CHAPTER
9

Additional Topics in Memory Design

CHAPTER OUTLINE

9.1 Introduction

Memory is occupying a larger and larger percentage of the total area of digital chips. When memory is integrated on the same chip with logic, it is referred to as *embedded memory*. It has been forecast by the International Roadmap for Semiconductors (ITRS) that embedded memory will represent more than 50% of the design content of digital ICs by the year 2005. Many large microprocessor and graphics processing chips already contain over 50% memory, and in some cases it is well over 80%. An important reason for using large blocks of on-chip memory is to provide high data transfer rates between logic and memory. A sophisticated graphics processor chip, for instance, might require transfer of 128 bits of data every 5 ns, requirements that would be nearly impossible to meet with memory and logic in separate packages. Other requirements for embedded memory are found in cell phones and PDAs, where the need to simultaneously minimize physical size, power consumption, and

cost make it desirable to combine all logic and memory functions on a single chip. It is clear that memories, both embedded and stand-alone, will be a growing part of the integrated circuit market in the future.

Memory designs are differentiated by the cell structure, the use of static or dynamic storage techniques, access mechanisms, and whether or not the data storage is persistent even when the power is turned off. However, the most important metric for memory is the cost per bit. High-density and high-capacity memories are typically cheaper. There are a growing number of applications that require gigabytes of memory in hand-held and portable devices. Over the years, memory cells have evolved from six-transistor (6T) configurations to single-transistor (1T) configurations. The 6T cells are still the most popular in SRAMs. EEPROMs use 2T cells, and 1T cells are used by DRAMs, ROMs, EPROMs, Flash memories, and FRAMs.

The use of static versus dynamic techniques is primarily driven by the demand for high density. Static techniques are much more reliable and do not require refresh circuitry, but dynamic memories offer four times the density. On the other hand, static memories are much faster compared to dynamic memories, at the expense of additional power. When considering on-chip memories, static techniques are most frequently used due to the extra processing costs and complexity associated with dynamic memories. However, at some point, the density and cost-per-bit advantages of DRAM may outweigh many of the disadvantages.

Memories are also classified into the read/write and nonvolatile categories. Well-known members of read/write memories are obviously the SRAM and DRAM. Associative memories are a special type of read/write memory that use a keyword for the lookup process in place of address decoding. These are also called content-addressable memories (CAM). Serial memories such as FIFOs (first-in-first-out memories) and shift registers are also members of the read/write category. Nonvolatile memories retain their stored data even after the power supply is removed. Their primary role is to serve as read-only memories (ROM), although most have the ability to modify the stored data. Typically their write times are very long, but their read times are short and of the same order as other semiconductor memories. There are many ways to implement a nonvolatile memory cell. A well-known variety of this type of memory is the mask-programmable ROM. The other important members of this family are the EPROM, EEPROM, and Flash.

This chapter explores a variety of other semiconductor memories, their applications, access mechanisms, and cell configurations. Since SRAMs were described in Chapter 8, we will begin by examining CAMs, as they are a derivative of the SRAM architecture. We will also cover an important application of SRAM cells in the growing market segment of programmable logic called field-programmable gate arrays (FPGAs). Then, we will sequence through dynamic RAMs, mask-programmable ROMs, erasable programmable ROMs, electrically erasable ROMs, and Flash memories. The chapter concludes with a look at a memory cell based on ferroelectric materials called FRAMs.

*9.2 Content-Addressable Memories (CAMs)

The SRAMs discussed in Chapter 8 are used for data storage and lookup when the access mechanism is based on a known address. However, there are applications

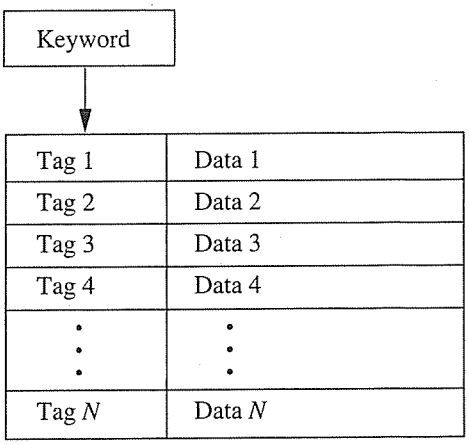


Figure 9.1
Associative memory lookup mechanism.

when the data that we seek is associated with a known binary keyword rather than a known binary address. The known keyword is compared against previously stored keywords, called *tags,* that reference the actual data that we seek. The tags are not stored in any particular order so we must match the address keyword with the tags that are already stored to access the desired data. This is illustrated in Figure 9.1. Memories that perform this type of function are referred to as content-addressable memories (CAMs). They are also called *associative memories* since we reference the data associated with a tag by matching the keyword to the stored tag.

CAMs use the basic architecture of the SRAM but allow the access of data through a matching scheme rather than an address decoding scheme. They are often used to build highly associative *cache* memories. The term *cache* is given to a local memory of a CPU that holds frequently used data by storing its address in one table and its associated block of data in another table. The addresses are not stored in any particular order, and only a small subset of the entire address space is represented in the cache. Instead of accessing data in the cache through a static address (direct mapped cache), we want to be able to access it through association with an address keyword. This requires a way of reading and writing the SRAM array based on different tags. The CAM stores the tag bits in a table that points to different parts of the SRAM array where the associated data is stored.

Another application of the CAM is to store an Internet routing table. Such a table is used to route network packets from a source to a destination. Given a destination keyword, the data portion of the table contains information about the next node (or next "hop" in Internet parlance) to send the packet in order for it to move closer to the destination. Since packets contain the Internet protocol (IP) address of the destination, it can be processed at each node in the network to extract the destination address. The destination is used as a keyword to retrieve the next hop information. The routing table can be implemented as a CAM by using the destination as a keyword to look up the next hop data in the SRAM array.

A conceptual diagram of a CAM array and the contents of each CAM cell are shown in Figure 9.2. The array contains n-bit stored tags that are to be compared to

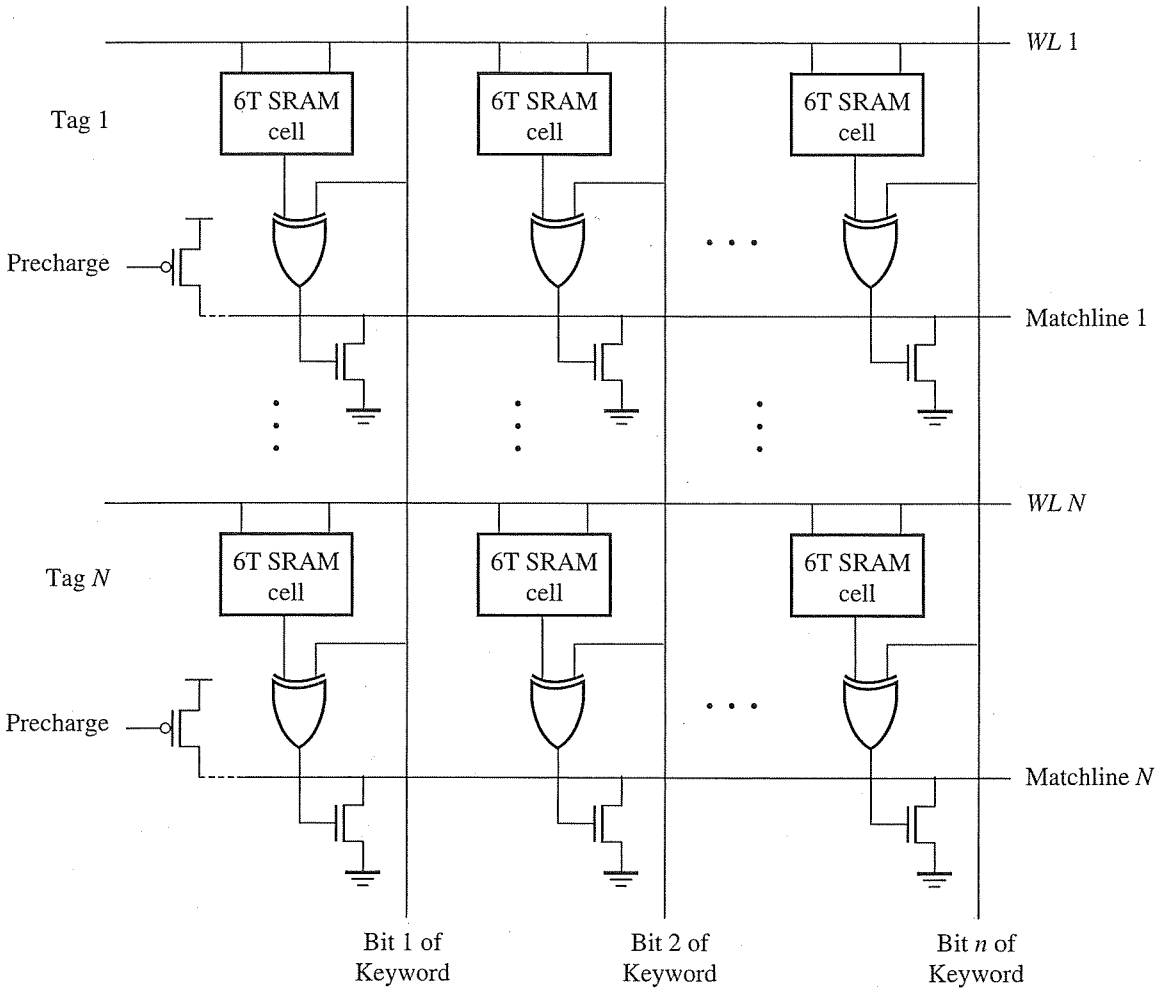


Figure 9.2
Arrangement of CAM lookup array.

the incoming n-bit keyword. Each row holds a different tag; each bit of the tag is stored in a separate 6T SRAM cell. The SRAM cell holds two values, the true value and its complement. The reference side holding the true value of the cell is fed to an XOR gate. Each of the N rows also has its own matchline to indicate that the incoming keyword matches the tag.

The operation is as follows. Initially all N matchlines are precharged high. In effect, all rows start by assuming that a match has occurred with the keyword. The XOR gate compares each keyword bit with the tag bit stored in each SRAM memory cell. If there is a match, the NMOS pull-down device does not turn on and the precharged matchline stays high. If the keyword and stored tag do not match, the NMOS device discharges the matchline. Note that all bits of the tag, and all tags are compared to the keyword in parallel. Therefore, any one of the pull-down transistors can discharge the matchline, since any single bit mismatch implies a complete tag mismatch. Therefore, the matchline will only stay high (signifying a match)

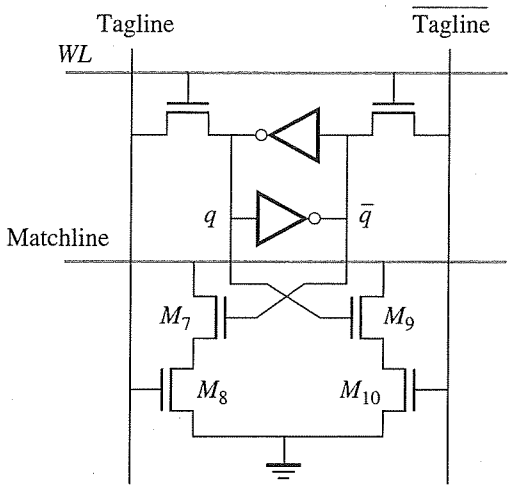


Figure 9.3
CAM cell schematic and layout.

when *all* tag bits match the keyword bits in a given row. Since all matchlines are initially precharged high, and all but one will go low, there is significant power dissipation. This is one of the limitations of large CAMs. Of course, there are cases when none of the contents of the CAM match the tag, or more than one row matches the tag. Actual CAM designs must handle a single match, multiple matches, and complete mismatches properly.

CAM cells may be implemented in many ways. One possible implementation is shown in Figure 9.3. The *WL* signal is routed in polysilicon while matchline is routed in metal 1. The two complementary bitlines, called *taglines* in this case, are routed in metal 2. The top part of the schematic is simply one SRAM cell, while the lower portion discharges the matchline if the cell contents do not match the values on the taglines. The functions of the XOR and pull-down transistor are combined and implemented using four transistors, M_7, M_8, M_9, and M_{10}. Due to these additional transistors, the layout of the CAM cell is 20–30% larger than the 6T SRAM cell. Other CAM cells can be designed to reduce area or increase speed depending on the application requirements.

Exercise 9.1

Explain how the CAM cell of Figure 9.3 works assuming that the stored tag bit is on the left side labeled *q* and the keyword bit comes in on the signal labeled *tagline*.

A block diagram of the CAM array (storing tags) alongside the SRAM array (holding data) is shown in Figure 9.4. The CAM array performs a parallel comparison of the incoming keyword with the stored tags. If a match occurs, it generates a matchline output that drives a wordline of the associated SRAM array. In a sense,

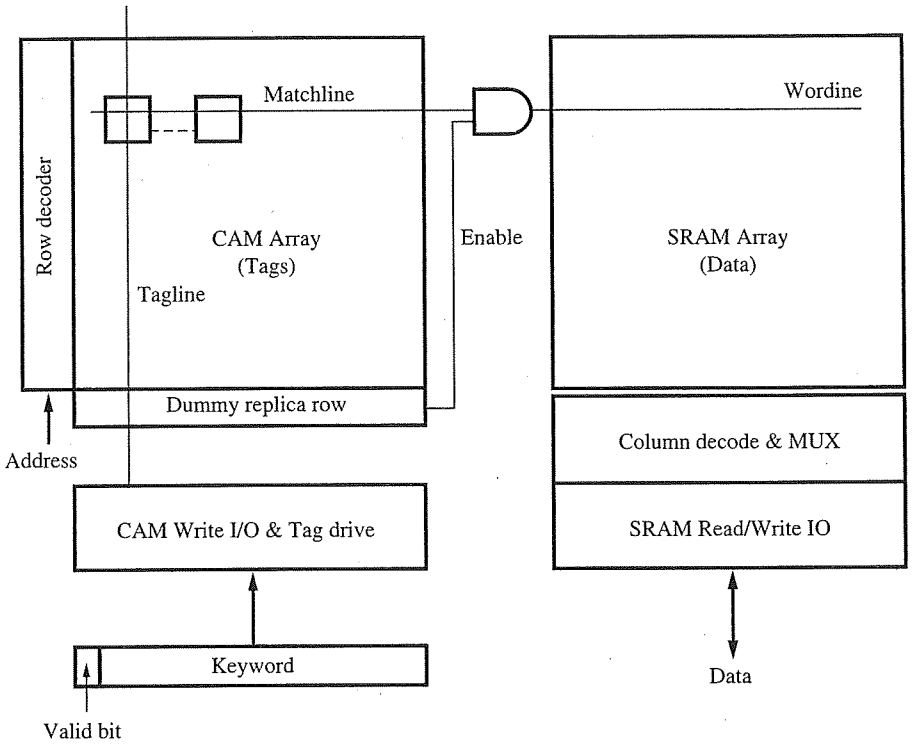


Figure 9.4
Block diagram of CAM and SRAM array.

the CAM plays the role of the decoder in a conventional SRAM. If the tag is 32 bits wide, each matchline would run horizontally through 32 CAM cells. Often, certain data in the SRAM is invalid so the corresponding tag is not valid. In order to indicate whether valid information exists, there is also a valid bit associated with each tag row. This must be matched along with the tag to activate the corresponding wordline. Additional address bits are used to select one or more of the columns out of the SRAM array, as defined by the application.

Each matchline must be ANDed with an enable signal that drives one of the wordlines high if there is a valid match. Timing of this enable signal is critical because all the matchlines start out high after precharge and only one of them will remain high at the end of a valid matching process. This means that the enable signal should not fire until the CAM has had enough time to resolve the worst case mismatch before selecting a wordline. In this case, the worst case corresponds to the condition that results in the longest time required to realize a mismatch. A replica circuit, similar to the one described in Chapter 8, is needed to generate this enable signal with the timing of a worst-case mismatch in the presence of process variations and changes in operating conditions.

A complete CAM design must handle unexpected cases, such as *multiple* matches. Of course, this can be avoided by checking if the tag already exists in the array before storing it. There can also be a complete mismatch with all the stored

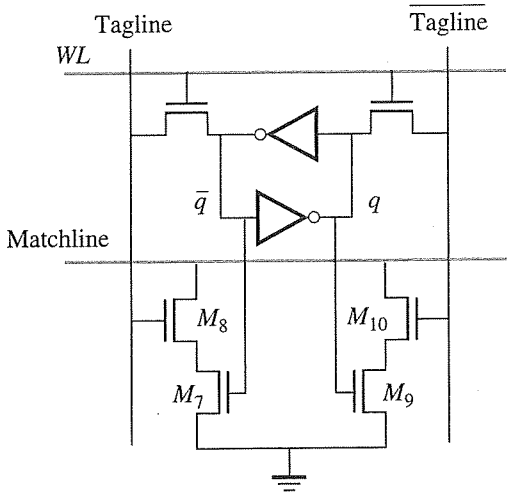


Figure 9.5
Reordering of XOR transistors for improved speed.

tags, in which case a *miss* has occurred. This requires an examination of all the matchlines to flag the case where none of them are high. This can be done with a large NOR gate. Upon detection of this case, the new tag and associated data can be written into the CAM and SRAM arrays, respectively.

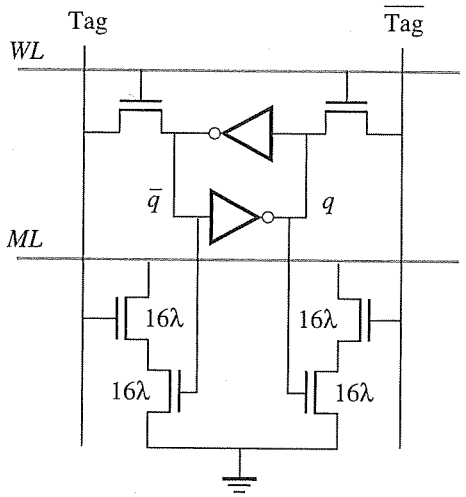
Since each CAM cell has an SRAM cell within it to store the tags, arrangements must be made to write to them. Therefore, the CAM array also has wordlines driven by the decoder that access each of the rows and bitlines to write the tag data into the cells. The bitlines for the 6T SRAM cell and the taglines for matching purposes use the same wires. The write process is similar to that for the SRAM array, but instead of pulling down a tagline, a tagline must be driven high. This is because both taglines must start off low to keep the internal XOR transistors off and enable the matchline to be precharged to a high value. This can be further understood by examining the cell shown earlier in Figure 9.3. If either M_8 or M_{10} are on, and the opposite side of the SRAM cell is high, then the matchline will inadvertently be pulled low. Therefore, the *taglines must start low* to avoid this.

The cell mismatch timing can be improved by reordering the transistors in the cell. In Chapter 6, it was pointed out that late arriving signals should be placed at the top of a series stack. We can use this fact to rearrange the transistors of the CAM cell as illustrated in Figure 9.5: The two column transistors, M_8 and M_{10}, are now connected to the matchline while the other transistors, M_7 and M_9, are connected to Gnd. Since *tagline* and $\overline{\textit{tagline}}$ can be considered as late arriving signals, they should be placed closer to the outputs. Of course, nodes internal to the cell are both stable and therefore may be viewed as early arriving signals. This new configuration allows M_8 or M_{10} to carry out most of the discharging process since M_7 or M_9 have already discharged the intermediate capacitance.

Example 9.1 **Delay of Match Operation**

Problem:

Using 0.18 μm technology parameters, compute the worst-case delay of a mismatch operation for a CAM array of Figure 9.4 that contains the CAM cell shown below. The array has 33 CAM cells in each row and 128 cells in each column. Draw the cells for one column and one row of the CAM cell. The taglines have a capacitance of 450 fF and the matchline has a capacitance of 200 fF due to the cells. The input drivers for the taglines have an input capacitance of 3 fF.



Solution:

The drawing of the row and column of the CAM should be similar to Figure 9.2. Using this figure, the total delay due to the taglines and the matchline is

$$\tau_{\text{total}} = \tau_{\text{tag}} + \tau_{\text{matchline}}$$

The delay along the taglines can be obtained using logical effort. The number of stages of drivers can be determined using FO4 rules:

$$N = \log_4\left(\frac{C_{\text{tag}}}{C_{\text{in}}}\right) = \log_4\left(\frac{450\text{ fF}}{3\text{ fF}}\right) = \log_4(150) = 3.6 \approx 4 \text{ stages}$$

The optimal stage effort (SE) for four stages is

$$SE^* = \sqrt[4]{150} = 3.5$$

The normalized delay is

$$D = 4(3.5) + 4(0.5) = 16$$

Note that the τ_{inv} value for 0.18 μm technology can be computed as

$$\tau_{\text{inv}} = 3C_gR_{eqn}L_n = 3(2\text{ fF}/\mu\text{m})(12.5\text{ k}\Omega)(0.2\ \mu\text{m}) = 15\text{ ps}$$

The actual delay is

$$\tau_{\text{tag}} = \tau_{\text{inv}} \times D = (15\text{ ps})(16) = 240\text{ ps}$$

The delay for the matchline to be pulled low in the worst case can be computed using an *RC* delay model for the CAM cell and matchline, respectively. The output capacitance is given as 200 fF. The CAM cell has one transistor that is always on in the pull-down stack since it is connected to the internal node of the SRAM cell. Therefore, the size of the upper transistor of the stack determines the resistance seen by the output capacitance. The on-resistance of the CAM cell is approximately

$$R_{eff} = 12.5\ \text{k}\Omega/8 \cong 1.6\ \text{k}\Omega$$

Therefore,

$$\tau_{\text{matchline}} = RC = (1.6\ \text{k}\Omega)(200\ \text{fF}) = 320\ \text{ps}$$

The total delay is

$$\tau_{\text{total}} = \tau_{\text{tag}} + \tau_{\text{matchline}} = 240\ \text{ps} + 320\ \text{ps} = 560\ \text{ps}$$

*9.3 Field-Programmable Gate Array

An important category of chips is called the *field-programmable gate array (FPGA)*. An FPGA is a stand-alone, programmable logic device that allows rapid implementation of complex logic systems. They can be broadly categorized into two groups: SRAM programmed and antifuse programmed. We focus on the SRAM-programmed FPGA in this section. This type of array uses SRAM bits to configure the chip to perform a desired function. FPGA customers purchase the stand-alone chips based on the size, speed, and power requirements of the target application. User customization of the chips is performed by setting the programmable SRAM bits in the FPGA to the desired values.

Traditionally, FPGAs were widely used for prototyping an ASIC before the chip was designed. Today, the densities and speeds of FPGAs are such that these devices are often found in finished products. There are also many other examples of its use, such as interface chips for automatic test equipment. It is a fast-growing segment of the semiconductor market, competing with both ASIC and SOC design styles. Recently, the use of on-chip or embedded programmable fabrics has been proposed, and there is much research underway in this area.

The basic architecture of a stand-alone FPGA is shown in Figure 9.6. This architecture is highly regular with a programmable logic block (PLB) replicated horizontally and vertically, and I/O pads placed along the periphery. Both the logic blocks and the I/O are programmable. The task of placing and routing the logic onto the FPGA is a highly automated process. Beginning with an RTL description of the system, usually represented in Verilog or VHDL, a synthesis and optimization step is used to generate the corresponding logic level representation. The logic is then placed and routed by FPGA CAD tools for a target FPGA chip. This process ultimately produces a set of programming bits that must be written into the SRAMs inside the FPGA to implement the final system.

The logic blocks and routing architecture are illustrated in Figure 9.7. This figure shows how the logic outputs are connected to the neighboring routing channels.

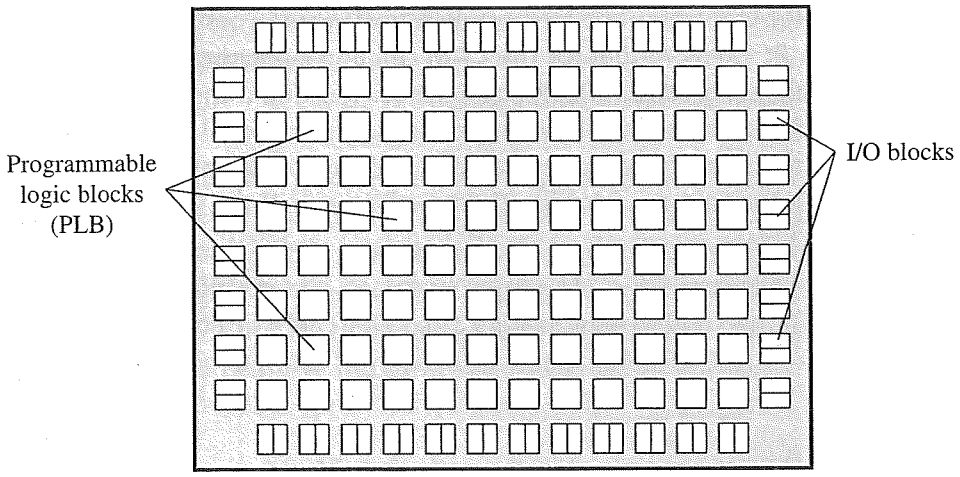


Figure 9.6
Overall architecture of FPGA chip.

The outputs of the block intersect the vertical or horizontal tracks at *connection blocks*. These connection blocks feed the vertical or horizontal tracks via programmable connections. The signals are routed along these tracks in one direction to the *switch block*, or *switch matrix*, at which point they can be routed in one of the other three directions. The design decisions at this level involve the number of vertical

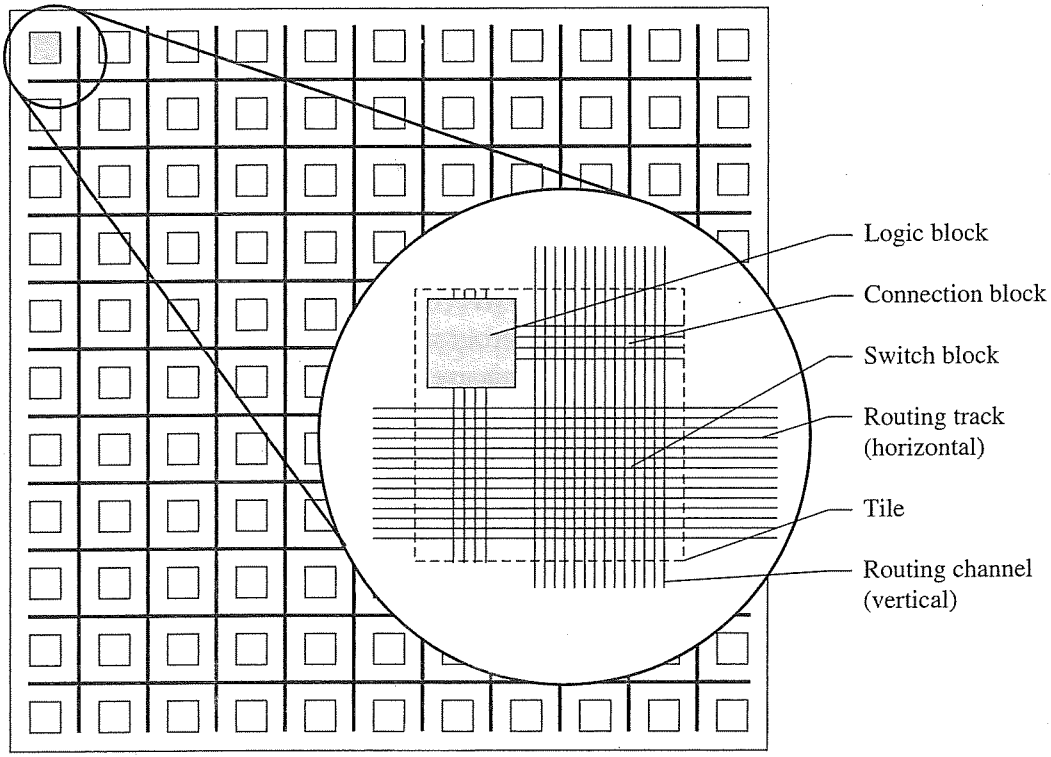


Figure 9.7
Island style FPGA architecture.

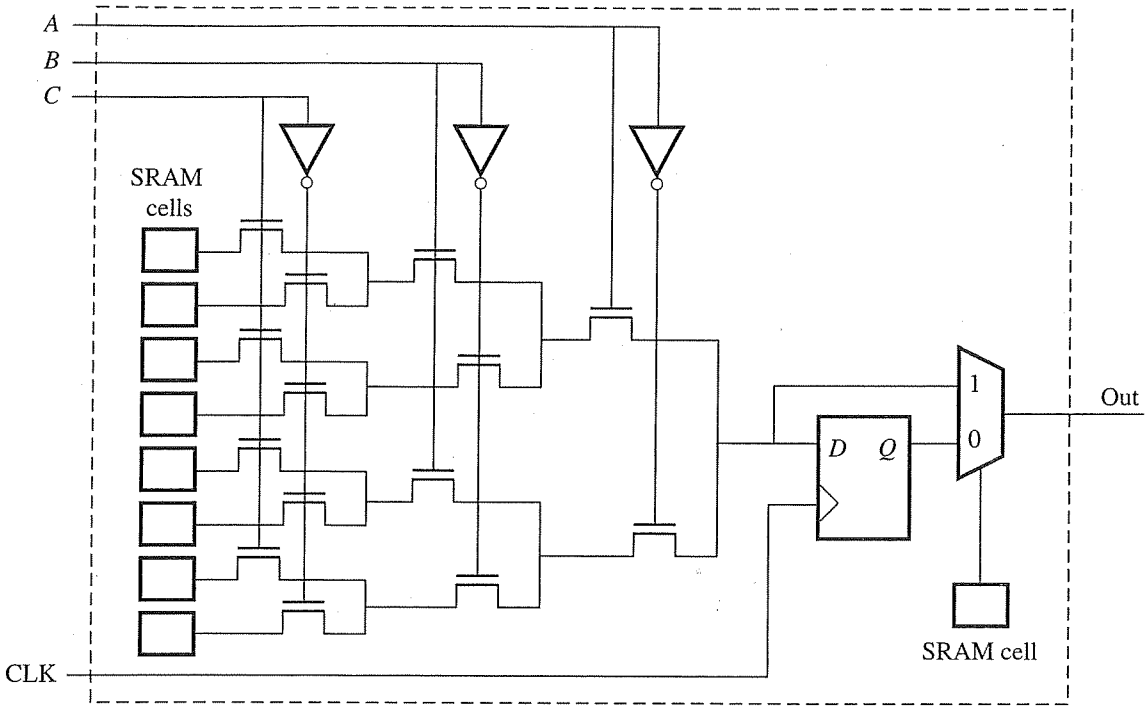


Figure 9.8
Basic programmable 3-LUT (lookup table).

and horizontal tracks in the connection block and the switch block, and the connection mechanisms between the wires. Once determined, a *tile* for the FPGA can be laid out and arrayed vertically and horizontally to form the core of the chip.

A simplified schematic of a PLB is shown in Figure 9.8. It consists of a number of SRAM cells, pass transistors, a D-flop, and a multiplexer. Actual blocks are more complicated, but this schematic is sufficient for our purposes of illustrating the basic concepts of the programmable block. The SRAM bits are programmable and serve two purposes: to store logic data and to select the configuration of the logic block.

The SRAM cells shown in the left portion of the diagram are used to store the truth table of the function being implemented. This particular implementation has three inputs and a total of eight cells for storage of the 2^3 possible combinations. This is referred to as a 3-input lookup table, or 3-LUT for short. Typically, 4-LUTs are implemented in FPGAs, but this may vary from manufacturer to manufacturer. User programming of an FPGA is performed by serially shifting desired values into the SRAM bits as shown in Figure 9.9. For this purpose, the SRAM bits need to be arranged as a shift register. There are a number of design considerations in the programming procedure, including power and overall programming time. For example, several serial chains can be used to minimize programming time, but this may increase the power dissipation.

Once the values are stored in the LUTs, they are accessed by applying the inputs *A*, *B*, and *C*. The input settings activate a single path through the pass transistor tree. In Figure 9.9, this tree can be viewed as an 8-to-1 multiplexer with the inputs acting

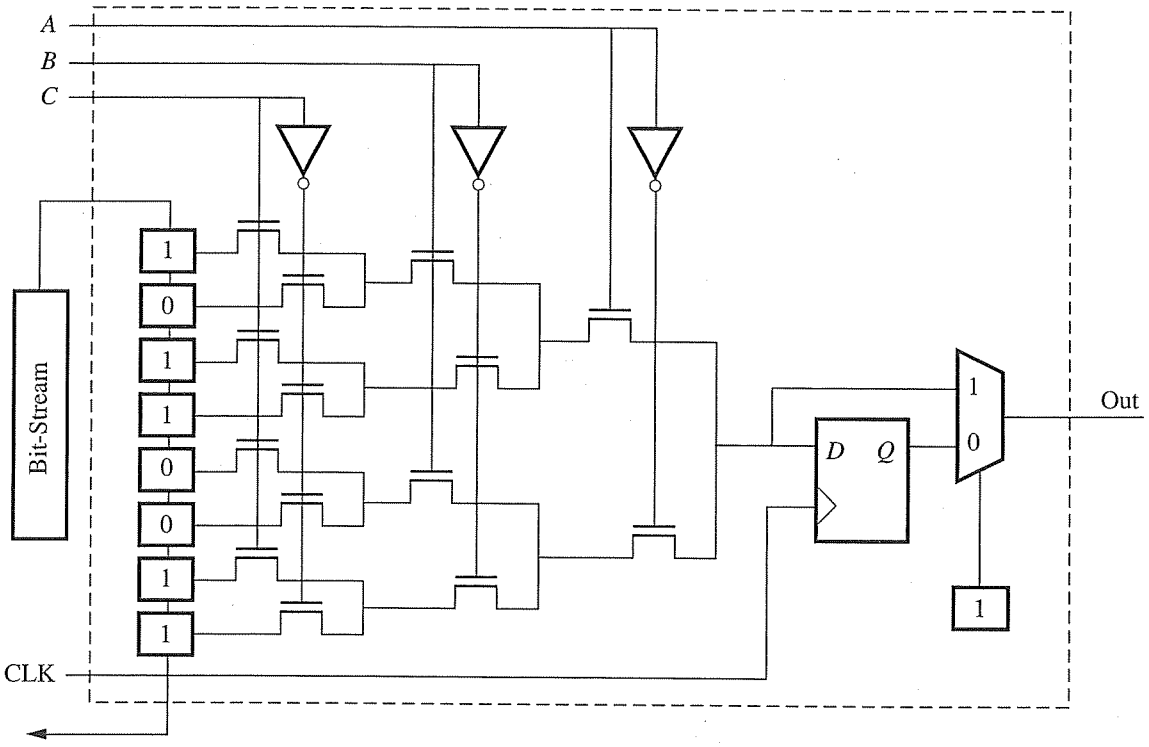


Figure 9.9
Programming the 3-LUT.

as the select variables. For example, the first stored value is accessed when $A = B = C = 1$. The second value is propagated to the output when $A = 0$, $B = C = 1$, and the eighth value is accessed when $A = B = C = 0$. The fact that these are n-channel pass transistors implies that they have a V_{TN} drop that degrades the output level. This can be overcome by boosting the gate drive above V_{DD} so that there is no reduction in the output voltage level along the pass transistor path.

On the right side of the figure are the elements that determine whether the PLB is sequential or combinational. The SRAM bit setting for the multiplexer determines whether the output is taken directly from the pass transistor tree or from the Q output of the D-flop. Choosing the pass transistor tree output produces a combinational circuit, while choosing the D-flop output creates a sequential circuit. The flop is edge-triggered and controlled by an additional CLK input to the PLB. Note that in FPGAs, the clock must be symmetrically routed to all PLBs since it is a required input to all flops. Of course, modern FPGAs have many additional clocks that must be routed to other types of blocks.

Example 9.2 **FPGA Logic Blocks**

Problem:

In Figure 9.9, what function is implemented in the PLB? Is the function combinational or sequential?

Solution:

Using a Karnaugh map, $Out = \overline{B} + AC$. This is a combinational function since the output of the D-flop is not gated through the multiplexer due to the "1" setting of the SRAM bit.

Connections between logic blocks are formed using prefabricated routing tracks. These routing tracks are connected to each other, and to the logic blocks, via pass transistors, each controlled by an SRAM bit. This is illustrated in Figure 9.10. Programmable connections are provided at the cross points of vertical and horizontal tracks using pass transistors. The transistors that have a stored "1" in their

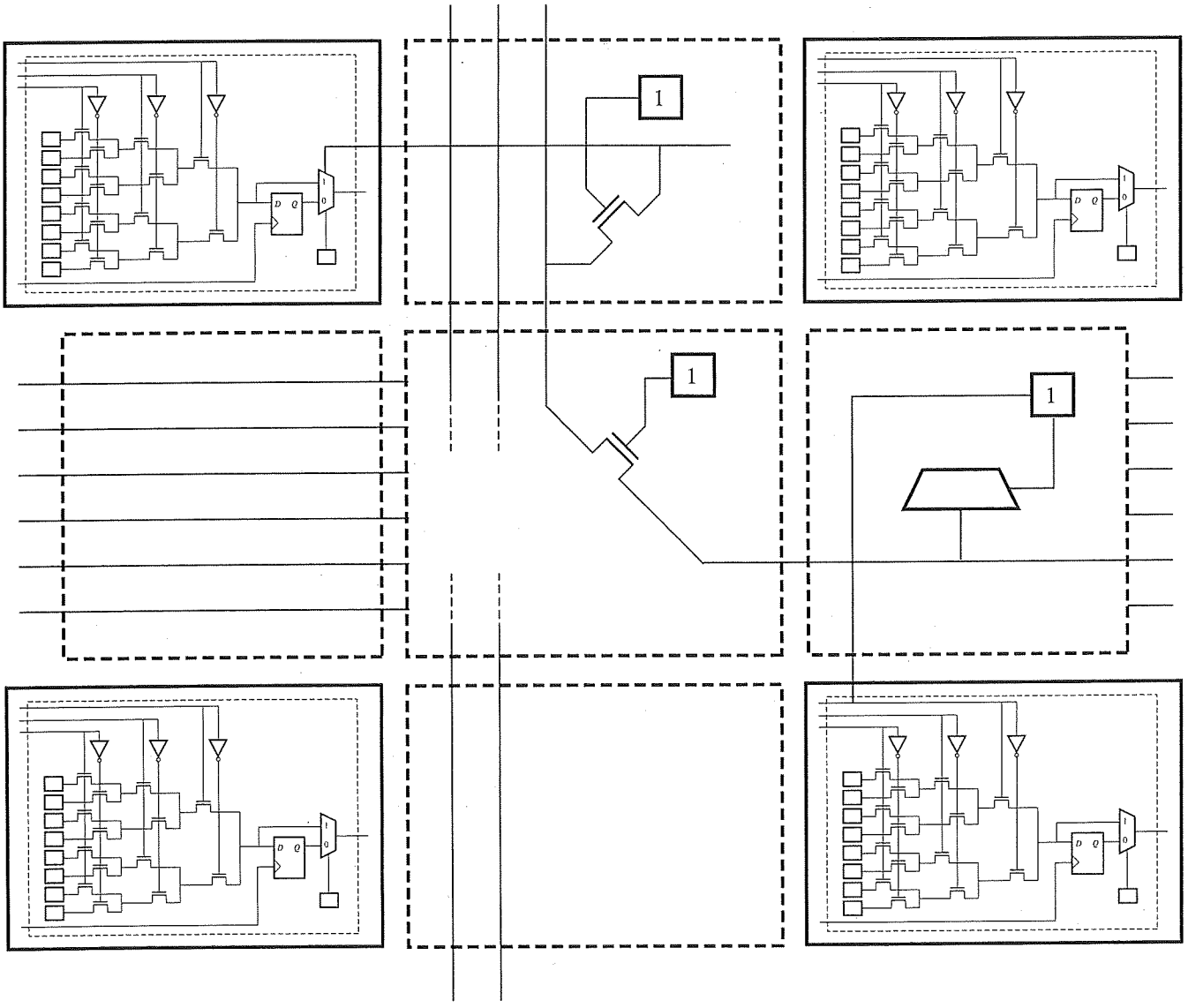


Figure 9.10
Programmable connections.

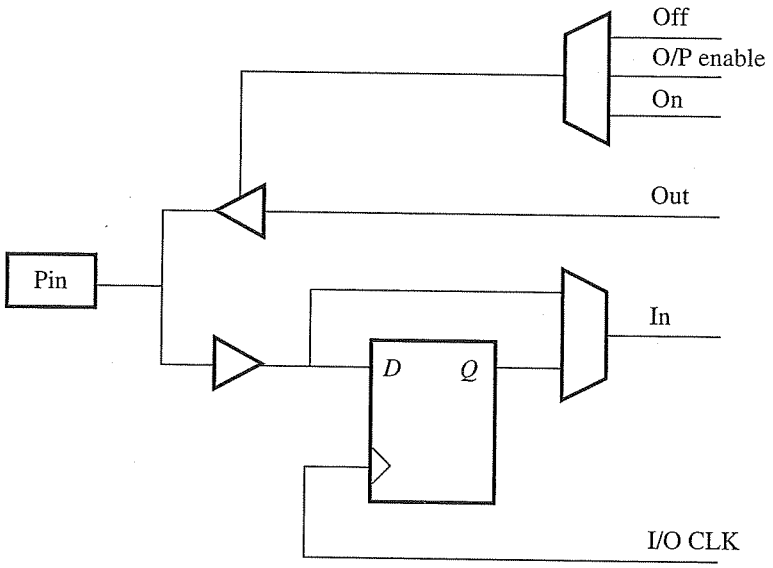


Figure 9.11
I/O buffer for FPGA.

SRAM bit are enabled while those with a stored "0" are disabled (transistor sizes are exaggerated in the figure). The role of the connection block transistors is to gate the signal from the source PLB to the routing tracks. The switch block transistors route the signal from the vertical tracks to a horizontal track, or vice versa. They can also connect one incoming horizontal track to an outgoing horizontal track and an incoming vertical track to an outgoing vertical track. Finally, a multiplexer selects the desired signal and routes it to the input of the destination PLB.

The I/O for an FPGA chip is also configurable, as illustrated in Figure 9.11. In general, the output is bidirectional but can be programmed to be either input-only or output-only with a programming bit (not shown) on the multiplexer. If it is transmitting data, the large output buffer is enabled and the value, Out, is propagated to the pin. The output can be set to high-impedance by turning off this buffer. The *O/P Enable* signal allows the state of the output buffer to be under control of on-chip logic. If the chip is receiving data from the pin, the value can be registered in the D-flop or sent directly into the chip, depending on the programming of the multiplexer. Today, FPGA I/O blocks can be configured to support many different I/O standards.

FPGAs are likely to become a dominant technology for system implementation in the future. However, there are a few limitations of these flexible devices. First, the performance tends to be lower than custom ICs or ASIC designs. Power dissipation is also much higher. This is true during normal operation, and during programming. Since the program bits are SRAM cells, there are concerns about the effect of α-particle induced single-event upset (SEU) as the internal capacitances continue to reduce in value due to scaling. However, these issues will be resolved over time.

Another limitation is associated with the size of a system that can be implemented on an FPGA. Typically an RTL description is automatically converted to a

gate-level description, and then placed and routed by CAD software for a target FGPA. Depending on the type of system being implemented, the LUTs may not be fully utilized, or the desired logic may not fit within a single LUT and must span several LUTs. This creates signal routing and congestion problems due to the limitations of the number of available tracks. While the reported gate counts of these devices are 5M or higher, the effective utilization of FPGAs may only be 10–20%. In addition, higher capacity FPGAs tend to be more expensive on a per chip basis, so they are not suitable for high-volume applications. Nevertheless, the market segment of FPGAs is expected to grow over the next few years.

In the near future, the programmable fabric associated with FPGAs will begin to appear on ASIC/SOC designs to permit programmability after fabrication. There are situations where an industry standard is evolving, or a block will be tailored to the end customer requirements, or perhaps the specifications for a block not fully defined at design time. In these cases, an FPGA-like programmable fabric can be embedded on the same chip along with processors, memory, and other logic blocks. The desired function can be programmed later using CAD software similar to that for stand-alone FPGAs. This is another reason why the interest in FPGAs is continuing to grow, and expected to continue for the foreseeable future.

9.4 Dynamic Read-Write Memories

The importance of reducing the cost per bit of memory led to simpler, smaller-area memory cells that could be more densely packed on a chip. The static memory cells described in Chapter 8 all require four to six transistors per cell and four or five lines connecting to each cell. In the late 1960s, it was realized that memory cells with reduced complexity, area, and power consumption could be designed if dynamic MOS concepts, similar to those introduced in Chapter 7, were used. Static memory cells store data as a stable state of a flip-flop, and data is retained as long as dc power is supplied. In contrast, dynamic cells store binary data as charge on a capacitance. Normal leakage currents can remove stored charge in a few milliseconds, so dynamic memories require periodic restoration, or *refreshing*, of stored charge, typically every millisecond or so. For memories of 64K bytes and larger, the cost of a complete dynamic memory system including provision for refresh cycles is lower than the cost of a system based on static memory components. However, the complexity of the peripheral circuitry, refresh circuitry, and timing leads to slower devices than their static counterparts.

Our study of dynamic RAMs (DRAMs) begins with the basic static RAM cell with the PMOS devices removed, as shown in Figure 9.12. At first glance, this configuration may not appear to work as a memory cell since the pull-ups have been removed. However, the access transistors, M_3 and M_4, act as pull-ups when they are turned on. Since the bitlines are precharged high in the previous cycle, they will act to keep one side high while the other side is low. Consider the case where the left side of the cell is high and the right side is low. When the access transistors are turned off, the high value is stored on the capacitor at node q. This high value keeps M_2 turned on which maintains a low output at node $\overline{q}$. This keeps M_1 turned off. When the wordline is raised, the two access transistors turn on. Charge sharing

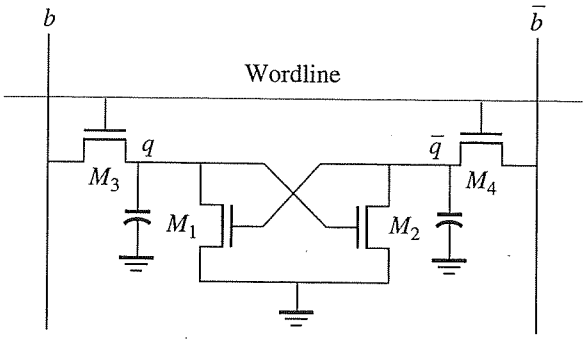


Figure 9.12
Static RAM cell with pull-ups removed.

occurs between b and q to keep node q high, while $\bar{b}$ discharges through M_4 and M_2. Since this cell relies on charge storage to maintain the value of a node, it is a dynamic cell. If we want to minimize the cell size of this dynamic memory, we can do substantially better.

9.4.1 Three-Transistor Dynamic Cell

The first widely used dynamic memory cell is shown schematically in Figure 9.13. Note that this three-transistor (3T) cell can be derived by eliminating M_2 from the NMOS cell shown in Figure 9.12. The read and write ports have been separated out and two wordlines are used in the cell. Contrary to static cell design, this cell does not require internal device ratios for proper operation. Transistors M_1, M_3, and M_4 can all be small devices to minimize cell area. Parasitic node capacitance C_1 is drawn in explicitly because it is essential to the operation. Charge stored on C_1 represents stored binary data. Selection lines for reading and writing must be separated

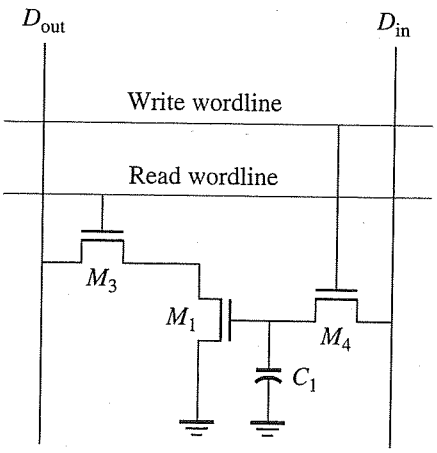


Figure 9.13
Three-transistor (3T) DRAM cell.

because the stored charge on C_1 would be lost if M_4 turned on during reading. Although separate column lines are shown here for data in (D_{in}) and data out (D_{out}), these two may be combined at the expense of some extra complexity in the read-write circuits.

The cell operates in two-phase cycles. The first phase of each read or write cycle is devoted to a precharge phase during which columns D_{in} and D_{out} are charged to a valid high logic level via column pull-up transistors. A "1" is assumed to correspond to a high level stored on C_1 and is written by turning on M_4 after D_{in} is high. The D_{in} line is highly capacitive because it is attached to many cells. The column I/O circuits do not need to hold D_{in} high because sharing the charge on D_{in} with C_1 does not significantly reduce the precharged high level. A "0" is written by turning on M_4 after the precharge phase is over, then simultaneously discharging D_{in} and C_1 via a grounded-source pull-down device (not shown) in the column I/O.

Reading is accomplished by turning on M_3 after the precharge is over. If 1 is stored, D_{out} will be discharged through M_1 and M_3. If 0 is stored, there will be no conducting path through M_1, so the precharged high level on D_{out} will not change significantly. The cell may be read repeatedly without disturbing the charge stored on C_1. Drain junction leakage and subthreshold current of M_4 reduce the stored charge over the span of milliseconds. Refreshing is performed by reading stored data before charge all leaks away, inverting the result, and writing back into the same location. This is performed simultaneously for all the bits in a row once every millisecond.

The level on the D_{out} line in principle can be detected with a simple inverter, but considerable delay would be encountered in achieving the needed 1- or 2-V swing on the D_{out} line. If a short access time is desired, a *current sensing* amplifier may be used. This amplifier would detect the flow of current into the cell and produce 1 at the output. If current did not flow into the cell, the output of the amplifier would be 0. For this 3T cell, output data is inverted compared to input data. However, memory component data input and output will have the same logic polarity if one extra inversion is included in either the read or write data path.

9.4.2 One-Transistor Dynamic Cell

Most modern dynamic RAMs have capacities of 64M bits or more. All use the one-transistor (1T) cell with a storage capacitor shown schematically in Figure 9.14. It can be derived from Figure 9.13 by removing the read access transistors, M_1 and M_3. There are many variations in the detailed realization of this cell, depending on the number of polysilicon layers, method of capacitor formation, conductors used for row and column, etc. For simplicity in this introductory evaluation we overlook the many differences and focus on the schematic representation that is common to all.

Selection for reading or writing is accomplished by turning on M_1 with the single row line. Data are stored as a high or low level on C_1. In the interests of minimum cell area, C_1 should be kept fairly small. However, as voltage levels drop, the total charge stored on this capacitor begins to diminish. Therefore, it is desirable to make C_1 large without compromising the cell size.

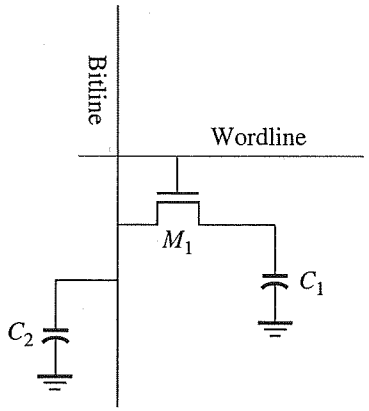


Figure 9.14
1T DRAM cell.

Data are written into the cells by forcing a high or low level on the selected column when the cell is selected. The internal node voltage should be as high as possible to further increase the stored charge. Rather than limiting the internal voltage to $V_{DD} - V_{TN}$, the wordline can be boosted above $V_{DD} + V_{TN}$ using bootstrapping or charge-pump circuits so that the internal node can rise to V_{DD}. The boosted wordline is used for both reading and writing.

When reading the cell, the charge stored on C_1 is shared with the capacitance C_2 of the column line which is typically 10 times larger. If the bitline capacitance is too large, the results of the charge sharing will not be significant enough to detect the stored value in the cell. A large value of C_1 would mitigate this problem. However, the requirement for a large C_1 is in conflict with the cell size requirements for high density. Two approaches have emerged to address this issue, as shown in Figure 9.15. On the left is the *trench capacitor* cross section where a vertical cut is made in the substrate and filled with an insulator-poly sandwich forming the capacitor. On the

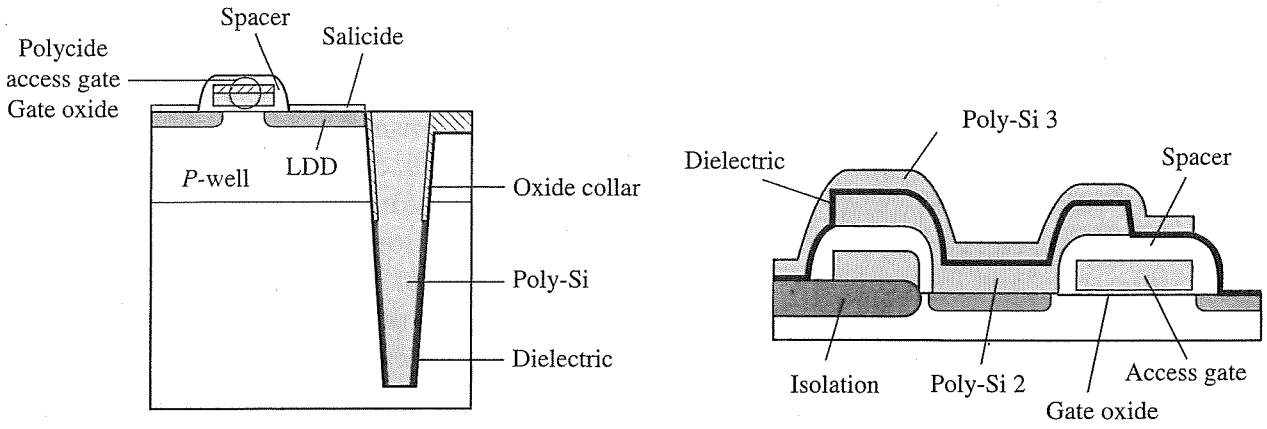


Figure 9.15
DRAM trench and stack capacitors.

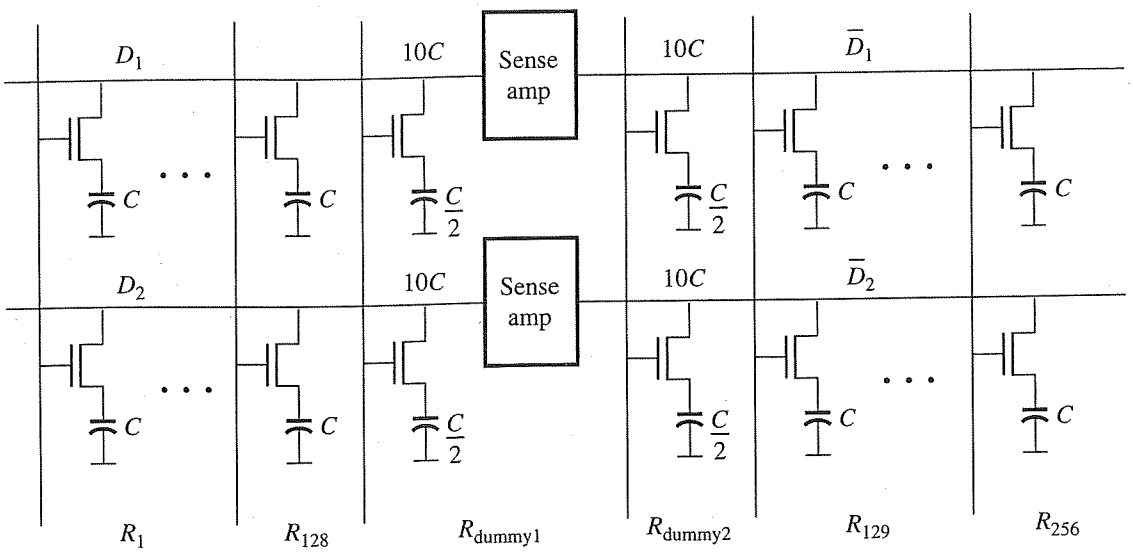


Figure 9.16
1T DRAM array configuration.

right is the *stacked capacitor* where two additional poly layers are deposited above the transistor with a dielectric material sandwiched between them. The stacked capacitor is now the preferred method to realize the cell capacitance. Both of these approaches yield capacitance values between 20–30 fF. This implies that the bitline capacitance should lie in the range of 200–300 fF for proper read operations.

During the read, the internal value of the cell is disturbed. The read process is referred to as a *destructive read-out* since the value in the cell is modified due to charge sharing. As a result, stored data must be regenerated every time they are read, in addition to refreshing them periodically even if they are not read. Read amplifier design for reliable detection of the small column signal is one of the most difficult aspects of 1T DRAM design.

The simplified schematic for a read-refresh circuit for a 1T DRAM is shown in Figure 9.16. Since the detection of a small voltage change in a single-ended configuration is difficult, the bitlines of the storage array are split in half so that equal capacitances are connected to each side of the flip-flop. This is called an open-bit architecture. The regenerative switching of a dynamic flip-flop (represented as a sense amplifier) detects the small data signal and restores the high or low signal level. As shown in Figure 9.16, each half column in the array has a single additional dummy cell that will be used as described below.

Reading from a 1T array proceeds as follows. A precharge signal sets the voltages on all column lines to $V_{DD}/2$ and sets the internal voltages in all dummy cells to $V_{DD}/2$ by turning on both R_{dummy} lines. Next, assume the wordline R_1 is pulled high so that the first set of cells on the left is selected. The dummy cells on the opposite side of the sense amplifier are selected simultaneously by raising the signal R_{dummy2}. The column voltage $\overline{D}$ on the right side connected to the selected dummy remains at $V_{DD}/2$ since the internal and bitline voltages are equal. However, the D voltage on the left side will either increase or decrease in value depending on

whether a 1 or 0 is stored in the cell. The amount of voltage change can be computed using the appropriate charge sharing equation. It is important to select cells on *both sides* of the sense amplifiers to balance the common-mode noise due to feedthrough and other effects. This is the reason why dummy cells are always selected on the opposite side of the array.

The resulting small differential voltage difference between the two sides of the array determines the final state of the flip-flop when a latching signal is applied. For example, if D is higher than $\overline{D}$, then the sense amplifier will interpret the difference as a stored 1. If D is lower than $\overline{D}$, then the sense amplifier will interpret the difference as a stored 0. The resulting data are taken out through a column decoder circuit to a final amplifier and output buffer.

Example 9.3 **Bitline Voltages for "1" and "0" in DRAM**

Problem:

In Figure 9.16, assume that the column lines are precharged to $V_{DD}/2$ and the internal dummy cell voltages are also set to $V_{DD}/2$. Compute the voltages on the bitlines when reading a "1" and reading a "0." Assume that boosted wordlines are used so that full V_{DD} levels can be stored in each cell when writing a "1." The column capacitance is 10C and the cell capacitance is C. Also compute the reference voltage level and then explain why it is required.

Solution:

The output voltage is due to charge sharing. For a stored "1," we can write

$$\frac{V_{DD}}{2} C_{\text{column}} + V_{DD} C_{\text{cell}} = V^*(C_{\text{column}} + C_{\text{cell}})$$

$$\therefore V^* = \frac{\frac{V_{DD}}{2} C_{\text{column}} + V_{DD} C_{\text{cell}}}{C_{\text{column}} + C_{\text{cell}}} = \frac{\frac{V_{DD}}{2} 10C + V_{DD} C}{10C + C} = 0.55\, V_{DD}$$

On the other hand, if a "0" is stored, the output on D is

$$\frac{V_{DD}}{2} C_{\text{column}} + 0 \times C_{\text{cell}} = V^*(C_{\text{column}} + C_{\text{cell}})$$

$$\therefore V^* = \frac{\frac{V_{DD}}{2} C_{\text{column}}}{C_{\text{column}} + C_{\text{cell}}} = \frac{\frac{V_{DD}}{2} 10C}{10C + C} = 0.45\, V_{DD}$$

The reference voltage sits at 0.5 V_{DD}. For a 1.8 V supply, there is approximately 90–100 mV of change relative to the reference level. This difference is converted to a full-swing value by the regenerative sense amplifier. The dummy cells are used to balance feedthrough and noise effects on both sides of the differential amplifier.

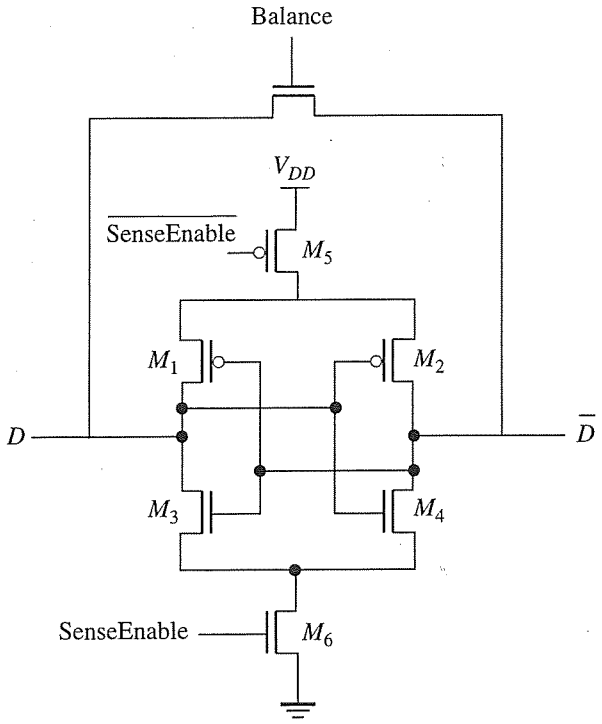


Figure 9.17
Simplified diagram of flip-flop style sense-refresh circuit.

The sense amplifier design is somewhat complicated but can be viewed as a latched-based clocked sense circuit as discussed in Chapter 8. A simplified version of the circuit is shown in Figure 9.17. Initially, a balance transistor is turned on to equalize the voltage on the two sides of the flip-flop, D and $\overline{D}$, to $V_{DD}/2$. The *SenseEnable* signal is held low until the difference voltage has been established on these lines with the activation of a regular wordline and a dummy wordline. Next, the pull-up transistor, M_5, and pull-down transistor, M_6, are turned on using the SenseEnable signal. The flip-flop then uses its regenerative property to establish true high and low values. These values are written back into the cell to complete the DRAM operation.

9.4.3 External Characteristics of Dynamic RAMs

The internal circuit design and timing diagram for widely used dynamic RAMs are much more complicated than described above, involving, for instance, a total of more than 20 internally generated clock phases for read, write, and refresh functions. These matters are beyond the scope of this book. To complete our study of dynamic RAMs, we provide a user's point of view and consider the terminal characteristics of these components.

The block diagram for a 64K-bit dynamic RAM is shown in Figure 9.18. The capacity of a single block can be increased by a factor of 4 to 16 without any serious penalty in operating speed. However, larger storage capacity on a single chip is best

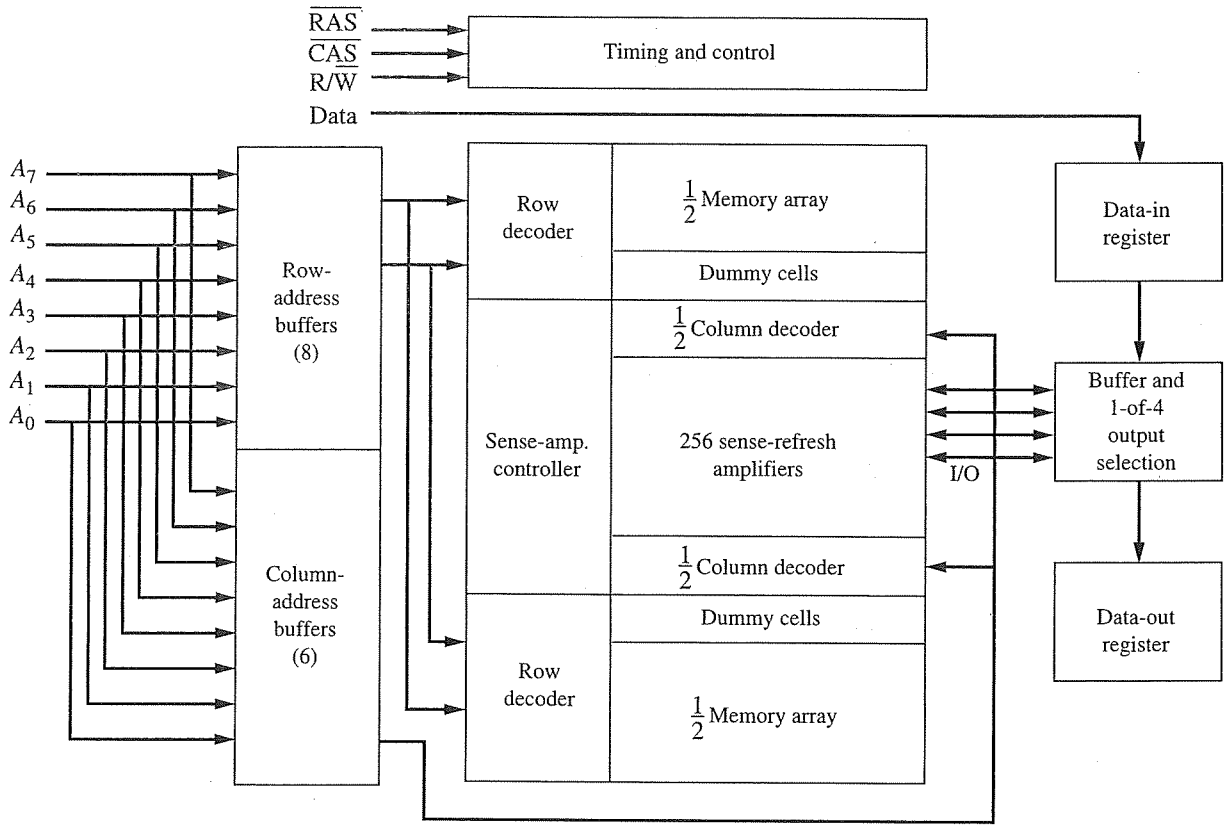


Figure 9.18
Block diagram of 64-kbit DRAM.

realized by replicating blocks like these many times over. Although internal process and circuit details vary, the block shown here is similar to those found in designs from many vendors. The 1T cells are organized in two halves as described above.

A read cycle would proceed as follows, with specific reference to the internal design of this particular diagram. First, row selection is achieved using eight row address bits that are latched into eight row address buffers. These buffers provide true and inverted outputs for each input bit. Seven address bits are routed to either the upper or lower set of row decoders, as determined by the state of the eighth row address bit. The upper or lower group of row decoders performs a 1 out of 128 selection, driving a single row line to a high level.

The high or low level stored in each of 256 cells along the selected row line is transferred to the column line, followed by a charge sharing event to establish a new voltage level. One flip-flop *sense amplifier* for each of the 256 columns is used to capture the relatively small change in column voltage and regeneratively restore it to a full 1 or 0 voltage level. The correct levels are restored to all selected cells without any involvement by column address selection circuits.

Address information for column selection comes in via the same eight pins used for row addresses. This is accomplished by a time-division multiplexing of these pins, as follows. After the row addresses are latched, they are disconnected from the address pins. Address changes at the pins do not affect row selection for

the remainder of the cycle. Therefore the pins are used for column selection. The purpose of address multiplexing is to reduce the package pin count, size, and cost. The column address is latched into eight column address buffers. Six of the column address bits are used to select one of 64 groups of four sense amplifiers for connection to four data buffers and a 4-to-1 multiplexer (using the last two address bits). A single selected bit is passed to the data output buffer and to an off-chip data bus.

A write cycle proceeds with many similarities, except that a single data input bit is used to set the state of one sense amplifier flip-flop, overriding the effect of the data previously present in the selected cell. An externally applied active low write enable signal, $\overline{W}$, specifies the write operation. Regardless of any other activity, every storage cell must be *refreshed* at least once every few milliseconds. This can be achieved by ensuring that every row in each half of the array is accessed at least this often. During each row access, the sense amplifiers perform their function of regenerating stored signal levels. The digital timing and control functions needed to perform the refresh functions can be provided by additional hardware on the memory board or by additional software executed by the central processing unit. Most systems (but not all) can be designed to allow the necessary refresh cycles without serious effects.

Dynamic RAMs derive the signals needed to perform these dynamic functions from two externally generated timing signals, known as the *row address strobe (RAS)* and *column address strobe (CAS)*. The complements of these signals, known as $\overline{RAS}$ (RAS bar) and $\overline{CAS}$ (CAS bar), are actually applied to the component pins. Complete specification of the read cycle timing relationships involves many timing parameters, all of which must be held within specified limits to ensure proper operation. Write cycle timing is similarly complex. Data sheets and application notes are the best source of detailed information for any particular component.

9.5 Read-Only Memories

Since we have reduced the memory cell contents to a single transistor in the DRAM, it is appropriate to examine other types of memory that contain only one cell transistor. These are collectively referred to as *nonvolatile* memory because they store values in the cells permanently, or at least semipermanently. The first memory in this category is the read-only memory or ROM. This type of memory is used to store constants, control information, and program instructions (*firmware*) in digital systems. ROMs may be thought of as components that provide a fixed, specified binary output for every binary input. In principle, the values are considered to be permanent and therefore do not expect to be changed after they are initially specified. In this section, we will study the internal circuits in the basic ROMs and then follow up with a look at other types of memories that can be reprogrammed to store new data.

9.5.1 MOS ROM Cell Arrays

The ROM architecture is constructed from the intersection of a set of wordlines and bitlines. A bit is stored in a ROM by the presence or absence of a transistor at the cross point. The absence of a path is achieved simply by having no circuit element

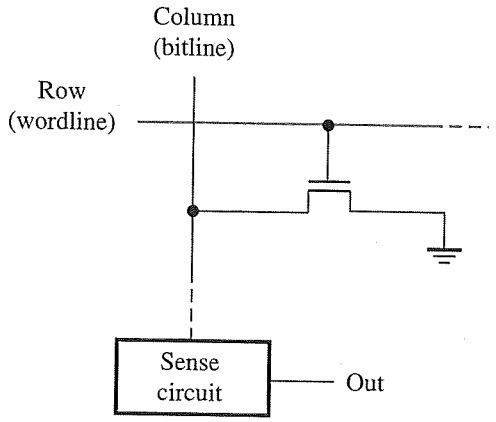


Figure 9.19
Basic ROM cell.

joining row and column. Figure 9.19 shows a single ROM cell with the gate attached to the wordline and the drain attached to the bitline. If the transistor is present at this cross point, it will pull the bitline low when the wordline goes high. However, if it is absent, the bitline will remain at its precharged voltage. This is the key concept in single transistor ROMs and provides the high density levels of the DRAM. This is different from DRAM cells since a transistor is present in every DRAM cell position and its operation relies mainly on charge sharing. In the case of the ROM, the cell may or may not contain a transistor and does require a capacitor for data storage.

Figure 9.20 shows the two basic forms of MOS ROM cell arrays: NOR arrays and NAND arrays. In each array, bits are stored according to the presence or absence of a transistor switch at each row-column intersection. The NMOS array of Figure 9.20a implements the NOR function in the sense that each column is one big NOR gate. A column goes low when any row, joined to the column with a transistor, is raised to a high level. In normal operation, all but one row line is held low. When a selected wordline is raised to V_{DD}, all transistors present in that row are turned on. The columns to which they are connected are pulled low. The remaining columns with transistors missing in their respective rows are held high by the pull-up or load devices shown at the top in this example. Column selection is performed using a decoder/MUX combination that enables a pass transistor in the data path.

ROMs of this type can be programmed by adjusting the contents of the masks, the so-called *mask-programmable* ROMs. Using positive logic, a stored "1" is defined as the absence of a transistor and a stored "0" by the presence of a transistor. Usually the array is formed with transistors at every row-column intersection. The desired bit pattern is placed in the array by omitting the drain or source connection at locations where a "1" is desired. Alternatively, a threshold implant can be used to raise the threshold voltage well above normal levels to "remove" a transistor from the array.

The array shown in Figure 9.20b is called a NAND ROM since each column forms one big NAND gate. The column output goes low only when all series bit

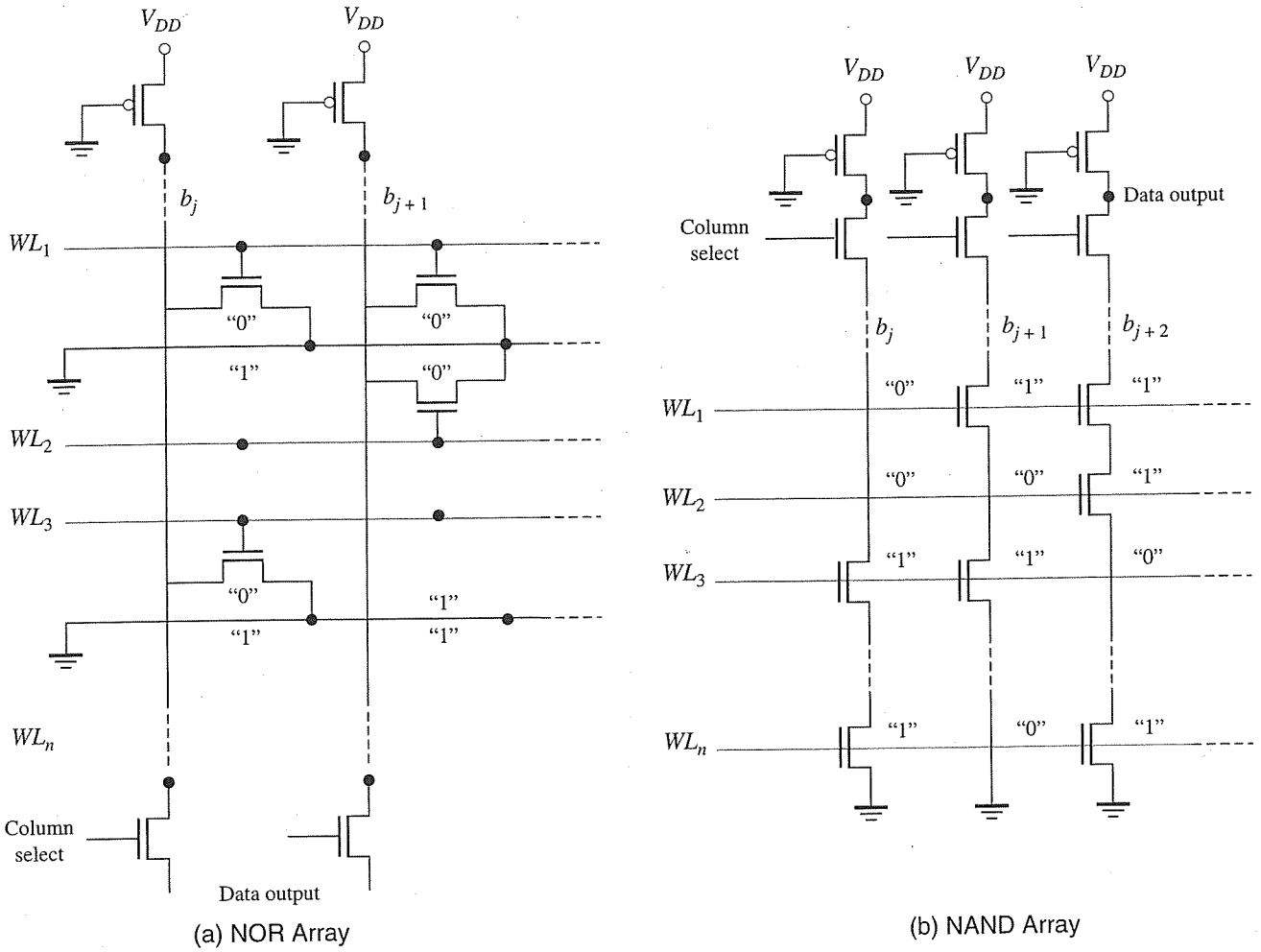


Figure 9.20
Alternative ROM array configurations.

locations provide a conducting path toward ground. In the NAND case, all the wordlines are normally held at V_{DD} in the standby condition. The selected row line is *pulled low,* and all transistors with gates connected to that line will turn off. The data output is taken at the top. In positive logic, a stored "1" is defined as the presence of a transistor, while a stored "0" is achieved with a direct connection (i.e., short) in place of the transistor. The array is formed with transistors at every intersection, but the source-drain paths of transistors at the desired "0" locations are shorted out by an implant or diffusion. Column selection transistors are placed in series with the bits to limit power dissipation.

The NOR array usually exhibits a faster access time and has the advantage that the stored bit pattern can be determined by the mask defining contacts to the transistors or the metal interconnection layer. Therefore, these ROMs may be kept in inventory with most of the fabrication processing completed, then customized quickly to a particular bit pattern by preparing and using a mask that makes contact only to transistors at 0-bit locations.

The NAND-based ROMs have longer access times and must be customized (usually with a masked *n*-type implant to form electrical connections between source and drain where "0" is to be stored) near the beginning of the manufacturing process. Their bit density per unit area is considerably higher than that for a NOR-based ROM using the same process and design rules. This is the key advantage of the NAND array.

The access time of a ROM is limited by the resistance and capacitance of row and column lines and the currents available to drive these lines. For large ROMs with a low cost per bit, emphasis is on high circuit layout density. This requires cell and decoder devices with small W and L, resulting in relatively small driving currents. Decoder drive current is limited by power consumption considerations as well as by device area considerations.

The detection of a "0" requires the use of a sensing circuit due to the large capacitance of the bitline. Typically, a current sensing amplifier is needed since the voltage will not drop appreciably in the available time to reliably detect a "0" in the presence of noise. Recall that in an SRAM, differential bitlines were used so that a voltage sensing amplifier could be used to effectively suppress common-mode noise. Here, a single-ended current sensing amplifier can be used to regenerate full logic levels, as shown in Figure 9.21.

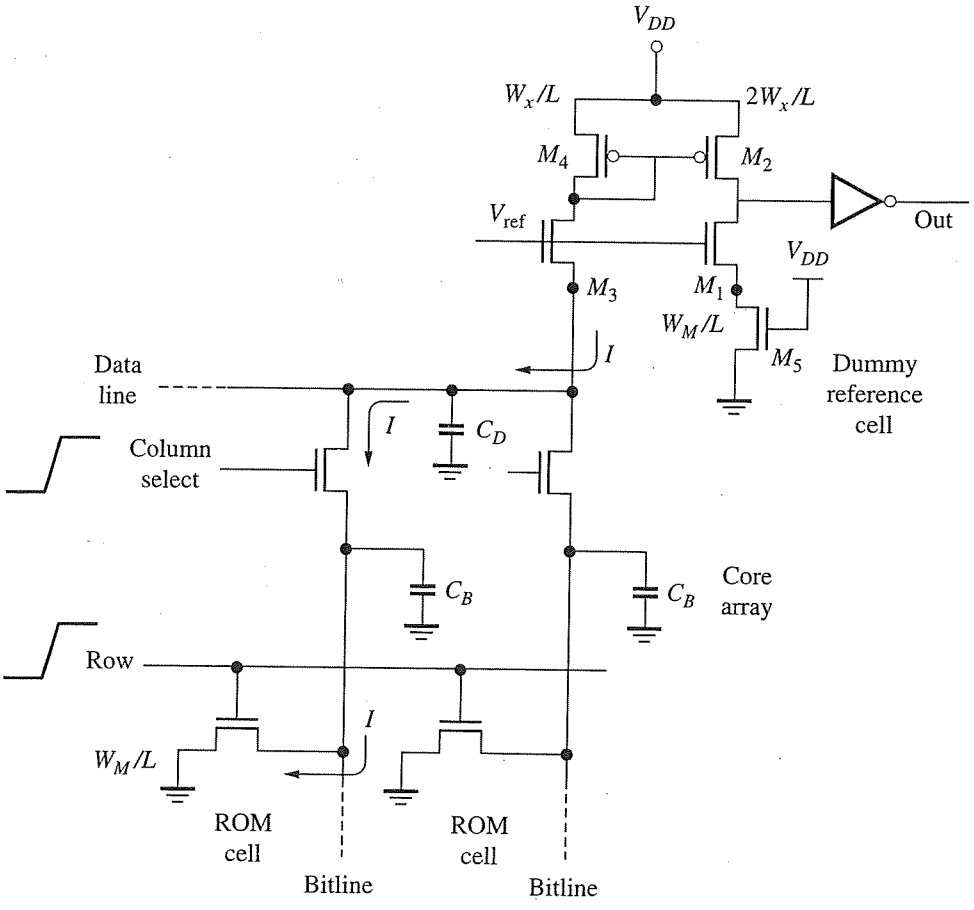


Figure 9.21
Current sensing amplifier.

The amplifier consists of a current mirror load for a common-gate amplifier controlled by a voltage, V_{ref}. When a row line selects a cell, current begins to flow in that cell. With the column select line high, this current flows from the data line capacitance and pulls the line down slightly. This turns on transistor M_3 and, as a result, current flows through M_4. This current is mirrored and doubled in M_2 since it is twice as large as M_4. Half of this current flows through M_1 while the rest flows to the gate of the inverter to charge up its input. Finally, node *Out* is discharged to zero. Hence, the presence of a transistor produces 0 V at the output.

If the transistor were not present in the ROM cell, no current would flow in the left branch of the amplifier. Therefore, M_4 and M_2 would have no current flowing through them. However, since V_{ref} keeps M_1 on, when M_5 turns on, the input of the inverter is pulled low. Therefore, its output is high. Hence, the absence of the transistor implies a 1 at the output. The dummy transistor, M_5, represents the cell transistor. In reality, the dummy cell is much more complicated than a single transistor since it must track process and operating condition variations that are experienced by the ROM cell transistor. However, it is shown as a single device connected to V_{DD} for simplicity.

9.6 EPROMs and E²PROMs

There are many applications where it is necessary to reprogram the ROM from time to time. For this purpose, the memory cell must be made erasable in some manner and then rewritable. This is not possible in conventional mask-programmable ROMs, but there are other more flexible ROM cells that are appropriate. This section describes two such memories: the EPROM and E²PROM.

The most widely used form of erasable and programmable ROM (EPROM) relies on a special MOS device structure shown in Figure 9.22a. Two layers of polysilicon form a double gate; gate 1 is a *floating gate* that has no electrical contact, while gate 2 is the *control gate* used to apply an external gate voltage. A circuit equivalent for the structure, with critical capacitances drawn, and its corresponding circuit symbol are provided in Figure 9.22b.

Initially, assume no charge on gate 1 or gate 2 and that the drain and source are connected to Gnd. As V_{G2} is increased, V_{G1} also rises but at a lower rate as determined by the capacitive divider. The effect of the series capacitances should be familiar from the bootstrapping and feedthrough concepts of Chapter 7. From Figure 9.22b, the series capacitance gives rise to the following relationship:

$$\Delta V_{G1} = \frac{C_2 \Delta V_{G2}}{C_1 + C_2}$$

$$\therefore V_{G1,new} = V_{G1,old} + \frac{C_2(V_{G2,new} - V_{G2,old})}{C_1 + C_2} \tag{9.1}$$

Since the transistor uses V_{G1} as its gate-source voltage, the control gate 2 must be made higher than usual to turn on the device. If V_T is the nominal threshold voltage of the device, it is not sufficient to make $V_{G2} > V_T$ to turn on the device. In

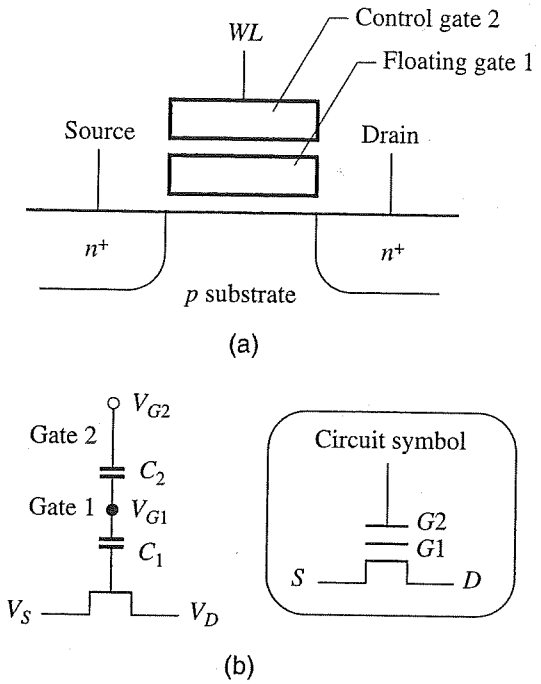


Figure 9.22
EPROM cell structure.

fact, we must make $V_{G1} > V_T$. Using this criteria and Equation (9.1), it is possible to determine the required voltage at V_{G2} to turn on the device. Assuming that all voltages initially start at 0 V, we require that

$$V_{G1} \geq V_T$$

$$\therefore V_{G1} = \frac{C_2 V_{G2}}{C_1 + C_2} \geq V_T$$

$$\therefore V_{G2} \geq \frac{(C_1 + C_2) V_T}{C_1}$$

Clearly, the voltage at V_{G2} must be higher than V_T by a factor of $(C_1 + C_2)/C_1$ so that V_{G1} exceeds the threshold voltage of the device to turn it on. Seen from external gate 2, the effective threshold voltage of the device has been increased.

The EPROM uses this feature to define the notion of a stored "0" and stored "1." The key is to program the value of the initial voltage of gate 1, called $V_{G1,old}$ in Equation (9.1), so that the external threshold voltage appears to be modified. If we can lower initial gate voltage at V_{G1} to some suitable negative value, then a nominal V_{DD} value at V_{G2} would not be able to turn on the device. From Equation (9.1), if $V_{G1,old}$ is negative then $V_{G1,new}$ will not be above V_T. This would be a stored "1." If we program V_{G1} to 0 V, then when $V_{G2} = V_{DD}$, the device would turn on as usual. This

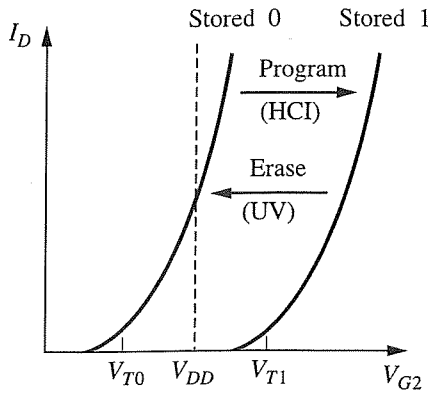


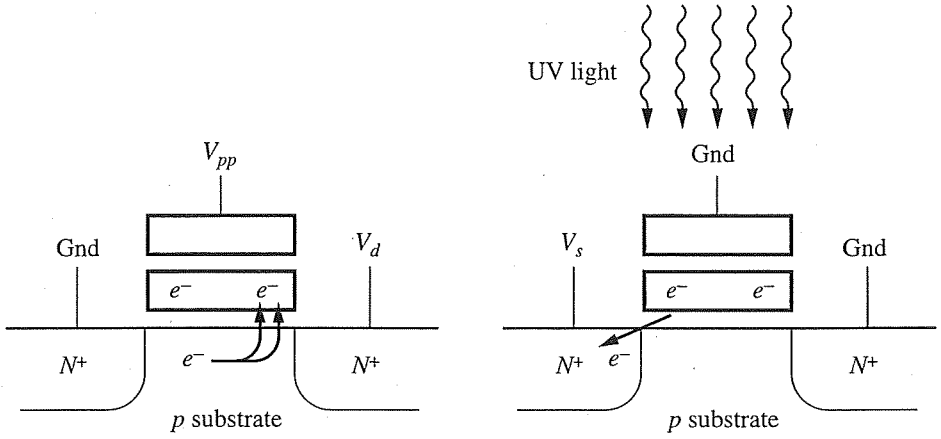
Figure 9.23
V_T adjustment when programming and erasing EPROM device.

would be considered a stored "0" as it would discharge the bitline like any other ROM cell.

The proper operation of this EPROM cell relies on the ability to store and remove charge from the internal floating gate. When programming the cell, we inject electrons through the oxide to the floating gate. From the perspective of gate 2, lowering the internal voltage is equivalent to raising the threshold voltage of this transistor from its initial value, V_{T0}, to a new value of V_{T1}. This is shown in Figure 9.23 in the plot of I_D versus V_{G2}. After programming, the effective threshold voltage is V_{T1}. When $V_{G2} = V_{DD}$, the device remains *off* because the voltage does not exceed V_{T1}. Since the device does not discharge the bitline, it is interpreted as a stored "1." To erase the cell, we must somehow remove the stored charge thereby increasing the internal voltage and reducing the effective threshold back to V_{T0}. When V_{DD} is applied at the gate in normal operation, the device will turn *on.*

To write a "1" in this cell, we use a mechanism called *hot-electron* injection. To initiate this mechanism, both gate 2 and the drain are raised well above 5 V while source and substrate remain grounded. A relatively large drain current flows due to normal device conduction characteristics. The high field in the drain-substrate depletion region results in an avalanche breakdown of the drain-substrate junction, with a considerable additional flow of current. The field accelerates electrons to high velocity, at which point they are referred to as *hot carriers.* A fraction of these carriers are injected into the thin oxide and become trapped on gate 1, as shown in Figure 9.24a. This so-called hot-carrier injection (HCI) reduces the internal node voltage until the vertical field is no longer sufficient to generate hot carriers—in this sense, it is a self-limiting process. When gate 2 and drain potentials are forced back to zero, the negative charge remains on gate 1 and forces its potential to a negative value through the capacitive feedthrough process. If V_{G2} for reading is limited to a few volts, a channel never forms since V_{G2} does not exceed V_{T1}. Thus, a logic "1" is stored in the cell as shown in Figure 9.23.

Once programmed, the data is preserved even if the power is turned off. Gate 1 is completely surrounded by silicon dioxide (SiO_2), an excellent insulator, so charge can be stored for many years. Data may be erased, however, by exposing the cells to



(a) Write process — hot-carrier injection (b) Erase process — UV light

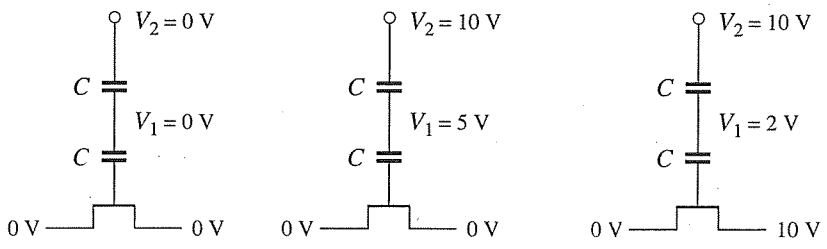
Figure 9.24
E^2PROM write/erase process.

strong ultraviolet (UV) light, as illustrated in Figure 9.24b. The *UV radiation* renders the SiO_2 slightly conductive by direct generation of hole-electron pairs in this material. With proper transistor biasing, the store charge can be removed. Unfortunately, these EPROMs must be assembled in packages with transparent covers so that they may be exposed to UV radiation. As such, these devices are only suitable as stand-alone parts. For embedded applications, other alternatives such as E^2PROMs and Flash memories must be considered.

Example 9.4 **EPROM Programming**

Problem:

An EPROM with a threshold voltage of 1 V relative to gate 1 starts with the floating gate voltage at 0 V as shown in (a). What is the threshold voltage relative to gate 2? An external voltage of 10 V is applied. The internal node voltage reaches 5 V since the two series capacitances have the same value as shown in (b). After hot-carrier injection, the internal node reduces to 2 V as shown in (c). Will this device turn on if an external voltage at gate node 2 is 5 V? If not, at what external voltage will it turn on?



(a) Initial state (b) After applying 10 V (c) After programming

Solution:

The V_T relative to gate 2 is 2 V based on the capacitive divider equation. After programming, when the external voltage is returned to 0 V, the internal voltage will drop to −3 V. When 5 V is applied, the internal voltage will reach −0.5 V which means that the device is off. To reach 1 V at the internal node, the external node will have to rise to 8 V. Therefore, the new threshold voltage relative to gate 2 is 8 V. Relative to Figure 9.23, $V_{T0} = 2$ V and $V_{T1} = 8$ V.

Electrically erasable PROMs (EEPROMs or E²PROMs) employ a somewhat different structure and mechanism for writing and erasing. The two-transistor (2T) structure for an E²PROM cell is shown in Figure 9.25. Since each cell requires two transistors, the bit density is lower than regular ROMs or EPROMs. In E²PROM cells, one transistor stores the content of the cell while the other is used for cell selection. The selection transistor has a normal gate oxide thickness. The storage transistor looks like the EPROM cell except that it has two oxide thicknesses. As seen in Figure 9.25, a portion of the dielectric separating gate 1 from body and drain is reduced in thickness to a value 10 nm or below. This is referred to as a *floating gate tunneling oxide* (FLOTOX) structure.

When approximately 10 V is applied across this thin dielectric, electrons flow to/from gate 1 by a conduction mechanism known as Fowler-Nordheim (FN) tunneling. That is, when the electric field across the oxide exceeds 10^7 V/cm, current flow through the oxide is possible using tunneling. The current due to tunneling increases linearly with the applied voltage, as indicated in Figure 9.26, and this mechanism is reversible. Therefore, erasure is achieved simply by reversing the applied voltage. The effect of injecting electrons into the floating gate is the same as before: the effective threshold voltage increases. Therefore, a normal voltage level applied at the gate of the FLOTOX device does not turn on the device after programming. If electrons are removed from the floating gate in the reverse process, the effective threshold voltage is reduced. At typical high gate voltage levels, the device will turn on.

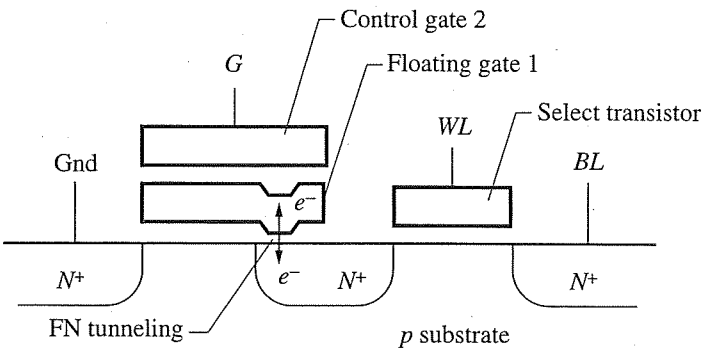


Figure 9.25
E²PROM 2T structure.

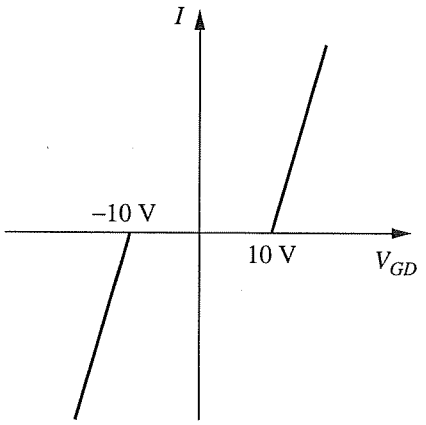
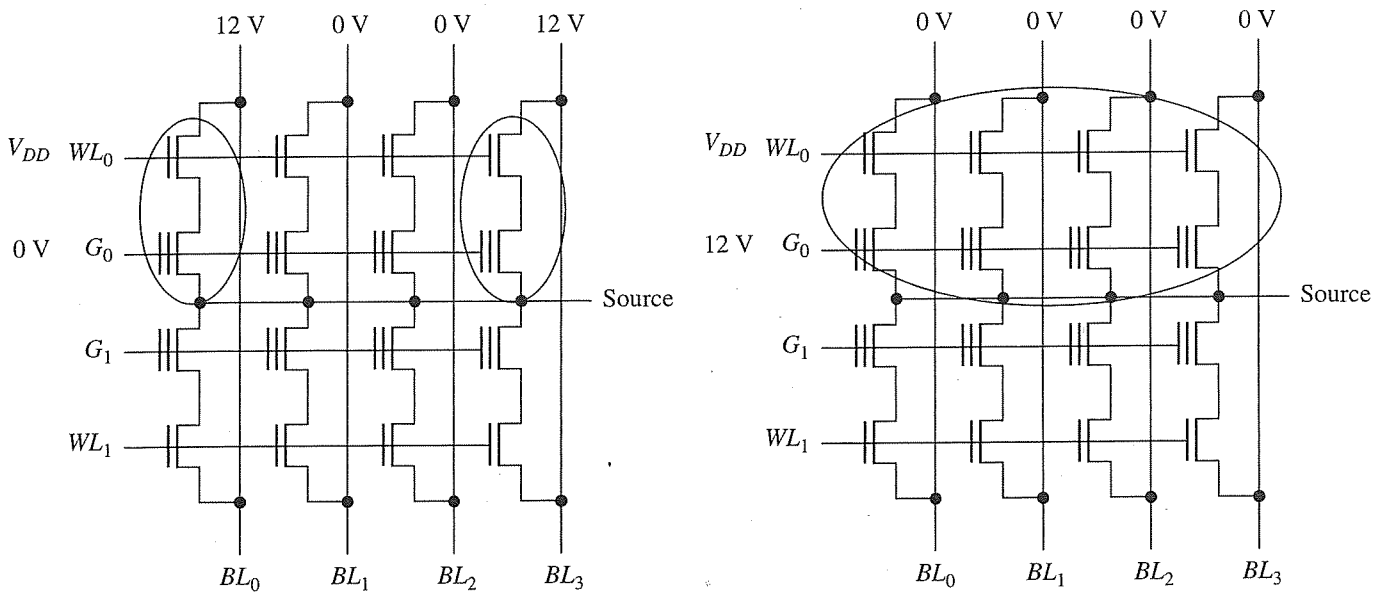


Figure 9.26
Fowler-Nordheim tunneling characteristic for 10 nm oxide.

Unfortunately, the tunneling mechanism is not self-limiting like HCI. A problem occurs if too much charge is removed during erasure. The V_T may go below 0 V to effectively create a device that does not turn off even when 0 V is applied at the gate. The reason for the select device is that the threshold voltage is difficult to control precisely since it depends on variations in the fabrication process and on the initial charge stored on the floating gate. If the desired internal voltage levels cannot be reached reliably due to initial stored charge, then the memory cell will not function properly. To avoid this problem, a select transistor is placed in series with the FLOTOX transistor and connected to the wordline and bitline. The select transistor uses the normal wordline voltage levels, whereas the FLOTOX transistor has an appropriate gate voltage that sits between the two possible thresholds, as shown in Figure 9.23. This way, the proper gate voltage can be selected after fabrication to ensure reliable operation.

The arrangement of eight E²PROM cells in an array is shown in Figure 9.27 using the 2T cells of Figure 9.25. The wordline drives the selection transistors while a separate control is used for the gate of the storage transistors. Consider the write operation for an E²PROM cell in Figure 9.27a. When a wordline is selected, the bitline is raised to 12 to 15 V for programming while the gate is held at ground and the source is left floating. This forces electrons to tunnel from gate 1 through the thin oxide to produce a net positive charge on gate 1. The effect of this is to reduce the threshold voltage of the storage transistor. Erasure is performed by applying a large voltage at the gate of the memory transistor while setting the bitline to 0 and letting the source float, as shown in Figure 9.27b. Electrons now flow through the thin oxide to gate 1. Erasure stops when the threshold voltage increases to a value above V_{DD}. Typically, write and erase operations are performed on bytes, although bit-by-bit access is possible in E²PROMs.

One issue arises if the E²PROM is programmed and then erased on a frequent basis. The two threshold voltages of Figure 9.23 begin to shift towards each other. In fact, after one million write/erase cycles or more, the difference in the two threshold voltages is too small to be of value in this application. E²PROMs require thresholds



(a) Write (reduce V_T of circled FLOTOX devices)

(b) Erase (raise V_T of circled FLOTOX devices)

Figure 9.27
E²PROM array configurations for write/erase operations.

that are well separated as in Figure 9.23. Therefore, these devices are suitable only for 10^5–10^6 write/erase cycles before reliability becomes a problem.

To read from the array, the wordline for the select transistor is raised while the gate drive of the FLOTOX transistor is held at a constant voltage. This is illustrated in Figure 9.28. Here we assume that the FLOTOX gate drive is V_{DD}, although it does

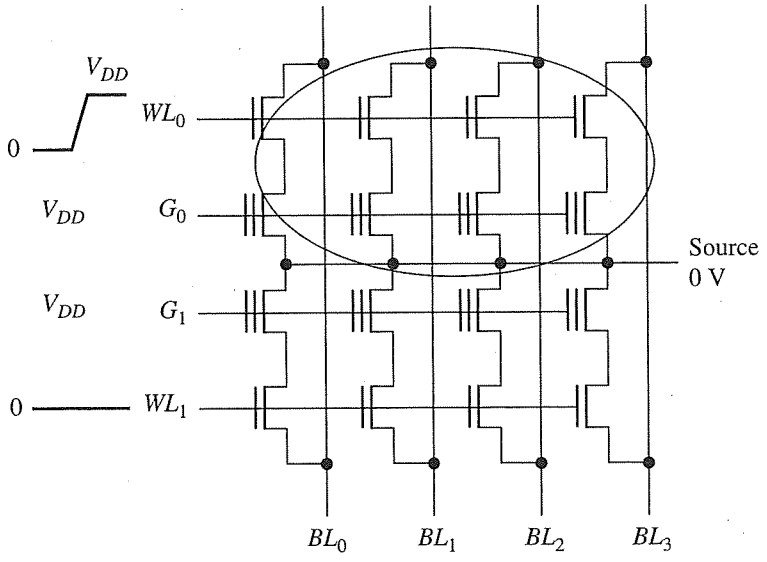


Figure 9.28
E²PROM read operation.

not necessarily have to be this value. The source is grounded during the read operation. If the FLOTOX device is *on*, the bitline will be discharged. If it is *off*, the bitline will remain high. The read operation proceeds in a similar way to the EPROM except that it is performed by activating the selection transistor, rather than the storage transistor.

*9.7 Flash Memory

Recently, E^2PROMs have seen increased competition for embedded applications from another type of nonvolatile memory referred to as *Flash* memory. The 1T cells used in Flash memories are preferred over 2T E^2PROMs due to the higher density. A NOR Flash memory architecture is shown in Figure 9.29. All cell transistors in the array have thin oxides and the select transistor of the E^2PROM is no longer present in this type of memory. This provides the higher density but does not allow a bit-by-bit modification of the array. The write/erase mechanisms of this NOR architecture are hot-carrier injection and Fowler-Nordheim tunneling, respectively. The source switch is adjusted depending on a write, erase, or read operation.

The write/erase operations are illustrated in Figure 9.30. During the write process, the source voltage is connected to Gnd as shown in Figure 9.30a. Then, a high value of V_d is applied to the selected bitline while a high value of V_{pp} is applied to the wordline. This creates channel hot carriers that enter the gate oxide and get trapped at the floating gate, thus raising the threshold voltage. This device will now

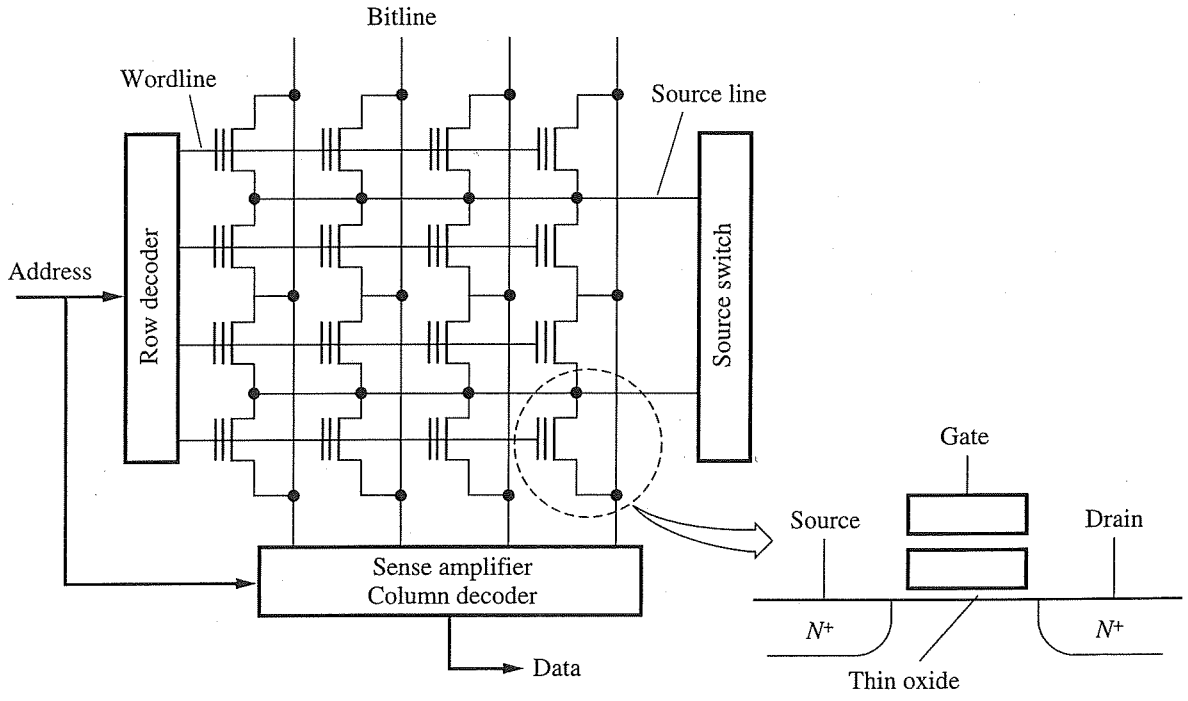


Figure 9.29
NOR Flash memory architecture.

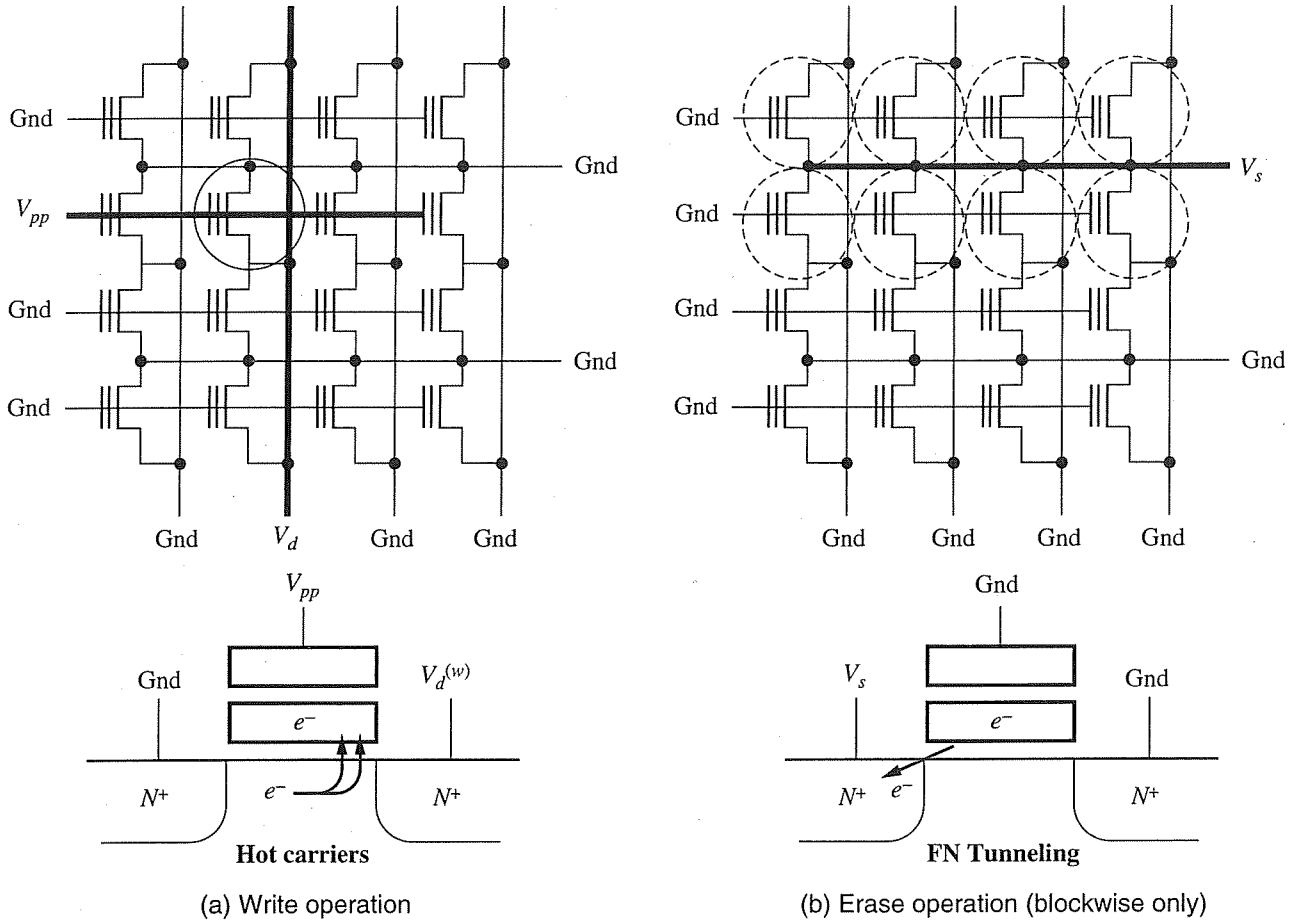


Figure 9.30
Write/erase operations of NOR Flash memory.

remain off even when the wordline is activated. Write operations can take place on a bit-by-bit basis. The erase operation requires the application of Gnd to the gate node and a high voltage of V_S to the source node. The erase mechanism is FN tunneling through the thin oxide. As shown in Figure 9.30b, the other transistors connected to the source connection are all erased as the same time, hence the term *Flash*. As such, erase operations are always carried out in block-mode in Flash memories.

The read operation is performed by applying Gnd to the source connection, precharging the bitline to V_d, and enabling the wordline with a voltage V_{read}. This is shown in Figure 9.31. The proper value of V_{read} must be selected to lie between the two thresholds of the device. If a drain current flows, then the bitline is discharged and interpreted as a stored "0." Otherwise, the bitline stays high and a "1" can be inferred. A sense amplifier needed for this type of memory is also shown in Figure 9.31.

A smaller and more dense Flash memory can be implemented if a NAND array is used, as illustrated in Figure 9.32. This type of memory uses FN tunneling for

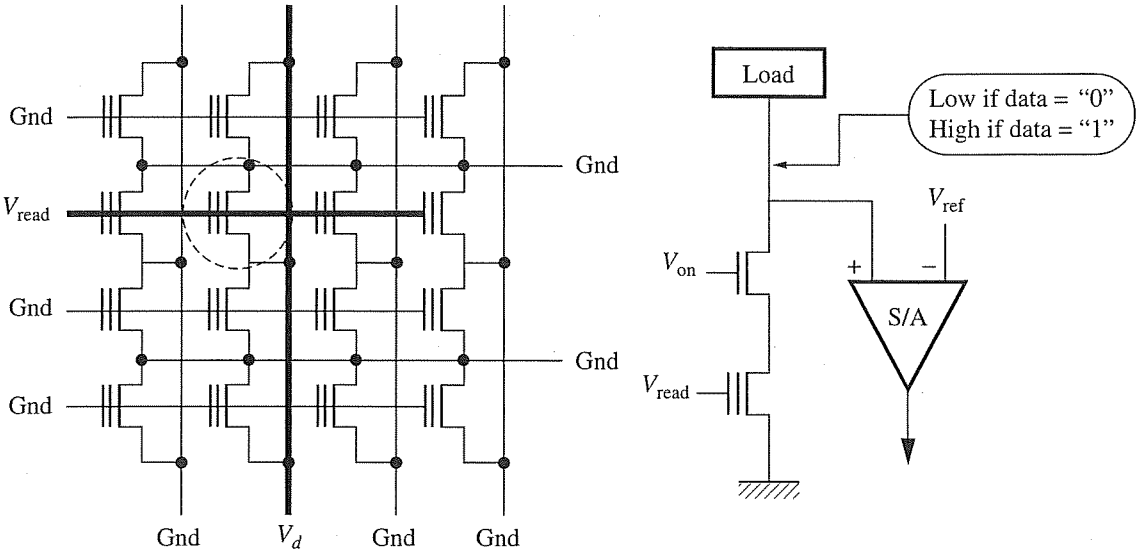


Figure 9.31
Read operation for NOR Flash memory.

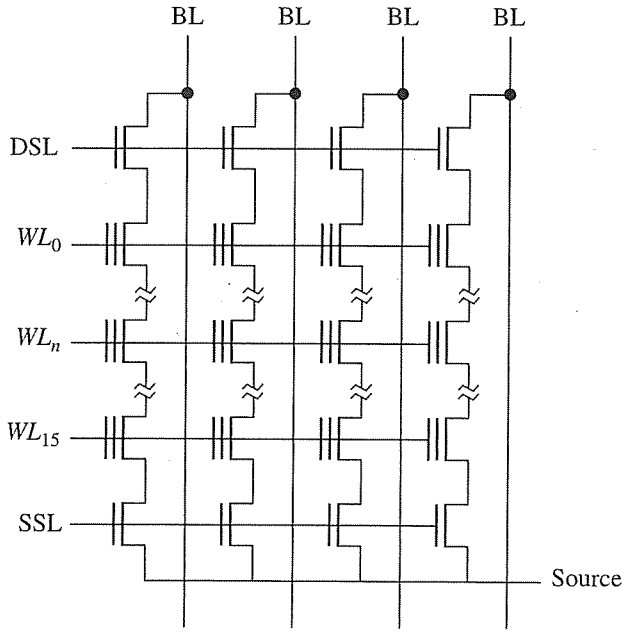


Figure 9.32
NAND Flash array.

write and erase which allows a much larger write/erase cycle limit, above 10^6 cycles. This is because programming using HCI damages the device more rapidly than programming with FN tunneling. The NAND Flash operates similar to the NAND ROM that was described earlier in this chapter. Wordlines are normally high, but

one will go low when the decoder is activated. Programming is carried out by setting the V_T to a high value using FN tunneling. Memories with 2 Gbits of storage capacity have been developed using this architecture which has applications in mass storage, MP3 players, and digital cameras.

*9.8 FRAMs

There is another flavor of nonvolatile memory based on ferroelectric materials called *FRAMs*. Recently, FRAMs have been shown to be useful in certain applications such as smart cards and may be more attractive in the future due to their extremely high storage density. The FRAM cell, shown in Figure 9.33, is similar to the DRAM cell with one transistor and one capacitor. However, the capacitor is composed of a *Perovskite crystal* material that can be polarized in one direction or the other to store the desired value.

When an electric field is applied in one direction or the other, the crystals polarize and hold that state even when the power supply is removed, thereby creating a nonvolatile memory. The polarization characteristic, P, exhibits hysteresis, as shown in the curves in Figure 9.33, which allows the capacitor to retain the direction of polarization. During a write operation, a voltage is applied in one direction ($-V_{CC}$) to store a "0" and in the opposite polarity (V_{CC}) to store a "1."

When reading, the bitline is held high and the wordline is enabled. The current level associated with the cell depends on the direction of the stored polariziation, P_r. If the dipoles switch state due to the applied field, a high current is detected by a sense amplifier. Otherwise, the current level is lower. Unfortunately, during the read process, the stored value of the cell is disturbed. That is, FRAMs also have a *destructive read-out*. Therefore, the value must be read and then written back into the cell similar to DRAM cells.

FRAMs require much less power and provide high density. The other notable feature of FRAMs is that they can endure hundreds of billions of read/write cycles

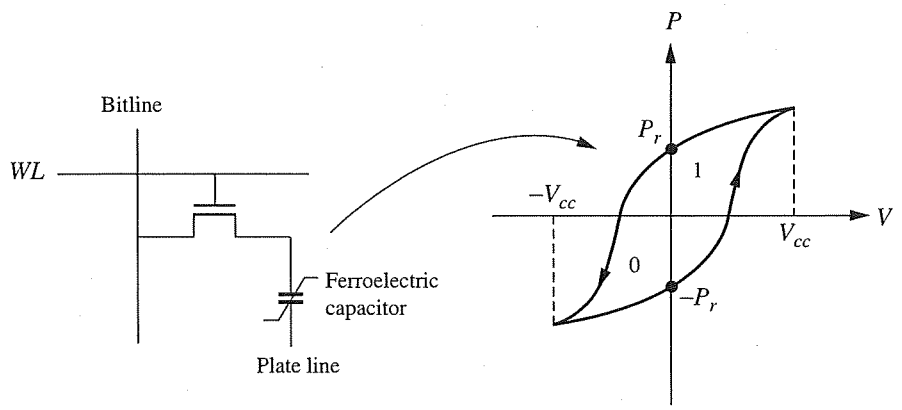
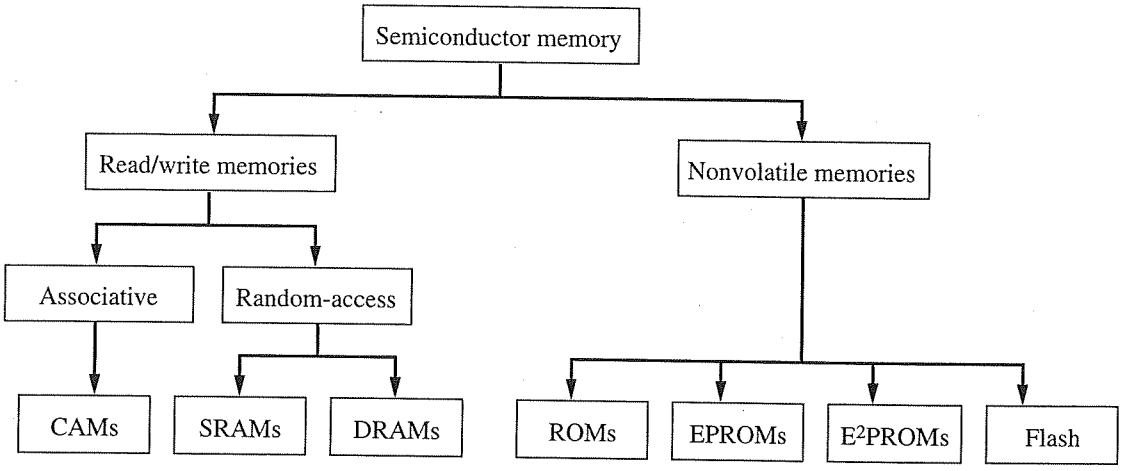


Figure 9.33
FRAM cell and polarization characteristic.

before reliability becomes an issue. In this respect, they are superior to all of the other nonvolatile memories discussed in this chapter. However, semiconductor memories are preferred over ferroelectric memories for most applications because of their advantages in cost, operating speed, and physical size.

9.9 Summary

A summary of the different types of semiconductor memories covered in Chapters 8 and 9 are shown below:



A summary of programmable nonvolatile memories is given in Table 9.1.

Table 9.1
Summary of programmable nonvolatile memories

Memory	Program Type	Erase Type	Erase Resolution	Cycles	Cell Size	Speed	Power
EPROM	HCI	UV	Full Memory	10^2	Small	Fast	High
E^2PROM	FN	FN	Bit/Byte	10^6	Large	Fast	Low
Flash	HCI/FN	FN	Block	10^5	Small	Fast	Low
FRAM	Polarization	Polarization	Bit	10^{10}–10^{12}	Small	Fast	Low

REFERENCES

1. B. Prince, *Emerging Memories: Technologies and Trends,* Kluwer Academic Publishers, Boston, MA, 2002.
2. K. Itoh, *VLSI Memory Chip Design,* Springer-Verlag, Heidelberg, 2001.
3. J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Designer Perspective,* Second Edition, Prentice-Hall, Upper Saddle River, NJ, 2003.
4. S. Trimberger, *Field-Programmable Gate Array Technology,* Kluwer Academic Publishers, Boston, MA, 1994.

5. V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, Boston, MA, 1999.

6. H. Veendrick, *Deep-Submicron CMOS ICs*, Second Edition, Kluwer Academic Publishers, Boston, MA, 2000.

PROBLEMS

P9.1. In the CAM design of Figure 9.3, the taglines are first precharged low and then the matchline is precharged high. Why are the tag lines precharged low rather than high? Why are they precharged low before the matchline is precharged high?

P9.2. You want to build a CAM cell with fewer transistors and one of your fellow designers comes up with the two designs shown in Figure P9.2. Which one do you prefer and why?

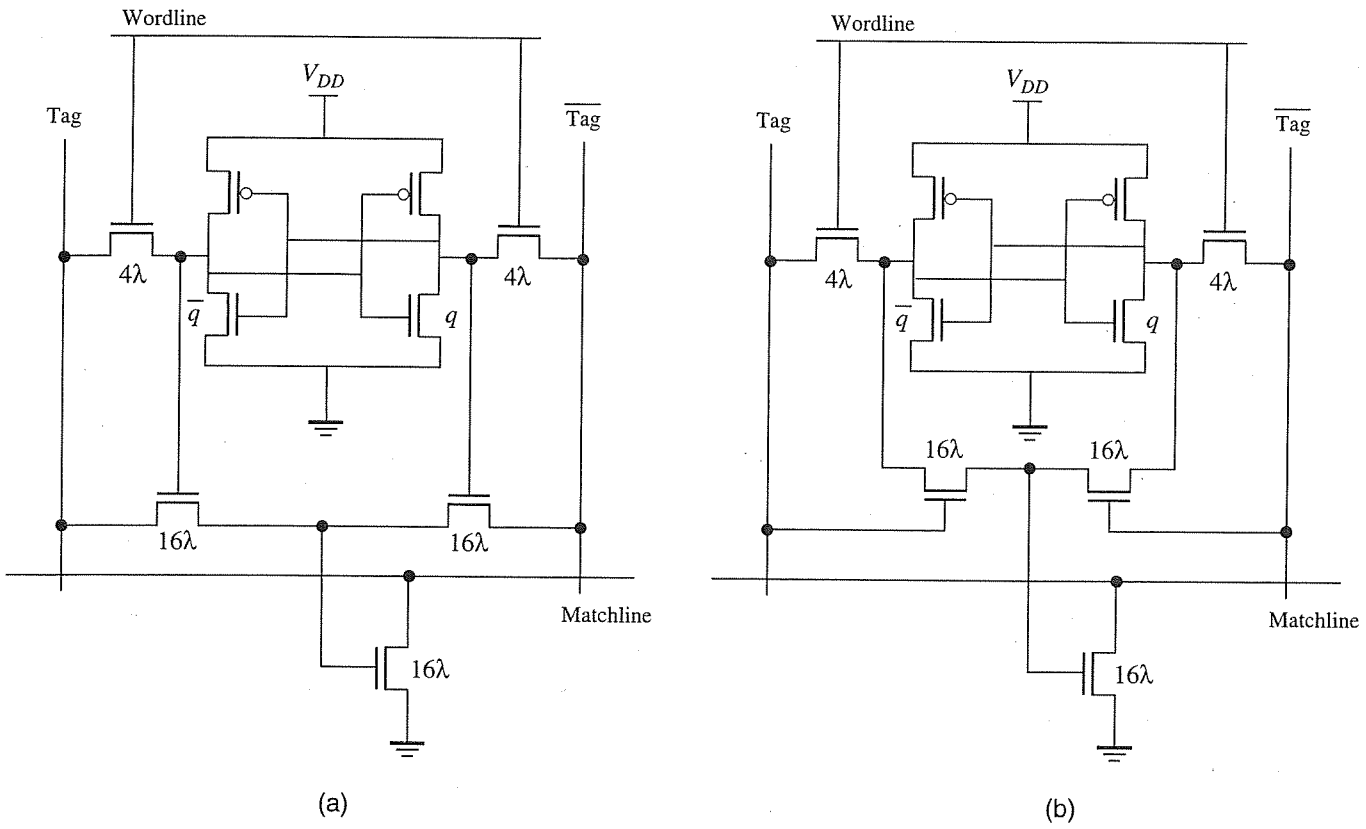


Figure P9.2

P9.3. Using the CAM cell of Problem P9.2(a), estimate the worst-case matchline delay. Assume that the CAM array portion is 33 cells across × 128 cells vertically. The cells are each 50λ wide by 80λ high. Identify the worst-case delay condition in the array. Then, compute the tagline

capacitance and matchline capacitances in detail. Assume that the input capacitance for the first tagline driver is 3 fF. Next, compute the tagline and matchline delays. Use 0.18 μm technology parameters.

P9.4. Redesign the transistor sizes for the CAM cell in Figure 9.5 to minimize the tagline and matchline delays. All transistors should be the same size. Use 0.18 μm technology parameters.

P9.5. Design a replica circuit for the CAM described in the problem above, assuming that the overall architecture is given in Figure 9.4. Describe the considerations in constructing an effective replica for this architecture.

P9.6. Consider the FPGA architecture shown in Figure 9.10. How would you implement a 4-input logic function in this array? Note that the PLBs contain 3-LUTs, not 4-LUTs so your function must span at least two PLBs.

P9.7. For the dynamic RAM cell shown in Figure 9.16, what value appears on the D column line when reading a "0" and when reading a "1"? What value appears on the $\overline{D}$ line as a reference voltage? Draw the bitline voltages for reading a "0" and a "1" assuming that the initial precharge value is $V_{DD}/2$. Include the effect of the regenerative sense amplifier. Assume 0.18 μm technology parameters.

P9.8. Compare SRAMs and DRAMs for speed, density, and power. Which technology is more easily embedded in logic chips?

P9.9. Compare the NOR array and NAND array for ROMs and Flash memories. Why is the NAND architecture used in many Flash applications while the NOR array is most popular for ROM applications?

P9.10. The circuit of Figure P9.10 is the core array of a 4 × 4 read-only memory (ROM). Metal 2 runs horizontally and Metal 1 runs vertically. The presence of a transistor in the array indicates a stored "1" while the absence of a transistor is a stored "0." For questions (a)–(d) below, you may use transistors or gates to complete the memory circuit by drawing the necessary circuits for the core array.

(a) Explain how this circuit works by labeling the wordlines and bitlines, and describe the operation of one read cycle.

(b) The circuit uses precharge devices for the bitline pullups. Draw in the pull-up devices on the bitlines. Are there any ratioing or charge-sharing issues to worry about? If so, explain. If not, why not?

(c) The circuit wordlines are to be controlled by a decoder circuit. Draw the appropriate decoder circuits using gates (not transistors) with the appropriate clock qualification. Why do we need the wordline to be driven by a qualified clock (i.e., ANDed with the clock signal)?

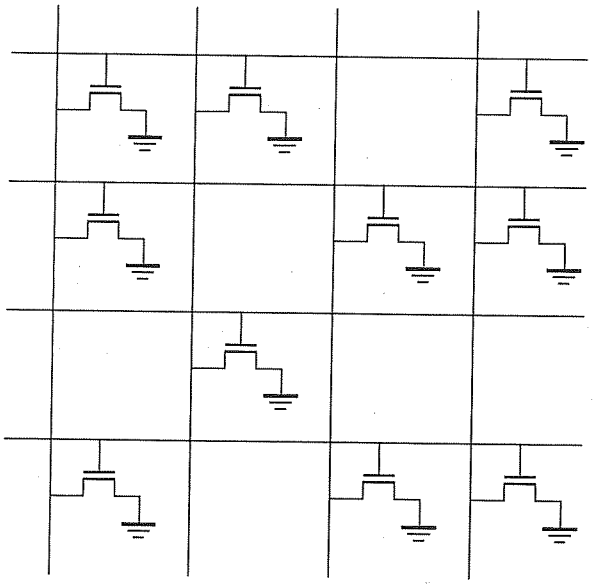


Figure P9.10

(d) The output of this circuit is a single bit from the bitlines (i.e., a bit addressable ROM). Draw a multiplexer circuit to select one bitline using transistors and gates to accomplish this on the diagram. Ensure that there is no V_T loss in the output and use any inverters/buffers that you need to produce the correct value at the output. Should we use qualified clock signal here? Why or why not?

P9.11. Explain why the select transistor is needed in an E²PROM cell? How does the EPROM cell avoid the need for this extra transistor?

P9.12. Consider the charge pump circuit shown in Figure P9.12 that is intended to generate a boosted wordline for a memory application. Assume that CLK is high initially and the output of the inverter is at Gnd. Also assume that node A is at $V_{DD} - V_T$ and node B is at Gnd.

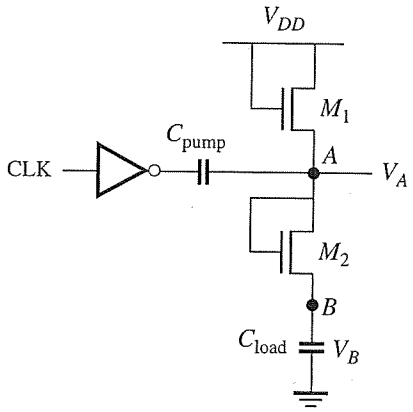


Figure P9.12

Draw the waveforms for nodes A and B as CLK switches back and forth from high to low. Check your results using SPICE.

P9.13. The 3T DRAM cell of Figure P9.13 uses minimum size devices. Assume that a 64K DRAM is designed in a 256 × 256 array configuration. Compute the cell current when reading a "0." How much cell current flows when reading a "1"? Draw the signal waveforms for the read and write operations for the wordlines, bitlines, and the internal node associated with C_1.

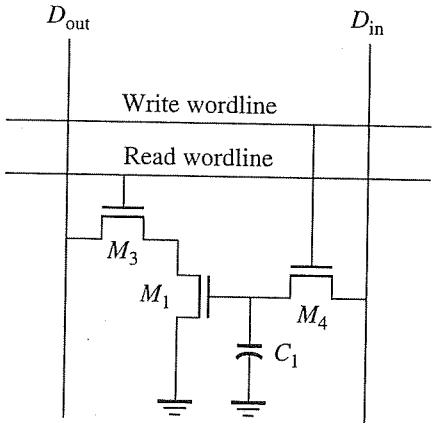


Figure P9.13

P9.14. Explain the difference between Fowler-Nordheim tunneling and hot-carrier injection. Which mechanism causes more damage to the oxide, and therefore limits the number of programs/erase cycles of nonvolatile memories?