TABLE 9.4-1
CMOS gate array library

Typical logic functions	
Triple 2-in NAND	2-2 O–A–I
Dual 3-in NAND	2-2 A–O–I
Triple 4-in NAND	D flip-flop
5-in NAND	D flip-flop with set, reset
Dual 2-in NAND/AND	2-to-1 mux
Triple 3-in NAND/AND	1-of-4 decoder with enable (act "L")
Triple 2-in NOR	2-bit magnitude comparator
Dual 3-in NOR	Mux D flip-flop with reset
Triple 4-in NOR	D flip-flop with preset, reset
5-in NOR	Toggle enable flip-flop with reset
Dual 2-in NOR/OR	D latch with reset
Triple 3-in NOR/OR	4-bit S–I/P–O SR
Triple clock buffer	Noninverting Schmitt
Quad inverter	1-bit ALU
Dual tri-state buffer	Full-adder
Tri-state noninv. buffer	4-to-1 data mux
EX-OR	4 bit parity checker
NAND latch plus 2-in NOR	4-bit S–I, P–I/P–O SR
Triple NAND latch	Presettable down counter with reset
NOR latch plus 2-in NOR	4-in mux with enable tri-state
Triple NOR latch	J-K flip-flop with set, reset

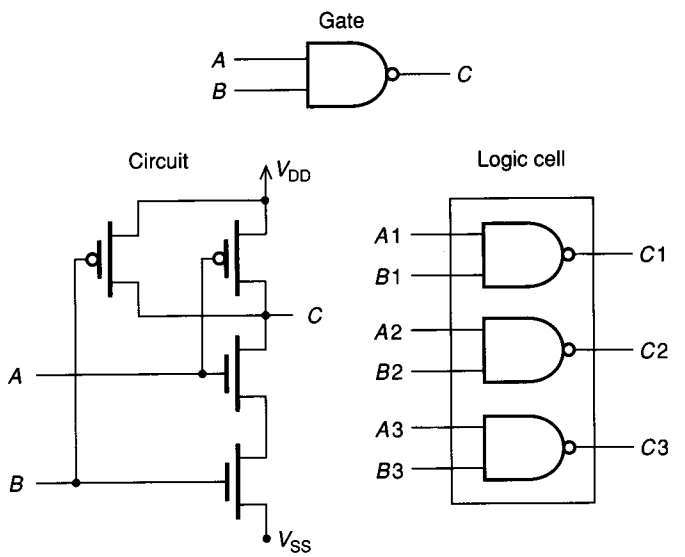

FIGURE 9.4-3
Triple two-input NAND logic block.

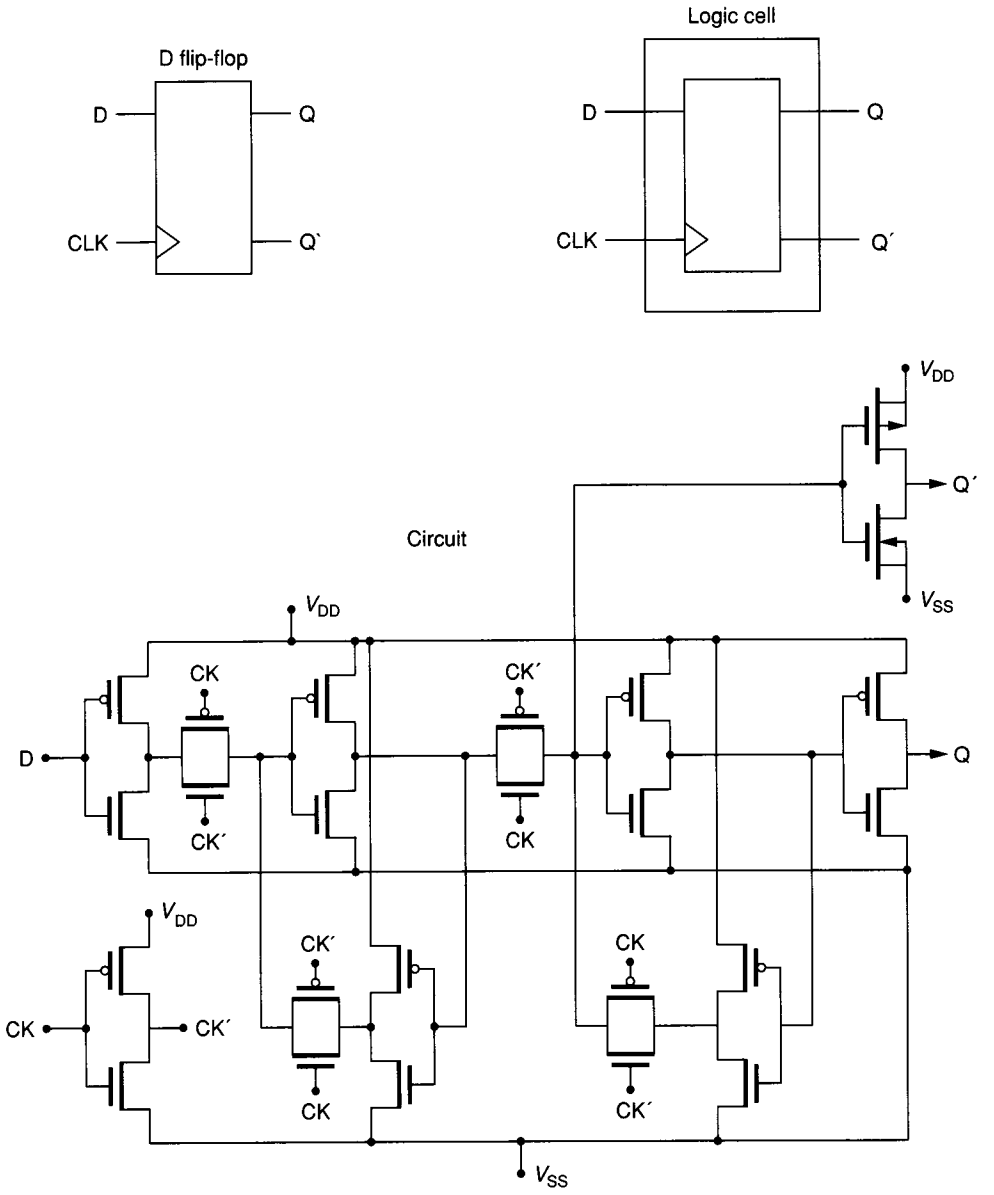

FIGURE 9.4-4
D flip-flop logic block.

Vertical metal interconnection between devices is accomplished through channels provided between each vertical row of blocks. Horizontal metal interconnection is accomplished through the blocks and around the ends of the rows. Ten horizontal wiring tracks are available per block for the typical circuits of Figs. 9.4-3 and 9.4-4. Most gate array manufacturers provide software tools to assist with placement of the logic modules and interconnection wiring. They also pro-

vide computer simulation tools for designers to verify the logical correctness and timing characteristics of proposed designs. There is interaction between placement and timing because interconnection lengths and, therefore, load capacitance change with logic module placements. Accurate simulation is a necessity for gate array designs as with all microelectronic circuits; it is not feasible to patch an integrated circuit with jumper wires, as is customary in correcting printed circuit board wiring errors.

Gate arrays are widely used to customize application-specific logic that would otherwise require many more IC packages. This results in considerable savings in area for many products. At least two levels of metal interconnect are generally required to implement designs with gate arrays. As the capability of technology has increased, the number of wiring layers for gate arrays has increased to three and even four layers of interconnect. Gate arrays are offered in sizes ranging from a few thousand gates to more than 100,000 gates at the present time.

In this section, logic gate arrays including both channeled gate arrays and the newer channelless, or sea-of-gates, structures were introduced. With gate arrays, logic designers can access the advantages of microelectronic circuits without becoming experts in the details of MOS circuit design. Both the global structure of gate arrays and examples of logic building blocks used in gate arrays were discussed. Finally, the requirement for CAD tools such as placement and routing programs, logic simulators, and accurate timing simulators was highlighted.

9.5 MOS CLOCKING SCHEMES

In preparation for the introduction of more complex digital systems containing storage devices and finite-state machines, the concept of clocking methods to control the movement of information is presented here. The combinational logic devices discussed previously do not require time-based control signals for their operation. The output of an ideal combinational logic circuit is completely defined at any time by the binary signals present as inputs to that circuit. For many applications, it is expedient to cause the output of a digital circuit to depend on both present and past inputs. For example, the output of a hand-held calculator in response to the "=" key depends on previous data and function inputs to the calculator. Digital circuits whose outputs depend on both present and past inputs are called *sequential* circuits; synchronous sequential circuits require a control signal to mark the passage of time and thereby delineate present inputs from past inputs. A digital signal called a *clock* serves this purpose by controlling the transfer of binary variables from one storage location to another.

An ideal clock signal is simply a periodic alternation of logic high and low voltage levels, as shown in Fig. 9.5-1*a*. Figure 9.5-1*b* shows a typical clock signal waveform as it might be observed on an oscilloscope. For 5 V logic, a single clock cycle is defined by

$$\text{clk} = 5 \text{ V} \qquad \text{for } 0 \leq t < t_1 \tag{9.5-1}$$

and

$$\text{clk} = 0 \text{ V} \qquad \text{for } t_1 \leq t < T \tag{9.5-2}$$

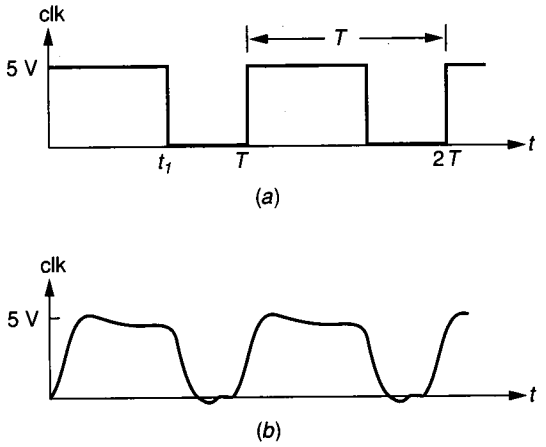

FIGURE 9.5-1
(a) Ideal clock signal, *(b)* Typical oscilloscope display of clock.

The time T represents the *period* of the clock signal; the inverse relationship $f = 1/T$ gives the *frequency* of the clock; and t_1/T is defined as the *duty cycle* of the clock. The square-wave output of a signal generator is an example of a clock signal with a 50% duty cycle. Circuits that operate from a single clock signal are said to use a *single-phase* clock. Much digital circuitry in the past was designed using single-phase clocks. Single-phase clocks could be used because of readily available binary storage devices (clocked flip-flops) that used one edge (either rising or falling) of the single-phase clock to update their stored value.

For reasons to be shown later, MOS logic circuits typically use *multiphase* clocks. Two-phase clocking schemes and four-phase clocking schemes derived from two-phase clocks are common. A *two-phase* clock is composed of two related sequences of alternating high and low voltage levels with the same frequency. A two-phase clock may be supplied to an integrated circuit from an external source or may be generated within the integrated circuit itself by special clock generation circuitry. Normally, a two-phase clock is composed of two nonoverlapping single-phase clock signals. Figure 9.5-2 shows nonoverlapping two-phase clock signals. The two-phase clock signals, ϕ_1 and ϕ_2 are said to be

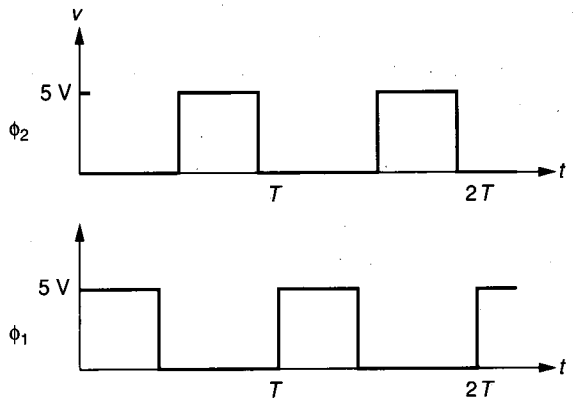

FIGURE 9.5-2
Nonoverlapping two-phase clock signals.

nonoverlapping because they are never both in the high state at the same time. This is represented by the logical AND function as

$$\phi_1 \cdot \phi_2 = 0 \qquad (9.5\text{-}3)$$

It is frequently desirable to derive a nonoverlapping two-phase clock from a single-phase clock signal. It might seem that this could be accomplished with the simple inverter circuit shown in Fig. 9.5-3*a*. However, a physical inverter circuit exhibits delay between its input and output signals, causing the derived two-phase clock signals to overlap when the input clock signal changes from a low to a high voltage level. This overlap condition is shown in Fig. 9.5-3*b*. The length of the overlap condition depends on the inverter delay. For practical inverter circuits, an overlap condition will always occur with this circuit.

Although a simple inverter cannot be used to generate a nonoverlapping two-phase clock from a single clock input, a slightly more complicated connection of simple logic gates will generate a nonoverlapping two-phase clock from a single clock input. The circuit in Fig. 9.5-4*a* generates a nonoverlapping two-phase clock with a nonoverlap time of at least one gate delay at each clock change. Figure 9.5-4*b* shows ideal waveforms for this circuit assuming identical, symmetric gate delays of length Δ for the inverter and the NOR gates. The reader should verify that the delays shown in Fig. 9.5-4*b* are correct. Asymmetric delays will change the time between clock edges, but will not cause overlap of the clock signals (see Prob. 9.12).

Clock waveforms have been presented in this section without restrictions on such clock characteristics as frequency, nonoverlap time, duty cycle, or rise and fall times. In fact, the clock signals shown have been idealized with zero rise and fall times. In practical circuits, clock signals are frequently required to drive a large number of gate inputs. The resulting capacitive loading can seriously degrade the rise and fall characteristics of clock signals, as was shown in Section

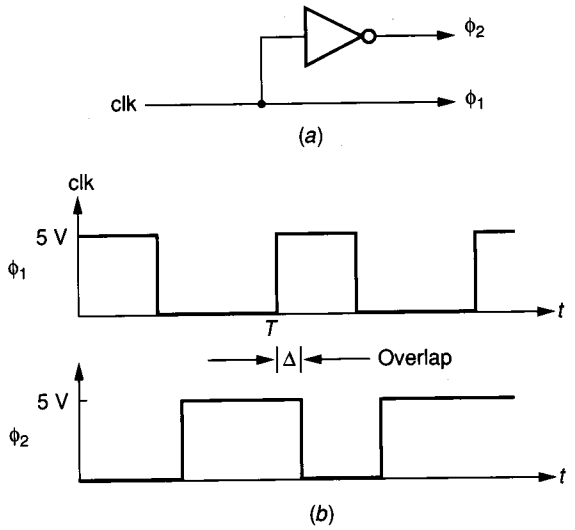

FIGURE 9.5-3
(a) Inverter used to generate two-phase clock, *(b)* Resulting clock waveforms showing overlap condition.

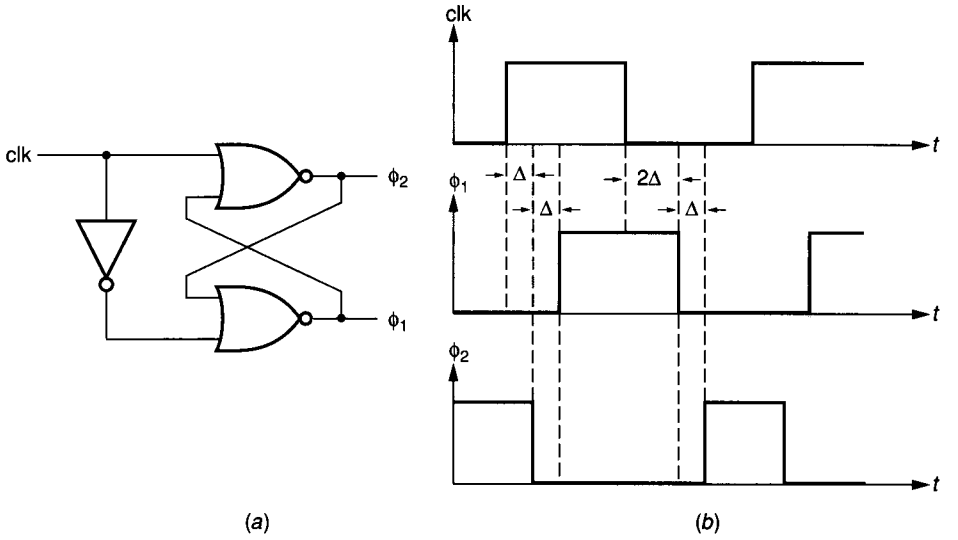

FIGURE 9.5-4
(a) Circuit used to generate a nonoverlapping two-phase clock from a single-phase clock input, *(b)* Ideal clock waveforms from the circuit of *(a)* assuming symmetrical clock delays.

7.9. To speed system operation and to minimize rise and fall times, clock signals are usually buffered and/or regenerated as they are used throughout the circuit. Exact device sizing for clock drivers depends on the size of the capacitive load and the speed with which the load must be driven. Techniques such as those discussed in Secs. 7.8 and 7.9 are used to calculate delays and set device sizes. Other characteristics of clock signals will be discussed in this chapter as they apply to the circuit under discussion.

9.6 DYNAMIC MOS STORAGE CIRCUITS

With the inverter, pass transistor or transmission gate, and multiphase clocks introduced in previous sections, the tools are in place to look at a useful storage mechanism within MOS circuits. This simple storage mechanism is termed *dynamic storage* and is widely used for momentary storage of data in digital circuits. The following subsections outline the structure and operation of dynamic MOS storage devices, particularly dynamic shift registers.

9.6.1 Dynamic Charge Storage

Among the technologies widely used for digital design, MOS technology provides two unusual features that lead to a particularly efficient way to store data momentarily. These two features are the MOS transistor's extremely high input resistance and the ability of an MOS transistor to function as a nearly ideal electrical switch. The circuit combination of these features with the source terminal

of one MOS transistor connected to the gate terminal of a second MOS transistor allows electrical charge to be stored momentarily on or removed from the gate terminal of the second transistor.

The three circuits of Fig. 9.6-1 show typical circuit configurations used to achieve dynamic charge storage. The pass transistor and transmission gate devices are often designed with minimum-size transistors to reduce layout area. The inverters are the simple inverters of Secs. 7.3 and 7.5 except for a higher sizing ratio k as explained in the next paragraph. Figure 9.6-1*a* is useful with NMOS circuits, while the other two are examples from CMOS circuits. Operation of the NMOS circuit will be explained to demonstrate dynamic charge storage, and then the changes required for CMOS will be noted.

Operation of the circuit of Fig. 9.6-1*a* depends on whether the pass transistor is off or on. If the gate of the pass transistor is at a high logic voltage, then the

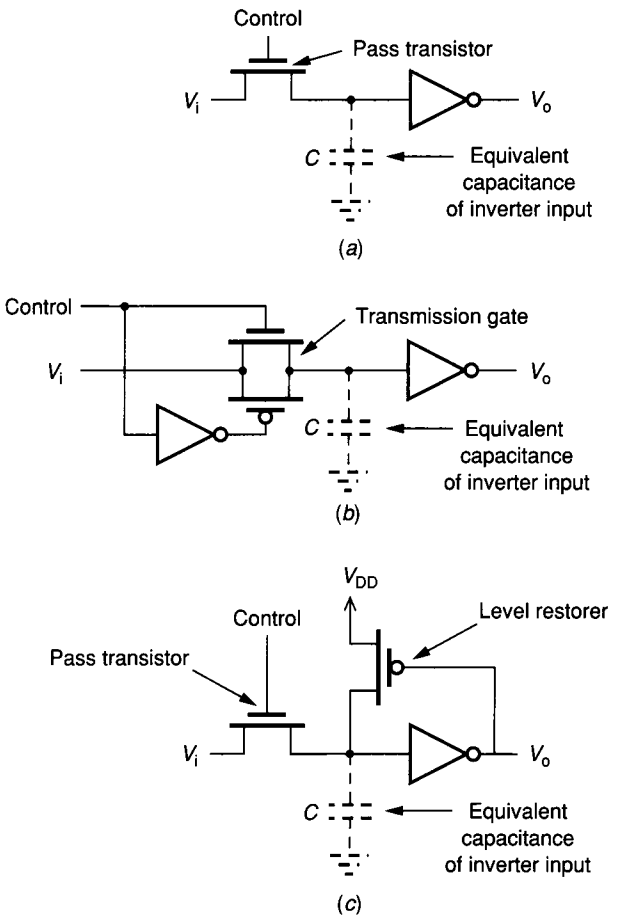

FIGURE 9.6-1
(a) NMOS dynamic storage circuit, *(b)* CMOS dynamic storage circuit, *(c)* CMOS pass transistor storage circuit with level restoration.

pass transistor conducts. In this case, the gate terminal of the inverter input transistor will be charged or discharged according to the logic voltage level at the input to the pass transistor. The time required to charge or discharge the gate terminal will depend on the gate capacitance, the pass transistor resistance, and the signal source. The gate can be discharged to 0 V, or can be charged to $V_{DD} - V_{TN}$. This sets the gate terminal to either a logic 0 or a logic 1 value, respectively. Because the inverter input voltage range has been reduced from the normal range of V_L to V_{DD} to a smaller range of V_L to $V_{DD} - V_{TN}$ by the pass transistor, the switching threshold voltage V_M should be lowered by increasing the inverter sizing ratio k from 4 to about 8. This can be shown through an analysis like that of Sec. 7.3.2.

When the gate of the pass transistor is at a logic low voltage, the pass transistor is off, thereby isolating any charge on the gate capacitance of the inverter input transistor. This charge (or the lack thereof) represents the stored logic value. If the stored charge were perfectly isolated, the logic value would be stored indefinitely. However, the isolation is less than perfect, primarily because of leakage through the reverse-biased diode created between the pass transistor source diffusion and the substrate. Leakage also occurs through the pass transistor switch. Because the stored charge will leak away over time, this circuit is termed a *dynamic storage circuit*. The following example examines the temporal characteristic of a typical dynamic storage node.

Example 9.6-1. For the NMOS circuit of Fig. 9.6-1*a* assume a pass transistor source diffusion area of 4 $\mu \times 5$ μ and an inverter input gate area of 9 μ^2. If the gate terminal capacitance is 1 fF/μ^2 and diffusion leakage current to substrate is 0.2 fA/μ^2, how long does it take for the stored voltage to change by 2.5 V?

Solution. The approximate capacitance is given by

$$C = 9\ \mu^2 \times 1\ \text{fF}/\mu^2 + 20\ \mu^2 \times 0.12\ \text{fF}/\mu^2 + 18\ \mu \times 0.2\ \text{fF}/\mu' = 15\ \text{fF}$$

and the leakage current is

$$I_r = 2\ \mu^2 \times 0.2\ \text{fA}/\mu^2 = 4 \times 10^{-3}\ \text{pA}$$

Then the time it takes to discharge the capacitance by 2.5 V is given by

$$t_{2.5} = 2.5C/I_r = 2.5\ \text{V} \times 15\ \text{fF}/4 \times 10^{-3}\ \text{pA} = 9.38\ \text{s}$$

This is clearly a long time compared to the clock periods of most digital circuits.

For dynamic storage with present MOS devices, the primary charge leakage path occurs through the diode between the source diffusion and the substrate. As MOS processes are created with linearly scaled-down devices, subthreshold leakage through the pass transistor's channel will become the predominant leakage factor.[12]

Dynamic storage can be implemented in CMOS by replacing the pass transistor with a transmission gate, as shown in Fig. 9.6-1*b*. Note the increase

in circuit complexity caused by the addition of the p-channel transistor in the transmission gate and the requirement for a dual-polarity control signal. This situation can be alleviated somewhat with the circuit of Fig. 9.6-1*c*. In this circuit, called a *level-restoring inverter*, an n-channel pass transistor is followed by a special inverter with a weak p-channel feedback transistor to restore the logic high level. The p-channel transistor must be sized to have an equivalent resistance much greater than the series pulldown resistance of the pass transistor and any circuit that drives the pass transistor input. The pass transistor can discharge the inverter gate to 0 V to give a good low logic level. In this case, the inverter output is high and the p-channel feedback transistor is off. As explained in Chapter 7, an n-channel pass transistor cannot raise the voltage high enough to ensure that the p-channel transistor of the inverter is off. Nevertheless, the pass transistor can pull the inverter input voltage high enough to force the inverter's output to a low logic voltage. This low voltage turns on the p-channel feedback transistor, thereby pulling the inverter input to the upper supply voltage and holding it there.

Dynamic storage is widely used within MOS circuits because of the simplicity of the required circuitry. The NMOS version of Fig. 9.6-1*a* requires only three transistors, while the CMOS version of Fig. 9.6-1*c* requires just four transistors. Thus, dynamic storage is area-efficient compared to the static storage circuits to be discussed later. A frequent use of dynamic storage circuits is to create shift registers. The following discussion shows shift registers that are built upon a generic MOS dynamic storage stage consisting of a pass transistor and a simple inverter for NMOS or a level-restoring inverter for CMOS.

9.6.2 Simple Shift Register

Figure 9.6-2 shows a multistage MOS shift register with each stage composed of a pass transistor and an inverter. The operation of this shift register can be described as follows based on a nonoverlapping two-phase clock. Assume that a logic signal is placed at the input of shift register stage A while the ϕ_1 clock

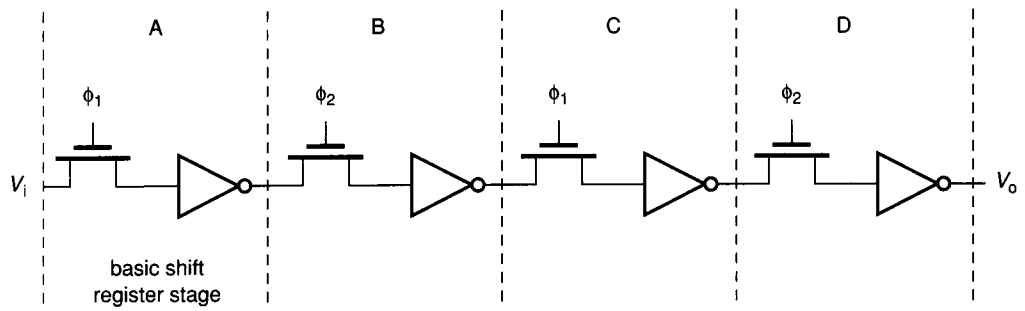

FIGURE 9.6-2
A linear shift register.

is low. Because the ϕ_1 clock is low, the pass transistor in stage A is off. Next, when the ϕ_1 clock goes high, if the signal at the input to stage A is held constant, it will be propagated to the input of inverter A. After a short delay, the output of inverter A will provide the inverted logic signal to the input of shift register stage B. At this time, the ϕ_2 clock is low and the pass transistor in stage B will not pass this input. When the clocks change so that ϕ_2 is high, the pass transistor of stage B will propagate the output signal of stage A to inverter B and then to the output of stage B. The signal will be stopped by the pass transistor of stage C because ϕ_1 is low while ϕ_2 is high. This sequence continues through the shift register chain as the clock signals alternate, causing the original input signal to propagate through the shift register stages.

At this point, a question should arise about the input to the inverter of shift register stage A when the ϕ_1 clock signal is low. In this instance, the input pass transistor of stage A is off, and the logic value is held by the dynamic storage of the input of the inverter. While the input to inverter A remains at its stored logic value, the output of inverter A will actively drive the input of stage B to the complementary logic level.

Each time the ϕ_1 clock changes to a high level, the shift register input signal will propagate to the gate of inverter A and on to the output of stage A. A sequence of alternating ϕ_1 and ϕ_2 clock signals will cause an input signal to propagate or shift through the structure at the rate of two stages of the shift register for each complete cycle of the clock signals. After N clock cycles, a logic input value will have shifted through $2N$ stages of the shift register. When a two-phase clock is used to control a shift register, it is important that the two clock phases do not overlap. If both phases of the clock were high simultaneously, a data value could propagate through multiple stages during the clock overlap time. This would result in uncontrolled operation of the shift register circuit and erratic movement of stored information.

Shift registers such as the one just described are used frequently within integrated circuits to provide temporary storage of digital signals. Such shift register storage can be used as a simple way to delay the arrival of a signal for a specific number of clock cycles. Shift register storage is also frequently used as the temporary memory for a sequential logic circuit. It will be shown later that a shift register can be combined with a PLA to provide a regular, expandable sequential machine. In general, shift registers provide dense, limited access memory for many applications within digital integrated circuits.

Figure 9.6-3 shows a parallel set of shift registers used to shift a group of signals in lock step fashion. As an example, such a parallel shift of 8, 16, or 32 data bits is sometimes required in microprocessor circuits. The basic structure of this set of shift registers demonstrates two principles important for the efficient geometrical layout of digital circuits. In Fig. 9.6-4, a symbolic layout diagram showing the geometrical topology for this circuit shows that the data for the shift register flows from left to right while the control signals (ϕ_1 and ϕ_2 clocks) flow from top to bottom. Such an orthogonal structure of data paths and control signals within a subsystem is widely used to provide a regular organization of logic

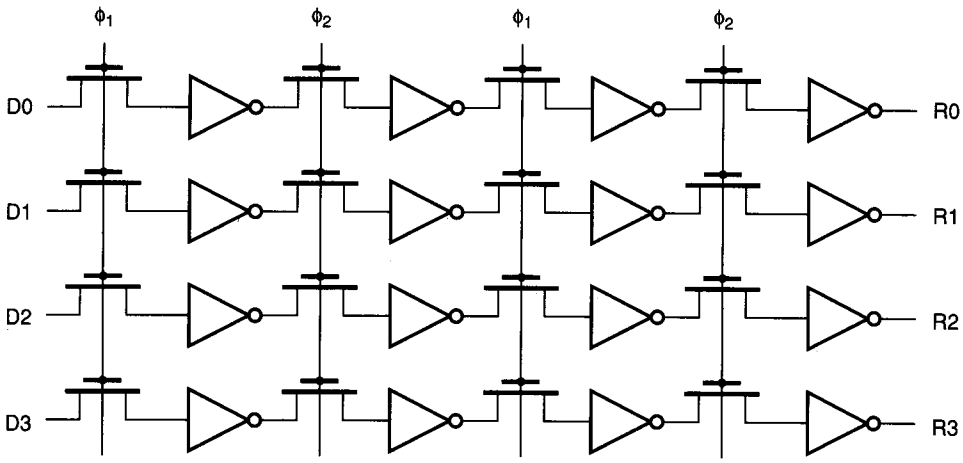

FIGURE 9.6-3
A parallel set of linear shift registers.

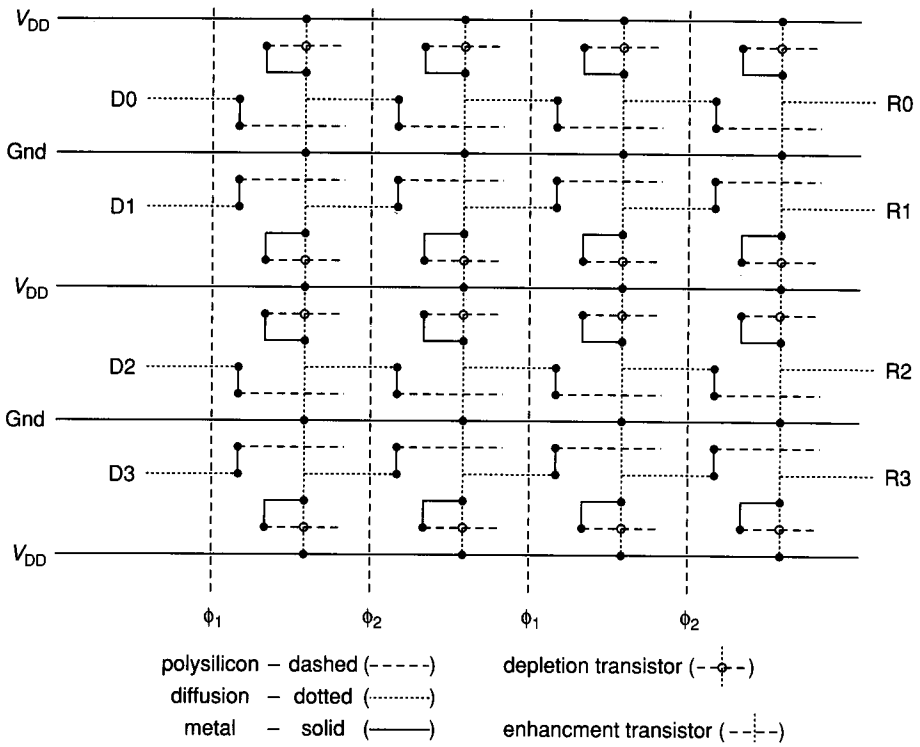

FIGURE 9.6-4
Symbolic layout diagram for a parallel set of linear shift registers.

circuits within an integrated circuit chip. The shift register stages are mirrored vertically about the ground and V_{DD} lines. This mirroring technique allows shared power and ground connections and reduces required circuit layout area. It is important to minimize the size of the basic shift register stage because this stage is repeated many times in a large shift register.

9.6.3 Other Shift Registers

A slightly different connection of the basic shift register cell is used to demonstrate another useful operation on a group of data signals. The need to shift the entire contents of a data word in one direction or the other is common in digital systems. If a data word is shifted toward the less significant bits, the equivalent binary weight of each bit is halved by each shift. If a data word is shifted toward its more significant bits, the equivalent binary weight of each bit is doubled by each shift. This doubling operation is useful in the conventional "shift and add" algorithm for binary multiplication.

Figure 9.6-5 shows a simple connection of shift register stages that allows a data word to be shifted toward higher significance or shifted directly along the data path according to a control signal. The control signal is ANDed with one

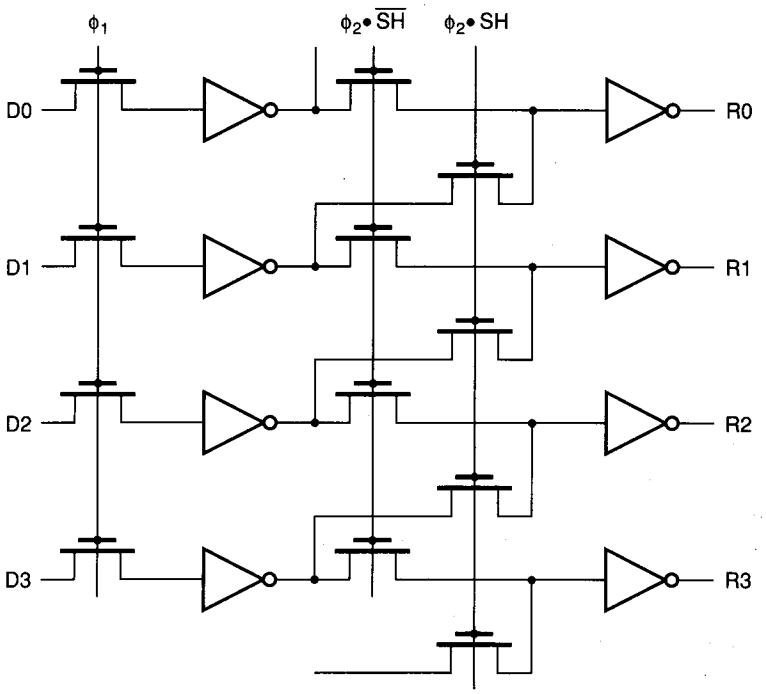

FIGURE 9.6-5
Four-bit shift-over, shift-up shift register.

clock phase to determine whether the data word is shifted or just passed along the data path. The layout of this circuit is similar to that of the parallel group of shift registers in Fig. 9.6-4 because the data and control signals are introduced orthogonally.

The purpose of this section has been to introduce dynamic storage, a most useful capability of MOS circuits. The ability to momentarily store logic values with minimal transistor and area requirements is widely used in digital ICs. The transient nature of the storage, typical storage circuits, and an application of dynamic storage to implement shift registers were all described.

9.7 CLOCKED CMOS LOGIC

Clocked logic in various forms has been used within digital MOS designs for many years.[13] Early use of clocked logic circuits was intended to minimize power dissipation in PMOS or NMOS logic. Present use of CMOS clocked logic circuits allows reduction of the number of transistors required within a design as compared with complementary static logic. Classical static CMOS logic gates require $2N$ transistors for an N-input gate, while NMOS logic gates require only $(N + 1)$ transistors. Clocked logic circuits for CMOS reduce the number of transistors to $(N + k)$ where k is a small constant overhead. This is accomplished by requiring dynamic storage of logic values within the gate structure (see Sec. 9.6). Clocked logic styles retain the desirable CMOS property of essentially zero static power dissipation. For this purpose transistors gated by clock signals, instead of complementary transistors gated by logic signals, are used to break the path between power and ground. Three styles of clocked CMOS logic are described here. The latter two have found wide application in large-scale digital circuits such as microprocessors and signal processors.

9.7.1 C²MOS

A dynamic shift register in CMOS is complicated by the need for a transmission gate and complementary clock signals rather than a pass transistor as described in Sec. 9.6. Figure 9.7-1 shows the four transistors and two clock signals required to construct a CMOS dynamic shift register stage. This construction can be

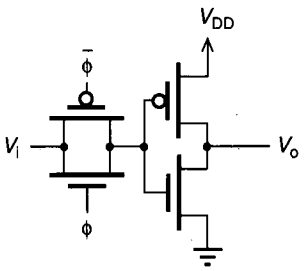

FIGURE 9.7-1
CMOS dynamic shift register stage.

simplified somewhat by the circuit configuration of Fig. 9.7-2. This form of dynamic shift register is called *clocked CMOS logic* or *C^2MOS*. In this circuit, the clocked transistors are placed in series with the p-channel and n-channel transistors of a standard inverter. The primary use of C^2MOS is within dynamic shift registers. All transistors can normally be sized as minimum-size devices because each stage is only required to drive the capacitance of an identical shift register stage.

Although the C^2MOS circuit requires the same number of transistors, external connections, and clock phases as the standard CMOS dynamic shift register of Fig. 9.7-1, the layout is simplified because the source/drain regions of the two p-channel transistors can be merged, and the corresponding regions of the two n-channel transistors can be merged. This reduces circuit capacitance, number of contacts, and layout area.

Operation of the C^2MOS circuit is quite simple. The gates of the p-channel pullup transistor and the n-channel pulldown transistor of the inverter are both connected to the input signal. For a valid logic input, one of these transistors will be off while the other is on. Clocked transistors placed in series with the pullup and pulldown transistors serve to connect these transistors to the output when the clock is high. For a high logic input, the output storage node will be discharged when the clock is high; for a low logic input, the output storage node will be charged when the clock is high. Otherwise, the output node will remain in its present state. In contrast to other clocked logic circuits, which are introduced in the following sections, the output of C^2MOS is available during the entire clock cycle, although it is actively driven only when the clock is high.

A problem with the C^2MOS circuit is that the load capacitance is the storage node for the dynamic charge. In the standard dynamic shift register, the storage node is inherently buffered from the output because the inverter gate is the storage node. Thus, the C^2MOS circuit is more susceptible to interference from the load circuit attached to the stage. If the load is an identical C^2MOS stage, the gates of the next stage can provide sufficient capacitance for the dynamic charge storage.

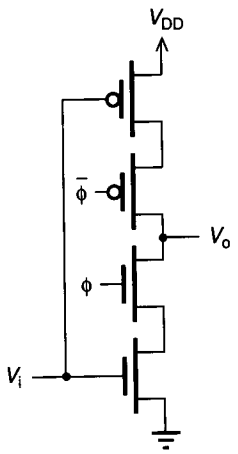

FIGURE 9.7-2
C^2MOS dynamic shift register stage.

9.7.2 Precharge-Evaluate Logic

A more general form of clocked logic, called *precharge-evaluate logic*, or *P-E logic*, provides low power dissipation like that obtainable with CMOS logic, and yet requires a transistor count comparable to NMOS logic. A basic tenet of such clocked logic is a tradeoff of output availability against power dissipation caused by the resistive pullup device of NMOS logic. If the path between power and ground is broken by two series transistors that are on at mutually exclusive times, no dc current path from power to ground will exist, nor will static power be consumed. Also, because there is no dc current path to place constraints on device sizing, minimum-size transistors can be used throughout to conserve layout area. The path to V_{DD} is used to precharge the output node during part of the clock cycle, and the path to ground is used to selectively discharge the output node during another part of the clock cycle. The output is taken high during the precharge time and is logically valid during the discharge cycle only after time is allowed to selectively discharge the output. Thus, for a square-wave clock signal, valid output availability is less than 50%.

The circuit of Fig. 9.7-3 shows a three-input NAND gate in P-E form. If the precharge transistor is a p-channel device and the discharge enable transistor is an n-channel device, a single clock signal will suffice. When the clock is in the low state, the p-channel transistor conducts and the output node is precharged to V_{DD}. When the clock signal goes to the high state, the n-channel transistor will enable discharge of the output node depending on the logic condition at the circuit's inputs. For the circuit of Fig. 9.7-3, the output is discharged only if all three inputs A, B, and C are in the high logic state. If any of these inputs is in the low logic state, the discharge path is broken and the output node is left charged to a logic high value. Thus, the gate realizes the NAND logic

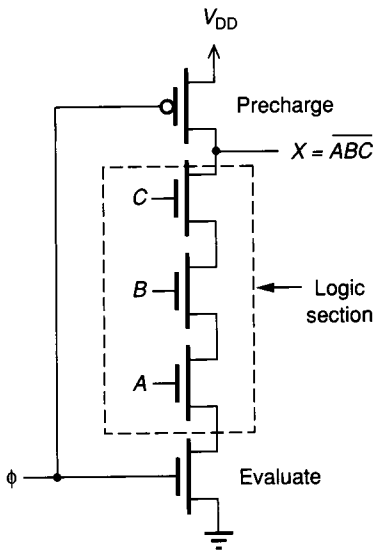

FIGURE 9.7-3
Three-input NAND gate in P-E form.

function. The P-E logic form allows realization of complex logic functions, as demonstrated in Fig. 9.7-4.

P-E CMOS logic has both advantages and disadvantages compared with classical static CMOS logic. In general, P-E logic requires less area than classical static CMOS logic because it does not require complementary transistor structures. The logic structure is ratioless, allowing use of minimum-size transistors throughout the gate logic. Because there is no dc pulldown current, a large number of transistors can be placed in series within the logic section. As another plus, it is possible for P-E logic to be faster than static logic because of lower gate loading on logic signals.

On the negative side, P-E logic has several disadvantages. The logic output value can be affected by a phenomenon called *charge sharing*. If a discharged node internal to the logic section is connected to the output node when the logic function is not satisfied, the output node charge will be shared with the discharged internal node, thereby degrading the output voltage level. Care must be exercised in circuit design to ensure that the output capacitance is larger than the internal node capacitances. P-E logic requires the addition of clock signals. There is a minimum clock rate because of the dynamic nature of the output signal, and the maximum clock rate is limited by circuit characteristics. The inputs must be stable during evaluation; otherwise, an incorrect value on the input could erroneously discharge the output node. Finally, the outputs must be stored during precharge if they are required during this phase of operation. These disadvantages are overcome by placing limits on allowable clock frequencies, and by careful selection of the types of circuits that are connected to P-E logic.

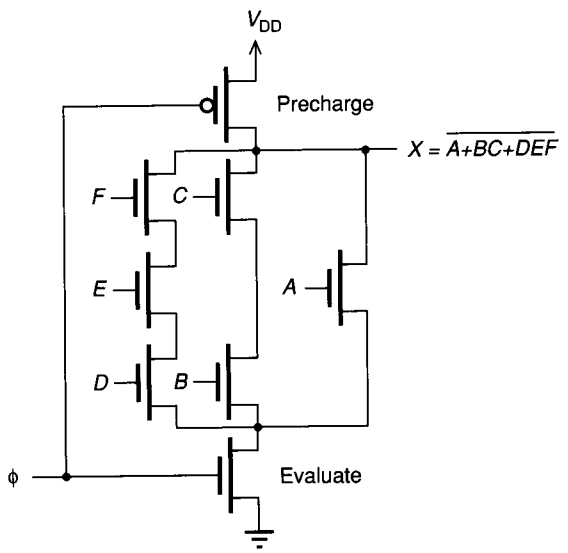

FIGURE 9.7-4
Complex logic gate in P-E form.

Multiple stages of P-E logic based on the same clock signal cannot be cascaded. Because the output of each stage of logic is driven to a logic high during the precharge phase, use of this output to drive secondary stages of P-E logic could erroneously satisfy their logic conditions immediately after the clock signal is pulled high. This could discharge the output of the secondary stage, preventing proper logical operation. A solution to this problem is to use cascaded stages with multiple clock signals so that the inputs to a stage are stable during its evaluation phase. Explanation of multiphase clock operation can be found in other sources.[14]

9.7.3 Domino CMOS

A variation on P-E logic, called *domino CMOS*, was popularized during the development of the BellMac microprocessor.[15] A domino logic gate consists of two elements: a P-E logic gate followed by a static inverter buffer at the output. The logic can be built in two forms: mostly n-channel, where the transistors comprising the logic are n-channel devices; and mostly p-channel, where the logic is performed by p-channel devices. The transistors used within the logic section can be minimum-size transistors. The static inverter at the output serves to buffer the logic part of the circuit from its output load, resulting in a more robust logic gate than standard P-E logic. The output inverter can be sized as desired, for example, to achieve symmetric output drive or to quickly drive a large capacitive load.

The behavior is explained based on the mostly n-channel form shown in Fig. 9.7-5. As with P-E logic, there is a precharge phase and an evaluation phase.

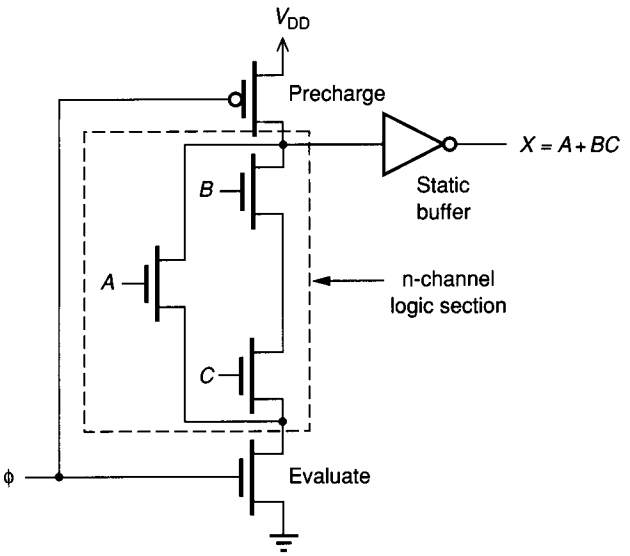

FIGURE 9.7-5
Domino CMOS logic gate.

During the precharge phase, the internal logic output is precharged to the high logic condition. This is inverted by the static buffer, providing a low logic output for the domino CMOS gate during the precharge phase. During the evaluation phase, the output of the logic part of the gate is selectively pulled low according to the logic input values. If the logic condition of the gate is satisfied, the internal output node is pulled low. This is subsequently inverted by the static buffer to provide a high logic output condition.

The domino CMOS gate has many of the same advantages and disadvantages described for P-E logic when compared to static logic. In addition, domino CMOS has advantages over the simpler P-E clocked logic form. For example, the static buffer provides output drive capability to either V_{DD} or ground. In P-E logic, the output can be driven only to ground in response to logical conditions, not to V_{DD}. When the logic condition of the P-E gate is not satisfied, dynamic charge storage at the output must maintain the high output value. The dynamic logic section of a domino CMOS gate always has a fan-out of one, thereby simplifying device sizing within the gate structure. As contrasted with P-E logic, domino CMOS stages can be cascaded successfully. The p- and n-channel transistors are easily grouped into a common n- or p-well, depending on the technology used. The fact that domino CMOS is a noninverting logic form provides at least one disadvantage over P-E logic. Lack of an inverting capability means that domino CMOS is not logically complete in the sense described in Sec. 7.2.

Examining the operation of the cascade of domino logic gates shown in Fig. 9.7-6 provides a basis to explain the choice of name for this clocked logic form. During the precharge phase with the clock signal near ground, the output of each domino stage is at the low logic condition. Thus, inputs to all subsequent domino stages are low. During evaluation, as the clock signal is pulled high, the outputs of some first-tier stages move to the high logic condition if their inputs are satisfied. The outputs of these gates may satisfy the logic for some second-

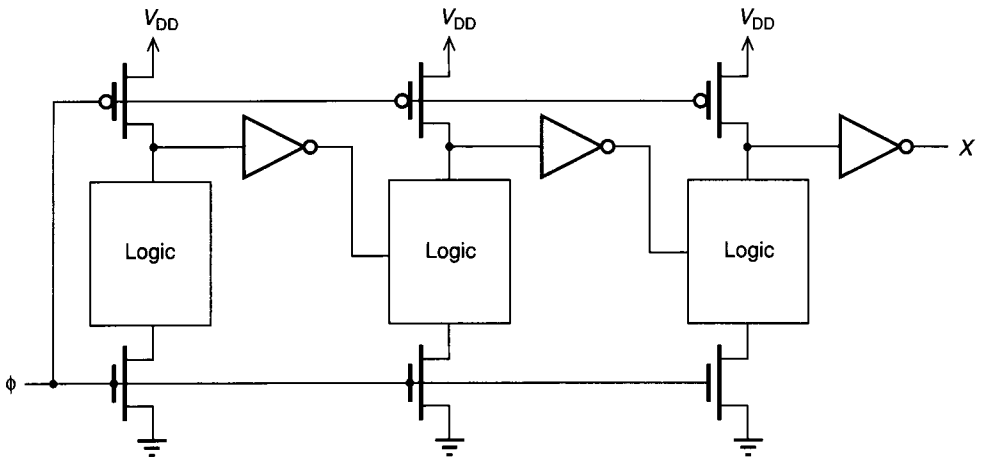

FIGURE 9.7-6
Cascade of domino logic stages.

tier stages, resulting in a high value at their outputs. In fact, during evaluation, logic decisions propagate through a cascade of stages like a falling domino chain.

As described, domino CMOS logic is a dynamic logic form. The precharged high at the input to the static inverter buffer is held by charge stored at the input to the static inverter and will not remain indefinitely. If this high could be maintained while the clock was stopped, a static logic form would result. The circuit of Fig. 9.7-7 shows a "keeper" transistor used to maintain a precharged high. This transistor can be formed as a weak static pullup device that contributes little to the pulldown current during evaluation or to static power dissipation. A weak static pullup is created by a large L:W ratio of 10 or 20:1 to increase the equivalent resistance of the transistor. This pullup transistor also improves noise immunity and allows a longer evaluation time.

Three styles of clocked logic were examined in this section. The first, C^2MOS, is useful primarily in shift registers. The latter two, P-E logic and domino CMOS logic are widely used for high-density, low-power implementations of logic equations. Both require only $N + k$ transistors, where N is the number of inputs and $k = 2$ for P-E logic and $k = 4$ for domino logic.

9.8 SEMICONDUCTOR MEMORIES

Integrated circuit memories provide the opportunity for semiconductor manufacturers to excel at their forte. That is, they create large, dense arrays of small cells with a process that is finely tuned for yield and performance characteristics. They strive to optimize the circuit design and the layout of individual memory cells to provide the maximum data storage capability for a memory part. Because the demand for dense semiconductor memory seems insatiable, a manufacturer has the opportunity to recover significant development costs with large sales volume

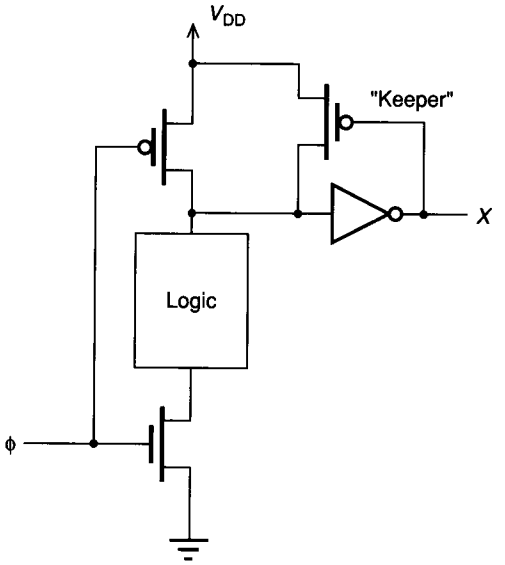

FIGURE 9.7-7
Use of "keeper" transistor to form static domino CMOS logic gate.

for a new memory design. With fierce competition for domination of the large market for memory devices, manufacturers constantly search for ways to create devices with smaller geometries and for clever circuit ideas that shrink the size of the basic memory cell or enhance its performance.

The excellent results of the research effort to provide small, dense semiconductor memories are rapidly incorporated into useful components of digital integrated system design. In many new microprocessors, ROM (read-only memory) partially replaces random logic in the control section. Designers of digital systems often find that dense memory is a good replacement for logic in other applications as well. Standard cell design libraries often provide forms of semiconductor memory as basic building blocks for systems. This trend toward increased use of memory requires digital system designers to be familiar with various types of semiconductor memory as tools for structured integrated circuit design.

9.8.1 Memory Organization

Semiconductor memories are universally organized as arrays of single-bit storage cells. These arrays are encircled by address decoding logic and interface circuitry to external signals. Figure 9.8-1 provides a block diagram of a typical memory chip organization. The memory array nominally uses a square organization ($m = n$ in Fig. 9.8-1) to minimize the external decoding circuitry necessary to select a particular memory cell. The reason for the square design can be seen by considering a memory part that contains 16k 1-bit storage cells. A memory array with 16k locations requires 14 address lines to allow selection of each bit ($2^{14} = 16,384$). If the array were organized as a single row of 16k bits, a 14-to-16,384-line decoder would be necessary to allow individual selection of the bit addressed by the 14 address lines. However, if this memory is organized as a 128-row by 128-column square, one 7-to-128-line decoder is required to select a row, and a second 7-to-128-line decoder is necessary to select a column. Note that a 128-row by 128-column matrix contains 16,384 crosspoints that allow access to individual memory bits. Thus, the square organization requires much less area for the address decoding circuitry than the linear organization.

Most memory chips operate such that the row address enables all cells along the selected row. The contents of these cells become available along the column lines. The column address is used to select the particular column containing the desired data bit. This data bit is ultimately routed to drive an output pin of the memory part. Some memory parts are organized so that n bits are accessed simultaneously. For these memories, the data from n columns are selected and gated to n data output pins simultaneously. Additional circuitry, including sense amplifiers, control logic, and tri-state input/output buffers, is normally required to create a functional memory part. However, the size of the memory storage cell and the resulting memory array are of primary importance in determining the size of the complete memory chip.

Several types of MOS semiconductor memory are in wide use today. These include ROM (read-only memory), EPROM (erasable programmable ROM), EEPROM (electrically erasable programmable ROM), SRAM (static random

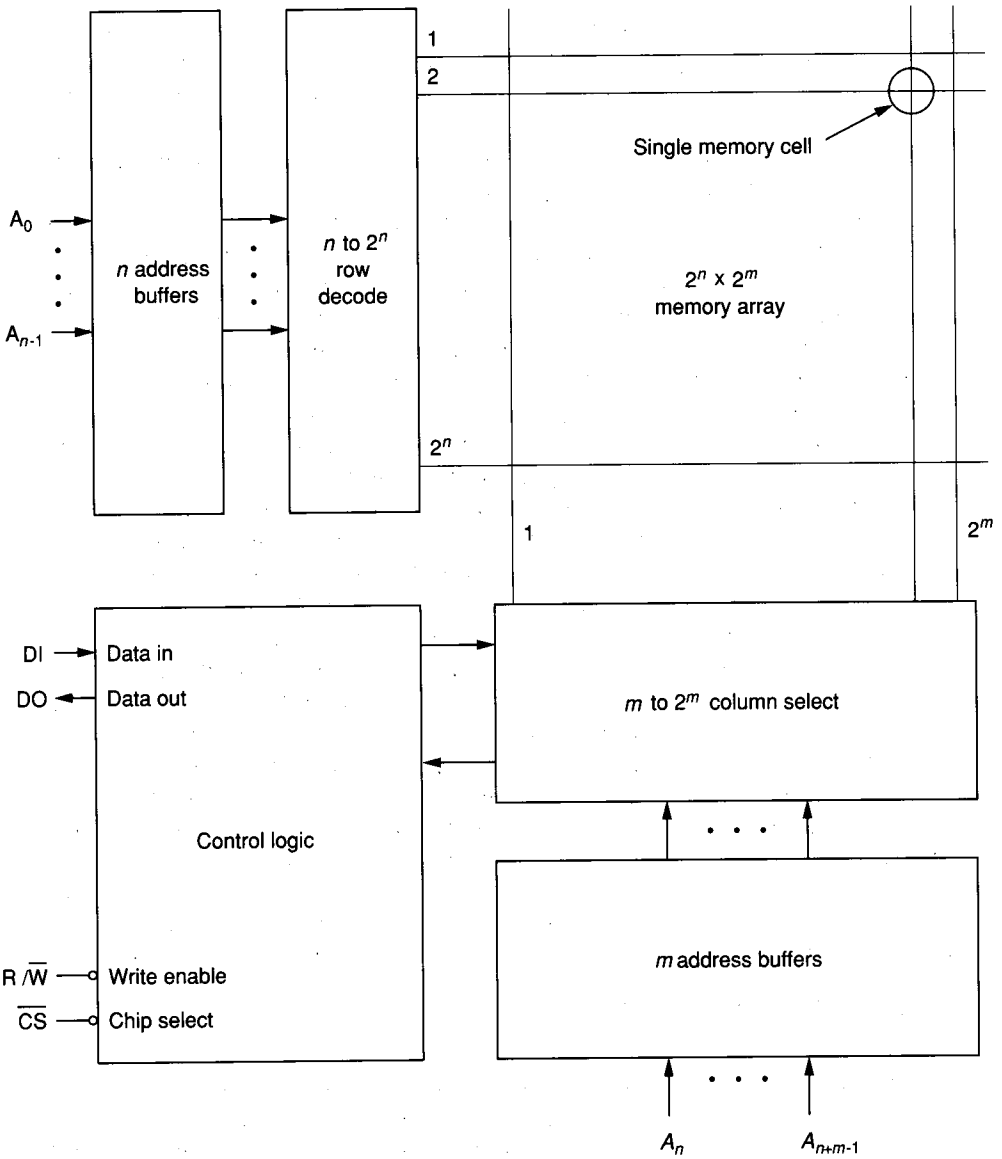

FIGURE 9.8-1
Typical memory chip architecture.

access memory), and DRAM (dynamic random access memory). These memory types derive their unique characteristics and advantages from the basic storage cell used in each, although the associated support circuitry will also vary. ROM-style devices will be discussed in the next section, and SRAM and DRAM memories will be addressed in two subsequent sections. Register array memories will be described in a fourth section.

9.9 READ-ONLY MEMORY

Read-only memory is the densest form of random-access semiconductor memory, using the presence or absence of a single transistor as the storage mechanism. Figure 9.9-1 shows the relationship between storage cells and the row and column lines of the memory matrix. Part of the memory address is decoded to select an individual row line and drive it to a positive voltage. Previously, each of the column lines had to be pulled to a high level. Each storage cell transistor has its gate connected to a row line, its drain connected to a column line, and its source grounded as shown by Fig. 9.9-1. Only if a transistor is placed where the selected row (corresponding to the row address) crosses a column will that column be pulled to a low level. Thus, each column line will be high or low to reflect the stored data along the selected row line. The remainder of the address lines are decoded to select the desired column or columns to provide the requested data.

The ROM memory contents are programmed by selectively placing transistors within the memory array. Therefore, the contents of a ROM memory are fixed as the part is manufactured and cannot be changed at a later time. In mass production, ROMs are the least expensive form of semiconductor memory; however, each unique ROM incurs relatively expensive start-up costs. As a result,

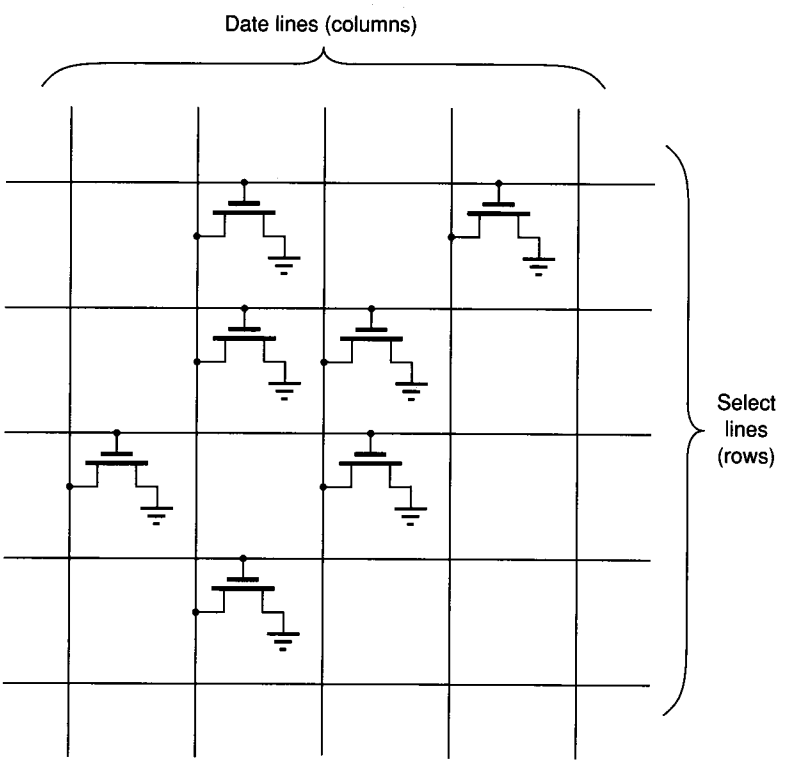

FIGURE 9.9-1
ROM memory array.

ROMs are normally feasible only for applications that require a minimum of several thousand identical memory parts with permanently stored data.

Each storage cell of a ROM may contain a single enhancement transistor. The size of the cell is set by the area required for the transistor and its associated row and column lines. An area of about 16 μ^2 per storage cell is required for today's ROMs. The memory array, requiring only enhancement transistors, can be fabricated in either NMOS or CMOS technology. The peripheral circuitry such as sense amplifiers, decoders, and control logic can also be fabricated in either technology. However, CMOS is usually chosen for newer ROMs to reduce static power dissipation of the peripheral circuitry.

9.9.1 Erasable Programmable Read-Only Memory

Many applications require semiconductor memory that is nonvolatile like ROM, yet can be reprogrammed to correct unintentional errors in the contents of the memory or to change program-based characteristics of a system. A *nonvolatile* memory is one that retains its stored data even while power is off. The ROM just described is nonvolatile and cannot be changed once it is manufactured. Other popular forms of semiconductor memory, such as SRAMs and DRAMs, have read/write capability but are *volatile*—that is, their contents are lost when power is lost. As a solution to this dilemma, the EPROM (erasable programmable ROM) was developed. An EPROM provides dense, nonvolatile storage yet can be reprogrammed as necessary. As a result, these memories are widely used in microprocessor systems and other circuits requiring nonvolatile storage where the cost of a unique ROM device cannot be justified.

The EPROM memories acquire their useful characteristics through use of a unique storage cell. This cell was originated by Intel Corporation and was called the *FAMOS* technology (for floating-gate, avalanche-injection, metal oxide semiconductor). Figure 9.9-2 shows that the storage cell consists of a transistor

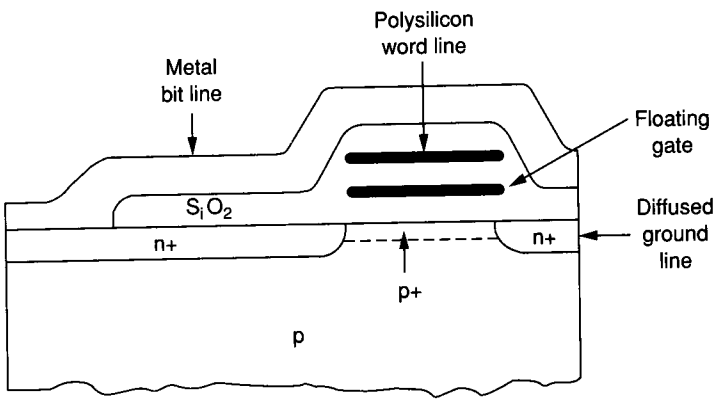

FIGURE 9.9-2
6×6 μ^2 EPROM storage cell.

with two gates, one of which is isolated from the circuit. If this floating gate somehow acquires charge to represent a stored data value, the charge remains there for a long period of time because the gate is insulated from its surroundings by silicon dioxide. The leakage paths for this circuit are of such high resistance that the time constant for the discharge path of the EPROM memory cell is tens of years. The presence or absence of stored charge is the mechanism by which the cell stores a data value, but how is the data value changed once it has been stored?

The mechanism for programming new data comes from the second part of the FAMOS name, *avalanche injection*. If a relatively high voltage (about 25 V—less for newer parts) is applied to the floating gate–substrate region, avalanche injection of electrons onto the gate takes place. This phenomenon serves to program memory by placing charge on the gate. Memory is erased by removing undesired charge from a gate as follows. If the floating gate is exposed to strong light of the proper wavelength (UV–2537 Å) for a period of time, enough energy is imparted to the stored charge to remove it from the floating gate. A quartz window is incorporated into the memory chip package; thus, all memory cells are exposed and erased simultaneously. Typical erase times for EPROMs are in the range of 20 to 30 minutes.

EPROMs are widely used in electronic systems where product volume is insufficient to justify the high initial cost of ROM parts. They are also used in prototyping microprocessor systems where reprogrammable but nonvolatile memory is required. A disadvantage of EPROM memory is the usual need to remove the memory part from the system if it becomes necessary to erase and reprogram the EPROM. The next section describes an improvement to the EPROM that was designed to overcome this undesirable characteristic.

9.9.2 Electrically Erasable Programmable Read-Only Memory

As useful as EPROM memories are, there are many applications requiring nonvolatile memory that can be reprogrammed quickly without removing the memory part from the system. For example, it is often desirable to program a standard CRT terminal with serial interface characteristics that will not be lost when power is disconnected. Yet, these characteristics must be changeable if the terminal is connected to a computer using a different serial data format. Applications such as this created a need for an EPROM whose contents could be changed electrically. Several manufacturers now offer memory parts called EAPROMs (electrically alterable) or EEPROMs (electrically erasable) that fulfill this need.

Interestingly, the EEPROMs solve the programming and erasure problems with two simple techniques. First, programming has been simplified by on-chip generation of the programming voltage. Instead of requiring an external connection to 25 V, a circuit called a *charge pump* is used to generate the necessary programming voltage from the standard 5 V supply. Second, instead of using ultraviolet light to erase the data, an internal connection is provided to reverse the electron injection phenomenon, allowing charge to be removed from the floating gate of the EEPROM.

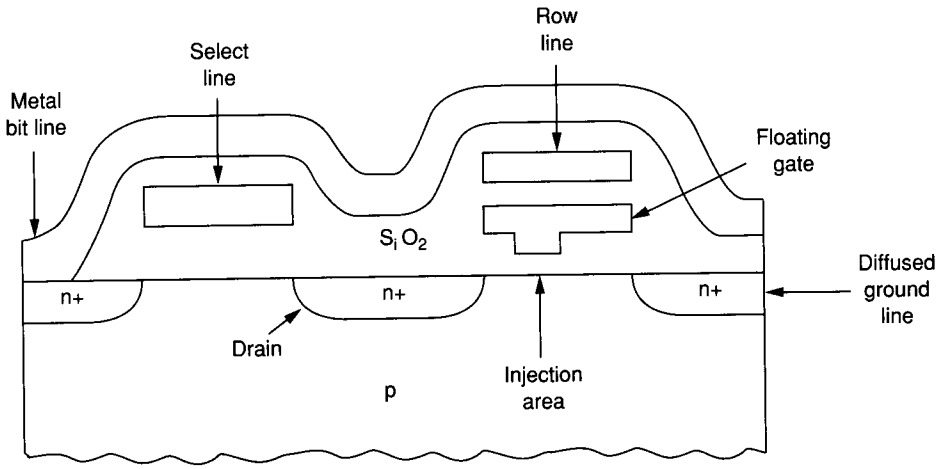

FIGURE 9.9-3
EEPROM storage cell.

The basic memory cell for an EEPROM consists of a memory transistor and a select transistor, as shown in Fig. 9.9-3. The memory transistor is composed of a dual-stacked polysilicon structure in which the bottom gate is floating. A small, thin oxide (<150 Å) isolates the floating gate from the drain and provides an injection area for electrons to and from the floating gate. Sending a short (several ms), high-voltage pulse to the row line while grounding the drain causes tunneling of electrons from the drain to the floating gate (erase). A similar pulse to the drain with the row line grounded causes tunneling of electrons from the floating gate to the drain (write). Integrating this device into a memory array requires an additional select transistor per bit as shown in Fig. 9.9-3 to avoid disturbing unwanted cells during erase or write. As a result, EEPROM memory is not as dense as EPROM memory.

The semiconductor memories described in this section each use the same basic array organization to achieve dense, nonvolatile storage. Ultimately, the storage capacity of these memory parts depends on the size of the basic memory cell used as the storage mechanism. The ROM is very dense, using only a single transistor as a storage cell; the EPROM is less dense, using a single transistor with a select gate and a floating gate as a storage cell; and the EEPROM is the least dense, requiring two transistors for each storage cell.

9.10 STATIC RAM MEMORIES

In this section the basic memory cell and organization used in static semiconductor memory circuits is examined as another example of a highly successful form of structured design. Static random-access memories, or SRAMs, are composed of static storage cells as the basic storage mechanism. Each static storage cell is formed by a pair of cross-coupled inverters. For SRAM memory, many of these cells are formed into a large memory array organized as explained in Sec. 9.8.

SRAMs differ from ROMs because they need continuous power to maintain the feedback required to hold a stored value. If power is lost, the active feedback path is eliminated, and the memory contents are destroyed. As power is restored, the memory cell will settle to a logic value that is independent of the previously stored data. Thus, the SRAM is classed as a volatile memory and depends on continuous power to maintain stored values.

For SRAMs to be useful as read/write memories, it must be possible to store desired data values in each cell. The basic memory organization must allow selection of each memory cell and accessing or storing binary values within that cell. A common structure consisting of select lines and data access lines for a SRAM memory cell is shown in Fig. 9.10-1. This figure shows a CMOS storage cell; a corresponding circuit structure is used for NMOS storage cells. A unique

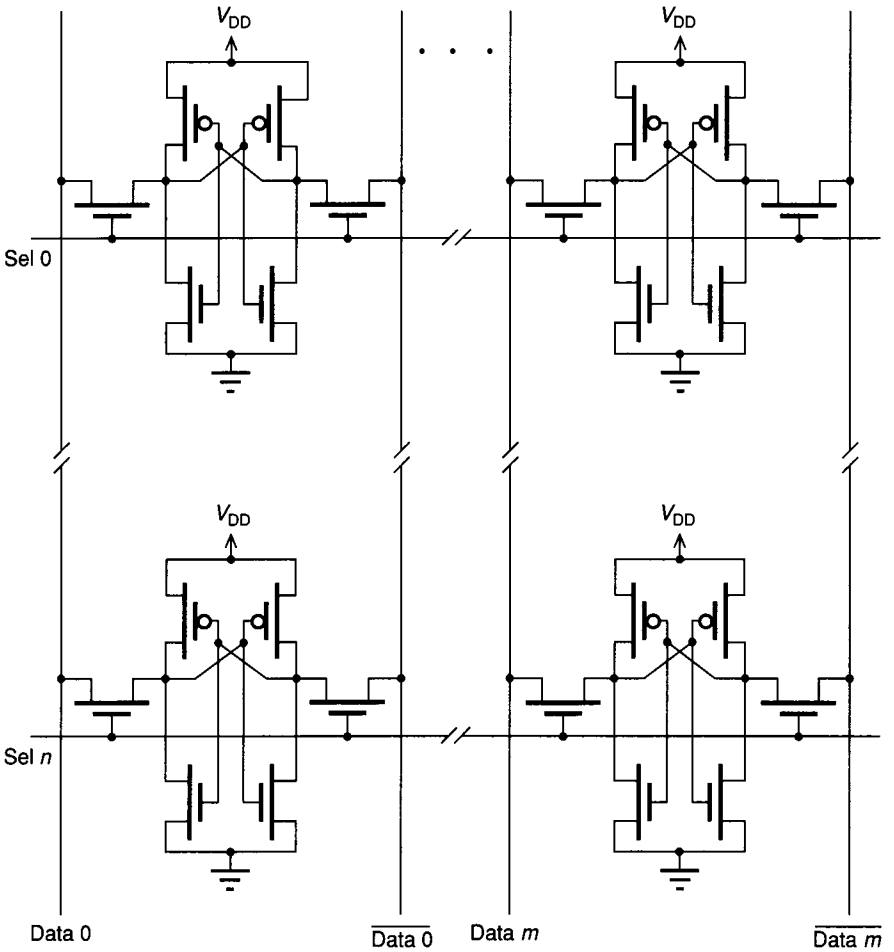

FIGURE 9.10-1
SRAM storage cell array.

select line is provided for each row of the memory array and, when high, selects all memory cells on its row. A pair of data lines are dedicated to each column to allow data from a selected cell on that column to be read or written. During a memory read, the two lines of each data line pair will be forced to opposite logic states by the contents of the selected memory cell. From a logical viewpoint, only one data line is needed to access the data within a memory cell. However, for a memory write, the two lines of the selected data line pair must be driven to opposite logic states to store a desired value within the memory cell. Because one cell along every column is selected by the row select, only data line pairs corresponding to the column containing the desired cell must be driven during a write. Other data line pairs along the row select will perform read operations.

As shown in the memory chip architecture of Fig. 9.8-1, n address lines are decoded to generate 2^n row select lines. The row select lines can be generated by the NOR decoder of Fig. 9.10-2, where $n = 3$. Here, the horizontal row lines are each pulled high by a pullup device. The pullup device could be a depletion transistor, a p-channel transistor with its gate grounded, or a clocked p-channel transistor, depending on the technology. The vertical address lines are converted

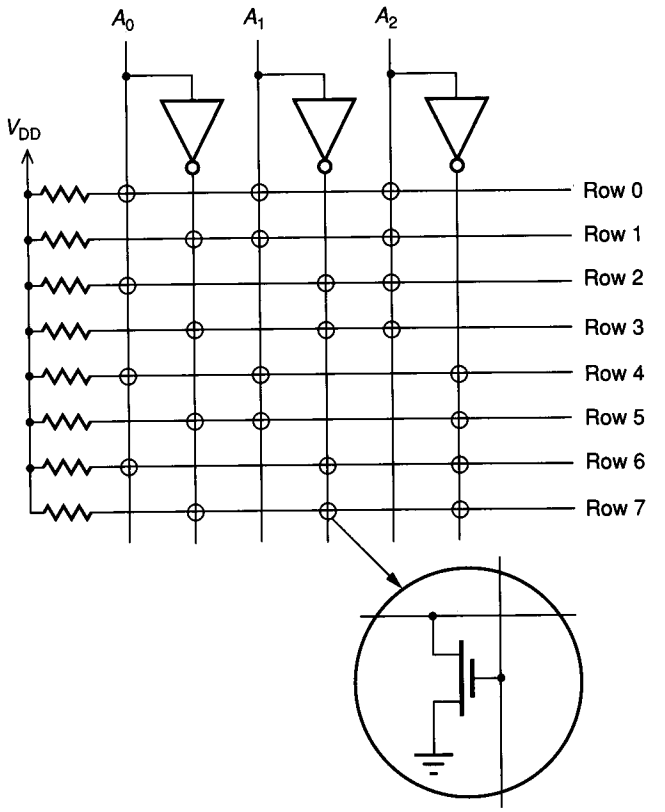

FIGURE 9.10-2
Address-to-row-select decoder.

to double-rail form and are used as the gates of pulldown transistors in the NOR structure. For example, row 0 of Fig. 9.10-2 is the logical NOR of A0, A1, and A2. If any of the three address lines is high, then the row 0 select line is pulled low. Only if A0, A1, and A2 are all low will the row 0 line be left high. Thus, the condition to select the row 0 memory cells is that A0, A1, and A2 must be low. To quickly charge the large capacitance inherent in the row select lines, a buffer stage (not shown) is usually inserted between the row decoder and the memory array.

Once a given row is selected, data from all memory cells on that row are available on the column data lines. The memory chip architecture of Fig. 9.8-1 shows that m address lines are used to select 1 of 2^m columns to access the desired data. Figure 9.10-3 shows an address-to-column selector circuit with $m = 3$ that gates one of eight columns to the data output. Address lines A3, A4, and A5 are used in a select array to choose the desired column. A3 selects the odd columns, while the complement of A3 selects the even columns. After the A3 stage, odd and even columns are paired because only one of the pair can be active. This reduces the number of available column paths by one-half. Subsequent stages of selection each divide the number of column paths by two. After m select stages, where 2^m is the number of memory columns, only a single column line is left. This line contains the desired data value. The data value must be buffered to drive the data output pin on the memory chip. To reduce the time between selection of a memory cell and availability of data at the chip output, sensitive

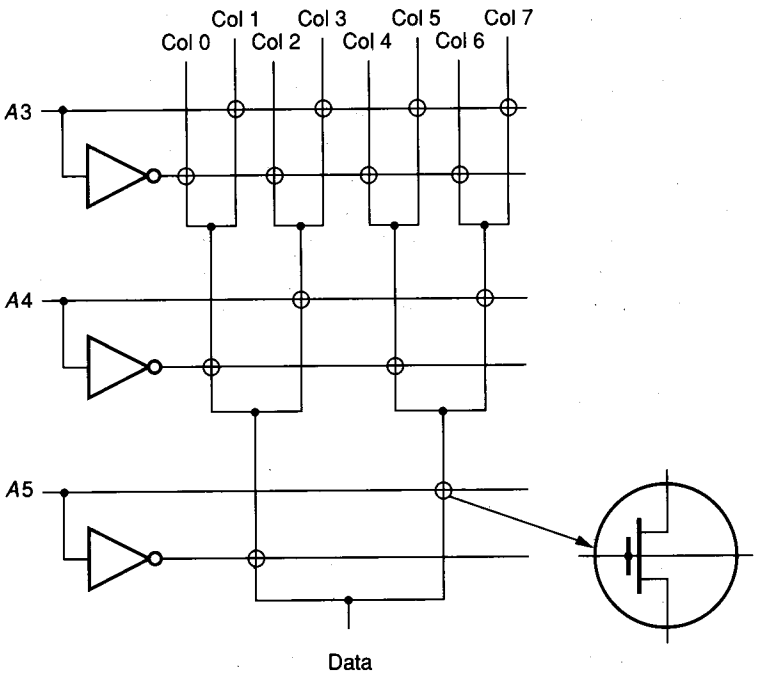

FIGURE 9.10-3
Address-to-column selector.

sense amplifiers (not shown) are included within the column select structure. The purpose of these amplifiers is early detection of the data value as the memory cell drives the large capacitance of its column line pair.

Electrical characteristics of the memory cells, data lines, and select lines are particularly important when a large array of memory storage cells must be operated at high speed. The select lines cross an entire array of memory cells where they drive the gate terminals for $2c$ transistors for a c-wide array. These lines are normally run in polysilicon instead of metal because they can directly gate the select transistors without a space-consuming contact at each select transistor. The metal level is reserved for the data lines. The length (and therefore area) of the polysilicon select line and the many gates to be driven provide a highly capacitive load for the select line drivers. As a result, careful consideration must be given to the delay caused by this capacitive load when the select line buffers are designed.

Example 9.10-1. Select line delay calculation Assume a 16k × 1-bit SRAM memory is organized as a square array of memory cells with 128 cells on a side. Further, assume that the actual memory array is 2 mm × 2 mm, the polysilicon select line is 2 μ wide, and the select transistor gates are $2\mu \times 2\mu$. The select line is driven from one end. The polysilicon select line resistance is 22 Ω per square. Capacitance to substrate for the polysilicon is 0.08 fF/μ^2 and gate capacitance is 1 fF/μ^2. Estimate the select line delay as the approximate 10% to 90% rise time at the far end assuming that the select line is driven by an ideal voltage step.

Gross solution. First consider a simple solution with the total select line resistance and capacitance represented as a low-pass *RC* filter. The resistance can be found by calculating the number of squares from one end of the select line to the other and multiplying by the resistance per square for polysilicon. A 2 mm × 2 μ line is 1000 squares long. The capacitance can be calculated from the area of the polysilicon select line and the area of gates for the select transistors. The area of the select line less the gate area is (2 mm−256 × 2 μ) × 2 μ or 2976 μ^2. The total area of the gates will be 2 × 128 × 4 μ^2 or 1024 μ^2.

$$C_T = 2976\ \mu^2 \times 0.08\ \text{fF}/\mu^2 + 1024\ \mu^2 \times 1\ \text{fF}/\mu^2$$

$$C_T = 0.238\ \text{pF} + 1.024\ \text{pF} = 1.262\ \text{pF}$$

$$t_{10\%} = 0.105\tau = 0.105 \times 1.262\ \text{pF} \times 22{,}000\ \Omega = 2.92\ \text{ns}$$

$$t_{90\%} = 2.303\tau = 2.303 \times 1.262\ \text{pF} \times 22{,}000\ \Omega = 63.94\ \text{ns}$$

$$t_d = t_{90\%} - t_{10\%} = 63.94\ \text{ns} - 2.92\ \text{ns} = 61.02\ \text{ns}$$

Distributed lumped-parameter solution. For this solution, break the polysilicon line into 10 segments with the resistance and capacitance divided equally among the 10 segments. A SPICE simulation (refer to Chapter 4) can be used to find the 10% to 90% delay times. The results are given below.

$$t_{10\%\ \text{voltage}} = 4.39\ \text{ns}$$

$$t_{90\%\ \text{voltage}} = 33.77\ \text{ns}$$

$$t_d = t_{90\%} - t_{10\%} = 33.77\ \text{ns} - 4.39\ \text{ns} = 29.38\ \text{ns}$$

The lumped parameter solution with 10 equal segments is very close to the exact solution. The simple *RC* method presented first can be used to give a worst-case estimate of the delay at the end of the select line. (It more than doubles the actual delay in this example.) In fact, it has been shown that an estimate of half the simple *RC* delay is a good approximation for a distributed *RC* line.[16]

The delay characteristics of the data lines are also of concern because they traverse the entire memory array in the vertical direction, providing a large capacitive load. These lines are usually run in metal rather than diffusion because they must cross the polysilicon select lines. Unfortunately, during a read operation the memory cells themselves must drive the data line capacitance. Special line-driver circuits are not available to overcome this speed limitation because their size prohibits providing a line driver for each memory cell. The memory cells are designed for minimum size to increase the overall memory density, and they cannot provide good capacitive drive characteristics. As may be seen from Fig. 9.10-1 the selected SRAM storage cell must drive the complementary data lines of the cell column in opposite directions. Thus, the limiting delay condition is for the data line that must be driven high where the memory cell p-channel pullup device must charge the data line capacitance. For NMOS cells, if the pullup transistor provides a low resistance path to V_{DD}, the memory array will dissipate an undesirably large amount of power because one inverter of each memory cell always conducts. Providing a high resistance for the pullup transistors would cause an unacceptable delay in charging data line capacitance. A typical resolution of this conflict is presented next.

A common method for minimizing the pullup time for a highly capacitive line driven by ratio-type circuits is to pull the line to a voltage above the logic threshold voltage when it is not in use. This technique is called *precharge* because the line is precharged to a value at or near the high logic condition. For this scheme to work, it must be possible to isolate the line from any driving sources during the precharge time. Such sources can be isolated from the line with pass transistors as they are in a memory array. When the capacitive line is to reflect the logic condition of a driving circuit, the corresponding pass transistor is turned on. If the driving circuit has a high voltage output and the line has been precharged to a high level, the correct logic output is available immediately. If the driving circuit has a low logic output, the pulldown transistor must discharge the output line before the correct logic value is available. Because the resistance of the pulldown transistor has little effect on power dissipation, the driving circuit can have a low-resistance pulldown transistor, allowing the discharge time to be shortened. The use of precharged lines provides a means of bypassing the asymmetric drive characteristics of a ratio logic output stage for situations where a high-capacitance line must be driven. The precharge scheme is often used for buses that must be driven by many sources.

Example 9.10-2. Optimum precharge voltage for data lines Consider that an optimum precharge voltage level might exist for a capacitive line driven by ratio logic. That voltage level would equalize the charge and discharge times of the output line for a given driving circuit with a pullup/pulldown ratio R_u/R_d. Also, assume