

Lab 09 Digital 1**Ejercicio 1**

```

1  module ffd(input clk, input reset, input en, input d, output reg q);
2
3      always @ (posedge clk, posedge reset)
4          if (reset) q <= 1'b0;
5          else if (en == 1) q <= d;
6
7  endmodule
8
9  module ffd2(input clk, reset, en, input [1:0] d, output [1:0] q);
10
11      ffd a0(clk, reset, en, d[0], q[0]);
12      ffd a1(clk, reset, en, d[1], q[1]);
13
14  endmodule
15
16  module ffd4(input clk, reset, en, input [3:0] d, output [3:0] q);
17
18      ffd a0(clk, reset, en, d[0], q[0]);
19      ffd a1(clk, reset, en, d[1], q[1]);
20      ffd a2(clk, reset, en, d[2], q[2]);
21      ffd a3(clk, reset, en, d[3], q[3]);
22
23  endmodule

```

En el primer ejercicio se realizó un módulo principal, el cual es un flip flop tipo d, con el cual se prosiguieron a realizar otros dos flip flops tipo de d. Los cuales eran de 2 y 4 bits, lo cual se logró llamando el módulo principal las veces necesarias para lograr nuestra cantidad de bits. En el caso del de 2 bits se llamó un total de 2 veces el FFD, mientras que para el de 4 bits se llamó un total de 4 veces, aunque también se pudo llamar el FFD de 2 bits un total de 2 veces para acortar un poco el código.

```

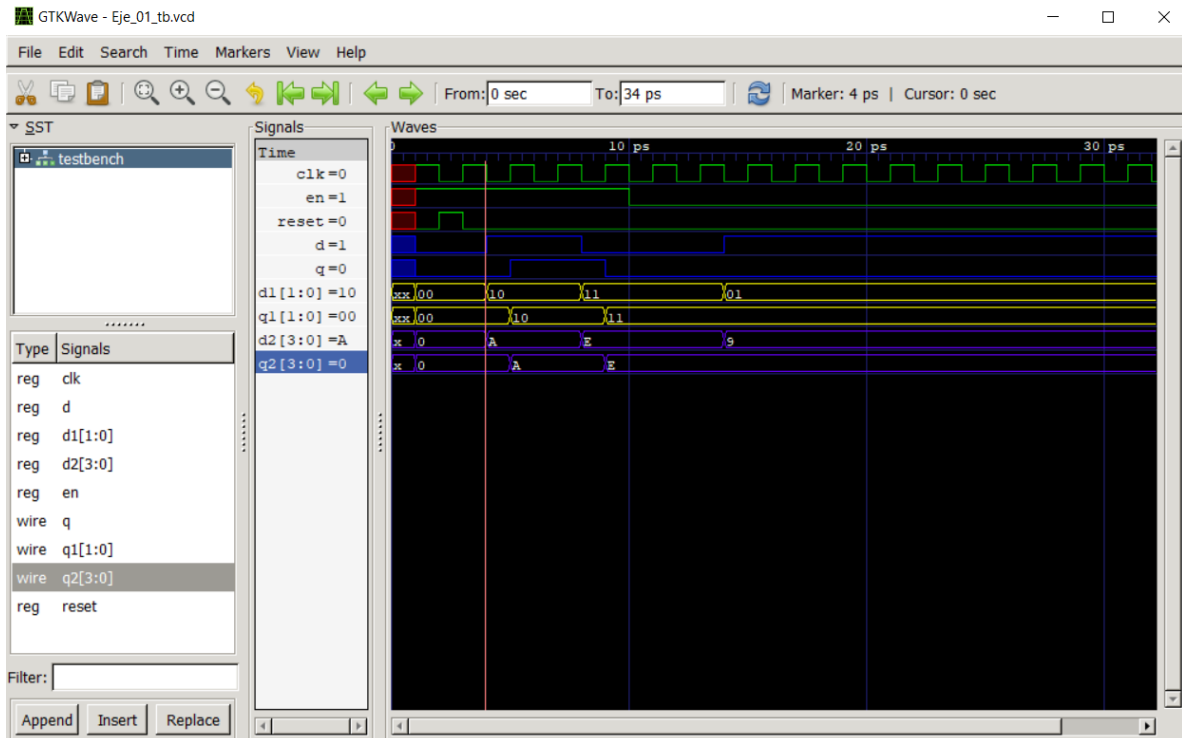
1  module testbench();
2
3      reg clk, reset, en, d;
4      reg [1:0] d1;
5      reg [3:0] d2;
6      wire q;
7      wire [1:0] q1;
8      wire [3:0] q2;
9
10     ffd a1(clk, reset, en, d, q);
11     ffd2 a2(clk, reset, en, d1, q1);
12     ffd4 a3(clk, reset, en, d2, q2);
13
14     //FF de 1 bit
15     initial begin
16
17         #1
18         clk = 0; en = 1; reset = 0; d = 0; d1 = 2'b00; d2 = 4'b0000;
19         $display("Ejercicio 01");
20         $display("| D | Q | 1 bit | D | Q | 2 bits | D | Q | 4 bits ");
21         $monitor("| %b | %b |      | %b | %b |      | %b | %b |", d, q, d1, q1, d2, q2);
22
23         #1 reset = 1;
24         #1 reset = 0;
25         #1 d = 1; d1 = 2'b10; d2 = 4'b1010;
26         #4 d = 0; d1 = 2'b11; d2 = 4'b1110;
27         #2 en = 0;
28         #4 d = 1; d1 = 2'b01; d2 = 4'b1001;
29
30         #20 $finish;
31     end

```

```

32
33     always
34     begin
35         #1 clk = ~clk;
36     end
37
38     initial begin
39         $dumpfile("Eje_01_tb.vcd");
40         $dumpvars(0, testbench);
41     end
42
43
44 endmodule

```

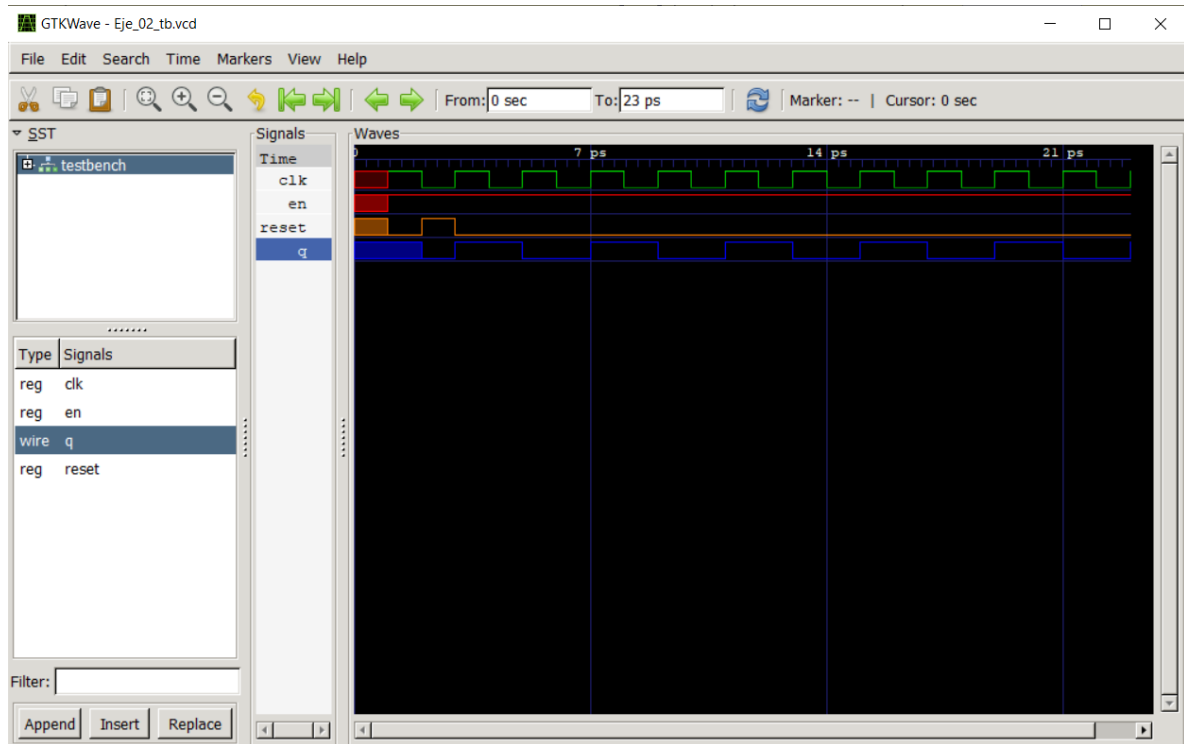


## Ejercicio 2

```
1 module ffd(input clk, input reset, input en, input d, output reg q);
2
3     always @ (posedge clk, posedge reset)
4         if (reset) q <= 1'b0;
5         else if (en) q <= d;
6
7 endmodule
8
9 module fft(input clk, en, reset, output q);
10
11     wire w;
12     assign w = ~q;
13
14     ffd a(clk, reset, en, w, q);
15
16 endmodule
```

Primero se construyo un flip flop tipo d, luego se prosiguió con la construcción del toggle FF. Para lograrlo se utilizó un cable, el cual conecta la salida Q del FFD a una compuerta NOT, la cual entrara en la entrada D del FFD. Logrando así al llamar el módulo del FFD un ToggleFF.

```
1 module testbench();
2
3     reg clk, reset, en;
4     wire q;
5
6     ffd a(clk, en, reset, q);
7
8     initial begin
9
10         #1
11         clk = 0; en = 1; reset = 0;
12         $display("Ejercicio 02: Flip Flop T");
13         $display("| Q |");
14         $monitor("| %b |", q);
15         #1 reset = 1;
16         #1 reset = 0;
17
18         #20 $finish;
19     end
20
21     always
22         #1 clk = ~clk;
23
24     initial begin
25         $dumpfile("Eje_02_tb.vcd");
26         $dumpvars(0, testbench);
27     end
28
29 endmodule
```



### Ejercicio 3

Logic Friday

File Operation Truthtable Equation Gates View Help

Funci... Inputs Outputs True False DC PI Gates

Funci...	Inputs	Outputs	True	False	DC	PI	Gates
D	3	1	4	4	0	2	Not mapped

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

Entered by truthtable:  
 $D = J' K' Q + J K' Q' + J K' Q + J K Q'$

Minimized:  
 $D = K' Q + J Q'$

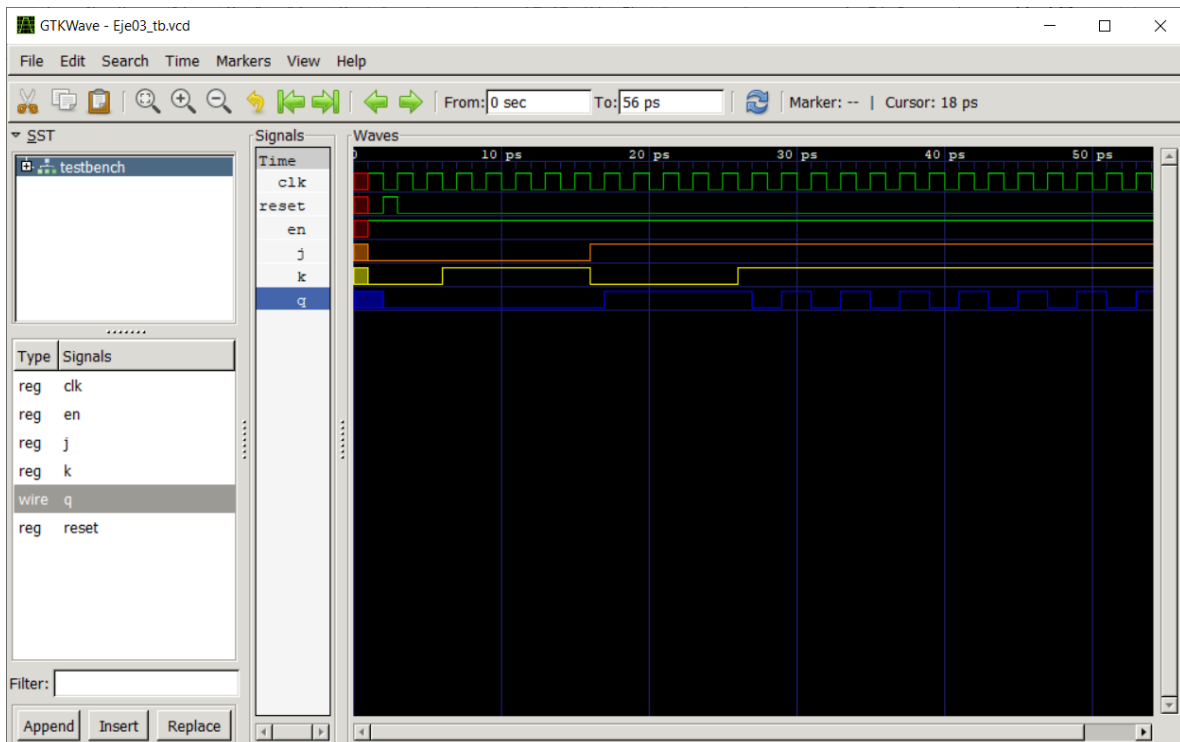
```
1 module ffd(input clk, input reset, input en, input d, output reg q);
2
3     always @ (posedge clk, posedge reset)
4         if (reset) q <= 1'b0;
5         else if (en == 1) q <= d;
6
7     endmodule
8
9 module ffjk(input clk, reset, en, j, k, output q);
10
11     wire d;
12
13     assign d = (~k & q) | (j & ~q);
14
15     ffd a1(clk, reset, en, d, q);
16
17 endmodule
```

Primero se construyó una tabla de verdad, la cual describiría el comportamiento de un FFJK por medio de una nube combinacional. Luego se prosiguió a construir el módulo con el FFD, el cual fue llamado a otro modulo, en donde se primero se aplico la nube combinacional que entraría en la entrada D del FFD y se llamo al módulo el FFD.

```

1  module testbench();
2
3      reg clk, reset, en, j, k;
4      wire q;
5
6      ffjk a1(clk, reset, en, j, k, q);
7
8      initial begin
9          #1
10
11         clk = 0; reset = 0; en = 1; j = 0; k = 0;
12         $display("Ejercicio 3: Flip Flop JK");
13         $display("j | k | Q");
14         $monitor("%b | %b | %b", j, k, q);
15         #1 reset = 1;
16         #1 reset = 0;
17         #3 k = 1;
18         #10 j = 1; k = 0;
19         #10 k = 1;
20
21         #30 $finish;
22     end
23
24     always
25     begin
26         #1 clk = ~clk;
27     end
28
29     initial begin
30         $dumpfile("Eje03_tb.vcd");
31         $dumpvars(0, testbench);
32     end
33

```

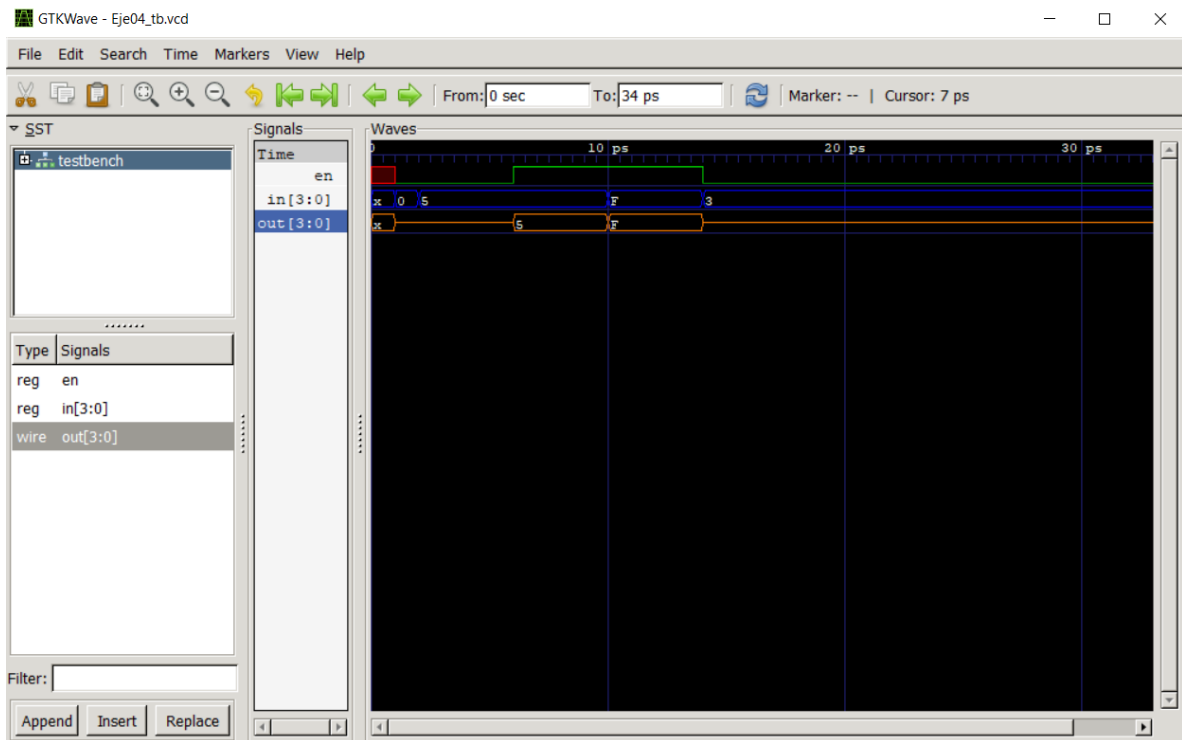


## Ejercicio 4

```
1 module buftri(input wire en, input wire [3:0]in, output wire [3:0]out);
2
3     assign out = (en == 1) ? in :
4                 (en == 0) ? 4'bz : 4'bx;
5
6 endmodule
```

Primero se llamaron las variables, de entrada, tenemos el enable y una de 4 bits llamada in, mientras que de salida tenemos una variable de 4 bits llamada out. Luego empezamos a construir el buffer triestado, asignamos a out un condicional en donde si en es igual a 1 deja pasar a in, de lo contrario estará en alta impedancia.

```
1 module testbench();
2
3     reg en;
4     reg [3:0] in;
5     wire [3:0] out;
6
7     buftri a1(en, in, out);
8
9     initial begin
10
11         #1
12
13         en = 0; in = 4'b0000;
14         $display("Ejercicio 4: Buffer tri estado de de 4 bits");
15         $display("EN | In | OUT");
16         $monitor("%b | %b | %b", en, in, out);
17         #1 in = 4'b0101;
18         #4 en = 1;
19         #4 in = 4'b1111;
20         #4 en = 0; in = 4'b0011;
21
22         #20 $finish;
23     end
24
25     initial begin
26         $dumpfile("Eje04_tb.vcd");
27         $dumpvars(0, testbench);
28     end
29
30 endmodule
```





## Ejercicio 5

```
1 module ROM(input [6:0] d, output reg [12:0] q);
2
3 always @ (d) begin
4
5     case(d)
6         7'b?????0: q = 13'b1000000001000;
7         7'b00001?1: q = 13'b0100000001000;
8         7'b00000?1: q = 13'b1000000001000;
9         7'b00011?1: q = 13'b1000000001000;
10        7'b00010?1: q = 13'b0100000001000;
11        7'b0010??1: q = 13'b0001001000010;
12        7'b0011??1: q = 13'b1001001100000;
13        7'b0100??1: q = 13'b0011010000010;
14        7'b0101??1: q = 13'b0011010100000;
15        7'b0110??1: q = 13'b1011010100000;
16        7'b0111??1: q = 13'b1000000111000;
17        7'b1000?11: q = 13'b0100000001000;
18        7'b1000?01: q = 13'b1000000001000;
19        7'b1001?11: q = 13'b1000000001000;
20        7'b1001?01: q = 13'b0100000001000;
21        7'b1010??1: q = 13'b0011011000010;
22        7'b1011??1: q = 13'b1011011100000;
23        7'b1100??1: q = 13'b0100000001000;
24        7'b1101??1: q = 13'b0000000001001;
25        7'b1110??1: q = 13'b0011100000010;
26        7'b1111??1: q = 13'b1011100100000;
27        default: q = 13'b0;
28
29    endcase
30 end
31
32
33 endmodule
```

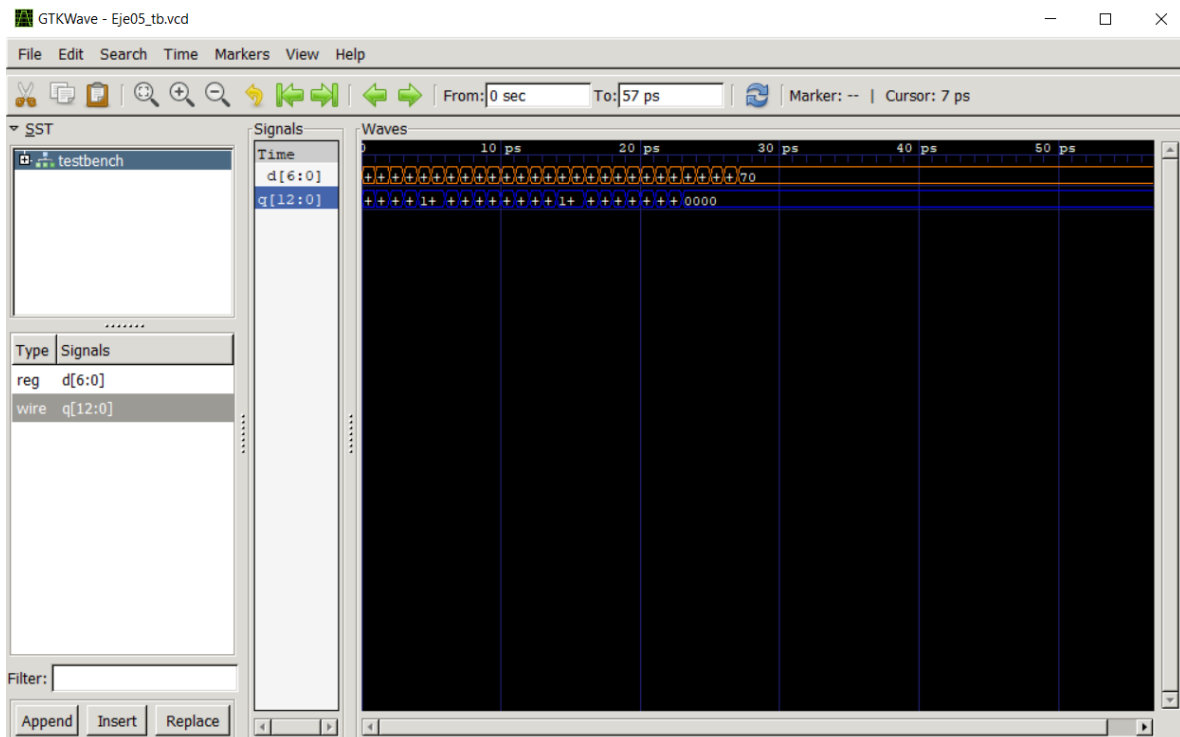
La memoria ROM se construye de manera sencilla con un case, en el cual hay una entrada de 7 bits llamada d y una salida de 13 bits llamada q. En el case se colocaron las 21 combinaciones posibles que existen en la tabla, siendo los don't cares representados en verilog por un signo '?'.

```
1 module testbench();
2
3     reg [6:0] d;
4     wire [12:0] q;
5
6     ROM a1(d, q);
7
8     initial begin
9
10        #1
11        d = 7'b0000000;
12        $display("Ejercicio 5: Memoria Rom con Cases");
13        $display("d | q");
14        $monitor("%b | %b", d, q);
15        #1 d = 7'b?????0;
16        #1 d = 7'b00001?1;
17        #1 d = 7'b00000?1;
18        #1 d = 7'b00011?1;
19        #1 d = 7'b00010?1;
20        #1 d = 7'b0010??1;
21        #1 d = 7'b0011??1;
22        #1 d = 7'b0100??1;
23        #1 d = 7'b0101??1;
24        #1 d = 7'b0110??1;
25        #1 d = 7'b0111??1;
26        #1 d = 7'b1000?11;
27        #1 d = 7'b1000?01;
28        #1 d = 7'b1001?11;
29        #1 d = 7'b1001?01;
30        #1 d = 7'b1010??1;
31        #1 d = 7'b1011??1;
32        #1 d = 7'b1100??1;
33        #1 d = 7'b1101??1;
```

```

34      #1 d = 7'b1110??1;
35      #1 d = 7'b1111??1;
36      #1 d = 7'b0101010;
37      #1 d = 7'b1110001;
38      #1 d = 7'b0000000;
39      #1 d = 7'b1100110;
40      #1 d = 7'b1110000;
41
42      #30 $finish;
43      end
44
45      initial begin
46          $dumpfile("Eje05_tb.vcd");
47          $dumpvars(0, testbench);
48      end
49
50
51      endmodule

```



Link Repositorio:

[https://github.com/fernando19030/LaboratoriosElectronica\\_Digital\\_1-19030.git](https://github.com/fernando19030/LaboratoriosElectronica_Digital_1-19030.git)