**Laboratorio #9**

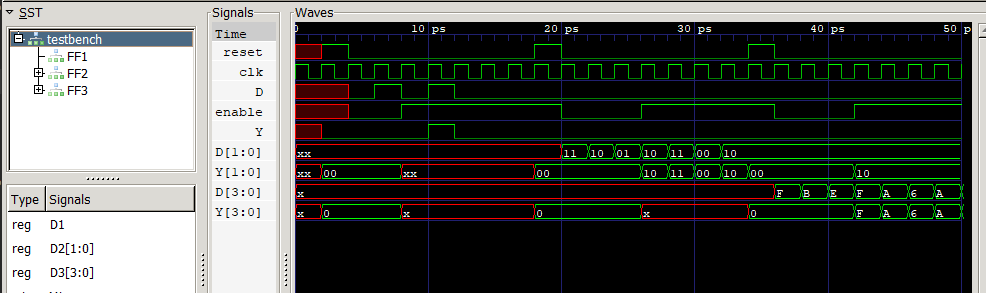
**Enlace al repositorio de GitHub:** [**https://github.com/gar19421/Lab-08-Digital.git**](https://github.com/gar19421/Lab-08-Digital.git)

**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.

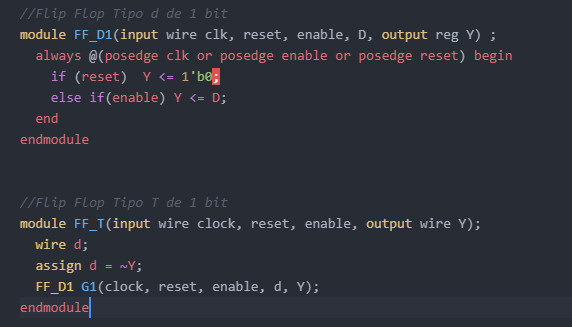
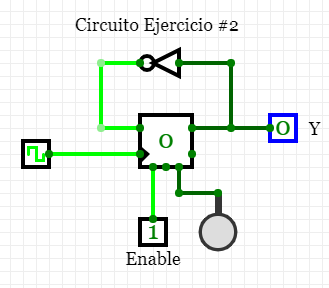


Como se observa en el diagrama de timing, cuando la señal de enable está en 1, la señal D de cada flipflop pasa a la señal de salida en el flanco de reloj.

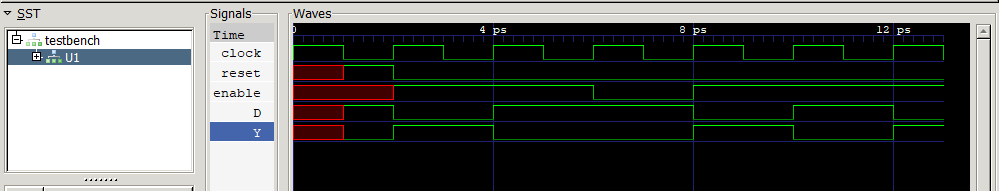


**Ejercicio 2:**

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.



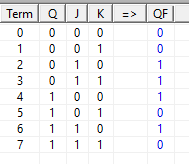
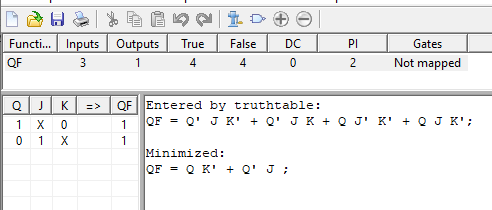
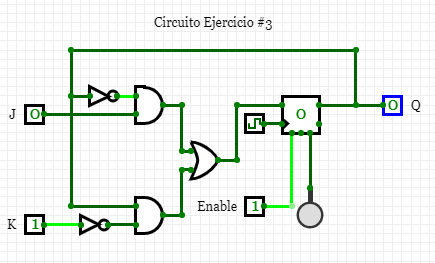
Como se puede observar en el diagrama de timing, el funcionamiento es el correcto ya que la señal de salida Y es el negado de la señal D, en cada flanco de reloj y cuando el enable se vuelve cero mantiene su valor anterior.

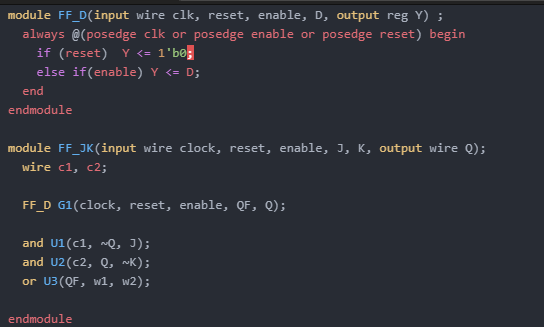


**Ejercicio 3:**

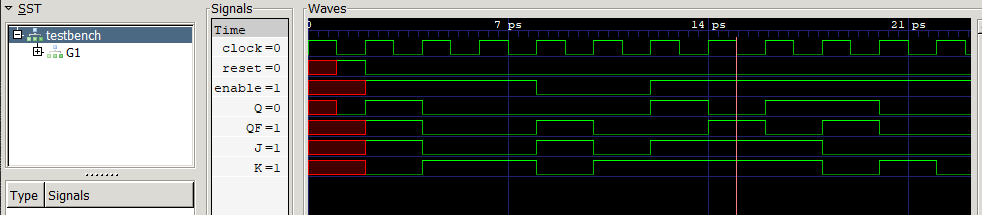
En este ejercicio, se implementó un flipflop JK con flipflop tipo D del ejercicio 1 tanto en verilog como en circuitverse, con sus entradas clock, J y K, y su salida Q la cual Si J y K son 0 entonces la salida Q mantiene su valor anterior. Si J = 1 y K = 0 entonces Q = 1. Si J = 0 y K = 1 entonces Q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

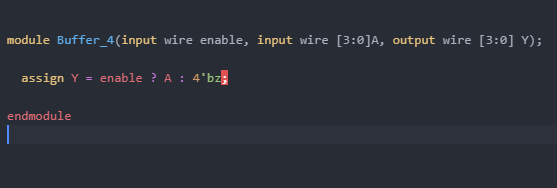


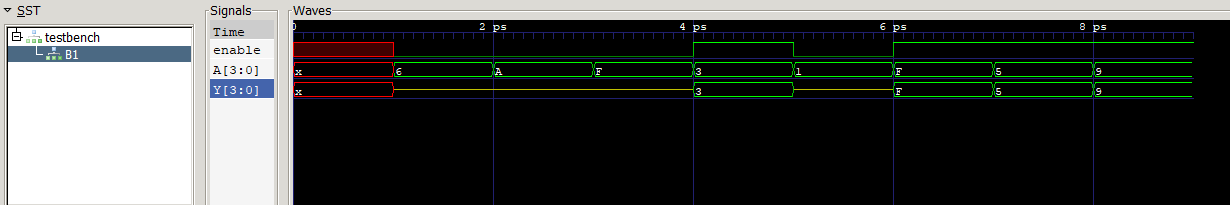
Como se observa en el diagrama de timing, las combinaciones descritas anteriormente se cumplen correctamente en cada flanco de reloj y cuando enable está en cero, la salida mantiene su valor anterior.



**Ejercicio 4:**

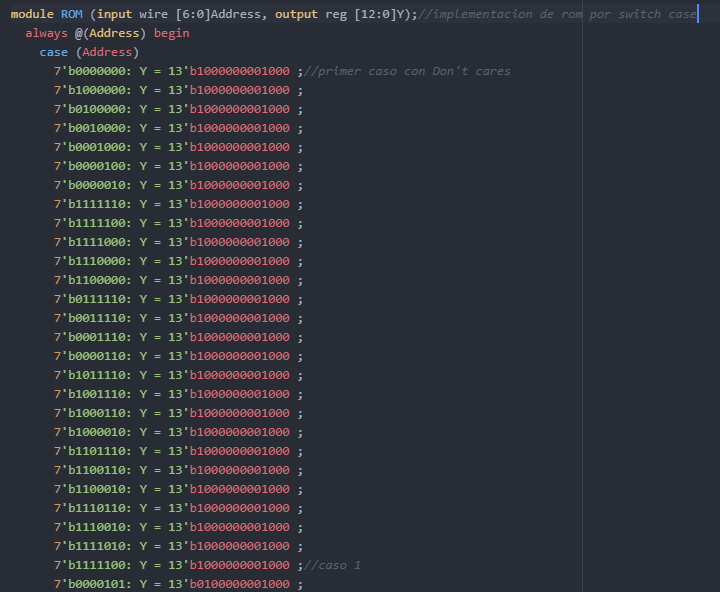
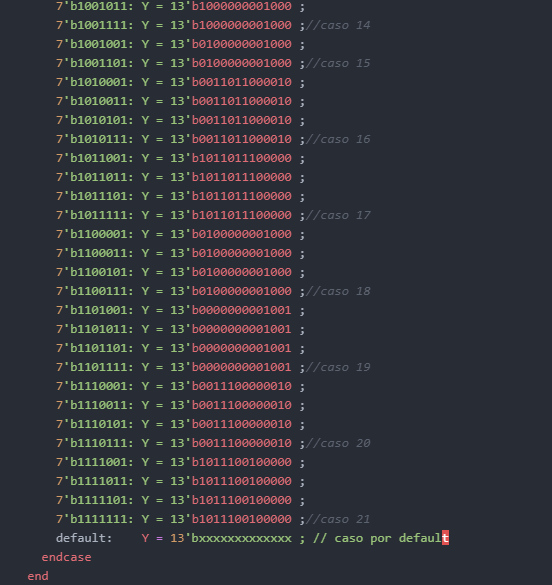
En este ejercicio se implementó un buffer triestado de 4 bits en verilog. El funcionamiento de este es que cuando la señal de entrada del enable es 1, deja pasar la señal A hacia la salida Y, sin embargo, cuando el enable es 0, la salida se encuentra en alta impedancia tal y como se observa tanto en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

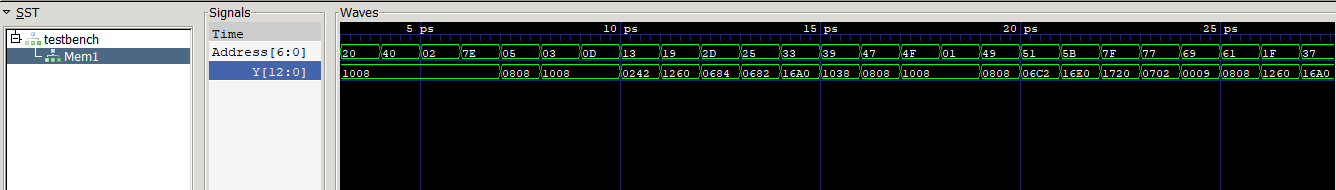


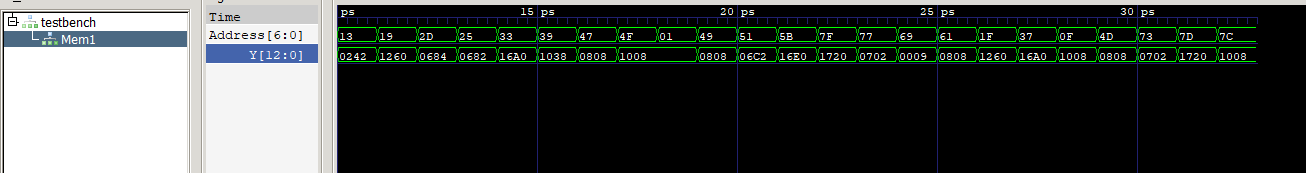


**Ejercicio 5:**

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.

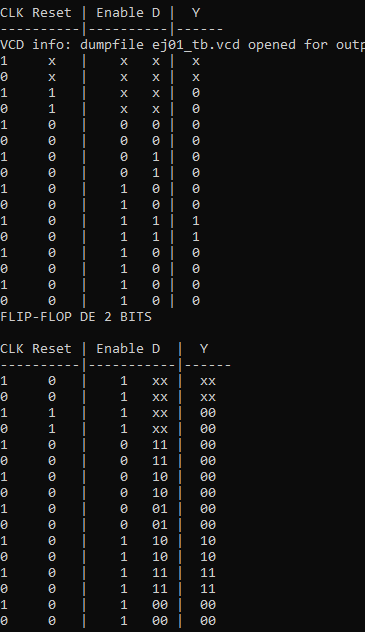
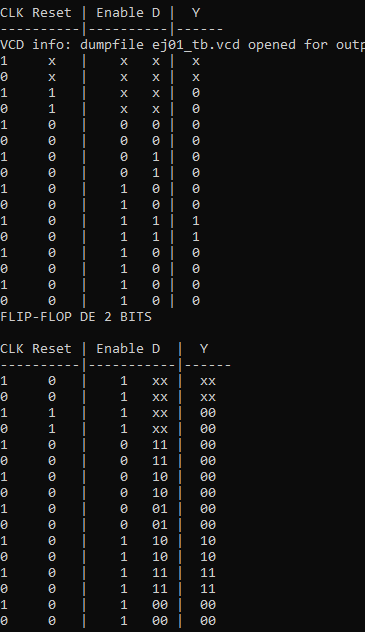
 

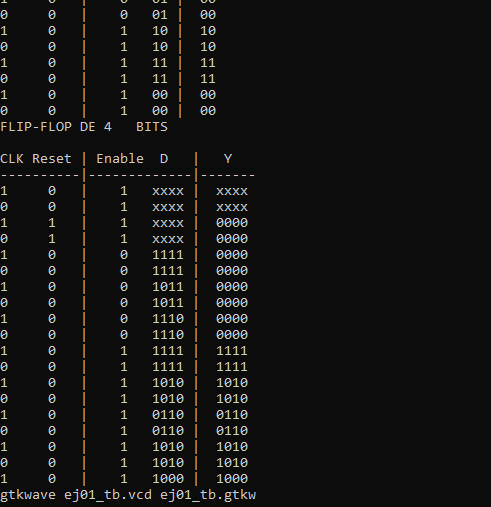
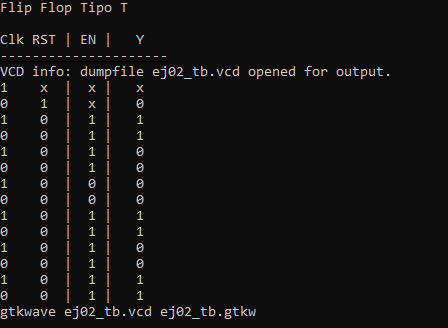


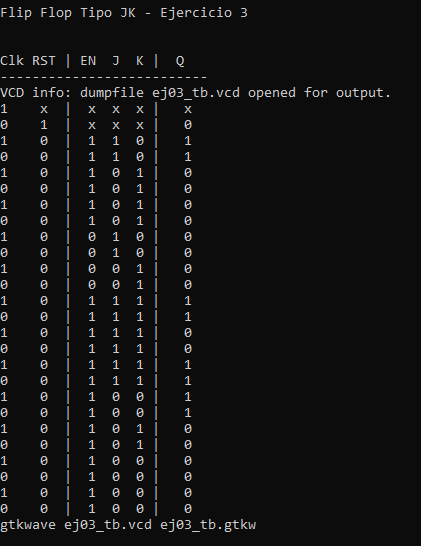
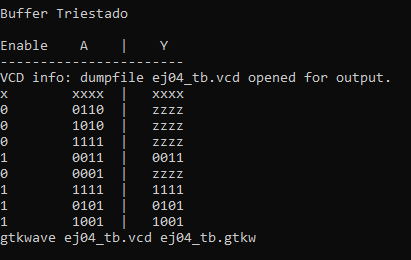


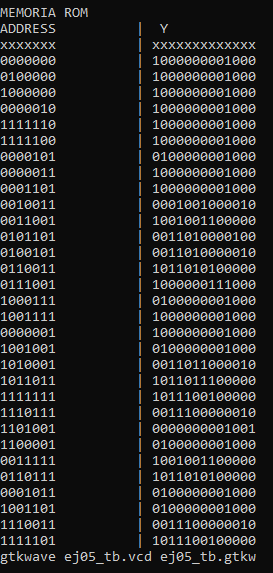
**Otras evidencias (Screenshots):**

- Pruebas en consola:



- Testbech:

