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.

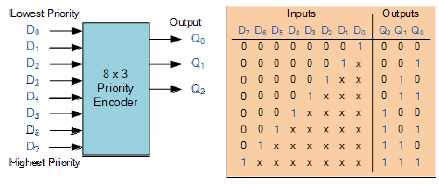
# Objetivos generales:

* 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.

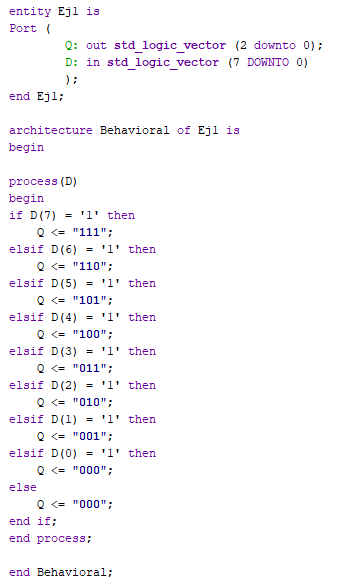
# Procedimiento:

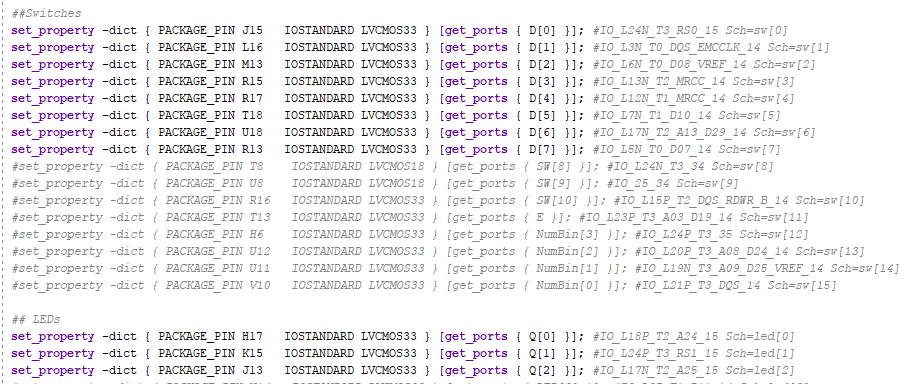
## Codificador

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:

  
**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.

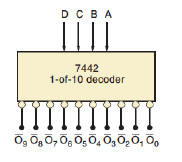




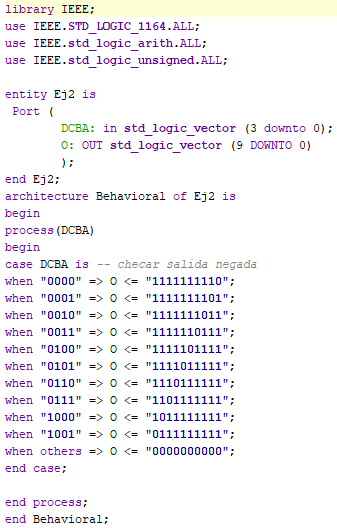
Nota: es cierto que la última comprobación y el else pueden ser fusionados, sin embargo, no se hace para distinguir que el 000 correspondiente a la entrada 0 es por ser la codificación de la entrada, mientras que el 000 del else es por no quedar más opciones.

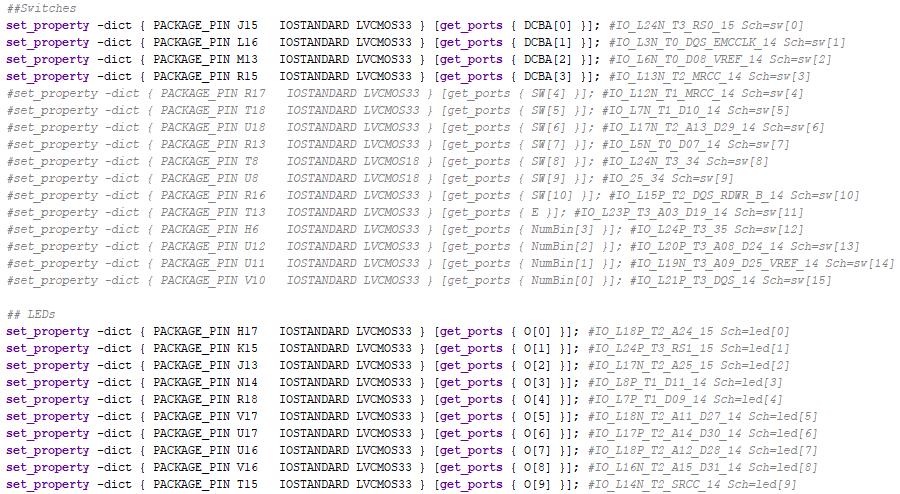
## Decodificador

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.

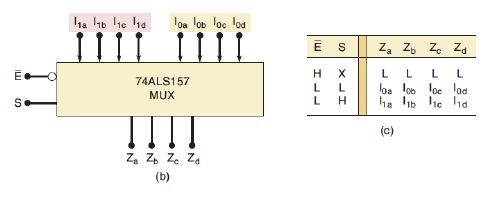
Para el desarrollo del codificador se introduce el comportamiento de case, es parecido a un if, solo que la comprobación que hace no es una condición como tal, sino que confirma si la variable evaluada posee dicho valor.



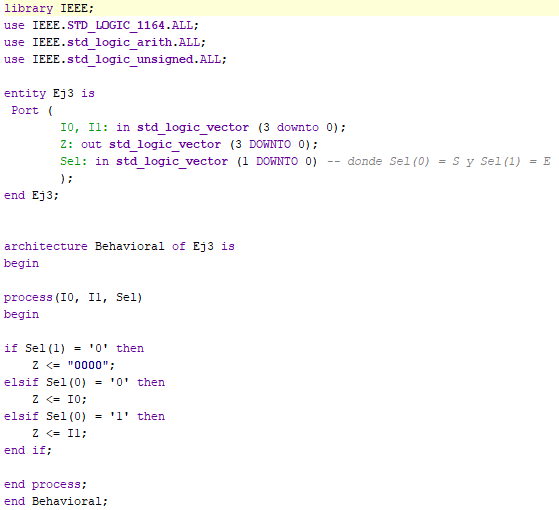
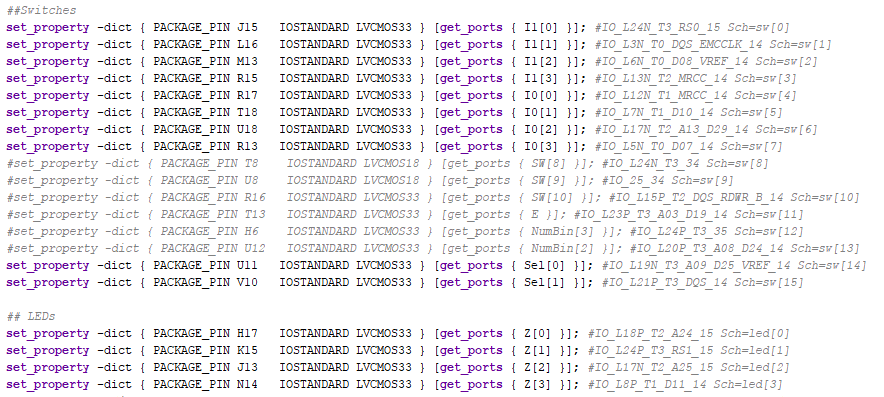


## Multiplexor

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:

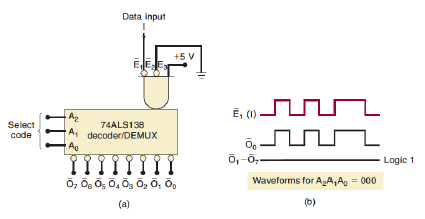
  
**Figura 3.** Comportamiento y representación gráfica de un multiplexor

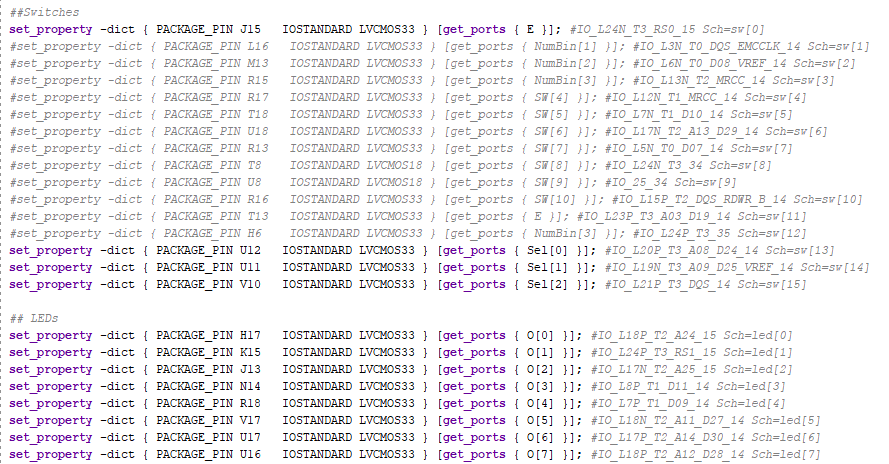
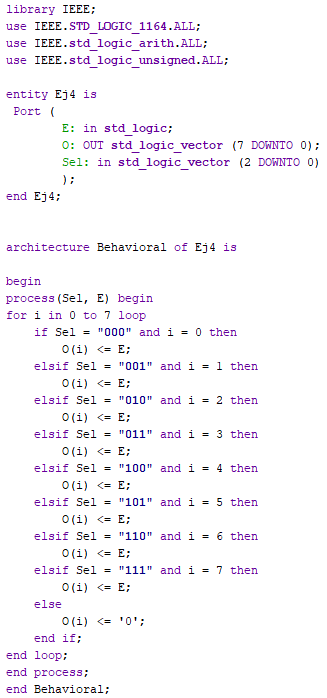
Hacerlo de manera secuencial podría fusionar las dos combinaciones posibles donde la posición E del selector representa que no se va a indicar ninguna salida. Pues nomás es comprobar que se cumpla que E = L, es decir, que su negado sea H.

## 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.

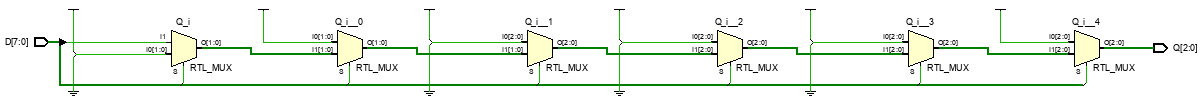
Para desarrollar el demultiplexor mediante el uso de procesos, se ha considerado cambiar un poco la receta e introducir los ciclos, estos harán acciones repetidas veces o hasta que deje de cumplirse la condición que los mantienen activos.

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.

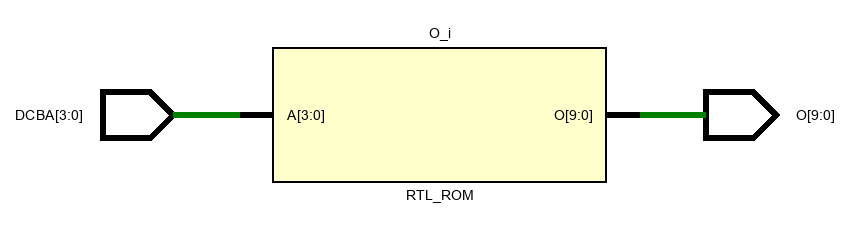
# Resultados:

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:

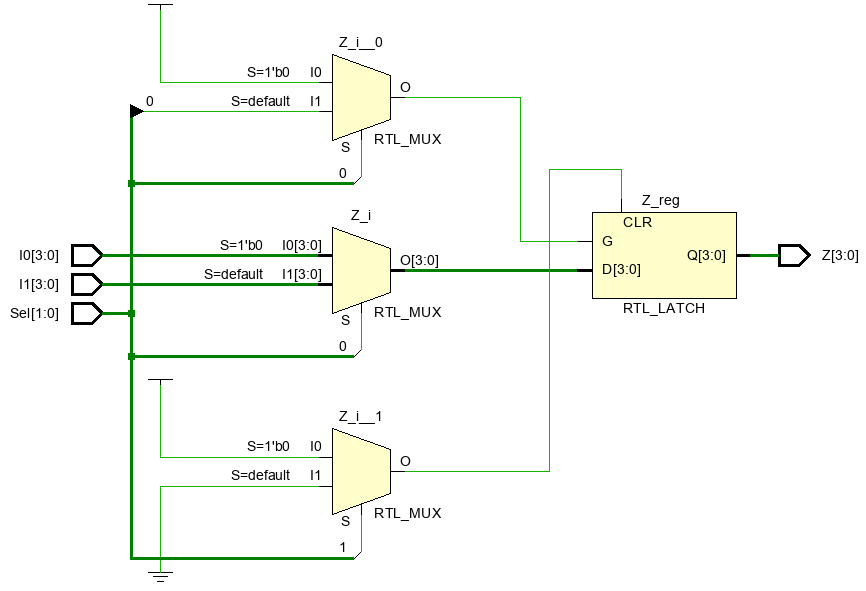
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **D** | **C** | **B** | **A** | **Salida** |
| 0 | 0 | 0 | 0 | 1111111110 |
| 0 | 0 | 0 | 1 | 1111111101 |
| 0 | 0 | 1 | 0 | 1111111011 |
| 0 | 0 | 1 | 1 | 1111110111 |
| 0 | 1 | 0 | 0 | 1111101111 |
| 0 | 1 | 0 | 1 | 1111011111 |
| 0 | 1 | 1 | 0 | 1110111111 |
| 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:

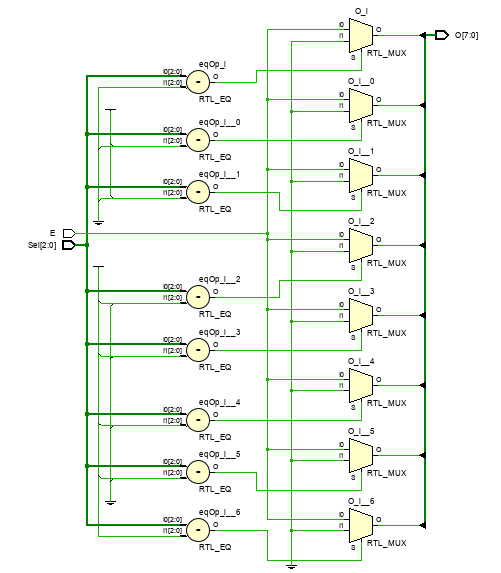
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **I0** | **I1** | **ES** | **Not(E)S** | **Z** |
| XXXX | YYYY | 00 | 10 | 0000 |
| XXXX | YYYY | 01 | 11 | 0000 |
| XXXX | YYYY | 10 | 00 | XXXX |
| XXXX | YYYY | 11 | 01 | YYYY |

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.