INGENIERÍA DE COMPUTADORES III

Solución al Ejercicio de Autocomprobación 9

PREGUNTA 1 (3 puntos)

1.a) (0.5 puntos) Diseñe en VHDL un multiplexor de dos señales de un bit. Hágalo describiendo el comportamiento del circuito, empleando una sentencia concurrente condicional (**when-else**). La **entity** se muestra a continuación. La señal `s` es la señal de control.

```
entity mux2_1bit is
   port (  d      : out std_logic;
           a, b   : in  std_logic;
           s      : in  std_logic );
end entity mux2_1bit;
```

1.b) (0.5 puntos) Diseñe en VHDL un multiplexor de dos señales de un bit mediante la descripción de su comportamiento, pero en esta ocasión empleando una sentencia concurrente de selección (**with-select**).

1.c) (2 puntos) Diseñe en VHDL un multiplexor de dos señales de cuatro bits. Hágalo describiendo la estructura del circuito. Es decir, mediante la conexión de cuatro multiplexores de dos señales de un bit. La **entity** del multiplexor de dos señales de cuatro bits es:

```
entity mux2_4bit is
   port (  d0, d1, d2, d3  : out std_logic;
           a0, a1, a2, a3  : in  std_logic;
           b0, b1, b2, b3  : in  std_logic;
           s               : in  std_logic );
end entity mux2_4bit;
```

Solución a la Pregunta 1

La solución a los tres apartados de la pregunta es el Código VHDL 1.1 – 1.3. El diagrama del multiplexor de 2 señales de 4 bits, diseñado mediante la conexión de 4 multiplexores de 2 señales de 1 bit, se muestra en la Figura 1.1.

```
-------------------------------------------------
-- MUX 2 a 1
library IEEE;
use IEEE.std_logic_1164.all;

entity mux2_1bit is
    port (   d       : out std_logic;
             a, b    : in   std_logic;
             s       : in   std_logic );
end entity mux2_1bit;

architecture mux_cond of mux2_1bit is
begin
    d <= a when ( s = '0') else
         b;
end architecture mux_cond;
-------------------------------------------------
```

Código VHDL 1.1: Solución al Apartado 1.a: MUX de 2 señales de 1 bit, descrito mediante una sentencia concurrente condicional (**when-else**).

```
-------------------------------------------------
-- MUX 2 a 1
library IEEE;
use IEEE.std_logic_1164.all;

entity mux2_1bit is
    port (   d       : out std_logic;
             a, b    : in   std_logic;
             s       : in   std_logic );
end entity mux2_1bit;

architecture mux_sel of mux2_1bit is
begin
   with s select
    d <= a when '0',
         b when others;
end architecture mux_sel;
-------------------------------------------------
```

Código VHDL 1.2: Solución al Apartado 1.b: MUX de 2 señales de 1 bit, descrito mediante una sentencia concurrente de selección (**with-select**).

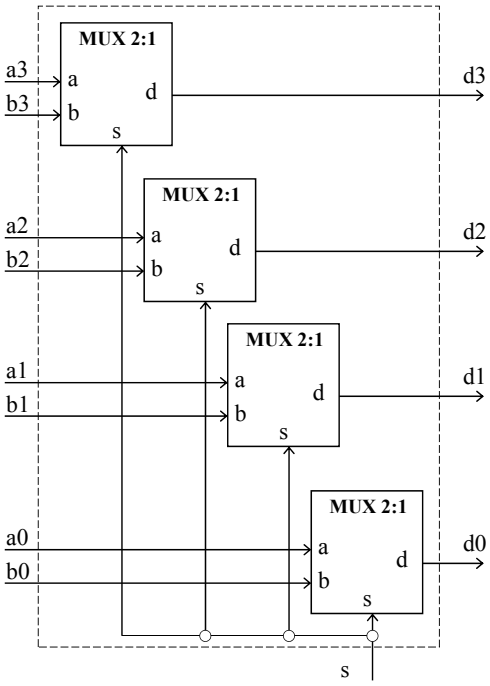


Figura 1.1: Multiplexor de 2 señales de 4 bits, diseñado mediante la conexión de 4 multiplexores de 2 señales de 1 bit.

```
-------------------------------------------
-- MUX 2:1 de 4 bit.
library IEEE;
use IEEE.std_logic_1164.all;

entity Mux2_4bit is
   port (   d0, d1, d2, d3   : out    std_logic;
            a0, a1, a2, a3   : in     std_logic;
            b0, b1, b2, b3   : in     std_logic;
            s                : in     std_logic );
end entity Mux2_4bit;

architecture Mux2_4bit of Mux2_4bit is
    component Mux2_1bit is
       port (   d        : out    std_logic;
                a, b : in     std_logic;
                s        : in     std_logic );
    end component Mux2_1bit;
begin
    Mux2_0  : Mux2_1bit   port map ( d => d0, a => a0, b => b0 , s => s );
    Mux2_1  : Mux2_1bit   port map ( d => d1, a => a1, b => b1 , s => s );
    Mux2_2  : Mux2_1bit   port map ( d => d2, a => a2, b => b2 , s => s );
    Mux2_3  : Mux2_1bit   port map ( d => d3, a => a3, b => b3 , s => s );
end architecture Mux2_4bit;
-------------------------------------------
```

Código VHDL 1.3: Solución al Apartado 1.c: descripción estructural de un multiplexor de 2 señales de 4 bits.

PREGUNTA 2 (3 puntos)

Describa en VHDL el comportamiento de un circuito con dos entradas de 8 bits que, interpretando las entradas como números binarios sin signo, calcule el valor absoluto de la diferencia entre las entradas. Es decir, si las entradas son a, b, el circuito calcula $|a - b|$. La **entity** del circuito es:

```
entity abs_dif is
   port (  resultado  : out std_logic_vector(7 downto 0);
           a, b       : in  std_logic_vector(7 downto 0) );
end entity abs_dif;
```

Solución a la Pregunta 2

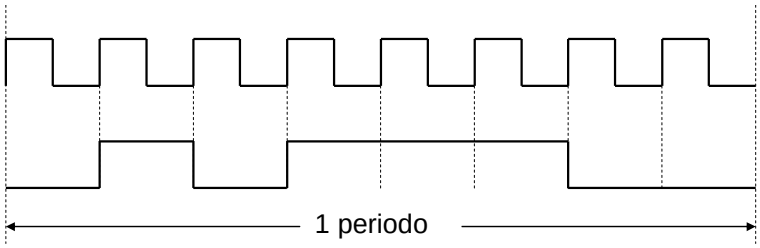
Existen varias formas de diseñar el circuito. Una de ellas es Código VHDL 1.4.

```
----------------------------------------
-- Valor absoluto de la diferencia
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity abs_dif is
    port (  resultado  : out   std_logic_vector(7 downto 0);
            a, b       : in    std_logic_vector(7 downto 0) );
end entity abs_dif;

architecture abs_dif1 of abs_dif is
    signal au, bu, ru, difab, difba: unsigned(7 downto 0);
begin
    au <= unsigned(a);
    bu <= unsigned(b);
    difab <= au - bu;
    difba <= bu - au;
    ru <= difab when (au >= bu) else
          difba;
    resultado <= std_logic_vector(ru);
end architecture abs_dif1;
```

Código VHDL 1.4: Diseño solución a la Pregunta 2.

PREGUNTA 3 (3 puntos)

Diseñe un generador de señales que obtenga la forma de onda mostrada en la parte inferior de la figura, a partir de la señal de reloj `clk` mostrada en la parte superior de la figura. Describa el comportamiento del circuito como una máquina de estado finito.



La **entity** del circuito es:

```
entity generador is
   port( wave :   out std_logic;
         clk  :   in  std_logic );
end entity generador;
```

Solución a la Pregunta 3

```
-------------------------------------------------------
-- Paquete con la definición de las constantes globales
library IEEE;
use IEEE.std_logic_1164.all;
package STATE_CONST is
   constant STATE_BITS: integer := 3;      -- Bits codifican estado
   constant S0: std_logic_vector(2 downto 0) := "000"; -- Estados
   constant S1: std_logic_vector(2 downto 0) := "001";
   constant S2: std_logic_vector(2 downto 0) := "010";
   constant S3: std_logic_vector(2 downto 0) := "011";
   constant S4: std_logic_vector(2 downto 0) := "100";
   constant S5: std_logic_vector(2 downto 0) := "101";
   constant S6: std_logic_vector(2 downto 0) := "110";
   constant S7: std_logic_vector(2 downto 0) := "111";
end package;
-------------------------------------------------------
```

Código VHDL 1.5: Paquete con la definición de constantes. Diseño solución a la Pregunta 3.

```
-------------------------------------------------------
---Generador de forma de onda, implementado como máquina de estados
library IEEE;
use IEEE.std_logic_1164.all;
use work.STATE_CONST.all;

entity generador is
   port(wave : out std_logic;
        clk  : in std_logic);
end entity generador;

architecture fsm of generador is
       signal state : std_logic_vector(STATE_BITS-1 downto 0);
       signal temp_wave : std_logic;
begin
--Cálculo del próximo estado
proximo_estado: process (clk)
begin
      if (rising_edge(clk)) then
           case state is
                when S0 => state <= S1; wave <= '0';
                when S1 => state <= S2; wave <= '1';
                when S2 => state <= S3; wave <= '0';
                when S3 => state <= S4; wave <= '1';
                when S4 => state <= S5; wave <= '1';
                when S5 => state <= S6; wave <= '1';
                when S6 => state <= S7; wave <= '0';
                when others => state <= S0; wave <= '0';
           end case;
      end if;
end process proximo_estado;

end architecture fsm;
-----------------------------------------------
```

Código VHDL 1.6: Solución a la Pregunta 3: diseño del generador de la forma de onda.

PREGUNTA 4 (1 punto)

Programe en VHDL un banco de pruebas para el circuito que ha diseñado al responder a la pregunta 3, de manera que pueda comprobarse mediante inspección visual que el circuito genera la forma de onda correctamente.

Solución a la Pregunta 4

```
-------------------------------------
-- Banco de pruebas del generador de la forma de onda
library IEEE;
use IEEE.std_logic_1164.all;

entity bp_generador is
end entity bp_generador;

architecture bp_generador of bp_generador is
   constant PERIODO    : time         := 100 ns; -- Reloj
   signal   wave       : std_logic; -- Salida UUT
   signal   clk        : std_logic   := '0'; -- Entrada UUT

   component generador is
       port ( wave : out std_logic;
              clk  : in std_logic);
   end component generador;
begin
   -- Instanciar y conectar UUT
   uut : component generador port map
                     (wave,  clk);
   clk <= not clk after (PERIODO/2);
end architecture bp_generador;
-------------------------------------
```

Código VHDL 1.7: Solución a la Pregunta 4: banco de pruebas.