

c) 100 posiciones x 8 bits = 800 bits

Sin embargo, como las RAM generalmente están dimensionadas en potencias de 2, en la práctica se necesitaría una RAM de al menos 128 posiciones de 8 bits, lo cual da un tamaño total de 1024 bits.