Diego A. Méndez
Carnet 19673

Laboratorio #6

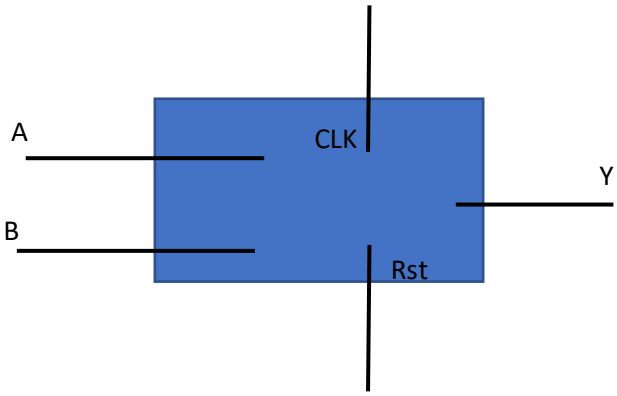
Electronica Digital

Ejercicio 1|

Estado	Codificación
S0	00
S1	01
S2	10

Estado Actual	A	B	Estado Futuro	Y	S1	S0	A	B	S'1	S'0	Y
S0	0	x	S0	0	0	0	0	x	0	0	0
S0	1	x	S1	0	0	0	1	x	0	1	0
S1	x	0	S0	0	0	1	x	0	0	0	0
S1	x	1	S2	0	0	1	x	1	1	0	0
S2	1	1	S2	1	1	0	1	1	1	0	1
S2	0	0	S0	0	1	0	0	0	0	0	0
S2	0	1	S0	0	1	0	0	1	0	0	0
S2	1	0	S0	0	1	0	1	0	0	0	0

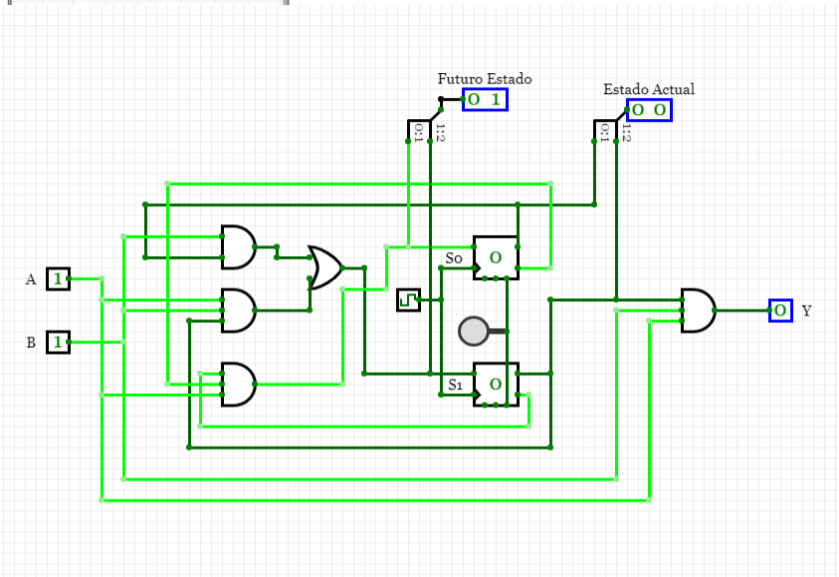
Term	S1	S0	A	B	=>	S'1	S'0	Y
0	0	0	0	0		0	0	0
1	0	0	0	1		0	0	0
2	0	0	1	0		0	1	0
3	0	0	1	1		0	1	0
4	0	1	0	0		0	0	0
5	0	1	0	1		1	0	0
6	0	1	1	0		0	0	0
7	0	1	1	1		1	0	0
8	1	0	0	0		0	0	0
9	1	0	0	1		0	0	0
10	1	0	1	0		0	0	0
11	1	0	1	1		1	0	1
12	1	1	0	0		X	X	X
13	1	1	0	1		X	X	X
14	1	1	1	0		X	X	X
15	1	1	1	1		X	X	X

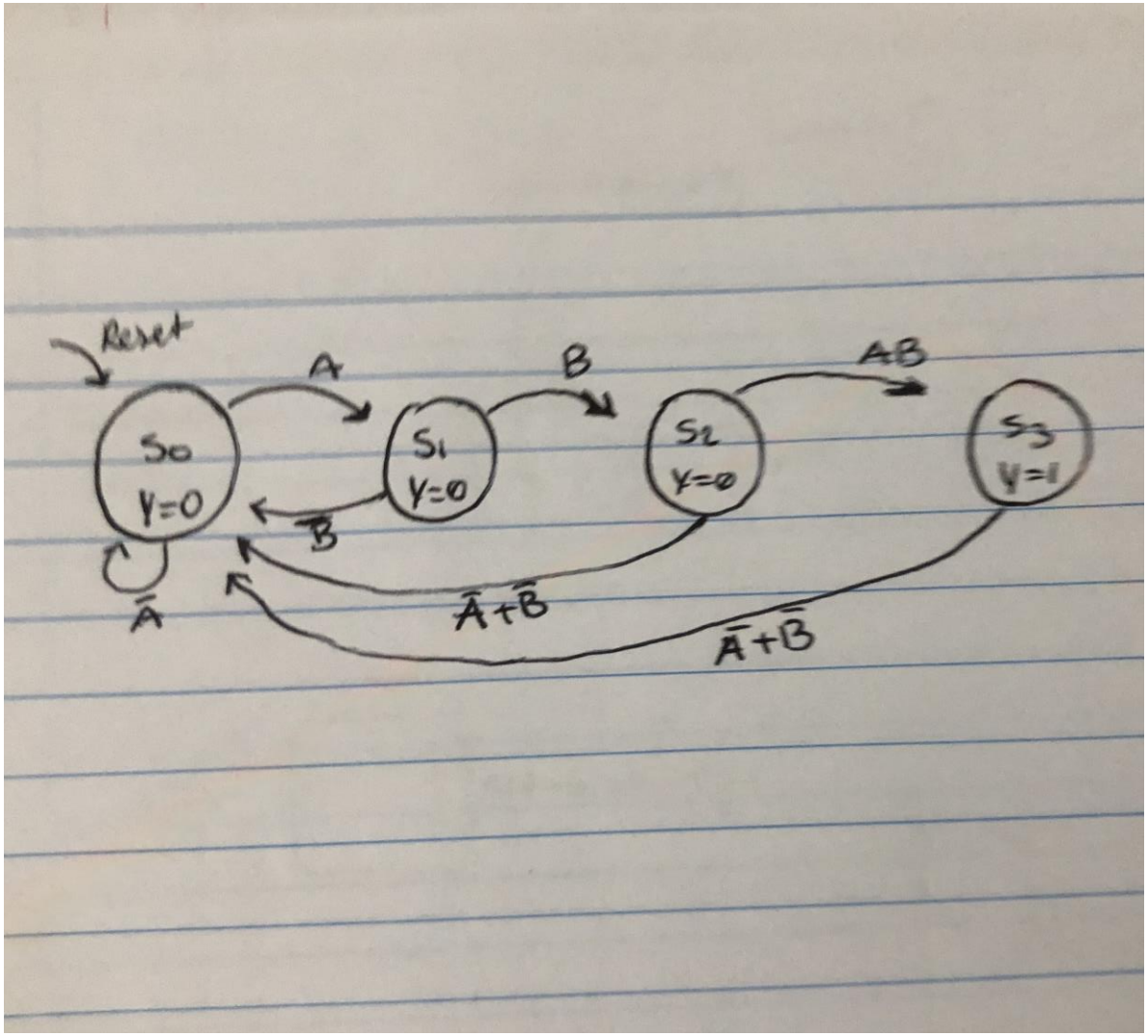


Functi...	Inputs	Outputs	True	False	DC	PI	Gates
S'1-Y	4	3	3, 2, 1	9, 10,...	4, 4, 4	4	Not mapped

S1	S0	A	B	=>	S'1	S'0	Y
X	1	X	1		1		
1	X	1	1		1		
0	0	1	X			1	
1	X	1	1				1

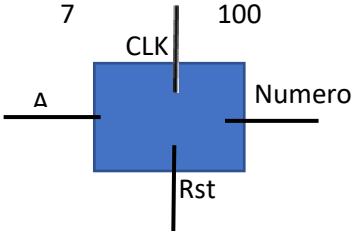
```
Entered by truthtable:
S'1 = S1' S0 A' B + S1' S0 A B + S1 S0' A B;
S'0 = S1' S0' A B' + S1' S0' A B;
Y = S1 S0' A B;

Minimized:
S'1 = S0 B + S1 A B;
S'0 = S1' S0' A ;
Y = S1 A B;
```




Reset
A
B
AB
S_0
Y=0
S_1
Y=0
S_2
Y=0
S_3
Y=1
$\bar{B}$
$\bar{A}$
$\bar{A}+\bar{B}$
$\bar{A}+\bar{B}$

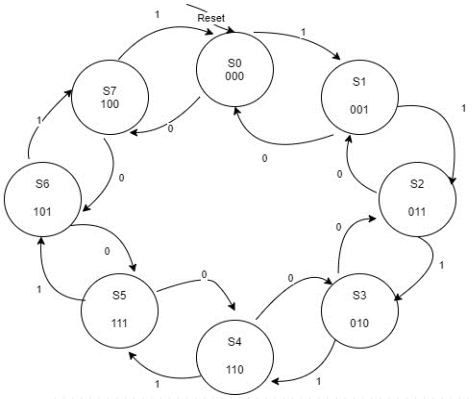
Ejercicio 3

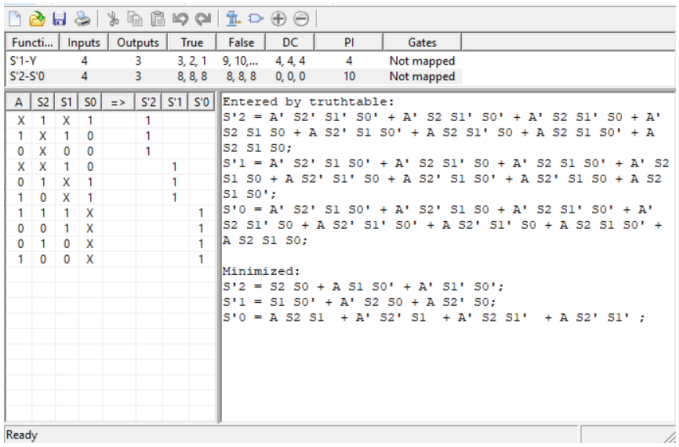
Número	Codificación Gray
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

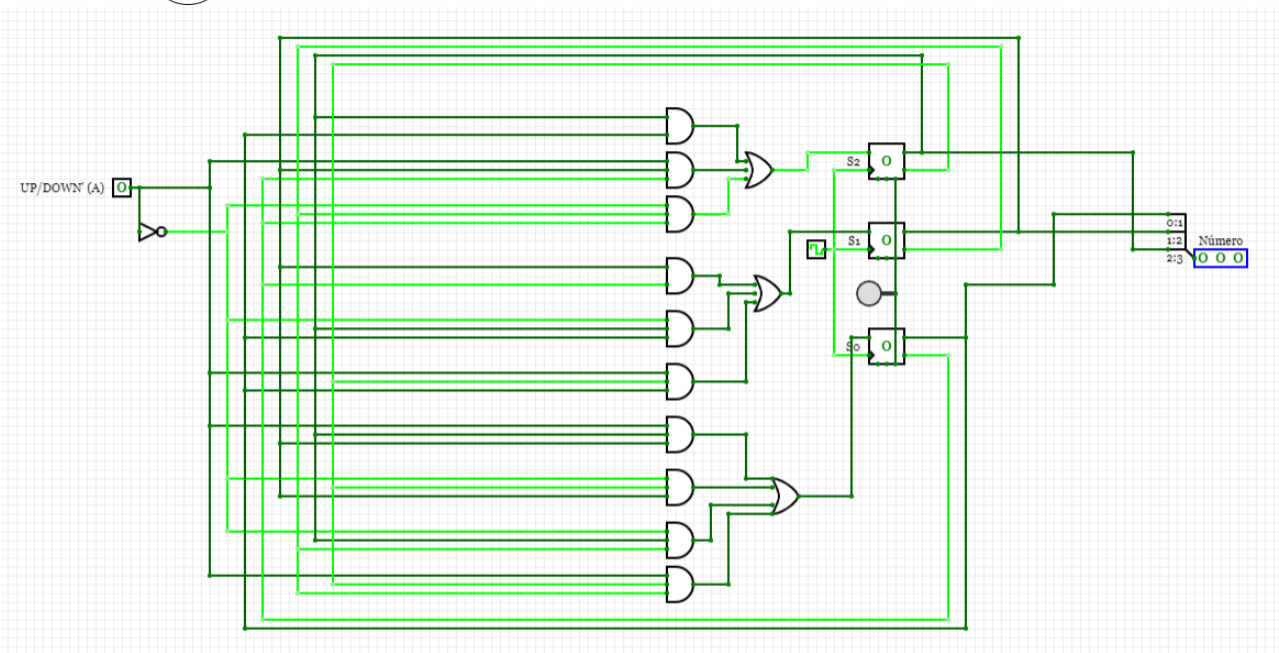


Estado Actual	A	Estado Futuro
000	1	001
001	1	011
011	1	010
010	1	110
110	1	111
111	1	101
101	1	100
100	1	000
000	0	100
100	0	101
101	0	111
111	0	110
110	0	010
010	0	011
011	0	001
001	0	000

Term	A	S2	S1	S0	=>	S'2	S'1	S'0
0	0	0	0	0		1	0	0
1	0	0	0	1		0	0	0
2	0	0	1	0		0	1	1
3	0	0	1	1		0	0	1
4	0	1	0	0		1	0	1
5	0	1	0	1		1	1	1
6	0	1	1	0		0	1	0
7	0	1	1	1		1	1	0
8	1	0	0	0		0	0	1
9	1	0	0	1		0	1	1
10	1	0	1	0		1	1	0
11	1	0	1	1		0	1	0
12	1	1	0	0		0	0	0
13	1	1	0	1		1	0	0
14	1	1	1	0		1	1	1
15	1	1	1	1		1	0	1







Ejercicio 4

Los Blocking Statements deben ser ejecutados antes de poder proseguir con el resto de la programación, es decir, que no permite que los statements sean ejecutados en paralelo. Mientras que los NonBlocking Statements permiten que el resto de la programación sea ejecutado al mismo tiempo que el NonBlocking Statement se ejecuta, es decir que permite la ejecución en paralelo.

Ejemplo Blocking

Son los comandos que hemos utilizado en laboratorios anteriores

```
// Products of sum
  or U4 (out01,inA,outBN,inC,inD);
  or U5 (out02,inA,outBN,inC,outDN);
  or U6 (out03,inA,outBN,outCN,inD);
  or U7 (out04,inA,outBN,outCN,outDN);
  or U8 (out05,outAN,inB,inC,outDN);
  or U9 (out06,outAN,inB,outCN,outDN);
  or U10 (out07,outAN,outBN,inC,inD);
  or U11 (out08,outAN,outBN,inC,outDN);
  or U11 (out09,outAN,outBN,outCN,outDN);

  and UR(outres,out01,out02,out03,out04,out05,out06,out07,

  initial begin

    $display("A B C D| Y");
    $display("----------");
    $monitor("%b %b %b %b %b",inA,inB,inC,inD,outres);
       inA=0; inB=0; inC=0; inD=0;
    #1 inA=0; inB=0; inC=0; inD=1;
    #1 inA=0; inB=0; inC=1; inD=0;
    #1 inA=0; inB=0; inC=1; inD=1;
    #1 inA=0; inB=1; inC=0; inD=0;
    #1 inA=0; inB=1; inC=0; inD=1;
    #1 inA=0; inB=1; inC=1; inD=0;
```

Unblocking

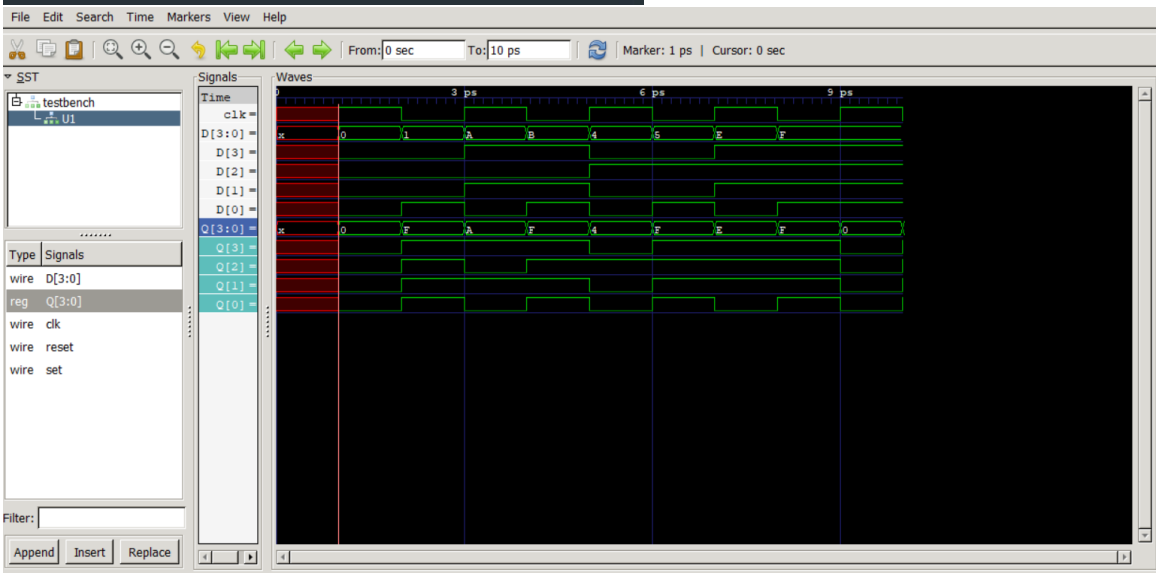
```
//Hacemos que el reset haga Q
if (reset) begin
  Q <= 0;
  end
//Hacemos que Q se vuelva D
else
  Q <= D;
end
```

Ejercicio 5

```
//Diego A. Méndez
//Ejercicio 5

//FlipFlop 4 Bits
module ffD(input clk, reset, set,
           input[3:0] D,
           output reg[3:0] Q);



    always @(posedge clk, posedge reset, posedge set) begin
      //Hacemos que el reset haga Q = 0000
      if (reset) begin
        Q <= 4'b0;
        end
      //Hacemos que el set haga Q=1111
      else if (set) begin
        Q <= 4'b1111;
        end
      //Hacemos que Q se vuelva D
      else
        Q <= D;
      end


endmodule
```

```
module testbench();

  reg clk, reset, set;
  reg[3:0] D;
  wire[3:0] Q;

    always
      begin
      #1 clk <= 1; set<=0 ;#1 clk<=0; set<=1;
      end

    ffD U1(clk, reset, set, D, Q);


     initial begin //iniciamos el mux8_1
      $display(" ");
      $display("  ");
      $display(" ");
      $display("CLK | RST | ST | D | Q");
      $display("----------");
      $monitor("%b | %b | %b | %b | %b",clk ,reset, set, D, Q);
           reset<=0;
        #1 D<=000;
        #1 D<=001;
        #1 D<=010;
        #1 D<=011;
```

```
      $display(" ");
      $display("CLK | RST | ST | D | Q");
      $display("----------");
      $monitor("%b | %b | %b | %b | %b",clk ,reset, set, D, Q);
           reset<=0;
        #1 D<=000;
        #1 D<=001;
        #1 D<=010;
        #1 D<=011;
        #1 D<=100;
        #1 D<=101;
        #1 D<=110;
        #1 D<=111;
        #1 reset<=1;
        #1 reset<=0;

        end

  initial
  #10 $finish;

  initial begin
    $dumpfile("FFD_tb.vcd");
    $dumpvars(0, testbench);
  end

endmodule
```



Ejercicio 1

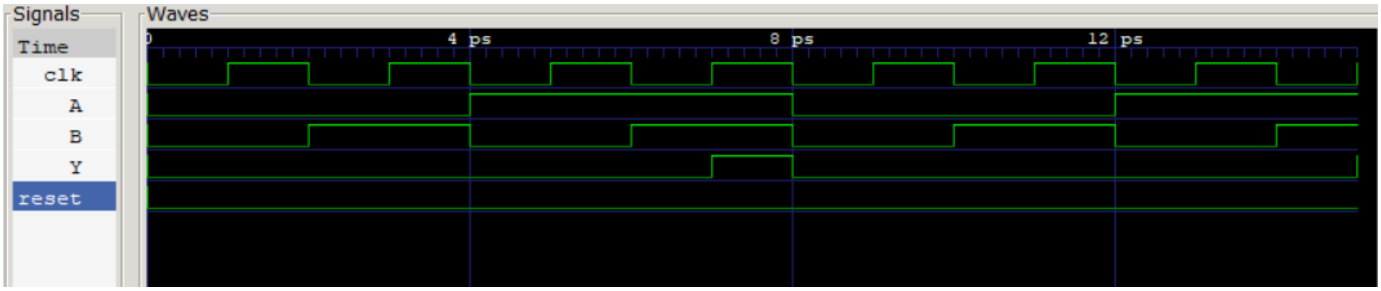
```
//Diego A. Méndez
//Ejercicio 1

//FlipFlop del ejercicio 5
module ffD(input clk, reset, input D, output reg Q);

    always @(posedge clk, posedge reset) begin
      //Hacemos que el reset haga Q = 0
      if (reset) begin
        Q <= 0;
        end
      //Hacemos que Q se vuelva D
      else
        Q <= D;
      end
endmodule
//Modulo para FSM
module FSM(input wire A, B, clk, reset, output Y );
  wire Sf1, Sf0;
  wire S1, S0;
//negamos los cables de estado
```

```
// conectamos los flip flops
  ffD U1(clk, reset, Sf1, S1);
  ffD U2(clk, reset, Sf0, S0);



endmodule
```

```
//Diego A. Méndez
//TestBench Ejercicio5

module testbench();

  reg clk, reset;
  reg A, B;
  wire Y;

//Iniciamos el clock
    always
      begin
      clk <= 0; reset<=0; #1 clk<=1; #1;

      end
//Llamamos al modelo FSM
      FSM U1(A, B, clk, reset, Y);


      initial begin //iniciamos el display
            $display(" ");
```

```
            $display("CLK | RST | A | B | Y");
            $display("----------");
            $monitor("%b | %b | %b | %b | %b",clk ,reset, A, B, Y);
             A=0; B=0;
            #2 A=0; B=1;
            #2 A=1; B=0;
            #2 A=1; B=1;
            #2 A=0; B=0;
            #2 A=0; B=1;
            #2 A=1; B=0;
            #2 A=1; B=1;
            #2 A=1; B=1;
            //

            //  #2 reset=1;

          end

        initial
        #15 $finish;
```

```
---> WARNING: no PCF file found (.pcf)

iverilog -o FSM_tb.out -D VCD_OUTPUT=FSM_tb C:/Users/diego/.apio
FSM.v FSM_tb.v
vvp FSM_tb.out



CLK | RST | A | B | Y
----------
VCD info: dumpfile FSM_tb.vcd opened for output.
0 | 0 | 0 | 0 | 0
1 | 0 | 0 | 0 | 0
0 | 0 | 0 | 1 | 0
1 | 0 | 0 | 1 | 0
0 | 0 | 1 | 0 | 0
1 | 0 | 1 | 0 | 0
0 | 0 | 1 | 1 | 0
1 | 0 | 1 | 1 | 1
0 | 0 | 0 | 0 | 0
1 | 0 | 0 | 0 | 0
0 | 0 | 0 | 1 | 0
1 | 0 | 0 | 1 | 0
0 | 0 | 1 | 0 | 0
1 | 0 | 1 | 0 | 0
0 | 0 | 1 | 1 | 0
1 | 0 | 1 | 1 | 1
gtkwave FSM_tb.vcd FSM_tb.gtkw
```



```
//Diego A. Méndez
//Ejercicio 3

//FlipFlop del ejercicio 1
module ffD(input clk, reset, input D, output reg Q);

    always @(posedge clk, posedge reset) begin
      //Hacemos que el reset haga Q = 0
      if (reset) begin
        Q <= 0;
        end
      //Hacemos que Q se vuelva D
      else
        Q <= D;
      end
endmodule
//Modulo basado en el ejercicio 1
module FSM(input wire A, clk, reset, output wire[2:0] Y);
  wire Sf2, Sf1, Sf0;
  wire S2, S1, S0, nS0, nS1, nS2;




//negacion de las compuertas
  not(nS2, S2);
  not(nS1, S1);
  not(nS0, S0);
  not(nA, A);
```

```
//negacion de las compuertas
  not(nS2, S2);
  not(nS1, S1);
  not(nS0, S0);
  not(nA, A);
//assignasion de las ecuaciones de estado proximo
  assign Sf2 = S2 & S0 + A & S1 & nS0 + nA & nS1 & nS0;
  assign Sf1 = S1 & nS0 + nA & S2 & S0 + A & nS2 & S0;
  assign Sf0 = A & S2 & S1 + nA & nS2 & S1 + nA & S2 & nS1 + A & nS2 & nS1;

//conexion de los flip flops
  ffD U3(clk, reset, Sf2, S2);
  ffD U1(clk, reset, Sf1, S1);
  ffD U2(clk, reset, Sf0, S0);


   assign Y[0]=S1;
   assign Y[1]=S1;
   assign Y[2]=S2;
  endmodule
```

```
//Diego A. Méndez
//TestBench Ejercicio3

module testbench();

  reg clk, reset;
  reg A;
  wire[2:0] Y;

//Iniciamos el clock
    always
      begin
      clk <= 1 ; #1 clk<=0; #1;

      end
//Llamamos al modelo FSM
      FSM U1(A, clk, reset, Y);


      initial begin //iniciamos el display
            $display(" ");
            $display("  ");
            $display(" ");
            $display("CLK | RST | A | Y");
            $display("----------");
            $monitor("%b | %b | %b | %b",clk ,reset, A, Y);

               A=1; reset=1;
            #2 A=1; reset=0;
```

```
      end
//Llamamos al modelo FSM
      FSM U1(A, clk, reset, Y);


      initial begin //iniciamos el display
            $display(" ");
            $display("  ");
            $display(" ");
            $display("CLK | RST | A | Y");
            $display("----------");
            $monitor("%b | %b | %b | %b",clk ,reset, A, Y);

               A=1; reset=1;
            #2 A=1; reset=0;
            #2 A=1;
            #2 A=1;
            #2 A=1;
            #2 A=1;
            #2 A=1;
            #2 A=1;
            #2 A=0;
            #2 A=0;
            #2A=0;
            #2A=0;
```