Brandon Amisael Garrido Ramírez
Carné: 19421
Sección: 20

Ing. Mecatrónica
Electrónica Digital 1

Laboratorio #9

Enlace al repositorio de GitHub: https://github.com/gar19421/Lab-09.git

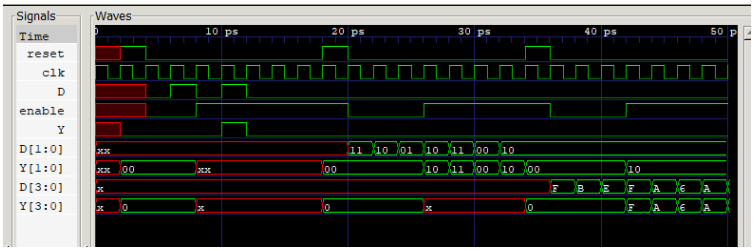
---

Ejercicio 1:

En este ejercicio se implementó un Flip-Flop tipo D de 1 bit con un enable que cuando estuviera en 1, este permitiría dejar pasar el valor de la entrada D a la salida Y en cada flanco de reloj. Posteriormente se implementó con este un Flip-Flop de 2 bits y posteriormente de 4 bits, con base al mismo funcionamiento de primero, tal y como se muestra en la imagen.

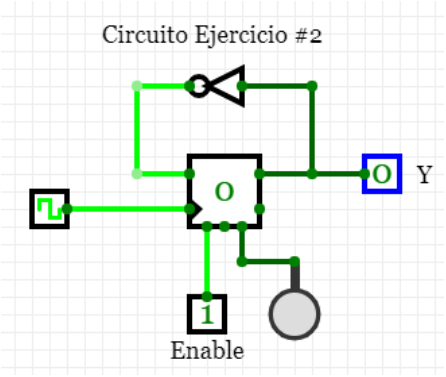
```
module FF_1BIT(input wire clk, reset, enable, D, output reg Y) ;
  always @(posedge clk or posedge enable or posedge reset) begin
    if (reset)  Y <= 1'b0;
    else if(enable) Y <= D;
  end
endmodule


module FF_2BIT(input wire clk, reset, enable, input wire [1:0] D, output wire [1:0] Y) ;
  FF_1BIT FF1(clk, reset, enable, D[1], Y[1]);
  FF_1BIT FF2(clk, reset, enable, D[0], Y[0]);
endmodule

module FF_4BIT(input wire clk, reset, enable, input wire [3:0] D, output wire [3:0] Y) ;
  FF_1BIT FF1(clk, reset, enable, D[3], Y[3]);
  FF_1BIT FF2(clk, reset, enable, D[2], Y[2]);
  FF_1BIT FF3(clk, reset, enable, D[1], Y[1]);
  FF_1BIT FF4(clk, reset, enable, D[0], Y[0]);
endmodule
```

Como se observa en el diagrama de timing, cuando la señal de enable está en 1, la señal D de cada flipflop pasa a la señal de salida en el flanco de reloj.



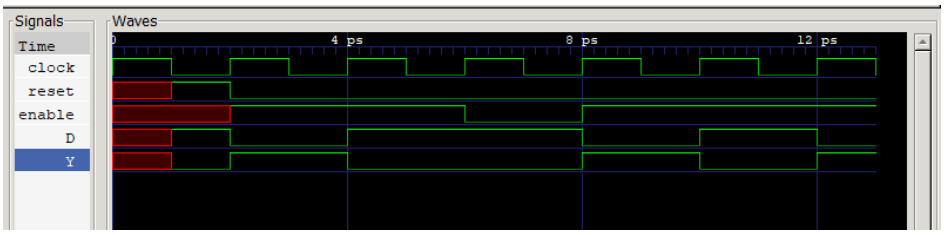
Ejercicio 2:

En este ejercicio se implementó un flipflop tipo T, utilizando el mismo flipflop tipo D de 1 bit del ejercicio anterior. Se tiene como entradas el clock, enable y reset, y a la salida Y que es la salida del flipflop que es D', el cual es un wire interno en el fliflop. Se implementó tanto en circuitverse, como en verilog, como se observa en las siguientes imágenes.




```verilog
//Flip Flop Tipo d de 1 bit
module FF_D1(input wire clk, reset, enable, D, output reg Y) ;
  always @(posedge clk or posedge enable or posedge reset) begin
    if (reset)  Y <= 1'b0;
    else if(enable) Y <= D;
  end
endmodule


//Flip Flop Tipo T de 1 bit
module FF_T(input wire clock, reset, enable, output wire Y);
  wire d;
  assign d = ~Y;
  FF_D1 G1(clock, reset, enable, d, Y);
endmodule
```

Como se puede observar en el diagrama de timing, el funcionamiento es el correcto ya que la señal de salida Y es el negado de la señal D, en cada flanco de reloj y cuando el enable se vuelve cero mantiene su valor anterior.



Ejercicio 3:

En este ejercicio, se implementó un flipflop JK con flipflop tipo D del ejercicio 1 tanto en verilog como en circuitverse, con sus entradas clock, J y K, y su salida Q la cual Si J y K son 0 entonces la salida Q mantiene su valor anterior. Si J = 1 y K = 0 entonces Q = 1. Si J = 0 y K = 1 entonces Q = 0. Si J y K son = 1 entonces la salida Q = Q' es decir se invierte.

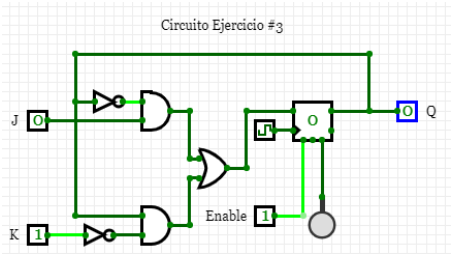
Para implementar dicho funcionamiento se trabajó las distintas combinaciones como una FSM con el funcionamiento en específico y se colocó la tabla de estados y salidas en logic Friday para obtener las ecuaciones booleanas e implementarlas en circuitverse y trabajarlo de forma structural en verilog.

Term	Q	J	K	=>	QF
0	0	0	0		0
1	0	0	1		0
2	0	1	0		1
3	0	1	1		1
4	1	0	0		1
5	1	0	1		0
6	1	1	0		1
7	1	1	1		0

Functi...	Inputs	Outputs	True	False	DC
QF	3	1	4	4	0

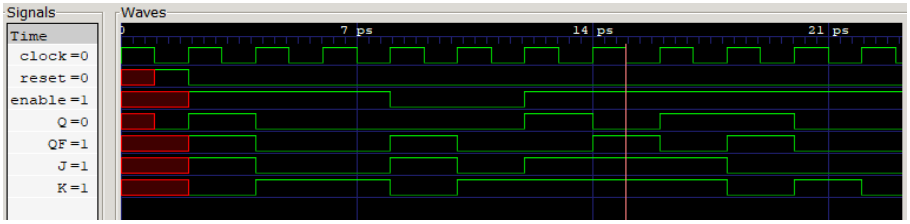
Q	J	K	=>	QF
1	X	0		1
0	1	X		1

```
Entered by truthtab
QF = Q' J K' + Q' J

Minimized:
QF = Q K' + Q' J ;
```



```
module FF_D(input wire clk, reset, enable, D, output reg Y) ;
  always @(posedge clk or posedge enable or posedge reset) begin
    if (reset)  Y <= 1'b0;
    else if(enable) Y <= D;
  end
endmodule

module FF_JK(input wire clock, reset, enable, J, K, output wire Q);
  wire c1, c2;

  FF_D G1(clock, reset, enable, QF, Q);

  and U1(c1, ~Q, J);
  and U2(c2, Q, ~K);
  or U3(QF, w1, w2);

endmodule
```

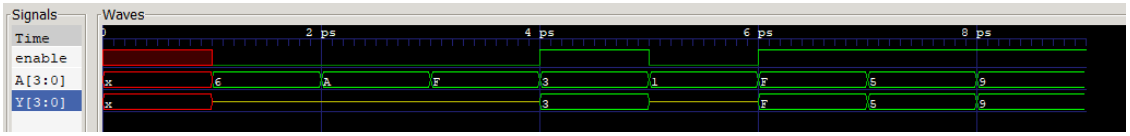
Como se observa en el diagrama de timing, las combinaciones descritas anteriormente se cumplen correctamente en cada flanco de reloj y cuando enable está en cero, la salida mantiene su valor anterior.



Ejercicio 4:

En este ejercicio se implementó un buffer triestado de 4 bits en verilog. El funcionamiento de este es que cuando la señal de entrada del enable es 1, deja pasar la señal A hacia la salida Y, sin embargo, cuando el enable es 0, la salida se encuentra en alta impedancia tal y como se observa tanto en verilog como en el diagrama de timing.

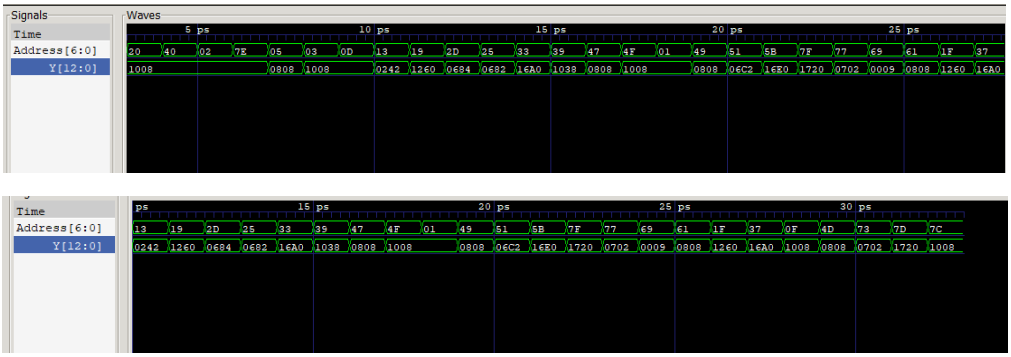
```
module Buffer_4(input wire enable, input wire [3:0]A, output wire [3:0] Y);

  assign Y = enable ? A : 4'bz;

endmodule
```



Ejercicio 5:

En este ejercicio se implementó la tabla de verdad mostrada en la guía de laboratorio, como una memoria ROM utilizando la sentencia switch-case, probando cada una de las opciones necesarias, tanto con don't cares, como las distintas combinaciones de entradas con sus respectivas salidas según la dirección (Address) implementadas dentro de un bloque always.

```
module ROM (input wire [6:0]Address, output reg [12:0]Y);//implementacion de rom
  always @(Address) begin
    case (Address)
      7'b0000000: Y = 13'b1000000001000 ;//primer caso con Don't cares
      7'b1000000: Y = 13'b1000000001000 ;
      7'b0100000: Y = 13'b1000000001000 ;
      7'b0010000: Y = 13'b1000000001000 ;
      7'b0001000: Y = 13'b1000000001000 ;
      7'b0000100: Y = 13'b1000000001000 ;
      7'b0000010: Y = 13'b1000000001000 ;
      7'b1111110: Y = 13'b1000000001000 ;
      7'b1111100: Y = 13'b1000000001000 ;
      7'b1111000: Y = 13'b1000000001000 ;
      7'b1110000: Y = 13'b1000000001000 ;
      7'b1100000: Y = 13'b1000000001000 ;
      7'b0111110: Y = 13'b1000000001000 ;
      7'b0011110: Y = 13'b1000000001000 ;
      7'b0001110: Y = 13'b1000000001000 ;
      7'b0000110: Y = 13'b1000000001000 ;
      7'b1011110: Y = 13'b1000000001000 ;
      7'b1001110: Y = 13'b1000000001000 ;
      7'b1000110: Y = 13'b1000000001000 ;
      7'b1000010: Y = 13'b1000000001000 ;
      7'b1101110: Y = 13'b1000000001000 ;
      7'b1100110: Y = 13'b1000000001000 ;
      7'b1100010: Y = 13'b1000000001000 ;
      7'b1110110: Y = 13'b1000000001000 ;
      7'b1110010: Y = 13'b1000000001000 ;
      7'b1111010: Y = 13'b1000000001000 ;
      7'b1111100: Y = 13'b1000000001000 ;//caso 1
      7'b0000101: Y = 13'b0100000001000 ;
      7'b1001011: Y = 13'b1000000001000 ;
      7'b1001111: Y = 13'b1000000001000 ;//caso 14
      7'b1001001: Y = 13'b0100000001000 ;
      7'b1001101: Y = 13'b0100000001000 ;//caso 15
      7'b1010001: Y = 13'b0011011000010 ;
      7'b1010011: Y = 13'b0011011000010 ;
      7'b1010101: Y = 13'b0011011000010 ;
      7'b1010111: Y = 13'b0011011000010 ;//caso 16
      7'b1011001: Y = 13'b1011011100000 ;
      7'b1011011: Y = 13'b1011011100000 ;
      7'b1011101: Y = 13'b1011011100000 ;
      7'b1011111: Y = 13'b1011011100000 ;//caso 17
      7'b1100001: Y = 13'b0100000001000 ;
      7'b1100011: Y = 13'b0100000001000 ;
      7'b1100101: Y = 13'b0100000001000 ;
      7'b1100111: Y = 13'b0100000001000 ;//caso 18
      7'b1101001: Y = 13'b0000000001001 ;
      7'b1101011: Y = 13'b0000000001001 ;
      7'b1101101: Y = 13'b0000000001001 ;
      7'b1101111: Y = 13'b0000000001001 ;//caso 19
      7'b1110001: Y = 13'b0011100000010 ;
      7'b1110011: Y = 13'b0011100000010 ;
      7'b1110101: Y = 13'b0011100000010 ;
      7'b1110111: Y = 13'b0011100000010 ;//caso 20
      7'b1111001: Y = 13'b1011100100000 ;
      7'b1111011: Y = 13'b1011100100000 ;
      7'b1111101: Y = 13'b1011100100000 ;
      7'b1111111: Y = 13'b1011100100000 ;//caso 21
      default:    Y = 13'bxxxxxxxxxxxxx ; // caso por default
    endcase
  end
```



Otras evidencias (Screenshots):

- Pruebas en consola:

```
CLK Reset | Enable D |  Y
----------|----------|------
VCD info: dumpfile ej01_tb.vcd
1     x   |    x   x |  x
0     x   |    x   x |  x
1     1   |    x   x |  0
0     1   |    x   x |  0
1     0   |    0   0 |  0
0     0   |    0   0 |  0
1     0   |    0   1 |  0
0     0   |    0   1 |  0
1     0   |    1   0 |  0
0     0   |    1   0 |  0
1     0   |    1   1 |  1
0     0   |    1   1 |  1
1     0   |    1   0 |  0
0     0   |    1   0 |  0
1     0   |    1   0 |  0
0     0   |    1   0 |  0
```

```
FLIP-FLOP DE 2 BITS

CLK Reset | Enable D  |  Y
----------|-----------|------
1     0   |    1   xx |  xx
0     0   |    1   xx |  xx
1     1   |    1   xx |  00
0     1   |    1   xx |  00
1     0   |    0   11 |  00
0     0   |    0   11 |  00
1     0   |    0   10 |  00
0     0   |    0   10 |  00
1     0   |    0   01 |  00
0     0   |    0   01 |  00
1     0   |    1   10 |  10
0     0   |    1   10 |  10
1     0   |    1   11 |  11
0     0   |    1   11 |  11
1     0   |    1   00 |  00
0     0   |    1   00 |  00
```

```
FLIP-FLOP DE 4   BITS

CLK Reset | Enable  D    |   Y
----------|-------------|-------
1     0   |    1   xxxx |  xxxx
0     0   |    1   xxxx |  xxxx
1     1   |    1   xxxx |  0000
0     1   |    1   xxxx |  0000
1     0   |    0   1111 |  0000
0     0   |    0   1111 |  0000
1     0   |    0   1011 |  0000
0     0   |    0   1011 |  0000
1     0   |    0   1110 |  0000
0     0   |    0   1110 |  0000
1     0   |    1   1111 |  1111
0     0   |    1   1111 |  1111
1     0   |    1   1010 |  1010
0     0   |    1   1010 |  1010
1     0   |    1   0110 |  0110
0     0   |    1   0110 |  0110
1     0   |    1   1010 |  1010
0     0   |    1   1010 |  1010
1     0   |    1   1000 |  1000
gtkwave ej01_tb.vcd ej01_tb.gtkw
```

```
Flip Flop Tipo T

Clk RST | EN |   Y
--------------------
VCD info: dumpfile ej02_tb.vcd opened
1    x  |  x |   x
0    1  |  x |   0
1    0  |  1 |   1
0    0  |  1 |   1
1    0  |  1 |   0
0    0  |  1 |   0
1    0  |  0 |   0
0    0  |  0 |   0
1    0  |  1 |   1
0    0  |  1 |   1
1    0  |  1 |   0
0    0  |  1 |   0
1    0  |  1 |   1
0    0  |  1 |   1
gtkwave ej02_tb.vcd ej02_tb.gtkw
```

```
Flip Flop Tipo JK - Ejercicio 3

Clk RST | EN  J  K |  Q
-------------------------
VCD info: dumpfile ej03_tb.vcd op
1    x  |  x  x  x |  x
0    1  |  x  x  x |  0
1    0  |  1  1  0 |  1
0    0  |  1  1  0 |  1
1    0  |  1  0  1 |  0
0    0  |  1  0  1 |  0
1    0  |  1  0  1 |  0
0    0  |  1  0  1 |  0
1    0  |  0  1  0 |  0
0    0  |  0  1  0 |  0
1    0  |  0  0  1 |  0
0    0  |  0  0  1 |  0
1    0  |  1  1  1 |  1
0    0  |  1  1  1 |  1
1    0  |  1  1  1 |  0
0    0  |  1  1  1 |  0
1    0  |  1  1  1 |  1
0    0  |  1  1  1 |  1
1    0  |  1  0  0 |  1
0    0  |  1  0  0 |  1
1    0  |  1  0  1 |  0
0    0  |  1  0  1 |  0
1    0  |  1  0  0 |  0
0    0  |  1  0  0 |  0
1    0  |  1  0  0 |  0
0    0  |  1  0  0 |  0
```

```
Buffer Triestado

Enable    A     |    Y
-----------------------
VCD info: dumpfile ej04_tb.vcd opened
x         xxxx  |   xxxx
0         0110  |   zzzz
0         1010  |   zzzz
0         1111  |   zzzz
1         0011  |   0011
0         0001  |   zzzz
1         1111  |   1111
1         0101  |   0101
1         1001  |   1001
gtkwave ej04_tb.vcd ej04_tb.gtkw
```

```
MEMORIA ROM
ADDRESS          |  Y
xxxxxxx          | xxxxxxxxxxxx
0000000          | 1000000001000
0100000          | 1000000001000
1000000          | 1000000001000
0000010          | 1000000001000
1111110          | 1000000001000
1111100          | 1000000001000
0000101          | 0100000001000
0000011          | 1000000001000
0001101          | 1000000001000
0010011          | 0001001000010
0011001          | 1001001100000
0101101          | 0011010000100
0100101          | 0011010000010
0110011          | 1011010100000
0111001          | 1000000111000
1000111          | 0100000001000
1001111          | 1000000001000
0000001          | 1000000001000
1001001          | 0100000001000
1010001          | 0011011000010
1011011          | 1011011100000
1111111          | 1011100100000
1110111          | 0011100000010
1101001          | 0000000001001
1100001          | 0100000001000
0011111          | 1001001100000
0110111          | 1011010100000
0001011          | 0100000001000
1001101          | 0100000001000
1110011          | 0011100000010
1111101          | 1011100100000
gtkwave ej05_tb.vcd ej05_tb.gtkw
```

- Testbech:

```
ej01_tb.v   ej01.v
reg [1:0] D2;
reg [3:0] D3;
wire Y1;
wire [1:0] Y2;
wire [3:0] Y3;


always//instanciacion del clock
  begin
    clock <= 1;
    #1 clock <= ~clock;// se realiza el cambio del reloj
    #1;
end


FF_1BIT FF1(clock, reset, enable, D1, Y1);
FF_2BIT FF2(clock, reset, enable, D2, Y2);
FF_4BIT FF3(clock, reset, enable, D3, Y3);


initial begin// test de FF1
  $display("FLIP-FLOP DE 1 BIT \n");
  $display("CLK Reset | Enable D |  Y");
  $display("----------|----------|------ ");
  $monitor("%b     %b   |    %b   %b | %b", clock,reset,enable,D1,Y1);
  #2 reset = 1;
  #2 reset = 0; enable = 0; D1 = 0;
  #2 D1 = 1;
  #2 enable = 1; D1 = 0;
  #2 D1 = 1;
```

```
initial begin// test de FF1
  $display("FLIP-FLOP DE 1 BIT \n");
  $display("CLK Reset | Enable D |  Y");
  $display("----------|----------|------ ");
  $monitor("%b     %b   |    %b   %b | %b", clock,reset,enable,D1,Y1);
  #2 reset = 1;
  #2 reset = 0; enable = 0; D1 = 0;
  #2 D1 = 1;
  #2 enable = 1; D1 = 0;
  #2 D1 = 1;
  #2 D1 = 0 ;
end

initial begin// test de FF2
  #16
  $display("FLIP-FLOP DE 2 BITS\n");
  $display("CLK Reset | Enable D  |  Y");
  $display("----------|-----------|------ ");
  $monitor("%b     %b   |    %b   %b | %b", clock,reset,enable,D2,Y2);
  #2 reset = 1;
  #2 enable = 0; D2[0] = 1; D2[1] = 1; reset = 0;
  #2 D2[0] = 0; D2[1] = 1;
  #2 D2[0] = 1; D2[1] = 0;
  #2 enable = 1; D2[0] = 0; D2[1] = 1;
  #2 D2[0] = 1; D2[1] = 1;
  #2 D2[0] = 0; D2[1] = 0;
  #2 D2[0] = 0; D2[1] = 1;
end
```

```
module testbench();

  reg clk, rst, enable;
  wire Y;

  FF_T U1(clk, rst, enable, Y);

always // instanciacion del clock
  begin
    clk <= 1;
    #1 clk <= ~clk;
    #1;
end

initial begin
    $display("\nFlip Flop Tipo T\n");
    $display(" Clk RST | EN |   Y    ");
    $display(" ---------------------- ");
    $monitor(" %b    %b   |  %b |   %b ", clk, rst, enable, Y);

    #1 rst = 1; //Reset inicial
    #1 rst = 0; enable = 1; //Reset inicial
    #4 enable = 0;
    #2 enable = 1;
    #5 $finish;
end
```

```
module testbench();

reg clk, rst, enable, J, K;
wire Q;

FF_JK G1(clk, rst, enable, J, K, Q);

always
  begin
    clk <= 1;
    #1 clk <= ~clk;
    #1;
end

initial begin
$display("\n");
  $display("Flip Flop Tipo JK - Ejercicio 3");
  $display("\n");
  $display(" Clk RST | EN  J  K |  Q    ");
  $display(" ------------------------- ");
  $monitor(" %b    %b  |  %b  %b  %b |   %b ", clk, rst, enable, J, K, Q);

  #1 rst = 1; //Reset inicial
  #1 rst = 0; enable = 1; J = 1; K = 0; //Reset inicial
  #2  J = 0; K = 1;
  #4 enable = 0;  J = 1; K = 0;
  #2  J = 0; K = 1;
  #2 enable = 1; J = 1; K = 1;
  #6  J = 0; K = 0;
  #2  J = 0; K = 1;
  #2  J = 0; K = 0;
  #3 $finish;
```

```
module testbench();

  reg enable;
  reg [3:0] A;
  wire[3:0] Y;

  Buffer_4 B1(enable, A, Y);

  initial begin
      $display("\nBuffer Triestado\n");
      $display(" Enable    A    |    Y    ");
      $display(" ------------------------");
      $monitor(" %b        %b   |   %b ", enable, A, Y);
      #1 enable = 0; A[0]=0; A[1]=1; A[2]=1; A[3]=0;
      #1 enable = 0; A[0]=0; A[1]=1; A[2]=0; A[3]=1;
      #1 enable = 0; A[0]=1; A[1]=1; A[2]=1; A[3]=1;
      #1 enable = 1; A[0]=1; A[1]=1; A[2]=0; A[3]=0;
      #1 enable = 0; A[0]=1; A[1]=0; A[2]=0; A[3]=0;
      #1 enable = 1; A[0]=1; A[1]=1; A[2]=1; A[3]=1;
      #1 enable = 1; A[0]=1; A[1]=0; A[2]=1; A[3]=0;
      #1 enable = 1; A[0]=1; A[1]=0; A[2]=0; A[3]=1;
      #1 $finish;

  end

  initial begin
    $dumpfile("ej04_tb.vcd");
    $dumpvars(0, testbench);
  end

endmodule
```

```
module testbench ();
reg [6:0]ADDRESS;
wire [12:0]Y;

ROM Mem1(ADDRESS,Y);

initial begin
#1
$display("\n");
$display(" MEMORIA ROM ");
$display("ADDRESS \t |  Y");
$monitor("%b \t | %b ", ADDRESS, Y );
#1 ADDRESS = 7'b0000000;//caso1
#1 ADDRESS = 7'b0100000;//caso 1
#1 ADDRESS = 7'b1000000;//caso 1
#1 ADDRESS = 7'b0000010;//caso 1
#1 ADDRESS = 7'b1111110;//caso 1
#1 ADDRESS = 7'b0000101;//caso 2
#1 ADDRESS = 7'b0000011;//caso 3
#1 ADDRESS = 7'b0001101;//caso 4
#1 ADDRESS = 7'b0010011;//caso 6
#1 ADDRESS = 7'b0011001;//caso 7
#1 ADDRESS = 7'b0101101;//caso 9
#1 ADDRESS = 7'b0100101;//caso 8
#1 ADDRESS = 7'b0110011;//caso 10
#1 ADDRESS = 7'b0111001;//caso 11
#1 ADDRESS = 7'b1000111;//caso 12
#1 ADDRESS = 7'b1001111;//caso 14
#1 ADDRESS = 7'b0000001;//caso 13
#1 ADDRESS = 7'b1001001;//caso 15
#1 ADDRESS = 7'b1010001;//caso 16
#1 ADDRESS = 7'b1011011;//caso 17
```

```
#1 ADDRESS = 7'b0110011;//caso 10
#1 ADDRESS = 7'b0111001;//caso 11
#1 ADDRESS = 7'b1000111;//caso 12
#1 ADDRESS = 7'b1001111;//caso 14
#1 ADDRESS = 7'b0000001;//caso 13
#1 ADDRESS = 7'b1001001;//caso 15
#1 ADDRESS = 7'b1010001;//caso 16
#1 ADDRESS = 7'b1011011;//caso 17
#1 ADDRESS = 7'b1111111;//caso 21
#1 ADDRESS = 7'b1110111;//caso 20
#1 ADDRESS = 7'b1101001;//caso 19
#1 ADDRESS = 7'b1100001;//caso 18
#1 ADDRESS = 7'b0011111;//caso 7
#1 ADDRESS = 7'b0110111;//caso 10
#1 ADDRESS = 7'b0001111;//caso 4
#1 ADDRESS = 7'b1001101;//caso 15
#1 ADDRESS = 7'b1110011;//caso 20
#1 ADDRESS = 7'b1111101;//caso 21
#1 ADDRESS = 7'b1111100;//caso 1

#1 $finish;
end

initial begin
      $dumpfile("ej05_tb.vcd");
      $dumpvars(0, testbench);
    end

endmodule
```