Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías

División de Electrónica y Computación
Departamento de Ciencias Computacionales
Licenciatura en Ingeniería en Computación
Arquitectura de computadoras
Clave: CC210 Sección: D02
19:00 – 20:55 Martes Jueves
Reporte Actividad 8.
Berrospe Barajas Héctor Eduardo
21-Marzo-2016
López Arce Delgado Jorge Ernesto

Introducción

ROM (Read-only Memory), este circuito se utiliza en computadoras y otros dispositivos electrónicos, es un medio de almacenamiento, la información que se guarda en una ROM no pueden ser modificados por el usuario común, este tipo de memoria la mayor parte del tiempo es usada para almacenar el firmware (software vinculado a hardware)

Objetivos

El alumno conocerá la estructura de una memoria estática, de solo lectura.

ROM (Read-only Memory):

Una memoria de solo lectura, es una circuito combinacional y no tiene un estado interno. Su salida depende solo de la entrada (dirección de lectura). A pesar de que es un circuito combinacional, en esta práctica se realizara esta memoria de manera síncrona.

Desarrollo

Se describe como ROM a un circuito que tiene varios bloques de memoria para almacenar información, cada uno de los bloques de memoria se asocia con líneas principales de bits y líneas de sub-bits, se conectan con bloques de transistores, en esta práctica solo realizaremos la ROM de forma comportamental como se hizo en prácticas anteriores.

1. Diseñar una memoria tipo ROM capaz de almacenar 48 datos almacenados de un ancho de palabra de 8 bits cada uno (48x8)
    a. Debe tener una señal de entrada de reloj, así como un bus para las direcciones de lectura (4 bits). Y una salida en un bus de 8 bits.
    b. Los datos almacenados se asignan de la siguiente manera, 19 espacios de memoria para su nombre(s), 19 para sus apellidos, 9 para su código de estudiante y la dirección 47 para almacenar el carácter de espacio en ASCII, representado por el número '32' en decimal. El acomodo de los datos dependerá de lo asignado en clase.
    c. Generar un archivo de testbench donde tendrán que realizar la lectura a la memoria de tal forma para que en la simulación se observen los datos de la siguiente manera, Nombre1, espacio, nombre2, espacio, código, espacio, apellido1, apellido2.
    Ingrese la cantidad de caracteres que quepan de su nombre, el carácter de espacio solo debe estar guardado una vez y será en la dirección 47.

El primer paso para diseñar las prácticas hechas y por hacer es identificar los puertos de entrada y salidas y declararlos.
Una vez declarados nos preocupamos por el funcionamiento de nuestro circuito y entendemos el porqué de cada una de nuestras variables, que en este caso son las entradas y salidas, en el diseño nos piden una señal de reloj **clk** el cual tiene como función habilitar la salida de la posición en memoria que se encuentra en cada flanco de subida.
El **addr** es la entrada que indica la dirección de memoria a leer.
La salida **d** (data) es el que muestra lo que contiene la posición de memoria.

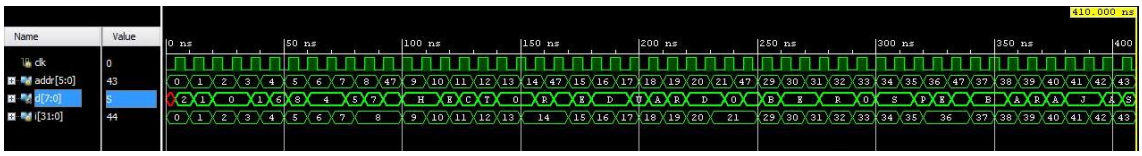
Una vez sabiendo esto el siguiente paso fue declarar el bloque always, el que va contener todo

nuestro programa, para simular la memoria ROM como indica el diseño necesitamos que cada que haya un flanco de subida describa cierta actividad, es decir, pasar a la salida el contenido de la dirección de memoria seleccionada, por lo tanto nuestro bloque always debe de tener un posedge clk, que significa que realizara un algoritmo cada que el programa identifique un flanco de subida del clk.
Entrando al bloque always se anexo el algoritmo que define el funcionamiento de nuestra memoria ROM, en este caso se utilizó un switch case con referencia a la dirección, de esa manera cada que haya un flanco de subida, accionara la estructura de control el cual lee el valor de dirección de memoria y dependiendo de la dirección de memoria le asigna a la salida el dato correspondiente a esa dirección.

```
always@(posedge clk)
  begin
    case(addr)
      //Codigo de alumno
      6'b000000 : d <=  8'b00110010; //[2] Codigo ASCII
      6'b000001 : d <=  8'b00110001; //[1] Codigo ASCII
```

Testbench

Para el testbench se utilizó un ciclo for el cual leyera secuencialmente la memoria con condiciones las cuales indiquen un espacio cada que acabe cierta dirección de memoria, en mi caso fue en los caso 8, 14, 21 y 36, estas direcciones de memoria es cada que acaba mi código , mi primer nombre, segundo nombre y mi primer apellido.
También existe otra condición, pues la memoria es de 47 espacios y los primeros 9 son para el código, los siguientes 19 son para mis nombres, pero mi nombre no usa los 19 espacios disponibles, por lo tanto cuando la dirección de memoria es 29 indica un brinco en el modo de lectura hasta la dirección 28 que es en donde empieza mi apellido y después de eso siga leyendo secuencialmente.



Nota: La variable i es una variable auxiliar que se utilizó para leer secuencialmente la memoria ROM

Conclusiones

En esta práctica se entendió el funcionamiento de la memoria ROM, es decir, solo guarda cierta información, la manera en que se comporta es bastante sencillo, pues solo tiene una dirección de memoria y en la salida muestra los datos que se encuentran en esa dirección, para el testbech se utilizó un for para evitar la fatiga de escribir todos los espacios en memoria uno por uno.