UNIVERSIDAD NEBRIJA

Hoja de Ejercicios 3

1. Dibuja el esquemático del circuito descrito por el código HDL siguiente. Simplifica el esquemático para que muestre un número mínimo de puertas.

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

entity exercise1 is
   port(a, b, c: in STD_LOGIC;
        y, z: out STD_LOGIC);
end;

architecture synth of exercise1 is
begin
   y <= (a and b and c) or (a and b and not c) or
        (a and not b and c);
   z <= (a and b) or (not a and not b);
end;
```

2. Dibuja el esquemático del circuito descrito por el código HDL siguiente. Simplifica el esquemático para que muestre un número mínimo de puertas.

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

entity exercise2 is
     port(a: in STD_LOGIC_VECTOR(3 downto 0);
     y: out STD_LOGIC_VECTOR(1 downto 0));
end;

architecture synth of exercise2 is
begin
 process(a)
 begin
     if a(0)='1' then y <= "11";
     elsif a(1)='1' then y <= "10";
     elsif a(2)='1' then y <= "01";
     elsif a(3)='1' then y <= "00";
     else y <= a(1 downto 0);
     end if;
 end process;
end;
```

3. Escribe un módulo HDL que calcule una función XOR de cuatro entradas. La entrada es *a3:0*, y la salida es *y*.

4. Escribe un multiplexor 8:1 llamado mux8 con entradas *s2:0, d0, d1, d2, d3, d4, d5, d6, d7*, y salida *y*.

5. Escribe un módulo estructural que calcule la función lógica $y = a\bar{b} + \bar{b}\,\bar{c} + \bar{a}bc$ usando el multiplexor 8:1 del ejercicio anterior.

6. En VHDL, ¿por qué puede ser necesario escribir la siguiente línea?

```
q <= '1' when state = S0 else '0';
```

   en vez de simplemente poner

```
q <= (state = S0);
```

7. Cada uno de los siguientes módulos HDL contienen un error. Por brevedad solo se muestra la arquitectura. Asume que las cláusulas de librarías y la declaración de entidad son correctas. Explica el error y cómo solucionarlo:

(a)
```
architecture synth of latch is
   begin
          process(clk) begin
                 if clk = '1' then q <= d;
                 end if;
          end process;
   end synth;
```

(b)
```
architecture proc of gates is
   begin
          process(a) begin
                 Y1 <= a and b;
                 y2 <= a or b;
                 y3 <= a xor b;
                 y4 <= a nand b;
                 y5 <= a nor b;
          end process;
   end proc;
```

(c)
```
architecture synth of flop is
   begin
          process(clk)
                 if rising_edge(clk) then
                        q <= d;
                 end;
```

(d)
```
architecture synth of priority is
   begin
          process(all) begin
                 if a(3) then y <= "1000";
                 elsif a(2) then y <= "0100";
                 elsif a(1) then y <= "0010";
                 elsif a(0) then y <= "0001";
                 end if;
          end process;
   end synth;
```

(e)
```
architecture synth of divideby3FSM is
   type statetype is (S0, S1, S2);
   signal state, nextstate: statetype;
   begin
          process(clk, reset) begin
                 if reset then state <= S0;
                 elsif rising_edge(clk) then
                        state <= nextstate;
                 end if;
          end process;
          process(state) begin
                 case state is
                        when S0 => nextstate <= S1;
                        when S1 => nextstate <= S2;
                        when S2 => nextstate <= S0;
                 end case;
          end process;
          q <= '1' when state = S0 else '0';
   end synth;
```

(f)
```
architecture asynchronous of floprs is
   begin
          process(clk, reset) begin
                 if reset then
                        q <= '0';
                 elsif rising_edge(clk) then
                        q <= d;
                 end if;
          end process;
          process(set) begin
                 if set then
                        q <= '1';
                 end if;
          end process;
   end asynchronous;
```