Universidad del Valle de Guatemala

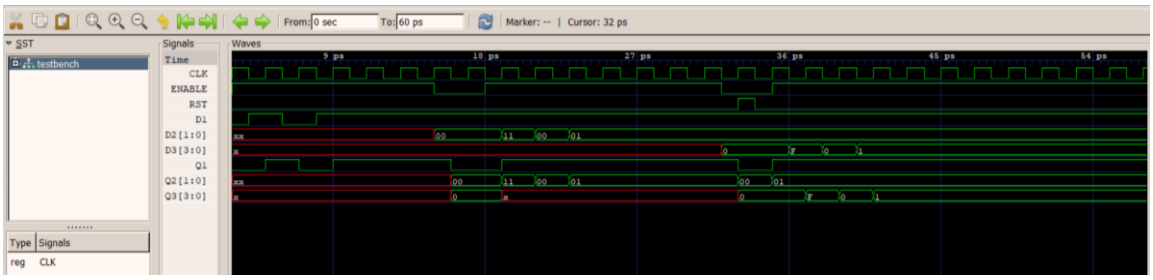
Digital 1

Rodrigo García 19085

Laboratorio #09

Link Repositorio:https://github.com/gar19085/Digital-1-Garcia19085/tree/master/Laboratorios/Lab09Digital1Gar19085

Ejercicio #01:



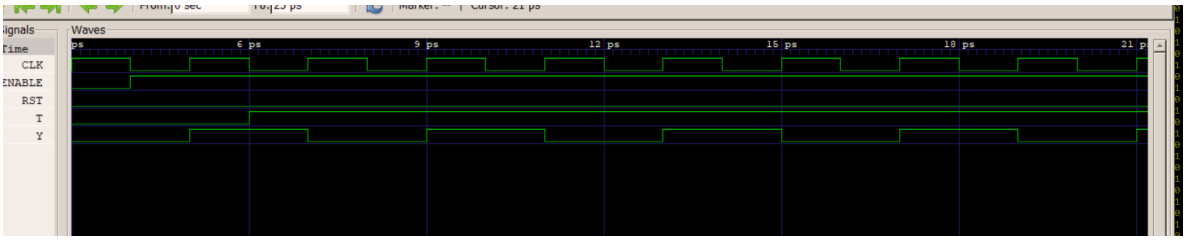

```
FFD 1 BIT
CLK | RST | ENABLE | D1 | Q1
VCD info: dumpfile FFD_tb.vcd opened for output.
0   | 0   |    0   | 0  | x
1   | 1   |    0   | 0  | 0
0   | 0   |    0   | 0  | 0
1   | 0   |    1   | 0  | 0
0   | 0   |    1   | 1  | 0
1   | 0   |    1   | 1  | 1
0   | 0   |    1   | 0  | 1
1   | 0   |    1   | 0  | 0
0   | 0   |    1   | 1  | 0
1   | 0   |    1   | 1  | 1
0   | 0   |    1   | 1  | 1
1   | 0   |    1   | 1  | 1
0   | 0   |    1   | 1  | 1
1   | 0   |    1   | 1  | 1
0   | 0   |    1   | 1  | 1
```

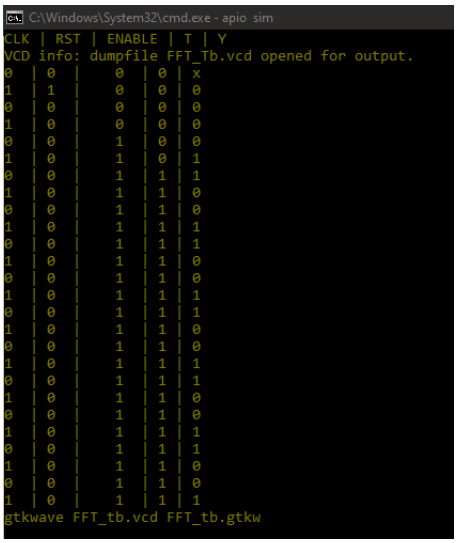
```
FFD 2 BIT
CLK | RST | ENABLE | D2 | Q2
1   | 0   |    0   | 00 |  xx
0   | 0   |    0   | 00 |  00
1   | 0   |    0   | 00 |  00
0   | 0   |    1   | 00 |  00
1   | 0   |    1   | 11 |  11
0   | 0   |    1   | 11 |  11
1   | 0   |    1   | 00 |  00
0   | 0   |    1   | 00 |  00
1   | 0   |    1   | 01 |  01
0   | 0   |    1   | 01 |  01
1   | 0   |    1   | 01 |  01
0   | 0   |    1   | 01 |  01
1   | 0   |    1   | 01 |  01
0   | 0   |    1   | 01 |  01
1   | 0   |    1   | 01 |  01
0   | 0   |    1   | 01 |  01
1   | 0   |    1   | 01 |  01
```

```
FFD 4 BIT
CLK | RST | ENABLE |  D3  | Q3
0   | 0   |    0   | 0000 |  xxxx
1   | 1   |    0   | 0000 |  0000
0   | 0   |    0   | 0000 |  0000
1   | 0   |    1   | 0000 |  0000
0   | 0   |    1   | 1111 |  0000
1   | 0   |    1   | 1111 |  1111
0   | 0   |    1   | 0000 |  1111
1   | 0   |    1   | 0000 |  0000
0   | 0   |    1   | 0001 |  0000
1   | 0   |    1   | 0001 |  0001
0   | 0   |    1   | 0001 |  0001
1   | 0   |    1   | 0001 |  0001
0   | 0   |    1   | 0001 |  0001
1   | 0   |    1   | 0001 |  0001
0   | 0   |    1   | 0001 |  0001
1   | 0   |    1   | 0001 |  0001
0   | 0   |    1   | 0001 |  0001
1   | 0   |    1   | 0001 |  0001
0   | 0   |    1   | 0001 |  0001
1   | 0   |    1   | 0001 |  0001
```

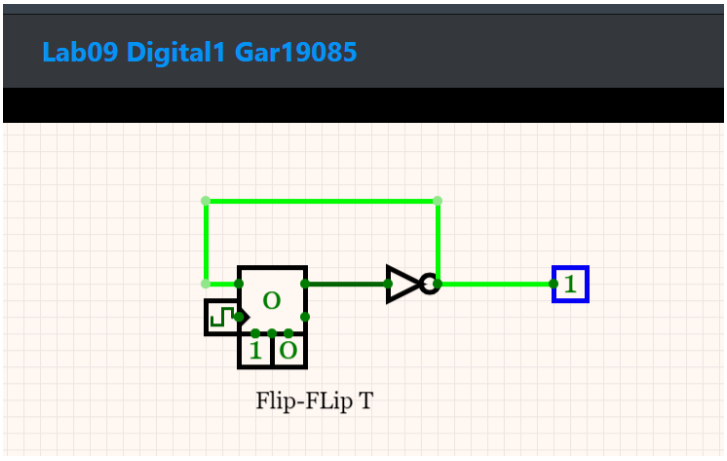
Se creo un modulo para el flip flip tipo D en el cual se indica en que momento sucede una acción dependiendo del flanco de reloj, se establecieron los parámetros de funcionamiento del flip flip para su funcionamiento con el enable y el reset. Luego para implementar el Flip Flip de 2 bits solo se llamo al modulo del FF de 1 bit, luego para el Flip Flop de 4 bits se hizo lo mismo solo que se llamo dos veces al modulo del Flip Flop de 2 bits, el cual también puede realizarse llamando 4 veces al modulo del Flip Flop de 1 bit.

Ejercicio #02:

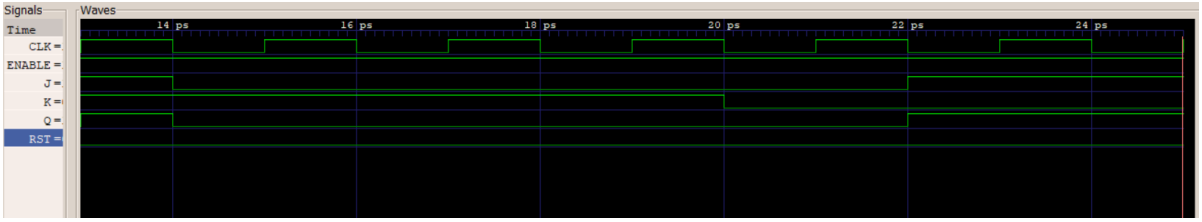




Se creo un modulo para un flip flop de 1 bit, para así implementar el Flip-Flop T en el cual solo se agrega una compuerta not para que así este pueda invertir la salida en cada flanco de reloj.

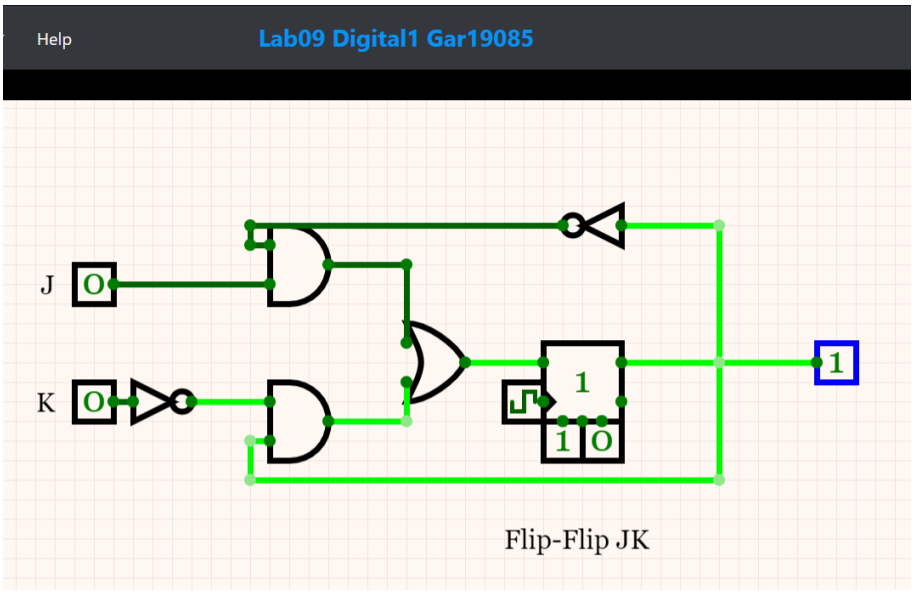


Ejercicio #03:

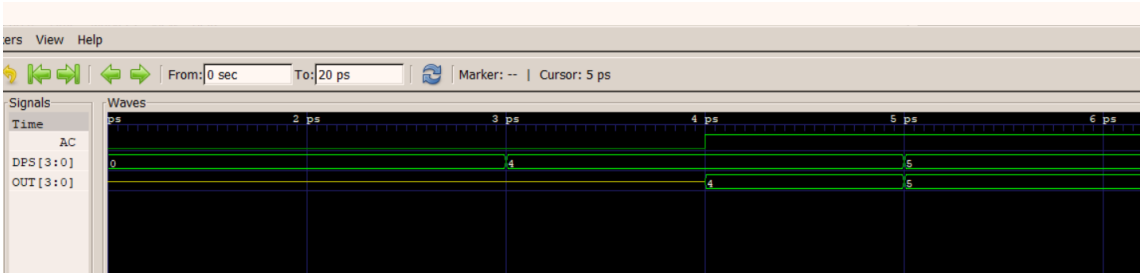



```
C:\Windows\System32\cmd.exe - apio  sim
CLK | RST | ENABLE | J  | K   | Q
VCD info: dumpfile FFJK_tb.vcd opened for output.
0   | 0   |    0   | 0 |  0  | x
1   | 1   |    0   | 0 |  0  | 0
0   | 0   |    0   | 0 |  0  | 0
1   | 0   |    0   | 0 |  0  | 0
0   | 0   |    1   | 0 |  0  | 0
1   | 0   |    1   | 0 |  0  | 0
0   | 0   |    1   | 1 |  0  | 1
1   | 0   |    1   | 1 |  0  | 1
0   | 0   |    1   | 1 |  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   | 0 |  1  | 0
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   |    1   | 0 |  1  | 0
0   | 0   |    1   | 0 |  0  | 0
1   | 0   |    1   | 0 |  0  | 0
0   | 0   |    1   | 1 |  0  | 1
1   | 0   |    1   | 1 |  0  | 1
0   | 0   |    1   | 1 |  0  | 1
1   | 0   |    1   | 1 |  0  | 1
gtkwave FFJK_tb.vcd FFJK_tb.gtkw
```

Se creo un modulo para un flip flop de 1 bit, para así implementar el Flip-Flop JK el cuál funciona por medio de una nube combinacional conformada por dos nots, dos ands y 1 or. Ya que este depende del estado futuro y presente de la salida, de J y de K.



Ejercicio #04:

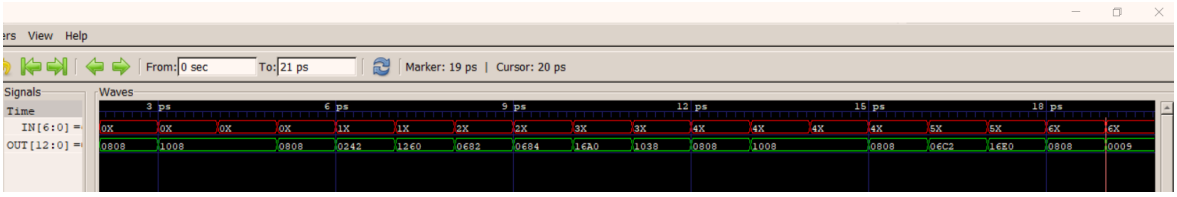



```
iverilog -o Buffer_tb.out -D VCD_OUTPUT=Buffer_tb C:/Users/ro
vvp Buffer_tb.out


Buffer
A | AC | OUT
VCD info: dumpfile Buffer_tb.vcd opened for output.
0000 |  0  |  zzzz
0100 |  0  |  zzzz
0100 |  1  |  0100
0101 |  1  |  0101
gtkwave Buffer_tb.vcd Buffer_tb.gtkw
```

Para implementar el Buffer Tri-Estado se utilizo el comando Case, para que el input que habilita o deja pasar la entrada al buffer tenga indicadas sus instrucciones para dejar pasar la información del input o dejarla como alta impedancia.

Ejercicio #05:




```
Buffer
IN |  OUT
VCD info: dumpfile ROM_tb.vcd opened for output.
0000000  |    1000000001000
xxxxxx0  |    1000000001000
00001x1  |    0100000001000
00000x1  |    1000000001000
00011x1  |    1000000001000
00010x1  |    0100000001000
0010xx1  |    0001001000010
0011xx1  |    1001001100000
0100xx1  |    0011010000010
0101xx1  |    0011010000100
0110xx1  |    1011010100000
0111xx1  |    1000000111000
1000x11  |    0100000001000
1000x01  |    1000000001000
1001x11  |    1000000001000
1001x01  |    0100000001000
1010xx1  |    0011011000010
1011xx1  |    1011011100000
1100xx1  |    0100000001000
1101xx1  |    0000000001001
1110xx1  |    0011100000010
1111xx1  |    1011100100000
gtkwave ROM_tb.vcd ROM_tb.gtkw
```

Para realizar el Rom se indico el tamaño del input y el tamaño del output para poder realizar las operaciones indicadas por la tabla, por medio del uso de Case en el modulo.