Práctica Nº 5. Multiplexores

1. Datos de la práctica

Carrera	INGENIERÍA ELECTRÓNICA		
Semestre		Grupo	
Tipo de Práctica	☐ Laboratorio ☐ Simulación	Fecha	
Asignatura	Electrónica Digital I		
Unidad Temática			
Nº Alumnos por práctica	2	Nº Alumnos por reporte	2
Nombre del Profesor			
Nombre(s) de Alumno(s)	1. 2.		
Tiempo estimado		Vo. Bo. Profesor	
Comentarios			

2. Objetivo

Comprender el concepto de multiplexor mediante la implementación de un diseño en VHDL que intercambie entre las distintas operaciones aritméticas que se han elaborado en laboratorios anteriores.

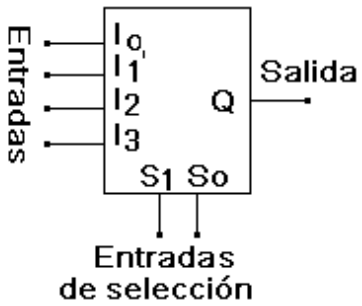
3. Medios a utilizar

Por cada práctica y por cada puesto de laboratorio, los materiales a utilizar son:

Cantidad	Descripción
1	Computadora
1	Tarjeta de desarrollo Basys2 Digilent
1	Software Xilinx ISE® Webpackv14.7

4. Introducción

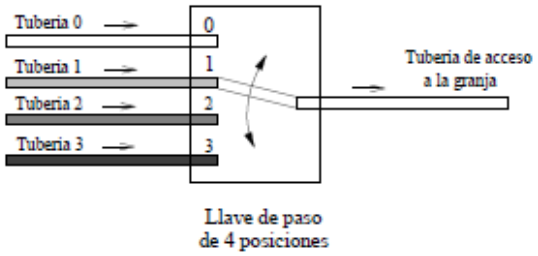
Un multiplexor es un circuito lógico que acepta varias entradas de datos digitales y solo uno de ellos, el que hayamos seleccionado, es el que aparece en la salida. Es decir, que es un circuito que nos permite seleccionar que datos pasan a través de dicho componente. Una aplicación común en computadoras es el multiplexor entre registros del microprocesador y su unidad aritmético lógica. En el interior de los microprocesadores se pueden encontrar multiplexores de 8 bits, que tiene varias entradas de datos de 8 bits. Sin embargo, se puede trabajar con multiplexores que tengan 4 bits, 2 bits o hasta 1 bit por cada entrada.



Dentro de un multiplexor hay que destacar 3 tipos de señales: los datos de entrada, las entradas de selección y la salida. En esta práctica de laboratorio se describirá el concepto básico de un multiplexor descrito en VHDL.

5. Actividades previas

En una granja hay cuatro tuberías por las que circulan distintos fluidos. Dos transportan agua potable a diferentes partes de la casa y las siguientes dos se utilizan para regar los cultivos. Existe una llave de paso con cuatro posiciones y dos selectores para dar paso a la tubería 0, 1, 2 ó 3. Diseñe un circuito combinacional lo más simplificado posible de tal manera que seleccione una tubería a la vez.



6. Desarrollo de la práctica

Multiplexor de 4 Entradas de Datos y 2 Entradas de Selección

1. Crear un nuevo proyecto en ISE® llamado Mux.
2. Crear el módulo **Mux4** descrito en el siguiente programa.

```
entity Mux4 is
    Port ( c : in  STD_LOGIC_VECTOR (3 downto 0);
           s : in  STD_LOGIC_VECTOR (1 downto 0);
           y : out  STD_LOGIC);
end Mux4;

architecture Behavioral of Mux4 is

begin
     p1: process(c,s)
     begin
          case s is
               when "00" => y <= c(0);
               when "01" => y <= c(1);
               when "10" => y <= c(2);
               when "11" => y <= c(3);
               when others => y <= c(0);
          end case;
     end process;
end Behavioral;
```

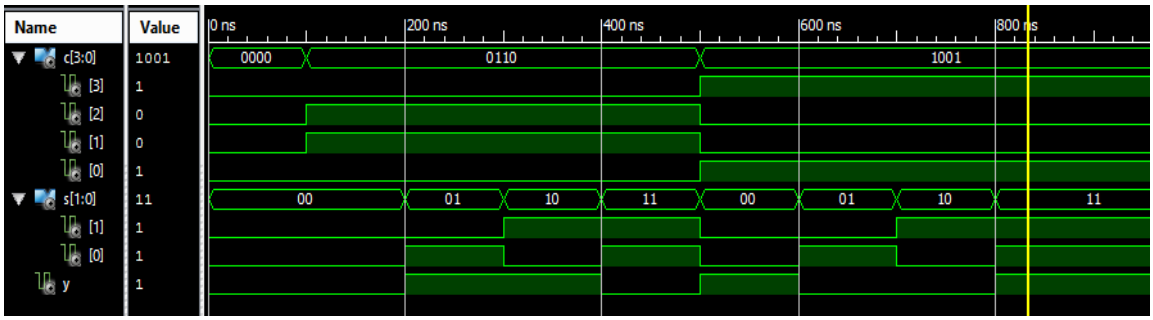
Simulación del Multiplexor

1. Verificamos la sintaxis, agregamos una nueva fuente **"New Source"** y escogemos el tipo de archivo **"VHDL Test Bench"** el cual llamaremos **"Mux_tb"**.
2. Recordemos pasarnos al modo Simulación y en el archivo VHDL Test Bench comentar todas las líneas que hacen referencia a clock.
3. Añadir las siguientes líneas después del cometario `--Insert stimulus here` y antes de la sentencia `end`.

```
c <= "0110";
s <= "00";
wait for 100 ns;
c <= "0110";
s <= "01";
wait for 100 ns;
c <= "0110";
s <= "10";
wait for 100 ns;
c <= "0110";
s <= "11";
wait for 100 ns;
```

```
c <= "1001";
s <= "00";
wait for 100 ns;
c <= "1001";
s <= "01";
wait for 100 ns;
c <= "1001";
s <= "10";
wait for 100 ns;
c <= "1001";
s <= "11";
wait for 100 ns;
```

4. Una vez agregado el estímulo deberíamos obtener las formas de onda de la siguiente imagen.



Expansión de un Multiplexor

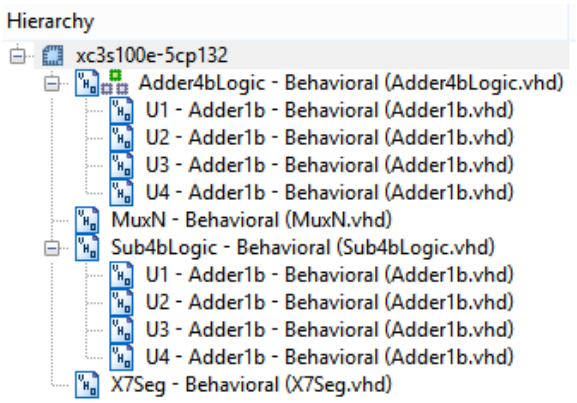
Para expandir un multiplexor con programación en VHDL utilizaremos la sentencia **`generic`** que define el ancho del bus (constante) del multiplexor. En este caso hemos descrito que las entradas **a** y **b** tienen 3 bits cada uno y la variable "salida" también es de 3 bits.

1. Crear un nuevo proyecto en ISE® llamado **MuxExpansion**.
2. Crear el módulo **MuxN** descrito en el siguiente programa.

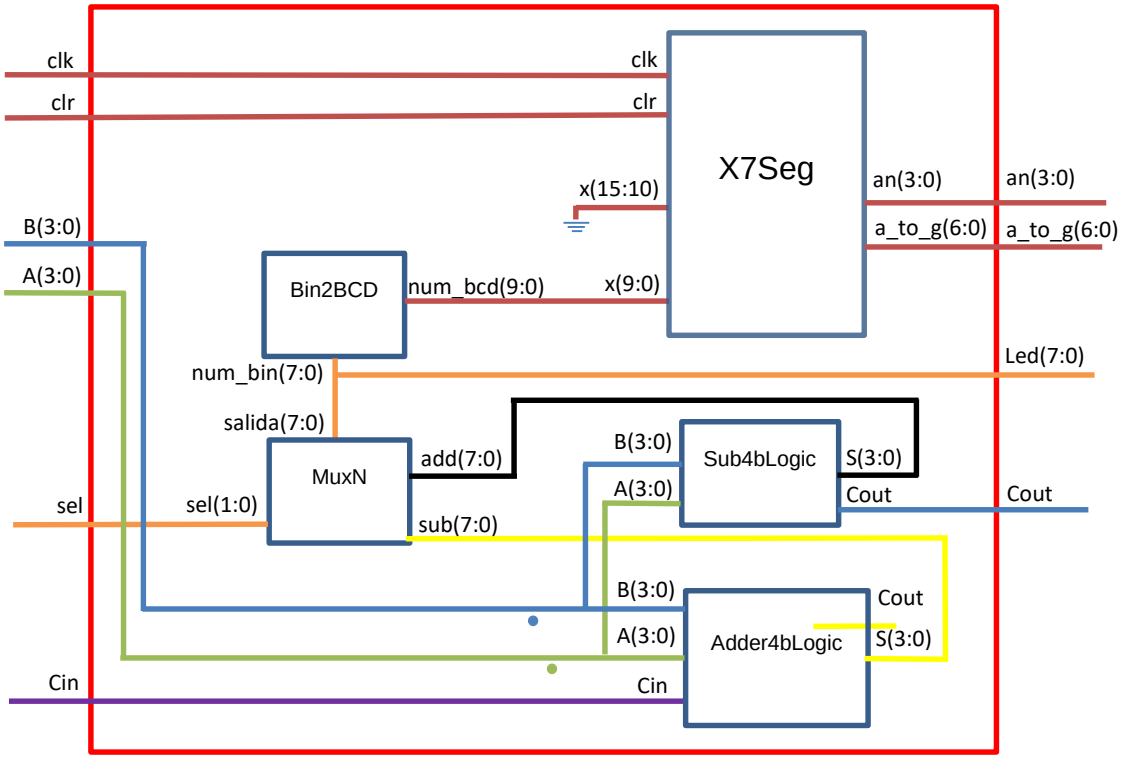
```
entity MuxN is
generic (N: integer :=8; M: integer :=2);
Port ( add : in  STD_LOGIC_VECTOR (N-1 downto 0);--Suma
       sub : in  STD_LOGIC_VECTOR (N-1 downto 0);--Resta
       mul : in  STD_LOGIC_VECTOR (N-1 downto 0);--Multiplicacion
       div : in  STD_LOGIC_VECTOR (N-1 downto 0);--Division
       sel : in  STD_LOGIC_VECTOR(M-1 downto 0);
       salida : out  STD_LOGIC_VECTOR (N-1 downto 0));
end MuxN;

architecture Behavioral of MuxN is
begin
p1: process (add,sub,mul,div,sel)
     begin
      case (to_integer(unsigned(sel))) is
                when 0 => salida <= add;
                when 1 => salida <= sub;
                when 2 => salida <= mul;
                when 3 => salida <= div;
                when others => salida <= add;
           end case;
     end process;
end Behavioral;
```

3. Agregar los archivos .VHD del Sumador Completo de 4 bits (**Adder4bLogic.vhd y Adder1b.vhd**), del Restador Completo de 4 bits (**Sub4bLogic.vhd**) y del Decodificador de 7 Segmentos (**X7Seg.vhd**) con la opción **Add Source of Copy**.

4. Verificar que los archivos agregados queden de la siguiente manera:

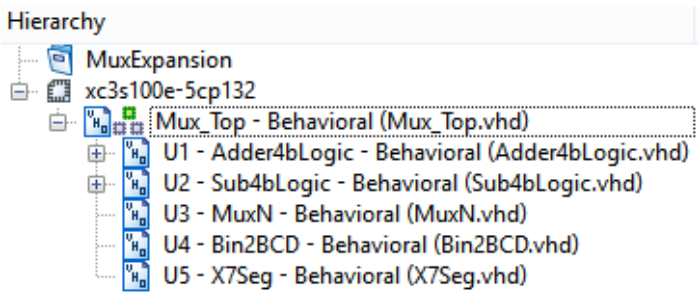


5. Cree un archivo Top llamado **Mux_Top** que tenga las siguientes especificaciones y que cumpla con el diagrama de bloques de la Figura.



Mux_Top

6. Asegúrese que la jerarquía quede de la siguiente manera:



7. Realice la Simulación del ejercicio anterior dándoles valores a las entradas A, B, Cin, Sel y clear en los estímulos del Archivo **Test Bench**.
8. Implemente todo el modulo Top del Proyecto para visualizarlo en la Basys 2.

7. Actividades propuestas

Práctica

En la práctica de laboratorio se implementó un multiplexor que direccionará las salidas del sumador completo y el restador completo hacia el modulo del decodificador de 7 segmentos. Modifique el multiplexor para que acepte ahora las cuatro operaciones aritméticas que se han elaborado en reportes de laboratorios anteriores.

Ahora el multiplexor, multiplexará la operación de suma, resta, multiplicación y división hacia el módulo Bin2BCD para que luego pueda ser visualizado en los displays de 7 segmentos.

Nota: Recuerde que para modificar el **MuxN** solo tiene que agregar más entradas para los proyectos (**Mult4bLogic y Div4bLogic**) y aumentar la cantidad de bits de **sel**.

El reporte debe incluir:

- Descripción del circuito en VHDL (Código).
- Verificación del programa en ISE. Utilice las capturas de pantalla (tecla: PRTSC) para explicar el procedimiento.
- Comprobación del funcionamiento mediante simulación en ISim.

Adjunte sus resultados y conclusiones de este ejercicio.