IE0323
Sistemas Digitales I

Laboratorio #8
Máquinas de Estado

Objetivos

- Diseñar y modelar una máquina de estado finito en Verilog para resolver un problema específico.
- Comprender la estructura básica de una FSM y cómo utilizar estados y transiciones para controlar el flujo de un sistema digital.
- Implementar el comportamiento de una FSM en una FPGA utilizando Intel Quartus Prime.
- Desarrollar habilidades en la depuración de FSM, identificando posibles errores en la transición de estados o en la lógica de control.

Máquinas de Estado Finito

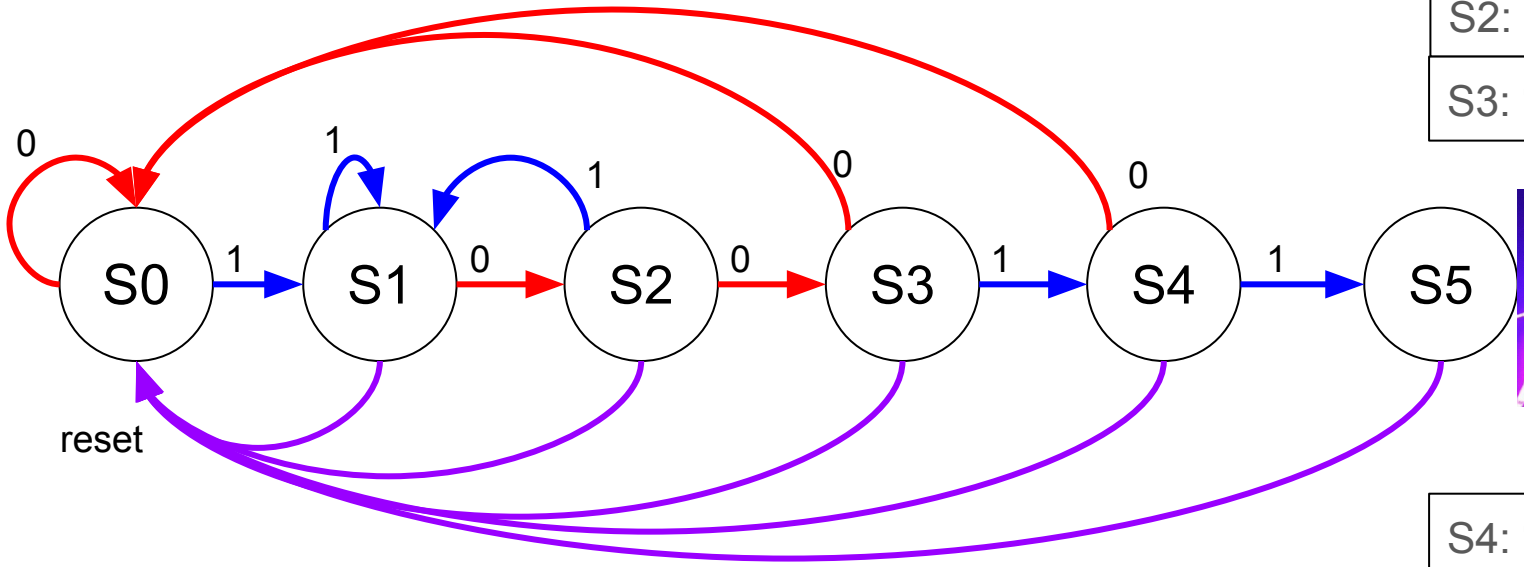
Modelo matemático de computación que puede estar en uno (de un conjunto finito) de estados en cualquier tiempo dado.

Tiene transiciones entre estados, dependiendo de su estado actual y las entradas

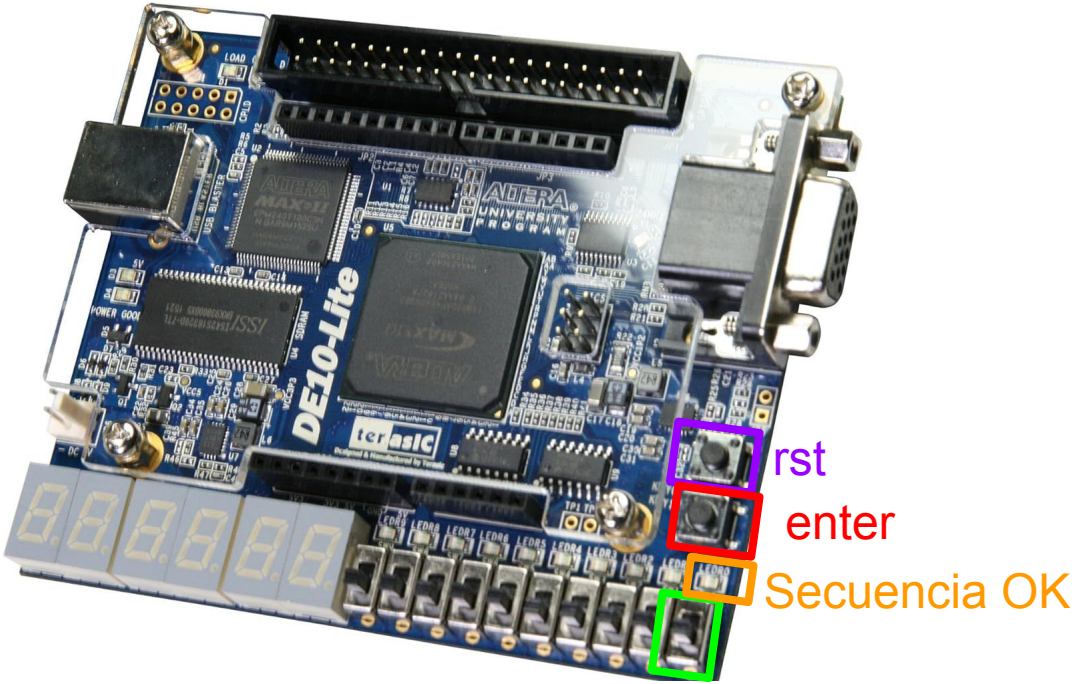
Es un sistema con un conjunto definido de condiciones (estados) que cambian en respuesta a eventos específico (entradas)

Ejemplo: Detector de Secuencia 1 0 0 1 1

S0: Ningún elemento de la secuencia
S1: “1” recibido
S2: “10” recibido
S3: “100” recibido



S4: “1001” recibido
S5: “10011” recibido


DE10-Lite
rst
enter
Secuencia OK
Valor Ingresado

Código en Verilog

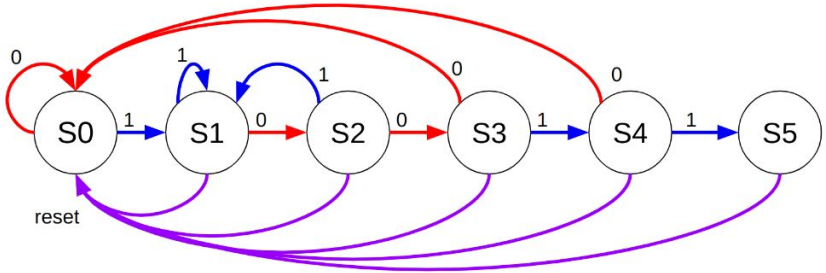
```
module lab8_ejemplo ( clk , rst , enter , in , out );
  // Las entradas son :
  // Reloj , reset , boton de enter y dato del switch
  input clk , rst , enter , in ;
  // La salida es el LED
  output reg out ;
  // Codificacion estados
  localparam S0 = 3'b000;
  localparam S1 = 3'b001;
  localparam S2 = 3'b010;
  localparam S3 = 3'b011;
  localparam S4 = 3'b100;
  localparam S5 = 3'b101;

  reg [ 2 : 0 ] state , next_state ;
```

Lógica de Próximo Estado

Codificación de Estados

```
  // Cuando llega un reset , o se presiona el boton
  //Se calcula la l gica de pr ximo estado
  always @ ( negedge enter , negedge rst ) begin
    // Si hay reset , vuelva a S0
    if ( rst == 0) next_state = S0 ;
    // Sino , siga el diagrama de estados
    else begin
      case ( state )
        S0 : begin
          if ( in == 1) next_state = S1 ;
          else next_state = S0 ;
        end
        S1 : begin
          if ( in == 1) next_state = S1 ;
          else next_state = S2 ;
        end
        S2 : begin
          if ( in == 1) next_state = S1 ;
          else next_state = S3 ;
        end
        S3 : begin
          if ( in == 0) next_state = S0 ;
          else next_state = S4 ;
        end
        S4 : begin
          if ( in == 0) next_state = S0 ;
          else next_state = S5 ;
        end
        S5 : next_state = state ;
        default: next_state = state ;
      endcase
    end
  end
```



```
17  // Cuando llega un reset , o se presiona el boton
18  //Se calcula la l gica de pr ximo estado
19  always @ ( negedge enter , negedge rst ) begin
20    // Si hay reset , vuelva a S0
21    if ( rst == 0) next_state = S0 ;
22    // Sino , siga el diagrama de estados
23    else begin
24      case ( state )
25        S0 : begin
26          if ( in == 1) next_state = S1 ;
27          else next_state = S0 ;
28        end
29        S1 : begin
30          if ( in == 1) next_state = S1 ;
31          else next_state = S2 ;
32        end
33        S2 : begin
34          if ( in == 1) next_state = S1 ;
35          else next_state = S3 ;
36        end
37        S3 : begin
38          if ( in == 0) next_state = S0 ;
39          else next_state = S4 ;
40        end
41        S4 : begin
42          if ( in == 0) next_state = S0 ;
43          else next_state = S5 ;
44        end
45        S5 : next_state = state ;
46        default: next_state = state ;
47      endcase
48    end
49  end
```

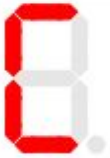
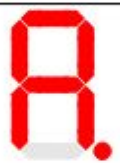
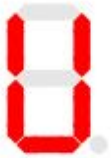
```
51    // Cada flanco positivo del reloj
52    always @ ( posedge clk ) begin
53      // Actualice el estado
54      state <= next_state ;
55      // Si esta en S5 , encienda el LED
56      if ( state == S5 ) out <= 1;
57      // Sino apaguelo
58      else out <= 0;
59    end
60
61  endmodule
```


Lógica de Salida

Trabajo en Clase

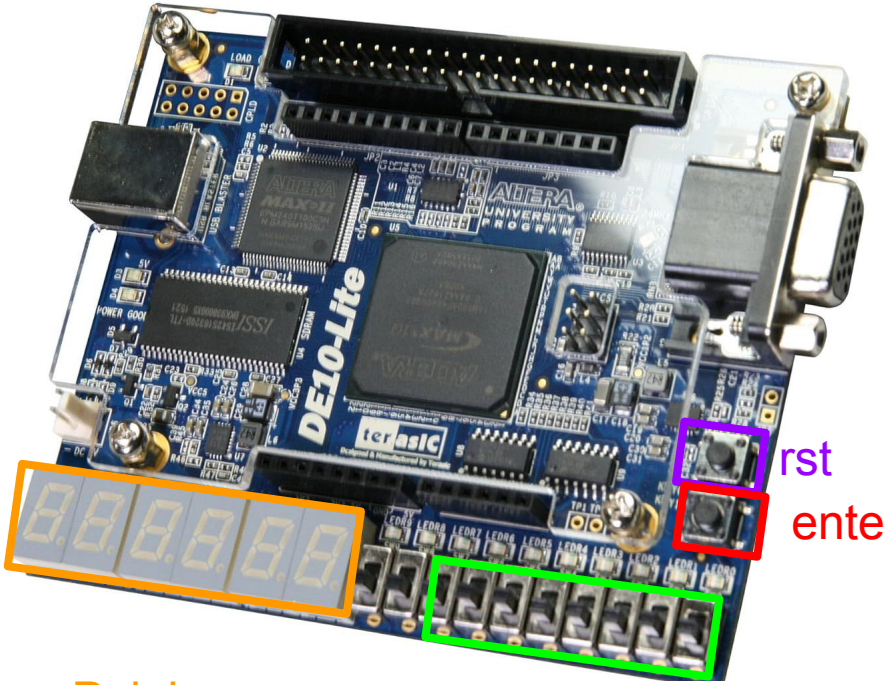
- El circuito recibirá 6 letras codificadas en ASCII por medio de los interruptores SW7-SW0
- El circuito capturará cada letra cuando se presione el botón Key1
- El circuito capturará 6 letras y mantendrá el orden en que fueron ingresadas (la primera letra ingresada deberá desplegarse en el display más a la izquierda). Estas 6 letras serán interpretadas como una palabra
- Cuando se capturen las 6 letras, el circuito mostrará la palabra capturada en los displays de siete segmentos usando la codificación indicada en la Tabla
- Si la entrada seleccionada no corresponde a una letra válida, se deberá desplegar la palabra "ERROR" utilizando los displays

Señal	Componente
Caracter a desplegar	SW7-SW0
Reset	Key0
Capturar caracter	Key1
Despliegue texto	HEX5-HEX0

Letra	ASCII	Representación en 7seg	Letra	ASCII	Representación en 7seg
C	8'b01000011		O	8'b01001111	
E	8'b01000101		R	8'b01010010	
I	8'b01001001		U	8'b01010101	

Demo

1. Diagrama de Estados
2. FSM funcionando



Palabra

`letra5 ... letra0`

Letra Ingresada