Jonathan Emanuel Pu Aguilera
carné 19249

Fecha entrega: 15.08.2020
Sección lab: 11

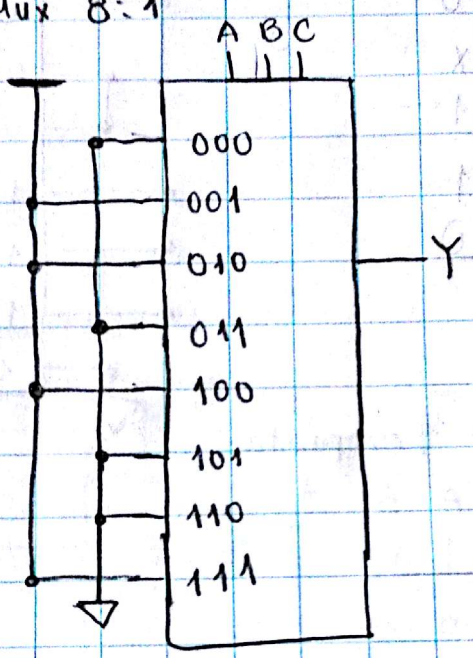
Lab. 05

Ejercicio 01.

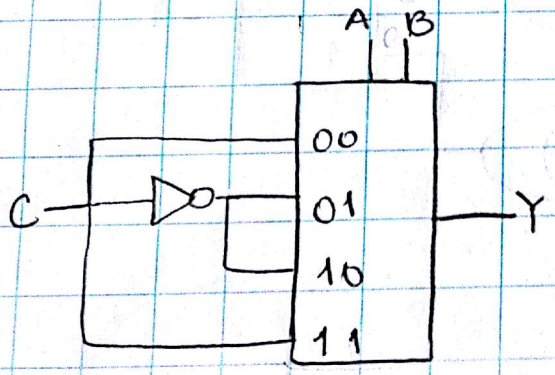
Tabla 01

A	B	C	Y	
0	0	0	0	
0	0	1	1	← XOR
0	1	0	1	← XOR
0	1	1	0	
1	0	0	1	← XNOR
1	0	1	0	XNOR
1	1	0	0	XNOR
1	1	1	1	← XNOR

1. Mux 8:1



2. Mux 4:1 & compuertas



3. Mux 2:1 & compuertas

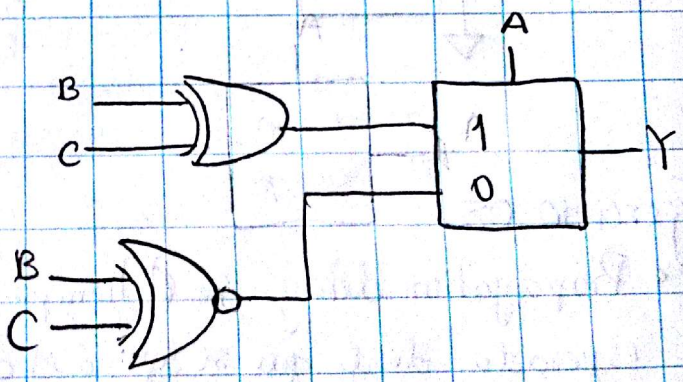
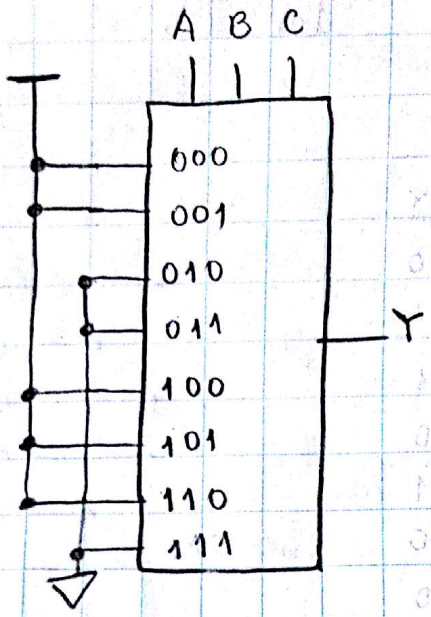


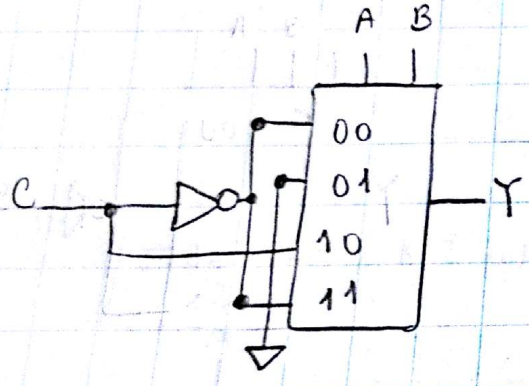
Tabla 02

	A	B	C	Y	
MUX	0	0	0	1	←
	0	0	1	X	→ 0
	0	1	0	0	
	0	1	1	0	
XOR	1	0	0	X	0
	1	0	1	1	←
	1	1	0	1	←
	1	1	1	0	

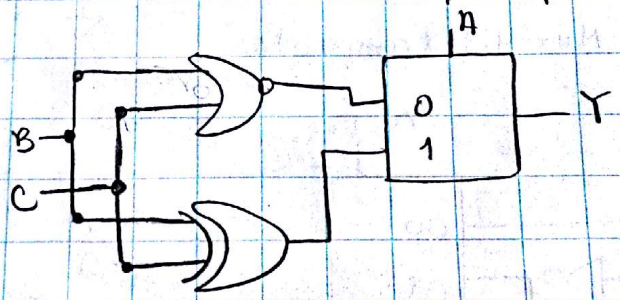
1. Mux 8:1



2. Mux 4:1 & compuertas



3. Mux 2:1 & compuertas



Ejercicio 05.

- Propagation delay: es el tiempo que toma un circuito en devolver la salida esperada desde que se aplicó el cambio en la entrada.
- Contamination delay: es el tiempo que un circuito comienza a cambiar su salida pero sin estabilizarse en el estado esperado.
- Ruta crítica: es el camino más largo que toma una señal en el circuito hasta llegar a la salida, luego de pasar por los componentes en esa ruta.
- Ruta corta: es el camino más corto que tiene una señal hasta la salida, esto no significa que en esa ruta se llegue al punto estable en la salida.

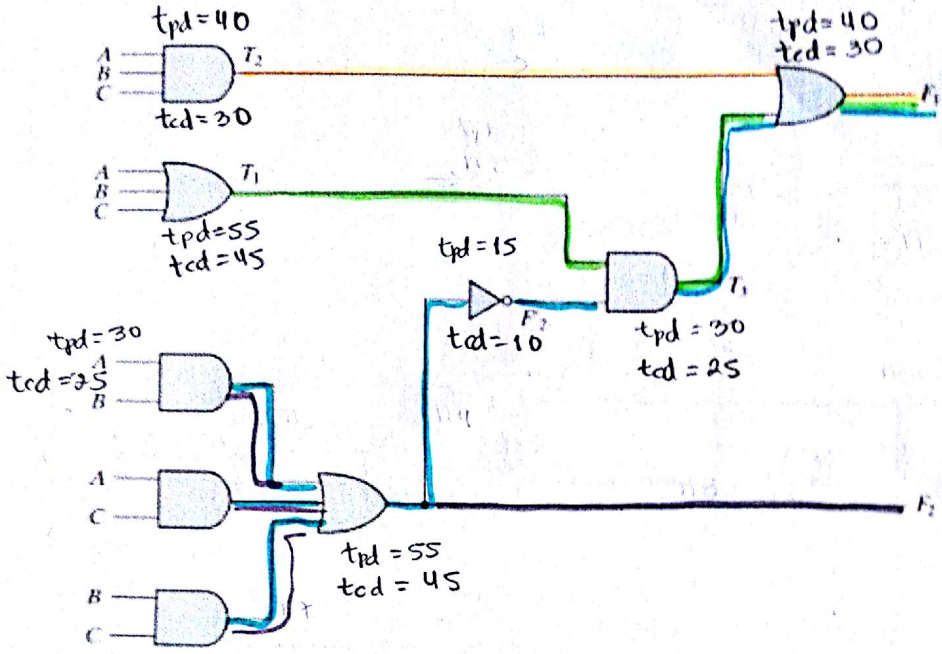


Figure 2: Circuito 01

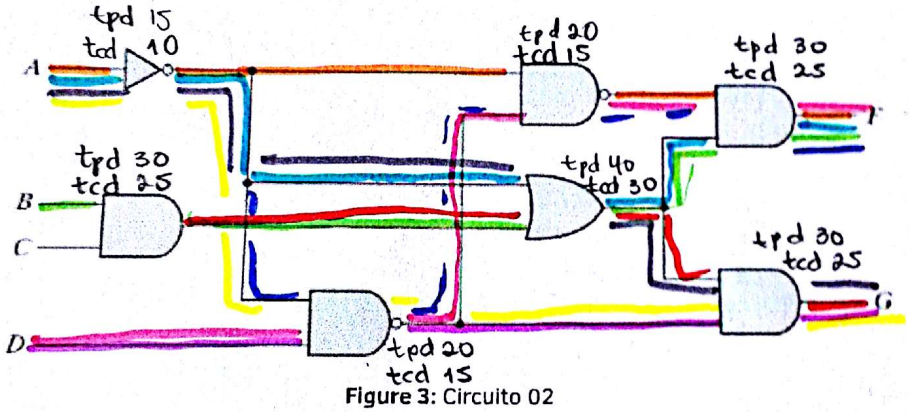


Figure 3: Circuito 02

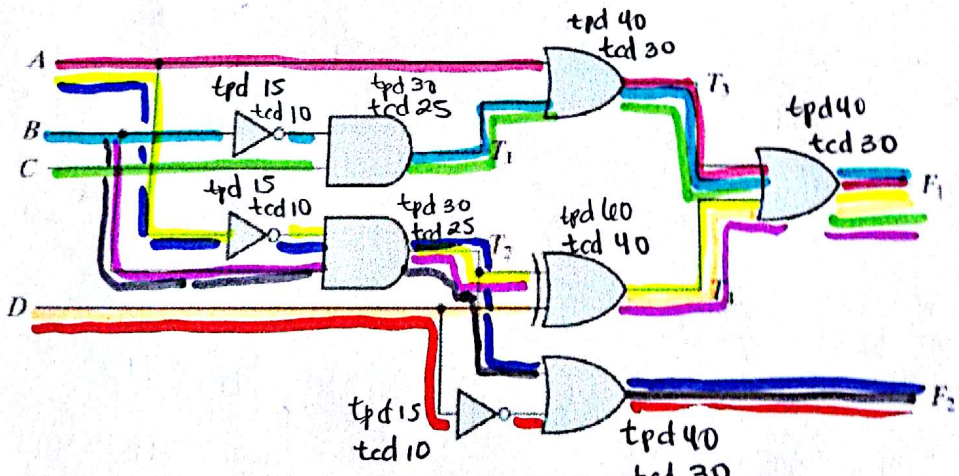


Figure 4: Circuito 03

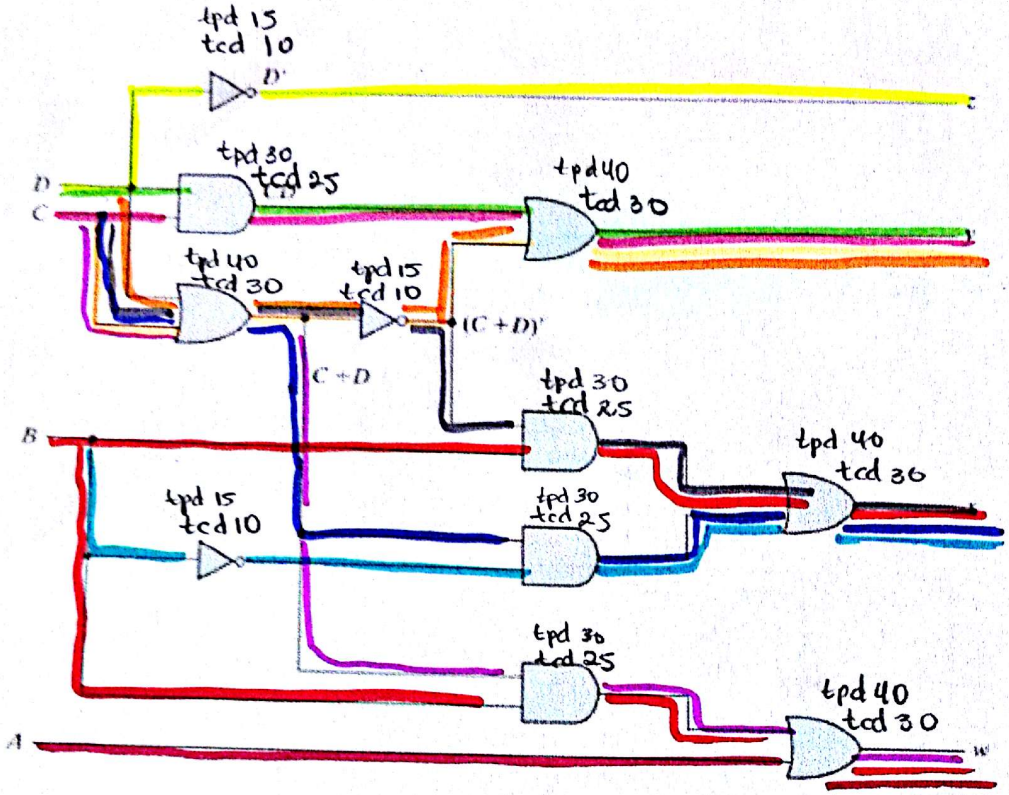


Figure 5: Circuito 04

Ejercicio 06

Circuito 01 — Salida F1

- 80ps tpd — 60ps tcd ← ruta corta
- 125ps tpd — 100ps tcd
- 170ps tpd — 135ps tcd [3]

{ 3 rutas críticas

Salida F2

85 ps tpd — 70ps tcd [3]

3 rutas críticas — 3 rutas cortas

Circuito 02 — salida F

- 65ps tpd — 50ps tcd
- 85ps tpd — 65ps tcd
- 95ps tpd — 65ps tcd
- ruta crítica 100ps tpd — 80ps tcd
- 70ps tpd — 55ps tcd ruta corta

Salida G

- 65ps tpd — 50ps tcd
- ruta crítica 100ps tpd — 80ps tcd
- 50ps tpd — 40 ps tcd ruta corta
- 85ps tpd — 65ps tcd

Circuito 03 — salida F1 — ruta corta

- 80ps tpd — 60ps tcd — 110ps tpd 85ps tcd
- ruta crítica 145ps tpd — 105ps tcd
- 125ps tpd — 95ps tcd
- 130ps tpd — 95ps tcd
- 100ps tpd — 70ps tcd

Salida F2

- 55ps tpd — 40ps tcd ruta corta
- ruta crítico 85ps tpd — 65ps tcd
- 70ps tpd — 55ps tcd

Circuito 04

Salida z

ruta crítica 15 ps tpd — 10 ps tcd ruta corta

Salida y

70 ps tpd — 55 ps tcd [2] 2 rutas cortas

2 rutas críticas 95 ps tpd — 70 ps tcd [2]

Salida x

2 rutas críticas 125 ps tpd — 95 ps tcd [2]

70 ps tpd — 55 ps tcd ruta corta

110 ps tpd — 85 ps tcd [2]

85 ps tpd — 65 ps tcd

Salida w

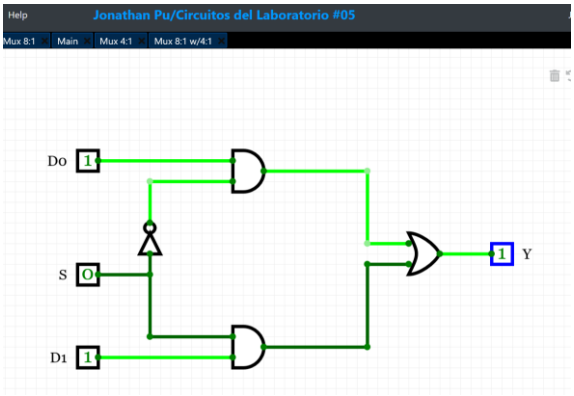
2 rutas críticas 110 ps tpd — 85 ps tcd [2]

70 ps tpd — 55 ps tcd

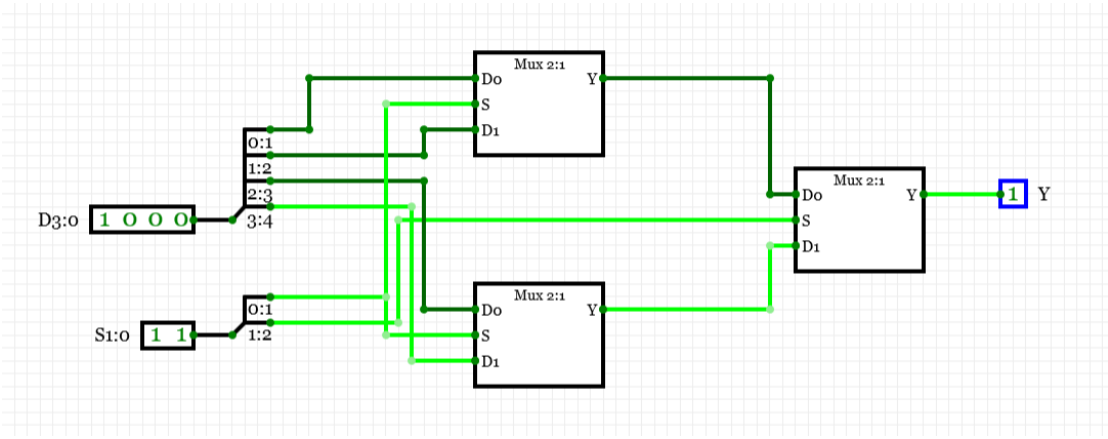
40 ps tpd — 30 ps tcd ruta corta

Universidad del Valle de Guatemala
Digital 1
Jonathan Emanuel Pu Aguilera

Fecha entrega: 16.08.2020
Sección lab: 11
Carné: 19249

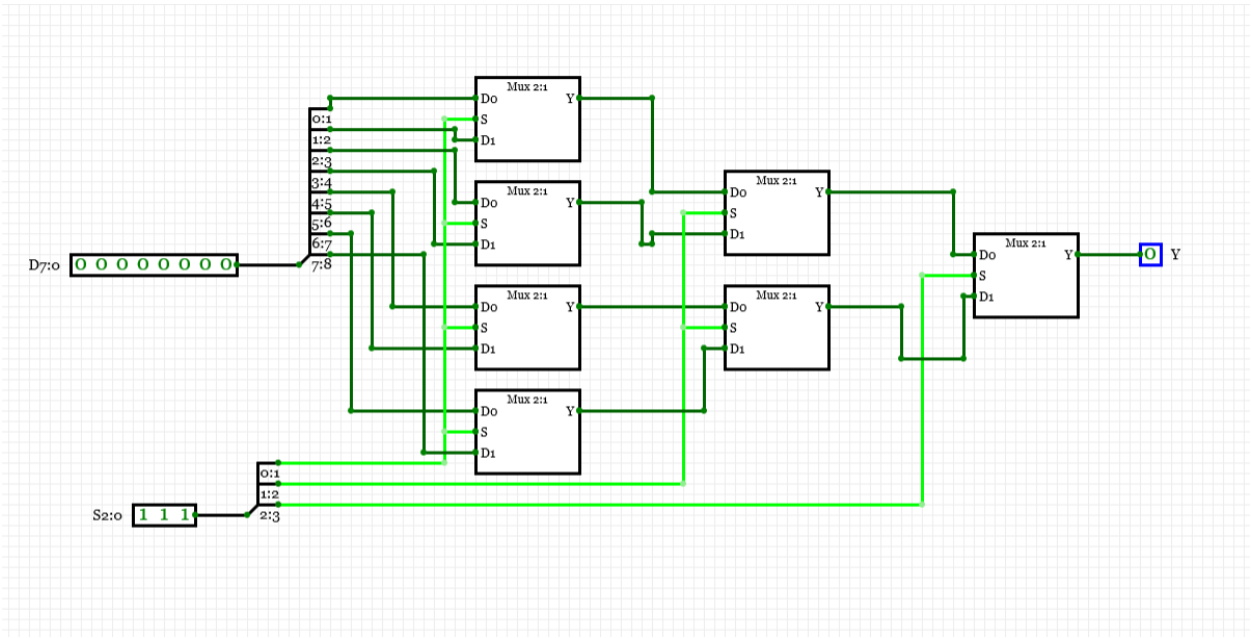


Mux 2:1



Mux 4:1

Mux 8:1 usando 2:1



Mux 8:1 usando 4:1

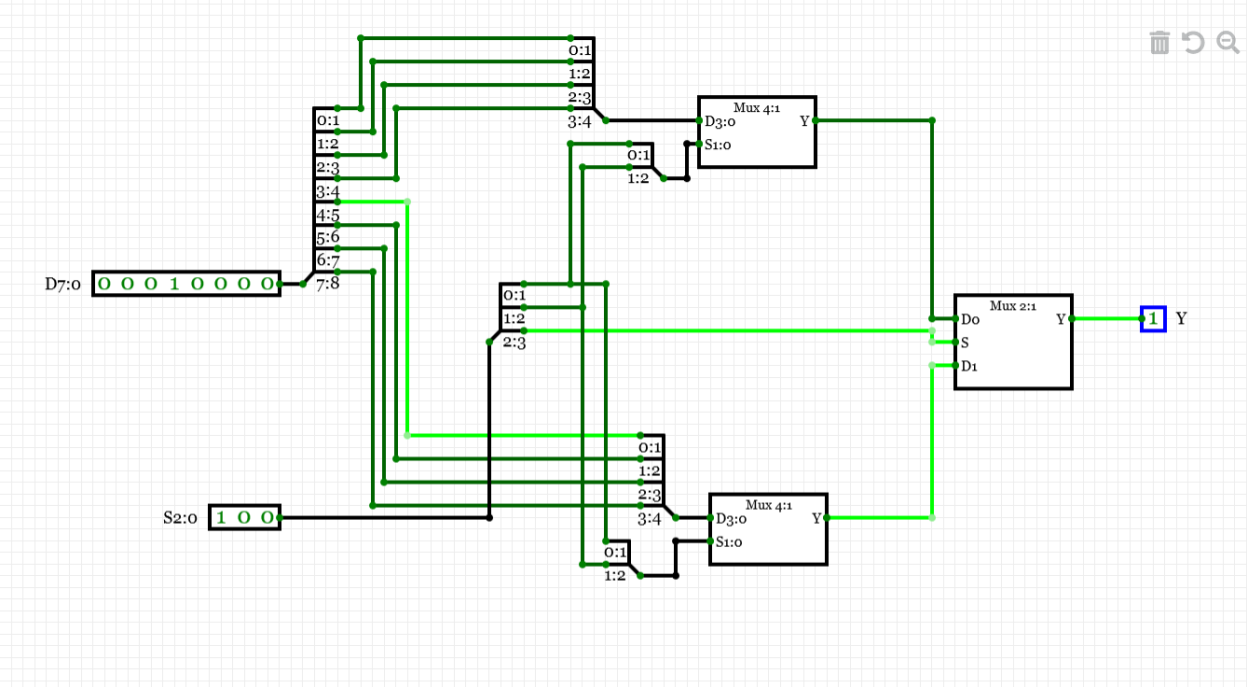
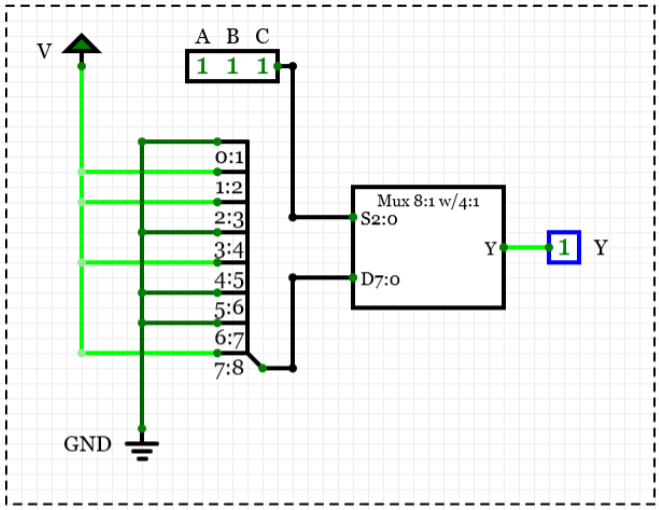


Tabla 01. MUX 8:1



Tablao1. MUX 4:1

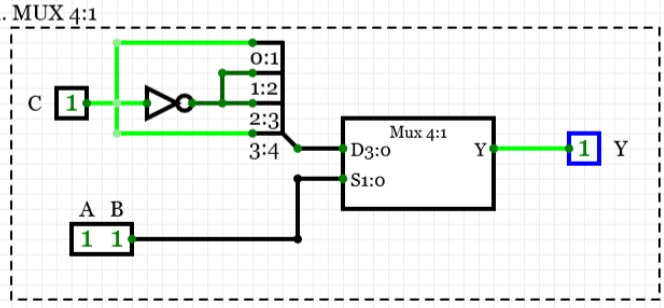


Tabla01. MUX 2:1

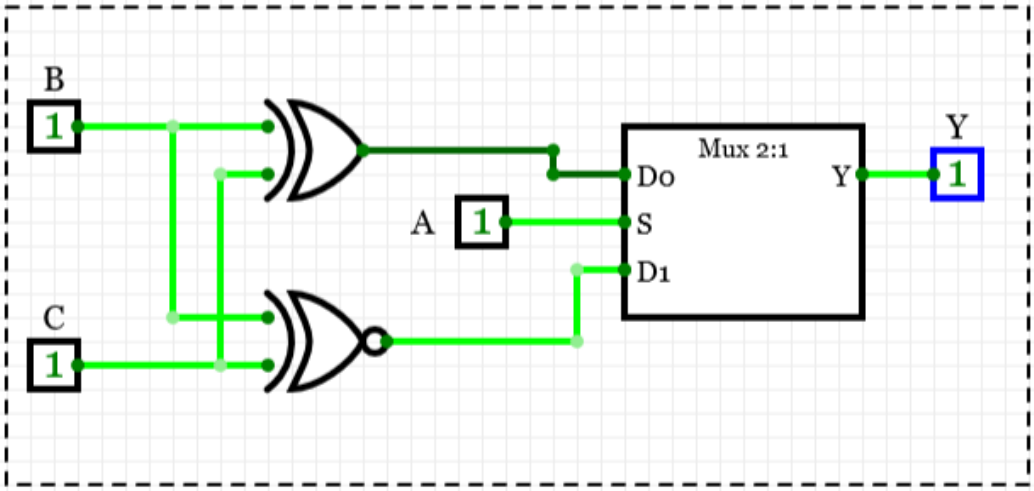


Tabla02. MUX 8:1

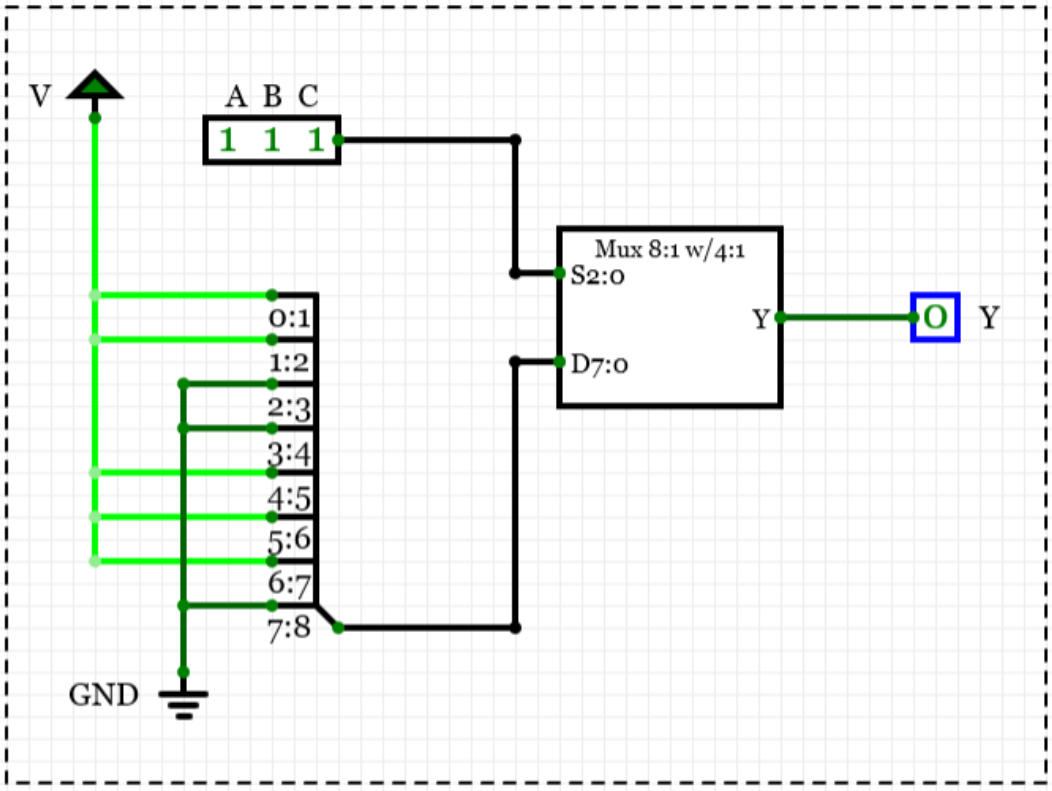


Tabla02. MUX 4:1

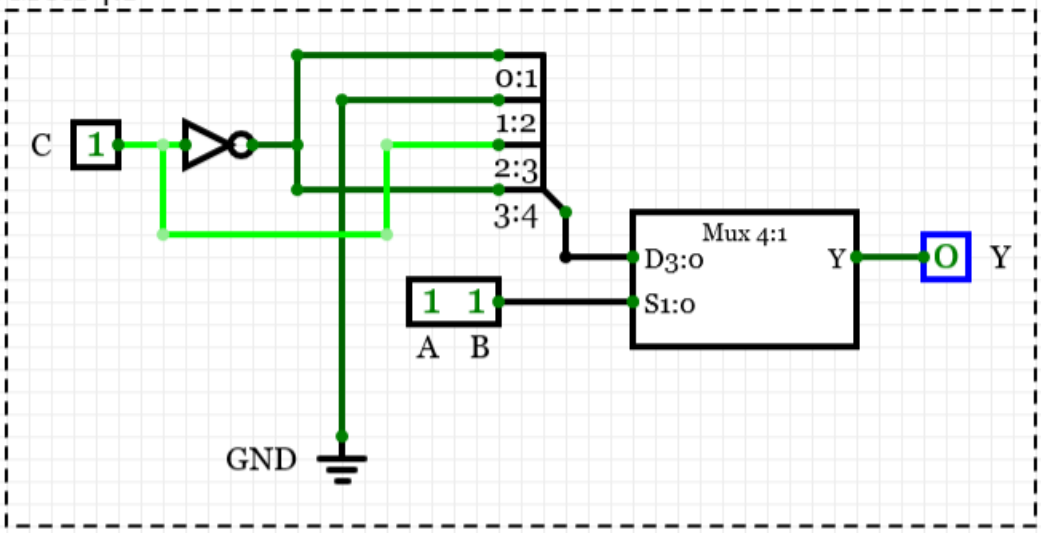


Tabla02. MUX 2:1

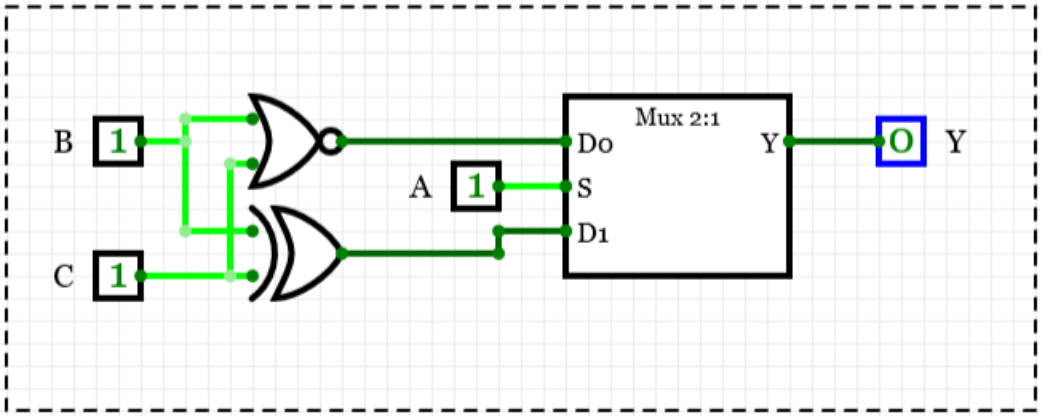


Tabla01. Decoder

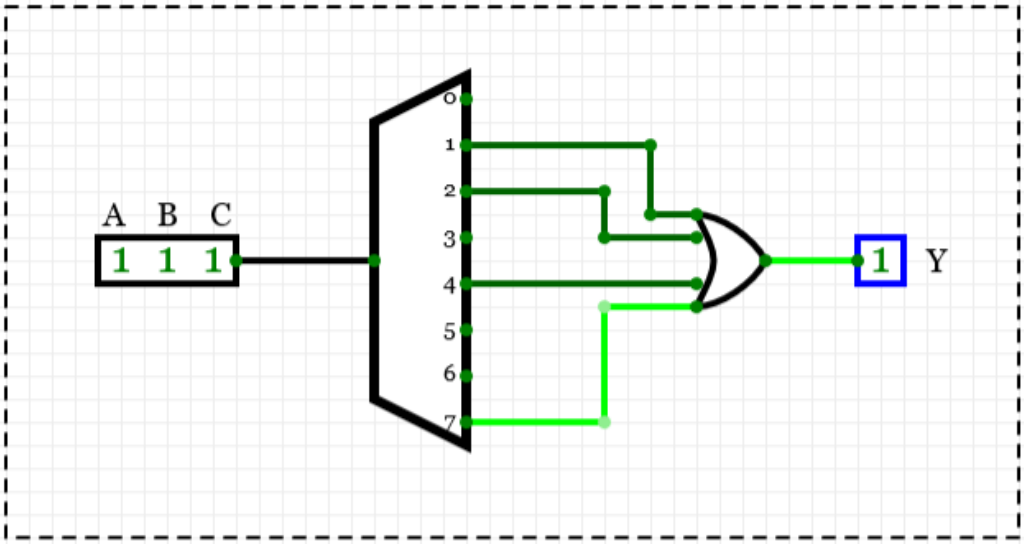
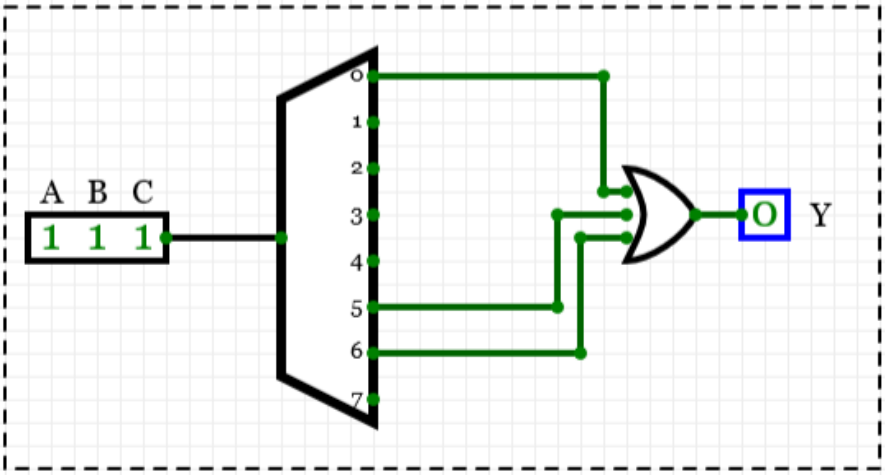


Tabla02. Decoder



Código para el mux 2x1

```
//Modulo mux 2:1
module Mux_2x1(output wire Y, input wire D0, D1, S);

    assign Y = S ? D1 : D0; //Si S es 0, D0 se cumple, si S es falso D1 se cumple
endmodule
```

Código para el mux 4x1

```
//Modulo mux 4:1
module Mux_4x1(output wire Y2, input wire D0, D1, D2, D3, S0, S1);
    //declarar los cables que van a conectar los muxes2x1 entre si
    wire Y0, Y1;
    //utilizar el modulo 2x1 para conectar los selectores
    Mux_2x1 c1(Y0, D0, D1, S0); //cable de subsalida Y0, entra selector S0 y D0, D1
    Mux_2x1 c2(Y1, D2, D3, S0); //cable de subsalida Y1, entra selector S0, D2, D3
    Mux_2x1 c3(Y2, Y0, Y1, S1); //cable de salida Y, entra selector S1, y entran las salidas de los anteriores

endmodule
```

Código para el mux 8x1

```
//Modulo mux 8:1
module Mux_8x1(output wire Y5, input wire D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2);
    //declarar los subcables de los muxies4x1
    wire Y3, Y4;
    //utilizar dos modulos 4x1 primero
    Mux_4x1 c4(Y3, D0, D1, D2, D3, S0, S1);
    Mux_4x1 c5(Y4, D4, D5, D6, D7, S0, S1);
    //utilizar un modulo 2x1
    Mux_2x1 c6(Y5, Y3, Y4, S2);

endmodule
```

Código de implementación Tabla01 con 8x1

```
//Implementacion del ejercicio01 con muxes generados
//
//Tabla01 mux 8:1
module Tabla01_8x1(output wire Y6, input wire inA, inB, inC);
    //utilizar el modulo 8x1
    wire GND, V;
    assign GND = 0;
    assign V = 1;
    Mux_8x1 T1_8x1(Y6, GND, V, V, GND, V, GND, GND, V, inA, inB, inC);

endmodule
```

```
module testbench();

//Tabla01 con mux 8x1
    reg p1, p2, p3;
    wire out1;
    //mi modulo Tabla01_8x1 recibe la salida primero, luego las entradas
    Tabla01_8x1 T1a(out1, p1, p2, p3);

    initial begin
        #1
        $display("\n");
        $display(" Tabla01 8x1 ");
        $display("A  B  C  |  Y");
        $display("---------|---");
        $monitor("%b  %b  %b  |  %b", p1, p2, p3, out1);
            //entradas comienzan todas en 0
            p1  = 0; p2 = 0; p3 = 0;
        #1 p1 = 0; p2 = 0; p3 = 1;
        #1 p1 = 0; p2 = 1; p3 = 0;
        #1 p1 = 0; p2 = 1; p3 = 1;
        #1 p1 = 1; p2 = 0; p3 = 0;
        #1 p1 = 1; p2 = 0; p3 = 1;
        #1 p1 = 1; p2 = 1; p3 = 0;
        #1 p1 = 1; p2 = 1; p3 = 1;

    end
```

Código de implementación Tabla01 con 4x1

```
//Tabla01 mux 4:1
module Tabla01_4x1(output wire Y7, input wire inA, inB, inC);
    //declaracion de cables para las entradas de C
    wire N1;
    assign N1 = ~ inC;
    //utilizar el modulo 4x1
    Mux_4x1 T1_4x1 (Y7, inC, N1, N1, inC, inA, inB);

endmodule
```

```
//Tabla01 con mux 4x1
    reg p4, p5, p6;
    wire out2;

    Tabla01_4x1 T1b(out2, p4, p5, p6);
    initial begin
        #9
        $display("\n");
        $display(" Tabla01 4x1 ");
        $display("A  B  C  |  Y");
        $display("---------|---");
        $monitor("%b  %b  %b  |  %b", p4, p5, p6, out2);
            //entradas comienzan todas en 0
            p4  = 0; p5 = 0; p6 = 0;
        #1 p4 = 0; p5 = 0; p6 = 1;
        #1 p4 = 0; p5 = 1; p6 = 0;
        #1 p4 = 0; p5 = 1; p6 = 1;
        #1 p4 = 1; p5 = 0; p6 = 0;
        #1 p4 = 1; p5 = 0; p6 = 1;
        #1 p4 = 1; p5 = 1; p6 = 0;
        #1 p4 = 1; p5 = 1; p6 = 1;

    end
```

Código de implementación Tabla01 con 2x1

```
//Tabla01 mux 2:1
module Tabla01_2x1(output wire Y8, input wire inA, inB, inC);
    wire compuerta1, compuerta2;
    assign compuerta1 = (inB) ^ (inC);
    assign compuerta2 = (inB) ~^ (inC);
    //utilizar el modulo 2x1
    Mux_2x1 T1_2x1(Y8, compuerta1, compuerta2, inA);

endmodule
```

```
//Tabla01 con mux 2x1
    reg p7, p8, p9;
    wire out3;

    Tabla01_2x1 T1c(out3, p7, p8, p9);
    initial begin
        #17
        $display("\n");
        $display(" Tabla01 2x1 ");
        $display("A  B  C  |  Y");
        $display("---------|---");
        $monitor("%b  %b  %b  |  %b", p7, p8, p9, out3);
            //entradas comienzan todas en 0
            p7  = 0; p8 = 0; p9 = 0;
        #1 p7 = 0; p8 = 0; p9 = 1;
        #1 p7 = 0; p8 = 1; p9 = 0;
        #1 p7 = 0; p8 = 1; p9 = 1;
        #1 p7 = 1; p8 = 0; p9 = 0;
        #1 p7 = 1; p8 = 0; p9 = 1;
        #1 p7 = 1; p8 = 1; p9 = 0;
        #1 p7 = 1; p8 = 1; p9 = 1;

    end
```

Código de implementación Tabla02 con 8x1

```
//Tabla02 mux 8:1
module Tabla02_8x1(output wire Y9, input wire inA, inB, inC);
    //declarar tierra y voltaje
    wire GND, V;
    assign GND = 0;
    assign V = 1;
    //utilizar el modulo 8x1 (recibe primero la salida)
    Mux_8x1 T2_8x1(Y9, V, V, GND, GND, V, V, V, GND, inC, inB, inA);

endmodule
```

```
//Tabla02 con mux 8x1
    reg p10, p11, p12;
    wire out4;

    Tabla02_8x1 T2a(out4, p10, p11, p12); //se tomaron las x's como 1, 1
    initial begin
        #25
        $display("\n");
        $display(" Tabla02 8x1 ");
        $display("A  B  C  |  Y");
        $display("---------|---");
        $monitor("%b  %b  %b  |  %b", p10, p11, p12, out4);
            //entradas comienzan todas en 0
            p10  = 0; p11 = 0; p12 = 0;
        #1 p10 = 0; p11 = 0; p12 = 1;
        #1 p10 = 0; p11 = 1; p12 = 0;
        #1 p10 = 0; p11 = 1; p12 = 1;
        #1 p10 = 1; p11 = 0; p12 = 0;
        #1 p10 = 1; p11 = 0; p12 = 1;
        #1 p10 = 1; p11 = 1; p12 = 0;
        #1 p10 = 1; p11 = 1; p12 = 1;
    end
```

Código de implementación Tabla02 con 4x1

```
//Tabla02 mux 4:1
module Tabla02_4x1(output wire Y10, input wire inA, inB, inC);
    //declarar el cable negado C y tierra
    wire NC, GND;
    assign NC  = ~(inC);
    assign GND = 0;
    //utilizar el modulo 4x1
    Mux_4x1 T2_4x1(Y10, NC, GND, inC, NC, inB, inA);

endmodule
```

```
//Tabla02 con mux 4x1
    reg p13, p14, p15;
    wire out5;

    Tabla02_4x1 T2b(out5, p13, p14, p15); //se tomaron las x's como 0, 0
    initial begin
        #33
        $display("\n");
        $display(" Tabla02 4x1 ");
        $display("A  B  C  |  Y");
        $display("---------|---");
        $monitor("%b  %b  %b  |  %b", p13, p14, p15, out5);
            //entradas comienzan todas en 0
            p13  = 0; p14 = 0; p15 = 0;
        #1 p13 = 0; p14 = 0; p15 = 1;
        #1 p13 = 0; p14 = 1; p15 = 0;
        #1 p13 = 0; p14 = 1; p15 = 1;
        #1 p13 = 1; p14 = 0; p15 = 0;
        #1 p13 = 1; p14 = 0; p15 = 1;
        #1 p13 = 1; p14 = 1; p15 = 0;
        #1 p13 = 1; p14 = 1; p15 = 1;
    end
```

Código de implementación Tabla02 con 2x1

```
//Tabla02 mux 2:1
module Tabla02_2x1(output wire Y11, input wire inA, inB, inC);
    //necesito declarar dos compuertas
    wire compuerta1, compuerta2;
    assign compuerta1 = (inB) ~| (inC); //compuerta NOR
    assign compuerta2 = (inB) ^ (inC); //compuerta XOR
    //utilizar modulo 2x1
    Mux_2x1 T2_2x1(Y11, compuerta1, compuerta2, inA);

endmodule
```

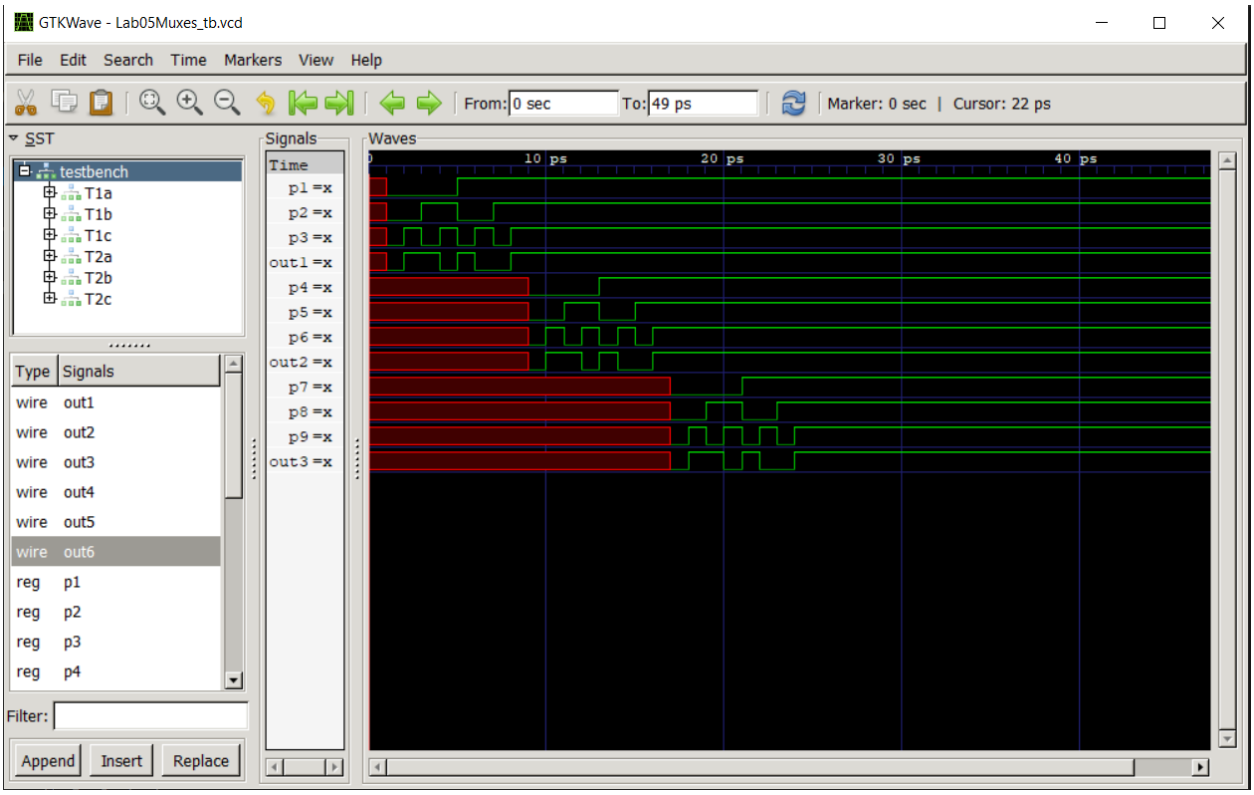
```
//Tabla02 con mux 2x1
    reg p16, p17, p18;
    wire out6;

    Tabla02_2x1 T2c(out6, p16, p17, p18); //se tomaron las x's como 0, 0
    initial begin
        #41
        $display("\n");
        $display(" Tabla02 2x1 ");
        $display("A  B  C  |  Y");
        $display("---------|---");
        $monitor("%b  %b  %b  |  %b", p16, p17, p18, out6);
            //entradas comienzan todas en 0
            p16  = 0; p17 = 0; p18 = 0;
        #1 p16 = 0; p17 = 0; p18 = 1;
        #1 p16 = 0; p17 = 1; p18 = 0;
        #1 p16 = 0; p17 = 1; p18 = 1;
        #1 p16 = 1; p17 = 0; p18 = 0;
        #1 p16 = 1; p17 = 0; p18 = 1;
        #1 p16 = 1; p17 = 1; p18 = 0;
        #1 p16 = 1; p17 = 1; p18 = 1;
    end
```

```
//Fin del codigo
    initial
        #49 $finish;

//GTK Wave
    initial begin
        $dumpfile("Lab05Muxes_tb.vcd"); //nombre del archivo
        $dumpvars(0, testbench); //Nombre de este modulo
    end

//fin del modulo para las 6 tablas
endmodule
```

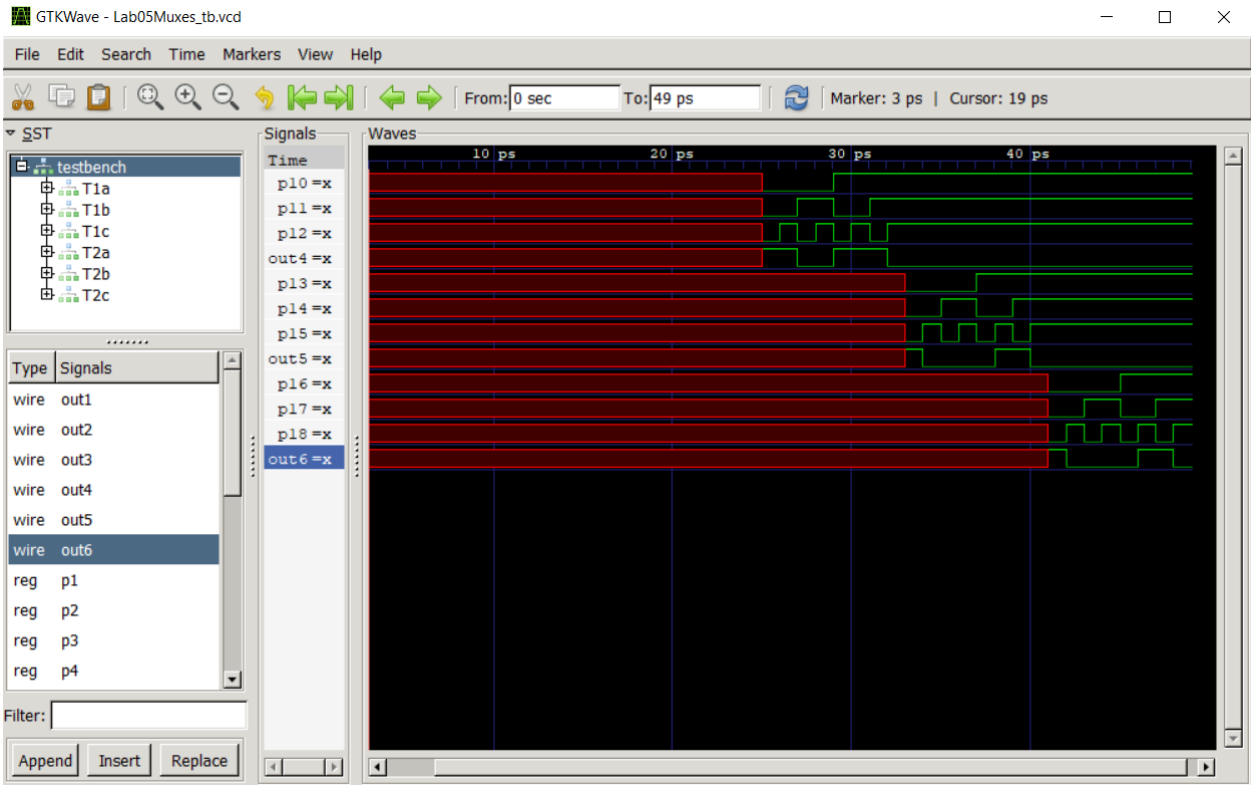
```
PS C:\Users\jpu20\Documents\GitHub\Repositorio D1 19249\Lab05 D1> apio sim

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

iverilog -o Lab05Muxes_tb.out -D VCD_OUTPUT=Lab05Muxes_tb C:/Users/jpu20/.apio\packages\toolchain-yosys\share\yosys/ice40/cells_sim.v Lab05Muxes.v Lab05Muxes_tb.v
vvp Lab05Muxes_tb.out
VCD info: dumpfile Lab05Muxes_tb.vcd opened for output.


Tabla01 8x1
A  B  C  |  Y
---------|---
0  0  0  |  0
0  0  1  |  1
0  1  0  |  1
0  1  1  |  0
1  0  0  |  1
1  0  1  |  0
1  1  0  |  0
1  1  1  |  1


Tabla01 4x1
A  B  C  |  Y
---------|---
0  0  0  |  0
0  0  1  |  1
0  1  0  |  1
0  1  1  |  0
1  0  0  |  1
1  0  1  |  0
1  1  0  |  0
1  1  1  |  1


Tabla01 2x1
A  B  C  |  Y
---------|---
0  0  0  |  0
0  0  1  |  1
0  1  0  |  1
0  1  1  |  0
1  0  0  |  1
1  0  1  |  0
1  1  0  |  0
1  1  1  |  1
```

```
Tabla02 8x1
A  B  C  |  Y
---------|---
0  0  0  |  1
0  0  1  |  1
0  1  0  |  0
0  1  1  |  0
1  0  0  |  1
1  0  1  |  1
1  1  0  |  1
1  1  1  |  0


Tabla02 4x1
A  B  C  |  Y
---------|---
0  0  0  |  1
0  0  1  |  0
0  1  0  |  0
0  1  1  |  0
1  0  0  |  0
1  0  1  |  1
1  1  0  |  1
1  1  1  |  0


Tabla02 2x1
A  B  C  |  Y
---------|---
0  0  0  |  1
0  0  1  |  0
0  1  0  |  0
0  1  1  |  0
1  0  0  |  0
1  0  1  |  1
1  1  0  |  1
1  1  1  |  0
gtkwave Lab05Muxes_tb.vcd Lab05Muxes_tb.gtkw
```

GTK wave para la tabla01



GTK Wave para la tabla02



https://github.com/pu19249/Repositorio-D1-19249.git