

# Array Subsystems

9

## 9.1 Introduction

Memory arrays often account for the majority of transistors in a CMOS system-on-chip. Arrays may be divided into categories as shown in Figure 9.1. *Programmable Logic Arrays* (PLAs) perform logic rather than storage functions, but are also discussed in this chapter.

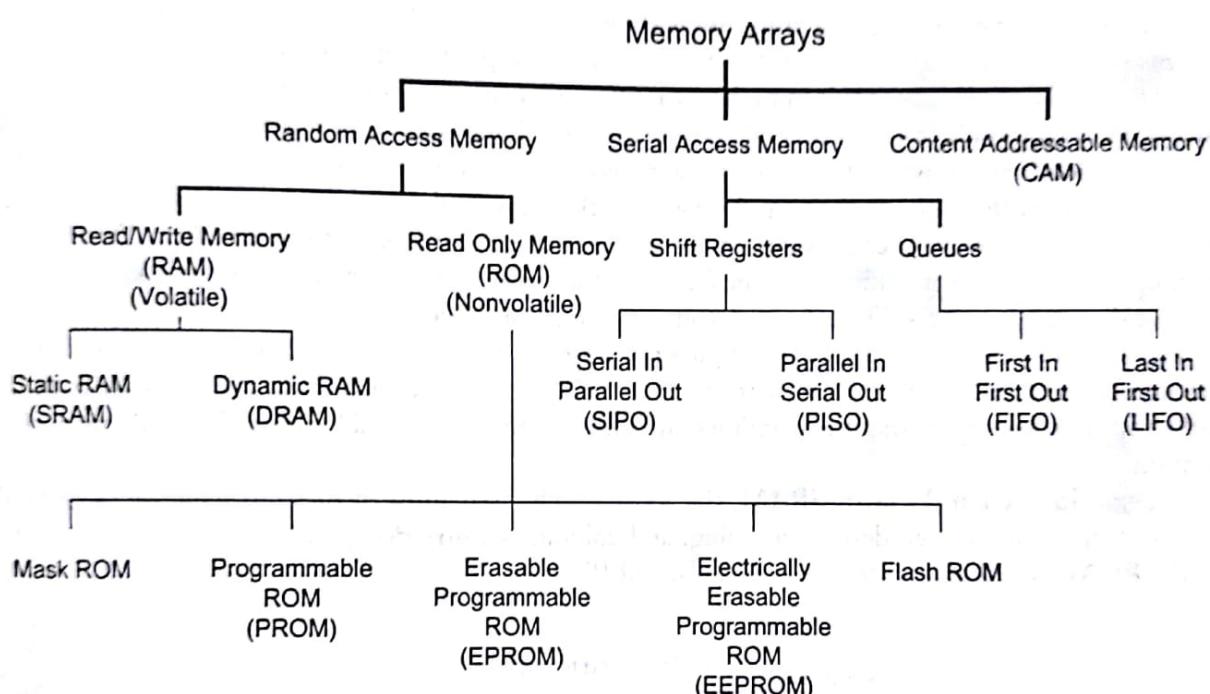


FIG 9.1 Categories of memory arrays

*Random access memory* is accessed with an *address* and has a latency independent of the address. In contrast, *serial access memories* are accessed sequentially so no address is necessary. *Content addressable memories* determine which address(es) contain data that matches a specified *key*.

Random access memory is commonly classified as *read-only memory* (ROM) or *read/write memory* (confusingly called RAM). Even the term ROM is misleading because many ROMs can be written as well. A more useful classification is *volatile vs. nonvolatile* memory. Volatile memory retains its data as long as power is applied, while nonvolatile memory will hold data indefinitely. RAM is synonymous with volatile memory, while ROM is synonymous with nonvolatile memory.

Like sequencing elements, the memory cells used in volatile memories can further be divided into *static* structures and *dynamic* structures. Static cells use some form of feedback to maintain their state, while dynamic cells use charge stored on a floating capacitor through an access transistor. Charge will leak away

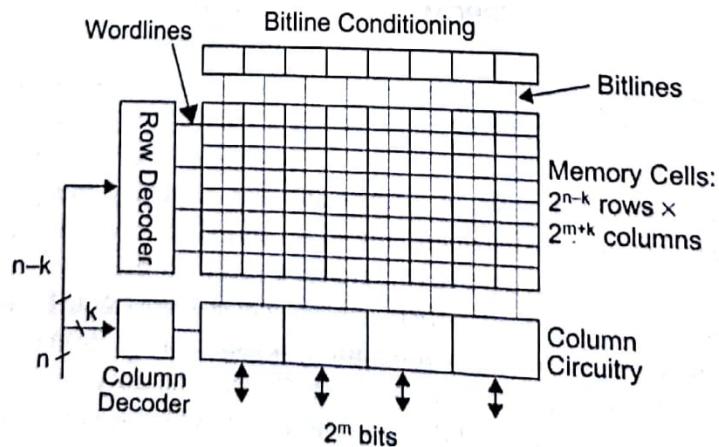
through the access transistor even while the transistor is OFF, so dynamic cells must be periodically read and rewritten to refresh their state. Static RAMs (SRAMs) are faster and less troublesome, but require more area per bit than their dynamic counterparts (DRAMs).

Some nonvolatile memories are indeed read-only. The contents of a mask ROM are hardwired during fabrication and cannot be changed. But many nonvolatile memories can be written, albeit more slowly than their volatile counterparts. A programmable ROM (PROM) can be programmed once after fabrication by blowing on-chip fuses with a special high programming voltage. An erasable programmable ROM (EPROM) is programmed by storing charge on a floating gate. It can be erased by exposure to ultraviolet (UV) light for several minutes to knock the charge off the gate. Then the EPROM can be reprogrammed. Electrically erasable programmable ROMs (EEPROMs) are similar, but can be erased in microseconds with on-chip circuitry. Flash memories are a variant of EEPROM that erases entire blocks rather than individual bits. Sharing the erase circuitry across larger blocks reduces the area per bit. Because of their good density and easy in-system reprogrammability, Flash memories have replaced other nonvolatile memories in most modern CMOS systems.

Memory cells can have one or more *ports* for access. On a read/write memory, each port can be read-only, write-only, or capable of both read and write.

A typical small memory array architecture is shown in Figure 9.2. Central to the design is a memory array consisting of  $2^n$  words of storage of  $2^m$  bits each. In the simplest design, the array is organized with one row per word and one column per bit in each word. Often there are far more words in the memory than bits in each word, which would lead to a very tall, skinny memory that is hard to fit in the chip floorplan and slow because of the long vertical wires. Therefore, the array is often folded into fewer rows of more columns. After folding, each row of the memory contains  $2^k$  words, so the array is physically organized as  $2^{n-k}$  rows of  $2^{m+k}$  columns or bits. The row decoder activates one of the rows by asserting one of the *wordlines*. During a read operation, the cells on this wordline drive the *bitlines*, which may have been conditioned to a known value in advance of the memory access. The column decoder controls a multiplexer in the column circuitry to select  $2^m$  bits from the row as the data to access. The figure illustrates an 8-word by 4-bit 2-way multiplexed memory folded into a 4-row by 8-column array with  $n = 3$ ,  $m = 2$ ,  $k = 1$ . Larger memories are generally built from multiple smaller subarrays so that the wordlines and bitlines remain reasonably short, fast, and low in power dissipation.

We begin in Section 9.2 with SRAM, the most widely used form of on-chip memory. SRAM also illustrates all the issues of cell design, decoding, and column circuitry design. Subsequent sections address DRAMs, ROMs, serial access memories, CAMs, and PLAs.



**FIG 9.2** General memory array architecture

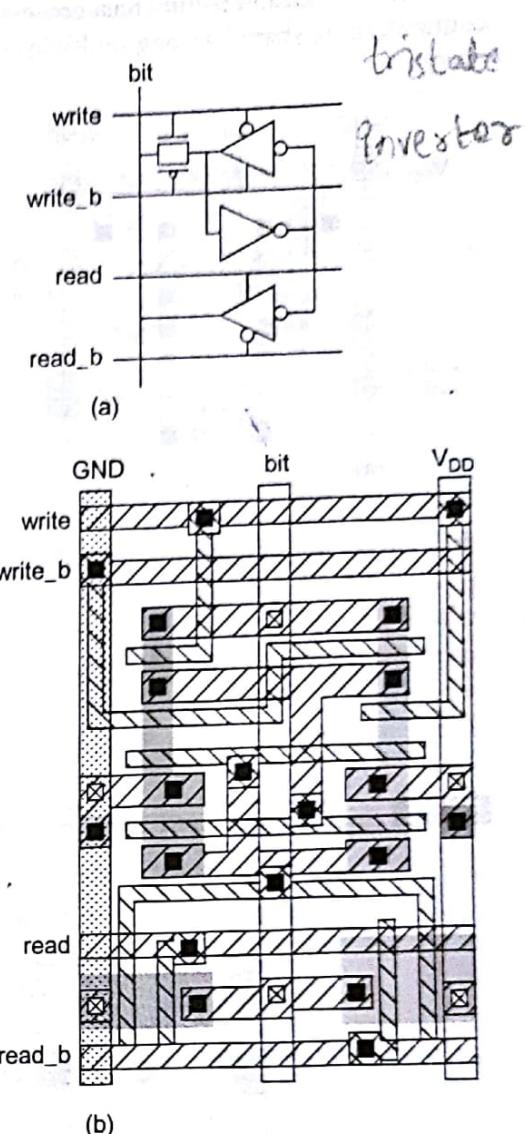
## 9.2 SRAM

The fundamental building block of a static RAM is the SRAM memory cell. The cell is activated by raising the wordline and is read or written through the bitline. Figure 9.3(a) shows a 12-transistor SRAM cell built from a simple static latch and tristate inverter. The cell has a single bitline. True and complementary read and write signals are used in place of a single wordline. A representative layout in Figure 9.3(b) has an area of  $46 \times 75 \lambda$ . The power and ground lines can be shared between mirrored adjacent cells, but the area is still limited by the wires and is undesirably large. However, the cell is easy to design because all nodes swing rail-to-rail and it is fast when used in small RAMs and register files.

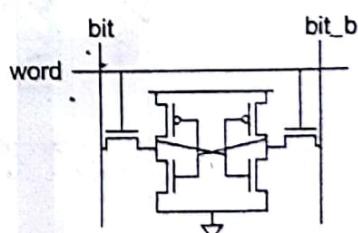
Figure 9.4 shows a 6-transistor (6T) SRAM commonly used in practice. Such a cell uses a single wordline and both true and complementary bitlines. The complementary bitline is often called bit\_b or bit. The cell contains a pair of cross-coupled inverters and an access transistor for each bitline. True and complementary versions of the data are stored on the cross-coupled inverters. If the data is disturbed slightly, positive feedback around the loop will restore it to  $V_{DD}$  or GND. The wordline is asserted to read or write the cell. The nMOS access transistors are best at passing '0's. For reads, the bitlines are initially precharged high and one is pulled down by the SRAM cell through the access transistor. For writes, the bitline or its complement is actively driven low and this low value overpowers the cell to write the new value. Careful choice of transistor sizes is necessary for correct operation, as will be examined in Section 9.2.1. The 6T cell achieves its compactness at the expense of more complex peripheral circuitry for reading and writing the cells. This is a good tradeoff in large RAM arrays where the cell size dominates the area. The small cell size also offers shorter wires and hence lower power consumption.

SRAM cells require clever layout to achieve good density. Figure 9.5(a) shows a stick diagram of a typical design. The cell is designed to be mirrored and overlapped to share  $V_{DD}$  and GND lines between adjacent cells along the cell boundary, as shown in Figure 9.5(b). Note how a single diffusion contact to the bitline is shared between a pair of cells. This halves the diffusion capacitance, and hence reduces the delay discharging the bitline during a read access. The wordline is run in both metal1 and polysilicon; the two layers must occasionally be strapped (e.g., every four or eight cells). Sample layouts derived from the stick diagram are shown in Figure 9.6.

Figure 9.6(a) shows a conservative cell of  $26 \times 45 \lambda$ , obeying the MOSIS submicron design rules (see also inside front cover). In this layout, the metal1 and polysilicon wordlines are contacted in each cell. The substrate and well are also contacted in each cell. Figure 9.6(b) shows a scanning electron micrograph of the polysilicon and diffusion layers in a more aggressive ( $20 \times 22 \lambda$ ) cell in the LSI Logic 130 nm process [Kong01w]. SRAM is so important that design rules are very carefully studied and bent where possible to

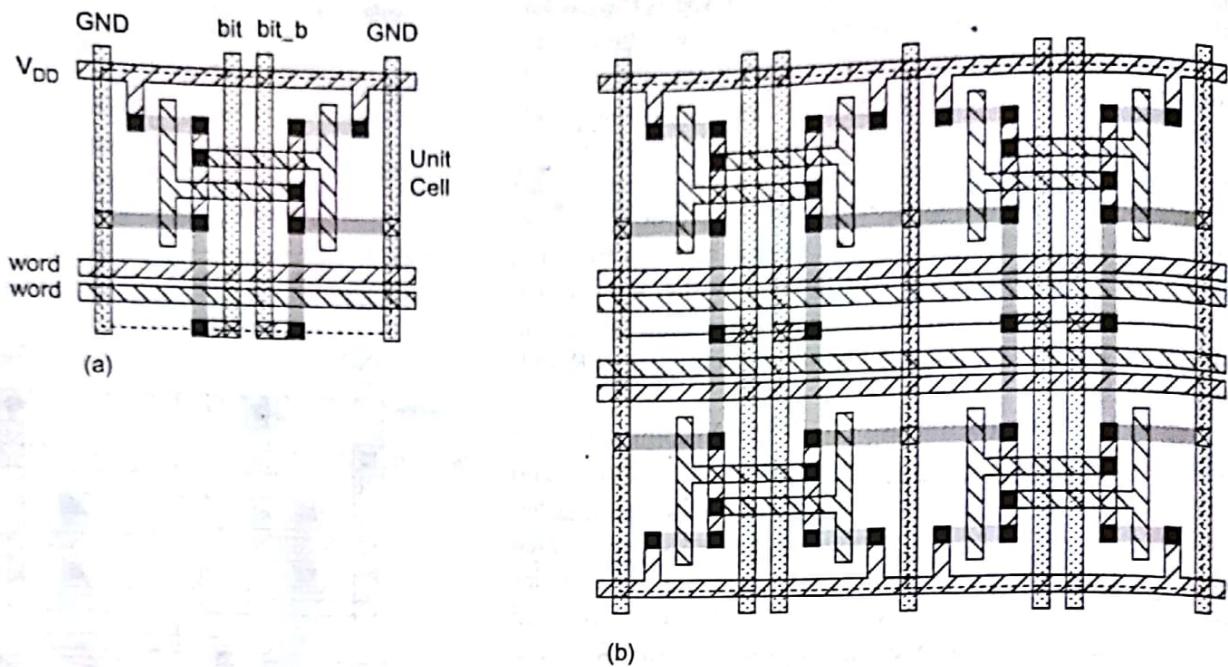


**FIG 9.3** 12-transistor SRAM cell

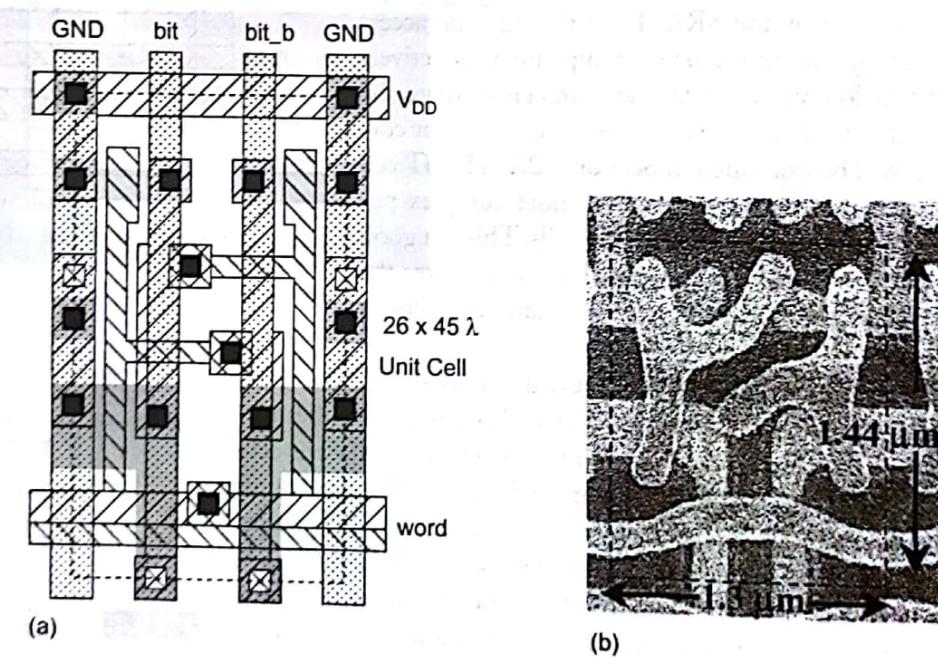


**FIG 9.4** 6-transistor SRAM cell

minimize cell area in commercial processes. Notice that diagonal lines are also used. Moreover, each substrate contact can be shared among multiple cells to save area at the expense of regularity.



**FIG 9.5** Stick diagram of 6T SRAM cell



**FIG 9.6** Layout of 6T SRAM cell. © IEEE 2001. (See color version given at the beginning of the book.)

SRAM operation is divided into two phases. As described in Section 7.4.3, the phases will be called  $\phi_1$  and  $\phi_2$ , but may actually be generated from  $clk$  and its complement  $clkb$ . Assume that in phase 2, the SRAM is precharged. In phase 1, the SRAM is written or read by raising the appropriate wordline and either driving

the bitlines to the value that should be written or leaving the bitlines floating and observing which one is pulled down. Reading a large SRAM can be slow because the capacitance of all the cells sharing the bitline is large. Sense amplifiers accelerate reads by detecting small differences between the bitline and its complement. The following sections discuss the role of each block from Figure 9.2 in the SRAM operation.

### 9.2.1 Memory Cell Read/Write Operation

Figure 9.7 shows a SRAM cell being read. The bitlines are both initially floating high. Without loss of generality, assume  $A$  is initially '0' and thus  $A_b$  is initially '1.'  $A_b$  and  $bit_b$  both should remain '1'. When the wordline is raised,  $bit$  should be pulled down through transistors  $N1$  and  $N2$ . At the same time  $bit$  is being pulled down, node  $A$  tends to rise.  $A$  is held low by  $N1$ , but raised by current flowing in from  $N2$ . Hence,  $N1$  must be stronger than  $N2$ . Specifically, the transistors must be ratioed such that node  $A$  remains below the switching threshold of the  $P2/N3$  inverter. This constraint is called *read stability*. Waveforms for the read operation are shown in Figure 9.7(b) as a 0 is read onto  $bit$ . Observe that  $A$  momentarily rises, but does not glitch badly enough to flip the cell.

Figure 9.8 shows the same cell in the context of a full column from the SRAM. During phase 2, the bitlines are precharged high. The wordline only rises during phase 1; hence, it can be viewed as a  $_q1$  qualified clock. Many SRAM cells share the same bitline pair, which acts as a distributed dual-rail footless dynamic multiplexer. The capacitance of the entire bitline must be discharged through the access transistor. The output can be sensed by a pair of HI-skew inverters. By raising the switching threshold of the sense inverters, delay can be reduced at the expense of noise margin. The outputs are dual-rail monotonically rising signals, just as in a domino gate.

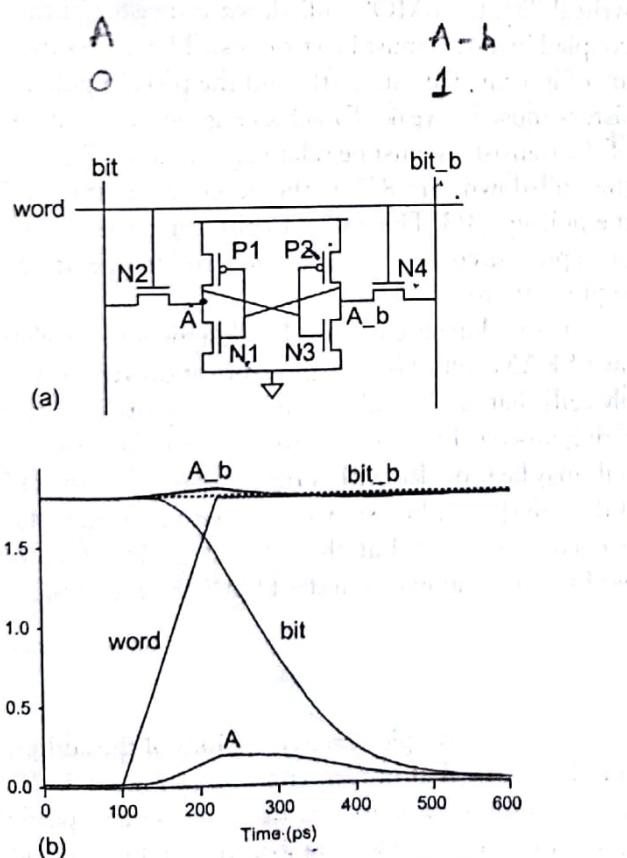


FIG 9.7 Read operation for 6T SRAM cell

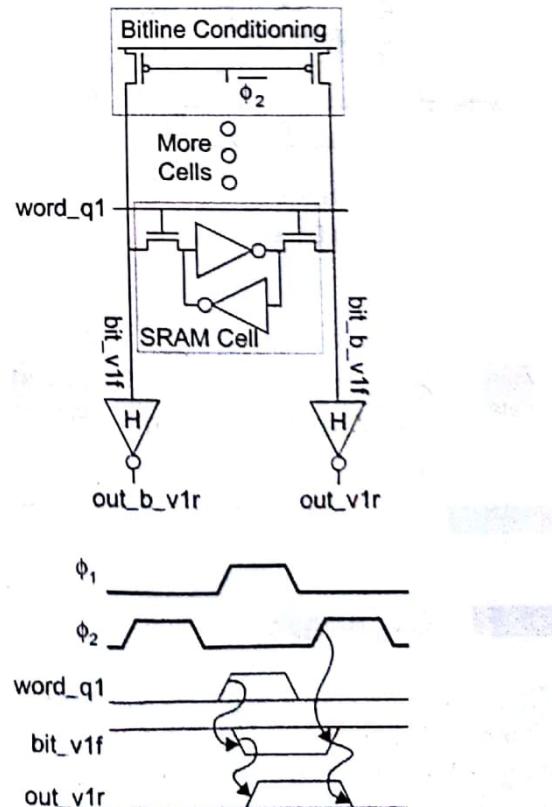


FIG 9.8 SRAM column read

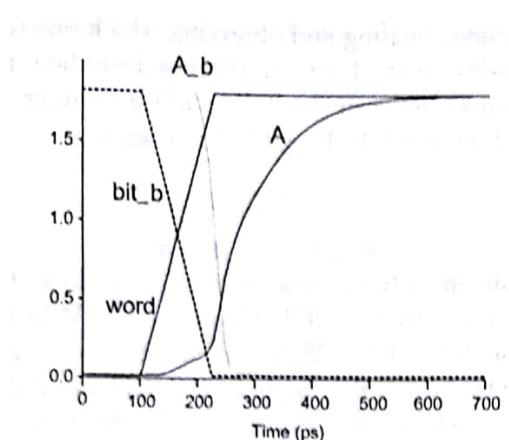


FIG 9.9 Write operation for 6T SRAM cell

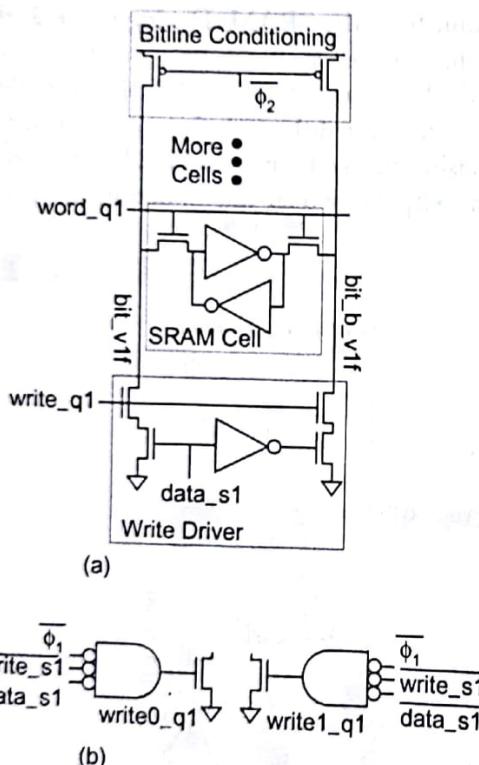


FIG 9.10 SRAM column write

### 9.2.2 Decoders

The simplest decoder is a collection of AND gates using true and complementary versions of the address bits. Figure 9.11 shows several straightforward implementations of the address lines. The first implementation in Figure 9.11(a) is a static NAND gate followed by an inverter. This structure is useful for up to 5–6 inputs or more if speed is not critical. The NAND transistors are usually made minimum size to reduce the load on the buffered address lines because there are  $2^{n-k}$  transistors on each true and complementary address line in the row decoder. The second implementation shown in Figure 9.11(b) uses a pseudo-nMOS NOR gate buffered with two inverters. The NOR gate transistors can be made minimum size and the inverters can be scaled appropriately to drive the wordline.

The waveforms of Figure 9.9 show the SRAM cell being written. Again, assume  $A$  is initially '0' and that we wish to write a '1' into the cell.  $bit$  is precharged high and left floating.  $bit\_b$  is pulled low by a write driver. We know on account of the read stability constraint that  $bit$  will be unable to force  $A$  high through  $N2$ . Hence, the cell must be written by forcing  $A\_b$  low through  $N4$ .  $P2$  opposes this operation; thus,  $P2$  must be weaker than  $N4$  so that  $A\_b$  can be pulled low enough. This constraint is called writeability. Once  $A\_b$  falls low,  $N1$  turns OFF and  $P1$  turns ON, pulling  $A$  high as desired.

Figure 9.10(a) again shows the cell in the context of a full column from the SRAM. During phase 2, the bitlines are precharged high. Write drivers pull the bitline or its complement low during phase 1 to write the cell. The write drivers can consist of a pair of transistors on each bitline for the data and the write enable, or a single transistor driven by the appropriate combination of signals (Figure 9.10(b)). In either case, the series resistance of the write driver, bitline wire, and access transistor must be low enough to overpower the pMOS transistor in the SRAM cell.

In summary, to ensure both read stability and writeability, the nMOS pull-down transistor in the cross-coupled inverters must be strongest. The access transistors are of intermediate strength, and the pMOS pull-up transistors must be weak. To achieve good layout density, all of the transistors must be relatively small. In Figure 9.6(a), the pull-downs are  $8/2 \lambda$ , the access transistors  $4/2$ , and the pull-ups  $3/3$ . The SRAM cells must operate correctly in all process corners at all voltages and temperatures. This requires thorough simulation.

It is no longer common for designers to develop their own SRAM cells. Usually, the fabrication vendor will supply cells that are carefully tuned to the particular manufacturing process. In high-performance processes, two or more cells may be provided with different speed/density tradeoffs. [Glasser85] describes SRAM cells with four transistors and two large resistors, but the 6T cell is almost universally used in contemporary standard CMOS processes.

and one access must wait. Increasing the number of banks offers more parallelism and lower probability of bank conflicts.

### 9.3 DRAM

Dynamic RAMs (DRAMs) store their contents as charge on a capacitor rather than in a feedback loop. Thus, the basic cell is substantially smaller than SRAM, but the cell must be periodically read and refreshed so that its contents do not leak away. Commercial DRAMs are built in specialized processes optimized for dense capacitor structures. They offer an order of magnitude greater density ( $\text{bits}/\text{cm}^2$ ) than high-performance SRAM built in a standard logic process, but they also have much higher latency. DRAM circuit design is a very specialized art and is the topic of excellent books such as [Keeth01]. This section provides an overview of the general technique.

A 1-transistor (1T) dynamic RAM cell consists of a transistor and a capacitor, as shown in Figure 9.25(a). Like SRAM, the cell is accessed by asserting the wordline to connect the capacitor to the bitline. On a read, the bitline is first precharged to  $V_{DD}/2$ . When the wordline rises, the capacitor shares its charge with the bitline, causing a voltage change  $\Delta V$  that can be sensed, as shown in Figure 9.25(b). The read disturbs the cell contents at  $x$ , so the cell must be rewritten after each read. On a write, the bitline is driven high or low and the voltage is forced onto the capacitor! Some DRAMs drive the wordline to  $V_{DDP} = V_{DD} + V_r$  to avoid a degraded level when writing a '1.'

The DRAM capacitor  $C_{cell}$  must be as physically small as possible to achieve good density. However, the bitline is contacted to many DRAM cells and has a relatively large capacitance  $C_{bit}$ . Therefore, the cell capacitance is typically much smaller than the bitline capacitance. According to the charge-sharing equation, the voltage swing on the bitline during readout is

$$\Delta V = \frac{V_{DD}}{2} \frac{C_{cell}}{C_{cell} + C_{bit}} \quad (9.1)$$

We see that a large cell capacitance is important to provide a reasonable voltage swing. It also is necessary to retain the contents of the cell for an acceptably long time and to minimize soft errors, as was mentioned in Section 4.7.7. For example, 30 fF is a typical target. The most compact way to build such a high capacitance is to extend into the third dimension. For example, Figure 9.26 shows a cross-section and scanning electron microscope (SEM) image of trench capacitors etched under the source of the transistor. The walls of the trench are lined with an oxide-nitride-oxide dielectric. The trench is then filled with a polysilicon conductor that serves as one terminal of the capacitor attached to the transistor drain, while the heavily doped substrate serves as the other terminal. A variety of three-dimensional capacitor structures have been used in specialized DRAM processes that are not available in conventional CMOS processes.

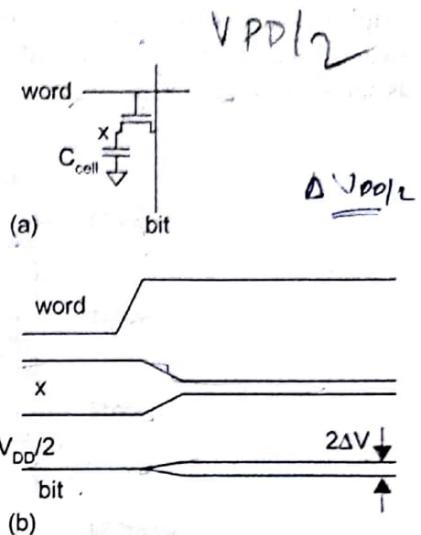


FIG 9.25 DRAM cell read operation

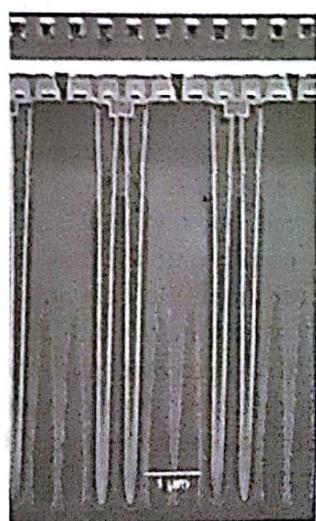
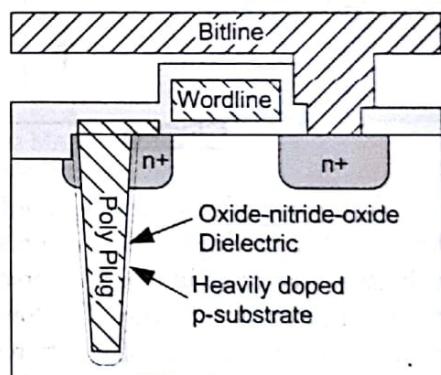


FIG 9.26 Trench capacitor

serted. This consumes very little power because the voltage is reached by sharing charge between one bitline at  $V_{DD}$  and the other at GND.

Figure 9.33 puts together the complete column circuitry serving two folded subarrays. Each subarray column produces a pair of signals,  $bit$  and  $bit^*$ . The  $CSEL$  signal, produced by the column decoder, determines if this column will be connected to the I/O line for the array. Each subarray has its own equalization transistors and pMOS portion of the sense amplifier. However, the nMOS sense amplifier and I/O lines are shared between the subarrays. Either  $ISO1$  or  $ISO2$  is asserted to connect one subarray to the I/O lines while leaving the other isolated. During a read operation, the data is read onto the I/O lines. During a write, one I/O line is driven high and the other low to force a value onto the bitlines. The cross-coupled pMOS transistors pull the bitlines to a full logic level during a write to compensate for the threshold drop through the isolation transistor.

## 9.4 Read-only Memory

(Read-only Memory (ROM) cells can be built with only one transistor per bit of storage) A ROM is a nonvolatile memory structure in that the state is retained indefinitely even without power. A ROM array is commonly implemented as a single-ended NOR array using any of the NOR gate structures studied so far, including the pseudo-nMOS and the footless-dynamic NOR gate. As in SRAM cells and other footless dynamic gates, the wordline input must be low during precharge on dynamic NOR gates. In situations where DC power dissipation is acceptable and the speed is sufficient, the pseudo-nMOS ROM is the easiest to design, requiring no timing. (The DC power dissipation can be significantly reduced in multiplexed ROMs by placing the pull-up transistors after the column multiplexer)

Figure 9.34 shows a 4-word by 6-bit ROM using pseudo-nMOS pull-ups with the following contents:

word0: 010101  
word1: 011001  
word2: 100101  
word3: 101010

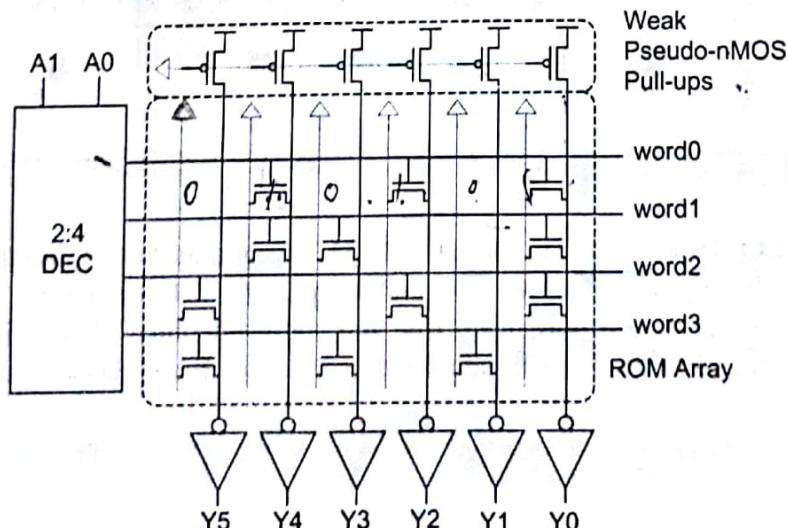


FIG 9.34 Pseudo-nMOS ROM

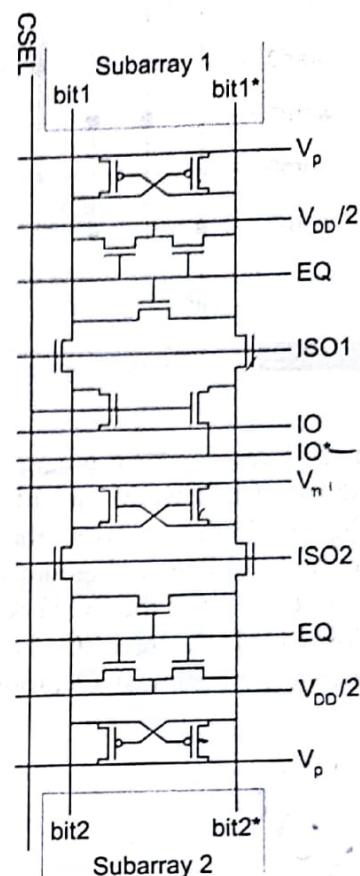
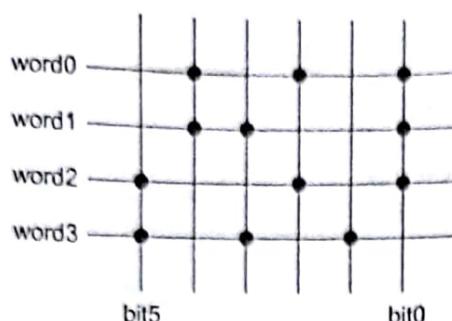


FIG 9.33 Column circuitry

100010  
001100  
100101  
101010  
  
array tunnel shifter



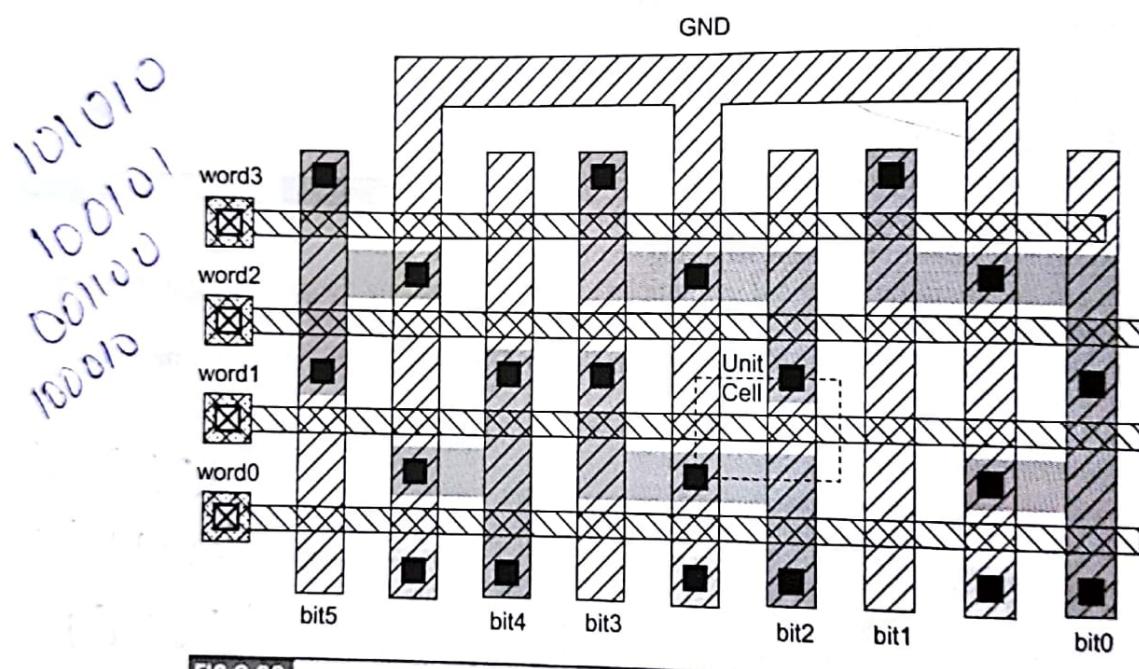
**FIG 9.35** Dot diagram representation of ROM

The contents of the ROM can be symbolically represented with a dot diagram in which dots indicate the presence of 1's, as shown in Figure 9.35. The dots actually correspond to nMOS pull-down transistors connected to the bitlines, but the outputs are inverted.

Mask-programmed ROMs can be configured by the presence or absence of a transistor or contact, or by a threshold implant that turns a transistor permanently OFF where it is not needed. Omitting transistors has the advantage of reducing capacitance on the wordlines and power consumption.

Programming with metal contacts was once popular because such ROMs could be completely manufactured except for the

metal layer, and then programmed according to customer requirements through a metallization step. The advent of EEPROM and Flash memory chips has reduced demand for such mask-programmed ROMs. Figure 9.36 shows a layout for the 4-word by 6-bit ROM array. The wordlines run horizontally in polysilicon, while the bitlines and grounds run vertically in metal1. Notice how each ground is shared between a pair of cells. Each bit of the ROM occupies a  $12 \times 8 \lambda$  cell<sup>1</sup>. Polysilicon wordlines are only appropriate for small or slow ROMs. A larger ROM can run metal2 straps over the polysilicon and contact the two periodically (e.g., every eight columns). Occasional substrate contacts are also required.

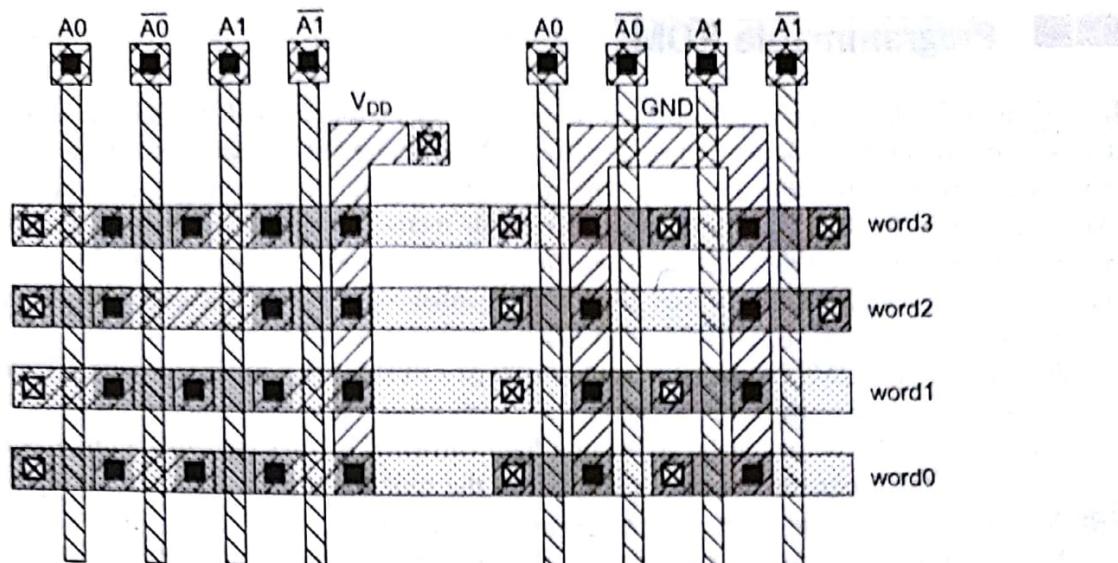


**FIG 9.36** ROM array layout

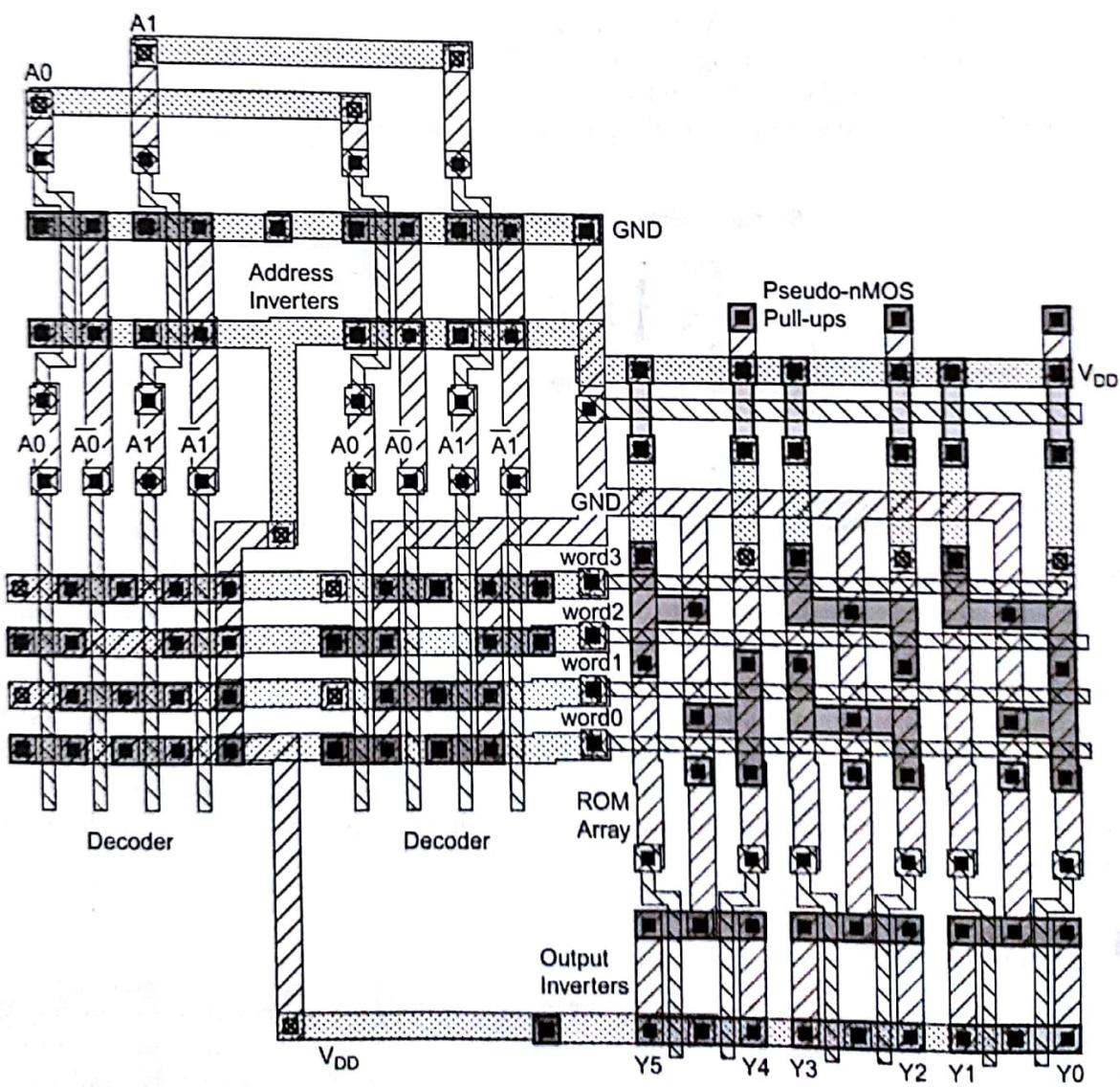
Row decoders for ROMS are similar to those for RAMs except that they are usually very constrained by the ROM wordline pitch. Figure 9.37 shows how each output of a 2:4 decoder can be shoehorned into a single horizontal track using vertical polysilicon true and complementary address lines and metal supply lines. Column decoders for ROMs are usually simpler than those for RAMs because single-ended sensing is commonly employed.

Figure 9.38 shows a complete pseudo-nMOS ROM including row decoder, cell array, pMOS pull-ups, and output inverters.

<sup>1</sup>The cell can be reduced to  $11 \times 7 \lambda$  by running the ground line in diffusion and by reducing the width and spacing to  $3 \lambda$ .



**FIG 9.37** Row decoder layout on tight pitch



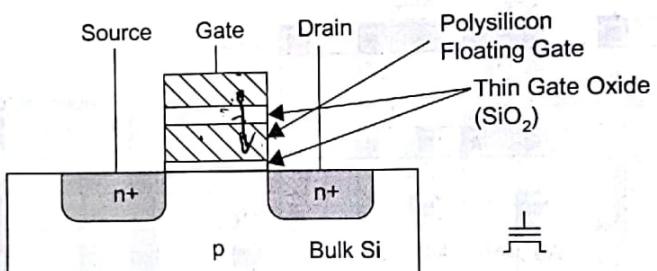
**FIG 9.38** Complete ROM layout

### 9.4.1 Programmable ROMs

(It is often desirable for the user to be able to program or reprogram a ROM after it is manufactured.) As discussed in Section 9.1, this contradicts the "Read Only" nature of the device; ROM has in practice become synonymous with *nonvolatile*, not read-only memory. Programming/writing speeds are generally slower than read speeds for ROMs. Four types of nonvolatile memories include Programmable ROMs (PROMs), Erasable Programmable ROMs (EPROMs), Electrically Erasable Programmable ROMs (EEPROMs), and Flash memories. All of these memories require some enhancements to a standard CMOS process: PROMs use fuses while EPROMs, EEPROMs, and Flash use charge stored on a floating gate.

(Programmable ROMs can be fabricated as ordinary ROMs fully populated with pull-down transistors in every position. Each transistor is placed in series with a fuse made of polysilicon, nichrome, or some other conductor that can be burnt out by applying a high current.) The user typically configures the ROM in a specialized PROM programmer before putting it in the system. (As there is no way to repair a blown fuse, PROMs are also referred to as *one-time programmable* memories.)

As technology has improved, reprogrammable nonvolatile memory has largely displaced PROMs. These memories, including EPROM, EEPROM, and Flash, use a second layer of polysilicon to form a floating gate between the primary gate and the channel, as shown in Figure 9.39. The floating gate is a good conductor, but it is not attached to anything. Applying a high voltage to the upper gate causes electrons to jump through the thin oxide onto the floating gate through the processes called avalanche injection or Fowler-Nordheim tunneling. Injecting the electrons induces a negative voltage on the floating gate, effectively increasing the threshold voltage of the transistor to the point that it is always OFF.



**FIG 9.39** Cross-section of floating gate nMOS transistor

[Itoh01k] and [Rabaey03] offer a good overview of these memories. In brief, (EPROM is programmed electrically, but it is erased through exposure to ultraviolet light that knocks the electrons off the floating gate.) (It offers a dense cell, but it is inconvenient to erase and reprogram.) (EEPROM and Flash can be erased electrically without being removed from the system.) (EEPROM offers fine-grained control over which bits are erased, while Flash is erased in bulk.) (EEPROM cells are larger to enable them to provide this versatility, so Flash has become the most economical form of convenient nonvolatile storage.) For example, (Flash memory cards are widely used in digital cameras to store pictures even after the camera is turned off. Flash is also useful for firmware or configuration data because it can be rewritten to upgrade a system in the field without opening the case or removing parts.)

### 9.4.2 NAND ROMS

(The ROM from Figure 9.34 is called a NOR ROM because each of the bitlines is just a pseudo-nMOS NOR gate.) (The bitline pulls down when a wordline attached to any of the transistors is asserted high. The size of the cell is limited by the ground line.) (Figure 9.40 shows a NAND ROM that uses active-low wordlines.) (Transistors are placed in series and the transistors on the nonselected rows are ON.) (If no transistor is associated with the selected word, the bitline will pull down.) (If a transistor is present, the bitline will remain high.)

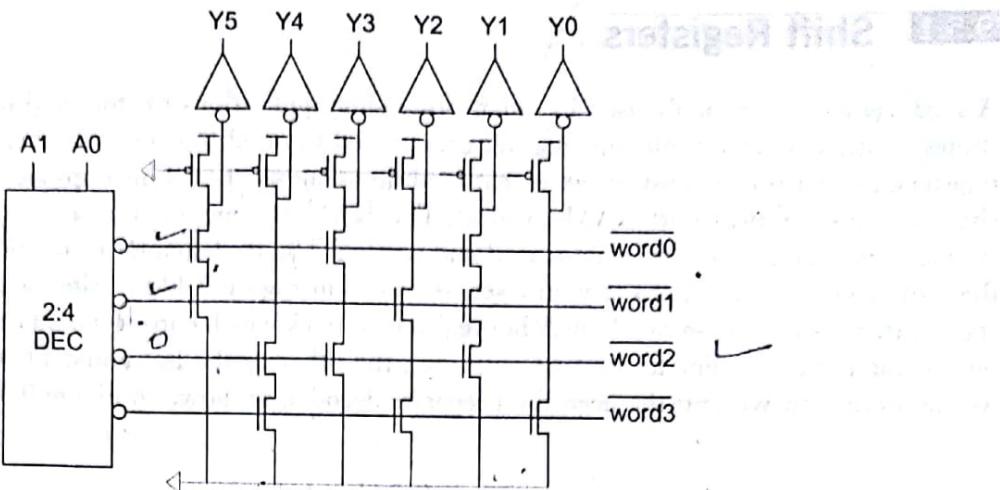


FIG 9.40 Pseudo-nMOS NAND ROM

Figure 9.41(a) shows a layout of the NAND ROM. (The cell size is only  $7 \times 8 \lambda$ .) The contents are specified by using either a transistor or a metal jumper in each bit position. (The contacts limit the cell size.) Figure 9.41(b) shows an even smaller layout in which transistors are located at every position. In this design, an extra implantation step can be used to create a negative threshold voltage, turning certain transistors permanently ON where they are not needed. In such a process, the cell size reduces to only  $6 \times 5 \lambda$ , assuming that the decoder and bitline circuitry can be built on such a tight pitch.

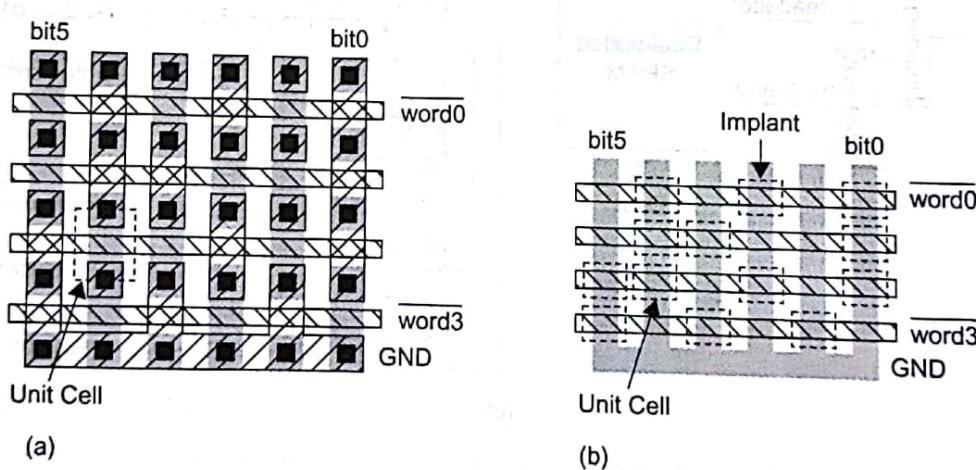


FIG 9.41 NAND ROM array layouts

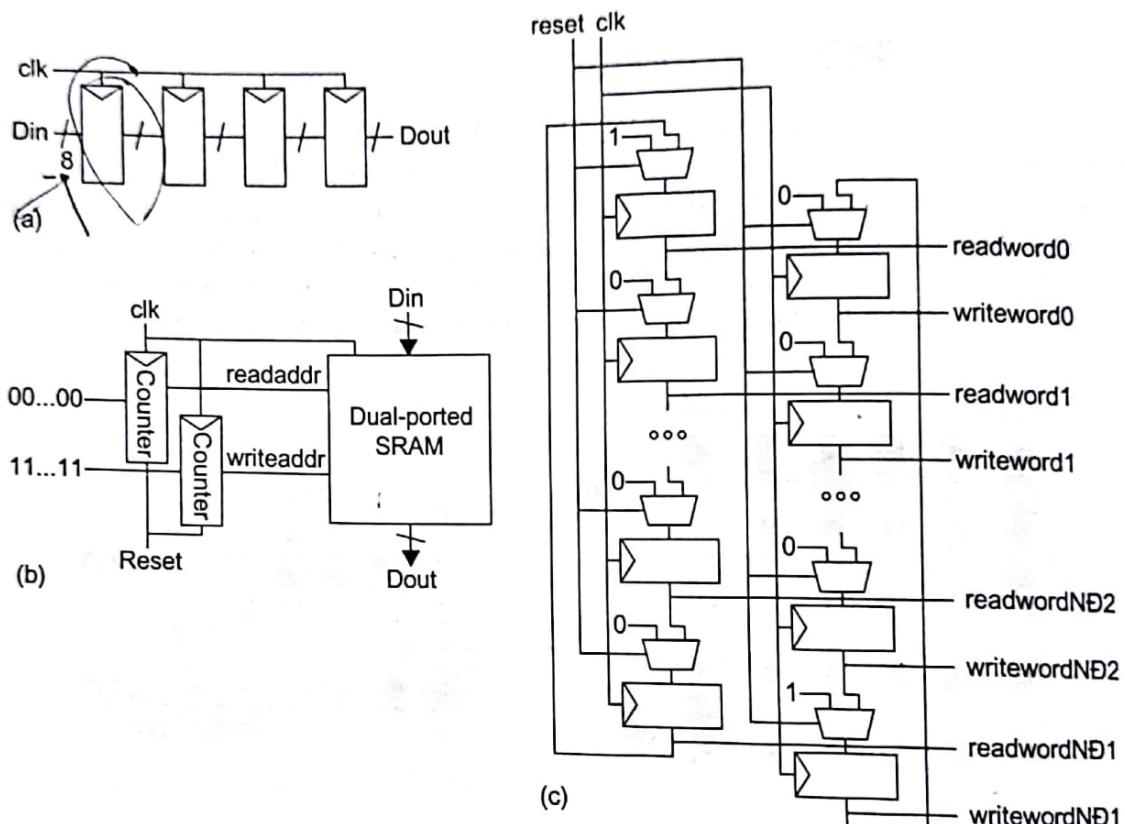
(A disadvantage of the NAND ROM is that the delay grows quadratically with the number of series transistors discharging the bitline.) NAND structures with more than 8–16 series transistors become extremely slow, so NAND ROMs are often broken into multiple small banks with a limited number of series transistors. Nevertheless, these NAND structures are attractive for Flash memories in which density and cost are more important than access time.)

## 9.5 Serial Access Memories

Using the basic SRAM cell and/or registers, we can construct a variety of serial access memories including shift registers and queues. These memories avoid the need for external logic to track addresses for reading or writing.

### 9.5.1 Shift Registers

A *shift register* is commonly used in signal-processing applications to store and delay data. Figure 9.42(a) shows a simple 4-stage 8-bit shift register constructed from 32 flip-flops. As there is no logic between the registers, particular care must be taken that hold times are satisfied. Flip-flops are rather big, so large, dense shift registers use dual-port RAMs instead. The RAM is configured as a circular buffer with a pair of counters specifying where the data is read and written. The read counter is initialized to the first entry and the write counter to the last entry on reset, as shown in Figure 9.42(b). Alternately, the counters in an  $N$ -stage shift register can use two 1-of- $N$  hot registers to track which entries should be read and written. Again, one is initialized to point to the first entry and the other to the last entry. These registers can drive the wordlines directly without the need for a separate decoder, as shown in Figure 9.42(c).



**FIG 9.42** Shift registers

One variant of a shift register is a *tapped delay line* that offers a variable number of stages of delay. Figure 9.43 shows a 64-stage tapped delay line that could be used in a video processing system. Delay blocks are built from 32-, 16-, 8-, 4-, 2-, and 1-stage shift registers. Multiplexers control pass-around of the delay blocks to provide the appropriate total delay.

Another variant is a serial/parallel memory. Figure 9.44(a) shows a 4-stage Serial In Parallel Out (SIPO) memory and Figure 9.44(b) shows a 4-stage Parallel In Serial Out (PISO) memory. These are often useful in signal processing and communications systems.

### 9.5.2 Queues (FIFO, LIFO)

Queues allow data to be read and written at different rates. Figure 9.45 shows an interface to a queue. The read and write operations each are controlled by their own clocks that may be asynchronous. The queue asserts the

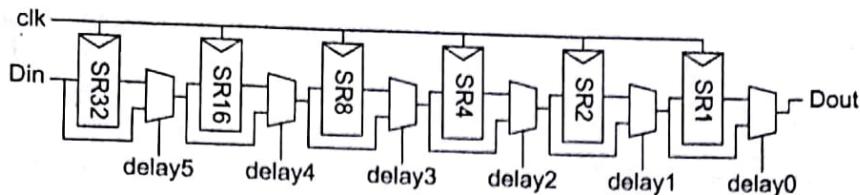


FIG 9.43 Tapped delay line

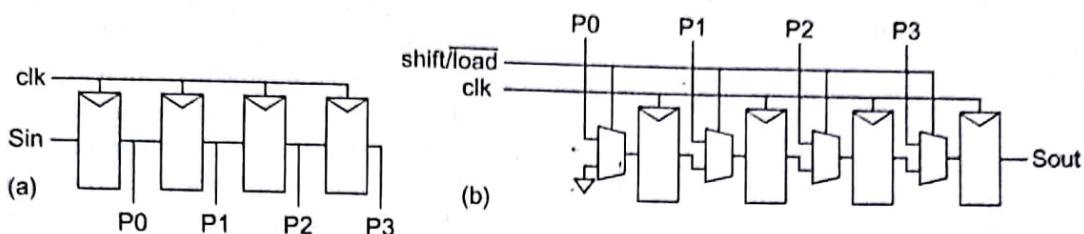


FIG 9.44 Serial/parallel memories

*FULL* flag when there is no room remaining to write data and the *EMPTY* flag when there is no data to read. Because of other system delays, some queues also provide *ALMOST-FULL* and *ALMOST-EMPTY* flags to communicate the impending state and halt write or read requests. The queue internally maintains read and write pointers indicating which data should be accessed next. As with a shift register, the pointers can be counters or 1-of-*N* hot registers.

*First In First Out* (FIFO) queues are commonly used to buffer data between two asynchronous streams. Like a shift register, the FIFO is organized as a circular buffer. On reset, the read and write pointers are both initialized to the first element and the FIFO is *EMPTY*. On a write, the write pointer advances to the next element. If it is about to catch the read pointer, the FIFO is *FULL*. On a read, the read pointer advances to the next element. If it catches the write pointer, the FIFO is *EMPTY* again.

*Last In First Out* (LIFO) queues, also known as *stacks*, are used in applications such as subroutine or interrupt stacks in microcontrollers. The LIFO uses a single pointer for both read and write. On reset, the pointer is initialized to the first element and the LIFO is *EMPTY*. On a write, the pointer is incremented. If it reaches the last element, the LIFO is *FULL*. On a read, the pointer is decremented. If it reaches the first element, the LIFO is *EMPTY* again.

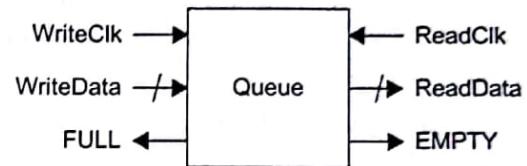


FIG 9.45 Queue

## 9.6 Content-addressable Memory

Figure 9.46 shows the symbol for a content-addressable memory (CAM) [Grosspietsch92, Schultz95, Miyatake01]. The CAM acts as an ordinary SRAM that can be read or written given adr and data, but also performs matching operations. Matching asserts a matchline output for each word of the CAM that contains a specified key.

A common application of CAMs is translation lookaside buffers (TLBs) in microprocessors supporting virtual memory. The virtual address is given as the key to the TLB CAM. If this address is in the CAM, the corresponding matchline is asserted. This

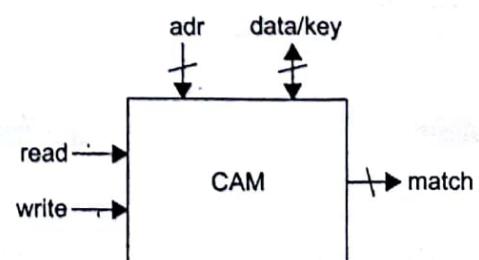


FIG 9.46 Content-addressable memory

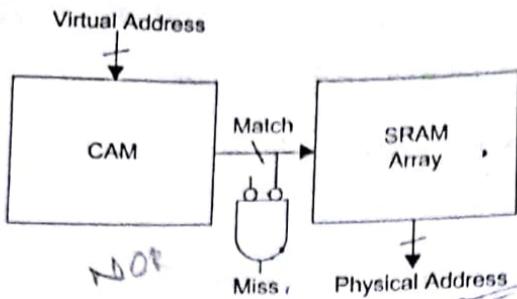


FIG 9.47 Translation Lookaside Buffer (TLB) using CAM

pseudo-nMOS gate. (The key is placed on the bitlines.) In Figure 9.48(a), if the key and the value stored in the cell differ, the matchline will be pulled down. (Only if all of the key bits match all of the bits stored in the word of memory will the matchline for that word remain high.) The key can contain a “don’t care” by setting both *bit* and *bit\_b* low. Figure 9.49 shows a layout of this cell in a  $56 \times 43 \lambda$  area; CAMs generally have about twice the area of SRAM cells. Figure 9.48(b) shows another CAM cell design with one fewer transistor (*N1* and *N2*) perform an XOR of the key and cell data. If the values disagree, *N3* is turned on to pull down the wordline. However, the gate of *N3* sees a degraded high logic level.

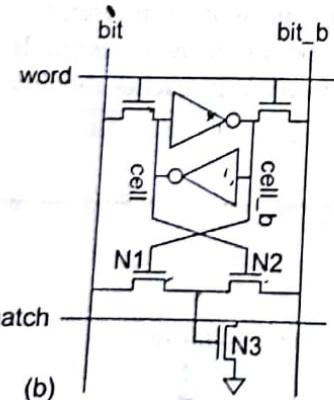
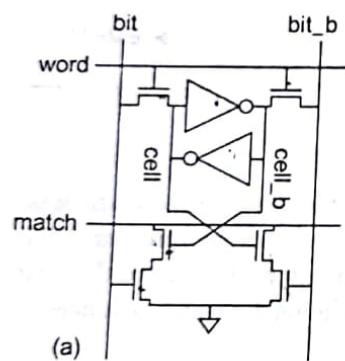


FIG 9.48 CAM cell implementations

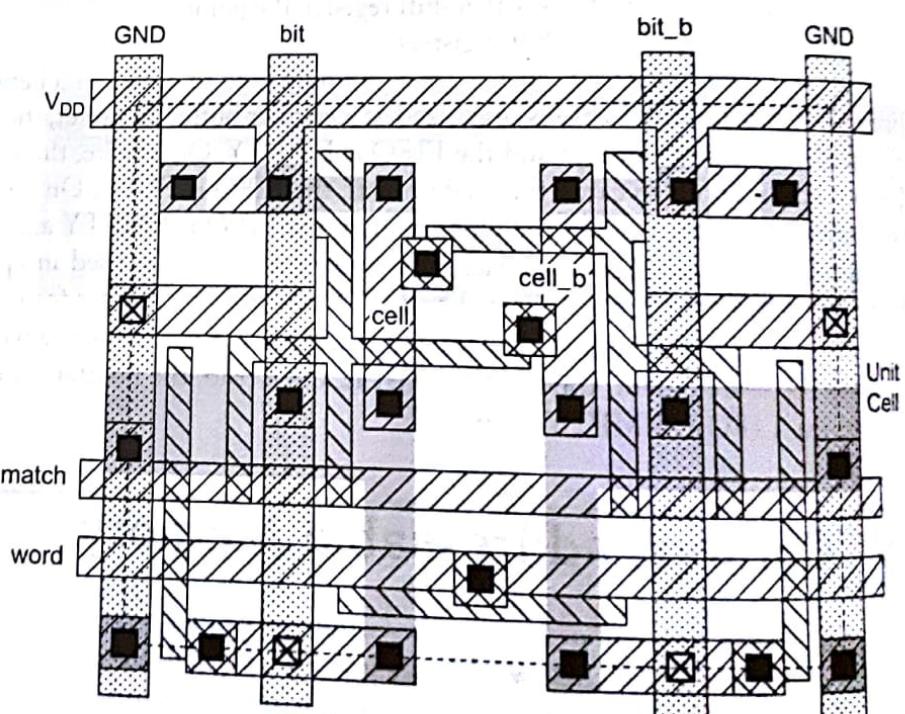


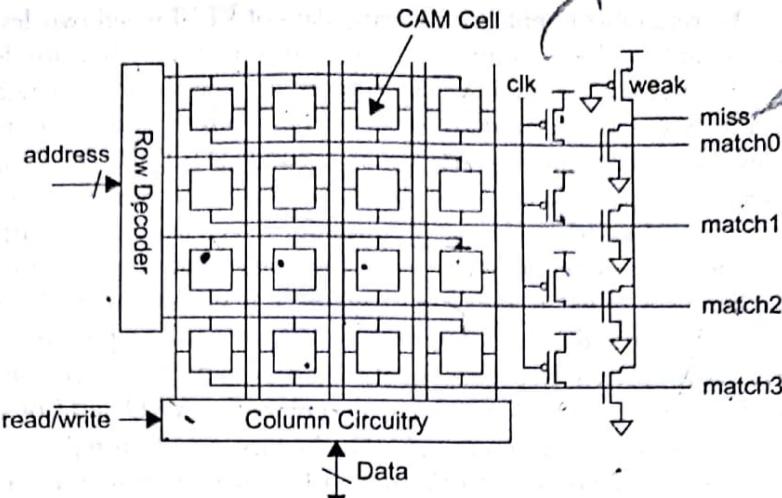
FIG 9.49 CAM cell layout. Color version given at the beginning of the book.

Figure 9.50 shows a complete  $4 \times 4$  CAM array. Like an SRAM, it consists of an array of cells, a decoder, and column circuitry. However, each row also produces a dynamic matchline. (The matchlines are precharged with the clocked pMOS transistors. The miss signal is produced with a distributed pseudo-nMOS NOR.)

matchline can serve as the wordline to access a RAM containing the associated physical address as shown in Figure 9.47. (A NOR gate processing all of the matchlines generates a *Miss* signal for the CAM.) Note that the *read*, *write*, and *adr* lines for updating the TLB entries are not drawn.

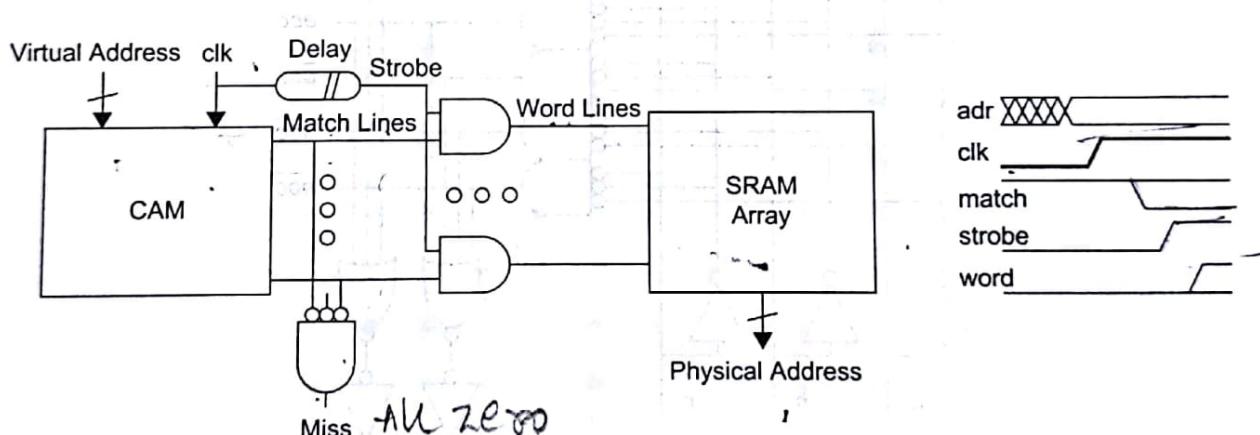
( $10\lambda$  and  $9\lambda$  implementations of the CAM cell are shown) in Figure 9.48. (The cells consist of a normal SRAM cell with additional transistors to perform the match.) Multiple CAM cells in the same word are tied to the same matchline. The matchline is either precharged or pulled high as a distributed

(Figure 9.48(a)) if the key and the value stored in the cell differ, the matchline will be pulled down. (Only if all of the key bits match all of the bits stored in the word of memory will the matchline for that word remain high.) The key can contain a “don’t care” by setting both *bit* and *bit\_b* low. Figure 9.49 shows a layout of this cell in a  $56 \times 43 \lambda$  area; CAMs generally have about twice the area of SRAM cells. Figure 9.48(b) shows another CAM cell design with one fewer transistor (*N1* and *N2*) perform an XOR of the key and cell data. If the values disagree, *N3* is turned on to pull down the wordline. However, the gate of *N3* sees a degraded high logic level.



**FIG 9.50** 4 × 4 CAM array

When the matchlines are used to access a RAM, the monotonicity problem must be considered. Initially, all the matchlines are high. During CAM operation, the lines pull down, leaving at most one line asserted to indicate which row contains the key. However, the RAM requires a monotonically rising wordline. Figure 9.51 refines Figure 9.47 with strobed AND gates driving the wordlines as early as possible after the matchlines have settled. The strobe can be timed with an inverter chain or replica delay line in much the same way that the sense amplifier clock for a SRAM was generated in Section 9.2.3. As usual, self-timing margin must be provided so the circuit operates correctly across all design corners.



**FIG 9.51** Refined TLB path with monotonic wordlines

Large CAMs can use many of the same techniques as large RAMs, including sense amplifiers and multiple subarrays. They tend to consume relatively large amounts of power because the bitlines and matchlines may all transition during a parallel search. [Miyatake01] describes a design of a mid-sized CAM using a pMOS match-line driver to reduce the swing of the matchlines and save power.

## 9.7 Programmable Logic Arrays

A programmable logic array (PLA) provides a regular structure for implementing combinational logic specified in sum-of-products canonical form. If outputs are fed back to inputs through registers, PLAs also can form