Laboratorio 03: Decodificadores, multiplexores y demultiplexores

Por:
Ian Gabriel Cañas Fernández, 1092228

.
.

Decodificadores, multiplexores y demultiplexores
Trabajando con vectores, procesos, prioridades y selección de camino.

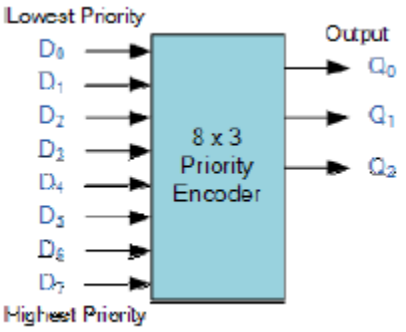
Objetivos generales:

- Conocer, analizar y comprobar el funcionamiento de multiplexores y demultiplexores e implementarlos en el kit de desarrollo de la FPGA.
- Conocer y comprobar los decodificadores de binario a decimal mediante el uso de procesos y buses.
- Introducir la opción de poder seleccionar una salida entre dos entradas dadas.
- Se procura destacar o decidir por donde saldría la señal de entrada mediante un selector.

Procedimiento:

Codificador

Para el desarrollo del código se precisó comprender que el codificador devuelve el valor correspondiente al número de la entrada introducida más alta de la siguiente manera:



Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

Figura 1. comportamiento de un codificador.

Por lo que la manera más optima es una comprobación secuencial que pare automáticamente se tenga el requerimiento deseado.

```vhdl
entity Ej1 is
Port (
        Q: out std_logic_vector (2 downto 0);
        D: in std_logic_vector (7 DOWNTO 0)
        );
end Ej1;

architecture Behavioral of Ej1 is
begin

process(D)
begin
if D(7) = '1' then
    Q <= "111";
elsif D(6) = '1' then
    Q <= "110";
elsif D(5) = '1' then
    Q <= "101";
elsif D(4) = '1' then
    Q <= "100";
elsif D(3) = '1' then
    Q <= "011";
elsif D(2) = '1' then
    Q <= "010";
elsif D(1) = '1' then
    Q <= "001";
elsif D(0) = '1' then
    Q <= "000";
else
    Q <= "000";
end if;
end process;

end Behavioral;
```

```
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { D[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { D[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { D[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { D[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { D[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { D[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { D[6] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { D[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
#set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { SW[8] }]; #IO_L24N_T3_34 Sch=sw[8]
#set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { SW[9] }]; #IO_25_34 Sch=sw[9]
#set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
#set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { E }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
#set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { NumBin[3] }]; #IO_L24P_T3_35 Sch=sw[12]
#set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { NumBin[2] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
#set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { NumBin[1] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
#set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { NumBin[0] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { Q[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { Q[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { Q[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
```

Nota: es cierto que la última comprobación y el else pueden ser fusionados, sin embargo, no se hace para distinguir que el 000 correspondiente a la entrada 0 es por ser la codificación de la entrada, mientras que el 000 del else es por no quedar más opciones.

Decodificador

El decodificador toma una entrada que representa un número binario, dicho dicho valor sería el valor de la salida a ser activada en LOW, es decir, 0.

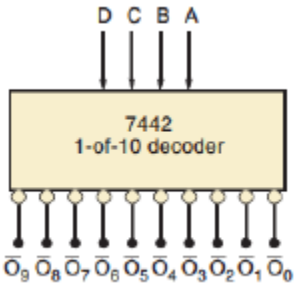


Figura 2. Representación gráfica de un decodificador.

Para el desarrollo del codificador se introduce el comportamiento de case, es parecido a un if, solo que la comprobación que hace no es una condición como tal, sino que confirma si la variable evaluada posee dicho valor.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;
use IEEE.std_logic_unsigned.ALL;

entity Ej2 is
 Port (
        DCBA: in std_logic_vector (3 downto 0);
        O: OUT std_logic_vector (9 DOWNTO 0)
        );
end Ej2;
architecture Behavioral of Ej2 is
begin
process(DCBA)
begin
case DCBA is -- checar salida negada
when "0000" => O <= "1111111110";
when "0001" => O <= "1111111101";
when "0010" => O <= "1111111011";
when "0011" => O <= "1111110111";
when "0100" => O <= "1111101111";
when "0101" => O <= "1111011111";
when "0110" => O <= "1110111111";
when "0111" => O <= "1101111111";
when "1000" => O <= "1011111111";
when "1001" => O <= "0111111111";
when others => O <= "0000000000";
end case;

end process;
end Behavioral;
```

```
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { DCBA[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { DCBA[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { DCBA[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { DCBA[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
#set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { SW[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { SW[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
#set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { SW[6] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
#set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { SW[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
#set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { SW[8] }]; #IO_L24N_T3_34 Sch=sw[8]
#set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { SW[9] }]; #IO_25_34 Sch=sw[9]
#set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
#set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { E }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
#set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { NumBin[3] }]; #IO_L24P_T3_35 Sch=sw[12]
#set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { NumBin[2] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
#set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { NumBin[1] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
#set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { NumBin[0] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { O[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { O[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { O[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { O[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { O[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { O[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { O[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { O[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { O[8] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports { O[9] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
```

Multiplexor

El multiplexor es un circuito que tomará varias entradas y, mediante un selector indica cuál de las entradas va a ser pasada a la salida o incluso si no se va a devolver ninguna salida. Su comportamiento es el siguiente:

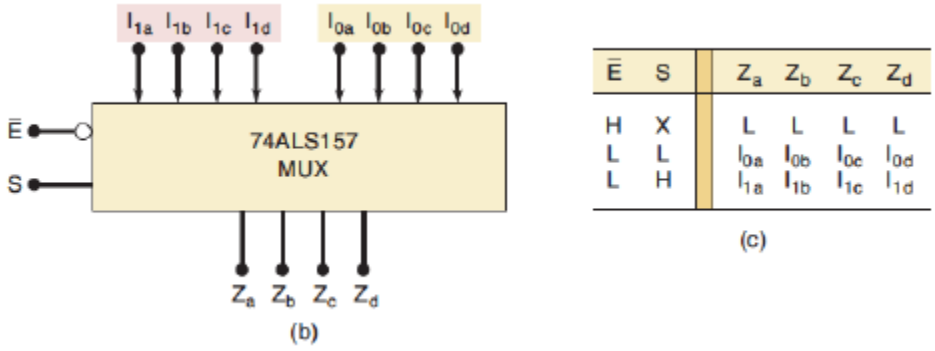


Figura 3. Comportamiento y representación gráfica de un multiplexor

Hacerlo de manera secuencial podría fusionar las dos combinaciones posibles donde la posición E del selector representa que no se va a indicar ninguna salida. Pues nomás es comprobar que se cumpla que E = L, es decir, que su negado sea H.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;
use IEEE.std_logic_unsigned.ALL;

entity Ej3 is
 Port (
        I0, I1: in std_logic_vector (3 downto 0);
        Z: out std_logic_vector (3 DOWNTO 0);
        Sel: in std_logic_vector (1 DOWNTO 0) -- donde Sel(0) = S y Sel(1) = E
        );
end Ej3;


architecture Behavioral of Ej3 is
begin

process(I0, I1, Sel)
begin

if Sel(1) = '0' then
    Z <= "0000";
elsif Sel(0) = '0' then
    Z <= I0;
elsif Sel(0) = '1' then
    Z <= I1;
end if;

end process;
end Behavioral;
```

```
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { I1[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { I1[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { I1[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { I1[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { I0[0] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { I0[1] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { I0[2] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { I0[3] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
#set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { SW[8] }]; #IO_L24N_T3_34 Sch=sw[8]
#set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { SW[9] }]; #IO_25_34 Sch=sw[9]
#set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
#set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { E }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
#set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { NumBin[3] }]; #IO_L24P_T3_35 Sch=sw[12]
#set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { NumBin[2] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { Sel[0] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { Sel[1] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { Z[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { Z[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { Z[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { Z[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
```

Demultiplexor

Un demultiplexor es un circuito que va a poseer una entrada única que, mediante un selector, se va a elegir cuál de las salidas va a tomar dicho valor mientras que las demás salidas van a poseer un valor nulo.

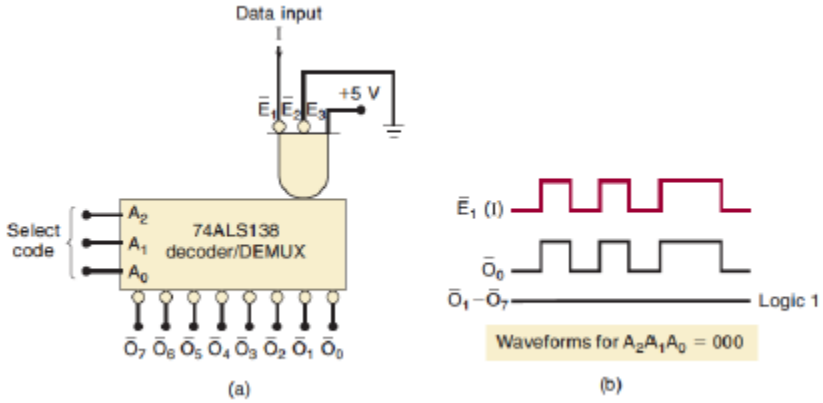


Figura 4. Representación gráfica del demultiplexor y su comportamiento.

Para desarrollar el demultiplexor mediante el uso de procesos, se ha considerado cambiar un poco la receta e introducir los ciclos, estos harán acciones repetidas veces o hasta que deje de cumplirse la condición que los mantienen activos.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;
use IEEE.std_logic_unsigned.ALL;

entity Ej4 is
 Port (
        E: in std_logic;
        O: OUT std_logic_vector (7 DOWNTO 0);
        Sel: in std_logic_vector (2 DOWNTO 0)
        );
end Ej4;


architecture Behavioral of Ej4 is

begin
process(Sel, E) begin
for i in 0 to 7 loop
    if Sel = "000" and i = 0 then
        O(i) <= E;
    elsif Sel = "001" and i = 1 then
        O(i) <= E;
    elsif Sel = "010" and i = 2 then
        O(i) <= E;
    elsif Sel = "011" and i = 3 then
        O(i) <= E;
    elsif Sel = "100" and i = 4 then
        O(i) <= E;
    elsif Sel = "101" and i = 5 then
        O(i) <= E;
    elsif Sel = "110" and i = 6 then
        O(i) <= E;
    elsif Sel = "111" and i = 7 then
        O(i) <= E;
    else
        O(i) <= '0';
    end if;
end loop;
end process;
end Behavioral;
```

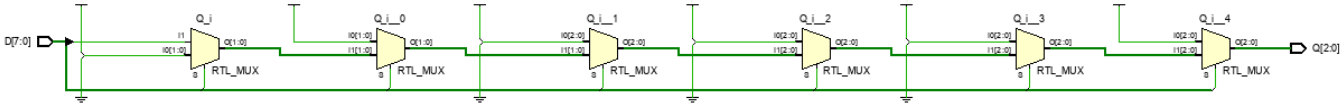
```
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { E }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
#set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { NumBin[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
#set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { NumBin[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
#set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { NumBin[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
#set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { SW[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { SW[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
#set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { SW[6] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
#set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { SW[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
#set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { SW[8] }]; #IO_L24N_T3_34 Sch=sw[8]
#set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { SW[9] }]; #IO_25_34 Sch=sw[9]
#set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
#set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { E }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
#set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { NumBin[3] }]; #IO_L24P_T3_35 Sch=sw[12]
set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { Sel[0] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { Sel[1] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { Sel[2] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { O[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { O[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { O[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { O[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { O[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { O[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { O[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { O[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
```

Véase que el ciclo va a tener una variable i que va a tomar los valores del 0 al 7, correspondientes a los índices del vector de salida, en cada iteración el if va a comprobar si la posición en la que se encuentra corresponde al número binario que representa el selector, caso contrario le va a asignar el valor 0 a dicha posición del vector.

La única situación en que se hagan verdaderas las condiciones del if o los else, es si estoy en la posición indicada. Además, no significa que necesariamente vaya a devolver un valor verdadero, pues en caso de que la entrada esté apagada, la salida lo estará, pero no por no cumplirse su respectiva condición, sino porque sí se está cumpliendo, pero el valor que le corresponde es 0.

Resultados:

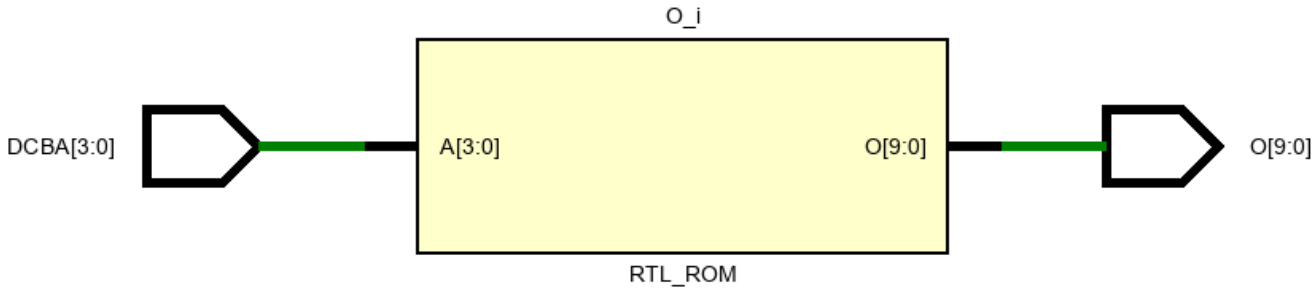
Los resultados obtenidos son los esperados. El **codificador** ignora el estado de las entradas menores a la entrada activada más grande, y devuelve el valor binario correspondiente. Es decir, si presento "00100000", "00101011" o "00110010" el circuito va a tomarlo como un "001XXXXX" y va a devolver "110". Y en caso de apagarlos todos va a apagar la salida.



El **decodificador** toma una entrada de tres bits y devuelve el valor de su salida correspondiente, al solo haber diez salidas, que son del 0 al 9, se van a establecer todas las luces en LOW como indicio de "emergencia" de la siguiente manera:

D	C	B	A	Salida
0	0	0	0	1111111110
0	0	0	1	1111111101
0	0	1	0	1111111011
0	0	1	1	1111110111
0	1	0	0	1111101111
0	1	0	1	1111011111
0	1	1	0	1110111111
0	1	1	1	1101111111
1	0	0	0	10111111111
1	0	0	1	01111111111
1	0	1	0	00000000000
1	0	1	1	00000000000
1	1	0	0	00000000000
1	1	0	1	00000000000
1	1	1	0	00000000000

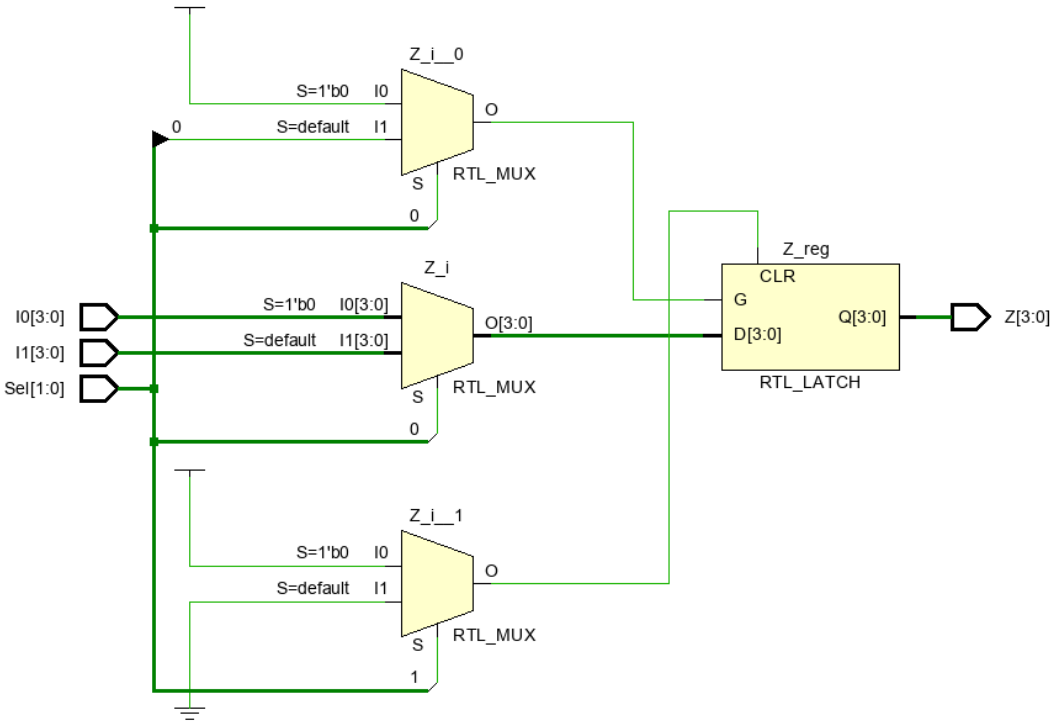
1	1	1	1	00000000000



El **multiplexor** devuelve los siguientes valores:

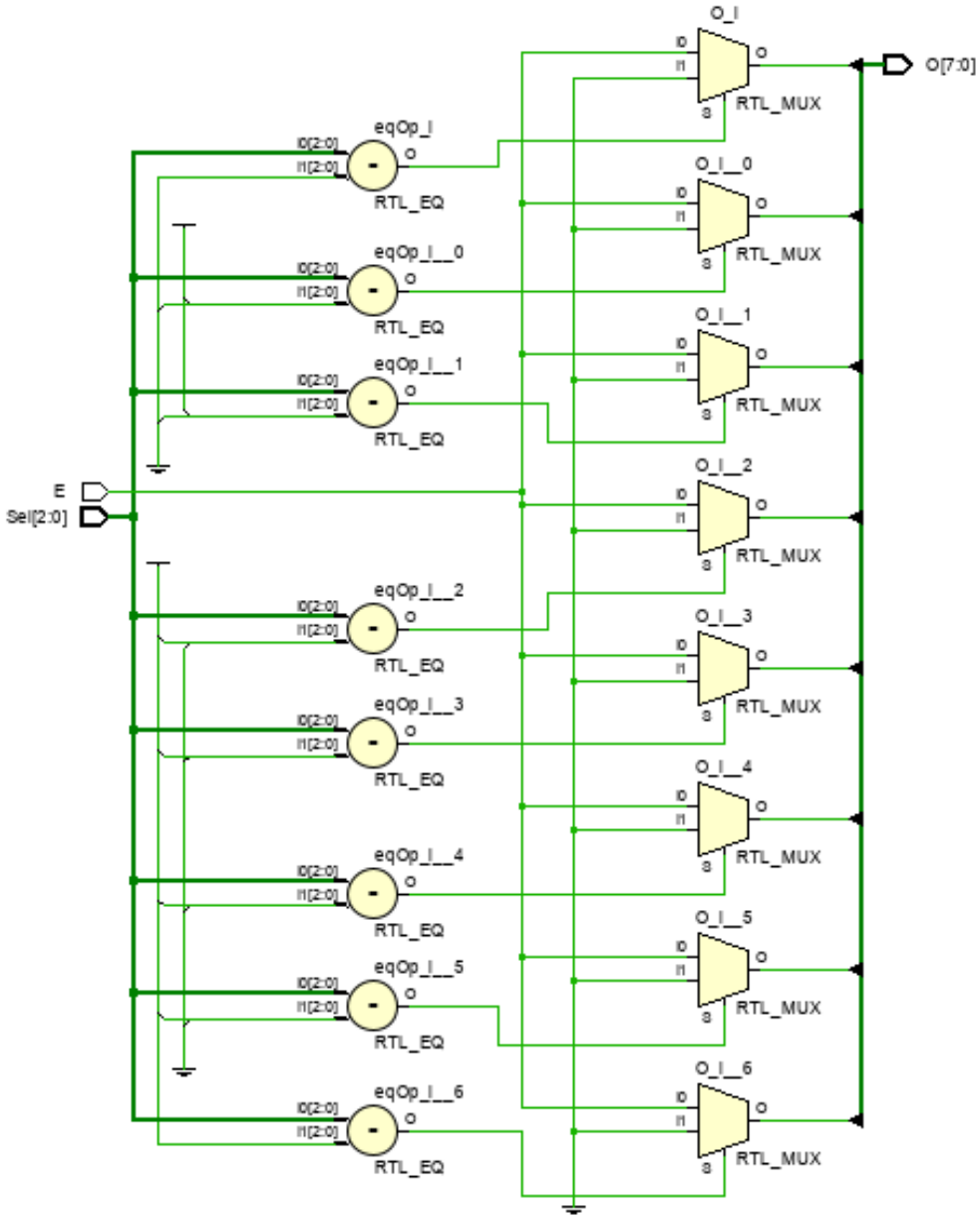
I0	I1	ES	Not(E)S	Z
XXXX	YYYY	00	10	0000
XXXX	YYYY	01	11	0000
XXXX	YYYY	10	00	XXXX
XXXX	YYYY	11	01	YYYY

Donde XXXX y YYYY, puede ser cualquier combinación de valores.



El **demultiplexor** le entrega le entrega a la salida que se le solicite el valor de la entrada que este posee, es decir, sea X el valor 0 o 1, la salida cumple con lo siguiente:

E	Sel	O
X	000	0000000X
X	001	000000X0
X	010	00000X00
X	011	0000X000
X	100	000X0000
X	101	00X00000
X	110	0X000000
X	111	X0000000



Análisis:
En el presente laboratorio se han implementado los cuatro códigos anteriormente presentados, todos conteniendo procesos, buses (también llamados vectores), ciclos, comparaciones lógicas y condicionales, trabajando de manera secuencial. Siendo esto probado mediante la implementación de codificadores, decodificadores, multiplexores y demultiplexores.

Luego de la presente, pudimos llegar a familiarizarnos con los procesos y notar que estos trabajan de manera secuencia, comportamiento muy útil a la hora de crear muchos comportamientos complejos, además, mediante estos se pudo simplificar el comportamiento de ciertas funciones al poder ignorar acciones posteriores gracias al resultado de una de mayor prioridad.

Por lo general los diseños presentados fueron optimizados aprovechando el uso de los procesos, sin embargo, los dos últimos diseños, el multiplexor y demultiplexor, pudieron haber funcionado de manera más simple si no se usase una lógica secuencial, pues bastaría con un WITH-WHEN o un WHEN-ELSE.