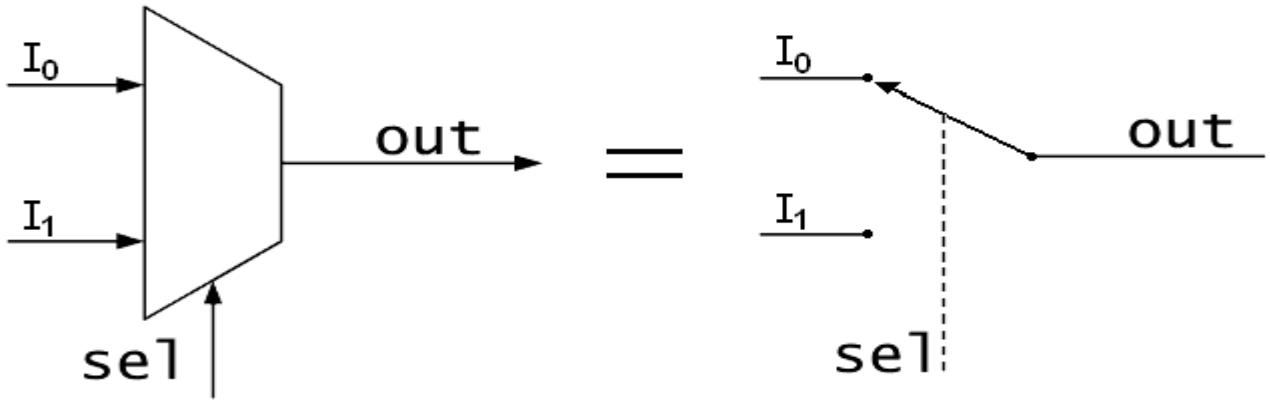
Multiplexores

¿Qué es un multiplexor?

En electronica digital un multiplexor equivale a un conmutador.

El multiplexor consta de:
N entradas de datos(I0,I1,I2, In...)
M entradas de selección (S0,S1,S2, Sn)
Una única salida Z
Un Enable (multiplexor encendido o apagado)



La relación entrada-selectores de 2^n

2^n entradas = n selectores

Por ello las entradas normalmente son potencias de 2, para 1 entrada de selección, 2 combinaciones(2-input MUX), para 2 entradas de selección 4 combinaciones (4-input MUX), para 3 entradas 8 combinaciones (8-iinput MUX) etc..

Enable:

El enable es una entrada de un solo bit, a 0 o a 1 que nos sirve para activar o desactivar el multiplexor.

Enable = 0 Desactivado
Enable = 1 Activado

Para cambiar esto podemos introducir un inversor a la entrada del enable y de esta manera:

Enable = 0 Activado
Enable = 1 Desactivado

Cuando tengamos el multiplexor este habilidado por el enable, la salida Z depende del valor de la entrada de seleción, que habilita las diferentes entradas dependiendo del valor en binario de las entradas de selección. La función de Z queda así:

$$Z = S1\ S0\ I0 + S0\ S1\ I1 + S1\ S0\ I2 + S1\ S1\ I3$$

Tabla de la verdad de un multiplexor con dos entradas de selección

S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

Puertas logicas: Implementacioón

Tenemos cuatro puertas AND, las cuales con que haya un 0 no dejan pasar la I correspondiente. Por ello, mirando la tablad de la verdad, si entra un 0, le ponemos un inversor para que se active la puerta y las demás no.

S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

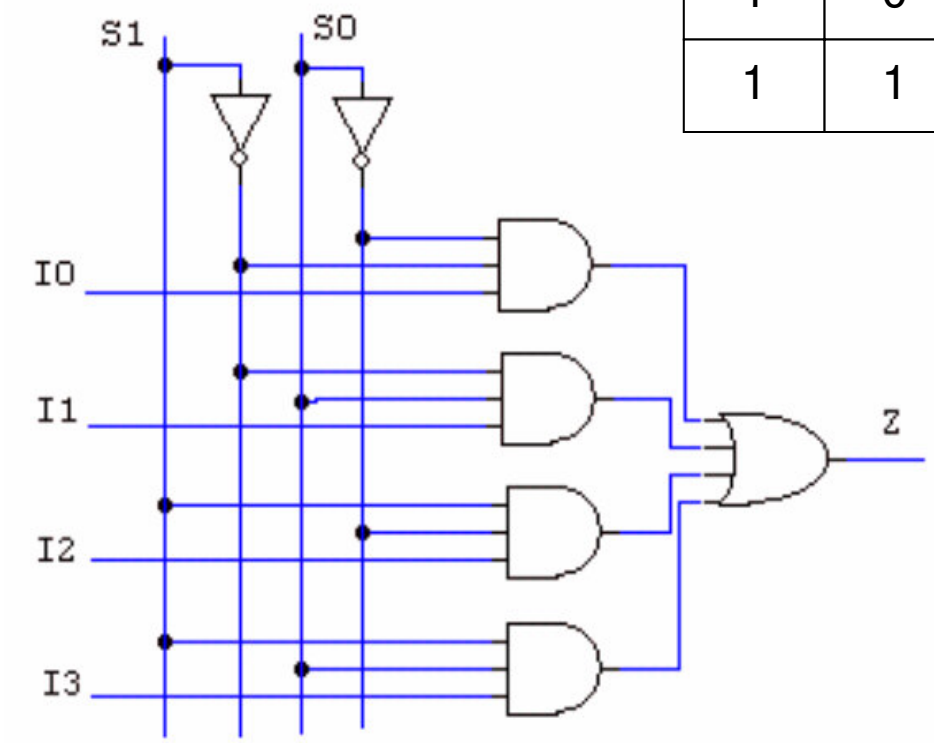
Ejemplo: para 01, S1 es 0, por lo cual esta puesta con un invesor, para que de 1, y el s0 entra con su valor original, 1, por lo que la puerta esta activa, y pasa la señal I1. Todas las demás puertas estan cerradas.

00

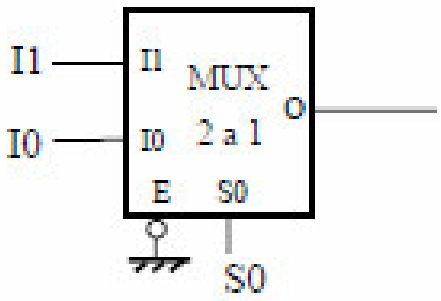
01

10

11



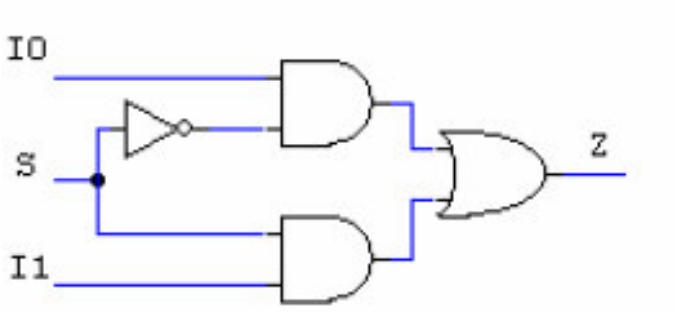
Multiplexor 2 a 1



S0	Z
0	I0
1	I1

$$Z = \overline{S0}\ I0 + S0\ I1$$

Implementación :

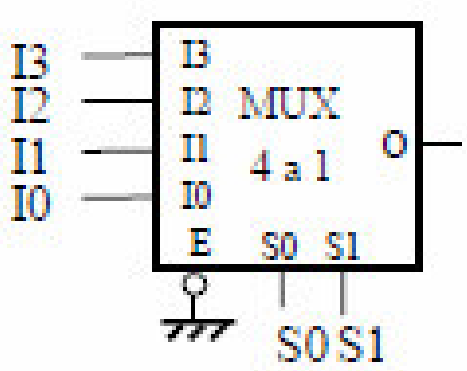


Descripción VHDL:

```
entity mux2a1 is port(
   I0, I1 : in  bit;
   S : in  bit;
   Z : out bit);
end mux2a1;
Architecture mux2_1 of
mux2a1 is
Begin
Process (I0,I1,S)
   Begin
      If (S='0') then Z<=I0;
      Else Z<=I1;
      End if;
End Process;
End mux2_1;
```

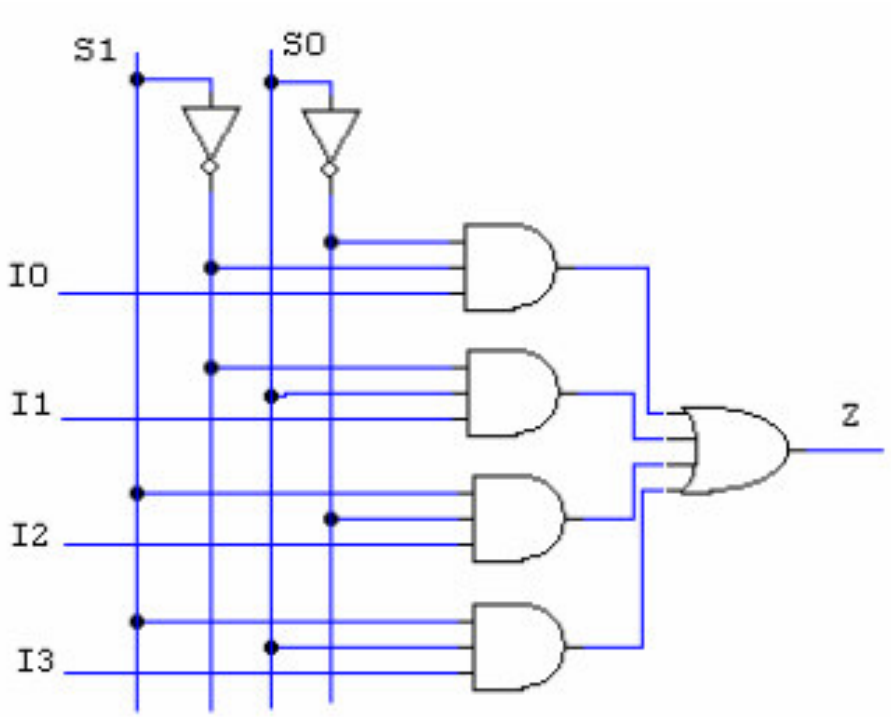
Multiplexor 4 a 1

S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3



$$Z = \overline{S1}\ \overline{S0}\ I0 + \overline{S1}\ S0\ I1 + S1\ \overline{S0}\ I2 + S1\ S0\ I3$$

Implementación :

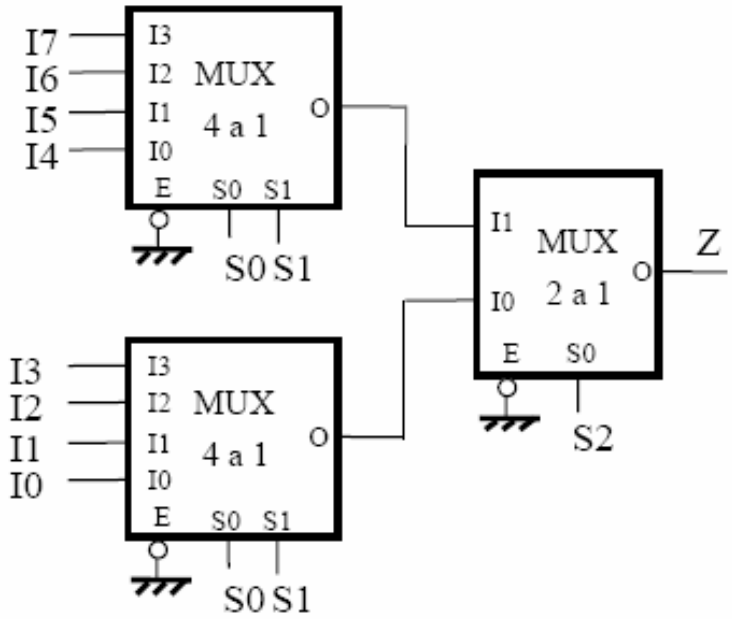


VHDL de un multiplexor 4 a 1:

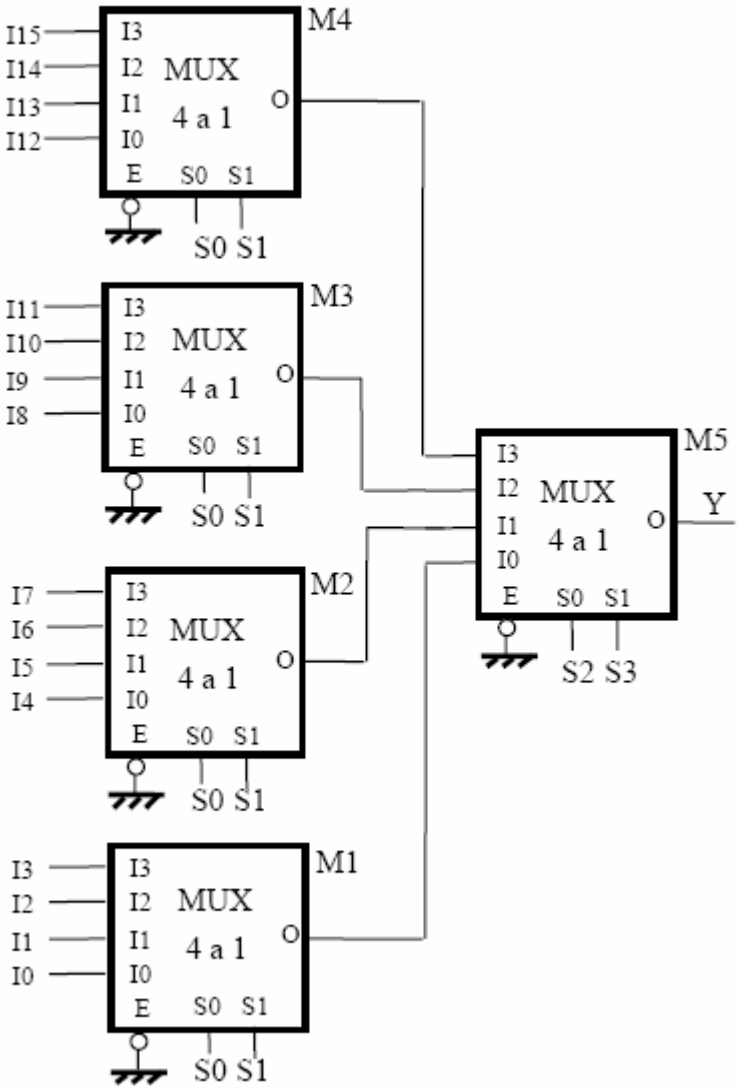
```
entity mux4 is port(
     I :in  std_logic_vector(3 downto 0);

     E:in  std_logic;
     S :in  std_logic_vector(1 downto 0);
     z : std_logic);
     end multi;
     architecture comportamiento of mux4 is
     begin
     process (I, S, E)
     begin
     if E='1' then z<='0';
     elsif E='0' then
        case S is
           when "00" => z <= I(0);
           when "01" => z <= I(1);
           when "10" => z <= I(2);
           when  "11" => z <= i(3);
         when others => z <= '0';
        end case;
     end if;
     end process;
     end comportamiento;
```

Multiplexores de X entradas

Para conseguir multiplexores con X entradas haremos combinaciones con diferentes multiplexores:



Multiplexor de 8 entradas



Multiplexor de 16 entradas

Multiplexores de X entradas

Descripción VHDL:

```
entity muxN_1 is

port(I0,I1,I2,...I2^N:  in bit_vector(2^N-1  downto 0);
     S:in bit_vector(N-1 downto 0);
     E:in bit;
     Z:out bit_vector);
end muxN_1;
Architecture comportamiento of muxN_1 is
Begin
Process (I0,I1,I2;...I2^N-1,S,E)
     begin
     if E=`1' then Z<='0';
     else
     Case S is
          When "00"=>Z<=I0;
          ...
          When "N"=>Z<=I2^N-1;
     end case;
     end if;
End process;
End comportamiento;
```

Dependiendo del numero de entradas al multiplexor el numero de entradas de control variara en 2^N, siendo N el numero de entradas.

Descripción VHDL de un Multiplexor de 16 entradas:

```
entity mux16 is port(
    I :in  std_logic_vector (15 downto 0);

    E  :in  std_logic;
    S :in  std_logic_vector (3 downto 0);
    Z :out std_logic;
    end mux16;
    architecture comportamiento of mux16 is
    begin
    process (I, S, E)
    begin
    if E='1' then z<='0';
    elsif E='0' then
        case S is
            when "0000" => z <= I(0);
            when "0001" => z <= I(1);
            when "0010" => z <= I(2);
            when "0011" => z <= i(3);
            when "0100" => z <= I(4);
            when "0101" => z <= I(5);
            when "0110" => z <= I(6);
            when "0111" => z <= i(7);
            when "1000" => z <= I(8);
            when "1001" => z <= I(9);
            when "1010" => z <= I(10);
            when "1011" => z <= i(11);
            when "1100" => z <= I(12);
            when "1101" => z <= I(13);
            when "1110" => z <= I(14);
            when "1111" => z <= i(15);
           when others => z <= '0';
        end case;
    end if;
    end process;
    end comportamiento;
```

Problemas propuestos

- Construir un multiplexor de 5 entradas
  a) utilizando puertas lógicas.
  b) utilizando multiplexores de dos entradas.

- Un circuito de "desplazamiento en barril" ("barrel-shifter") mueve los datos de entrada de forma que aparezcan en la salida girados el número de posiciones marcados por las señales de control. Construir utilizando multiplexores un "barrel-shifter" de 4 bits de entrada (a3a2a1a0) y 4 bits de salida (z3z2z1z0) con 4 posibles desplazamientos (dos señales de control c1c0):
  (c1c0) = 0 => (z3z2z1z0) = (a3a2a1a0),
  (c1c0) = 1 => (z3z2z1z0) = (a2a1a0a3),
  (c1c0) = 2 => (z3z2z1z0) = (a1a0a3a2),
  (c1c0) = 3 => (z3z2z1z0) = (a0a3a2a1).
  Realizar la descripción VHDL de este circuito.

SOLUCION PROBLEMA 2

C1	C0	Z3	Z2	Z1	Z0
0	0	A3	A2	A1	A0
0	1	A2	A1	A0	A3
1	0	A1	A0	A3	A2
1	1	A0	A3	A2	A1

VHDL:

```
Library ieee;
Use.ieee.std_logic_1164.all;
Entity BARRELSHIFTER is
    port(A:in std_logic_vector(3 downto 0);
        C:in std_logic_vector(1 downto 0);
        E:in std_logic;
        Z:out std_logic_vector(3 downto 0));
    end BARRELSHIFTER;
```

```
Architecture FUNCIONAMIENTO of BARRELSHIFTER is
      Begin
      Process (A,C,E)
            begin
            if E=`1´ then  Z<=`0´;
            else
            case C is
            when ¨00¨=>Z(3)<=A(3);
                        Z(2)<=A(2);
                        Z(1)<=A(1);
                        Z(0)<=A(0);
            when ¨01¨=>Z(3)<=A(2);
                        Z(2)<=A(1);
                        Z(1)<=A(0);
                        Z(0)<=A(3);
            when ¨10¨=>Z(3)<=A(1);
                        Z(2)<=A(0);
                        Z(1)<=A(3);
                        Z(0)<=A(2);
            when ¨11¨=>Z(3)<=A(0);
                        Z(2)<=A(3);
                        Z(1)<=A(2);
                        Z(0)<=A(1);
            when others =>Z<=`0´;
            end case;
            end if;
      end process;
End funcionamiento;
```

C1	C0	Z3	Z2	Z1	Z0
0	0	A3	A2	A1	A0
0	1	A2	A1	A0	A3
1	0	A1	A0	A3	A2
1	1	A0	A3	A2	A1

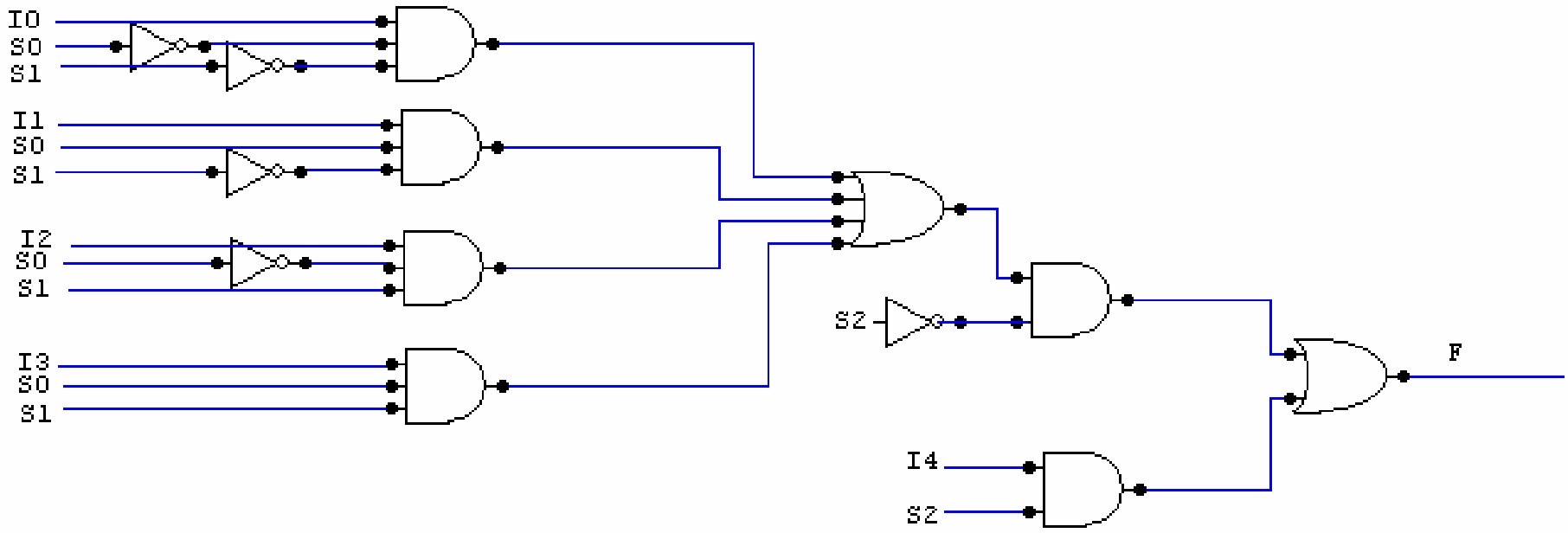
Problemas propuestos

- Construir un multiplexor de 5 entradas
  a) utilizando puertas lógicas.
  b) utilizando multiplexores de dos entradas.

- Un circuito de "desplazamiento en barril" ("barrel-shifter") mueve los datos de entrada de forma que aparezcan en la salida girados el número de posiciones marcados por las señales de control. Construir utilizando multiplexores un "barrel-shifter" de 4 bits de entrada (a3a2a1a0) y 4 bits de salida (z3z2z1z0) con 4 posibles desplazamientos (dos señales de control c1c0):
  (c1c0) = 0 => (z3z2z1z0) = (a3a2a1a0),
  (c1c0) = 1 => (z3z2z1z0) = (a2a1a0a3),
  (c1c0) = 2 => (z3z2z1z0) = (a1a0a3a2),
  (c1c0) = 3 => (z3z2z1z0) = (a0a3a2a1).
  Realizar la descripción VHDL de este circuito.

SOLUCION PROBLEMA 1

a) Puertas Logicas:



Problemas propuestos

- Diseñar un circuito multiplexor con prioridad de 4 bits. El circuito tiene 4 entradas de datos (I3-I0), 4 entradas de selección (S3-S0) y dos salidas Z y G. Cuando una o más de las entradas S están a 1, Z toma el valor de la entrada Ii, siendo i es el índice más alto de las entradas Si que están a 1; si todas las entradas S3-S0 están a 0, entonces Z toma el valor 0. La salida G se fija a 1 si al menos alguna entrada Si está a 1, en caso contrario se fija a 0.

  a) Mostrar en una tabla el comportamiento lógico del circuito. Encontrar las ecuaciones lógicas de la salidas Z y G expresándolas en dos niveles y en forma factorizada.

  b) Implementar la expresión factorizada de Z utilizando multiplexores de 2 entradas.

  c) Diseñar un multiplexor con prioridad de 16 bits en base a los multiplexores con prioridad de 4 bits diseñados.

  d) Realizar una descripción VHDL del multiplexor con prioridad.

Problemas propuestos

Implementar con multiplexores 2 a 1, un multiplexor de 2 entradas de 4 bits, un habilitador E(el circuito queda deshabilitado con 1), un selector S de 1 bit y una salida Z de 4 bits.