Práctica 9 – CERRADURA ELECTRÓNICA

Objetivos

El propósito de esta práctica es:

- Comprender el funcionamiento de las máquinas de estado.
- Diseñar y simular un detector de secuencia para una cerradura electrónica.
- Aprender a describir circuitos secuenciales con lenguajes de descripción de hardware 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 realizarán los siguientes ejercicios previos:

- **Diagrama de estados** del circuito detector de secuencia.
- Descripción en **VHDL de las máquinas de estado**.
- Lectura previa de las notas de clase sobre y tutorial de VHDL.
- Lectura del manual de la tarjeta de lógica programable DE1.

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 (tablas de verdad, expresiones lógicas, mapas de Karnaugh).
- Diagramas de estados, esquemas del circuito, código VHDL y su explicación.
- Simulaciones y su explicación.
- Resultados y conclusiones.

Introducción

Vamos a hacer un circuito secuencial para implementar una sencilla cerradura electrónica.

El circuito detectará una secuencia de pulsación de teclas y activará una salida cuando la secuencia sea completa. Para arrancar el sistema hay que apretar el pulsador que hace de **reset**. A continuación, los pulsadores que activan las señales p0 y p1 sirven para introducir una secuencia al circuito.

Cuando éste detecte la secuencia **p0 – p1 – p1 – p0**, entre las pulsaciones del usuario, debe iluminar un LED verde (**VALID**) para indicar que se ha recibido la secuencia anterior. Se permite el solapamiento de secuencias, de tal forma que si se pulsa **p0 – p1 – p1 – p0 – p1 – p1 – p0** se darán por válidas dos secuencias.

Dicho LED verde permanecerá iluminado hasta que se pulse de nuevo uno de los pulsadores, pudiendo volver a iluminarse si se vuelve a introducir de nuevo la secuencia correcta.

Mientras no exista una secuencia correcta, se iluminará un LED rojo (**ESPERA**).

El alumno debe tener en cuenta que pulsar uno de los interruptores genera un pulso que dura muchos ciclos de reloj. Por tanto, es necesario diseñar un detector de flanco para cada pulsador, en un bloque independiente llamado **DetectorFlanco**.

Desarrollo práctico

Para desarrollar esta práctica, hay que crear en Quartus II un proyecto con el nombre **Practica9**.

El diagrama general de bloques de la práctica estará compuesto por un bloque **DetectorSecuencia**. Además, es necesario el diseño de un bloque que llamaremos **DetectorFlanco**, que puede ser reutilizado de la práctica anterior. La figura 1 nos muestra este diagrama general de bloques.

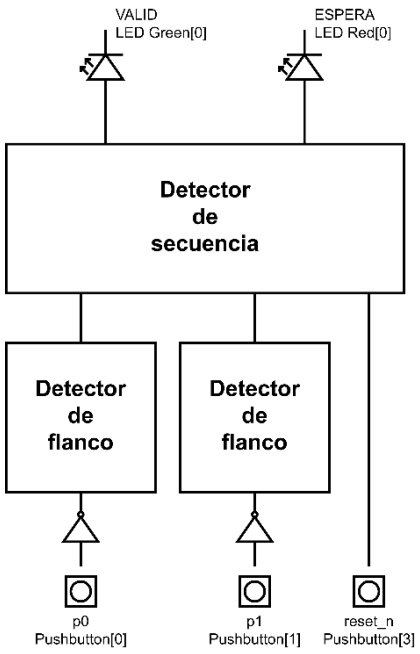

Figura 1. Diagrama de bloques del Detector de Secuencia.

Detector de Secuencia

Realiza la detección de la secuencia **p0 – p1 – p1 – p0**. Tiene como entradas los pulsadores filtrados por el detector de flanco. Las salidas son un **LED verde** y un **LED rojo**.

Detector de flanco

Realiza las funciones de filtrado de los pulsadores. Estos detectores de flanco tienen como entrada los pulsadores de secuencia. Sus salidas son las entradas a la máquina de estados del detector de secuencia.

Podéis utilizar un detector de flanco de subida, negando las entradas de los pulsadores (que son activas a nivel bajo), o podéis usar el detector de flanco de bajada diseñado en la práctica 8.

Inversores

Al ser los pulsadores activos a nivel bajo es conveniente invertir estas entradas, o en su lugar, utilizar detectores de flanco de bajada.

Tras generar el proyecto con el nombre **Practica9**, es necesario definir en VHDL los bloques **DetectorSecuencia** y **DetectorFlanco**. EL archivo de jerarquía superior se llamará Practica9.vht, donde estará el conexionado de todos los bloques.

Es necesario consultar el Manual de la Tarjeta DE1 para identificar el pin de la FPGA con el correspondiente puerto de entrada y de salida, rellenando la Tabla 2.

Señal	**Tipo**	**Componente**	**Pin FPGA**
p0	Entrada	Pushbutton[0]	
p1	Entrada	Pushbutton[1]	
reset_n	Entrada	Pushbutton[3]	
ESPERA	Salida	LED Red[0]	
VALID	Salida	LED Green[0]	
clk	Entrada	CLOCK_50	

Tabla 2. Asignación de las patillas de la FPGA a las señales del circuito.

Hay que tener en cuenta que los dispositivos LED de la placa son activos a nivel alto (para iluminar el LED hay que poner un ‘1’) en el caso de los pulsadores, se genera un ‘1’ lógico si no está presionado y un ‘0’ en el momento de pulsarlo.

También hay que tener en cuenta que al tratarse de un diseño secuencial vamos a usar un reloj para sincronizar el circuito. La placa DE1 provee varios osciladores conectados a los pines de la FPGA. Vamos a utilizar un reloj de 50MHz.

Realizado este paso hay que proceder a la compilación del circuito diseñado, corregir los errores, si los hubiera, en el código VHDL y en el conexionado de los bloques. A continuación, se debe realizar con la simulación en ModelSim. En el Anexo A se incluye un banco de prueba para generar los estímulos para la simulación. Hay que tener en cuenta que en la línea 7 del testbench se ha añadido la inicialización de la señal clk.

Anexo A - Testbench para la Práctica (Practica9.vht)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY Practica9_vhd_tst IS
END Practica9_vhd_tst;

ARCHITECTURE Practica9_arch OF Practica9_vhd_tst IS
SIGNAL clk : STD_LOGIC:='0';
SIGNAL espera : STD_LOGIC;
SIGNAL p0 : STD_LOGIC;
SIGNAL p1 : STD_LOGIC;
SIGNAL reset_n : STD_LOGIC;
SIGNAL valid : STD_LOGIC;
COMPONENT Practica9
PORT (
    clk : IN STD_LOGIC;
    espera : OUT STD_LOGIC;
    p0 : IN STD_LOGIC;
    p1 : IN STD_LOGIC;
    reset_n : IN STD_LOGIC;
    valid : OUT STD_LOGIC);
END COMPONENT;
BEGIN
i1 : Practica9
PORT MAP (
    clk => clk,
    espera => espera,
    p0 => p0,
    p1 => p1,
    reset_n => reset_n,
    valid => valid);

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

clk <= not clk after 50 ns;

always : PROCESS
BEGIN
    reset_n <= '0';
    p1 <= '1';
    p0 <= '1';
    wait for 100 ns;
    reset_n <= '1';
    wait for 10 ns;
    p0 <= '0'; -- Pulso p0
    wait for 200 ns;
    p0 <= '1';
    wait for 200 ns;
    p1 <= '0'; -- Pulso p1
    wait for 200 ns;
    p1 <= '1';
    wait for 300 ns;
    p1 <= '0'; -- Pulso p1
```

```
    wait for 200 ns;
    p1 <= '1';
    wait for 300 ns;
    p0 <= '0'; -- Fin de secuencia: pulso en p0
    wait for 200 ns;
    p0 <= '1';
    assert valid = '1'
        report "Error: No se activa la salida tras introducir una
                secuencia correcta"
        severity warning;
    -- Enlazamos con otra nueva secuencia
    wait for 200 ns;
    p1 <= '0'; -- Pulso p1
    wait for 200 ns;
    p1 <= '1';
    assert valid = '0'
        report "Error: No se desactiva la salida tras volver a pulsar
                otra tecla"
        severity warning;
    wait for 300 ns;
    p1 <= '0'; -- Pulso p1
    wait for 200 ns;
    p1 <= '1';
    wait for 300 ns;
    p0 <= '0'; -- Fin de secuencia: pulso en p0
    wait for 200 ns;
    p0 <= '1';
    wait for 300 ns;
    assert valid = '1'
        report "Error: No se activa la salida tras introducir una
                secuencia correcta"
        severity warning;
    -- Volvemos a introducir una nueva secuencia
    p0 <= '0'; -- Pulso p0
    wait for 200 ns;
    p0 <= '1';
    wait for 200 ns;
    p1 <= '0'; -- Pulso p1
    wait for 200 ns;
    p1 <= '1';
    wait for 300 ns;
    p1 <= '0'; -- Pulso p1
    wait for 200 ns;
    p1 <= '1';
    wait for 300 ns;
    p0 <= '0'; -- Fin de secuencia: pulso en p0
    wait for 200 ns;
    p0 <= '1';
    assert valid = '1'
        report "Error: No se activa la salida tras introducir una
                secuencia correcta"
        severity warning;
    assert false
        report "Fin de la simulación"
        severity failure;
END PROCESS always;
END Practica9_arch;
```