SEMICONDUCTOR MEMORIES

10.1. Introduction

Semiconductor memory arrays capable of storing large quantities of digital information are essential to all digital systems. The amount of memory required in a particular system depends on the type of application, but, in general, the number of transistors utilized for the information (data) storage function is much larger than the number of transistors used in logic operations and for other purposes. The ever-increasing demand for larger data storage capacity has driven the fabrication technology and memory development towards more compact design rules and, consequently, toward higher data storage densities. Thus, the maximum realizable data storage capacity of single-chip semiconductor memory arrays approximately doubles every two years. On-chip memory arrays have become widely used subsystems in many VLSI circuits, and commercially available single-chip read/write memory capacity has reached 64 megabits. This trend toward higher memory density and larger storage capacity will continue to push the leading edge of digital system design.

Memory circuits are generally classified according to the type of data storage and the type of data access. *Read-Only Memory* (ROM) circuits allow, as the name implies, only the retrieval of previously stored data and do not permit modifications of the stored information contents during normal operation. ROMs are *non-volatile* memories, i.e., the data storage function is not lost even when the power supply voltage is off. Depending on the type of data storage (data write) method, ROMs are classified as mask-programmed ROMs, Programmable ROMs (PROM), Erasable PROMs (EPROM), and Electrically Erasable PROMs (EEPROM).

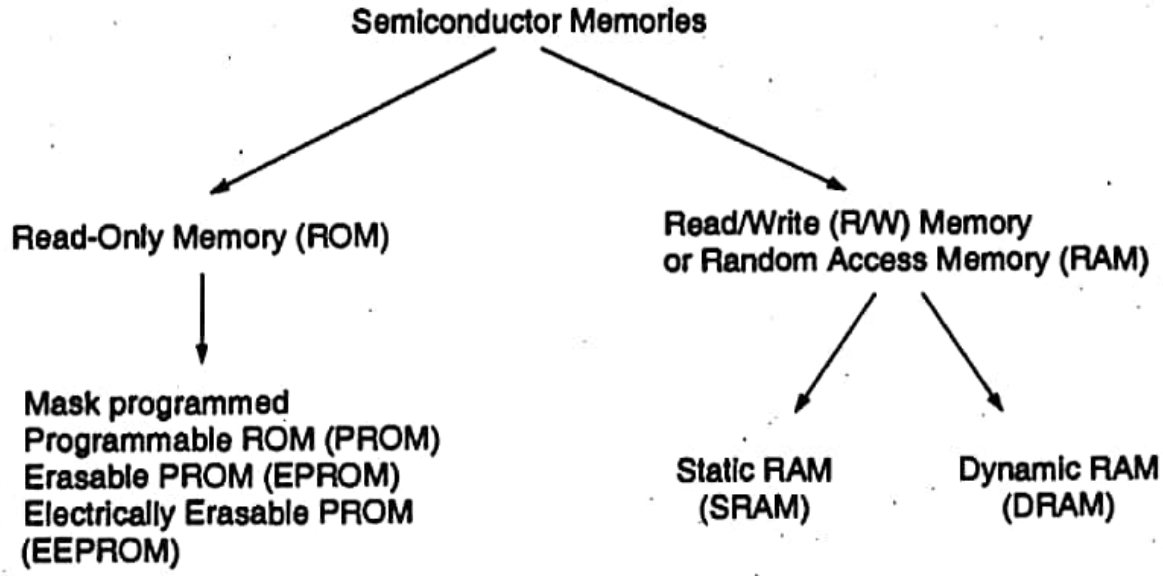


Figure 10.1. Overview of semiconductor memory types.

Read-write (R/W) memory circuits, on the other hand, must permit the modification (writing) of data bits stored in the memory array, as well as their retrieval (reading) on demand. This requires that the data storage function be *volatile*, i.e., the stored data are lost when the power supply voltage is turned off. The read-write memory circuit is commonly called *Random Access Memory* (RAM), mostly due to historical reasons. Compared to sequential-access memories such as magnetic tapes, any cell in the R/W memory array can be accessed with nearly equal access time. Based on the operation type of individual data storage cells, RAMs are classified into two main categories: *Static* RAMs (SRAM) and *Dynamic* RAMs (DRAM). Figure 10.1 shows an overview of the different memory types and their classifications.

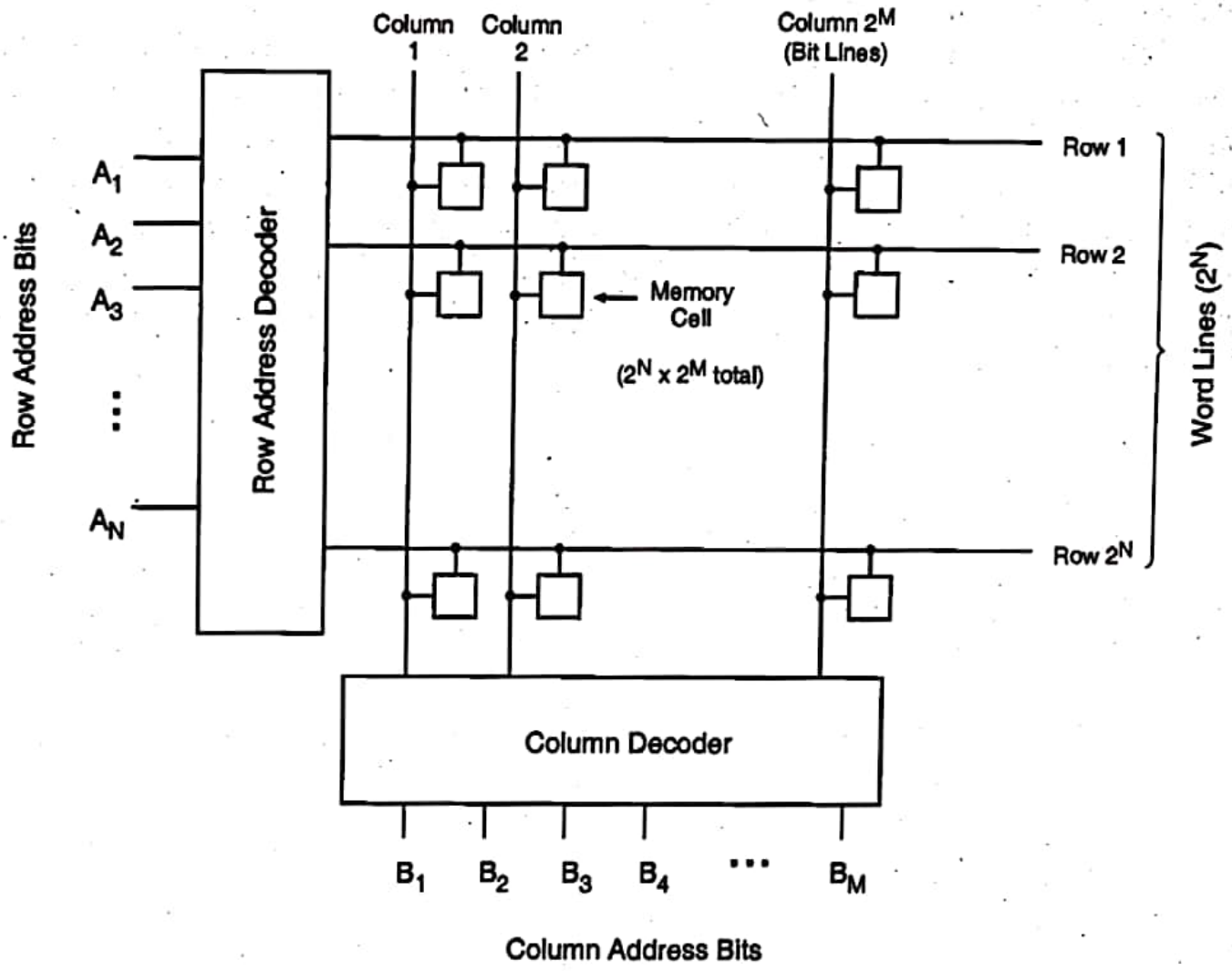


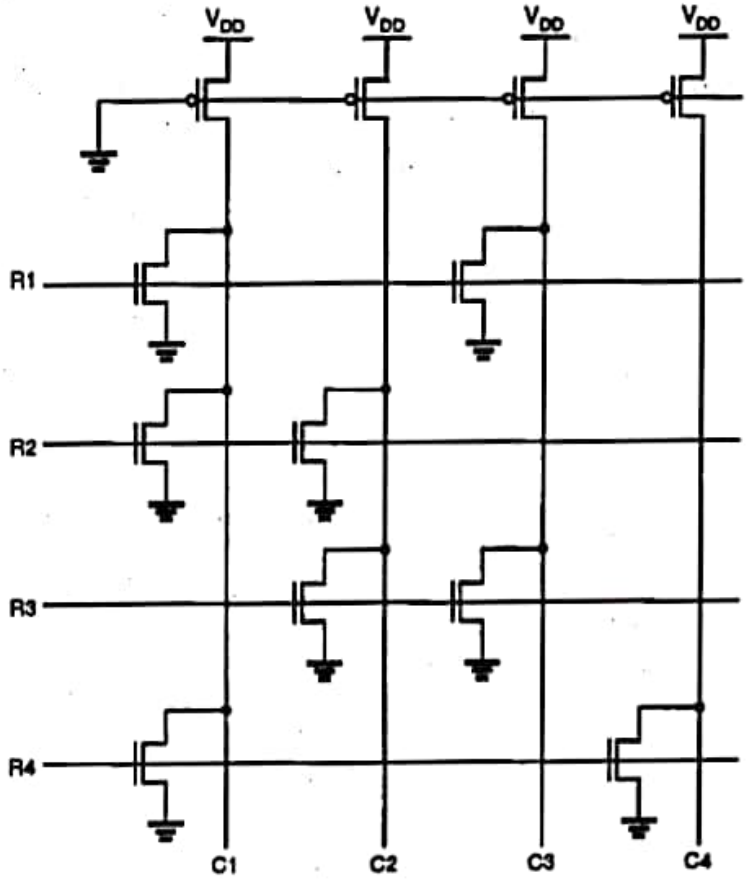
Figure 10.2. Typical random-access memory array organization.

To access a particular memory cell, i.e., a particular data bit in this array, the corresponding bit line *and* the corresponding word line must be activated (selected). The row and column selection operations are accomplished by row and column *decoders*, respectively. The row decoder circuit selects one out of 2^N word lines according to an N-bit row address, while the column decoder circuit selects one out of 2^M bit lines according to an M-bit column address. Once a memory cell or a group of memory cells are selected in this fashion, a data read and/or a data write operation may be performed on the selected single bit or multiple bits on a particular row. The column decoder circuit serves the double duties of selecting the particular columns and routing the corresponding data content in a selected row to the output.

We can see from this simple discussion that individual memory cells can be accessed for data read and/or data write operations in random order, independent of their physical locations in the memory array. Thus, the array organization examined here is called a *Random Access Memory* (RAM) structure. Notice that this organization can be used for both read-write memory arrays and read-only memory arrays. In the following sections, however, we will use the acronym RAM specifically for read-write memories, because it is the universally accepted abbreviation for this particular type of memory array.

10.2. Read-Only Memory (ROM) Circuits

The read-only memory array can also be seen as a simple combinational Boolean network which produces a specified output value for each input combination, i.e., for each address. Thus, storing binary information at a particular address location can be achieved by the presence or absence of a data path from the selected row (word line) to the selected column (bit line), which is equivalent to the presence or absence of a device at that particular location. In the following, we will examine two different implementations for MOS ROM arrays. Consider first the 4-bit x 4-bit memory array shown in Fig. 10.3. Here, each column consists of a pseudo-nMOS NOR gate driven by some of the row signals, i.e., the word lines.



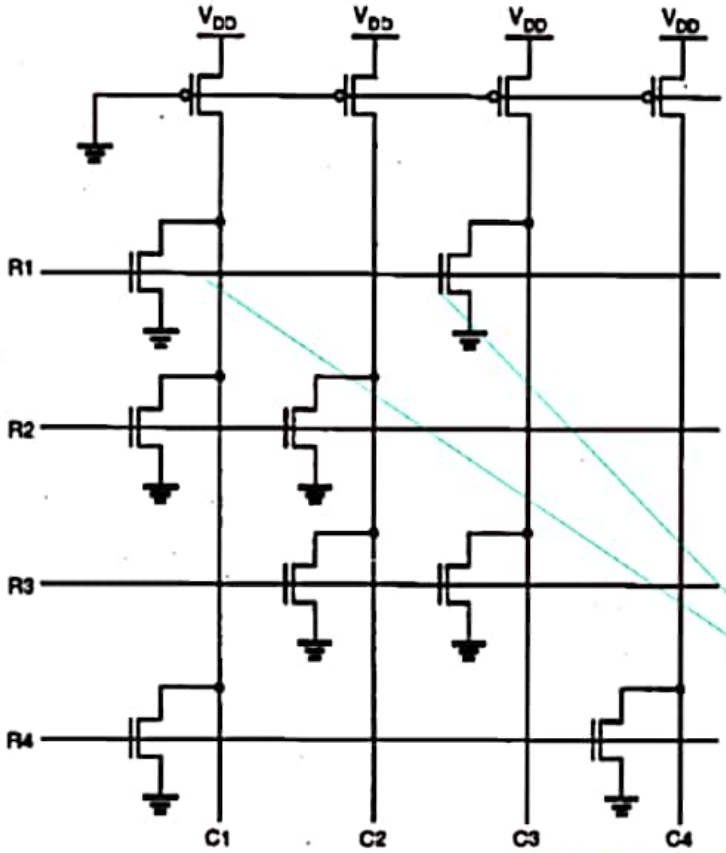
R1	R2	R3	R4	C1	C2	C3	C4
1	0	0	0	0	1	0	1
0	1	0	0	0	0	1	1
0	0	1	0	1	0	0	1
0	0	0	1	0	1	1	0

Figure 10.3. Example of a 4-bit x 4-bit NOR-based ROM array .

As described in the previous section, only one word line is activated (selected) at a time by raising its voltage to V_{DD}, while all other rows are held at a low voltage level. If an active transistor exists at the cross point of a column and the selected row, the column voltage is pulled down to the logic low level by that transistor. If no active transistor exists at the cross point, the column voltage is pulled high by the pMOS load device. Thus, a logic "1"-bit is stored as the absence of an active transistor, while a logic "0"-bit is stored as the presence of an active transistor at the crosspoint. To reduce static power consumption, the pMOS load transistors in the ROM array shown in Fig. 10.3 can also be driven by a periodic precharge signal, resulting in a *dynamic* ROM.

Row details	Column Details
$(1000)_{10}$	$(05)_{10}$
$(0100)_{10}$	$(03)_{10}$
$(0010)_{10}$	$(09)_{10}$
$(0001)_{10}$	$(06)_{10}$

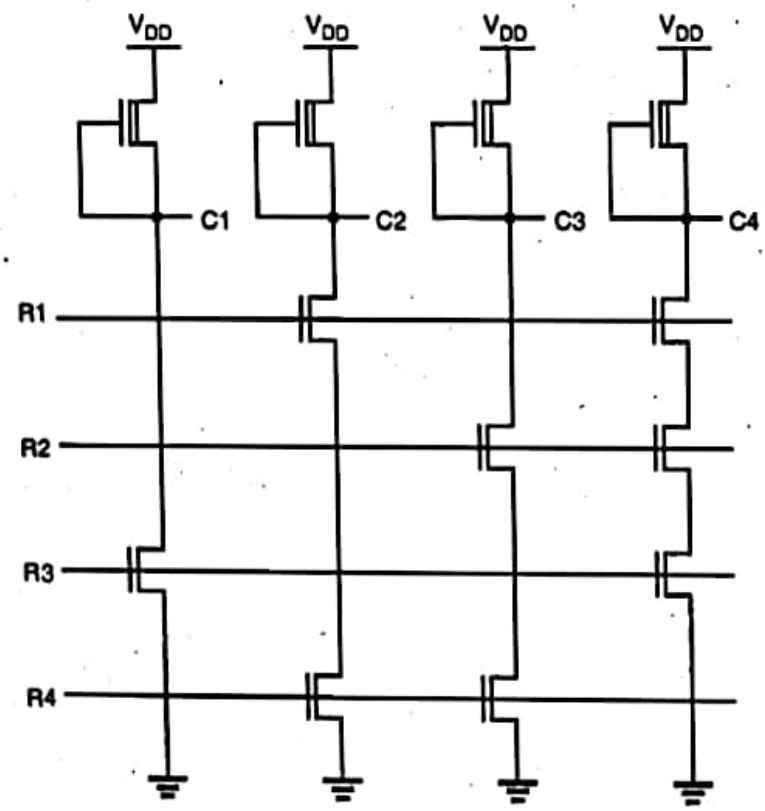
4x4-bit Nor-Based ROM Array



R1	R2	R3	R4	C1	C2	C3	C4
1	0	0	0	0	1	0	1
0	1	0	0	0	0	1	1
0	0	1	0	1	0	0	1
0	0	0	1	0	1	1	0

Add TX at 0 location

Next, we will examine a significantly different ROM array design, which is also called a NAND ROM (Fig. 10.8). Here, each bit line consists of a depletion-load NAND gate, driven by some of the row signals, i.e., the word lines. In normal operation, all word lines are held at the logic-high voltage level except for the selected line, which is pulled *down* to logic-low level. If a transistor exists at the crosspoint of a column and the selected row, that transistor is turned off and the column voltage is pulled high by the load device. On the other hand, if no transistor exists (shorted) at that particular crosspoint, the column voltage is pulled low by the other nMOS transistors in the multi-input NAND structure. Thus, a logic "1"-bit is stored by the presence of a transistor that can be deactivated, while a logic "0"-bit is stored by a shorted or normally on transistor at the crosspoint.



R1	R2	R3	R4	C1	C2	C3	C4
0	1	1	1	0	1	0	1
1	0	1	1	0	0	1	1
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	0

Figure 10.8. A 4-bit x 4-bit NAND-based ROM array.

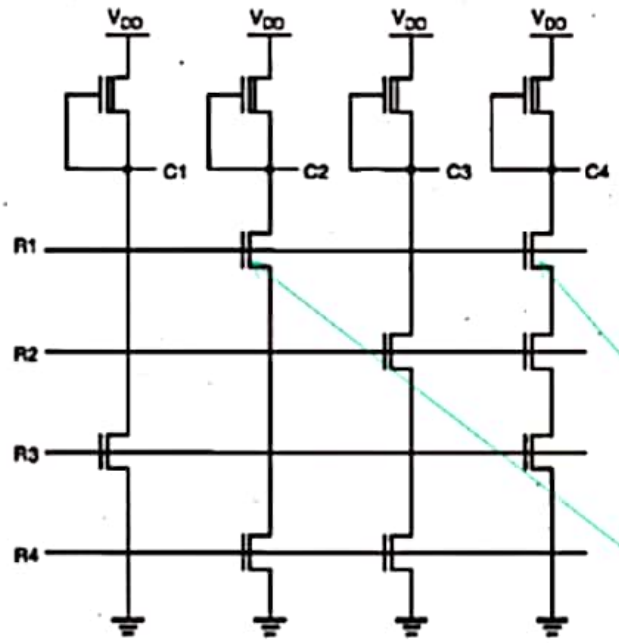
BVLSI Experiment NAND Based Memory

Before Implementing read the following instructions carefully.

The figure attached below the table has following details

Row details	Column Details
$(0111)_{10}$	$(05)_{10}$
$(1011)_{10}$	$(03)_{10}$
$(1101)_{10}$	$(09)_{10}$
$(1110)_{10}$	$(06)_{10}$

4X4 NAND-Based ROM Array

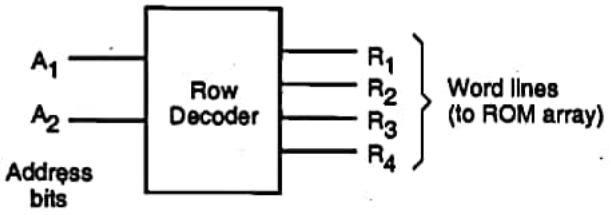


R1	R2	R3	R4	C1	C2	C3	C4
0	1	1	1	0	1	0	1
1	0	1	1	0	0	1	1
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	0

Add Tx at 1 location

Design of Row and Column Decoders

Now we will turn our attention to the circuit structures of row and column address decoders, which select a particular memory location in the array, based on the binary row and column addresses. A row decoder designed to drive a NOR ROM array must, by definition, select one of the 2^N word lines by raising its voltage to V_{OH}. As an example, consider the simple row address decoder shown in Fig. 10.10, which decodes a two-bit row address and selects one out of four word lines by raising its level.



A_1	A_2	R_1	R_2	R_3	R_4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Figure 10.10. Row address decoder example for 2 address bits and 4 word lines.

A most straightforward implementation of this decoder is another NOR array, consisting of 4 rows (outputs) and 4 columns (two address bits and their complements). Note that this NOR-based decoder array can be built just like the NOR ROM array, using the same selective programming approach (Fig. 10.11). The ROM array and its row decoder can thus be fabricated as two adjacent NOR arrays, as shown in Fig. 10.12.

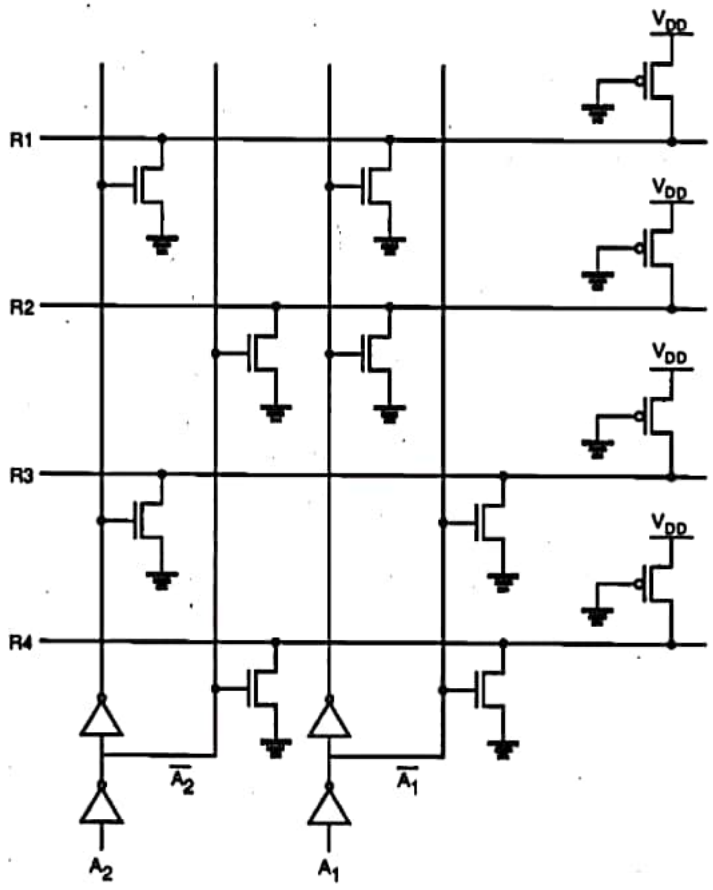


Figure 10.11. NOR-based row decoder circuit for 2 address bits and 4 word lines.

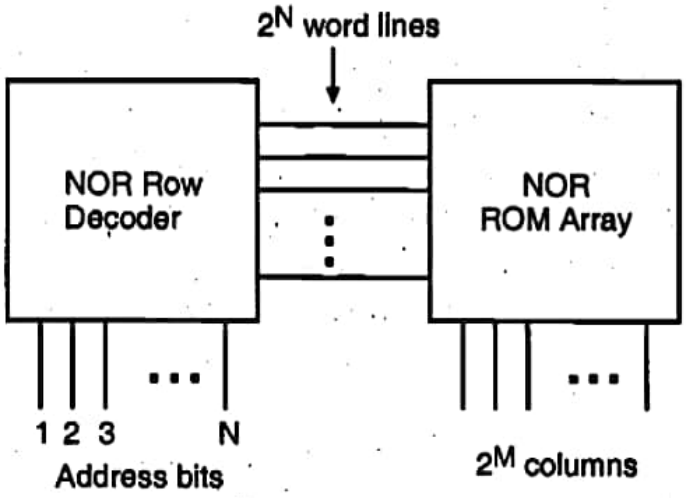


Figure 10.12. Implementation of the row decoder circuit and the ROM array as two adjacent NOR planes.

A row decoder designed to drive a NAND ROM, on the other hand, must lower the voltage level of the selected row to logic "0" while keeping all other rows at a logic-high level. This function can be implemented by using an *N*-input NAND gate for each of the row outputs. The truth table of a simple address decoder for four rows and the double NAND-array implementation of the decoder and the ROM are shown in Fig. 10.13. As in the NOR ROM case, the row address decoder of the NAND ROM array can thus be realized using the same layout strategy as the memory array itself.

A_1	A_2	R_1	R_2	R_3	R_4
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

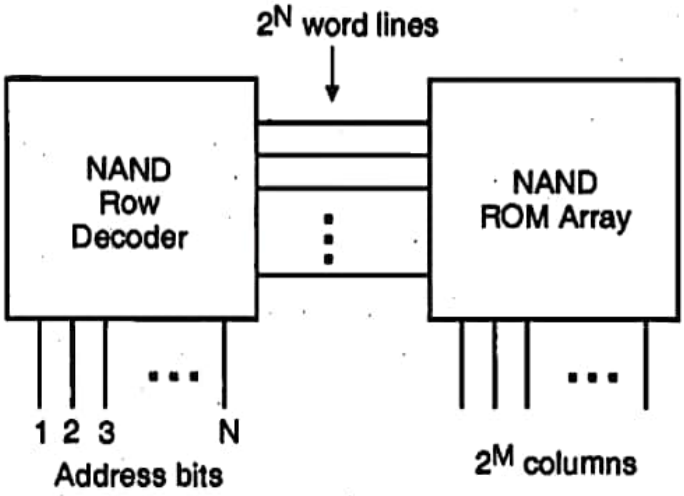


Figure 10.13. Truth table of a row decoder for the NAND ROM array, and implementation of the row decoder circuit and the ROM array as two adjacent NAND planes.

The column decoder circuitry is designed to select one out of 2^M bit lines (columns) of the ROM array according to an M-bit column address, and to route the data content of the selected bit line to the data output. A straightforward but costly approach would be to connect an nMOS pass transistor to each bit-line (column) output, and to selectively drive one out of 2^M pass transistors by using a NOR-based column address decoder, as shown in Fig. 10.14. In this arrangement, only one nMOS pass transistor is turned on at a time, depending on the column address bits applied to the decoder inputs. The conducting pass transistor routes the selected column signal to the data output. Similarly, a number of columns can be chosen at a time, and the selected columns can be routed to a *parallel* data output port.

Note that the number of transistors required for this column decoder implementation is $2^M(M+1)$, i.e., 2^M pass transistors for each bit line and M 2^M transistors for the decoder circuit. This number can quickly become excessive for large M, i.e., for a large number of bit lines.

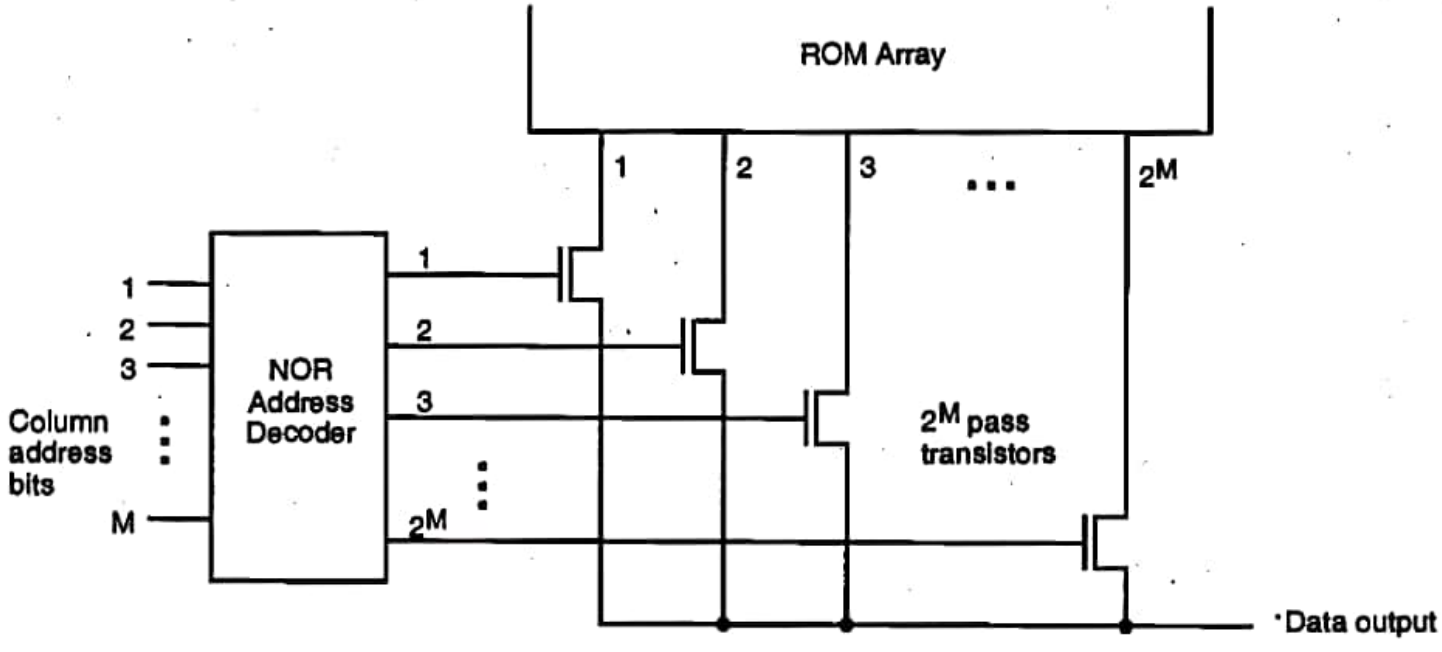


Figure 10.14. Bit-line (column) decoder arrangement using a NOR address decoder and nMOS pass transistors for every bit line.