Chapter 10: Modeling Memory

This chapter covers how to model memory arrays in Verilog. These models are technology independent, meaning that they can be ultimately synthesized into a wide range of semiconductor memory devices.

Learning Outcomes—After completing this chapter, you will be able to:

10.1 Describe the basic architecture and terminology for semiconductor-based memory systems.
10.2 Design a Verilog model for a read-only memory array.
10.3 Design a Verilog model for a read/write memory array.

10.1 Memory Architecture and Terminology

The term *memory* is used to describe a system with the ability to store digital information. The term *semiconductor memory* refers to systems that are implemented using integrated circuit technology. These types of systems store the digital information using transistors, fuses, and/or capacitors on a single semiconductor substrate. Memory can also be implemented using technology other than semiconductors. Disk drives store information by altering the polarity of magnetic fields on a circular substrate. The two magnetic polarities (north and south) are used to represent different logic values (i.e., 0 or 1). Optical disks use lasers to burn pits into reflective substrates. The binary information is represented by light either being reflected (no pit) or not reflected (pit present). Semiconductor memory does not have any moving parts, so it is called *solid state memory* and can hold more information per unit area than disk memory. Regardless of the technology used to store the binary data, all memory has common attributes and terminology that are discussed in this chapter.

10.1.1 Memory Map Model

The information stored in memory is called the **data**. When information is placed into memory, it is called a **write**. When information is retrieved from memory, it is called a **read**. In order to access data in memory, an **address** is used. While data can be accessed as individual bits, in order to reduce the number of address locations needed, data is typically grouped into *N-bit words*. If a memory system has $N = 8$, this means that 8-bits of data are stored at each address. The number of address locations is described using the variable M. The overall size of the memory is typically stated by saying "$M \times N$." For example, if we had a 16×8 memory system, that means that there are 16 address locations, each capable of storing a byte of data. This memory would have a **capacity** of $16 \times 8 = 128$ bits. Since the address is implemented as a binary code, the number of lines in the address bus (n) will dictate the number of address locations that the memory system will have ($M = 2^n$). Figure 10.1 shows a graphical depiction of how data resides in memory. This type of graphic is called a *memory map model*.

B. J. LaMeres, *Quick Start Guide to Verilog*, https://doi.org/10.1007/978-3-030-10552-5_10

Fig. 10.1
Memory map model

10.1.2 Volatile vs. Non-volatile Memory

Memory is classified into two categories depending on whether it can store information when power is removed or not. The term **non-volatile** is used to describe memory that *holds* information when the power is removed, while the term **volatile** is used to describe memory that loses its information when power is removed. Historically, volatile memory is able to run at faster speeds compared to non-volatile memory, so it is used as the primary storage mechanism while a digital system is running. Non-volatile memory is necessary in order to hold critical operation information for a digital system such as startup instructions, operations systems, and applications.

10.1.3 Read-Only vs. Read/Write Memory

Memory can also be classified into two categories with respect to how data is accessed. **Read-Only Memory (ROM)** is a device that cannot be written to during normal operation. This type of memory is useful for holding critical system information or programs that should not be altered while the system is running. **Read/Write** memory refers to memory that can be read and written to during normal operation and is used to hold temporary data and variables.

10.1.4 Random Access vs. Sequential Access

Random-Access Memory (RAM) describes memory in which any location in the system can be accessed at any time. The opposite of this is **sequential access** memory, in which not all address locations are immediately available. An example of a sequential access memory system is a tape drive. In order to access the desired address in this system, the tape spool must be spun until the address is in a position that can be observed. Most semiconductor memory in modern systems is random access. The terms RAM and ROM have been adopted, somewhat inaccurately, to also describe groups of memory with particular behavior. While the term ROM technically describes a system that cannot be written to, it has taken on the additional association of being the term to describe non-volatile memory. While the term RAM technically describes how data is accessed, it has taken on the additional association of being the term to describe volatile memory. When describing modern memory systems, the terms RAM and ROM are used most commonly to describe the characteristics of the memory being used; however, modern memory systems can be both read/write and non-volatile, and the majority of memory is random access.

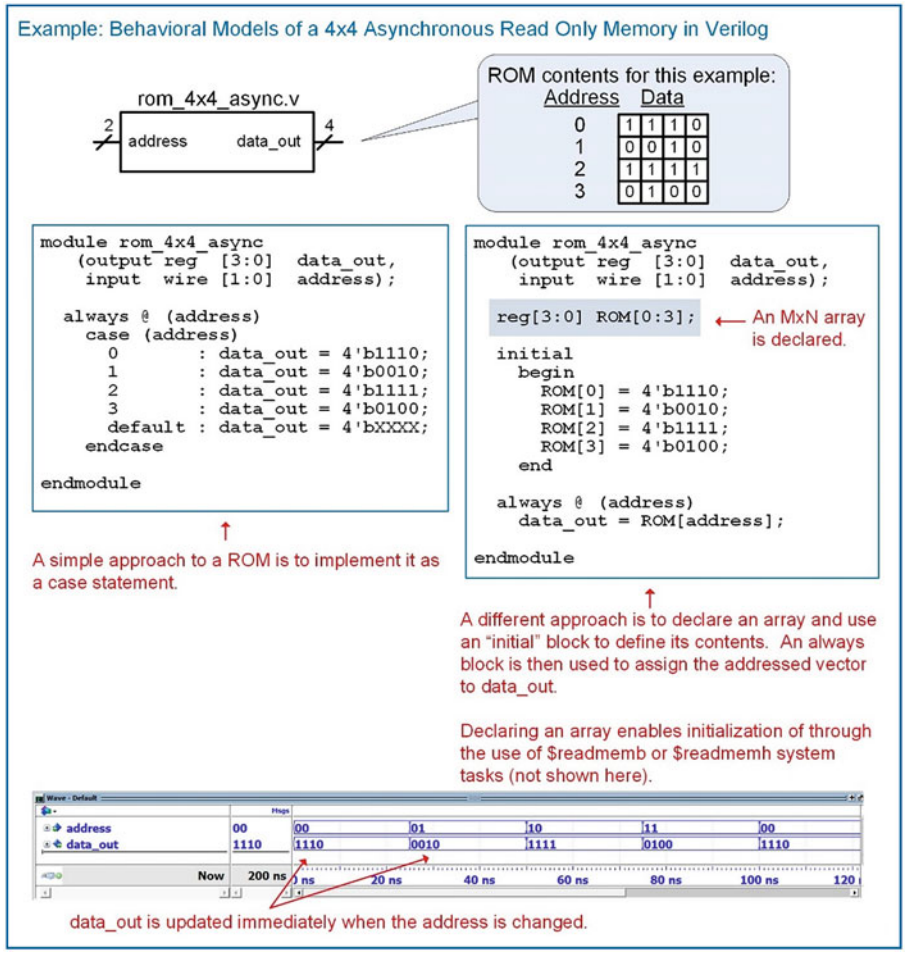
Concept Check

CC10.1 An 8-bit wide memory has eight address lines. What is its capacity in bits?

(A) 64 (B) 256 (C) 1024 (D) 2048

10.2 Modeling Read-Only Memory

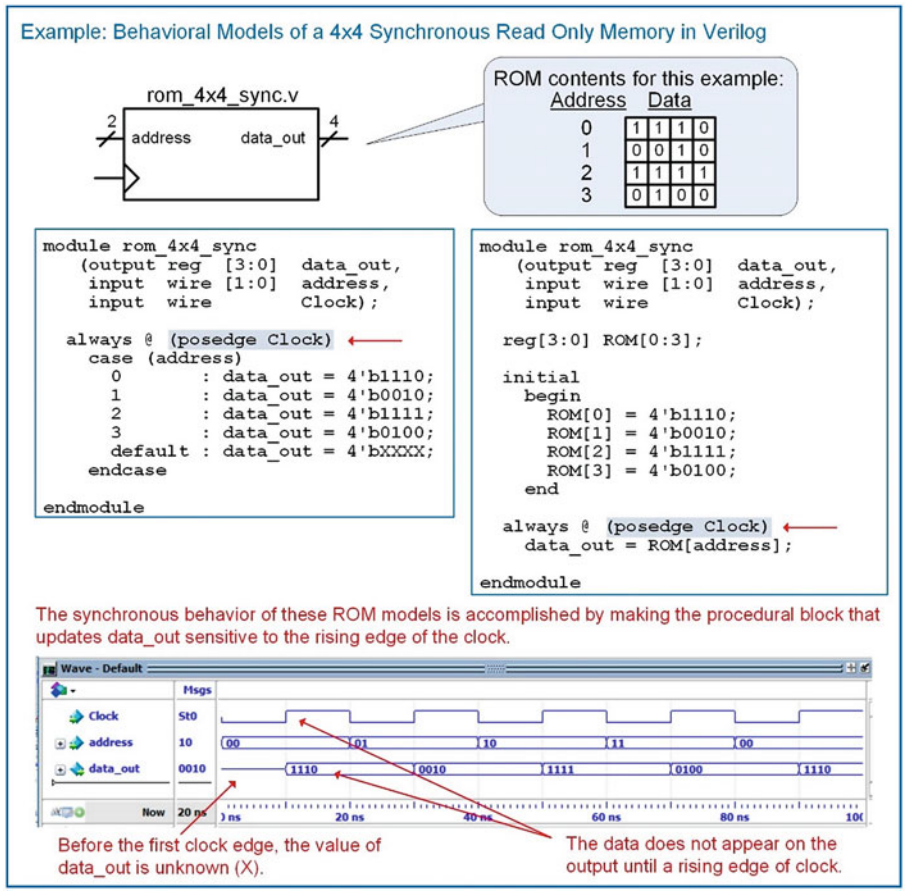
A read-only memory in Verilog can be defined in two ways. The first is to simply use a case statement to define the contents of each location in memory based on the incoming address. A second approach is to declare an array and then initialize its contents. When using an array, a separate procedural block handles assigning the contents of the array to the output based on the incoming address. The array can be initialized using either an initial block or through the file I/O system tasks $readmemb() or $readmemh(). Example 10.1 shows two approaches for modeling a 4 × 4 ROM memory. In this example the memory is asynchronous, meaning that as soon as the address changes the data from the ROM will appear immediately. To model this asynchronous behavior the procedural blocks are sensitive to the incoming address. In the simulation, each possible address is provided (i.e., "00," "01," "10," and "11") to verify that the ROM was initialized correctly.



Example 10.1
Behavioral models of a 4 × 4 asynchronous read-only memory in Verilog

A synchronous ROM can be created in a similar manner as in the asynchronous approach. The only difference is that in a synchronous ROM, a clock edge is used to trigger the procedural block that updates data_out. A sensitivity list is used that contains the clock to trigger the assignment. Example 10.2

shows two Verilog models for a synchronous ROM. Notice that prior to the first clock edge, the simulator does not know what to assign to data_out so it lists the value as unknown (X).



Example 10.2
Behavioral models of a 4 × 4 synchronous read-only memory in Verilog

Concept Check

CC10.2 Explain the advantage of modeling memory in Verilog without going into the details of the storage cell operation.

(A) It allows the details of the storage cell to be abstracted from the functional operation of the memory system.

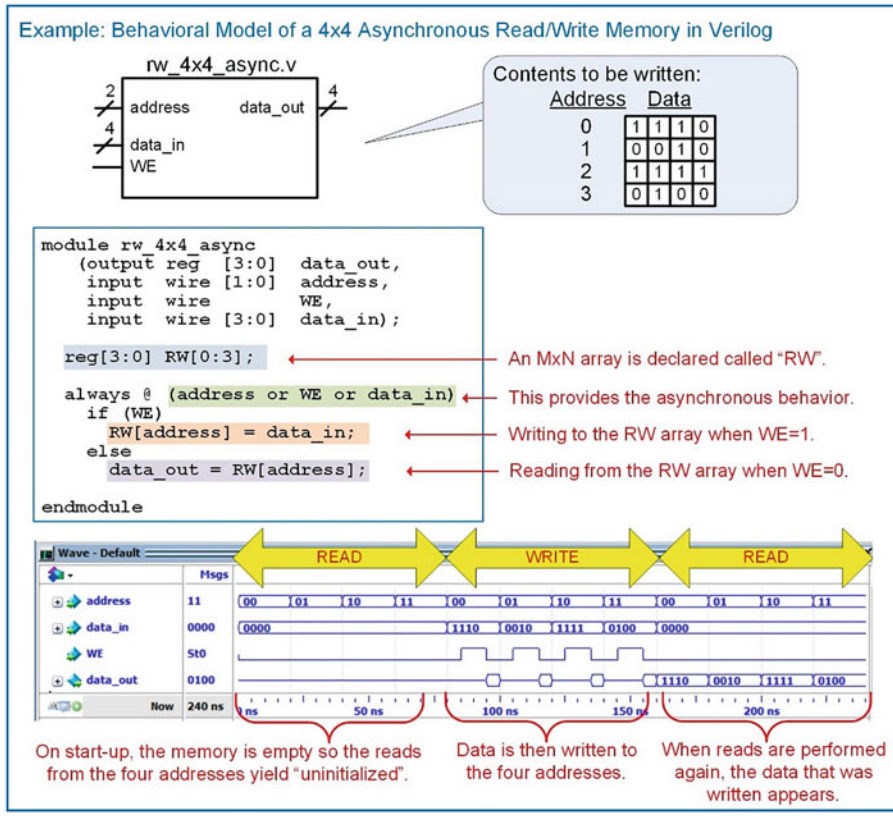
(B) It is too difficult to model the analog behavior of the storage cell.

(C) There are too many cells to model, so the simulation would take too long.

(D) It lets both ROM and R/W memory to be modeled in a similar manner.

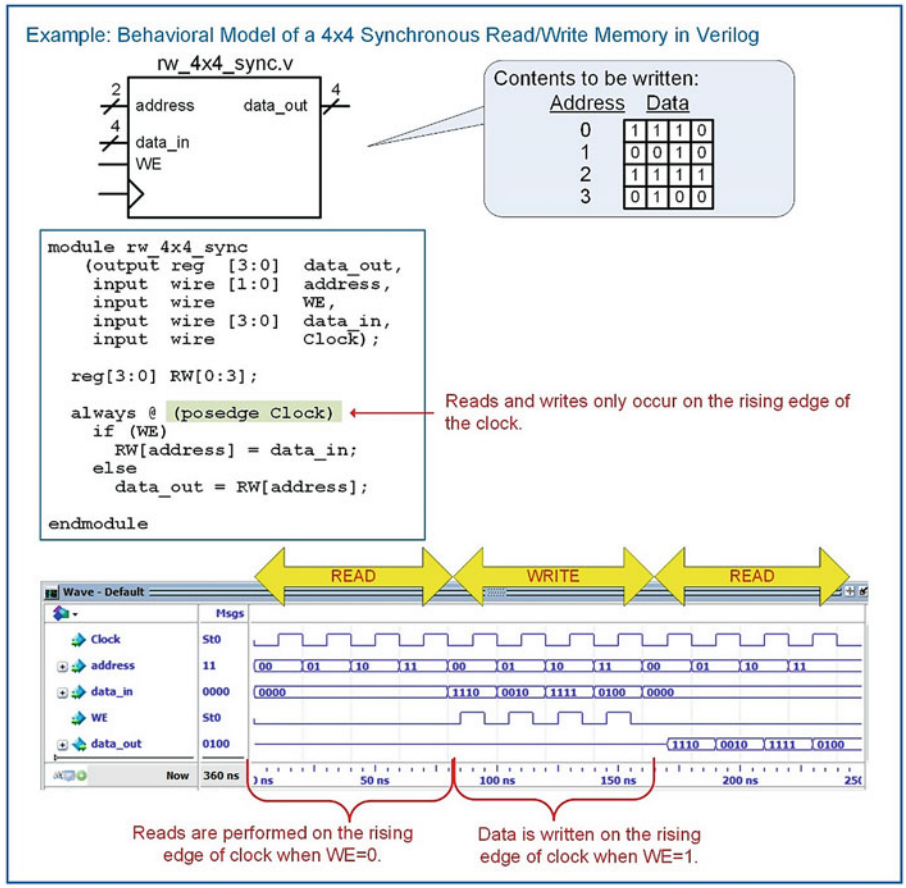
10.3 Modeling Read/Write Memory

In a simple read/write memory model, there is an output port that provides data when reading (data_out) and an input port that receives data when writing (data_in). Within the module, an array signal is declared with elements of type reg. To write to the array, signal assignment are made from the data_in port to the element within the array corresponding to the incoming address. To read from the array, the data_out port is assigned the element within the array corresponding to the incoming address. A *write enable* (WE) signal tells the system when to write to the array (WE = 1) or when to read from the array (WE = 0). In an asynchronous R/W memory, data is immediately written to the array when WE = 1 and data is immediately read from the array when WE = 0. This is modeled using a procedural block with a sensitivity list containing every input to the system. Example 10.3 shows an asynchronous R/W 4 × 4 memory system and functional simulation results. In the simulation, each address is initially read from to verify that it does not contain data. The data_out port produces unknown (X) for the initial set of read operations. Each address in the array is then written to. Finally, the array is read from verifying that the data that was written can be successfully retrieved.



Example 10.3
Behavioral model of a 4 × 4 asynchronous read/write memory in Verilog

A synchronous read/write memory is made in a similar manner with the exception that a clock is used to trigger the procedural block managing the signal assignments. In this case, the WE signal acts as a synchronous control signal indicating whether assignments are read from or written to the RW array. Example 10.4 shows the Verilog model for a synchronous read/write memory and the simulation waveform showing both read and write cycles.



Example 10.4
Behavioral model of a 4 × 4 synchronous read/write memory in Verilog

Concept Check

CC10.3 Does modeling the R/W memory as an uninitialized array accurately describe the behavior of real R/W memory technology?

(A) Yes. Read/Write memory is not initialized upon power-up.

(B) No. Read/Write memory should be initialized to all zeros to model the reset behavior found in memory.

Summary

- The term memory refers to arrays of storage. The technology used in memory is typically optimized for storage density at the expense of control capability. This is different from a D-flip-flop, which is optimized for complete control at the bit level.
- A memory device always contains an address bus input. The number of bits in the address bus dictates how many storage locations can be accessed. An *n*-bit address bus can access 2^n (or *M*) storage locations.

- The width of each storage location (N) allows the density of the memory array to be increased by reading and writing vectors of data instead of individual bits.
- A memory map is a graphical depiction of a memory array. A memory map is useful to give an overview of the capacity of the array and how different address ranges of the array are used.
- A read is an operation in which data is retrieved from memory. A write is an operation in which data is stored to memory.
- An asynchronous memory array responds immediately to its control inputs. A synchronous memory array only responds on the triggering edge of clock.
- Volatile memory will lose its data when the power is removed. Non-volatile memory will retain its data when the power is removed.
- Read-Only Memory (ROM) is a memory type that cannot be written to during normal operation. Read/Write (R/W) memory is a memory type that can be written to during normal operation. Both ROM and R/W memory can be read from during normal operation.
- Random-Access Memory (RAM) is a memory type in which any location in memory can be accessed at any time. In Sequential Access Memory the data can only be retrieved in a linear sequence. This means that in sequential memory the data cannot be accessed arbitrarily.
- ROM Memory can be modeled in Verilog using a case statement or an array data type consisting of elements of type reg that are initialized with an initial procedural block.
- R/W Memory can be modeled in Verilog using an array data type consisting of elements of type reg that are uninitialized.

Exercise Problems

Section 10.1: Memory Architecture and Terminology

10.1.1 For a 512k × 32 memory system, how many unique address locations are there? Give the exact number.

10.1.2 For a 512k × 32 memory system, what is the data width at each address location?

10.1.3 For a 512k × 32 memory system, what is the *capacity* in bits?

10.1.4 For a 512k × 32-bit memory system, what is the *capacity* in bytes?

10.1.5 For a 512k × 32 memory system, how wide does the incoming address bus need to be in order to access every unique address location?

Section 10.2: Modeling Read-Only Memory

10.2.1 Design a Verilog model for the 16 × 8, asynchronous, read-only memory system shown in Fig. 10.2. The system should contain the information provided in the memory map. Create a test bench to simulate your model by reading from each of the 16 unique addresses and observing data_out to verify it contains the information in the memory map.

Address	Data
0	x"00"
1	x"11"
2	x"22"
3	x"33"
4	x"44"
5	x"55"
6	x"66"
7	x"77"
8	x"88"
9	x"99"
10	x"AA"
11	x"BB"
12	x"CC"
13	x"DD"
14	x"EE"
15	x"FF"

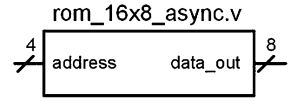


Fig. 10.2
16 x 8 Asynchronous ROM Block Diagram

10.2.2 Design a Verilog model for the 16 × 8, synchronous, read-only memory system shown in Fig. 10.3. The system should contain the information provided in the memory map. Create a test bench to simulate your model by reading from each of the 16 unique addresses and observing data_out to verify it contains the information in the memory map.

Address	Data
0	x"FF"
1	x"EE"
2	x"DD"
3	x"CC"
4	x"BB"
5	x"AA"
6	x"99"
7	x"88"
8	x"77"
9	x"66"
10	x"55"
11	x"44"
12	x"33"
13	x"22"
14	x"11"
15	x"00"

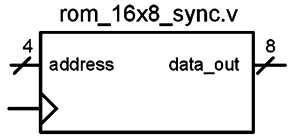


Fig. 10.3
16 x 8 Synchronous ROM Block Diagram

Section 10.3: Modeling Read/Write Memory

10.3.1 Design a Verilog model for the 16 × 8, asynchronous, read/write memory system shown in Fig. 10.4. Create a test bench to simulate your model. Your test bench should first read from all of the address locations to verify they are uninitialized. Next, your test bench should write unique information to each of the address locations. Finally, your test bench should read from each address location to verify that the information that was written was stored and can be successfully retrieved.

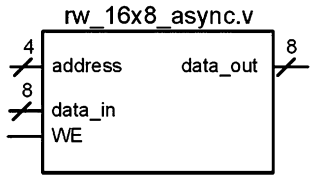


Fig. 10.4
16 x 8 Asynchronous R/W Block Diagram

10.3.2 Design a Verilog model for the 16 × 8, synchronous, read/write memory system shown in Fig. 10.5. Create a test bench to simulate your model. Your test bench should first read from all of the address locations to verify they are uninitialized. Next, your test bench should write unique information to each of the address locations. Finally, your test bench should read from each address location to verify that the information that was written was stored and can be successfully retrieved.

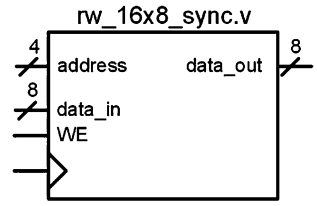


Fig. 10.5
16 x 8 Synchronous R/W Block Diagram