Informe de laboratorio N°2
Semisumador

Fernando Guiraud

Universidad tecnológica de Panamá
Microprocesadores
1EE141
fjguiraud@gmail.com

I. INTRODUCCIÓN

En esta experiencia de laboratorio trabajaremos con el entorno de VHDL de Xilin ISE, para generar un sumador de dígitos binarios. El programa de descripción de hardware debe ser capaz de generar el valor de la suma y adicional un digito de carry.

La suma de dos dígitos binarios (PLUS) es similar a la suma de dos números decimales, pero teniendo en cuenta que la salida también es un número binario. Esto es importante cuando sumo, por ejemplo, 1 y 1, ya que para codificar el resultado (2 en decimal) necesito dos bits (10). En este caso, el bit menos significativo lo llamaremos suma, mientras que el bit más significativo lo llamaremos acarreo (carry en inglés). En total, existen 4 posibilidades de sumar dos números binarios de 1 bit:

	0	0	1	1
PLUS	0	1	0	1
	—	—	—	—
	0	1	1	10

Figura 1: Valores esperados de salida del semisumador

El circuito que implementa esta función se denomina semi-sumador (HA o half–adder). Por lo tanto, un HA es el circuito que realiza la suma de dos bits. Como es obvio, precisa dos entradas (que vamos a llamar A y B) y dos salidas: la suma propiamente dicha (S o Σ) y el acarreo C.

II. OBJETIVO

- Escribir un programa en VHDL para implementar el medio sumador y verificar la funcionalidad.

III. MATERIAL Y EQUIPO

- Tarjeta Elbert V2 – Spartan 3A FPGA Development Board

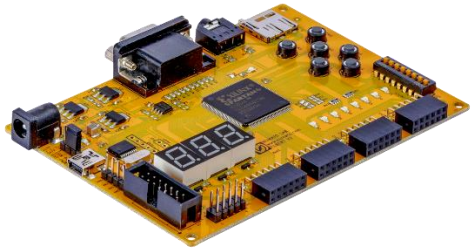


Figura 2: Tarjeta Spartan 3A FPGA

- Xilin ISE, 32-bit Project Navigator.

IV. DESARROLLO

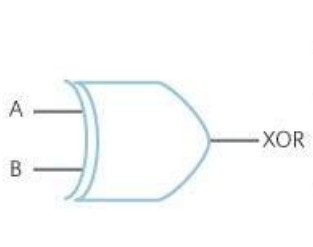
El primer paso para implementar el semisumador después de configurar la tarjeta Elbert V2, consiste en definir las entradas y salidas de este proyecto, tomando en cuenta que el ISE Project Navigator inicializa las librerías a utilizar por si solo:

```
entity sum is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : out  STD_LOGIC;
           S : out  STD_LOGIC);
end sum;
```

Se usan dos entradas y dos salidas, las entradas A y B corresponden a los valores booleanos que serán

sumados, mientras que S corresponde al valor de la suma de A y B, y C será el bit de carry.
Para generar la operación de suma entre estas dos entradas, podemos emplear el uso de la compuerta XOR.

$$X = A \oplus B$$



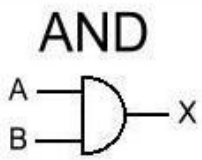
A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Figura 3: Compuerta XOR y su tabla de la verdad

Como podemos ver en la figura 3, los valores de la tabla de la verdad corresponden a las salidas deseadas de los valores deseados del primer digito de la figura 1. La salida de esta compuerta será almacenada en la variable S.

Para generar el carry podemos emplear otra compuerta lógica que en este caso corresponde a la compuerta AND.

Tabla de verdad AND		
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1



Esta salida corresponde al carry de la suma de las dos entradas y es almacenada en la variable C.

El código encargado de hacer las operaciones de las compuertas descritas anteriormente es el siguiente:

```
architecture Behavioral of sum is

begin

S <= A xor B;
C <= A and B;

end Behavioral;
```

V. RESULTADOS

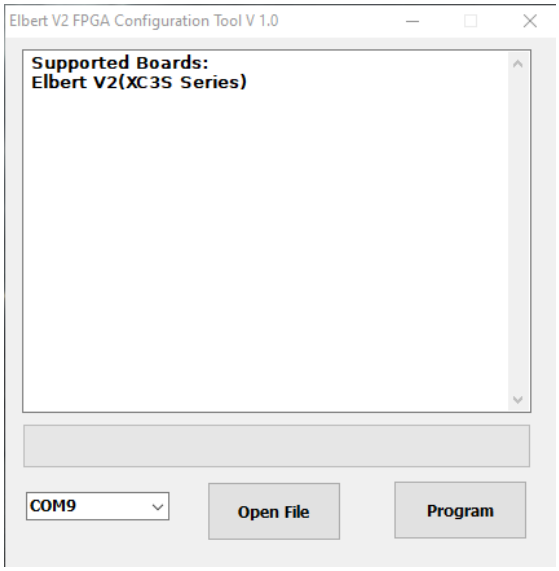
Para comprobar la funcionalidad de este código, procedemos a transferir el programa a la tarjeta, para esto tenemos que crear un archivo ucf que contenga la ruta de las entradas y salidas de la tarjeta que correspondan físicamente a el hardware deseado.

En este caso, representaremos las dos entradas como dos DP Switchs y cada salida con un led indicador.

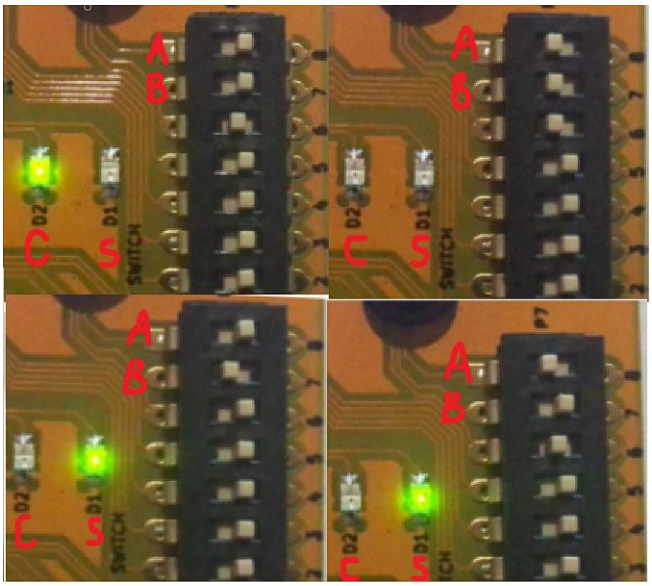
```
NET "A"         LOC = P70   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "B"         LOC = P69   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "C"             LOC = P54   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "S"             LOC = P55   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
```

Ahora procedemos a transferir el código a la tarjeta con el ejecutable proporcionado por el fabricante, en este caso Numato.



Escogemos el puerto correspondiente y enviamos el programa.



En la siguiente imagen podemos ver como las salidas corresponden a las salidas de en los leds, describiendo así las salidas propuestas en la figura 1.

VI. CONCLUSIÓN

Las operaciones de las compuertas lógicas pueden ayudarnos a simplificar los resultados de las entradas de manera significativa. Una compuerta lógica nos puede simplificar las cuatro posibles combinaciones de código necesario para representar de manera lógica las posibilidades de dos entradas y sus resultados dependiendo del operador.

Podemos representar entradas y salidas por medio de la tarjeta numato y así comprobar de manera sencilla el funcionamiento de los algoritmos diseñados en clase.

VII. REFERENCIAS

D. L. Perry, VHDL. New York: McGraw-Hill, 1991.