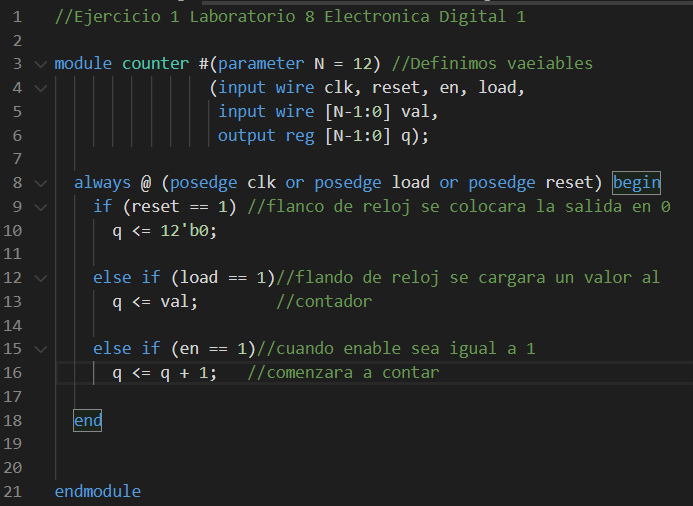
Fernando Arribas

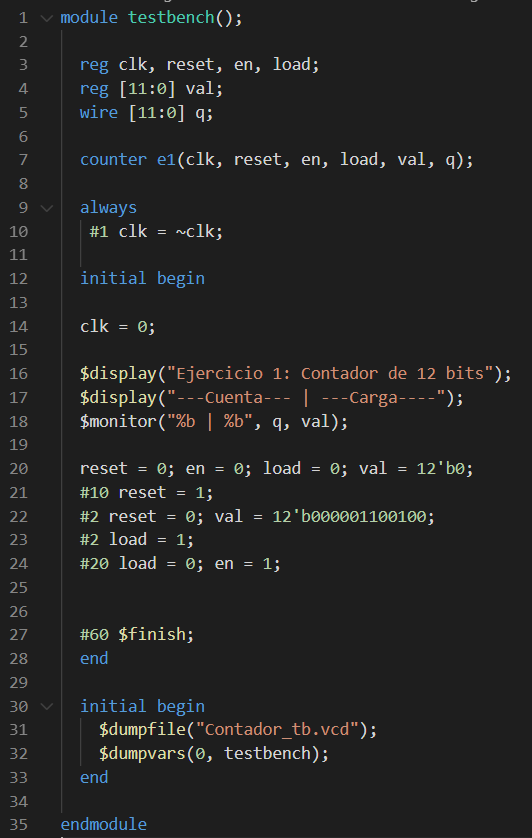
Carné: 19030

Lab 08 Digital 1

Ejercicio 1



Primero definimos las variables a utilizar las cuales son un clock, un reset, un enable (en), un load, un variable de 12 bits llamada val, y la salida de 12 bits; adicionalmente definimos un parámetro, el cual nos indicará de cuantos bits será nuestro contador, en este caso 12 bits. Luego todo eso se ingresa a un always, en donde el reloj, el reset y load trabajaran con flancos de reloj positivos. Dentro existen un total de 3 condicionales, el primero indica si hay un cambio de reloj para reset la salida se colocara en 0. Luego si existe un flanco de reloj en load la salida pasara a ser los que tengamos en la variable val. Por último, si enable es igual a 1 la salida del contador será el valor pasado del mismo más uno. Hay que mencionar que todas estas operaciones están realizadas con non-blocking assignment, para que el contador pueda saltar entre las distintas opciones sin ninguna restricción.



Testbench contador

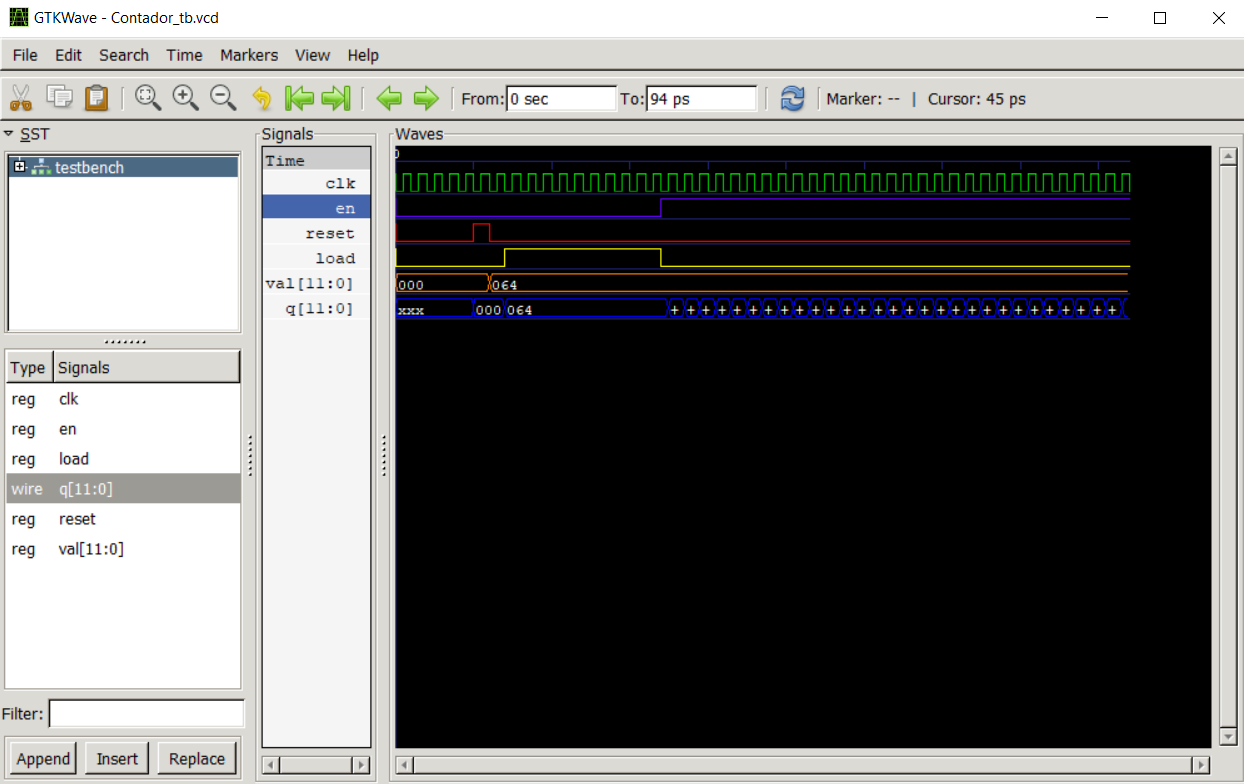
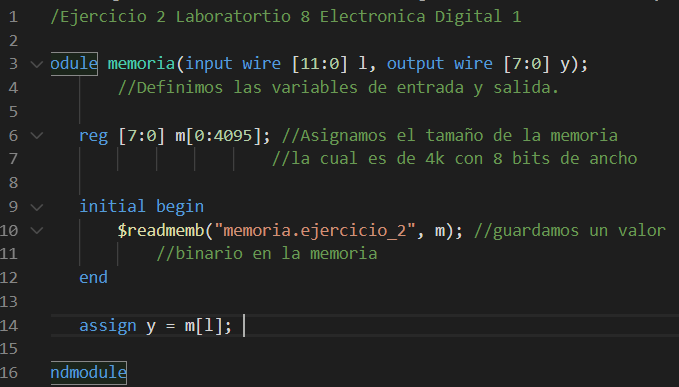
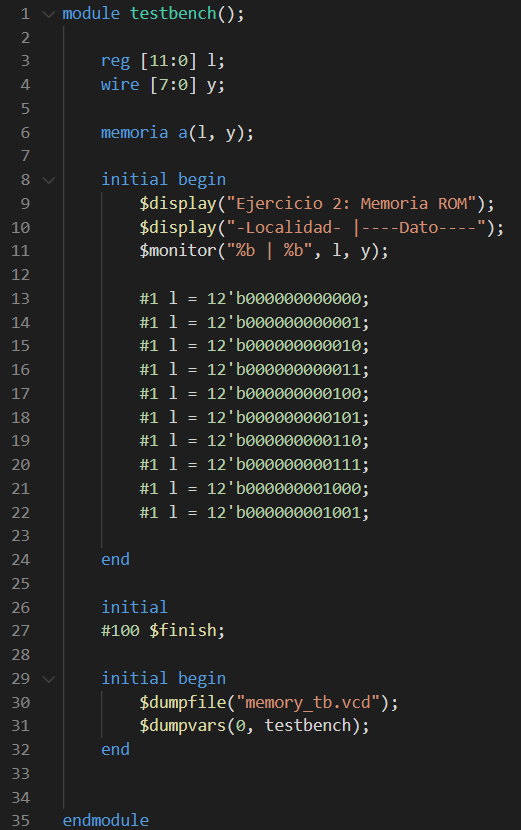


Diagrama de Timing Contador

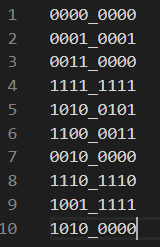
Ejercicio 2



Primero declaramos las variables de entrada, siendo una entrada L de 12 bits, la cual nos servirá para buscar en las 4096 localidades de nuestra memoria y una salida Y que nos desplegara lo que esta guardado en cierta localidad de nuestra memoria. Lo primero a realizar fue un array de memoria, el cual será de 4k localidades y 8 bits de ancho, la cual será una variable M. Luego por medio de la instrucción $readmemb leeremos el archivo externo que guardaremos en nuestra memoria, la cual será guardada en el arreglo m. Por ultimo asignamos que el valor de la salida será la localidad que seleccionemos con L.



Testbench Memoria



Datos Guardados en la Memoria

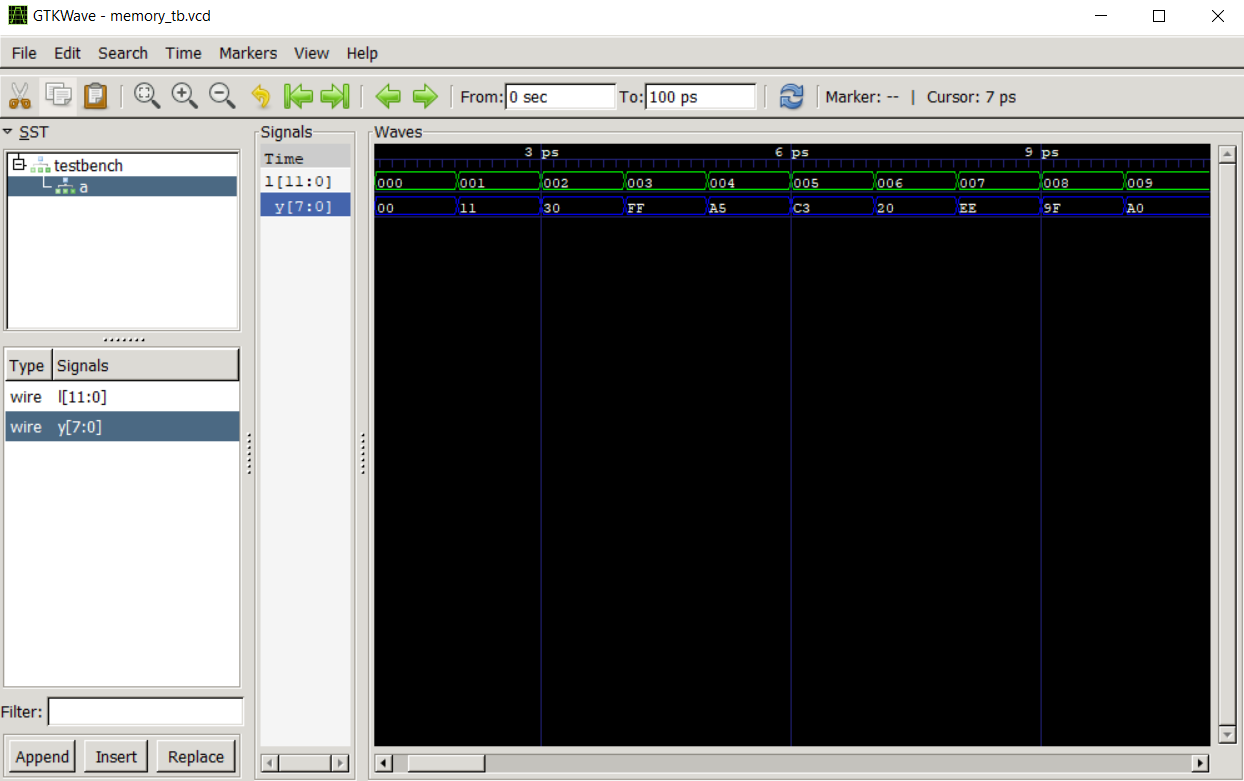
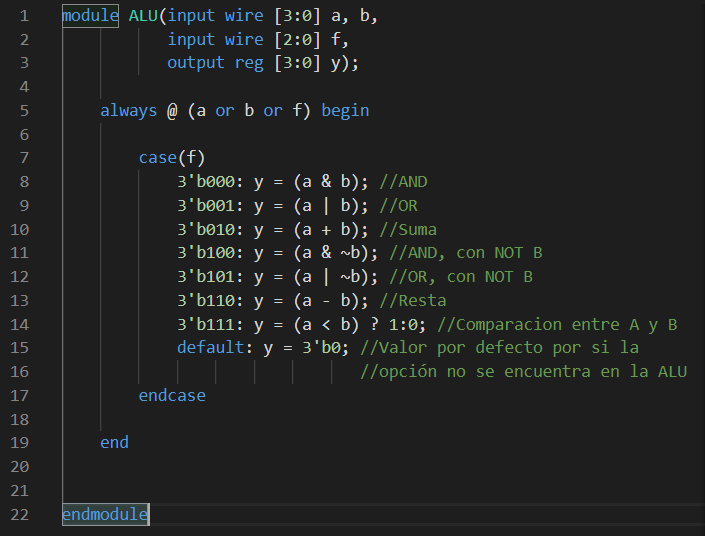


Diagrama de Timing Memoria

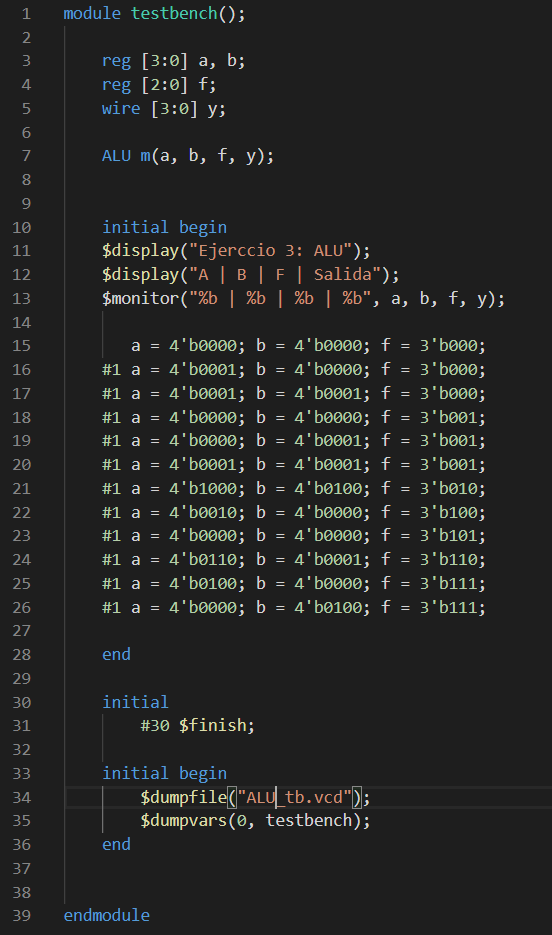
Array: se declara un reg, el cual poseerá cierta cantidad de bits, luego dentro del reg se colocará una variable a la cual en su derecha se le colocará otra cierta cantidad de variables. la cual será un array de 4096X8, o 4096 localidades con un ancho de ocho bits.

$readmemb se utiliza para leer entradas en binario, mientras que $readmemh se utiliza para leer entradas en hexadecimal.

Ejercicio 3



Lo primero es definir las variables a utilizar, las cuales son dos entradas de 4 bits llamadas A y B, el controlador de 3 bits F y la salida de 4 bits Y, Para describir el funcionamiento de la ALU se utilizo casos basados en F, en donde cada combinación de F dará paso a una acción, exceptuando la combinación 011 la cual no realizará nada. La primera acción (000) será un AND, luego (001) un OR, luego (010) una Suma, la combinación 011 no hace nada, la 100 es un AND con B negada, 101 es un OR con B negada, 110 es una resta y 111 es una comparación entre A y B para ver cuál es mayor. Por último, se colocará una salida ‘por defecto’ la cual dará paso si la instrucción ingresada no existe.



Testbench ALU

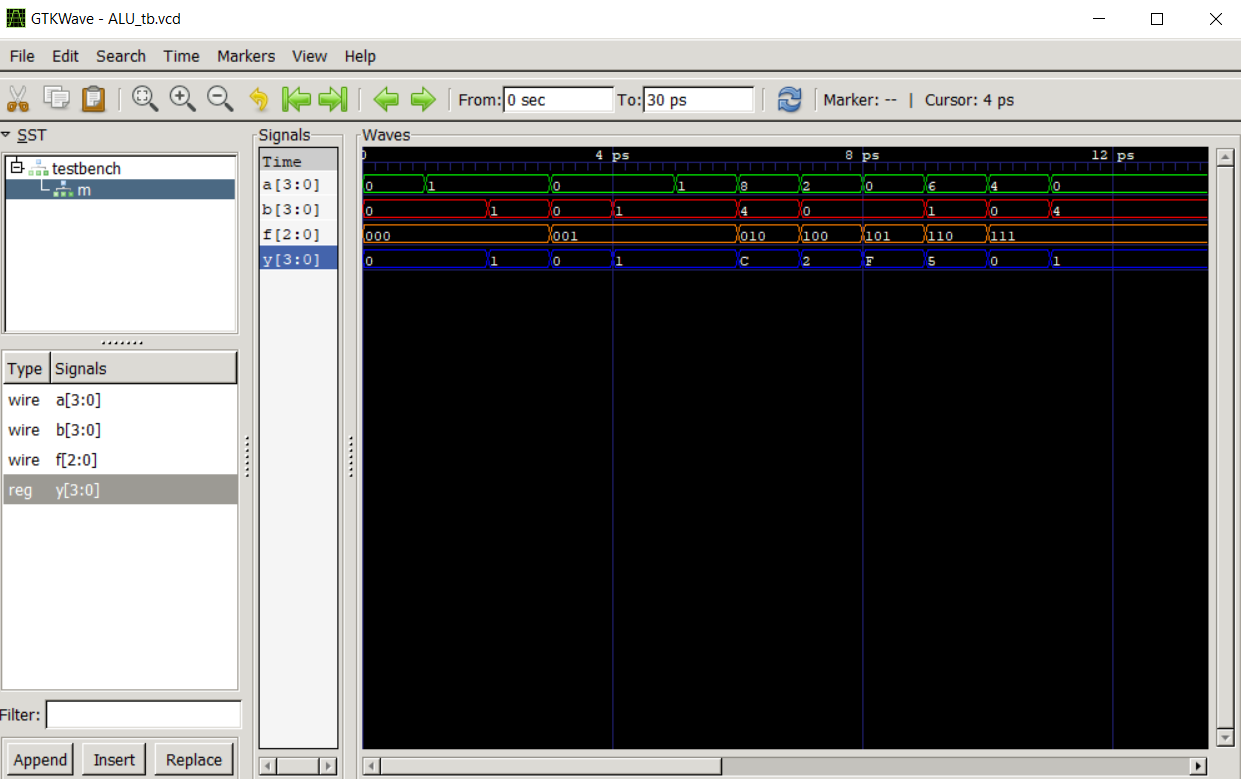


Diagrama de Timing ALU

Link Repositorio de Github: https://github.com/fernando19030/Laboratorios-Electronica\_Digital\_1-19030.git