Práctica 5 - CIRCUITOS ARITMÉTICOS. SUMADOR DE 5 BITS

Objetivos

El propósito de esta práctica es:

- Comprender el funcionamiento de un sumador.
- Usar los bucles **for … generate** para instanciar componentes en VHDL.

Material

- Ordenador personal con Quartus II y ModelSim.
- Tarjeta de desarrollo de lógica programable.

Duración

1 sesión.

Trabajo previo e información de consulta

En esta sesión se harán como ejercicios previos:

- Obtener las ecuaciones lógicas del sumador de 1 bit a partir de la tabla 5.1 de los apuntes (página 86).
- Diseño en VHDL de los bloques necesarios para la práctica.
- Lectura previa de las notas de clase y tutorial de VHDL.
- Lectura del manual de la tarjeta de lógica programable DE1 (disponible en la página web de la asignatura).

Trabajo posterior y entrega de resultados

Se debe entregar un informe de resultados y el trabajo desarrollado en el laboratorio en soporte informático que contenga el proyecto completo (zip de la carpeta de trabajo). Debe incluir:

- Descripción del experimento y los pasos seguidos (tablas de verdad, expresiones lógicas, K-maps).
- Esquemas del circuito, código VHDL y su explicación.
- Simulaciones y su explicación.
- Resultados y conclusiones.

Introducción

En esta práctica se va a diseñar un circuito para sumar números de 5 bits. También se usará el bloque BinA7Seg diseñado en la práctica anterior.

El diagrama de bloques del circuito a diseñar es el siguiente:

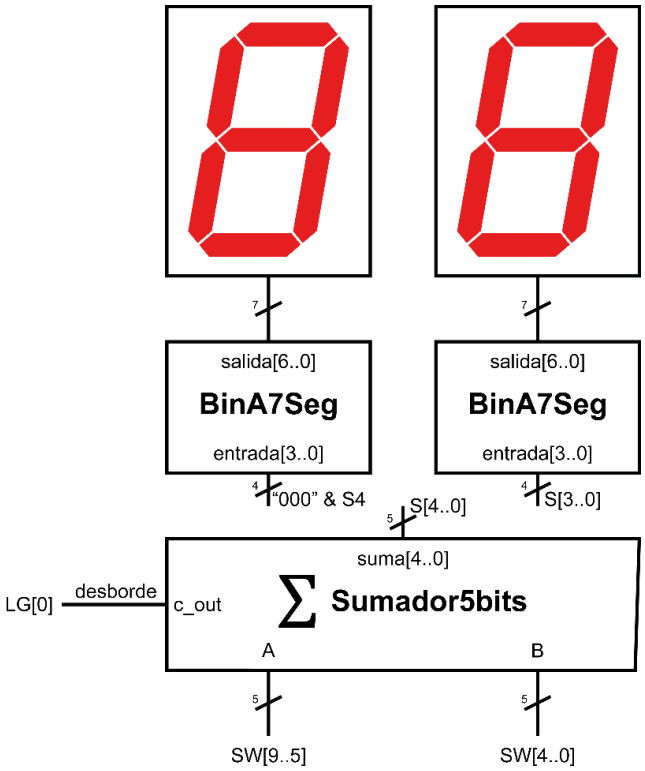

Figura 1. Diagrama de bloques del circuito

Desarrollo práctico

Organizad en Quartus II un proyecto con el nombre **Practica5**. El diagrama general de bloques de la práctica estará compuesto por dos decodificadores de 7 segmentos **BinA7Seg** y el bloque **Sumador5Bits**, además de **10 interruptores** (SW9-SW0) y **2 displays de 7 segmentos** (Display0 y Display1). La figura 1 nos muestra este diagrama general de bloques.

A su vez, el sumador de 5 bits ha de implantarse usando cinco sumadores de 1 bit. A este bloque se le denominará **Sumador1Bit**. Hay que tener en cuenta que la salida del sumador está formada por un número de 5 bits que es el resultado de la suma y un acarreo de salida.

Después de generar el proyecto (**Practica5**), procedemos a continuación con la descripción en VHDL de los bloques **Sumador1Bit.vhd** y **Sumador5Bits.vhd**.

Por último, hay que crear un nuevo archivo VHDL para la jerarquía completa del proyecto llamado **Practica5.vhd**. Este archivo tendrá una arquitectura de tipo *structural* para instanciar los componentes **Sumador5Bits** y **BinA7Seg**, así como todo el conexionado final.

A continuación, con el Manual de la Tarjeta DE1, hay que identificar el pin de la FPGA correspondiente a cada puerto de entrada y de salida. Estos pines se asignarán en el *Pin Planner*.

Señal	Tipo	Componente	Pin FPGA
a(0)	Entrada	SW0	
a(1)	Entrada	SW1	
a(2)	Entrada	SW2	
a(3)	Entrada	SW3	
a(4)	Entrada	SW4	
b(0)	Entrada	SW5	
b(1)	Entrada	SW6	
b(2)	Entrada	SW7	
b(3)	Entrada	SW8	
b(4)	Salida	SW9	
d_a(6)	Salida	Seven Segment Digit 0[a]	
d_a(5)	Salida	Seven Segment Digit 0[b]	
d_a(4)	Salida	Seven Segment Digit 0[c]	
d_a(3)	Salida	Seven Segment Digit 0[d]	
d_a(2)	Salida	Seven Segment Digit 0[e]	
d_a(1)	Salida	Seven Segment Digit 0[f]	
d_a(0)	Salida	Seven Segment Digit 0[g]	
d_b(6)	Salida	Seven Segment Digit 1[a]	
d_b(5)	Salida	Seven Segment Digit 1[b]	
d_b(4)	Salida	Seven Segment Digit 1[c]	
d_b(3)	Salida	Seven Segment Digit 1[d]	
d_b(2)	Salida	Seven Segment Digit 1[e]	
d_b(1)	Salida	Seven Segment Digit 1[f]	
d_b(0)	Salida	Seven Segment Digit 1[g]	
desborde	Salida	LED Green[0]	

Tabla 1. Asignación de patillas de la FPGA a las señales del circuito.

Posteriormente, se debe compilar el circuito diseñado y corregir los errores.

A continuación, procederemos con la **simulación en ModelSim-Altera**. En el anexo A se incluye un banco de prueba para generar los estímulos para la simulación. Ojo, en este caso el testbench sólo se realizará sobre el componente **Sumador5Bits**, y no sobre la jerarquía superior Practica5.

Finalmente volcad a la placa del circuito diseñado para su comprobación por el profesor.

Hay que tener en cuenta que el resultado en los displays aparecerá en modo hexadecimal. Si os sobra tiempo, intentad modificar el circuito para que se muestre el resultado en decimal. De la misma forma, se puede intentar modificar el sumador para que sea una estructura de un sumador de 1 bit con propagación y generación, y un módulo *Carry Look Ahead*.

Anexo A - Testbench para la Práctica (Sumador5Bits.vht)

En este caso, el testbench se aplicará únicamente al componente Sumador5Bits, por lo cual tendremos que crearlo a mano y no mediante la plantilla autogenerada.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY Sumador5Bits_vhd_tst IS
END Sumador5Bits_vhd_tst;

ARCHITECTURE Sumador5Bits_arch OF Sumador5Bits_vhd_tst IS
     -- constants
     -- signals
     SIGNAL a : STD_LOGIC_VECTOR(4 DOWNTO 0);
     SIGNAL b : STD_LOGIC_VECTOR(4 DOWNTO 0);
     SIGNAL c_out : STD_LOGIC;
     SIGNAL s : STD_LOGIC_VECTOR(4 DOWNTO 0);
     COMPONENT Sumador5Bits
     PORT (
         a : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
         b : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
         c_out : OUT STD_LOGIC;
         s : OUT STD_LOGIC_VECTOR(4 DOWNTO 0)
     );
END COMPONENT;

BEGIN
i1 : Sumador5Bits
     PORT MAP (
         -- list connections between master ports and signals
         a => a,
         b => b,
         c_out => c_out,
         s => s
     );

init : PROCESS
-- variable declarations
BEGIN
-- code that executes only once
WAIT;
END PROCESS init;

always : PROCESS
BEGIN
    -- code executes for every event on sensitivity list
    for i in 0 to 31 loop
        a <= std_logic_vector(to_unsigned(i, 5));
        for j in 0 to 31 loop
            b <= std_logic_vector(to_unsigned(j, 5));
            wait for 100 ns;
            assert unsigned(c_out&s) = i+j
                report "fallo en la suma"
                severity failure;
        end loop;
    end loop;
    WAIT;
END PROCESS always;

END Sumador5Bits_arch;
```