Circuitos Lógicos: SDI-11322
Práctica 3: Circuito Aritmético

Departamento Académico de Sistemas Digitales
ITAM

Otoño de 2018

1. Objetivos

Que el alumno:

- Implemente circuitos utilizando VHDL
- Implemente un circuito aritmético
- Diseñe y realize simulaciones de los circuitos implementados

2. Problema

Se requiere implementar un circuito que de entrada reciba hasta dos números de 16 bits y una señal que seleccione la operación aritmética requerida, entregándola en la salida. El resultado se debe desplegar utilizando cuatro displays de siete segmentos.

3. Esbozo de solución

3.1. Circuito de entrada

Para ésta práctica se requieren tres entradas:

- Dip switch 1 - Entrada A (número de 8 bits)
- Dip switch 2 - Entrada B (número de 8 bits)
- Dip switch 3 - Entrada de selección (3 bits)

3.2. Circuito Aritmético

En la tabla 1 se muestran las entradas posibles para el circuito. Utilizando la ecuación $G = A|Y|C_{in}$, llenar la tabla. ¿Cuáles son las operaciones que se están implementando con la tabla?

Select		Y	$G = (A\|Y\|C_{in})$	
S_1	S_0	Y	$C_{in} = 0$	$C_{in} = 1$
0	0	all 0s	$G =$	$G =$
0	1	B	$G =$	$G =$
1	0	$\overline{B}$	$G =$	$G =$
1	1	all 1s	$G =$	$G =$

Cuadro 1: Tabla de un Circuito Aritmético

3.3. Interconexiones de componentes

Lo que en prácticas pasadas se realizaba en el archivo de bloques *.bdf*, también se puede hacer en VHDL. Se deben incluir los archivos *.vhd* usados en la práctica 2. Crear un nuevo archivo VHDL, en éste ejemplo se utilizó *pract3.vhd*. En la Fig. 1 se muestra el diagrama del circuito que se va a interconectar.

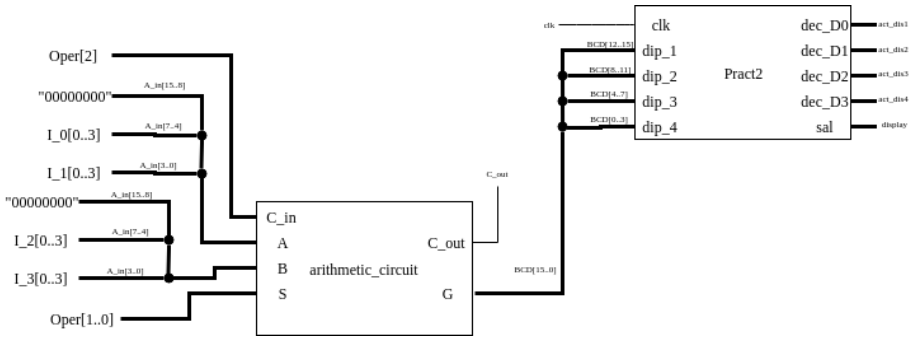

Figura 1: Esquema del circuito a implementar

Primero se incluyen las librerias necesarias.

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  LIBRARY work;
```

Después se asignan las entradas y salidas del proyecto. Éstas son a las que se les asignarán pines con el *pin planner*.

```
5  ENTITY pract3 IS
6          PORT
7          (
8                  clk      :  IN   STD_LOGIC;
9                  I_0      :  IN   STD_LOGIC_VECTOR(3 DOWNTO 0);
10                 I_1      :  IN   STD_LOGIC_VECTOR(3 DOWNTO 0);
11                 I_2      :  IN   STD_LOGIC_VECTOR(3 DOWNTO 0);
12                 I_3      :  IN   STD_LOGIC_VECTOR(3 DOWNTO 0);
```

```
13                  Oper     :  IN   STD_LOGIC_VECTOR(2 DOWNTO 0);
14                  act_dis1:  OUT  STD_LOGIC;
15                  act_dis2:  OUT  STD_LOGIC;
16                  act_dis4:  OUT  STD_LOGIC;
17                  act_dis3:  OUT  STD_LOGIC;
18                  C_out    :  OUT  STD_LOGIC;
19                  display  :  OUT  STD_LOGIC_VECTOR(6 DOWNTO 0);
20                  oper_led:  OUT  STD_LOGIC_VECTOR(2 DOWNTO 0)
21          );
22  END pract3;
```

Se define el nombre de la arquitectura

```
24  ARCHITECTURE bdf_type OF pract3 IS
```

Ahora se van a incluir todos los componentes que se utilizarán. El primer componente corresponde a la práctica 2; los nombres de las entradas y salidas deben coincidir con los utilizados en la práctica 2.

```
25  COMPONENT pract2
26      PORT
27      (
28          clk     : IN STD_LOGIC;
29          dip_1   : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
30          dip_2   : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
31          dip_3   : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
32          dip_4   : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
33          dec_D0  : OUT STD_LOGIC;
34          dec_D1  : OUT STD_LOGIC;
35          dec_D2  : OUT STD_LOGIC;
36          dec_D3  : OUT STD_LOGIC;
37          sal     : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
38      );
39  END COMPONENT;
```

El siguiente componente es el circuito aritmético.

```
41  COMPONENT arithmetic_circuit
42      PORT
43      (
44          C_in   : IN STD_LOGIC;
45          A      : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
46          B      : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
47          S      : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
48          C_out  : OUT STD_LOGIC;
49          G      : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
50      );
51  END COMPONENT;
```

Ahora se declaran señales para conectar los distintos componentes.

```
68  SIGNAL  A_in :  STD_LOGIC_VECTOR(15 DOWNTO 0);
69  SIGNAL  B_in :  STD_LOGIC_VECTOR(15 DOWNTO 0);
70  SIGNAL  G :  STD_LOGIC_VECTOR(15 DOWNTO 0);
71
72  BEGIN
73  --Aqui va mas codigo
120 END bdf_type;
```

A continuación se van a mapear las entradas y salidas de los componentes de acuerdo a las conexiones deseadas. Primero para el componente pract2 creado en la práctica pasada.

```
74  oper_led <= Oper;
75
76  b2v_inst_pract2 : pract2
77  PORT MAP
78      (
79          clk => clk,
80          dip_1 => G(15 DOWNTO 12),
81          dip_2 => G(11 DOWNTO 8),
82          dip_3 => G(7 DOWNTO 4),
83          dip_4 => G(3 DOWNTO 0),
84          dec_D0 => act_dis1,
85          dec_D1 => act_dis2,
86          dec_D2 => act_dis3,
87          dec_D3 => act_dis4,
88          sal => display
89      );
```

A continuación el circuito aritmético

```
91  b2v_instac : arithmetic_circuit
92  PORT MAP
93      (
94          C_in => Oper(2),
95          A => A_in,
96          B => B_in,
97          S => Oper(1 DOWNTO 0),
98          C_out => C_out,
99          G => G
100     );
```

Ahora se asignan las entradas I_X del circuito a distintos bus, que se utilizaron de entradas de algunos componentes.

```
110 A_in(7 DOWNTO 4) <= I_0;
111 A_in(3 DOWNTO 0) <= I_1;
112 B_in(7 DOWNTO 4) <= I_2;
113 B_in(3 DOWNTO 0) <= I_3;
```

```
114
115  A_in(15 DOWNTO 8) <= x"00";
116  B_in(15 DOWNTO 8) <= x"00";
```

4. Validación

Para la validación, conviene usar ModelSim para hacer simulaciones con distintos números A y B además de usar las distintas entradas posibles. En éste caso, conviene solamente simular el circuito aritmético, no todo el circuito implementado debido a que incluye circuitos ya simulados (práctica 2). Una vez que se comprobó que la simulación funciona correctamente, se pueden asignar los pines (los mismos que se utilizaron en la práctica 2 además de incluir los *dip switch* de la tarjeta) y programar la FPGA para comprobar su funcionamiento con los displays utilizando el circuito alambrado en la práctica pasada.