Máquina de estados

Tyrone Novillo
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
tyrone.novillo@ucuenca.edu.ec

***Resumen*—This project focuses on designing a VHDL state machine to control LED rotation on seven-segment displays using the Basys3 board. The state machine facilitates clockwise or counterclockwise LED rotation based on the switch state, with a rotation speed set at 0.5 seconds. Additionally, a center button resets the state machine to the initial state.**

***Keywords*—** VHDL, state machine, LED rotation, Basys3, seven-segment displays, clockwise, counterclockwise, switch, button, reset.

I. Introducción

Las máquinas de estados son modelos abstractos que representan el comportamiento de sistemas que pueden estar en un número finito de estados distintos en diferentes momentos. Estos sistemas se utilizan ampliamente en la ingeniería de sistemas para controlar y coordinar el funcionamiento de dispositivos electrónicos, como en el caso de esta práctica.

En el contexto de VHDL, un lenguaje de descripción de hardware, las máquinas de estados son herramientas fundamentales para diseñar y controlar la lógica digital de manera eficiente y estructurada. Permiten describir el comportamiento secuencial de sistemas digitales de una manera lógica y comprensible.

En este práctica en específica, se utilizará VHDL para diseñar una máquina de estados que controle la rotación de los LEDs en los displays de siete segmentos de la tarjeta Basys3. La máquina de estados se programará para seguir un patrón de rotación en sentido horario o antihorario dependiendo del estado del switch, y se establecerá un intervalo de tiempo de 0.5 segundos para la rotación de los LEDs. Además, se integrará la funcionalidad de reiniciar la máquina de estados al estado inicial con un clic en el botón central.

II. Desarrollo

El objetivo de este proyecto es diseñar una máquina de estados en VHDL que controle la rotación de los LEDs en los displays de siete segmentos de la tarjeta Basys3. La rotación debe ser en sentido horario (a→b→c→d→e→f→a) cuando el switch 1 está en nivel alto y en sentido antihorario en caso contrario. La velocidad de rotación de los LEDs debe ser de 0.5 segundos. Adicionalmente, se debe configurar el boton centro para que al hacer clic, la máquina de estados regrese al estado inicial

Previo a la implementación, es necesario definir los puertos de entrada y salida dentro de la entidad llamada `rotar_leds`.

II-A. Declaración de la entidad

Para describir la funcionalidad de la creación de la señal pwm, se ha creado el archivo de diseño `rotar_leds.vhd`. En este diseño, se define la entidad `rotar_leds` que representa el componente y describe sus puertos de entrada y salida.

La entidad tiene los siguientes puertos:

- `activ_displays`: array de salida para definir cual de los 4 displays está encendido .
- `clk_in`: Entrada de reloj proporcionado por la placa Basys 3 de 100MHz.
- `Display_7segments`: Salida que asigna los segmentos del display de la Basys 3.
- `switch`: Entrada que corresponde al switch para controlar el sentido de la rotación de los segmentos del display.
- `CLR`: Entrada que corresponde al botón del centro para reiniciar la máquina de estados al estado inicial.

II-B. Definición de la arquitectura

Previo a empezar la arquitectura, se definen las siguientes señales intermedias:

- `CLK`: Señal de 0.5 segundo que controla el proceso síncrono de la máquina de estados.
- `contador_reloj`: Este es el contador que permite dividir la frecuencia de 100MHz a una de 2Hz (0.5 segundos).

Para manejar correctamente los estados, se declara un nuevo tipo de señal para representar los estados de la máquina de estados finita (FSM). Este tipo tiene seis estados (cada uno correspondiente a una posición del segmento del display) y dos señales han sido declaradas de este tipo: PS y NS las cuales sirven para saber en que estado está presente la máquina de estados y para asignar el siguiente estado a la máquina.

II-B1. Generación de señales de reloj: Para obtener la senñal de 0.5s a partir de la de 100MHz, se hará un proceso de división de frecuencia. Para ello, se procede a hacer el cálculo del contador de la siguiente manera:

$$\text{Contador} = \frac{100MHz}{2Hz} \cdot \frac{1}{2} = 25\_000\_000$$

Cuando se detecta un flanco de subida en esta señal, el contador `contador_reloj` se incrementa en uno si es menor a 25000000. Cuando el contador alcanza este valor, se invierte el valor de la señal `CLK` (generando así una señal cuadrada de frecuencia reducida), y el contador se restablece a cero para comenzar de nuevo el conteo.

II-C. Proceso síncrono de la máquina de estados

El proceso síncrono de la máquina de estados maneja el reset asíncrono y la asignación del nuevo estado de acuerdo con al flanco de la senñal CLK. Este proceso se ejecuta cada 0.5 segundos, cuando se establece un cambio de estado o cuando se presiona el botón CLR. Lo único importante de este proceso con etiqueta `sync_proc` es que cambia del estado presente al estado siguiente cada medio segundo.

II-D. Proceso combinacional

Este proceso se ejecuta cuando la máquina de estados cambia de estado o cuando se modifica el valor del switch. Para definir la secuencia del espacio de estados se utiliza una declaración case en VHDL para determinar el estado siguiente (NS) basado en el estado presente (PS) y el valor del switch. En cada caso del estado presente, se asigna un valor específico al display_7segments para controlar qué segmento del display de siete segmentos se enciende. Luego, el valor del switch se evalúa para determinar la transición al siguiente estado. Si el switch está en nivel alto ('1'), la máquina de estados avanza al siguiente estado en secuencia (por ejemplo, de ST0 a ST1, de ST1

a ST2, y así sucesivamente). Si el switch está en nivel bajo ('0'), la máquina de estados retrocede al estado anterior en la secuencia (por ejemplo, de ST0 a ST5, de ST1 a ST0, y así sucesivamente). Este comportamiento asegura una rotación de LEDs en sentido horario cuando el switch está en '1' y en sentido antihorario en caso contrario. Si se produce un estado no especificado, el sistema vuelve al estado ST0 y el display se configura para encender el LED correspondiente a este estado inicial.

II-E. Constraints

El archivo constraints.xdc proporciona las instrucciones necesarias para establecer la conexión entre los pines físicos de la tarjeta Basys 3 y las entradas y salidas definidas en la entidad VHDL. Dentro del archivo, se realizan las siguientes configuraciones.

En primer lugar, se configura la señal de reloj conectando el pin W5 al puerto clk_in. Para los switches, se asigna el pin V17 al puerto switch. En cuanto al display de siete segmentos, se asignan los pines correspondientes a cada uno de los segmentos del display: el pin W7 al segmento 6, W6 al segmento 5, U8 al segmento 4, V8 al segmento 3, U5 al segmento 2, V5 al segmento 1 y U7 al segmento 0. Asimismo, se asignan los pines para los displays activos: el pin U2 al primer display, U4 al segundo, V4 al tercero y W4 al cuarto. Finalmente, el botón, se asigna el pin U18 al puerto CLK y se desactiva la ruta dedicada del reloj para esta señal con la propiedad CLOCK_DEDICATED_ROUTE FALSE en el net CLR_IBUF. Este archivo asegura que cada señal en el diseño se conecta a los pines correctos de la FPGA y se configura adecuadamente según el estándar de señal especificado.

III. Conclusiones

La máquina de estado permiten tener una estructura clara y ordenada para gestionar las transiciones entre estados, lo que facilita tanto el diseño como el mantenimiento del sistema. Además, permiten una mayor precisión en el control del comportamiento del hardware, asegurando que cada segmento del display se active en el momento y en la secuencia correctos.

La máquina de estados diseñada en este proyecto es de tipo Moore en donde la salida depende únicamente del estado actual y no de las entradas. La salida de cada estado se asigna directamente dentro del estado y no depende de una entrada externa. En este caso, esto significa que la salida a los displays de siete segmentos es estable y predecible, ya que no varía hasta que se produce una transición de estado.

Es recomendable para propósitos de debugging de la máquina de estados agregar una señal que permita resetear la máquina de estados a un estado inicial, en este caso, se utilizó el boton centro para regresar al estado ST0.

IV. Anexos

IV-A. Archivo rotar_leds.vhd

```
----------------------------
-- Carrera: Ingenieria en Telecomunicaciones
-- Instituto: Universidad de Cuenca
-- Autor: Tyrone Novillo
-- Fecha de creacion: 05/31/2024 09:13:02 AM

-- Nombre del diseno: rotar_leds
----------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity rotar_leds is
port (
    clk_in : in std_logic;
    activ_displays : out std_logic_vector(3 downto 0);
    Display_7segments : out std_logic_vector(6 downto 0
    switch : in std_logic;
    CLR : in std_logic
    );
end rotar_leds;

architecture Behavioral of rotar_leds is
    type state_type is (ST0, ST1, ST2, ST3, ST4, ST5);
    signal PS, NS : state_type;
    signal CLK : std_logic;
    signal contador_reloj : integer := 0;
begin
    activ_displays <= "1110";

    reloj_05s: process(clk_in)
    begin
        if rising_edge(clk_in) then
            if contador_reloj < 25_000_000 then
                contador_reloj <= contador_reloj + 1;
            else
                CLK <= not CLK;
                contador_reloj <= 0;
            end if;
        end if;

    end process reloj_05s;

    sync_proc: process(CLK, NS, CLR)
    begin
        if (CLR = '1') then
            PS <= ST0;
        elsif (rising_edge(CLK)) then
            PS <= NS;
        end if;
    end process sync_proc;

    comb_proc: process(PS, switch)
    begin
        case PS is
            when ST0 =>
                display_7segments <= "0111111";
                if (switch = '1') then NS <= ST1;
                else NS <= ST5;
                end if;
            when ST1 =>
                display_7segments <= "1011111";
                if (switch = '1') then NS <= ST2;
                else NS <= ST0;
                end if;
            when ST2 =>
                display_7segments <= "1101111";
                if (switch = '1') then NS <= ST3;
                else NS <= ST1;
                end if;
            when ST3 =>
                display_7segments <= "1110111";
                if (switch = '1') then NS <= ST4;
                else NS <= ST2;
                end if;
            when ST4 =>
                display_7segments <= "1111011";
                if (switch = '1') then NS <= ST5;
                else NS <= ST3;
                end if;
            when ST5 =>
                display_7segments <= "1111101";
                if (switch = '1') then NS <= ST0;
```

```
                else NS <= ST4;
                end if;
            when others =>
                display_7segments <= "0111111";
                NS <= ST0;
        end case;
    end process comb_proc;

end Behavioral;
```

IV-B. Archivo de Constraints Basys-3-Master.xdc

```
## Clock signal
set_property -dict { PACKAGE_PIN W5
↪   IOSTANDARD LVCMOS33 } [get_ports clk_in]


# Switches
set_property -dict { PACKAGE_PIN V17
↪   IOSTANDARD LVCMOS33 } [get_ports {switch}]

#7 Segment Display
set_property -dict { PACKAGE_PIN W7
↪   IOSTANDARD LVCMOS33 } [get_ports
↪   {Display_7segments[6]}]
set_property -dict { PACKAGE_PIN W6
↪   IOSTANDARD LVCMOS33 } [get_ports
↪   {Display_7segments[5]}]
set_property -dict { PACKAGE_PIN U8
↪   IOSTANDARD LVCMOS33 } [get_ports
↪   {Display_7segments[4]}]
set_property -dict { PACKAGE_PIN V8
↪   IOSTANDARD LVCMOS33 } [get_ports
↪   {Display_7segments[3]}]
set_property -dict { PACKAGE_PIN U5
↪   IOSTANDARD LVCMOS33 } [get_ports
↪   {Display_7segments[2]}]
set_property -dict { PACKAGE_PIN V5
↪   IOSTANDARD LVCMOS33 } [get_ports
↪   {Display_7segments[1]}]
set_property -dict { PACKAGE_PIN U7
↪   IOSTANDARD LVCMOS33 } [get_ports
↪   {Display_7segments[0]}]

set_property -dict { PACKAGE_PIN U2
↪   IOSTANDARD LVCMOS33 } [get_ports
↪   {activ_displays[0]}]
set_property -dict { PACKAGE_PIN U4
↪   IOSTANDARD LVCMOS33 } [get_ports
↪   {activ_displays[1]}]
set_property -dict { PACKAGE_PIN V4
↪   IOSTANDARD LVCMOS33 } [get_ports
↪   {activ_displays[2]}]
set_property -dict { PACKAGE_PIN W4
↪   IOSTANDARD LVCMOS33 } [get_ports
↪   {activ_displays[3]}]


##Buttons
set_property -dict { PACKAGE_PIN U18
↪   IOSTANDARD LVCMOS33 } [get_ports CLR]
set_property CLOCK_DEDICATED_ROUTE FALSE
↪   [get_nets CLR_IBUF]


## Configuration options, can be used for all
↪   designs
set_property CONFIG_VOLTAGE 3.3
↪   [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI
↪   boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE
↪   [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33
↪   [current_design]
set_property CONFIG_MODE SPIx4
↪   [current_design]
```

IV-C. Funcionamiento de la maquina de estados en la Basys 3

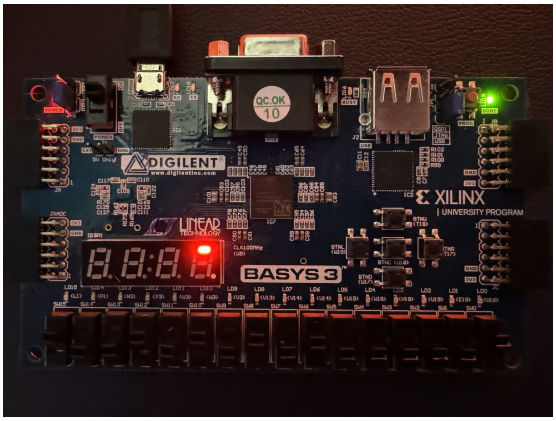

Figura 1: Primer estado de la máquina de estados ST0.

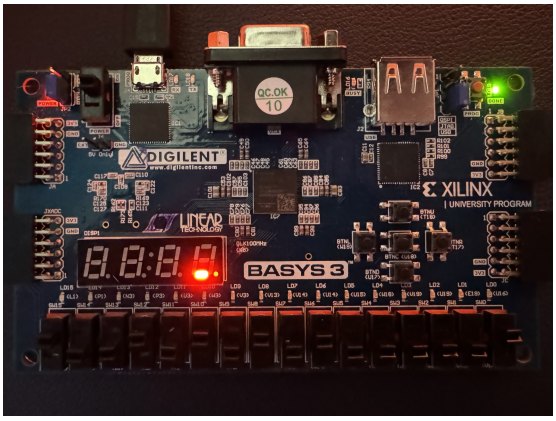

Figura 2: Cuarto estado de la máquina de estados ST3.

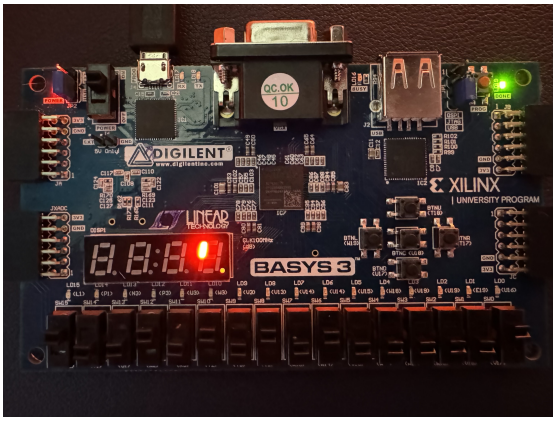

Figura 3: Sexto estado de la máquina de estados ST5.