Imagen que contiene Icono

Descripción generada automáticamenteLogotipo

Descripción generada automáticamente con confianza media

Instituto Politécnico Nacional

Escuela Superior de Cómputo

Arquitectura de Computadoras

**“Implementación de Memoria ROM en Verilog”**

Alumno:

Malagón Baeza Alan Adrian

Profesor:

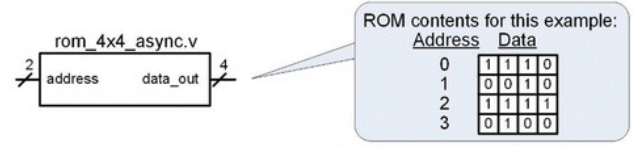
Alemán Arce Miguel Ángel

Grupo: 5CV1

**Introducción**

Una memoria ROM en Verilog se puede definir de dos maneras. La primera es simplemente usar una sentencia case para definir el contenido de cada ubicación en la memoria en función de la dirección entrante. Un segundo enfoque es declarar una matriz y luego inicializar su contenido. Cuando se usa una matriz, un bloque de procedimiento separado maneja la asignación del contenido de la matriz a la salida en función de la dirección entrante. La matriz puede ser inicializada utilizando un bloque inicial o a través de las tareas del sistema de E/S de archivos $readmemb() o $readmemh() (Opción 3). El siguiente ejemplo muestra dos enfoques para modelar una Memoria ROM 4×4. En este ejemplo, la memoria es asíncrona, lo que significa que como tan pronto como cambie la dirección, los datos de la ROM aparecerán inmediatamente. Para modelar este comportamiento asíncrono, los bloques de procedimiento son sensibles a la entrada entrante.

**Ejemplo: Modelos con descripción de comportamiento de una memoria ROM asíncrona 4x4 en Verilog**



**Opción** 1

Un enfoque simple para una rom es implementarla como una sentencia case

**Tabla

Descripción generada automáticamente**

**Opción 2**

Un enfoque diferente es declarar una matriz y usar un bloque "inicial" para definir su contenido. Luego se usa un bloque always para asignar el vector direccionado a data\_out.

Texto

Descripción generada automáticamente

Se declara una matriz MxN

Una ROM síncrona se puede crear de manera similar al enfoque asíncrono. La única diferencia es que, en una ROM síncrona, se usa un borde de reloj para activar el bloque de procedimiento que actualiza data\_out. Se utiliza una lista de sensibilidad que contiene el reloj para activar la asignación. El siguiente ejemplo muestra dos modelos de Verilog para una ROM síncrona.

**Ejemplo: Modelos con descripción de comportamiento de una memoria ROM síncrona 4x4 en Verilog**

Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza baja

**Opción** 1

**Texto, Tabla

Descripción generada automáticamente**

**Opción 2**

Texto

Descripción generada automáticamente

El comportamiento síncrono de estos modelos de ROM se logra haciendo que el bloque de procedimiento que actualiza data\_out sea sensible al flanco ascendente del reloj.

**Desarrollo**

**Opción 1**

**Código Verilog**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Código Verilog Testbench**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Implementación RTL en Vivado 2022.2**

Diagrama

Descripción generada automáticamente

Podemos observar cómo utilizo FlipFlops y un multiplexor para hacer la implementación con un Case.

**Resultado de la simulación:**

En la simulación, se proporciona cada dirección posible (es decir, "00", "01", "10”, y “11”) para verificar que la ROM se inicializó correctamente. Antes del primer borde del reloj, el simulador no sabe qué asignar a data\_out, por lo que enumera el valor como desconocido (X).

Interfaz de usuario gráfica, Diagrama, Escala de tiempo

Descripción generada automáticamente

*Hexadecimal*

Diagrama, Escala de tiempo

Descripción generada automáticamente

*Binario*

Los datos no aparecen en la salida hasta un flanco ascendente del reloj.

Antes del primer borde del reloj, el valor de data\_out es desconocido (X).

Como se observa en la simulación, los resultados son los almacenados en la memoria de acuerdo con la localidad.

**Opción 2**

**Código Verilog**

Texto, Correo electrónico

Descripción generada automáticamente

**Código Verilog Testbench**

Texto, Correo electrónico

Descripción generada automáticamente

**Descripción RTL en Vivado 2022.2**

Diagrama

Descripción generada automáticamente

Como se puede observar se utilizó un bloque de memoria ROM del FPGA seleccionado, y para el control del dato a la salida, se implementó mediante flipflops con su terminal de clk.

**Resultado de la simulación:**

**Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente**

*Hexadecimal*

Interfaz de usuario gráfica, Escala de tiempo

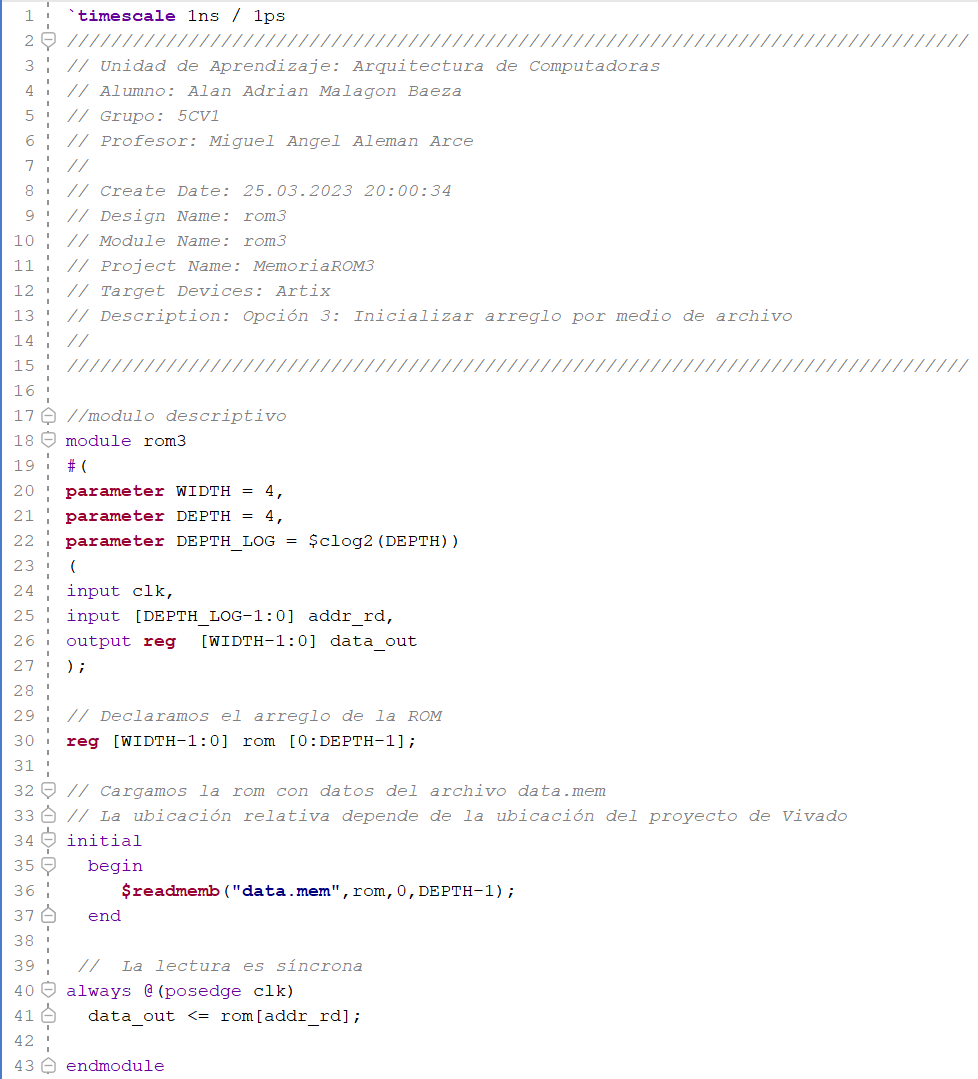
Descripción generada automáticamente

*Binario*

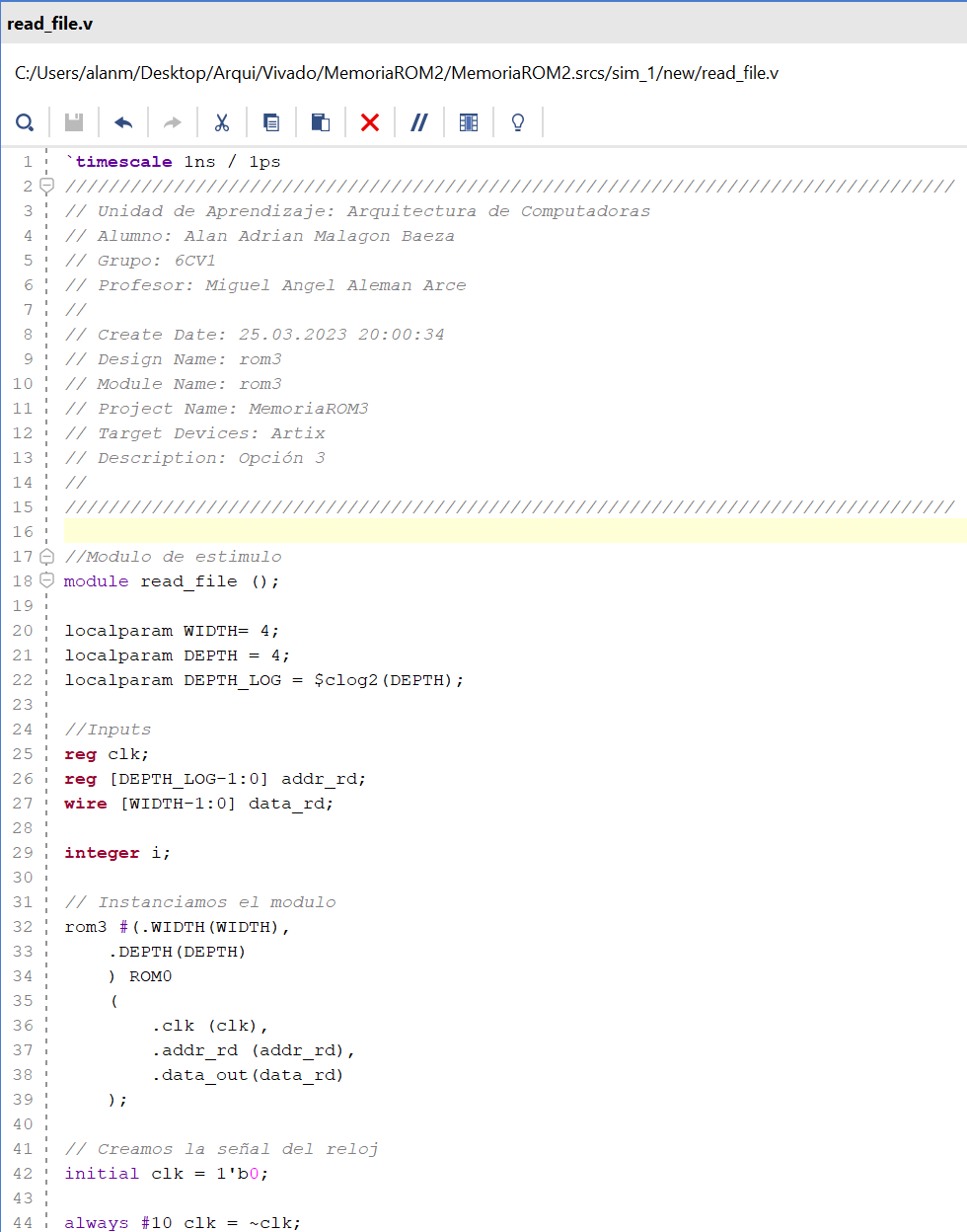
Como se observa en la simulación, los resultados son los almacenados en la memoria de acuerdo con la localidad.

**Opción 3**

**Código Verilog**

****

**Código Verilog Testbench**

****

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Archivo de memoria**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media**

**Descripción RTL en Vivado 2022.2**

Diagrama

Descripción generada automáticamente con confianza baja

Como se puede observar se utilizó un bloque de memoria ROM del FPGA seleccionado, y para el control del dato a la salida, se implementó mediante flipflops con su terminal de clk.

**Resultado de la consola:**

Tiempo, dirección de la rom (address), contenido de la rom (data\_rd)

**Texto, Carta

Descripción generada automáticamente**

**Resultado de la simulación:**

**Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente**

Como se observa en la simulación, los resultados son los almacenados en la memoria dados por el archivo data.mem de acuerdo con la localidad.

**Conclusión**

Las ventajas de modelar la memoria en Verilog son que permite abstraer los detalles de la celda de almacenamiento del funcionamiento funcional del sistema de memoria, es demasiado difícil modelar el comportamiento analógico de la celda de almacenamiento y hay demasiadas celdas para modelar, por lo que la simulación tardaría demasiado.

El término memoria se refiere a grandes conjuntos de almacenamiento digital. La tecnología utilizada en la memoria normalmente se optimiza para la densidad de almacenamiento a expensas de la capacidad de control. Esto es diferente de un D-flip-flop, que está optimizado para un control completo a nivel de bit.

Un dispositivo de memoria siempre contiene una entrada de bus de direcciones. El número de bits en el bus de direcciones determina a cuántas ubicaciones de almacenamiento se puede acceder. Un bus de direcciones de n-bits puede acceder a 2n (o M) ubicaciones de almacenamiento.

El ancho de cada ubicación de almacenamiento (N) permite aumentar la densidad de la matriz de memoria al leer y escribir vectores de datos en lugar de bits individuales.

Un mapa de memoria es una representación gráfica de una matriz de memoria. Un mapa de memoria es útil para dar una visión general de la capacidad de la matriz y cómo se utilizan los diferentes rangos de direcciones de la matriz.

Una lectura es una operación en la que se recuperan datos de la memoria.

Una matriz de memoria asíncrona responde inmediatamente a sus entradas de control. Una matriz de memoria síncrona solo responde en el borde de activación del reloj.

La memoria ROM es un tipo de memoria en la que no se puede escribir durante el funcionamiento normal.

La arquitectura básica de una ROM consta de líneas de bits (verticales) y líneas de palabras (horizontales) que se cruzan y contienen celdas de almacenamiento en sus puntos de cruce. Los datos se leen de la matriz ROM utilizando las líneas de bits.

La memoria se puede modelar en Verilog utilizando un tipo de datos de matriz que consta de elementos de tipo reg.

**Referencia**

1. Brock J. LaMeres, Introduction to Logic Circuits & Logic Design with Verilog, Springer, 1st Edition, USA, 2017.