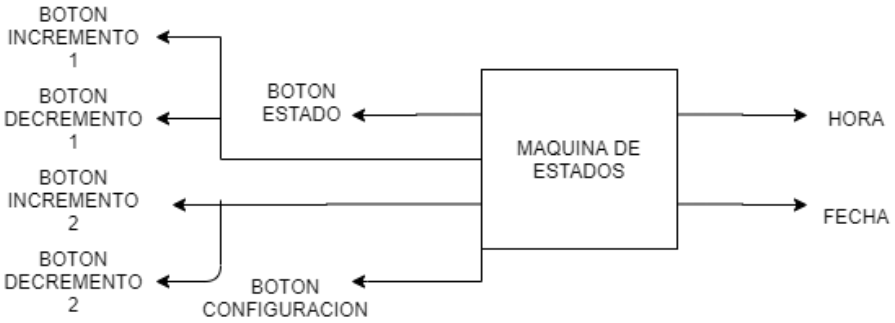
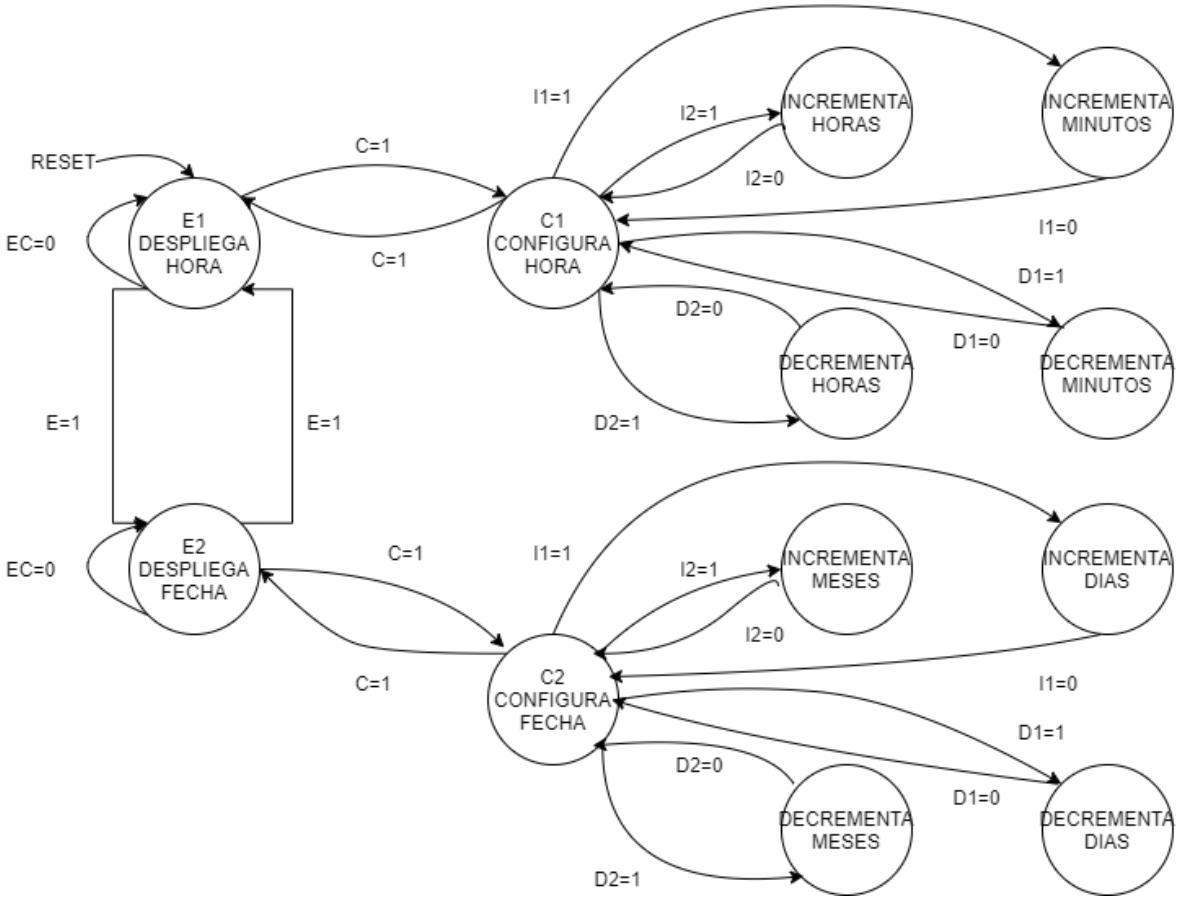
Máquina de estados finitos: Reloj programable





S0=DESPLIEGA HORA

S1=CONFIGURA HORA

S2=INCREMENTA MINUTOS

S3=INCREMENTA HORAS

S4=DECREMENTA MINUTOS

S5=DECREMENTA HORAS

S6=DESPLIEGA FECHA

S7=CONFIGURA FECHA

S8=INCREMENTA DIAS

S9=INCREMENTA MESES

S10=DECREMENTA DIAS

S11=DECREMENTA MESES

ESTADO ACTUAL	E	C	I1	I2	D1	D2	SIGUIENTE ESTADO
S0	0	0	X	X	X	X	S0
S0	0	1	X	X	X	X	S1
S0	1	0	X	X	X	X	S6
S1	X	1	0	0	0	0	S0
S1	X	0	1	0	0	0	S2
S1	X	0	0	1	0	0	S3
S1	X	0	0	0	1	0	S4
S1	X	0	0	0	0	1	S5
S2	X	X	0	X	X	X	S1
S2	X	X	1	X	X	X	S2
S3	X	X	X	0	X	X	S1
S3	X	X	X	1	X	X	S3
S4	X	X	0	X	0	X	S1
S4	X	X	1	X	1	X	S4
S5	X	X	0	X	X	0	S1
S5	X	X	1	X	X	1	S5
S6	0	0	X	X	X	X	S6
S6	0	1	X	X	X	X	S7
S6	1	0	X	X	X	X	S0
S7	X	1	0	0	0	0	S6
S7	X	0	1	0	0	0	S8
S7	X	0	0	1	0	0	S9
S7	X	0	0	0	1	0	S10
S7	X	0	0	0	0	1	S11
S8	X	X	0	X	X	X	S7
S8	X	X	1	X	X	X	S8
S9	X	X	X	0	X	X	S7
S9	X	X	X	1	X	X	S9
S10	X	X	0	X	0	X	S7
S10	X	X	1	X	1	X	S10
S11	X	X	0	X	X	0	S7
S11	X	X	1	X	X	1	S11

Bits de salida: EI1I2D1D2

E= ESTADO (0 hora, 1 fecha)

I1=INCREMENTO (0 no cambia, 1 incrementa)

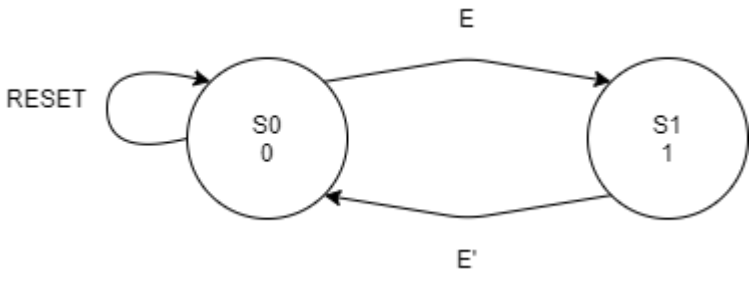
I2=INCREMENTO (0 no cambia, 1 incrementa)

D1=DECREMENTO (0 no cambia, 1 incrementa)

D2=DECREMENTO (0 no cambia, 1 incrementa)

DIAGRAMAS FACTORIZADOS

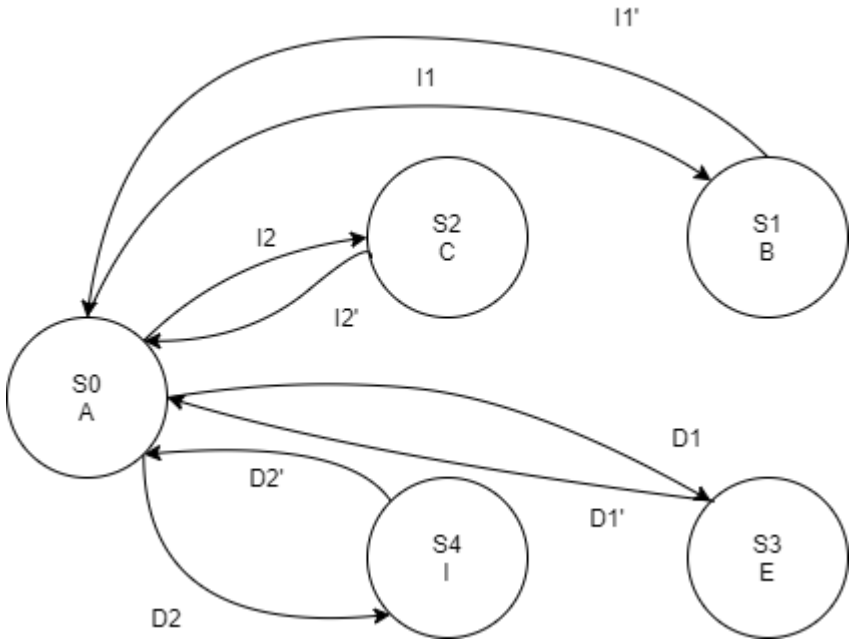
Diagrama de estados



S0 = Reloj

S1 = Fecha

Diagrama de Salidas



S0 = Configuración

S1 = Incrementa 1

S2 = Incrementa 2

S3 = Decrementa 1

S4 = Decrementa 2

Diagramas de estado

Tabla estados

ESTADO	E	FUTURO
S0	0	S0
S0	1	S1
S1	0	S0
S1	1	S1

Tabla salidas

ESTADO	SALIDA
S0	0
S1	1

Tabla transición codificada

E	S0	S0’
0	0	0
0	1	1
1	0	0
1	1	1

Ecuación dada por Logic Friday

S00 = E’ S0 + E S0 ;

Diagramas salida

Tabla Estados

ESTADO	I1	I2	D1	D2	FUTURO
S0	0	0	0	0	S0
S0	0	0	0	1	S4
S0	0	0	1	0	S3
S0	0	1	0	0	S2
S0	1	0	0	0	S1
S1	1	X	X	X	S1
S1	0	X	X	X	S0
S2	X	1	X	X	S2
S2	X	0	X	X	S0
S3	X	X	1	X	S3
S3	X	X	0	X	S0
S4	X	X	X	1	S4
S4	X	X	X	0	S0

Tabla salidas

ESTADO	SALIDA
S0	A
S1	B
S2	C
S3	E
S4	I

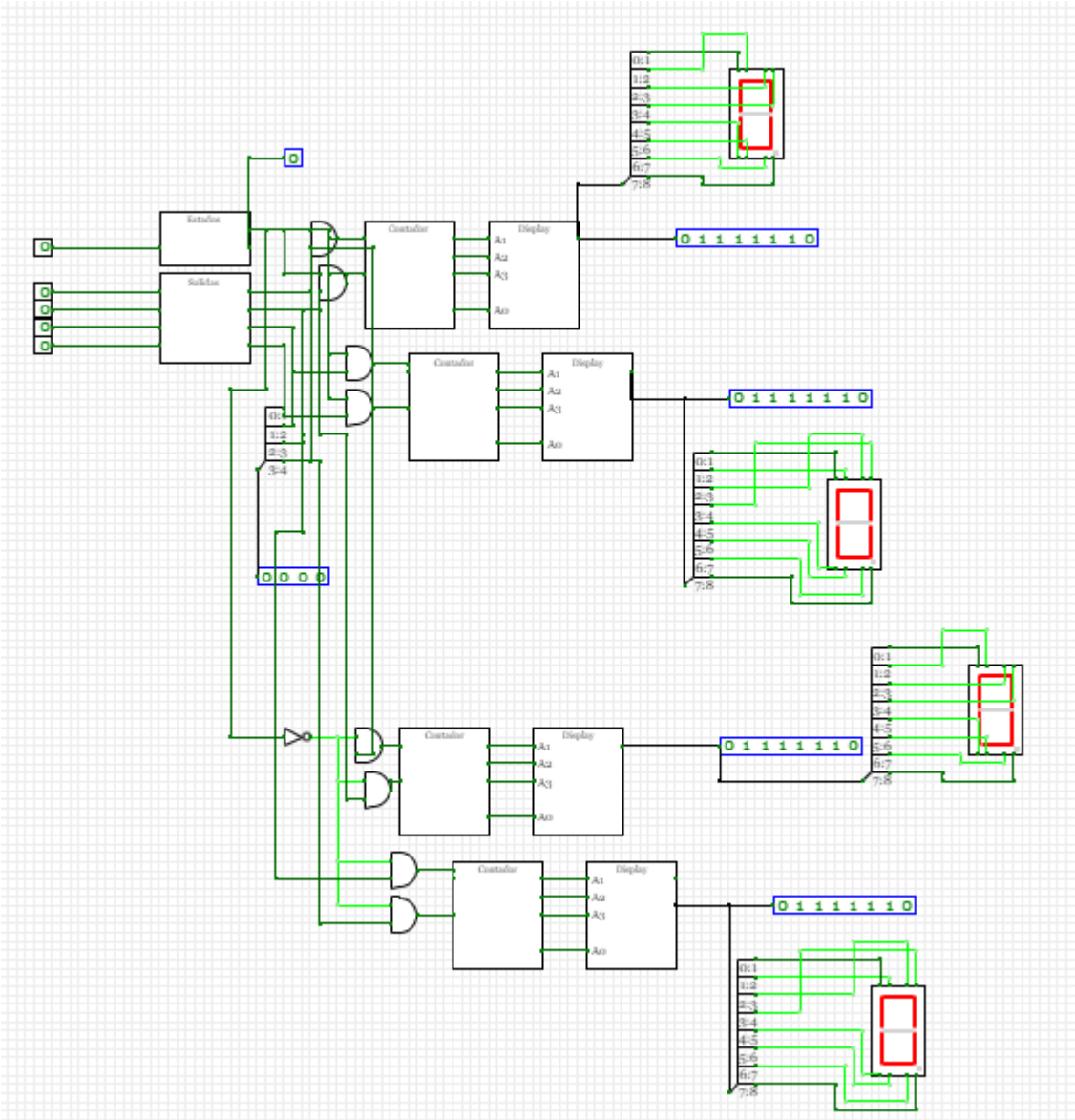
Tabla de transición codificada reducida en LF

I1	I2	D1	D2	S0	S1	S2	S3	S4	=>	S00	S11	S22	S33	S44
0	0	0	0	0	0	0	0	0		1				
0	0	0	0	1	0	0	0	0		1				
0	0	0	1	0	0	0	0	0						1
0	0	0	1	0	0	0	0	1						1
0	0	0	1	1	0	0	0	0						1
0	0	1	0	0	0	0	0	0					1	
0	0	1	0	0	0	0	1	0					1	
0	0	1	0	1	0	0	0	0					1	
0	1	0	0	0	0	0	0	0				1		
0	1	0	0	0	0	1	0	0				1		
0	1	0	0	1	0	0	0	0				1		
1	0	0	0	0	0	0	0	0			1			
1	0	0	0	0	1	0	0	0			1			
1	0	0	0	1	0	0	0	0			1			

Ecuaciones

```
S00 = I1' I2' D1' D2' S0' S1' S2'
S3' S4' + I1' I2' D1' D2' S0 S1'
S2' S3' S4';

S11 = I1 I2' D1' D2' S0' S1' S2'
S3' S4' + I1 I2' D1' D2' S0' S1
S2' S3' S4' + I1 I2' D1' D2' S0
S1' S2' S3' S4';

S22 = I1' I2 D1' D2' S0' S1' S2'
S3' S4' + I1' I2 D1' D2' S0' S1'
S2 S3' S4' + I1' I2 D1' D2' S0
S1' S2' S3' S4';

S33 = I1' I2' D1 D2' S0' S1' S2'
S3' S4' + I1' I2' D1 D2' S0' S1'
S2' S3 S4' + I1' I2' D1 D2' S0
S1' S2' S3' S4';

S44 = I1' I2' D1' D2 S0' S1' S2'
S3' S4' + I1' I2' D1' D2 S0' S1'
S2' S3' S4 + I1' I2' D1' D2 S0
S1' S2' S3' S4';
```

Finalizando en circuitverse



Adicionalmente a mi maquina de estados propuesta, era necesario como requisito poder demostrar la hora y la fecha. Por eso es que adicionalmente tuve que buscar como hacer el contador y como convertir de binario a hexadecimal. Por ese motivo; dado que son 2 funciones bastante comunes, es que busque en circuitverse los circuitos que mejor se adaptaran a mi necesidad.

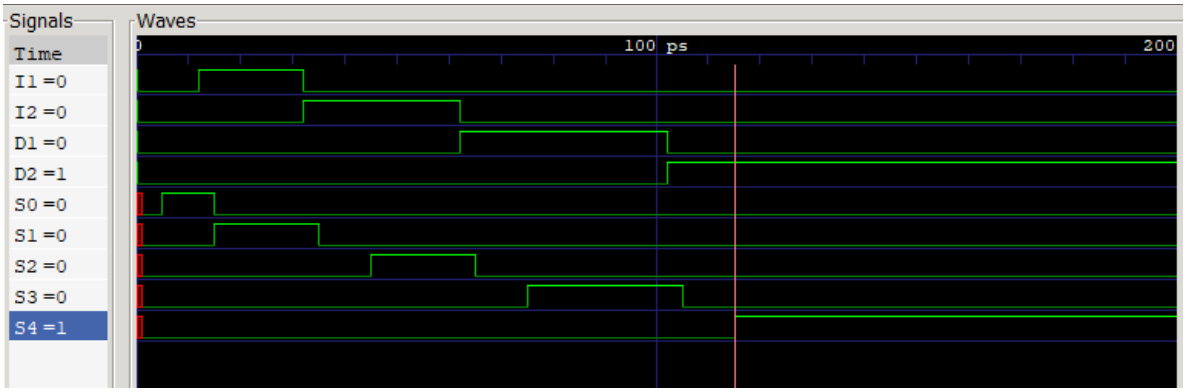
Circuito binario a hexadecimal https://circuitverse.org/users/29243/projects/megan-68533171-1e6c-44fd-ae88-d74eb0581ed1

Circuito de contador y reinicio binario https://circuitverse.org/users/29243/projects/rz-4-bit-binary-sync-counter-a629ff94-f625-4ece-a0e3-6fd160c5d91b

A estos 2 circuitos se les modifico las entradas y salidas para adaptar a mi circuito y con esto finalizar el proyecto para la fase de circuitverse.

Verilog

Como ultima fase del proyecto es comprobar que nuestro circuito puede ser simulado en un lenguaje de programación como lo es Verilog.




```
Salidas
Clk | reset | I1 | I2 | D1 | D2 | Sf | S  || Y1
VCD info: dumpfile Ejercicios_tb.vcd opened for
0   |   0   | 0 | 0 | 0 | 0 | 000xx | 0000x || |
0   |   1   | 0 | 0 | 0 | 0 | 00000 | 00001 || |
0   |   0   | 0 | 0 | 0 | 0 | 00000 | 00001 || |
1   |   0   | 0 | 0 | 0 | 0 | 00001 | 00001 || |
0   |   0   | 0 | 0 | 0 | 0 | 00001 | 00001 || |
0   |   0   | 1 | 0 | 0 | 0 | 00001 | 00010 || |
1   |   0   | 1 | 0 | 0 | 0 | 00010 | 00010 || |
0   |   0   | 1 | 0 | 0 | 0 | 00010 | 00010 || |
1   |   0   | 1 | 0 | 0 | 0 | 00010 | 00010 || |
0   |   0   | 1 | 0 | 0 | 0 | 00010 | 00010 || |
0   |   0   | 0 | 1 | 0 | 0 | 00010 | 00000 || |
1   |   0   | 0 | 1 | 0 | 0 | 00000 | 00000 || |
0   |   0   | 0 | 1 | 0 | 0 | 00000 | 00000 || |
1   |   0   | 0 | 1 | 0 | 0 | 00000 | 00000 || |
0   |   0   | 0 | 1 | 0 | 0 | 00000 | 00000 || |
1   |   0   | 0 | 1 | 0 | 0 | 00000 | 00000 || |
0   |   0   | 0 | 1 | 0 | 0 | 00000 | 00000 || |
0   |   0   | 0 | 0 | 1 | 0 | 00000 | 00000 || |
1   |   0   | 0 | 0 | 1 | 0 | 00000 | 00000 || |
0   |   0   | 0 | 0 | 1 | 0 | 00000 | 00000 || |
1   |   0   | 0 | 0 | 1 | 0 | 00000 | 00000 || |
0   |   0   | 0 | 0 | 1 | 0 | 00000 | 00000 || |
1   |   0   | 0 | 0 | 1 | 0 | 00000 | 00000 || |
0   |   0   | 0 | 0 | 1 | 0 | 00000 | 00000 || |
1   |   0   | 0 | 0 | 1 | 0 | 00000 | 00000 || |
0   |   0   | 0 | 0 | 1 | 0 | 00000 | 00000 || |
0   |   0   | 0 | 0 | 0 | 1 | 00000 | 00000 || |
1   |   0   | 0 | 0 | 0 | 1 | 00000 | 00000 || |
0   |   0   | 0 | 0 | 0 | 1 | 00000 | 00000 || |
1   |   0   | 0 | 0 | 0 | 1 | 00000 | 00000 || |
0   |   0   | 0 | 0 | 0 | 1 | 00000 | 00000 || |
1   |   0   | 0 | 0 | 0 | 1 | 00000 | 00000 || |
```

Estas primeras imágenes demuestran el funcionamiento lógico de nuestro programa de simulación GTKwave y de nuestro comando de sistema al interpretar nuestra programación.

En la primera imagen podemos ver como los estados dependen de nuestras entradas. Los estados se ven afectados directamente del cambio de las entradas pero no son inmediatas dado que hay que esperar el flanco de reloj de nuestro clk para realizar dicho cambio.

Esta dependencia del clk es algo que no se puede demostrar directamente en nuestra tabla de estados de la segunda imagen dado que esta solo demuestra el cambio en la salida después de que el cambio en la entrada ya se haya realizado.

```
initial begin
        $display("Salidas");
        $display("Clk | reset | I1 | I2 | D1 | D2 | Sf | S  || |");
        $monitor("  %b  |   %b   | %b | %b | %b | %b | %b | %b || |"
end

initial begin
        clk = 0;
        reset = 0;
        I1 = 0;
        I2 = 0;
        D1 = 0;
        D2 = 0;
        #1 reset = 1;
        #1 reset = 0;
        #10
        I1 = 1;
        I2 = 0;
        D1 = 0;
        D2 = 0;
        #20
        I1 = 0;
        I2 = 1;
        D1 = 0;
        D2 = 0;
        #30
        I1 = 0;
        I2 = 0;
        D1 = 1;
```

Aquí podemos demostrar lo “importante” de nuestro Testbench de la programación dado que vemos como se harán los cambios de las entradas y luego en la simulación veremos como estos se relacionan a las salidas.

```
module salidas(input I1, I2, D1, D2, clk, reset, output wire Q, outp
        wire S0, S1, S2, S0fut, S1fut, S2fut, S3, S3fut, S4, S4fut;

  not W0(NS0,S0);
  not W1(NS1,S1);
  not W2(NS2,S2);
  not W3(NS3,S3);
  not W4(NS4,S4);
  not W5(NI1,I1);
  not W6(NI2,I2);
  not W7(ND1,D1);
  not W8(ND2,D2);

        assign Ng = (NS0 & NS1 & NS2 & NS3 & NS4);

        assign S0fut = (NI1 & NI2 & ND1 & ND2 & Ng)|(NI1 & NI2 & ND1
        assign S1fut = (I1 & NI2 & ND1 & ND2 & Ng)|(I1 & NI2 & ND1 &
        assign S2fut = (NI1 & I2 & ND1 & ND2 & Ng)|(NI1 & I2 & ND1 &
        assign S3fut = (NI1 & NI2 & D1 & ND2 & Ng)|(NI1 & NI2 & D1 &
        assign S4fut = (NI1 & NI2 & ND1 & D2 & Ng)|(NI1 & NI2 & ND1 &

        FFD U3(clk, reset, S0fut, S0);
        FFD U4(clk, reset, S1fut, S1);
        FFD U5(clk, reset, S2fut, S2);
        FFD U6(clk, reset, S3fut, S3);
        FFD U7(clk, reset, S4fut, S4);


        assign Spres = {S4,S3,S2,S1,S0};
        assign Sfut = {S4fut,S3fut,S2fut,S1fut,S0fut};


endmodule
```

Esto es una demostración únicamente de como se realizo la programación. Para ver la programación completa favor abrir los archivos correspondientes ubicados en el zip de este archivo o en mi espacio de GitHub https://github.com/dar17320/Proyecto-maquinas-de-estado

Video

El video donde se explica mas a detalle se encontrará en este link una vez se termine de subir.

https://youtu.be/xi-YOqTQRWw