Hoja de Problemas 1 Tecnología de Computadores

1. Dibuje el esquemático del circuito descrito por el código VHDL de la Figura 1. Simplifique primero las ecuaciones lógicas para obtener el circuito mínimo.

```
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;
```

Figura 1

2. Escriba un módulo en VHDL que realice la función XOR para cuatro entradas. La entrada del módulo es "$a_{3:0}$" y la salida es "*y*".

3. Escriba un módulo en VHDL llamado "*minority*" con tres entradas A, B y C, y una salida Z. La salida del módulo es TRUE cuando al menos dos de las tres entradas son FALSE.

4. Escriba el código VHDL de un decodificador de 3:8.

5. Escriba el código VHDL de un flip-flop JK. Este tipo de flip-flops tiene 3 entradas (J, K y CLK) y una salida (Q). En el flanco de reloj de subida, Q mantiene el valor anterior si J = K = 0. Si J = 1, entonces Q = 1 y si K = 1 entonces Q = 0. Si ambas señales J = K = 1, entonces el valor de Q es invertido.

6. Escriba el código VHDL de un flip-flop D con señales de set (S) y reset (R). De tal modo que si se activan ambas a la vez la salida del flip-flop (Q) será el valor de la entrada D pero invertido. En los otros casos funciona normalmente según los valores de D, S y R.

7. Dibuje el diagrama de estados de la máquina de estados descrita por el código VHDL de la Figura 2.

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

entity fsml is
  port(clk, reset:   in  STD_LOGIC;
       taken, back:  in  STD_LOGIC;
       predicttaken: out STD_LOGIC);
end;

architecture synth of fsml is
  type statetype is (S0, S1, S2, S3, S4);
  signal state, nextstate: statetype;
begin
  process(clk, reset) begin
    if reset = '1' then state <= S2;
    elsif clk'event and clk = '1' then
      state <= nextstate;
    end if;
  end process;

  process(state, taken) begin
    case state is
      when S0 => if taken = '1' then
                      nextstate <= S1;
                 else nextstate <= S0;
                 end if;
      when S1 => if taken = '1' then
                      nextstate <= S2;
                 else nextstate <= S0;
                 end if;
      when S2 => if taken = '1' then
                      nextstate <= S3;
                 else nextstate <= S1;
                 end if;
      when S3 => if taken = '1' then
                      nextstate <= S4;
                 else nextstate <= S2;
                 end if;
      when S4 => if taken = '1' then
                      nextstate <= S4;
                 else nextstate <= S3;
                 end if;
      when others => nextstate <= S2;
    end case;
  end process;

  -- output logic
  predicttaken <= '1' when
                  ((state = S4) or (state = S3) or
                  (state = S2 and back = '1'))
    else '0';
end;
```

Figura 2

8. Dibuje el diagrama de estados asociado a la máquina de estados descrita por el código VHDL de la Figura 3.

```
-- library declaration
library IEEE;
use IEEE.std_logic_1164.all;
-- entity
entity fsm is
   port (  X,CLK : in  std_logic;
           RESET : in  std_logic;
           Z1,Z2 : out std_logic;
end fsm;
-- architecture
architecture fsm of fsm is
   type state_type is (A,B,C);
   signal PS,NS : state_type;
begin
   sync_proc: process(CLK,NS,RESET)
   begin
      if (RESET = '0') then PS <= C;
      elsif (rising_edge(CLK)) then PS <= NS;
      end if;
   end process sync_proc;

   comb_proc: process(PS,X)
   begin
      case PS is
         Z1 <= '0';    Z2 <= '0';
         when A =>
            Z1 <= '0';
            if (X='0') then NS<=A; Z2<='1';
            else  NS <= B; Z2 <= '0';
            end if;
         when B =>
            Z1 <= '1';
            if (X='0') then NS<=A; Z2<='0';
            else  NS <= C; Z2 <= '1';
            end if;
         when C =>
            Z1 <= '1';
            if (X='0') then NS<=B; Z2<='1';
            else  NS <= A; Z2 <= '0';
            end if;
         when others =>
            Z1 <= '1'; NS<=A; Z2<='0';
      end case;
   end process comb_proc;
end fsm;
```

Figura 3

9. Escriba el código VHDL que describe a la máquina de estados de la Figura 4.

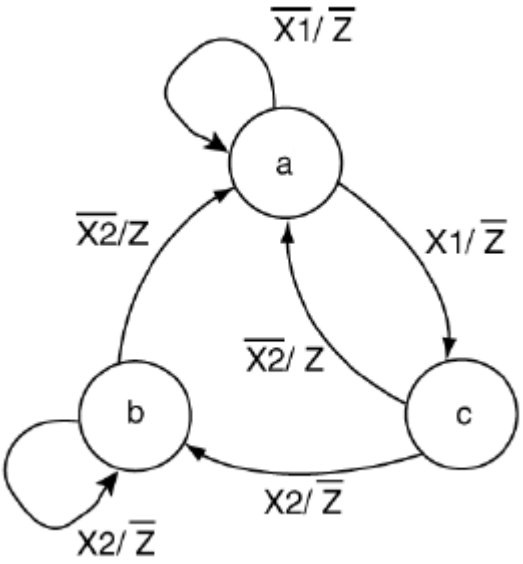

Figura 4

10. Escriba el código VHDL que describe a la máquina de estados de la Figura 5.

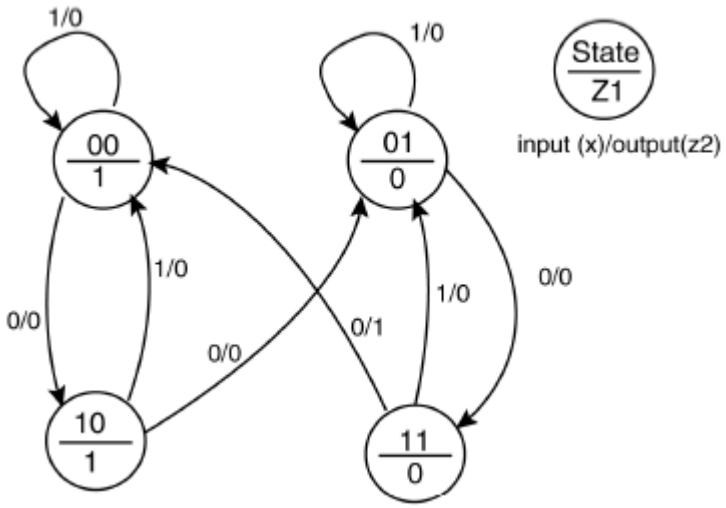

Figura 5