Práctica 8 – INTRODUCCIÓN A LOS BIESTABLES Y A LAS MÁQUINAS DE ESTADOS

Objetivos

El propósito de esta práctica es:

- Comprender el funcionamiento de un biestable.
- Manejar el osciloscopio en modo de disparo único para medir retardos.
- Implantar y simular máquinas de estado en VHDL.

Material

- Biestables 74HC573 y 74HC74.
- Tarjeta de entrada / salida y tarjeta de inserción.
- Osciloscopio Digital.
- Ordenador personal con Quartus II y ModelSim.

Duración

1 / 2 sesiones.

Trabajo previo e información de consulta

En esta sesión se harán como ejercicios previos:

- Estudiar los capítulos 7, 8 y 9 de los apuntes de clase. El tema 9 es necesario para la segunda parte de la práctica.
- Revisar las **hojas de características de los biestables** usados, anotando los tiempos de propagación de cada uno de los biestables. Están disponibles en Moodle.
- Diseñar una máquina de estados para detectar el **flanco de bajada** de una señal (segunda parte).

Trabajo posterior y entrega de resultados

Tras el desarrollo de la práctica se debe entregar un informe de resultados. Debe incluir:

- Descripción del experimento y los pasos seguidos.
- Esquemas del circuito, código VHDL y su explicación.
- Resultados y conclusiones.

Desarrollo Práctico

Esta práctica consta de dos partes. En la primera se van a ensayar dos biestables tipo D y en la segunda se va a implantar una máquina de estados sencilla en VHDL.

Primera parte

En esta primera parte se van a ensayar dos biestables tipo D, uno sincronizado por nivel (latch) y otro por flanco (flip-flop). Para cada biestable se realizará el siguiente proceso:

- Conectar todas las entradas de uno de los biestables del chip a interruptores, excepto el reloj que se conectará a un pulsador. En el caso del 74HC573 ha de conectarse la entrada *"Output Control"* a cero.
- Conectar cada una de las salidas a un LED.
- Conectar la alimentación y la tierra del chip (consultad la hoja de características para averiguar cómo conectarlas).
- Obtener la tabla de verdad del biestable. Compararla con la esperada según la hoja de características del fabricante.
- Medir el retardo desde la activación del reloj hasta el cambio de las salidas.
- En el 74HC74 medir también los retardos desde la activación de las entradas asíncronas (preset y clear) hasta el cambio correspondiente en las salidas.
- Comparar las medidas obtenidas con los tiempos especificados por el fabricante en la hoja de características.

En el informe de la práctica se incluirá un cronograma para especificar los retardos medidos en los biestables.

Segunda parte

Se implantará en VHDL una máquina de estados para detectar el flanco de bajada de una señal de entrada. Para ello ha de seguir el proceso siguiente:

- Crear un proyecto en Quartus II con el nombre **Practica8**.
- Describir en VHDL la máquina de estados. Hay que usar como punto de partida la expuesta en los apuntes de clase. El archivo usado para la descripción será el top-level de la práctica, ya que no vamos a implantarla en la placa.
- Compilar el proyecto y crear el **testbench**. En el apéndice A se muestra un testbench que permite excitar a un detector de flanco de subida. **Debe modificarse** para excitar al detector de flanco de bajada que ha diseñado.
- Realizar una simulación RTL, para poder ver la información de las variables de estado. Para visualizar el valor de estas variables, una vez arrancado el Modelsim es necesario seleccionar en la ventana de la izquierda la instancia i1 y en las señales que aparecen en la ventana *Objects* seleccionar las señales estado_act y estado_sig (o como las hayamos llamado) y arrastrarlas a la ventana *Wave*.
- Reiniciar la simulación para poder observar la evolución de las variables de estado.

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

Hay que adaptar este código para un testbench de detección de flanco de **bajada**.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY DetectorFlanco_vhd_tst IS
END DetectorFlanco_vhd_tst;

ARCHITECTURE DetectorFlanco_arch OF DetectorFlanco_vhd_tst IS
    SIGNAL clk : STD_LOGIC:='0';
    SIGNAL e : STD_LOGIC;
    SIGNAL reset_n : STD_LOGIC;
    SIGNAL s : STD_LOGIC;

COMPONENT DetectorFlanco
PORT (
    clk : IN STD_LOGIC;
    e : IN STD_LOGIC;
    reset_n : IN STD_LOGIC;
    s : OUT STD_LOGIC);
END COMPONENT;
BEGIN

i1 : DetectorFlanco
PORT MAP (
    clk => clk,
    e => e,
    reset_n => reset_n,
    s => s);
init : PROCESS
-- variable declarations
BEGIN
 -- code that executes only once
WAIT;
END PROCESS init;

clk <= not clk after 50 ns;
always : PROCESS

BEGIN
    reset_n <= '0';
    e <= '0';
    wait for 100 ns;
    reset_n <= '1';
    wait for 510 ns;
    e <= '1';
    wait for 500 ns;
    e <= '0';
    wait for 500 ns;
    e <= '1';
    wait for 100 ns;
    e <= '0';
    wait for 500 ns;
    assert false
        report "Fin de la simulacion" severity failure;
END PROCESS always;
END DetectorFlanco_arch;
```