Práctica Nº 3. Circuitos Aritméticos

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

Verificar el funcionamiento de circuitos aritméticos como sumadores y restadores, comparando diseños jerárquicos en el lenguaje VHDL.

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

En esta práctica de laboratorio implementaremos un sumador binario completo de 4 bits con su respectivo acarreo. El diseño se realizará de dos maneras:

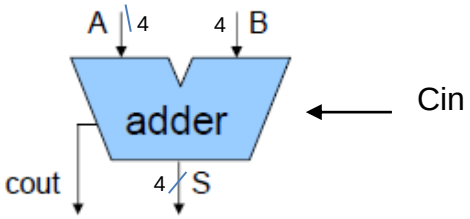
El primero es de tipo funcional porque expone la forma en que trabaja el sistema, es decir se considera la relación que hay entre las entradas y las salidas del circuito, sin importar como este organizado en su interior. El segundo es el nivel de flujo de datos empleando ecuaciones lógicas para el diseño operacional del circuito, aquí se toman en cuenta el diseño lógico mediante compuertas.

5. Actividades previas

Investigar la funcionalidad de la librería **IEEE.STD_LOGIC_unsigned.ALL** y donde será aplicado en nuestro diseño.

6. Desarrollo de la práctica

1. Crear un nuevo proyecto en ISE® llamado Adder4b.

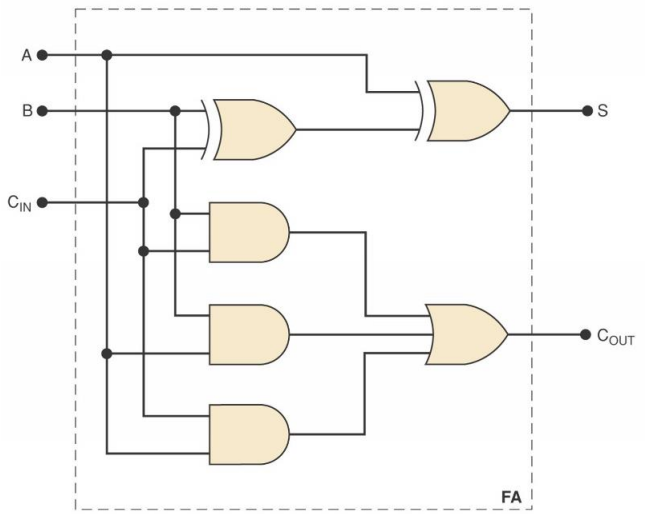


2. Crear el archivo Adder4b.vhd descrito en el siguiente programa.

```
entity Adder4b is
     port(
          A    :    in std_logic_vector(3 downto 0);
          B    :    in std_logic_vector(3 downto 0);
          Cin  :    in std_logic;
          Cout :    out std_logic;
          S    :    out std_logic_vector(3 downto 0)
     );
end Adder4b;

architecture Behavioral of Adder4b is

begin
     process(A,B,Cin)
          variable temp :     std_logic_vector(4 downto 0);
     begin
          temp := ('0' & A) + ('0' & B) + (“0000” & Cin);
          S <= temp(3 downto 0);
          Cout <= temp(4);
     end process;
end Behavioral;
```

3. Simular e implementar el código anterior.

Existen otras formas de describir el comportamiento de un circuito, por ejemplo, usando el modelado de flujo de datos que describe el camino que los datos siguen al ser transferidos por las operaciones efectuadas. En este caso este sumador de 4 bits se puede representar usando operaciones lógicas, es decir hacer el circuito con compuertas. A continuación, se expresa el circuito lógico y el código del sumador de 1 bit para poder instanciarlo y generar uno de mayor capacidad.

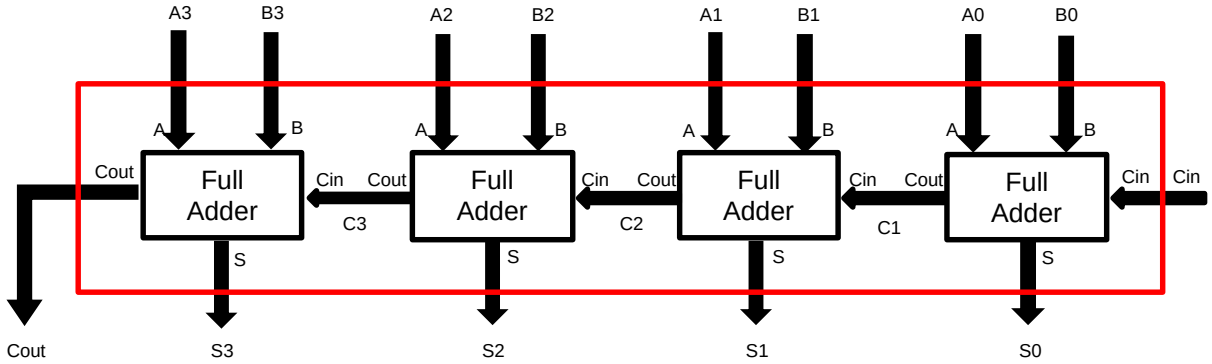


4. Cree un nuevo proyecto llamado **Adder4bLogic**, generando un archivo .vhd llamado **Adder1b** con el código que se muestra a continuación:

```
entity Adder1b is
port(
        A       :       in std_logic;
        B       :       in std_logic;
        Cin     :       in std_logic;
        Cout    :       out std_logic;
        S       :       out std_logic
);
end Adder1b;

architecture Behavioral of Adder1b is

begin

Cout <= (B and Cin) or (A and Cin) or (A and B);
S <= A xor (B xor Cin);
end Behavioral;
```

Creación de Sumador Completo de 4 Bits

5. Cree un Nuevo Archivo llamado **Adder4bLogic,** el cual será el archivo Top del archivo **Adder1b.vhd** del ejercicio anterior, para después instanciarlo según el diseño de la Figura.



6. El siguiente fragmento de código muestra cómo debe instanciarse el Adder1b, recordando que sebe de poner una sola sección de componente.

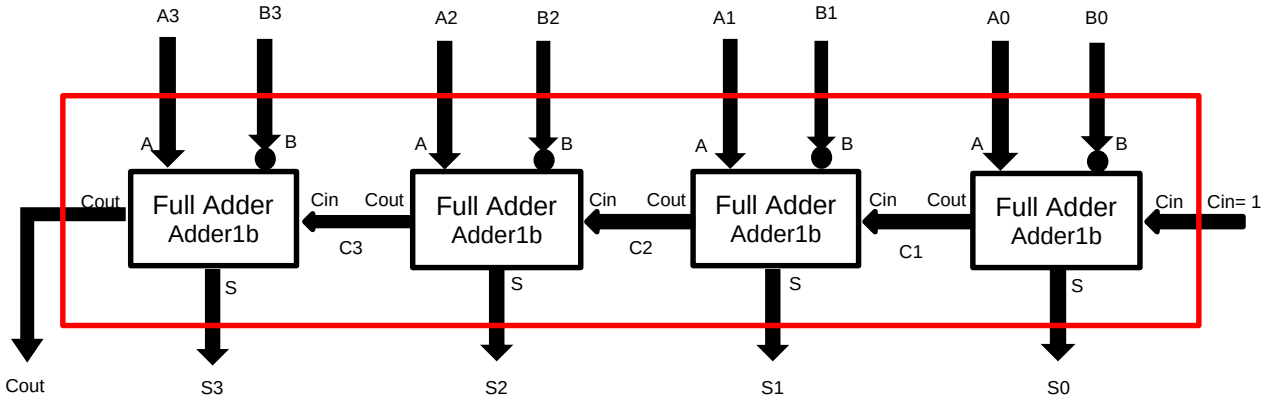
```
U1: Adder1b PORT MAP(
        A => A(0),
        B => B(0),
        Cin => Cin,
        Cout => c1,
        S => S(0)
);
U2: Adder1b PORT MAP(
        A => A(1),
        B => B(1),
        Cin => c1,
        Cout => c2,
        S => S(1)
);
U3: Adder1b PORT MAP(
        A => A(2),
        B => B(2),
        Cin => c2,
        Cout => c3,
        S => S(2)
);
U4: Adder1b PORT MAP(
        A => A(3),
        B => B(3),
        Cin => c3,
        Cout => Cout,
        S => S(3)
);
```

Nota: Recuerde solo poner una vez un componente para el archivo Adder1b y crear señales intermedias para poder hacer la instancia de los bloques.

7. Crear la simulación para el sumador completo de 4 Bits.
8. Implementar este diseño con los nets correspondientes en la Basys2.

Creación de Restador Completo de 4 Bits

9. Cree un nuevo Proyecto llamado **Sub4bLogic**.
10. Agregue los archivos **Adder1b.vhd** y **.ucf** del proyecto anterior del sumador completo de 4 bits. Esto lo hacen con la opción de **Add Copy of Source**.
11. Cree un nuevo archivo Top llamado **Sub4bLogic,** el cual se instanciará según el siguiente diagrama de bloques.



12. Realice la Simulación e impleméntelo en el Kit Basys 2.

7. Actividades propuestas

Práctica

1. Diseñe un circuito multiplicador de 4 bits **(Mult4bLogic)** utilizando el proyecto del Sumador de 4 Bits **(Adder4bLogic)**.
2. Diseñe un circuito divisor de 4 bits **(Div4bLogic)**.

El reporte debe incluir:

- Diagrama de Bloques generado por el ISE®
- Comprobación del funcionamiento mediante simulación en ISim. Utilice las capturas de pantalla (tecla: PRTSC) para explicar el procedimiento.

Actividades de aprendizaje

1. ¿Podría realizarse un circuito sumador/restador en un mismo diseño, de tal forma que se pueda seleccionar que la operación deseada sea suma o resta? ¿Y en VHDL? Proponga una solución a este reto con un algoritmo que satisfaga esta necesidad.