¿CÓMO CONSTRUIR UNA MÁQUINA DE ESTADOS Y NO MORIR EN EL INTENTO?

1. Máquina De Estados Finitos, un acercamiento a qué es y para qué sirve

(1)Una máquina de estados es una estructura de lógica, combinatoria y del siguiente estado, que sirve para el modelado de sistemas que se comportan de a cuerdo a un número finito de estados internos. Las transiciones entre estados dependen del estado actual y las entradas externas. A diferencia de un circuito secuencial normal, los cambios de estado del FSM (Finite State Machine) no representan un patrón repetitivo de un único camino, sino que describen una estructura más compleja, construida desde cero en función de la aplicación para la cual se diseña. Las máquinas de estados pueden presentar salidas de dos tipos:

Salidas tipo Moore, si están únicamente en función del estado.

Salidas tipo Mealy, si dependen del estado y de las entradas externas.

2. ¿Cómo funciona?

(1)El corazón de la máquina de estados finitos es el registro de estado y sus bits de estado, no es más que un conjunto de flip-flops tipo-D y una representación binaria de cada estado. En cada ciclo de reloj, los bits de estado cambian al siguiente estado. El siguiente estado es una función de lógica combinatoria de las entradas externas y el estado actual, a su vez; Las salidas del FSM son una función de lógica combinatoria de las entradas y el estado actual; tal como se ilustra en la figura 1.

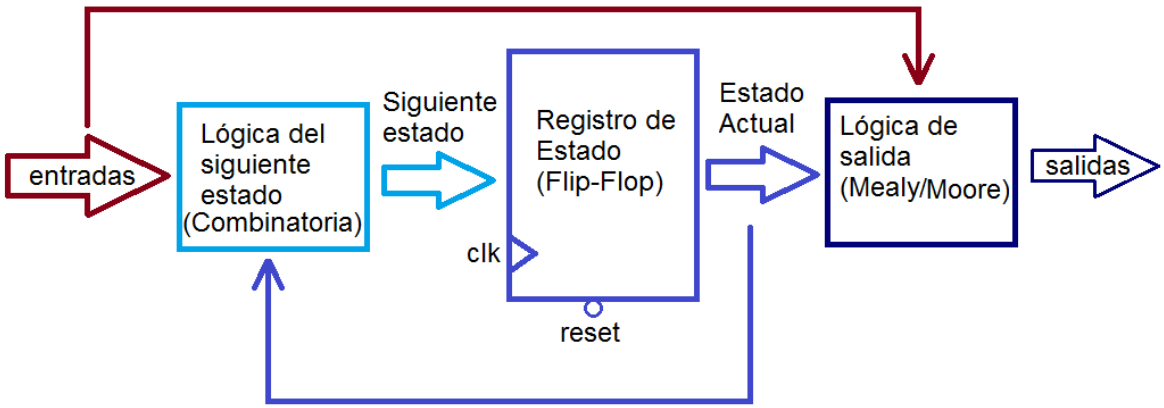

Figura 1 Diagrama de bloques de una máquina de estados.

3. Sí, pero ¿cómo se hace?

Una máquina de estados puede construirse de varias formas. Sin embargo se recomienda que conserve un orden lógico, de fácil lectura y comprensión; que además sea un sistema fiable, seguro, robusto; pero sobre todo, simple. Fácil, ¿verdad?

Para ilustrar todo, y ver lo sencillo que es, haremos algunos ejemplos de FSM que más tarde nos servirán de guía para hacer cualquier otro, ya sea para controlar un integrado, una pantalla LCD o un reactor nuclear. Primero vamos a construir una máquina de estados que modele el funcionamiento de un semáforo. El semáforo que haremos tiene dos comportamientos diferentes, dependiendo del valor de una entrada externa *on_off*, si está en bajo: el semáforo está fuera de servicio y se mostrará un patrón que alternará entre todas las luces apagadas y todas encendidas; si está en alto: inicia con la luz verde, luego de un tiempo pasa a la amarilla y finalmente a la roja, reiniciando el ciclo hasta que la entrada *on_off* vuelva estar en bajo.

3.1. **Primer paso:** Dibujar un diagrama de flujo del FSM, incluyendo estados, entradas y salidas.

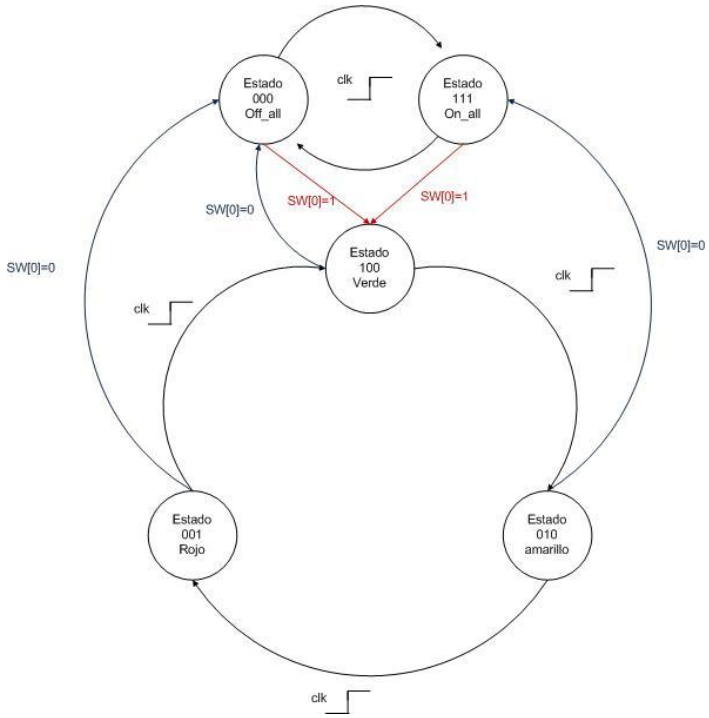

Figura 2 Diagrama de estados de la operación de las luces del semáforo

***Nota**: El reloj de la máquina de estados debe tener el periodo correspondiente a la duración de encendido y apagado de cada luz.*

3.1.1. Descripción del diagrama: El FSM inicia en estado de fuera de servicio. Alternando las luces, en cada pulso del reloj *clk,* entre los estados *off_all* y *on_all;* mientras *on_off*

esté en bajo. Si SW[0] cambia a alto, el FSM entrará en estado activo y, en cada pulso de reloj *clk,* seguirá el patrón de luces: verde, amarillo y rojo.

Tabla 1 Tabla de codificación de estado.

Estado	Luz Verde	Luz Amarilla	Luz Roja
Verde	1	0	0
Amarillo	0	1	0
Rojo	0	0	1
Off_all	0	0	0
On_all	1	1	1

3.1.2. Si el dibujo del diagrama no basta para empezar a escribir el HDL, otra alternativa es la tabla del siguiente estado. Son dos formas de representar el funcionamiento del sistema, puedes escoger el que más se te facilite.

Tabla 2 Tabla del siguiente estado

Estado actual	Salida	Entrada SW[0]	Estado siguiente
Off_all	000	0	On_all
On_all	111	0	Off_all
Off_all	000	1	Verde
On_all	111	1	Verde
verde	100	1	Amarillo
amarillo	010	1	Rojo
rojo	001	1	Verde
verde	100	0	Off_all
amarillo	010	0	Off_all
rojo	001	0	Off_all

3.2. Segundo paso: Traducir el diagrama de flujo en HDL. *Ver video*

Verilog 1 Módulo del semáforo

```
module semaforo(on_off, luzverde, luzamarilla, luzroja, clk);

//definición de entradas salidas
input on_off, clk;
output luzverde, luzamarilla, luzroja;

//DEFINICIÓN D ESTADOS
parameter on_all    =      3'b111;
parameter off_all   =      3'b000;
parameter verde     =      3'b100;
parameter amarillo  =      3'b010;
parameter rojo      =      3'b001;

reg [2:0] estado=off_all;

always@(posedge clk)
begin
       case (estado)
                     on_all:       if (on_off)
                                          estado<=verde;
                                          else
                                          estado<=off_all;
                     off_all:      if (on_off)
                                          estado<=verde;
                                          else
                                          estado<=on_all;
                     verde:        if (on_off)
                                          estado<=amarillo;
                                          else
                                          estado<=off_all;
                     amarillo:     if (on_off)
                                          estado<=rojo;
                                          else
                                          estado<=on_all;
                     rojo:         if (on_off)
                                          estado<=verde;
                                          else
                                          estado<=off_all;
                     default: estado<=verde;
              endcase
end

assign luzroja=estado[0];
assign luzamarilla=estado[1];
assign luzverde=estado[2];

endmodule
```

4. **Son consejos que te doy, porque Popeye el marino soy:**
    - Diseño pensando en Hardware
    - Escritura limpia y legible, de fácil interpretación
    - Dibujar diagrama de bloques
    - Dibujar diagrama de estados
    - Diseño modular, módulos que puedan compilarse mentalmente
    - Sincronizar las señales de las entradas de cada módulo
    - Diseño depurable
    - Haga simulación, si funciona en simulación probablemente funcione en la tarjeta, si no funciona en la simulación definitivamente no funcionará en ningún lado
    - Si no funciona a la primera: ríase, que de seguro es alguna bobada. A todos nos pasa.

5. Referencias

1. **Linn, Yair.** *Modular Reliable Desing.* 2008.

2. **Floyd, Thomas L.** *Fundamentos de Sistemas Digitales.* España : Prentice Hall, 2006.

3. **Chu, Pong P.** *FPGA Prototuping by Verilog Examples.* Canada : Jhon Wiley & Sons, 2008.

4. **Linn, Yair.** Diseño con FPGAs. [En línea] [Citado el: 16 de 11 de 2010.] http://sites.google.com/site/cursofpgasupbbga/.