Máquinas de Estado Finita (FSM)

Claudia Cedeño, Bhavini Ahir

Universidad Latina de Panamá

Resumen – una maquina de estado representa el comportamiento de un sistema, en donde la salida dependerá de la señal de entrada o transición y del estado en que se encuentre. Las maquinas de estado son de uso común en sistemas de transmisión de datos, uso de protocolos de comunicación, etc. Este trabajo ejemplifica el comportamiento de una máquina de estado.

I. INTRODUCCIÓN

Llamamos maquina de estado a un modelo de comportamiento de un sistema con entradas y salidas, en donde las salidas dependen tanto de las señales de entrada actuales como de las anteriores.

Las máquinas de estado finito, más conocidas por su acrónimo en inglés FSM (Finite State Machine), se utilizan ampliamente en el diseño de circuitos digitales, para describir el comportamiento de un sistema según el valor de sus entradas y de cómo van cambiando en el tiempo. Un sistema está compuesto de *estados* por los que va pasando el sistema, de *señales de entrada* que modifican esos estados y de *señales de salida* que pueden utilizarse para conocer el estado del sistema y actuar en consecuencia.

II. MARCO TEORICO

a. Máquina de estado finito (FSM)

Una Máquina de Estado Finito (Finite State Machine), llamada también Autómata Finito es una abstracción computacional que describe el comportamiento de un sistema reactivo mediante un número determinado de Estados y un número determinado de Transiciones entre dicho Estados.

Las Transiciones de un estado a otro se generan en respuesta a eventos de entrada externos e internos; a su vez estas transiciones y/o subsecuentes estados pueden generar otros eventos de salida. Esta dependencia de las acciones (respuesta) del sistema a los eventos de entrada hace que las FSM sean una herramienta adecuada para el diseño de Sistemas Reactivos y la Programación Conducida por Eventos (Event Driven Programming), cual es el caso de la mayoría de los sistemas embebidos basados en microcontroladores o microprocesadores.

Las MEF se describen gráficamente mediante los llamados Diagramas de Estado Finito (DEF), llamados también Diagramas de Transición de Estados.

Un ejemplo muy visual podría ser un semáforo, el cuál dispone de tres estados diferentes, uno para cada color.

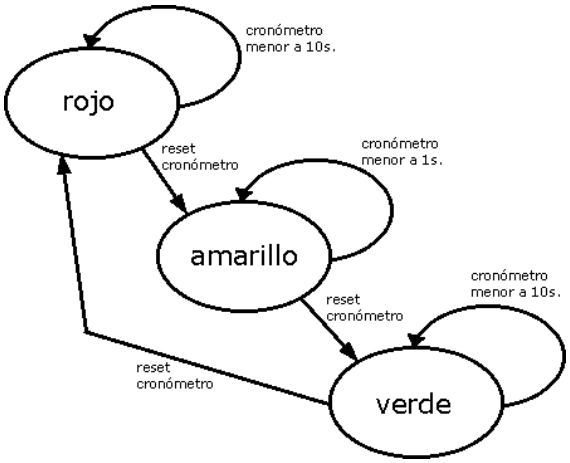

Figura 1. Ejemplo de maquina de estado en semáforo

Las entradas del sistema las podría generar un temporizador que activa una señal cada cierto tiempo, indicando que hay que pasar al siguiente estado. Por último, las salidas del sistema podrían ser tres señales que indiquen qué lámpara, de las tres disponibles, tiene que encenderse

TABLA 1
Ventajas y desventajas de las FSM

Ventajas	*Desventajas*
• Son intuitivas y fáciles de entender • Son universalmente aplicables • Su uso es común un sistema de transmisión de datos y el uso de protocolos de comunicación • Minimiza la programación	• No funcionan bien en sistemas con centenas o mas estados • No permiten una jerarquización de los componentes que minimice la repetición innecesaria de ciertos estados

III. DESARROLLO

A continuación, se describirán los pasos realizados para la creación del dispositivo:

a. Investigación

Las FSM, representan una técnica especial de modelado de circuitos lógicos secuenciales; esta conceptuación es de mucha utilidad en el diseño de circuitos donde todas sus funciones pueden ser exactamente listadas, conteniendo todos los posibles estados y sus condiciones para que estos evolucionen de uno a otro estado.

Existen dos representaciones fundamentales de las FSM una relacionada con sus especificaciones llamada diagrama de transición de estados y otra relacionada al hardware conocida como lógica combinacional/secuencial.

Para diseñar usando FSM el primer paso fundamental es la declaración de todos los posibles estados en los que puede caer el circuito a diseñar, estos se pueden listar, declarando un nombre corto del estado, la salida que debe reflejar en el momento de llegar a él y por supuesto las funciones de evolución de estados, con esta información ya podemos desarrollar una representación formal de la máquina.

a. Diagramas

La metodología aplicada para el desarrollo de este proyecto fue en primer punto, aplicar los conocimientos de materia de Diseño de Sistema Digital (FPGA) y una vez establecido el propósito procedemos a montaje del demás dispositivo y se realiza las pruebas finales.

1. Diagrama de bloques

A nivel físico, una de las formas más efectivas de implementar una FSM es según el siguiente esquema.

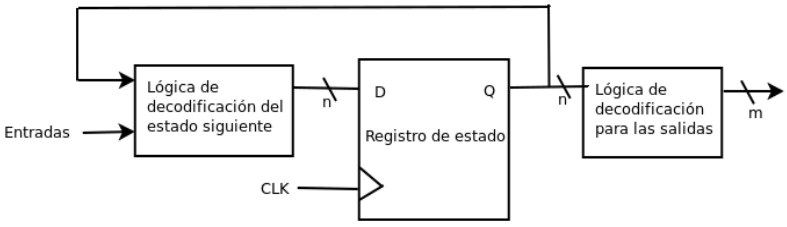

Figura 2. Diagrama de bloque de la FSM

Para almacenar el estado actual del sistema se necesita un elemento de memoria, en este caso usaremos un registro de biestables de tipo D, es por ende un bloque secuencial. El primer bloque se encarga de generar el estado siguiente a partir del estado actual y de las entradas del sistema. El segundo bloque es la memoria. El tercero genera las señales de salida a partir del estado actual (es una máquina de tipo Moore).

2. Diagrama esquemático

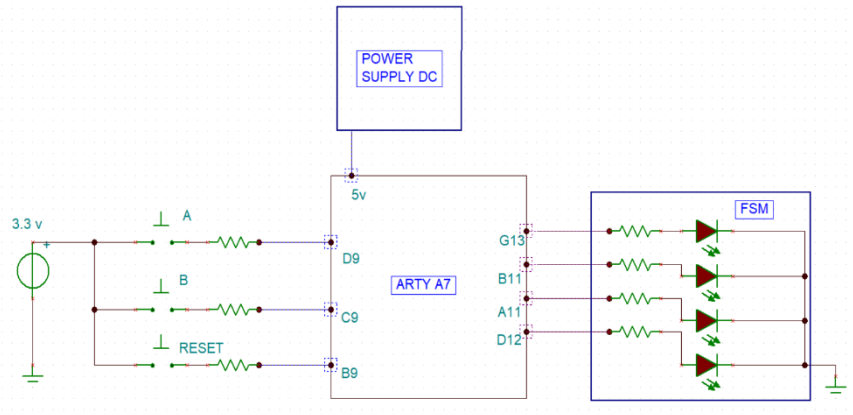

Figura 3. Circuito físico del FSM

b. *Materiales*

Para el desarrollo del dispositivo físico se utilizaron los siguientes materiales:

- Placa Arty A7
- Cable micro USB para programar y alimentar la placa
- Xilinx Vivado instalado
- Switch de entrada para las transiciones
- 4 pilots led para representar los estados

c. Programación

Empleando el conocimiento realizando las diversas practicas con FPGA y aplicando conocimientos teóricos de lo mismo empleamos la programación siguiente para poder observar el comportamiento de una máquina de estado.

Programación de pines en Arty

```
## Clock signal
set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports {CLK}];#IO_L12P_T1_MRCC_35 Sch=gclk[100]

create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK}];

##Buttons
set_property -dict { PACKAGE_PIN D9 IOSTANDARD LVCMOS33 } [get_ports { BTN[0] }]; #IO_L6N_T0_VREF_16 Sch=btn[0]
```

```
set_property -dict { PACKAGE_PIN C9 IOSTANDARD LVCMOS33 } [get_ports { BTN[1] }]; #IO_L11P_T1_SRCC_16 Sch=btn[1]

set_property -dict { PACKAGE_PIN B9 IOSTANDARD LVCMOS33 } [get_ports { BTN[2] }]; #IO_L11N_T1_SRCC_16 Sch=btn[2]


##Pmod Header JD
set_property -dict { PACKAGE_PIN D4 IOSTANDARD LVCMOS33 } [get_ports { LED[0] }]; #IO_L11N_T1_SRCC_35 Sch=jd[1]

set_property -dict { PACKAGE_PIN D3 IOSTANDARD LVCMOS33 } [get_ports { LED[1] }]; #IO_L12N_T1_MRCC_35 Sch=jd[2]

set_property -dict { PACKAGE_PIN F4 IOSTANDARD LVCMOS33 } [get_ports { LED[2] }]; #IO_L13P_T2_MRCC_35 Sch=jd[3]

set_property -dict { PACKAGE_PIN F3 IOSTANDARD LVCMOS33 } [get_ports { LED[3] }]; #IO_L13N_T2_MRCC_35 Sch=jd[4]
```

Programación de máquina de estado

```
--https://www.digilogic.es/maquinas-de-estado-finito-fsm-vhdl/
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity FSM is
    Port ( CLK : in STD_LOGIC;
           BTN : in STD_LOGIC_VECTOR (2 downto 0);
           LED : out STD_LOGIC_VECTOR (3 downto 0));
end FSM;

architecture Behavioral of FSM is
    -- alias
    alias RST : STD_LOGIC is BTN(2);
    alias A : STD_LOGIC is BTN(0);
    alias B : STD_LOGIC is BTN(1);
    alias Y0 : STD_LOGIC is LED(0);
    alias Y1 : STD_LOGIC is LED(1);
    alias Y2 : STD_LOGIC is LED(2);
    alias Y3 : STD_LOGIC is LED(3);

    -- declaraciones modelo FSM
    type STATES is (S0, S1, S2, S3);
    signal state_reg, state_next: STATES;
```

```
begin

    -- registro de estados
    process(CLK)
    begin
        if CLK'event and CLK='1' then
            if RST='1' then
                state_reg <= s0;
            else
                state_reg <= state_next;
            end if;
         end if;
    end process;

    -- Lógica de estado siguiente (circuito combinacional)
    process (state_reg, A, B)
    begin
        state_next <= state_reg;
        case state_reg is
        when S0 =>
            if A='1' then
                state_next <= S1;
            end if;

        when S1 =>
            if B='1' then
                state_next <= S2;
            end if;
        when S2 =>
            if A='1' then
                state_next <= S3;
            end if;
        when S3 =>
            state_next <= S3;
        end case;
     end process;

    -- salida tipo Moore
    process (state_reg)
    begin
        -- estableciendo la salida por defecto
        -- nos aseguramos de crear un circuito
        -- combinacional sin latches.
        Y0 <= '0';
        Y1 <= '0';
        Y2 <= '0';
        Y3 <= '0';
        case state_reg is
        when S0 => Y0 <= '1';
        when S1 => Y1 <= '1';
        when S2 => Y2 <= '1';
        when S3 => Y3 <= '1';
        end case;
    end process;

end Behavioral;
```

IV. CONCLUSIÓN

Las máquinas de estado aportan un componente visual que facilita el análisis y diseño del sistema ya que permite representar de manera sencilla y mas practica y entendible los componentes o entradas de un sistema y analizar todos los posibles estados a los que actuarían todas las señales de entrada.

Siendo así su mayor utilidad el análisis de comportamiento de un sistema con las variables (señales de entrada y estado) que posee dicho sistema.

Las máquinas de estado son una herramienta para el desarrollo de controladores digitales.

V. REFERENCIAS

- Máquinas de estado finito VHDL
  - Link:https://www.digilogic.es/maquinas-de-estado-finito-fsm-vhdl/
- Introducción a las Máquinas de Estado Finito
  - Link:http://tecbolivia.com/index.php/articulos-y-tutoriales-microcontroladores/13-introduccion-a-las-maquinas-de-estado-finito