02/12/20

UNIVERSIDAD POPULAR AUTÓNOMA DEL ESTADO DE PUEBLA

SISTEMAS EMBEBIDOS

MEC213-01

PROYECTO FINAL

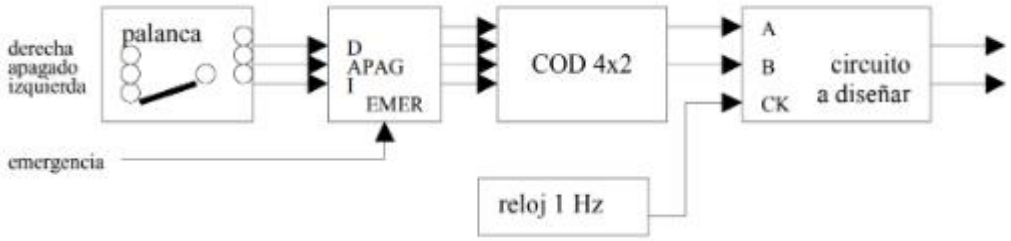
Sistema de Encendido de Intermitencia de un Coche Maquina de Estados de Moore

Profesor: Casimiro Gómez González

Alumnos:	Carrera
• AXEL ARRIOLA FONSECA	Ingeniería en Mecatrónica
• JOSE RICARDO ALVAREZ MERCADO	Ingeniería en Mecatrónica
• JORGE CISNEROS ORTEGA	Ingeniería en Mecatrónica

I. OBJETIVO

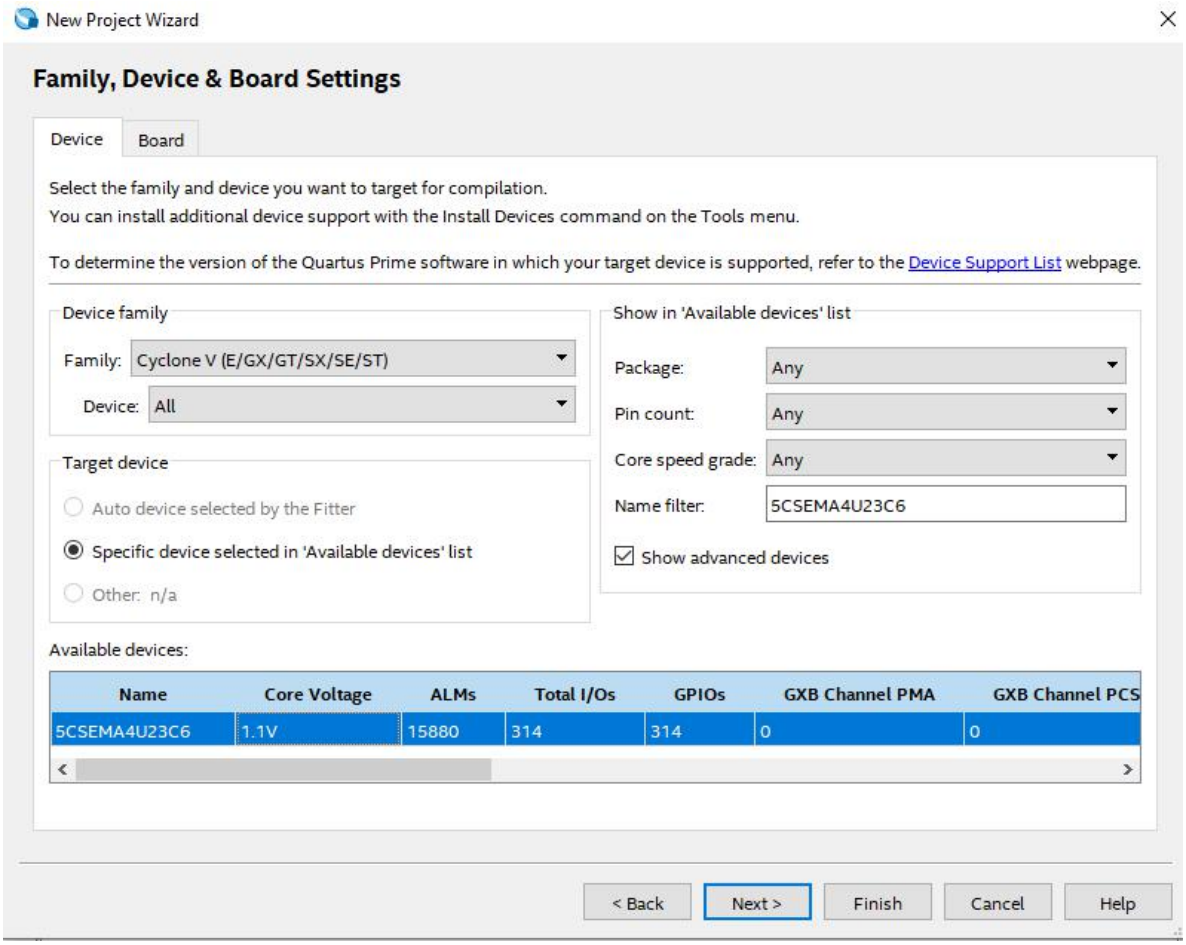
Realizar un circuito secuencial donde la palanca se coloque en la posición DERECHA, se deberá encender y apagar de forma intermitente la luz identificada como D, de forma síncrona con un reloj de 1 Hz. Cuando la palanca se coloque en la posición IZQUIERDA, se deberá encender y apagar de forma intermitente la luz identificada como I, de forma síncrona con un reloj de 1 Hz. Cuando la palanca se coloque en la posición central (APAGADO) no se encenderá ninguna luz. Cuando se active el interruptor de EMERGENCIA, se activarán ambas luces simultáneamente, y se desactivarán ambas de forma síncrona con el reloj,
independientemente de la posición de la palanca, es decir, la entrada de emergencia tiene prioridad absoluta.



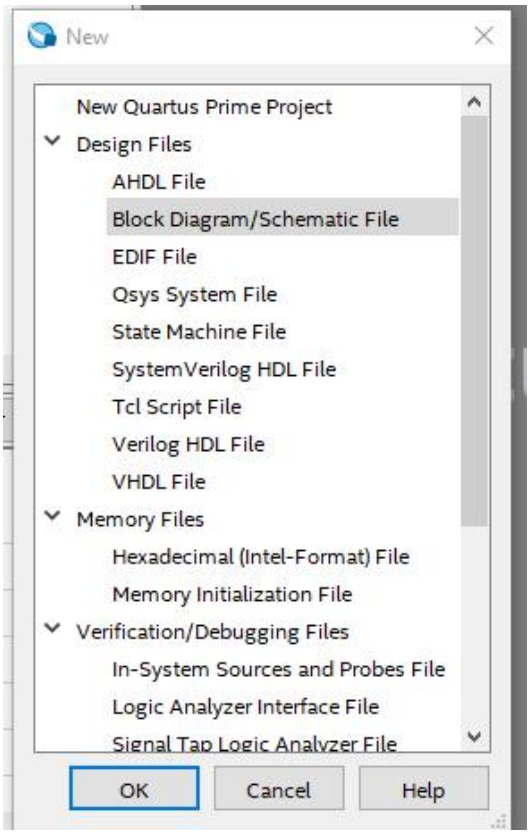
Procedimiento:

Intel Quartus Prime Lite Edition (20.1)

1. Dar click en New- Project-Wizard.
2. Te abrirá una sección para la creación de tu nuevo proyecto, en esta sección te pedirá el lugar donde colocarás el archivo y como lo nombraras, es importante que cada proyecto tenga una carpeta.
3. En este tipo de proyecto en específico utilizaremos la opción de proyecto vacío y no se agregan archivos existentes
4. En la sección de la “family, device & board settings” se colocara la informacion de la tarjeta que se utilizara que en este caso es una Deo-Nano-SoC Altera Cyclone® V SE con matrícula 5CSEMA4U23C6.



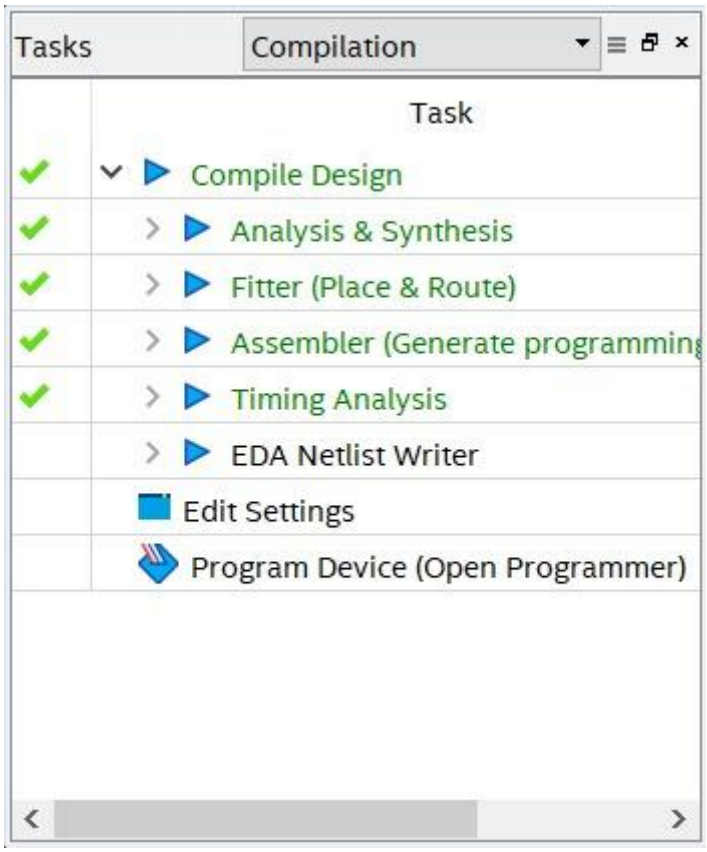
5. En siguiente paso de igual manera para este proyecto en específico no se coloca ninguna herramienta.
6. Para finalizar la creación del nuevo proyecto el programa quartus te genera un resumen de los aspectos y características de tu archivo.
7. Una vez creado el proyecto lo que prosigue es crear el programa o el circuito gráficamente, en nuestro caso se realizó el programa verilog por lo cual se irá a la pestaña de file/ new, en donde escogeremos “Verilog HDL file”



8. Nos aparecerá una pestaña de extensión .v ahí es donde crearemos el código de nuestro proyecto

9. Se realiza un nuevo programa creando nuevamente el proyecto y archivo verilog, una vez creado se escribe el código para la creación del programa

10. Compilamos el código y comprobamos que no tenga errores



11. Se genera el código con la ayuda de nuestra tabla de estados y diagrama, también el test bench que es una forma de representación de pasarlo a la tarjeta física.

Estados	**Trancisión**								**Salidas (D,I)**	
Entradas (ADI,E)	**00,0**	**01,0**	**10,0**	**11,0**	**00,1**	**01,1**	**10,1**	**11,1**	**0**	**1**
(Apagado) S0	S0	S1	S2	S0	S3	S3	S3	S3	0	0
(Derecha) S1	S0	S1	S2	S0	S3	S3	S3	S3	1	0
(Izquierda) S2	S0	S1	S2	S0	S3	S3	S3	S3	0	1
(Emergencia S3	S0	S1	S2	S0	S3	S3	S3	S3	1	1

Tabla de estados del problema

Entradas:

- Palanca de direcciones con 3 combinaciones (ADI) se conforma por 2 bits
  00: Apagado 01: Derecha 10: Izquierda 11:Basura
- Emergencia (E) 1 bit

Salidas:

- Luz intermitente derecha (D) 1 bit
- Luz intermitentes izquierda (I) 1 bit

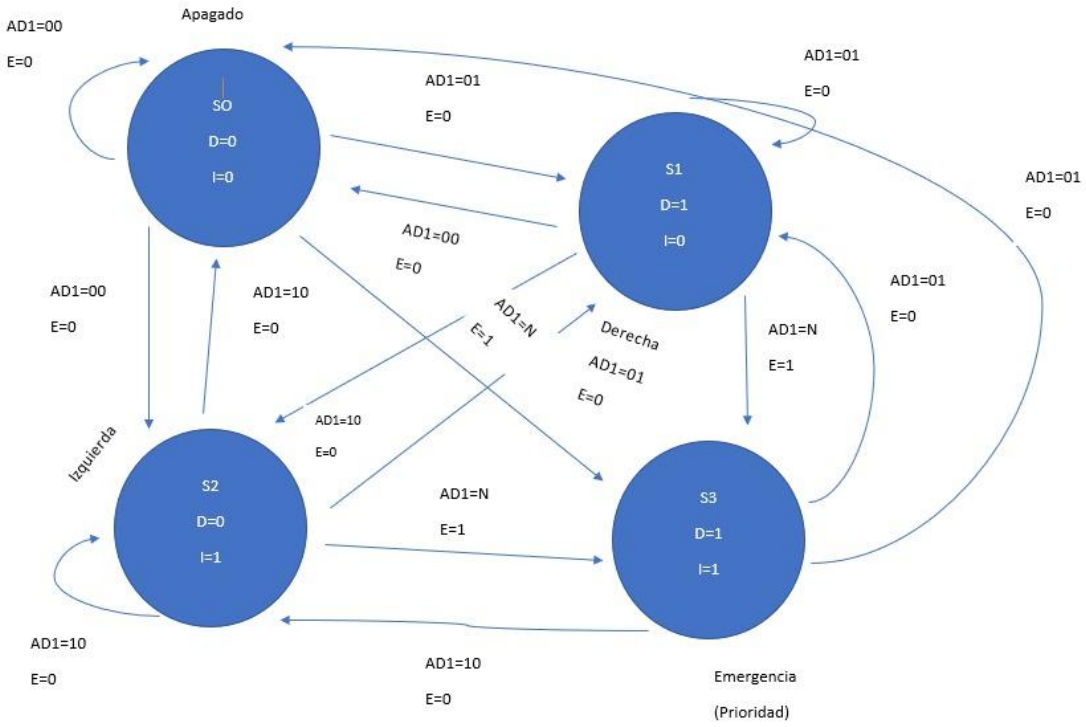
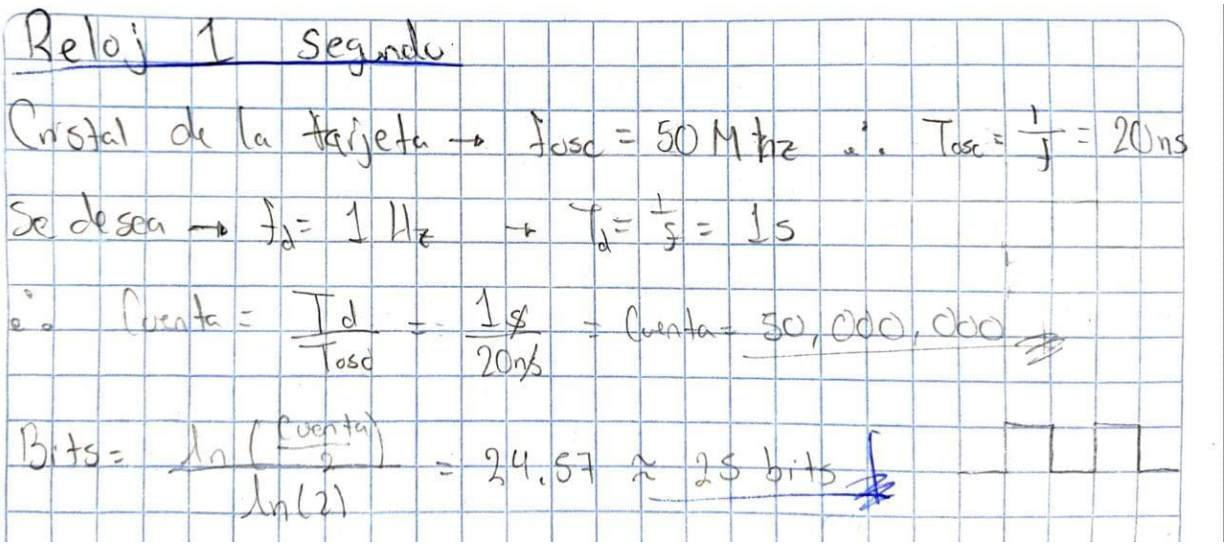


Diagrama de estados del problema

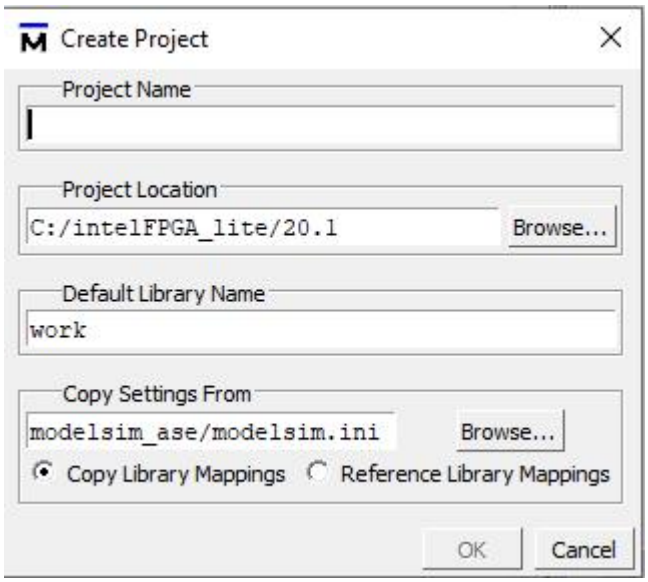


Calculos para cuenta y bits de reloj de 1 Hz (1 segundo)

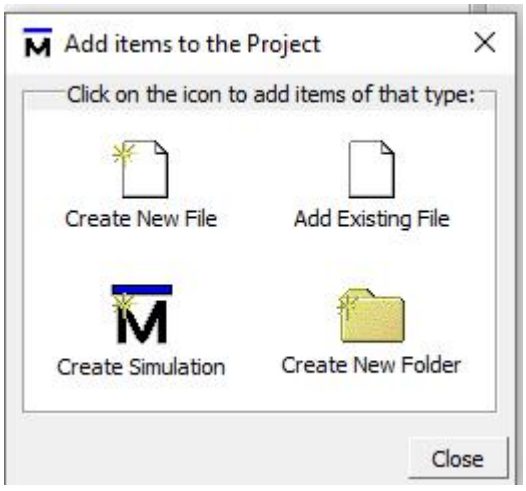
ModelSim – Intel FPGA Started Edition (Quartus 20.1)

12. Antes de realizar la simulación, en la carpeta de nuestro proyecto crearemos una carpeta con nombre de la simulación donde copiaremos el archivo .v de nuestro proyecto

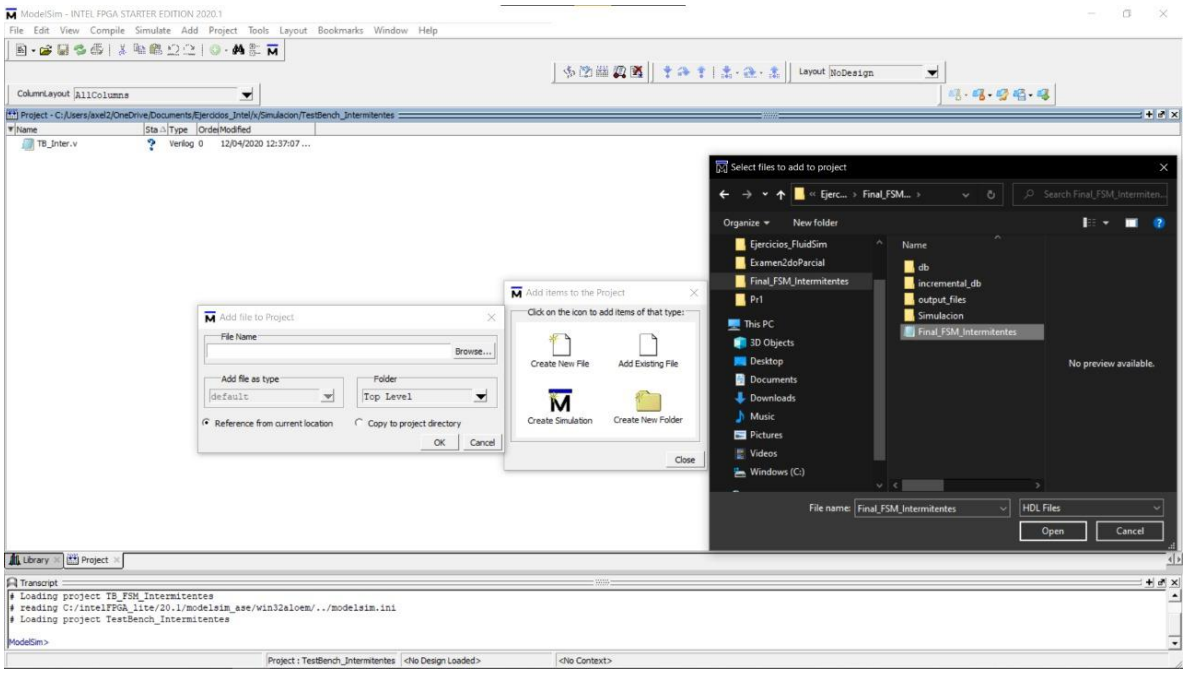
13. Se abrirá el programa llamado Model-Sim - Intel FPGA Starter Edition Model Technology donde generaremos un proyecto por lo cual iremos a la pestaña file/ new/ project donde colocaremos el nombre y ubicación de nuestro proyecto



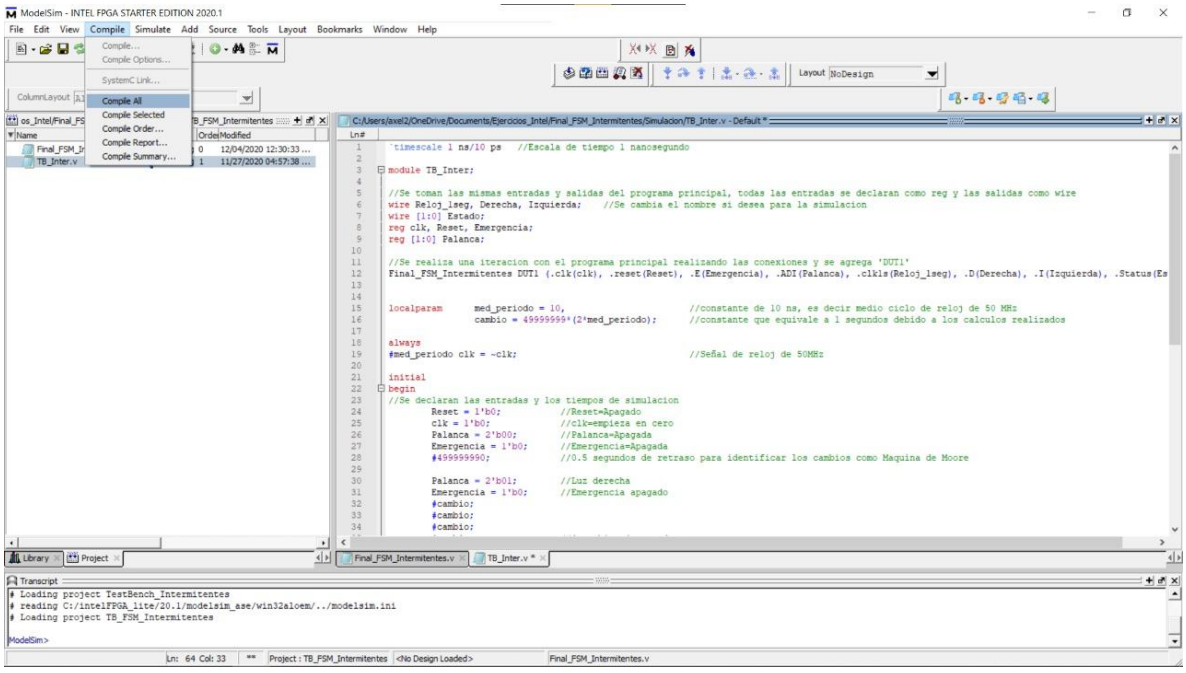
14. Una vez creado nuestro nuevo proyecto nos saldrá otra ventana donde agregaremos nuestro código con las extensión .v la



15. Al agregar nuestro código lo siguiente es crear un nuevo documento donde crearemos nuestro testbench.

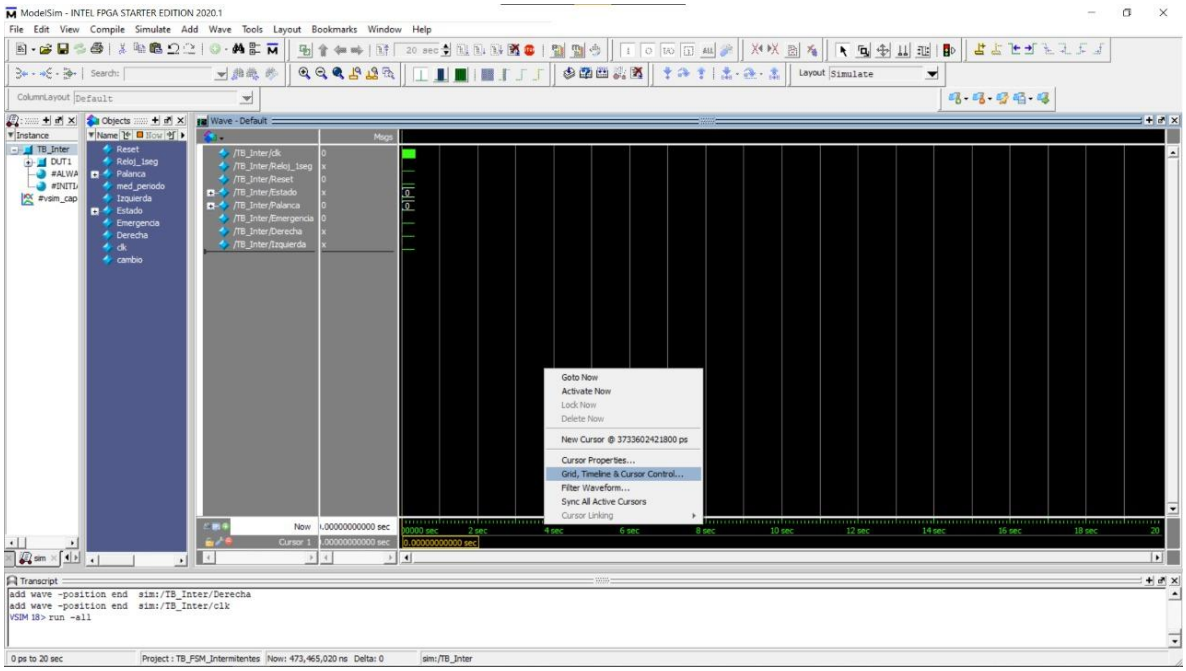


16. Una vez terminados los dos programas se realiza la compilación por lo cual iremos a la pestaña compile/compile all, cuando finalice de compilar nos debe mostrar unas palomitas en color vede en cada programa lo cual nos indica que está listo para la simulación.

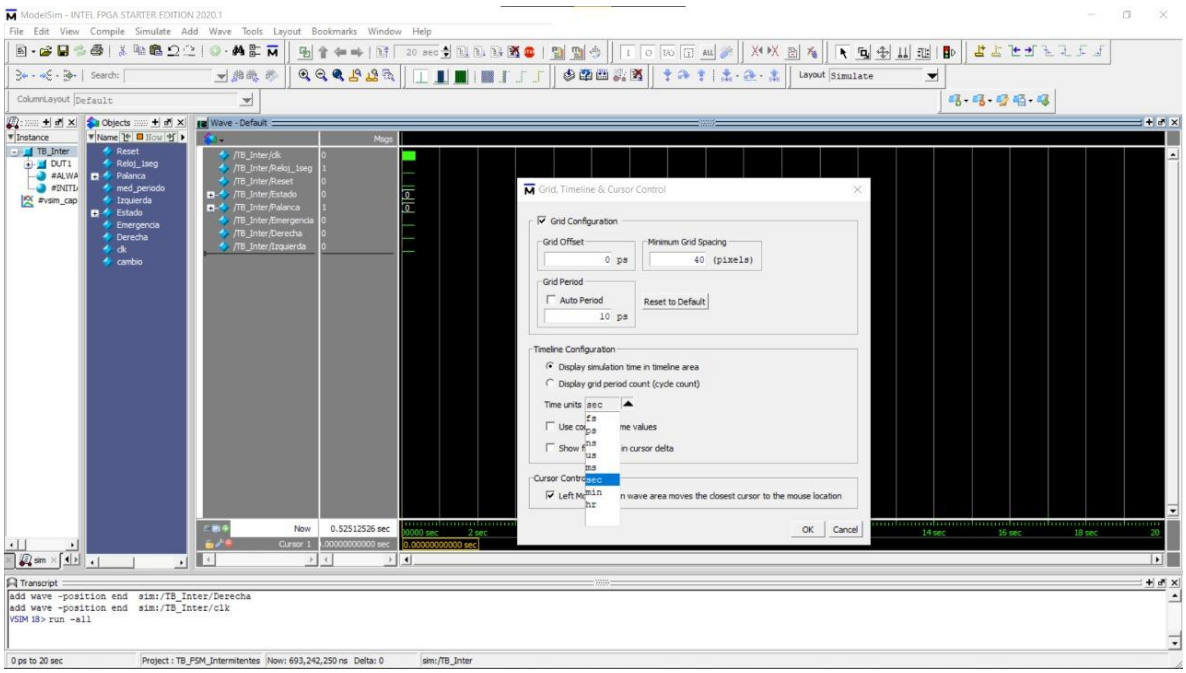


17. Iremos a la pestaña de Simulate le daremos en start simulate al hacer esto nos abrira una ventana nueva donde tendremos que darle en la cruz al lado de Work para buscar nuestro programa testbench y le daremos ok.

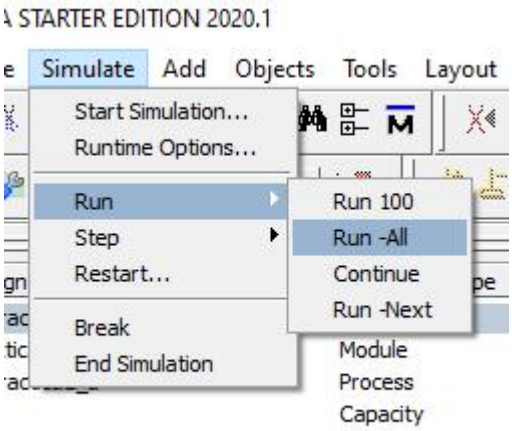
18. Una ves ya en el simulador lo único que se realiza es en cada variable que se desea ver se le da click derecho Add wave



19. De igual forma se le da click derecho y Add wave

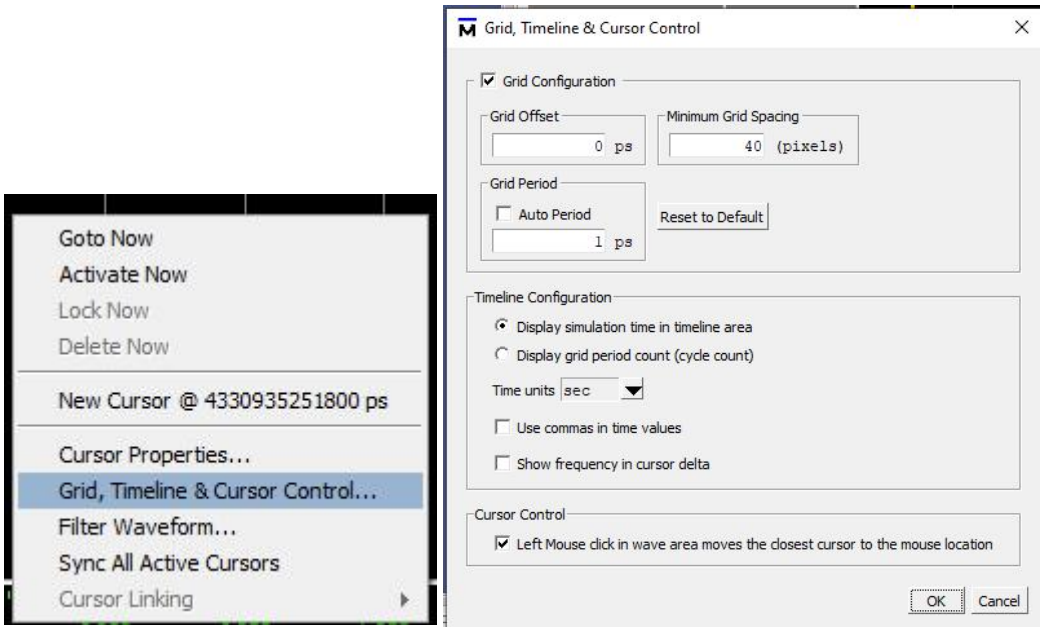


20. Al terminar de agregar las variables le daremos en la pestaña de Simulate/Run/Run all



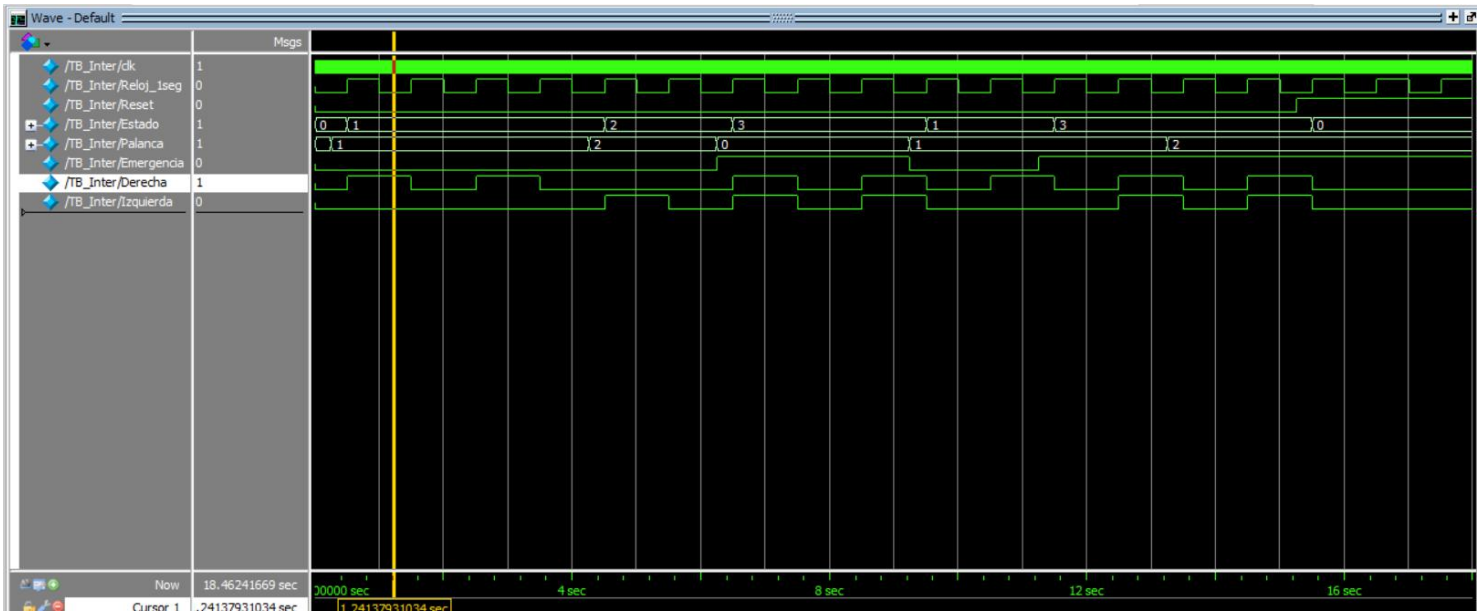
.

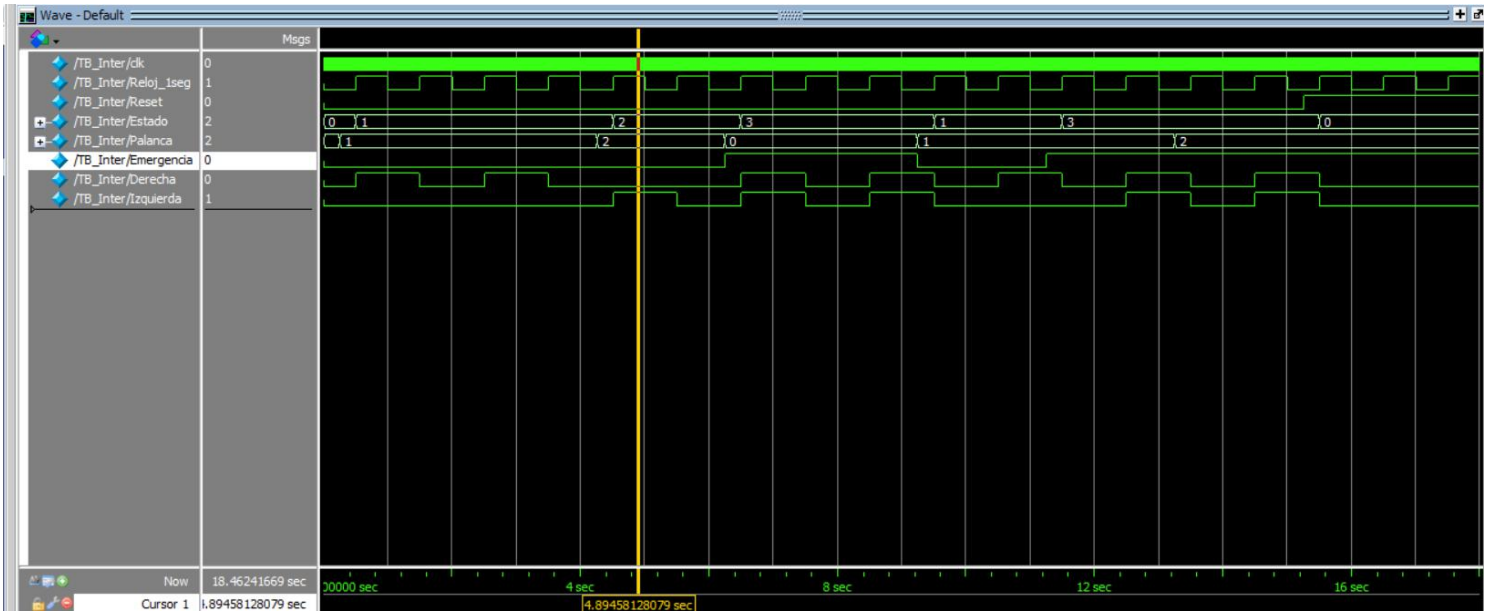
21. Para poder ver la señal se debe realizar cambios en la escala por lo cual nos iremos a la seccion donde dicen los segundos o nano segundos como este, botón derecho y escogemos “Grid, timeline& cursor….” Y en la sección de unidades de tiempo se escogerá en este caso segundos

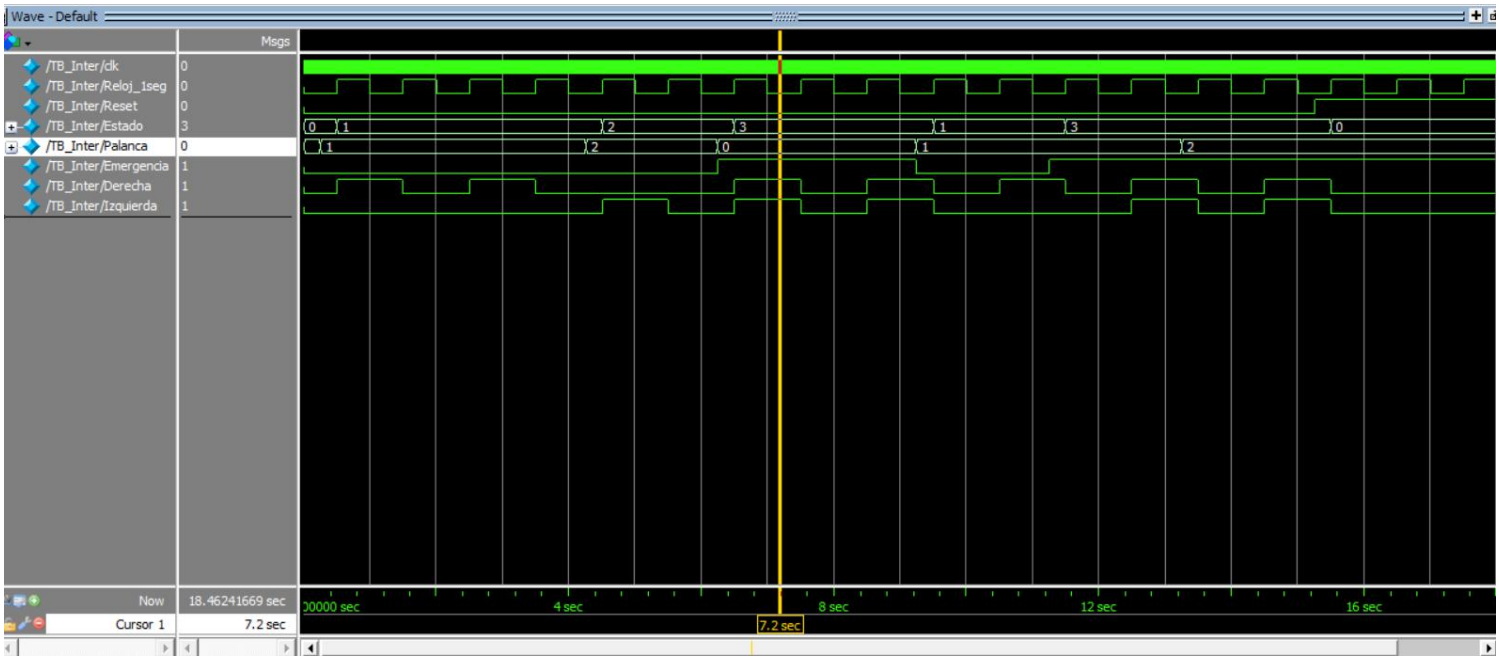


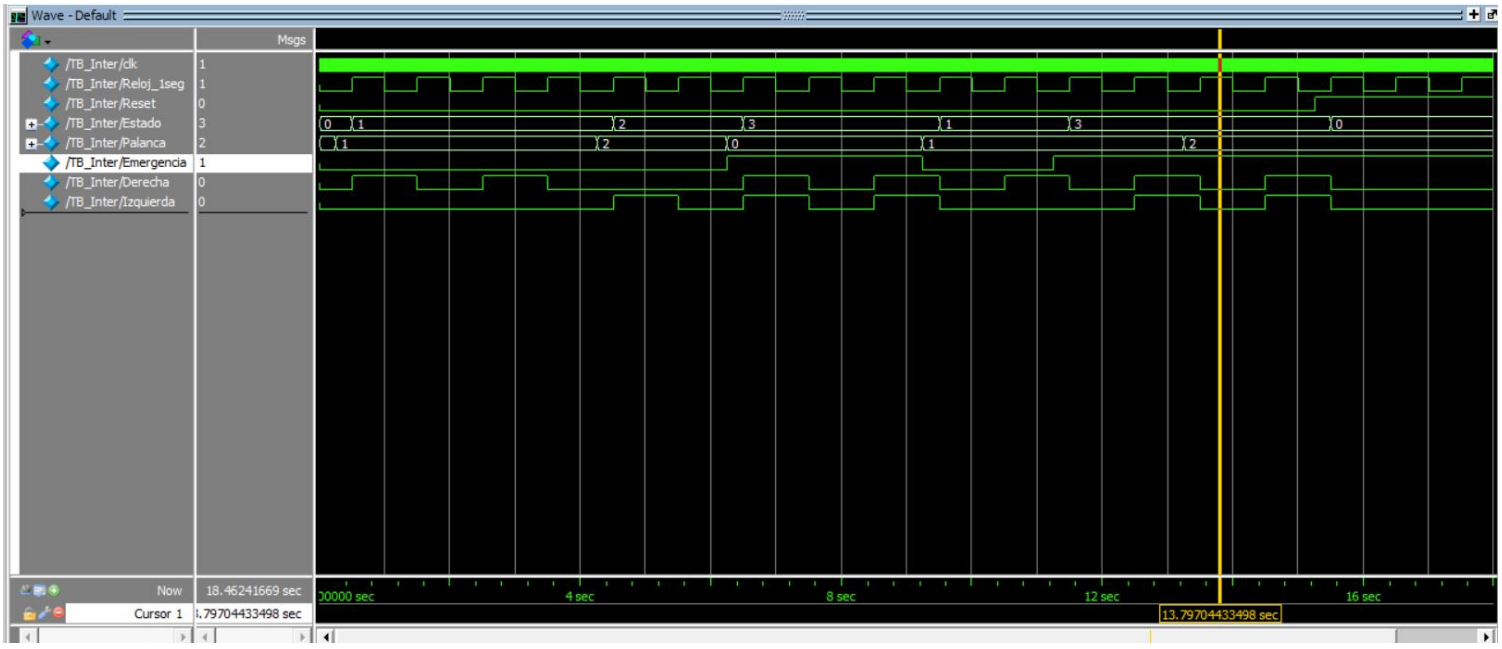
22. Para determinar también el tamaño hasta donde desees ver se debe dar botón derecho sobre la pantalla negra y escoger “zoom range” y ahí variar el tiempo de finalización

23. La simulación tardara por lo cual hay que esperar






Wave - Default
Msgs
/TB_Inter/clk
/TB_Inter/Reloj_1seg
/TB_Inter/Reset
/TB_Inter/Estado
/TB_Inter/Palanca
/TB_Inter/Emergencia
/TB_Inter/Derecha
/TB_Inter/Izquierda
Now
18.46241669 sec
Cursor 1
7.2 sec
4 sec
8 sec
12 sec
16 sec
7.2 sec


Wave - Default
Msgs
/TB_Inter/clk
/TB_Inter/Reloj_1seg
/TB_Inter/Reset
/TB_Inter/Estado
/TB_Inter/Palanca
/TB_Inter/Emergencia
/TB_Inter/Derecha
/TB_Inter/Izquierda
Now
18.46241669 sec
Cursor 1
4 sec
8 sec
12 sec
16 sec
13.79704433498 sec

Código principal

//PROGRAMA PRINCIPAL

```
module Final_FSM_Intermitentes (input clk, input reset, input E, input [1:0] ADI, output reg clk1s, output reg D, output reg I, output reg [1:0] Status);


/*
```

PROGRAMA DE INTERMITENTES CON SEÑAL DE RELOJ DE 1 SEGUNDO RESUELTO POR MAQUINA DE ESTADOS DE MOORE (VERILOG)

Entradas: clk: Reloj de tarjeta (50 MHz)

reset: Botón de reinicio

ADI: Palanca que indica Apagado (00) , Derecha (10), Izquierda (01), Nada (11)

E: Botón de emergencia (Prioridad)

```
Salidas:                clk1s: Reloj de 1 segundo (1 Hz) para la intermitencia de las salidas y
estados
                                D: Luz derecha se enciende de manera intermitente con
clk1s
                                I: Luz izquierda se enciende de manera intermitente con
clk1s
                                Status: Indica en que estado se encuentra el sistema
*/


localparam      S0 = 2'b00,     //Estado 0 (Apagado), ambas luces apagadas
                                S1 = 2'b01,     //Estado 1 (Derecha). luz derecha
encendida intermitentemente
                                S2 = 2'b10, //Estado 2 (Izquierda), luz izquierda encendida
intermitentemente
                                S3 = 2'b11; //Estado 3 (Emergencia), ambas luces
encendidas intermitentemente



reg [1:0] act_st;       //Estado actual
reg [1:0] nxt_st;       //Estado siguiente
reg [24:0] count;       //Cuenta para reloj 1 seg (25 bits para cumplir el conteo)
reg [1:0] LED_count;    //Cuenta para la intermitencia de luces
reg Intermitente;               //Variables para intermitencia (Derecha e Izquierda)


//SEÑAL DE RELOJ DE 1 SEGUNDO
always @ (posedge clk)
begin
count = count+1;
```

```
        if (count == 25'd24999999) //El reloj de la tarjeta debe contar hasta 25,000,000 para cumplir medio ciclo de 1 seg
                begin
                        count = 25'd0;
                        clk1s = ~clk1s;             //Después de cumplir la cuenta, se invierte para cumplir la otra mitad del periodo y formar el segundo completo
                end
end


//BLOQUE DE TRANSICIÓN
always @(posedge clk1s)
begin
        if (reset==1'b1)            //Si reset=1 entonces el estado actual se vuelve estado inicial
                begin
                        act_st = S0;
                end
        else                                            //Si reset=0 entonces el estado actual se vuelve el estado siguiente
                begin
                        act_st = nxt_st;
                end
end


//BLOQUE DE ESTADO SIGUIENTE
always @(*)                     //Siempre se ejecuta
begin
nxt_st = act_st;
```

```
        case (act_st)  //Case para las condiciones de transicion de estados segun cada caso (Tabla de estados)

                S0:     begin
                                if (ADI==2'b00 & E==0)  //Si Palanca=Apagado y Emergencia=Apagado
                                        begin
                                                nxt_st = S0;
        //S0 se dirige a S0
                                        end

                                if (ADI==2'b01 & E==0)  //Si Palanca=Derecha y Emergencia=Apagado
                                        begin
                                                nxt_st = S1;
        //S0 se dirige a S1
                                        end

                                if (ADI==2'b10 & E==0)  //Si Palanca=Izquierda y Emergencia=Apagado
                                        begin
                                                nxt_st = S2;
        //S0 se dirige a S2
                                        end

                                if (ADI==2'b11 & E==0)  //Si Palanca=Basura(Nada) y Emergencia=Apagado
                                        begin
                                                nxt_st = S0;
        //S0 se dirige a S0
                                        end
```

```
                                        if (ADI==2'b00 & E==1)  //Si
Palanca=Apagado y Emergencia=Encendido
                                            begin
                                                nxt_st = S3;
        //S0 se dirige a S3
                                            end

                                        if (ADI==2'b01 & E==1)  //Si
Palanca=Derecha y Emergencia=Encendido
                                            begin
                                                nxt_st = S3;
        //S0 se dirige a S3
                                            end

                                        if (ADI==2'b10 & E==1)  //Si
Palanca=Izquierda y Emergencia=Encendido
                                            begin
                                                nxt_st = S3;
        //S0 se dirige a S3
                                            end

                                        if (ADI==2'b11 & E==1)  //Si
Palanca=Basura(Nada) y Emergencia=Encendido
                                            begin
                                                nxt_st = S3;
        //S0 se dirige a S3
                                            end
                                    end
```

```
S1:     begin
//Son las mismas condiciones para cada estado
                    if (ADI==2'b00 & E==0)
                        begin
                            nxt_st = S0;
                        end

                    if (ADI==2'b01 & E==0)
                        begin
                            nxt_st = S1;
                        end

                    if (ADI==2'b10 & E==0)
                        begin
                            nxt_st = S2;
                        end

                    if (ADI==2'b11 & E==0)
                        begin
                            nxt_st = S0;
                        end

                    if (ADI==2'b00 & E==1)
                        begin
                            nxt_st = S3;
                        end

                    if (ADI==2'b01 & E==1)
                        begin
```

```
                    nxt_st = S3;
                end

            if (ADI==2'b10 & E==1)
                begin
                    nxt_st = S3;
                end

            if (ADI==2'b11 & E==1)
                begin
                    nxt_st = S3;
                end
        end

S2:     begin
            if (ADI==2'b00 & E==0)
                begin
                    nxt_st = S0;
                end

            if (ADI==2'b01 & E==0)
                begin
                    nxt_st = S1;
                end

            if (ADI==2'b10 & E==0)
                begin
                    nxt_st = S2;
```

```
                end

            if (ADI==2'b11 & E==0)
                begin
                    nxt_st = S0;
                end

            if (ADI==2'b00 & E==1)
                begin
                    nxt_st = S3;
                end

            if (ADI==2'b01 & E==1)
                begin
                    nxt_st = S3;
                end

            if (ADI==2'b10 & E==1)
                begin
                    nxt_st = S3;
                end

            if (ADI==2'b11 & E==1)
                begin
                    nxt_st = S3;
                end
        end
```

```
S3:     begin
                if (ADI==2'b00 & E==0)
                        begin
                                nxt_st = S0;
                        end

                if (ADI==2'b01 & E==0)
                        begin
                                nxt_st = S1;
                        end

                if (ADI==2'b10 & E==0)
                        begin
                                nxt_st = S2;
                        end

                if (ADI==2'b11 & E==0)
                        begin
                                nxt_st = S0;
                        end

                if (ADI==2'b00 & E==1)
                        begin
                                nxt_st = S3;
                        end

                if (ADI==2'b01 & E==1)
                        begin
                                nxt_st = S3;
```

```
                    end

                if (ADI==2'b10 & E==1)
                    begin
                        nxt_st = S3;
                    end

                if (ADI==2'b11 & E==1)
                    begin
                        nxt_st = S3;
                    end
            end

        endcase
end

//BLOQUE DE INTERMITENCIA
always @(posedge clk1s)                 //Depende de clk1s ya que la intermitencia debe
estar sincronizada con este
begin
LED_count = LED_count+1;
Intermitente = 1'b0;

            if (LED_count==2'b01)       //Cuando LED_count=1 entonces la
intermitente se prende
                begin
                    Intermitente = 1'b1;
                end
```

```
                    if (LED_count==2'b10)          //Cuando LED_count=2 entonces la
intermitente se apaga
                        begin
                            Intermitente = 1'b0;
                            LED_count = 2'b00;      //Se reinicia para seguir
prendiendo y apagando
                        end
end



//BLOQUE DE SALIDA
always @(*)
begin
D = 1'b0;
I = 1'b0;
Status = 2'b00;

    case (act_st)          //Case para las salidas en cada estado, estas no dependen de las
entradas ya que es Maquina de Moore

                        S0:     begin
    //En S0(Apagado), todo se encuentra apagado
                                    D = 1'b0;
                                    I = 1'b0;
                                    Status = S0;          //Indica el
estado actual
                                end

                        S1:     begin
    //En S1(Derecha), la salida 'Derecha' realiza intermitencia
```

```
                                        D = Intermitente;       //Se incorpora la salida de la intermitencia realizada en el bloque anterior
                                        I = 1'b0;
                                        Status = S1;            //Indica el estado actual
                                    end

                                S2:     begin
        //En S2(Izquierda), la salida 'Izquierda' realiza intermitencia
                                        D = 1'b0;
                                        I = Intermitente;
                                        Status = S2;            //Indica el estado actual
                                    end

                                S3:     begin
        //En S3(Emergencia), ambas salidas 'derecha' e 'izquierda' realizan intermitencia simultaneamente
                                        D = Intermitente;
                                        I = Intermitente;
                                        Status = S3;            //Indica el estado actual
                                    end

        endcase
end

//BLOQUE DE INICIALIZACIÓN
initial             //Se inicializan los valores para la simulacion con ModelSim (TestBench)
```

```
begin
        count = 25'd0;
        LED_count = 2'b00;
        D = 1'b0;
        I = 1'b0;
        Intermitente = 1'b0;
        Status = 2'b00;
        act_st = 2'b00;
        nxt_st = 2'b00;
        clk1s = 1'b0;
end

endmodule
```

Código TESTBENCH

```
`timescale 1 ns/10 ps     //Escala de tiempo 1 nanosegundo

module TB_Inter;

//Se toman las mismas entradas y salidas del programa principal, todas las entradas se declaran como reg y las salidas como wire
wire Reloj_1seg, Derecha, Izquierda;       //Se cambia el nombre si desea para la simulacion
wire [1:0] Estado;
reg clk, Reset, Emergencia;
reg [1:0] Palanca;

//Se realiza una iteracion con el programa principal realizando las conexiones y se agrega 'DUT1'
Final_FSM_Intermitentes DUT1
(.clk(clk), .reset(Reset), .E(Emergencia), .ADI(Palanca), .clk1s(Reloj_1seg), .D(Derecha), .I(Izquierda)
, .Status(Estado));
```

```
localparam        med_periodo = 10,                          //constante de 10 ns, es decir medio ciclo de reloj de 50 MHz

                  cambio = 49999999*(2*med_periodo);  //constante que equivale a 1 segundos debido a los calculos realizados


always

#med_periodo clk = ~clk;                                     //Señal de reloj de 50MHz


initial

begin

//Se declaran las entradas y los tiempos de simulacion

        Reset = 1'b0;           //Reset=Apagado

        clk = 1'b0;             //clk=empieza en cero

        Palanca = 2'b00;        //Palanca=Apagada

        Emergencia = 1'b0;      //Emergencia=Apagada

        #249999990;             //0.5 segundos de retraso para identificar los cambios como Maquina de Moore


        Palanca = 2'b01;        //Luz derecha

        Emergencia = 1'b0;      //Emergencia apagado

        #cambio;

        #cambio;

        #cambio;

        #cambio;                //4 cambios=4 segundos


        Palanca = 2'b10;        //Luz izquierda

        Emergencia = 1'b0;      //Emergencia apagado

        #cambio;
```

```
        #cambio;                //2 segundos


        Palanca = 2'b00;        //Ambas luces apagadas
        Emergencia = 1'b1;      //Emergencia encendido
        #cambio;
        #cambio;
        #cambio;                //3 segundos

        Palanca = 2'b01;        //Luz derecha
        Emergencia = 1'b0;      //Emergencia apagado
        #cambio;
        #cambio;                //2 segundos

        Palanca = 2'b01;        //Luz derecha
        Emergencia = 1'b1;      //Emergencia encendido
        #cambio;
        #cambio;                //2 segundos

        Palanca = 2'b10;        //Luz izquierda
        Emergencia = 1'b1;      //Emergencia encendido
        #cambio;
        #cambio;                //2 segundos

        Reset = 1'b1;           //Reset=encendido
end

endmodule
```

CONCLUSIONES

Se logro realizar la tabla de estado sin problema, la máquina de estados nos ayuda mucho a representar un diagrama de estado sin problema, nos ayuda a relacionar las entras y salidas de un conjunto de estados y nos ayude a determinar cada instante de cada estado.