CHAPTER 8

Semiconductor Memory Design

CHAPTER OUTLINE

8.1 Introduction
8.2 MOS Decoders
8.3 Static RAM Cell Design
8.4 SRAM Column I/O Circuitry
8.5 Memory Architecture
8.6 Summary

References
Problems

8.1 Introduction

Modern digital systems require the capability of storing and retrieving large amounts of information at high speeds. *Memories* are circuits or systems that store digital information in large quantity. This chapter addresses the analysis and design of VLSI memories, commonly known as *semiconductor memories*. Today, memory circuits come in different forms including SRAM, DRAM, ROM, EPROM, E^2PROM, Flash, and FRAM. While each form has a different cell design, the basic structure, organization, and access mechanisms are largely the same. In this chapter, we classify the different types of memory, examine the major subsystems, and focus on the static RAM design issues. This topic is particularly suitable for our study of CMOS digital design as it allows us to apply many of the concepts presented in earlier chapters.

Recent surveys indicate that roughly 30% of the worldwide semiconductor business is due to memory chips. Over the years, technology advances have been driven by memory designs of higher and higher density. Electronic memory capacity in digital systems ranges from fewer than 100 bits for a simple function to standalone chips containing 256 Mb (1 Mb = 2^{10} bits) or more.[1] Circuit designers usually

[1] Recently, a memory chip with 1 Gbit of data storage capacity has been announced.

speak of memory capacities in terms of bits, since a separate flip-flop or other similar circuit is used to store each bit. On the other hand, system designers usually state memory capacities in terms of *bytes* (8 bits); each byte represents a single alphanumeric character. Very large scientific computing systems often have memory capacity stated in terms of *words* (32 to 128 bits). Each byte or word is stored in a particular location that is identified by a unique numeric *address*. Memory storage capacity is usually stated in units of kilobytes (K bytes) or megabytes (M bytes). Because memory addressing is based on binary codes, capacities that are integral powers of 2 are most common. Thus the convention is that, for example, 1K byte = 1,024 bytes and 64K bytes = 65,536 bytes. In most memory systems, only a single byte or word at a single address is stored or retrieved during each cycle of memory operation. *Dual-port* memories are also available that have the ability to read/write two words in one cycle.

8.1.1 Memory Organization

The preferred organization for most large memories is shown in Figure 8.1. This organization is a *random-access* architecture. The name is derived from the fact that memory locations (*addresses*) can be accessed in random order at a fixed rate, independent of physical location, for reading or writing. The storage array, or *core*, is made up of simple cell circuits arranged to share connections in horizontal rows and vertical columns. The horizontal lines, which are driven only from outside the storage array, are called *wordlines*, while the vertical lines, along which data flow into and out of cells, are called *bitlines*.

A cell is accessed for reading or writing by selecting its row and column. Each cell can store 0 or 1. Memories may simultaneously select 4, 8, 16, 32, or 64 columns

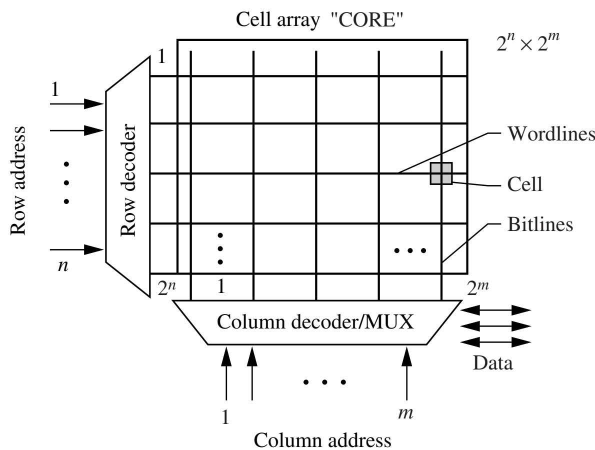


Figure 8.1
Organization of memory systems.

in one row depending on the application. The row and column (or group of columns) to be selected are determined by decoding binary address information. For example, an n-bit *decoder* for row selection, as shown in Figure 8.1, has 2^n output lines, a different one of which is enabled for each different n-bit input code. The column decoder takes m inputs and produces 2^m bitline access signals, of which 1, 4, 8,16, 32, or 64 may be enabled at one time. The bit selection is done using a multiplexer circuit to direct the corresponding cell outputs to data registers. In total, $2^n \times 2^m$ cells are stored in the core array.

An overall architecture of a 64 Kb random-access memory is shown in Figure 8.2. For this example, $n = m = 8$. Therefore, the core array has a total of 65,536 cells. The memory uses a 16-bit address to produce a single bit output.

Memory cell circuits can be implemented in a wide variety of ways. In principle, the cells can be based on the flip-flop designs listed in Chapter 5 since their intended function is to store bits of data. However, these flip-flops require a substantial amount of area and are not appropriate when millions of cells are needed.

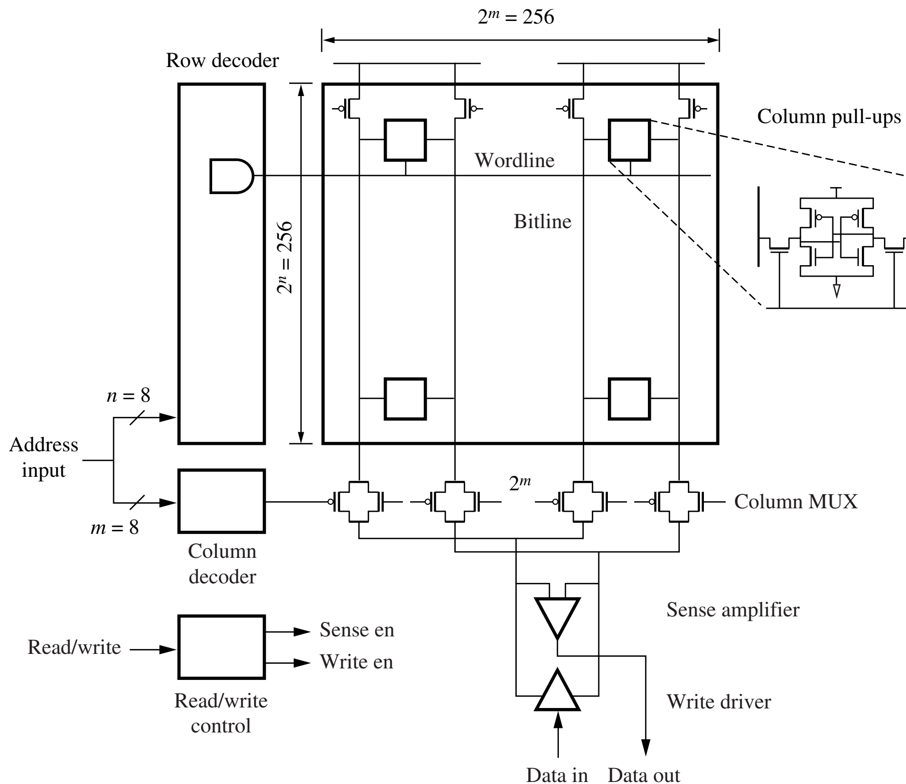


Figure 8.2
Overall architecture of memory design.

In fact, most memory cell circuits are greatly simplified compared to register and flip-flop circuits. While the data storage function is preserved, other properties including quantization of amplitudes, regeneration of logic levels, input-output isolation, and fanout drive capability may be sacrificed for cell simplicity. In this way, the number of devices in a single cell can be reduced to one to six transistors. Figure 8.2 illustrates a six-transistor memory cell.

At the level of a memory chip shown in Figure 8.2, the desired logic properties are recovered through use of properly designed *peripheral circuits*. Circuits in this category are the decoders, sense amplifiers, column precharge, data buffers, etc. These circuits are designed so that they may be shared among many memory cells. Read-write (R/W) circuits determine whether data are being retrieved or stored, and they perform any necessary amplification, buffering, and translation of voltage levels. Specific examples are presented in the following sections.

8.1.2 Types of Memory

Read-write random-access memories (RAM) may store information in flip-flop style circuits, or simply as charge on capacitors. Approximately equal delays are encountered in reading or writing data. Because read-write memories store data in active circuits, they are *volatile*; that is, stored information is lost if the power supply is interrupted. The natural abbreviation for read-write memory would be RWM. However, pronunciation of this acronym is difficult. Instead, the term RAM is commonly used to refer to read-write random-access memories. If the terms were consistent, both read-only (see below) and read-write memories would be called RAMs.

The two most common types of RAMs are the static RAM (SRAM) and the dynamic RAM (DRAM). The static and dynamic definitions are based on the same concepts as those introduced in earlier chapters. Static RAMs hold the stored value in flip-flop circuits as long as the power is on. SRAMs tend to be high-speed memories with clock cycles in the range of 5 to 50 ns. Dynamic RAMs store values on capacitors. They are prone to noise and leakage problems, and are slower than SRAMs, clocking at 50 ns to 200 ns. However, DRAMs are much more dense than SRAMs—up to four times more dense in a given generation of technology.

Read-only memories (ROMs) store information according to the presence or absence of transistors joining rows to columns. ROMs also employ the organization shown in Figure 8.1 and have read speeds comparable to those for read-write memories. All ROMs are *nonvolatile*, but they vary in the method used to enter (write) stored data. The simplest form of ROM is programmed when it is manufactured by formation of physical patterns on the chip; subsequent changes of stored data are impossible. These are termed *mask-programmed* ROMs. In contrast, *programmable* read-only memories (PROMs) have a data path present between every row and column when manufactured, corresponding to a stored 1 in every data position. Storage cells are selectively switched to the 0 state once after manufacture by applying appropriate electrical pulses to selectively open (blow out) row-column data paths. Once programmed, or *blown*, a 0 cannot be changed back to a 1.

Erasable programmable read-only memories (EPROMs) also have all bits initially in one binary state. They are programmed electrically (similar to the PROM), but all bits may be erased (returned to the initial state) by exposure to ultraviolet (UV) light. The packages for these components have transparent windows over the chip to permit the UV irradiation. *Electrically erasable* programmable read-only memories (EEPROMs, E^2PROM, or E-squared PROMs) may be written and erased by electrical means. These are the most advanced and most expensive form of PROM. Unlike EPROMs, which must be totally erased and rewritten to change even a single bit, E^2PROMs may be selectively erased. Writing and erasing operations for all PROMs require times ranging from microseconds to milliseconds. However, all PROMs retain stored data when power is turned off; thus they are termed nonvolatile.

A recent form of EPROM and E^2PROM is termed Flash memory, a name derived from the fact that blocks of memory may be erased simultaneously. Flash memory of the EPROM form is written using the hot-electron effect[2] whereas E^2PROM Flash is written using Fowler-Nordheim (FN) tunneling.[3] Both types are erased using FN tunneling. Their large storage capacity has made this an emerging mass storage medium. In addition, these types of memories are beginning to replace the role of ROMs on many chips, although additional processing is required to manufacture Flash memories in a standard CMOS technology.

Memories based on ferroelectric materials, so-called FRAMs or FeRAMs, can also be designed to retain stored information when power is off. The *Perovskite crystal* material used in the memory cells of this type of RAM can be polarized in one direction or the other to store the desired value. The polarization is retained even when the power supply is removed, thereby creating a nonvolatile memory. However, semiconductor memories are preferred over ferroelectric memories for most applications because of their advantages in cost, operating speed, and physical size. Recently, FRAMs have been shown to be useful nonvolatile memory in certain applications such as smart cards and may be more attractive in the future due to their extremely high storage density.

8.1.3 Memory Timing Parameters

The timing signals of random-access memories are illustrated in Figure 8.3. At a high level, the main inputs are the $(n + m)$-bit address, the k-bit input data, and the *read/write* signal (write operations are active low in this example). The output is the k-bit value that was stored in the memory location associated with the $(n + m)$-bit address. The *read access time*, t_{AC}, is the delay from presentation of an address until data stored at that address are available at the output. Maximum read access time is an important parameter, and it should not exceed the memory cycle time since

[2] Hot electrons are created by applying a high field in the channel region. These electrons enter the oxide and raise the threshold voltage of a device. Devices with this higher threshold voltage are viewed as a stored "1." Devices with the lower threshold voltage represent a stored "0."

[3] Fowler-Nordheim tunneling occurs through thin insulating material such as thin-oxide associated with the gate. Current flows through the oxide by tunneling through the energy barrier.

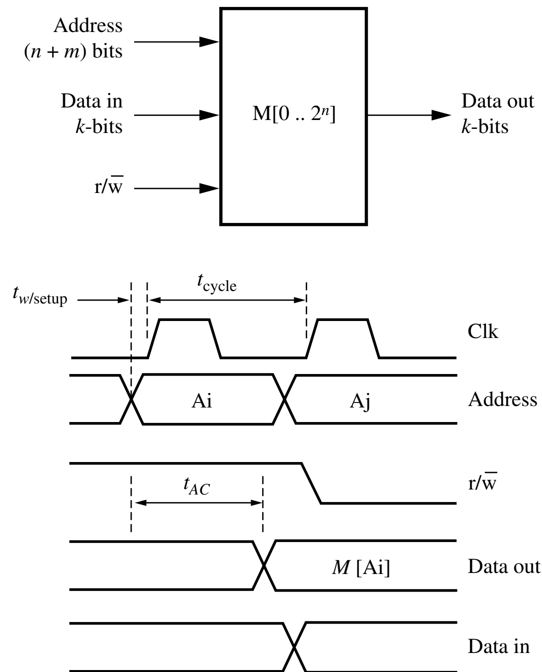


Figure 8.3
Random access memory timing parameters.

there are write setup operations needed before each memory operation, indicated by $t_{w/\text{setup}}$. The memory clock cycle time, t_{cycle}, is the minimum time needed to complete successive read or write operations.

The cycle time is essentially the reciprocal of the time rate at which address information is changed while reading or writing at random locations. Minimum access times for reading and writing are not necessarily the same, but for simplicity of design, most systems specify a single time for both reading and writing. For semiconductor read-write memories, the read access time is typically 50 to 80% of cycle time.

8.2 MOS Decoders

The row and column decoders identified in Figure 8.1 are essential elements in all random-access memories. Access time and power consumption of memories may be largely determined by decoder design. Similar designs are used in read-only and read-write applications. Row decoders take an n-bit address and produce 2^n outputs, one of which is activated. Obviously, the one that is activated depends directly on the address applied to the memory. Figure 8.4 shows two forms of decoders: AND and NOR decoders. The decoder can be implemented using AND gates or

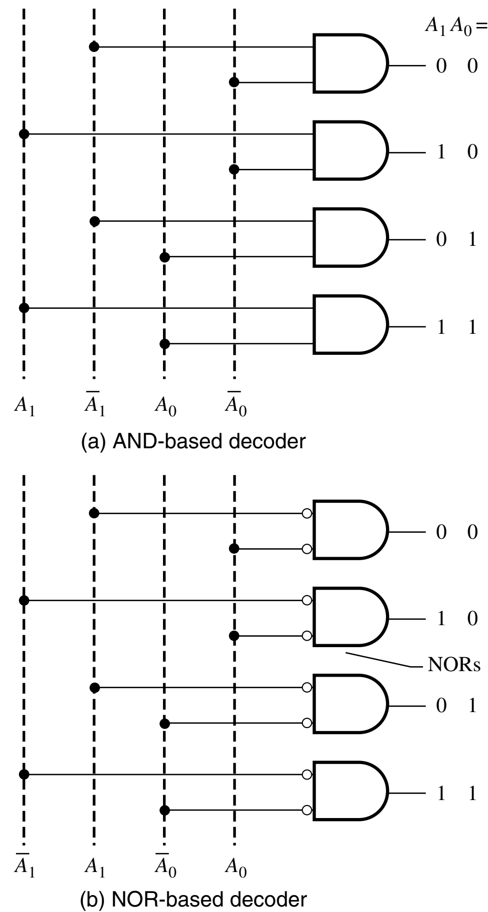


Figure 8.4
AND and NOR decoders.

NOR gates that take every possible combination of the inputs. There are two address bits in this example and we require both the true and complement of each address bit. The output line activated by each input combination is shown in the figure. Note that all outputs are normally low except one.

An n-bit decoder requires 2^n logic gates, each with n inputs. For example, with $n = 6$, we need 64 NAND6 gates driving 64 inverters to implement the decoder. From previous chapters, it is clear that gates with more than 3 or 4 inputs create large series resistances and long delays. Rather than using n-input gates, it is preferable to use a cascade of gates. Typically two stages are used: a predecode stage and a final decode stage. The predecode stage generates intermediate signals that are used by multiple gates in the final decode stage. In Figure 8.5, schematics are shown for two possible alternatives to implement a 6-input AND gate. We could choose three

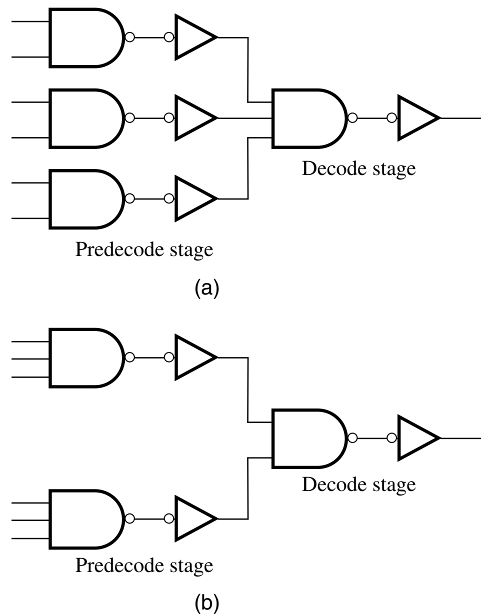


Figure 8.5
Predecoder configurations.

NAND2 gates and one NAND3 gate to implement each AND6 gate, as shown in Figure 8.5a. Alternatively, two NAND3 gates and one NAND2 gate of Figure 8.5b may be used. The better of the two can be determined using logical effort.

The main advantage of two-level decoding is that a large number of intermediate signals can be generated by the predecode stage and then reused by the final decoding stage. The result is a reduction in the number of inputs for each gate. Since this aspect is not clearly depicted in Figure 8.5, a more complete example for 6 address bits is shown in Figure 8.6. In the predecoder, a total of 12 intermediate signals are generated from the address bits and their complements. The signals that are generated in the predecoder are as follows: A_0A_1, $A_0\overline{A}_1$, $\overline{A}_0A_1$, $\overline{A}_0\overline{A}_1$, A_2A_3, $A_2\overline{A}_3$, etc. These signals may now be used by the final decoding stage to generate the 64 required outputs using NAND3/inverter combinations. This corresponds to the configuration shown in Figure 8.5a. Each predecoder output drives 16 NAND gates (i.e., 64 NAND3 gates $\times$ 3 inputs each / 12 intermediate outputs). Therefore, the branching effort $BE = 16$. The delay through the NAND2-inverter-NAND3-inverter stages can be minimized by sizing the gates using logical effort. A similar kind of two-level decoder can be constructed using the configuration of Figure 8.5b. Again, the better of the two approaches can be determined using logical effort. It is important to minimize the delay through the decoder as it may constitute up to 40% of the clock cycle.

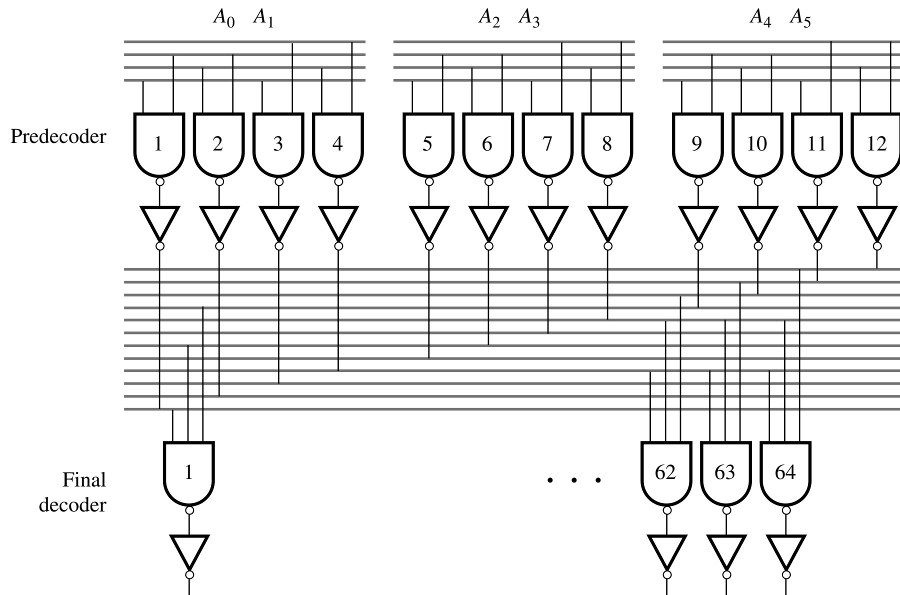


Figure 8.6
Structure of two-level decoder for 6-bit address.

Decoder Sizing Using Logical Effort Techniques **Example 8.1**

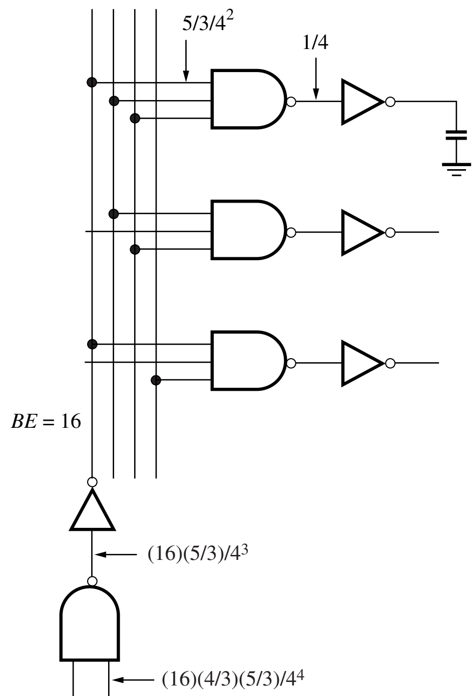
Problem:

Size the decoder of Figure 8.6 using FO4 rules assuming that the normalized output loading is 1. FO4 rules imply that the optimal stage effort is equal to 4.

$$\text{For each stage, } C_{in} = \frac{LE \times BE \times C_{out}}{4}$$

Solution:

Work backwards from the output to the input, taking into account logical effort and branching effort of each stage. First, the normalized output has a size of 1. Therefore, the inverter input capacitance is 1/4 using FO4 sizing rules. The *LE* of the NAND3 gate is 5/3. Therefore, its input capacitance is $(5/3)/4^2$. The branching effort at the output of the predecode stage is 16. Therefore, the input capacitance of the inverter is $(16)(5/3)/4^3$. Finally, the NAND2 gate has an input capacitance of $(16)(4/3)(5/3)/4^4$.



8.3 Static RAM Cell Design

Memories are said to be static if no periodic clock signals are required to retain stored data indefinitely. Memory cells in these circuits have a direct path to V_{DD} or Gnd or both. Read-write memory cell arrays based on flip-flop circuits are commonly referred to as static RAMs or SRAMs. The design issues for static memory cells are described in this section.

8.3.1 Static Memory Operation

The basic static RAM cell is shown in Figure 8.7a. It consists of two cross-coupled inverters and two access transistors. The access transistors are connected to the wordline at their respective gate terminals, and the bitlines at their source/drain terminals. The wordline is used to select the cell while the bitlines are used to perform read or write operations on the cell. Internally, the cell holds the stored value on one side and its complement on the other side. For reference purposes, assume that node q holds the stored value while node $\overline{q}$ holds its complement. The two complementary bitlines are used to improve speed and noise rejection properties, as described later in this chapter.

The VTC of cross-coupled inverters is shown in Figure 8.7b. This type of VTC was already described in Chapter 4 on single source noise margins, and in Chapter 5

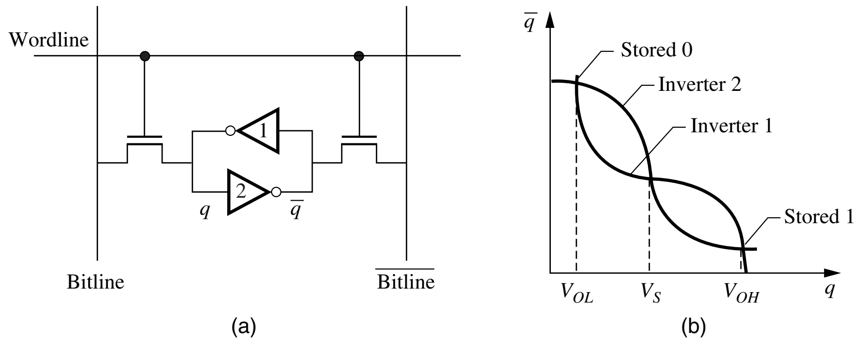


Figure 8.7
Basic SRAM cell and VTC.

on flip-flops. The VTC conveys the key cell design considerations for read and write operations. In the cross-coupled configuration, the stored values are represented by the two stable states in the VTC. The cell will retain its current state until one of the internal nodes crosses the switching threshold, V_S. When this occurs, the cell will flip its internal state. Therefore, during a read operation, we must not disturb its current state, while during the write operation we must force the internal voltage to swing past V_S to change the state.

The six transistor (6T) static memory cell in CMOS technology is illustrated schematically in Figure 8.8. The cross-coupled inverters, M_1, M_5 and M_2, M_6, act as the storage element. Major design effort is directed at minimizing the cell area and power consumption so that millions of cells can be placed on a chip. The steady-state power consumption of the cell is controlled by subthreshold leakage currents, so a larger threshold voltage is often used in memory circuits. To reduce area, the cell layout is highly optimized to eliminate all wasted area. In fact, some designs replace the load devices, M_5 and M_6, with resistors formed in undoped polysilicon.

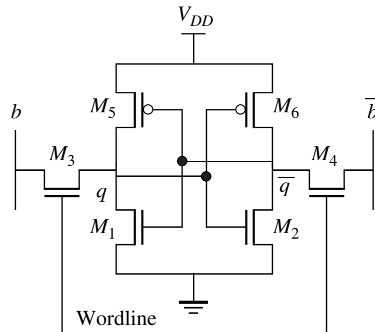


Figure 8.8
6T SRAM cell.

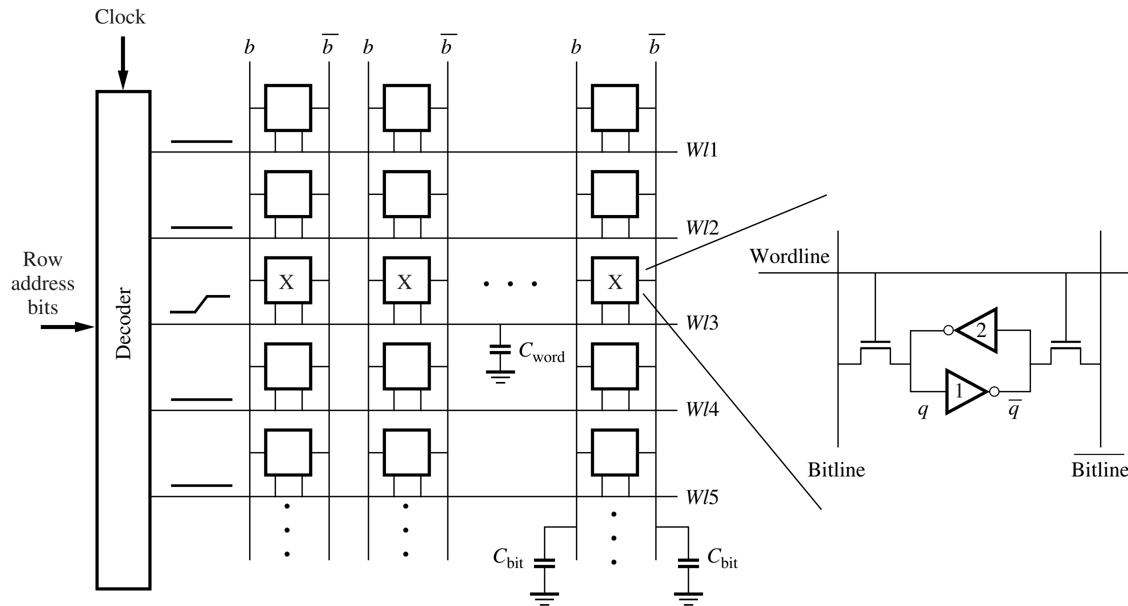


Figure 8.9
Wordline and double bitline configuration.

This is called a 4T cell since there are now only four transistors in the cell. To minimize power, the current through the resistors can be made extremely small by using very large pull-up resistances. Sheet resistance of these resistors is 10 MΩ per square or higher and the area is minimal. Standby currents are kept in the nanoampere range. Thus, power and area may be reduced at the expense of extra processing complexity to form the undoped polysilicon resistors. However, the majority of the designs today use the conventional 6T configuration of Figure 8.8.

The operation of an array of these cells is illustrated in Figure 8.9. The row select lines, or wordlines, run horizontally. All cells connected to a given wordline are accessed for reading or writing. The cells are connected vertically to the bitlines using the pair of access devices to provide a switchable path for data into and out of the cell. Two column lines, b and $\bar{b}$, provide a differential data path. In principal, it should be possible to achieve all memory functions using only one column line and one access device. Attempts have been made in this direction, but due to normal variations in device parameters and operating conditions, it is difficult to obtain reliable operation at full speed using a single access line. Therefore, the symmetrical data paths b and $\bar{b}$ as shown in Figure 8.9 are almost always used.

Row selection in CMOS memory is accomplished using the decoders described in the previous section. For synchronous memories, a clock signal is used in conjunction with the decoder to activate a row only when read-write operations are being performed. At other times, all wordlines are kept low. When one wordline goes high, say *wl3* in Figure 8.9, all the cells in that row are selected. The access transistors

are all turned on and a read or write operation is performed. Cells in other rows are effectively disconnected from their respective wordlines.

The wordline has a large capacitance, C_{word}, that must be driven by the decoder. It is comprised of two gate capacitances per cell and the wire capacitance per cell:

$$C_{\text{word}} = (2 \times \text{gate cap} + \text{wire cap}) \times \text{no. of cells in row} \tag{8.1}$$

Once the cells along the wordline are enabled, read or write operations are carried out. For a read operation, only one side of the cell draws current. As a result, a small differential voltage develops between b and $\bar{b}$ on all column lines. The column address decoder and multiplexer select the column lines to be accessed. The bitlines will experience a voltage difference as the selected cells discharge one of the two bitlines. This difference is amplified and sent to output buffers.

It should be noted that the bitlines also have a very large capacitance due to the large number of cells connected to them. This is primarily due to source/drain capacitance, but also has components due to wire capacitance and drain/source contacts. Typically, a contact is shared between two cells. The total bitline capacitance, C_{bit}, can be computed as follows:

$$C_{\text{bit}} = (\text{source/drain cap} + \text{wire cap} + \text{contact cap}) \times \text{no. of cells in column} \tag{8.2}$$

During a write operation, one of the bitlines is pulled low if we want to store 0, while the other one is pulled low if we want to store 1. The requirement for a successful write operation is to swing the internal voltage of the cell past the switching threshold of the corresponding inverter. Once the cell has flipped to the other state, the wordline can be reset back to its low value.

The design of the cell involves the selection of transistor sizes for all six transistors of Figure 8.8 to guarantee proper read and write operations. Since the cell is symmetric, only three transistor sizes need to be specified, either M_1, M_3, and M_5 or M_2, M_4, and M_6. The goal is to select the sizes that minimize the area, deliver the required performance, obtain good read and write stability, provide good cell read current, and have good soft error immunity (especially due to α-particles).

8.3.2 Read Operation

We now describe the design details of the 6T RAM cell for the read operation using Figure 8.10. Assume that a "0" is stored on the left side of the cell, and a "1" on the right side. Therefore, M_1 is on and M_2 is off. Initially, b and $\bar{b}$ are precharged to a high voltage around V_{DD} by a pair of column pull-up transistors (not shown). The row selection line, held low in the standby state, is raised to V_{DD} which turns on access transistors M_3 and M_4. Current begins to flow through M_3 and M_1 to ground, as shown in Figure 8.10a. The resulting cell current slowly discharges the capacitance C_{bit}. Meanwhile, on the other side of the cell, the voltage on $\bar{b}$ remains high since there is no path to ground through M_2. The difference between b and $\bar{b}$ is fed to a sense amplifier to generate a valid low output, which is then stored in a data buffer.

Figure 8.10
Design of transistor sizes for read operation.

Upon completion of the read cycle, the wordline is returned to zero and the column lines can be precharged back to a high value.

When designing the transistor sizes for read stability, we must ensure that the stored values are not disturbed during the read cycle. The problem is that, as current flows through M_3 and M_1, it raises the output voltage at node q which could turn on M_2 and bring down the voltage at node $\overline{q}$, as shown in Figure 8.10b. The voltage at node $\overline{q}$ may drop a little but it should not fall below V_S. To avoid altering the state of the cell when reading, we must control the voltage at node q by sizing M_1 and M_3 appropriately. We can accomplish this by making the conductance of M_1 about 3 to 4 times that of M_3 so that the drain voltage of M_1 does not rise above V_{TN}. In theory, the voltage should not exceed V_S, but this design must be carried out with due consideration of process variations and noise. In effect, the read stability requirement establishes the ratio between the two devices.

The other consideration in the read cycle design is to provide enough cell current to discharge the bitline sufficiently within 20 to 30% of the cycle time. Since the cell current, I_{cell}, is very small and the bitline capacitance is large, the voltage will drop very slowly at b, as shown in Figure 8.10b. The rate of change of the bitline can be approximated as follows:

$$I_{\text{cell}} = C_{\text{bit}}\frac{dV}{dt}$$

$$\therefore \frac{dV}{dt} = \frac{I_{\text{cell}}}{C_{\text{bit}}}$$

Clearly, I_{cell} controls the rate at which the bitline discharges. If we want a rapid full-swing discharge, we can make I_{cell} large. However, the transistors M_1 and M_3 would have to be larger. Since we have millions of such cells, the area and power of the memory would be correspondingly larger. Instead, we choose a different approach. We attach a sense amplifier to the bitlines to detect the small difference, ΔV, between b and $\overline{b}$ and produce a full-swing logic high or low value at the output. The trigger point relative to the rising edge of the wordline, $\Delta\tau$, for the enabling of the sense amplifier can be chosen by the designer based on the response characteristics of the amplifier. If the voltage swing ΔV and a target delay $\Delta\tau$ are specified according to Figure 8.10b, then

$$I_{\text{cell}} = \frac{C_{\text{bit}}\Delta V}{\Delta\tau}$$

This leads to the cell current value which, in turn, determines the final transistor sizes for M_1 and M_3. Alternatively, if the transistor sizes are determined to optimize the cell area, then the corresponding delay is computed as

$$\Delta\tau = \frac{C_{\text{bit}}\Delta V}{I_{\text{cell}}}$$

We now establish a rule of thumb for transistor sizes for the read cycle using an example.

Read Cycle Design Guidelines **Example 8.2**

Problem:

Compute the widths of M_1 and M_3 in Figure 8.10 given that the circuit can only tolerate a rise in voltage of 0.1 V at node q during the read operation. Assume that $C_{\text{bit}} = 2$ pF and that the specification calls for a 200 mV transition of the bitline in 2 ns. Use 0.13 μm technology parameters.

Solution:

When the wordline, wl, goes high, M_3 is a saturated enhancement load for the M_1 driver. The driver transistor is expected to be in the linear region of operation. Therefore, we can write the following equation:

$$\frac{W_1}{L_1}\frac{\mu_n C_{ox}}{\left(1+\frac{V_q}{E_{CN}L_1}\right)}\left[(V_{DD}-V_{T1})V_q - \frac{V_q^2}{2}\right] = \frac{W_3 v_{\text{sat}} C_{ox}(V_{DD}-V_q-V_{T3})^2}{(V_{DD}-V_q-V_{T3})+E_{CN}L_3}$$

We first eliminate C_{ox} from both sides of the equation. Now, setting $V_q = 0.1$ V and ignoring body effect, we obtain

$$\frac{W_1}{0.1\ \mu\text{m}} \frac{\left(270 \frac{\text{cm}^2}{V - \text{sec}}\right)}{\left(1 + \frac{0.1}{0.6}\right)} \left[(1.2 - 0.4)0.1 - \frac{0.1^2}{2}\right]$$

$$= \frac{W_3(8 \times 10^6\ \text{cm/s})(1.2 - 0.1 - 0.4)^2}{(1.2 - 0.1 - 0.4) + 0.6}$$

$$\therefore \frac{W_1}{W_3} \approx 1.7$$

This ratio would be smaller if body effect were taken into account. The actual values of the widths depend on the desired rate of change of the bitline voltage, the delay specification, and cell current. If we require a bitline transition of 200 mV in 2 ns, with a total bitline capacitance of 2 pF, then the cell current is

$$I_{\text{cell}} = C_{\text{bit}} \times \frac{\Delta V}{\Delta \tau} = 2\ \text{pF} \times \frac{200\ \text{mV}}{2\ \text{ns}} = 200\ \mu\text{A}$$

This is the average cell current through M_1 and M_3. As a rough estimate, we could simply use the current through the access transistor when it turns on:

$$I_{\text{cell}} \approx \frac{W_3(8 \times 10^6)(1.6\mu\text{F/cm}^2)(1.2 - 0.1 - 0.4)^2}{(1.2 - 0.1 - 0.4) + 0.6} = 200\ \mu\text{A}$$

$$\therefore W_3 = 0.4\ \mu\text{m}$$

This implies that $W_1 = 0.7\ \mu$m. These two sizes are larger than we would desire if we were trying to create a 1 Mbit SRAM. However, this example is intended to show the steps in the design process.

In practice, the device sizes are controlled by the RAM cell area constraints. As a rule of thumb, we typically use

$$\frac{W_1}{W_3} \approx 1.5 \tag{8.3}$$

and then optimize the sizes to provide the proper noise margin characteristics.

8.3.3 Write Operation

The operation of writing 0 or 1 is accomplished by forcing one bitline, either b or $\overline{b}$, low while the other bitline remains at about V_{DD}. In Figure 8.11, to write 1, $\overline{b}$ is forced low, and to write 0, b is forced low. The conditions when writing 1 are illustrated in Figure 8.11a. The cell must be designed such that the conductance of M_4 is several times larger than M_6 so that the drain of M_2 is pulled below V_S. This initiates a regenerative effect between the two inverters. Eventually, M_1 turns off and

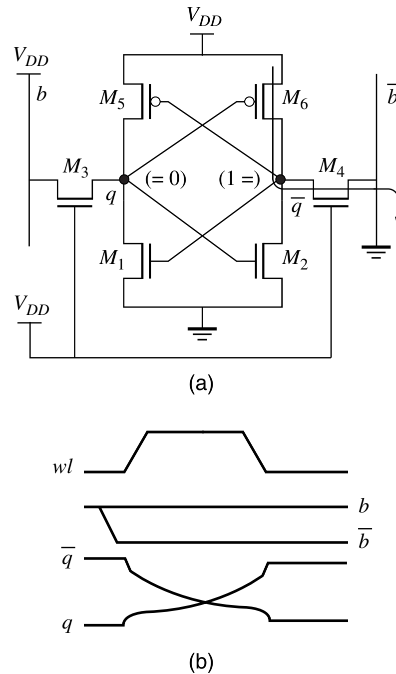


Figure 8.11
Write operation and waveforms for 6T SRAM.

its drain voltage rises to V_{DD} due to the pull-up action of M_5 and M_3. At the same time, M_2 turns on and assists M_4 in pulling output $\overline{q}$ to its intended low value. When the cell finally flips to the new state, the row line can be returned to its low standby level.

The design of the SRAM cell for a proper write operation involves the transistor pair M_6-M_4. As shown in Figure 8.11a, when the cell is first turned on for the write operation, they form a pseudo-NMOS inverter. Current flows through the two devices and lowers the voltage at node $\overline{q}$ from its starting value of V_{DD}. The design of device sizes is based on pulling node $\overline{q}$ below V_S to force the cell to switch via the regenerative action. This switching process is shown in Figure 8.11b. Note that the bitline $\overline{b}$ is pulled low *before* the wordline goes up. This is to reduce the overall delay since the bitline will take some time to discharge due to its high capacitance.

The pull-up to pull-down ratio for the pseudo-NMOS inverter can be determined by writing the current equation for the two devices and setting the output to V_S. To be conservative, a value much lower than V_S should be used to ensure proper operation in the presence of noise and process variations. Based on this analysis, a rule of thumb is established for M_6-M_4 sizing:

$$\frac{W_4}{W_6} \approx 1.5 \tag{8.4}$$

The two ratios provided in Equations (8.3) and (8.4) are only estimates. One should remember that the actual values will depend on a number of factors such as area, speed, and power considerations. However, these two rules of thumb can be used to validate the solution, once obtained.

Exercise 8.1 Compute the ratio of M_6 to M_4 for the circuit in Figure 8.11 assuming that the node $\bar{q}$ is to be pulled down to V_{TN}, which is well below the switching threshold.

8.3.4 SRAM Cell Layout

Once the cell transistors are designed, the next step is to construct a cell layout. A typical layout for a CMOS static cell is shown in Figure 8.12. Every effort is made to minimize the area of the memory cell. Extensive use is made of symmetry to allow the core array to be generated by simply "tiling" the cells together vertically and horizontally. Two levels of metal and one layer of poly are used to realize this memory cell. If we examine the layout carefully, we should be able to identify a total of six transistors.

Starting at the top, we have the two p-channel devices, M_5 and M_6, laid out horizontally. In the middle of the cell, we have M_3 and M_4, which are the two pull-down transistors laid out horizontally. Finally, near the bottom of the cell are the two access transistors, M_1 and M_2, laid out vertically. The bitlines, b and $\bar{b}$, are routed in Metal2 vertically, while the wordline is horizontally routed in both poly and Metal1

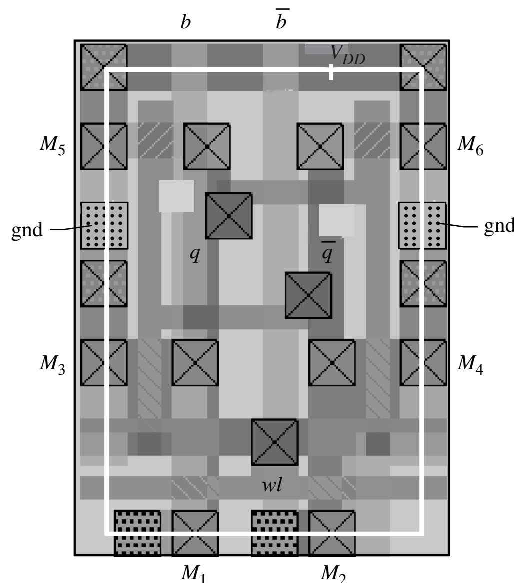


Figure 8.12
SRAM cell layout.

near the bottom of the cell. V_{DD} is routed in Metal1 at the top of the cell while Gnd is routed in Metal1 near the middle of the cell. The source/drain contacts are shared between pairs of neighboring cells by mirroring the cell vertically. The capacitance of the contacts per cell is therefore half the actual value due to sharing. The cell indicated by the center bounding box is replicated to create the core array. This cell is approximately 40λ by 30λ. Note that the substrate and well contacts are contained inside the cell. Removal of substrate and well plugs from the cell would result in a smaller cell.

The large number of devices connected to the wordline and bitlines gives rise to large capacitance (and resistance) values as described earlier. The row lines are routed in both Metal1 and poly to reduce resistance, while the bitlines are routed in Metal2. Calculations of total capacitance may be carried out using Equations (8.1) and (8.2).

Capacitance Calculations for the Wordline and Bitlines **Example 8.3**

Problem:

What is the capacitance of the wordline and the bitlines for a 64K SRAM that uses the cell layout of Figure 8.12 with access transistors that are 0.5 μm/0.1 μm in size? The contacts on the bitlines are shared between pairs of cells and have a capacitance of 0.5 fF each. Wire capacitance is 0.2 fF/μm. Assume 0.13 μm technology parameters. The cell layout is 40λ by 30λ. Note that $1\mu m = 20\lambda$.

Solution:

If we were to design a 64K SRAM, it would contain a core area of 256×256. Ignoring the resistance for the moment, the row capacitance would be due to the gate capacitance of 512 access transistors and the wire capacitance of 256 cells:

$$C_{word} = 512 \times 2 \text{ fF}/\mu m \times 0.5\ \mu m + 256 \times 30\lambda \times 0.2 \text{ fF}/\mu m \times 1\ \mu m/20\lambda$$
$$= 589 \text{ fF}$$

The bitline capacitance per cell due to the source/drain capacitance of the access transistors is lower than usual since the voltage drop across the junction is close to V_{DD}. In addition, there is wire capacitance and a half contact capacitance per cell. The total is

$$C_{bit} = 256 \times 0.5 \text{ fF}/\mu m \times 0.5\ \mu m + 256 \times 40\lambda \times 0.2 \text{ fF}/\mu m \times 0.1\ \mu m/2\lambda$$
$$+ 128 \times 0.5 \text{ fF} = 230 \text{ fF}$$

8.4 SRAM Column I/O Circuitry

In this section, we examine the column input/output (I/O) circuitry, which includes bitline precharge circuits, column multiplexers, write circuits, and read circuits. The column I/O must be designed with due consideration of the cell design and the timing specifications of the memory.

8.4.1 Column Pull-Ups

In both read and write operations, the bitlines are initially pulled up to a high voltage near V_{DD}. The circuits used to precharge the bitlines depend on the type of sensing that is used in the read operation. Figure 8.13 illustrates three possible precharge configurations. In Figure 8.13a, the precharge is similar to the dynamic logic precharge described earlier in Chapter 7. A precharge signal, PC, is applied to the two pull-ups and to a third transistor, called the *balance* transistor, connected between the two bitlines to equalize their voltage levels. When the wordline (*wl*) signal goes high, one bitline remains high and the other falls at a linear rate until *wl* goes low. The difference between the bitlines is fed into a voltage-sensing latch-based amplifier that is triggered when the differential voltage exceeds a certain threshold.

The precharge circuit of Figure 8.13b is reminiscent of the pseudo-NMOS circuits. Two static loads and a balance transistor form the precharge circuit. When PC is applied to the balance transistor, it simply equalizes the two voltage levels. Once the bitlines are precharged, the PC signal is turned off (raised to V_{DD}) and, at this point, the wordline can be activated. Of course, the pull-ups are still *on* so current will flow through one of them and into the cell side with the stored "0." Eventually, a steady-state output level will be reached by the bitline, as shown in the figure. This type of pull-up is suitable for current-sensing amplifiers since there is continuous

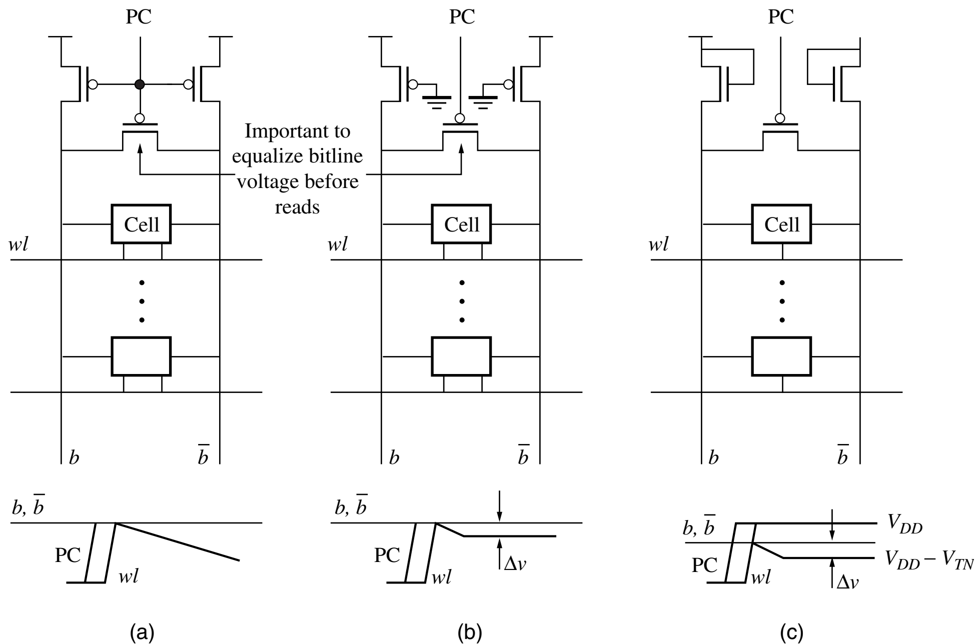


Figure 8.13
Column pull-up configurations.

current flow, or latch-based voltage-sensing amplifiers since the bitlines will establish a differential voltage, Δv.

Figure 8.13c is based on the NMOS saturated enhancement load. Therefore, the maximum possible voltage on the bitline is $V_{DD} - V_T$. When PC is applied to the balance transistor, it equalizes the two voltage levels. Once the lines are precharged high, the PC signal is turned off (raised to V_{DD}) and then *wl* goes high. At this point, the pull-ups are still active so current will flow through one of them into the cell side with the stored "0." Again, a steady-state output level will be reached by the corresponding bitline, as shown in the figure, although this value will be lower than the pseudo-NMOS case. This type of pull-up is suitable for differential voltage sensing amplifiers since the bitline voltages initially start at $V_{DD} - V_{TN}$. This lower voltage is needed for a proper biasing and output swing of the differential amplifier, as will be described later.

The PC signal may be generated in a variety of ways, but typically it is produced by an address transition detection (ATD) circuit. One form of this circuit is shown in Figure 8.14. The ATD signal is triggered by any transition on the address inputs. The basic circuit is comprised of a set of XOR gates, each with a delay element on one of the inputs. When an address line changes, it causes the XOR gate to generate a short pulse since the inputs differ in value for a short time. Circuits that generate a short pulse of this nature are called *one-shots*, which are part of the *monostable* family of circuits. The duration of the pulse, τ_D, is determined by the delay element. The delay line may be constructed from a simple inverter chain with an even number of inverters. In the figure, N is an even number and τ_{Pinv} is the inverter propagation delay.

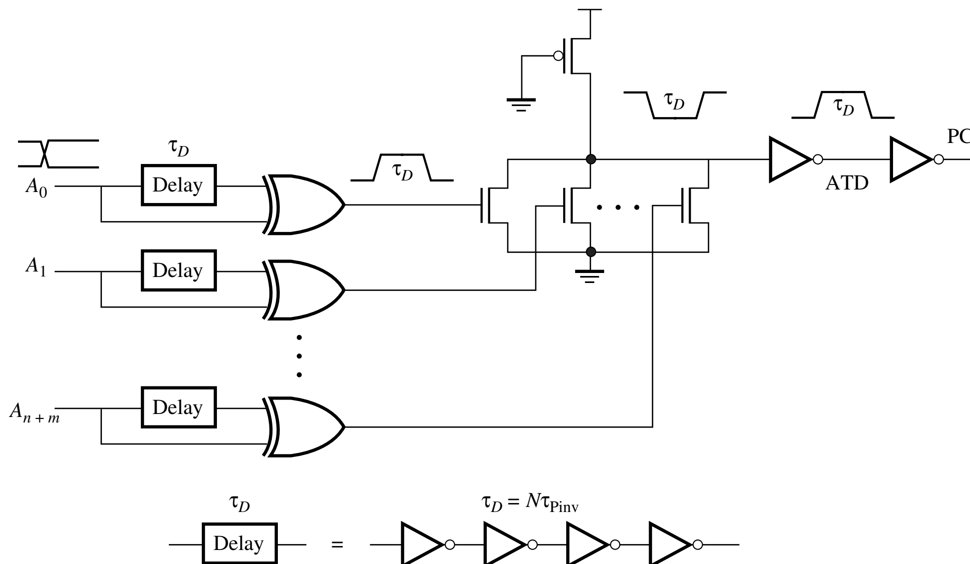


Figure 8.14
Address transition detection (ATD) circuit.

Once the pulse is generated, it turns on one of the pull-down transistors of the pseudo-NMOS NOR gate. A negative going pulse is generated at its output. This is passed through another inverter to generate the actual ATD signal. Many of the timing signals in SRAMs are derived from this basic circuit so it is required to drive a very high capacitance. Therefore, the signal should be properly buffered using logical effort or any other optimization method. Once generated, it can be inverted and applied to the bitline precharge elements as the PC signal. The address transitions usually take place before the beginning of a clock cycle and, as a result, the precharge operation typically occurs at the end of a previous memory cycle.

8.4.2 Column Selection

Once all the columns have been pulled up to a high voltage, the next step is to select the column(s) that will be involved in the read or write operation. This column selection is performed using a decoder/multiplexer combination. The m-bit column address is used to select one or more of the 2^m columns. This can be performed using a decoder, similar to the row decoder, driving a number of CMOS pass transistors as shown in Figure 8.15. The pass transistors require complementary signals. Of course, if an 8-bit output is desired, then the decoder outputs would each drive eight CMOS transmission gates, and fewer column address bits would be needed.

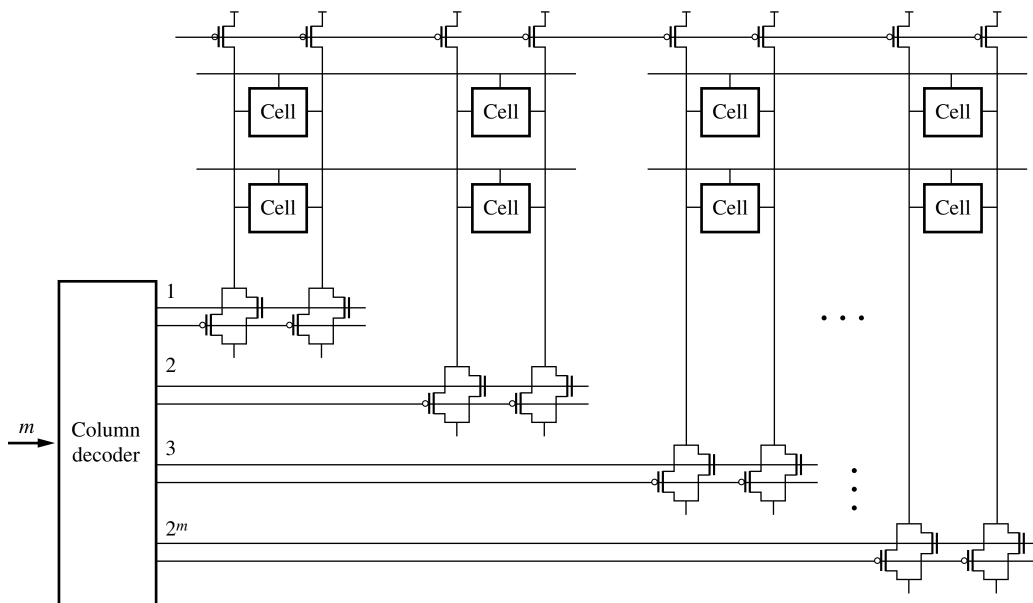


Figure 8.15
Column decoding and multiplexing.

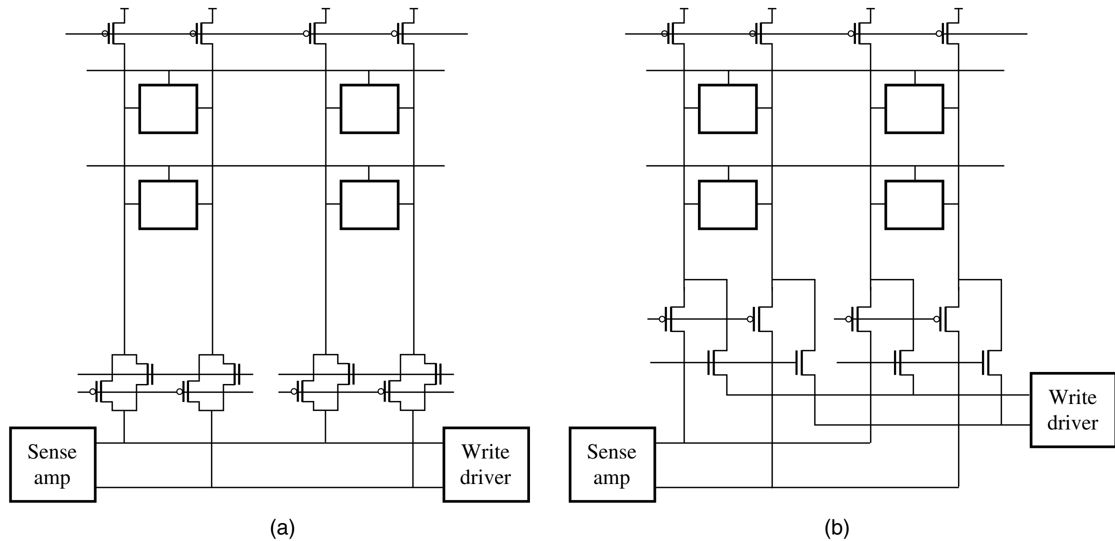


Figure 8.16
Column selection.

The optimal design of the column decoder proceeds in the same way as described earlier for the row decoder. The transmission gates driven by the decoder are also sized for optimal speed. They are connected to the sense amplifier for read operations and the write driver for write operations. This is shown in Figure 8.16a. Note that the use of the CMOS transmission gate presents a routing problem since each of the signals driving the pass transistors must be complementary (we are driving both PMOS and NMOS devices).

The routing can be simplified by realizing that the PMOS device is better at transmitting high signals while the NMOS device is better at transmitting low signals. Since the bitlines are near V_{DD} during a read, we should turn on the PMOS device during a read operation and leave the NMOS device off. During a write, one bitline is pulled to a low voltage. Therefore, we leave the PMOS device off and only turn on the NMOS device. It is possible to separate the NMOS and PMOS devices and only turn them on when needed. This is shown in Figure 8.16b. The NMOS devices are only connected to the write drivers while the PMOS devices are only connected to the sense amplifiers since they would be turned on during a read operation.

Now that the lines have been separated, there is one other improvement we can consider. Rather than a single level of multiplexing, it is possible to reduce the overall transistor count by using a tree decoding structure as shown in Figure 8.17. In this example, we have a 4-bit column address which would normally translate to 16 enable lines. Instead, we use two 2-to-4 decoders that select 1-out-of-4 pass transistors at each level. As we add more levels in the tree, the signal path is slower but the decoder size is reduced. For this example, we have shown two-level tree decoding

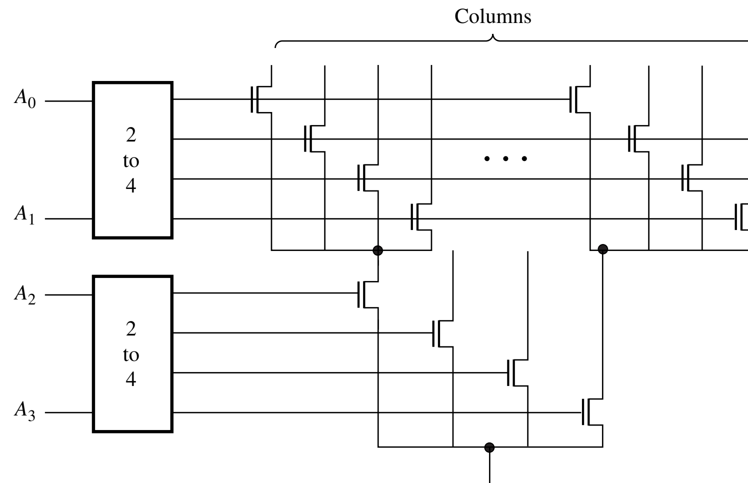


Figure 8.17
Two-level tree decoder for a 4-bit column address.

with exclusively NMOS pass transistors. We would need a corresponding set of PMOS transistors in a similar tree configuration. The tree decoding strategy requires less power but may be slower as more and more levels are added. These circuits should be designed based on the timing and power specifications for the memory.

8.4.3 Write Circuitry

Simplified write circuitry for the SRAM is shown in Figure 8.18. The operation is as follows. First, the columns are precharged to V_{DD} using M_7 and M_8. Next, the address and data signals are set up and held stable for a required amount of time before the clock is applied. The address signals are converted into column select and wordline activation signals. Before the wordline is enabled, the data and write signals are applied to pull one column to ground while leaving the other side at V_{DD}. This is done by ANDing the input data with the write signal. The pull-down transistors, M_{13}, M_{14}, and M_{15} in this case, are sized to discharge the column line in a specified amount of time. When the wordline goes high, current flows out of the cell and flips the sense of the cell. As described earlier, the internal cell voltage must be pulled below the switching threshold to enable the cell to flip state. Once it has switched, the wordline and column select lines may be returned to their standby values.

8.4.4 Read Circuitry

The read circuitry is activated when the wordline goes high. The cell transistors draw current from the highly capacitive column during a read operation. Therefore,

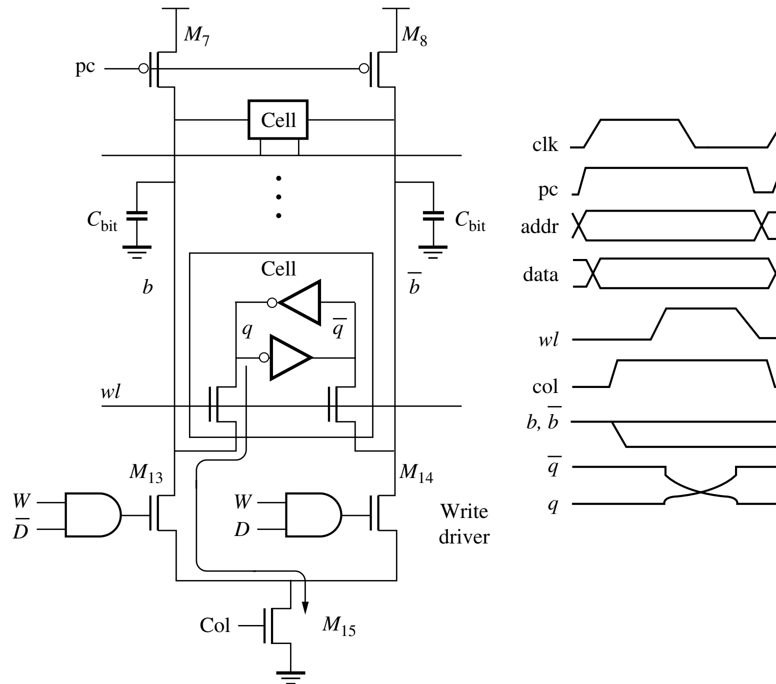


Figure 8.18
Write driver circuit.

the bitlines slowly drop in voltage and could potentially cause long access times. To reduce read access time, the memory is designed so that only a small voltage change on one column line or the other is needed to detect the stored value. Two or more amplifying stages are used to generate a valid logic output when the voltage difference between b and $\overline{b}$ is about 150–200 mV. Thus the column delay is only due to the time needed to achieve this small voltage change.

Figure 8.19 shows a simplified version of the read circuitry for a CMOS static memory. One of the precharge circuits of Figure 8.13 is used to pull the column lines high. In this case, the columns are biased at V_{DD} by transistors M_7, M_8. Then, the address, data (not used during read), and clock signals are applied. Again, the address signals translate into column enable and wordline activation signals. Usually the column selection and sense enable are activated at the same time. The sense amplifier depicted in Figure 8.19 is used to provide valid high and low outputs using the small voltage difference between inputs b and $\overline{b}$. The precharge circuit must be consistent with the sense amplifier circuit. Otherwise, the sense amplifier may not operate properly.

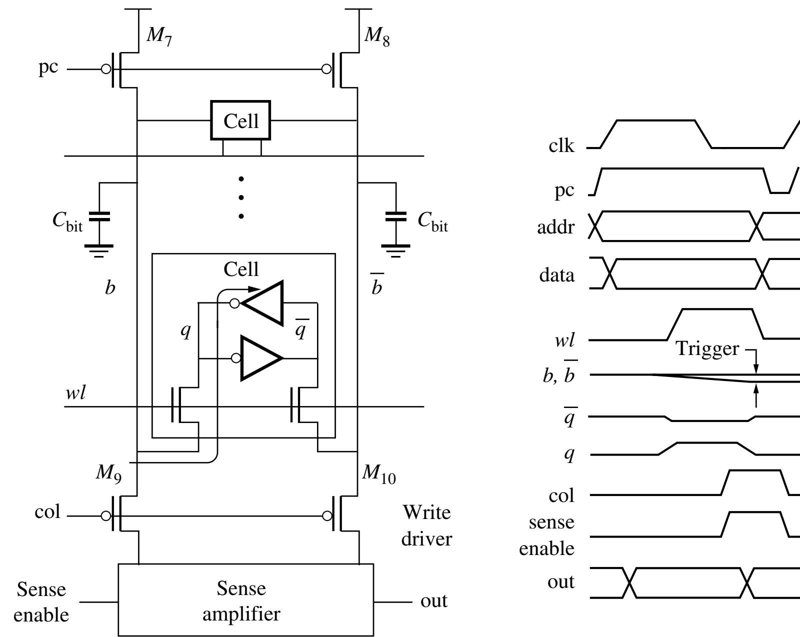


Figure 8.19
Basic read circuitry.

Figure 8.20 provides an example of a sense amplifier in the form of a conventional CMOS differential amplifier.[4] Typically, SRAMs may use eight identical sense amplifiers to provide simultaneous output of eight data bits. When using these types of sense amplifiers, the column pull-up transistors must be saturated enhancement loads (shown at the top of Figure 8.13c). Otherwise, the inputs would start at V_{DD} which would make it difficult to bias M_4 and M_5 properly for the desired output swing at node *out*.

The main reason for using this type of sense amplifier is to improve the noise immunity and speed of the read circuit. Since the voltage swing on the bitlines is limited due to the large capacitances, any noise on these lines may cause an error in the reading process. In fact, any noise that is common to b and $\overline{b}$ should not be amplified. We are only interested in the differential signal changes between the two bitlines. The sense amplifier shown in Figure 8.20 attenuates *common-mode noise* and amplifies *differential-mode signals*. The detailed operation of this circuit involves the concepts of analog circuit design. However, since we are using it for

[4] These are called *sense amplifiers* when used in memory applications; their role is to sense which bitline is dropping in voltage. Normally, such a circuit is used for small-signal voltage gain rather than large-signal sensing applications.

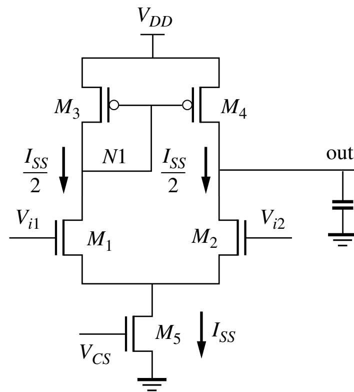


Figure 8.20
Differential voltage sense amplifier.

large-signal applications, we can study its properties from a more digital point of view.

The circuit can be divided into three components: the current mirror, common-source amplifier, and the biasing current source. All transistors are initially placed in the saturation region of operation so that the gain is large. They also use large values of channel length, L, to improve linearity.

The two transistors, M_3 and M_4, act to provide the same current to the two branches of the circuit. That is, the current flowing through M_3 is mirrored in M_4:

$$I_3 \approx I_4$$

Any difference in the two currents is due to differences in their V_{DS} values. The transistor M_5 sets the bias current, I_{SS}, which depends on the bias voltage V_{CS}. At steady-state, the current flowing through the two branches of the amplifier should be equal to $I_{SS}/2$.

The two input transistors, M_1 and M_2, form a source-coupled differential pair. The two input voltages, V_{i1} and V_{i2}, are connected to the column lines. The biasing of the circuit must be done carefully to allow the output node to have a large enough swing. Specifically, the transistors must be *on* and in the saturation region of operation for high gain. In order to accomplish this, the inputs to M_1 and M_2 must be set to approximately $V_{DD} - V_{TN}$ rather than V_{DD}. To understand this, consider the case when the inputs are precharged to V_{DD}. To keep the input devices in saturation, their two drain nodes, $N1$ and *out*, would be biased at the saturation voltage:

$$V_{N1} = V_{\text{out}} = \frac{(V_{DD} - V_{TN})E_{CN}L_N}{(V_{DD} - V_{TN}) + E_{CN}L_N} \approx (V_{DD} - V_{TN})$$

The above simplification is possible since the channel lengths are large.

A problem arises if the two nodes, $N1$ and *out*, are biased at this value: both *p*-channel devices would be at the edge of cutoff. In practice, the PMOS threshold voltage is higher in magnitude than the NMOS threshold voltage. Therefore, both M_3 and M_4 would be completely off in the steady-state condition. Instead, if we biased the inputs of M_1 and M_2 at $V_{DD} - V_{TN}$, then

$$V_{N1} = V_{\text{out}} \approx (V_{DD} - V_{TN}) - V_{TN} \approx V_{DD} - 2V_{TN}$$

Now there is enough *headroom* for the two PMOS devices to be comfortably *on* and in saturation. This input bias condition requires the use of the column pull-up circuits of Figure 8.13c.

With the biasing established, the sense amplifier operates as follows. Initially, the bias currents in the two branches are equal and the two inputs are at $V_{DD} - V_{TN}$. When the voltage at one input decreases, it decreases the current in that branch. At the same time, the current in the other branch increases in value to maintain a total of I_{SS} through M_5.

We examine the discharging and charging cases in Figure 8.21. Assume that M_1 is the input that has dropped below $V_{DD} - V_{TN}$ by the prescribed amount to turn it off, as in Figure 8.21a. This implies that the currents in M_3 and M_4 are both zero. However, since the current in M_5 is I_{SS}, it follows that this current must be discharging the output capacitance through M_2. Therefore, the output voltage is quickly forced to ground. In the other scenario depicted in Figure 8.21b, if the input V_{i2} drops by the prescribed amount, M_2 is turned off. Then all the current flows through M_1, M_3, and M_5. The current of M_3 is mirrored in M_4. Since the current in M_2 is zero, this current must flow to the output to charge it to V_{DD}.

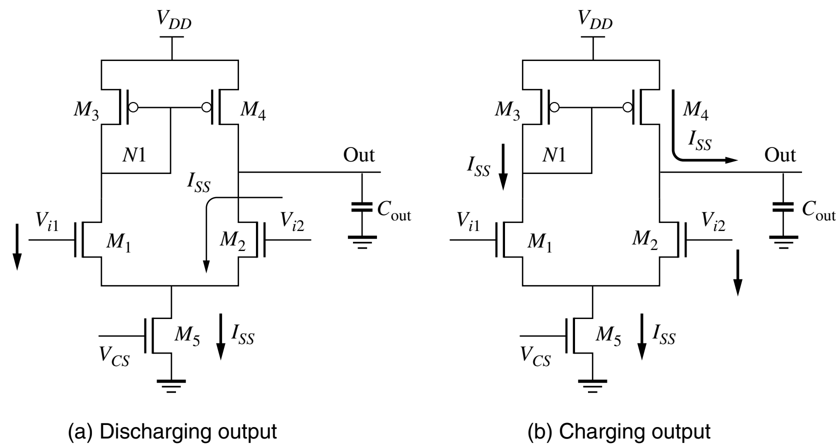


Figure 8.21
Detecting "0" and "1" using a differential sense amplifier.

This type of differential amplifier is used in high-speed applications. The required differential voltage for proper operation is of the order of 100 mV to 200 mV. Its speed can be adjusted depending on how much power dissipation can be tolerated. Consider modeling the output transition as a current source driving a capacitor. The rate of change of the output, whether switching high or low, is given by

$$\frac{dV}{dt} = \frac{I_{SS}}{C_{\text{out}}}$$

This is called the *slew rate* (i.e., dV/dt at the output). Rearranging, the delay through the sense amplifier is

$$\Delta\tau = \frac{C_{\text{out}}\,\Delta V_{\text{out}}}{I_{SS}}$$

To reduce the delay, a large I_{SS} can be selected. However, the power dissipation in steady-state is given by

$$P = I_{SS}V_{DD}$$

Therefore, a tradeoff exists between the speed and the power dissipation; both are controlled by the choice of I_{SS}. Once a suitable value of I_{SS} is selected, the W/L of the devices can be determined. For the input devices, the W/L determines the $V_{GS} - V_{TN}$ value. This value is the gate overdrive term that establishes the desired bitline swing value. As the input transistor sizes increase, the gate overdrive term decreases. Since we require a small gate overdrive, the input devices are required to be rather large. The sizes of the other transistors are based on the bias voltages needed at the internal nodes. The complete design of such amplifier circuits falls into the realm of analog circuit design. Further details can be obtained by consulting the references at the end of the chapter.

A second option for the sense amplifier is the latch-based circuit shown in Figure 8.22. The circuit is effectively a cross-coupled pair of inverters with an enabling transistor, M_1. This circuit relies on the (slower) regenerative effect of inverters to generate a valid high or low voltage. It is a lower power option since the circuit is not activated until the required potential difference has developed across the bitlines. However, it is slower since it requires a large input voltage difference and is not as reliable in the presence of noise as the previous sense amplifier.

The initial sequence of operations is similar to the differential sense amplifier described above. The bitlines are precharged to V_{DD} with either Figure 8.13a or Figure 8.13b. Then the wordline is enabled and one of the bitlines drops in voltage. As the bitline differential voltage reaches the prescribed amount, the sense enable is activated. The timing of the sense enable is critical as described later. For now, assume that it arrives at the proper time. At this point, the bitline difference is fed into the cross-coupled inverters. One side drops in voltage faster than the other side, since one side will always have more gate overdrive than the other. When the voltage drops below V_{TN} on one side, it turns off the pull-down transistor of the opposite side, and

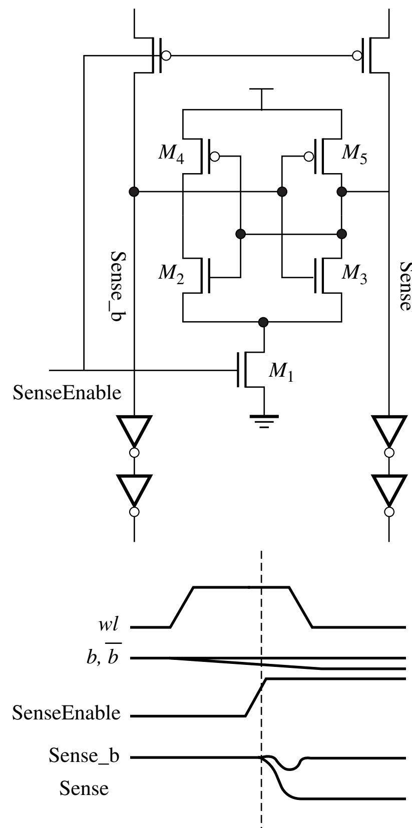


Figure 8.22
Latch-based sense amplifier.

the pull-up transistor acts to raise the voltage to V_{DD}. This regenerative process is shown in the timing diagram of Figure 8.22. The device sizing follows previously discussed methods for flip-flops.

The more important issue is the timing of the SenseEnable signal. If the latch is enabled too early, the latch may not flip in the proper direction due to noise. If it is enabled too late, then it will add unnecessary delay to the access time. In addition, process variations will control the actual timing of the signal. In order to guarantee that the signal will arrive at the proper time in the presence of process variations, one needs to introduce a replica circuit that mimics the delay of the actual signal path, shown in Figure 8.23. Here, the upper path emanating from the clock is the actual signal path to the bitlines. The SenseEnable should arrive as soon as the bitline swing reaches the desired value. By creating a second path (the lower path) that exhibits the same delay characteristics, we can ensure that the SenseEnable arrives at the correct time.

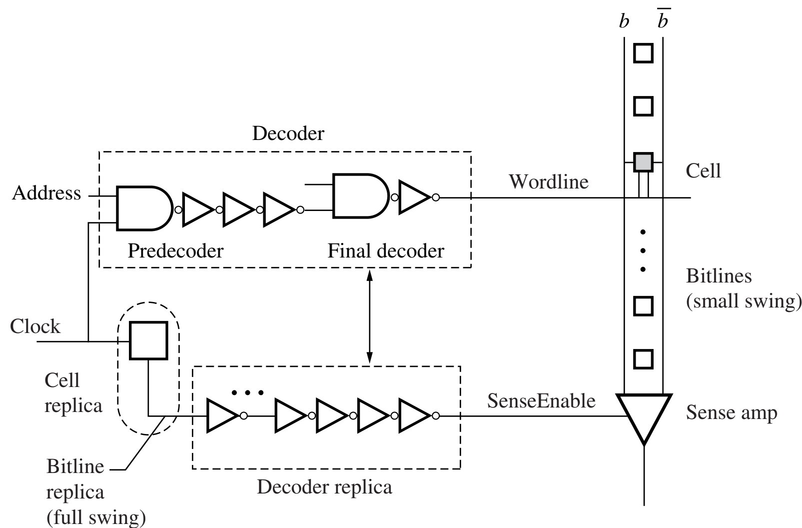


Figure 8.23

Replica circuit for sense amplifier clock enable.

The critical path for the read cycle starts from the clock and the address inputs and terminates at the sense amplifier inputs. The signal flows through the decoder and generates the wordline, which activates the memory cell that drives the bitlines. The swing on the bitlines is presented to the sense amplifier. This is the point at which we wish to enable the sense amplifier. The purpose of a replica circuit is to duplicate the delays along this path with circuits that correspond to each delay element. Essentially, we want to have a decoder replica that tracks the gate delays in the real decoder, and a cell replica that tracks the bitline discharge delay of the actual bit cell.

Note that we have placed the memory cell replica *before* the decoder in Figure 8.23. It is not appropriate to place the memory cell after the decoders in the replica path since it would have to drive all the sense amplifiers at the bottom of the memory. Since a small memory cell does not have the needed drive capability, we place the replica cell ahead of the decoder. We can keep the memory small and still deliver a full-swing signal as needed by the input to the decoder replica. The buffers of the decoder replica can be used to drive the sense amplifiers.

One issue for the replica memory cell in this configuration is that we require the full-swing output to have the same delay as the small swing on the actual bitlines. For example, if the actual cell requires 500 ps to transition by 180 mV, then the replica memory cell would require approximately 5 ns to transition by 1.8 V. This is not an acceptable delay in the replica path.

The replica cell should, in fact, be a replica column line with only enough cells to match the timing of the actual column. This is shown in Figure 8.24. For example,

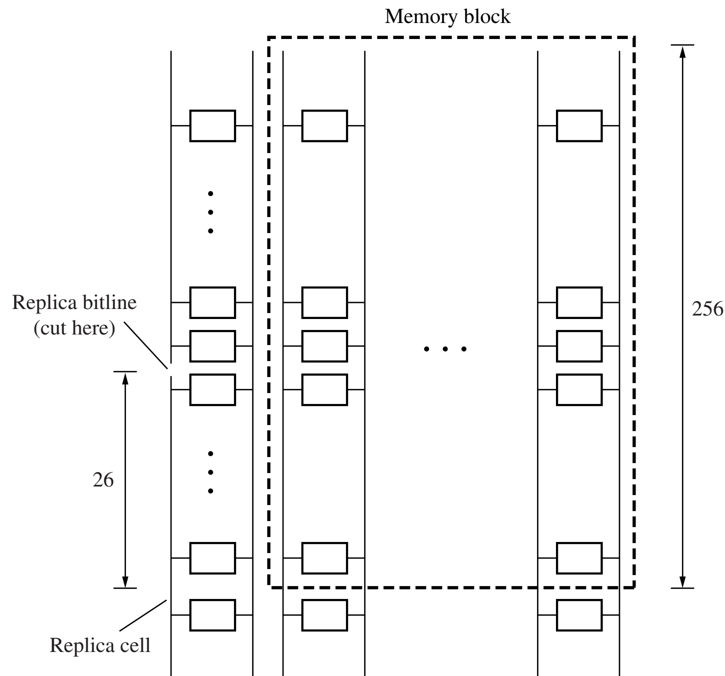


Figure 8.24
Replica cell design.

if we have 256 bits in the true bitline with a swing of 180 mV, then we only require roughly 26 cells in the replica circuit to produce a full swing of 1.8 V in the same time interval. The slight round off error in the number of cells used is not an issue since we will ensure that the replica path is slightly longer than the actual path delay. With the full swing from the replica path cells, we can drive the decoder replica gates. The needed 26 bits can be cut from a section of additional columns that are always fabricated alongside the main memory array to avoid "edge effects" on each end.

To ensure that the SenseEnable does not arrive too early for any reason, we should add an extra gate delay or two to the decoder replica. Designed in this manner, the SenseEnable will arrive at the proper time in the presence of process and environmental variations.

8.5 Memory Architecture

The overall memory architecture can now be described. In Figure 8.25, we illustrate a high-level layout of the memory array. The core array containing the cells is the largest block. The bitline precharge circuits are positioned above the core. The row

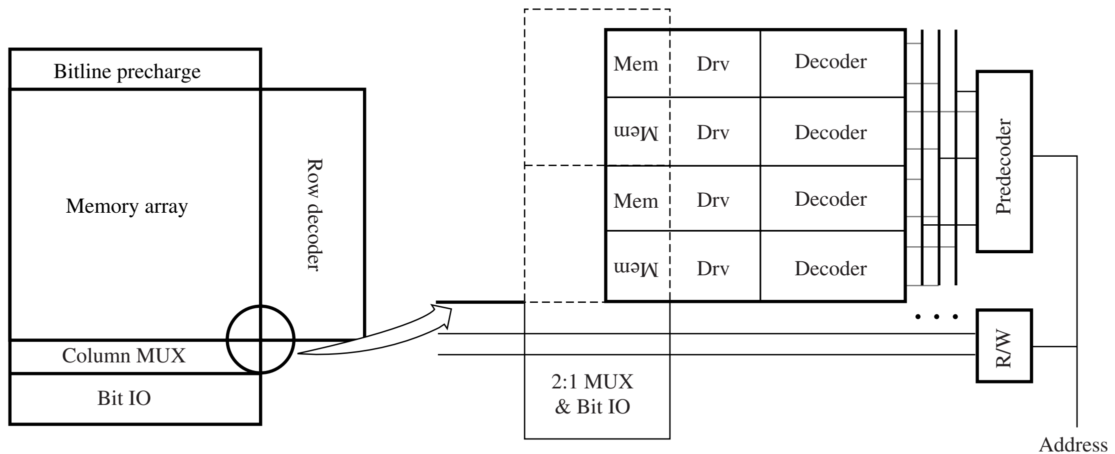


Figure 8.25
Basic memory architecture.

decoder is placed on the right side and the column multiplexer and bit I/O are located below the core array. If we zoom in on a corner region of the memory, as shown in the right-hand side of Figure 8.25, we see that the row decoder is comprised of a predecoder and a final decoder. The decoder drives the wordlines horizontally across the array. Each pair of bitlines feeds a 2:1 column decoder (in this case) which is connected to the bitline I/O circuits such as sense amplifier and write drivers. Each memory cell is mirrored vertically and horizontally to form the array, as indicated in the figure.

Several factors contribute to a limit on the maximum speed of operation. Delays in address buffers and decoders naturally increase as the number of inputs and outputs increase. Row lines are typically formed in polysilicon and may have substantial delays due to distributed *RC* parameters. A metal line may be placed in parallel and contacted to the poly line to reduce the delay (cf., Figure 8.12). Column lines are usually formed in metal, so resistance is not as significant, but the combined capacitance of the line and many parallel access transistors connected to them results in a large equivalent lumped capacitance on each of these lines. The large capacitances on the wordline and bitlines also contribute to excess power dissipation.

In order to reduce delay and power, a number of different partitioning approaches have been used. One technique is the divided wordline (DWL) strategy shown in Figure 8.26 for a 64K bit SRAM. Part of the 8 row address bits (6 in this case) are used to define global wordlines. A total of 64 global wordlines are created. These lines do not drive memory cells (i.e., the two access transistors within each cell) and therefore have far less capacitance than the regular wordlines. The remaining 2 bits of the address are used to generate local wordlines that actually drive the

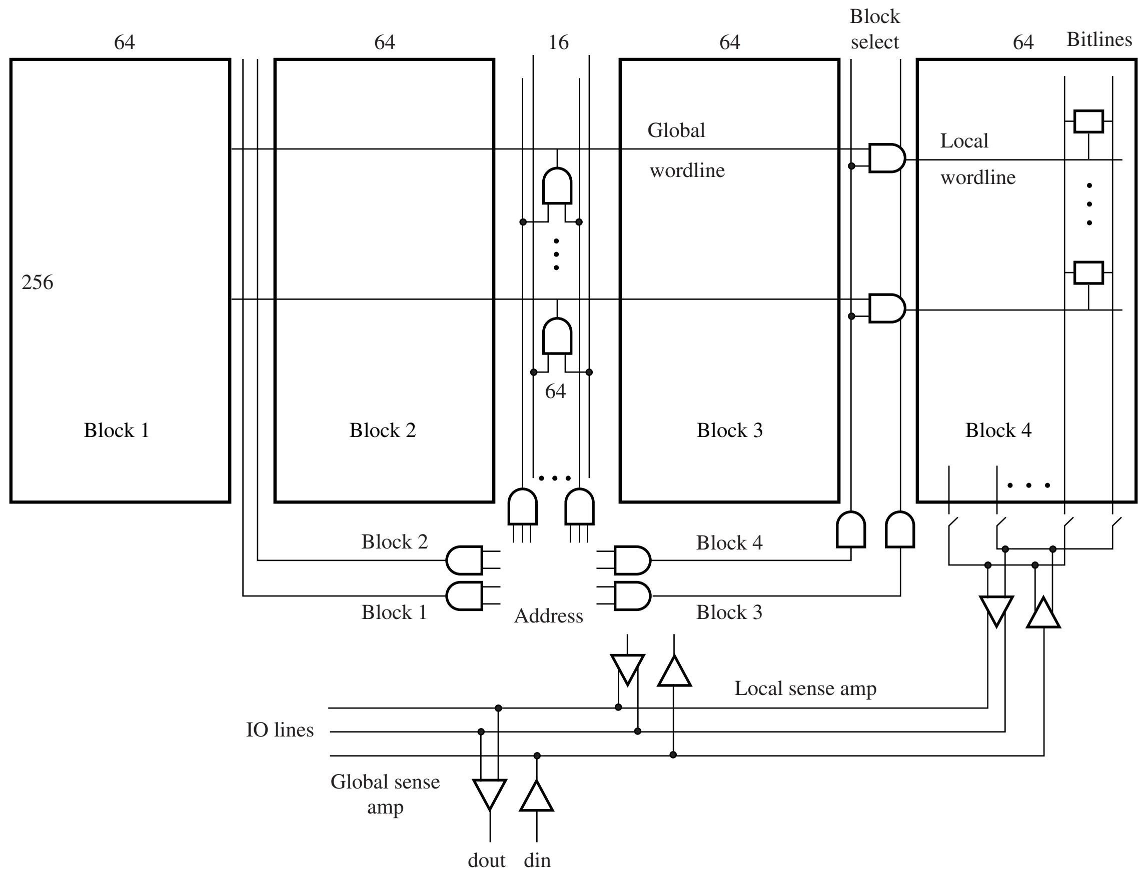


Figure 8.26
Divided wordline strategy to reduce power and delay.

cell access transistors. In this example, four blocks are created and accessed using the local wordlines. The total cell capacitance is reduced by up to a factor of 4. Therefore, the power will be reduced greatly. In addition, the delay along the wordlines is also reduced.

A similar partitioning strategy can be applied to the bitlines, as shown in Figure 8.27. An architecture without partitioning is shown in Figure 8.27a. For this case, neighboring pairs of bitlines are multiplexed to produce 128 outputs (i.e., there are 128 sense amplifiers in this example). If the bitlines are partitioned into two sections, the bitline capacitance is reduced by a factor of 2. The proper cell must be selected using a two-level multiplexing scheme of Figure 8.27b. To achieve the same bitline swing as in Figure 8.27a would only require roughly half the time. Further partitioning can be carried out with a corresponding increase in the complexity of multiplexing.

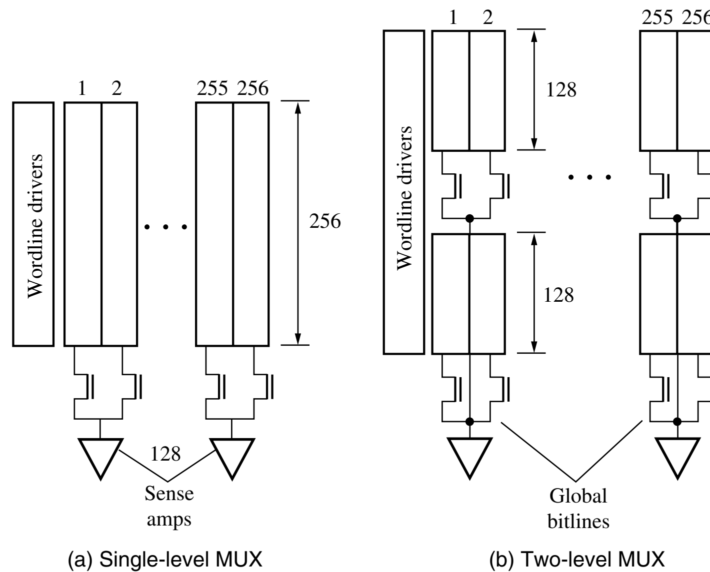


Figure 8.27
Bitline partitioning to reduce delay.

8.6 Summary

This chapter has focused on the application of material in previous chapters to the design of an SRAM. Modern memories are, of course, much more complicated but most of them can be understood with the basic concepts that have been presented in this chapter. Since memory is very regular in nature, the design process has reached a point where it can be implemented in software. Today, there are CAD tools called *memory compilers* that can generate a memory design within minutes. While the sizes of the memories that can be generated are still limited to some degree, the automatic synthesis approach for memories will be used more frequently in the future. This concludes the discussion on SRAM circuits. Other types of memory circuits are described in Chapter 9.

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. M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits, Analysis and Design*, Third Edition, McGraw-Hill, New York, 2003.
5. H. Veendrick, *Deep-Submicron CMOS ICs*, Second Edition, Kluwer Academic Publishers, Boston, MA, 2000.

6. J. P. Uyemura, *CMOS Logic Circuit Design*, Kluwer Academic Publishers, Boston, MA, 1999.
7. Behrad Razavi, *Design of Analog CMOS Integrated Circuits*, McGraw-Hill, New York, 2001.

PROBLEMS

P8.1. What are the main differences between the ROM, SRAM, DRAM, EPROM, E^2PROM, and Flash? Which is the most popular memory for embedded applications (i.e., on the same chip as the processor logic blocks)? Describe suitable applications for each one.

P8.2. Draw the circuit equivalent for the 6T SRAM of Figure 8.12. Estimate the height and width of the cell. Assume that contacts are 4λ by 4λ.

P8.3. Implement an 8-bit decoder using NAND2, NAND3, and NAND4 logic gates and inverters, following a two-level decoding scheme. You do not have to design the sizes of the transistors. Why is a two-level scheme preferred over a multilevel scheme? What is the *branching effort* of the decoder from input to output (see Chapter 6 for the definition of this term)?

P8.4. Consider the SRAM cell of Figure 8.8 with a stored 0 on the left side and a stored 1 on the right side. Design the transistors of the SRAM such that node q does not exceed V_{TN} during a read operation and node $\overline{q}$ drops below V_S during a write operation. The desired cell current during a read operation is 300 μA. Use 0.18 μm technology parameters.

P8.5. Redesign the SRAM cell of the previous problem by assuming worst-case conditions for a read operation as follows: the threshold voltage of M_3 is reduced by 10%, the width M_3 is increased by 10%, the threshold voltage of M_2 is decreased by 10%, and the width of M_2 is increased by 10%. Explain why this is considered to be worst case for a read operation. Simulate the circuit in SPICE to demonstrate its operation under worst-case conditions.

P8.6. Redesign the SRAM cell of the previous problem by assuming worst-case conditions for a write operation as follows: the threshold voltage of M_4 is increased by 10%, the width M_4 is decreased by 10%, the threshold voltage of M_1 is decreased by 10%, and the width of M_1 is increased by 10%. Explain why this is considered to be worst-case for the write operation. Simulate the circuit in SPICE to demonstrate its operation under worst-case conditions.

P8.7. Consider the 6T SRAM cell of Figure 8.8. Replace M_5 and M_6 by poly resistors that are 100 MΩ in value. Explain how this new 4T cell works for read and write operations. How does the internal node get pulled to a high value? Is the new cell static or dynamic?

P8.8. For the sense amplifier shown in Figure 8.20, answer the following questions using 0.18 μm technology parameters:

(a) If the sense amplifier is driving a load of 50 fF in 500 ps, what is the required value of I_{SS}?

(b) In order to turn off the input transistors with a bitline swing of 100 mV, what values of W/L are needed?

(c) Which of the three column pull-up configurations of Figure 8.13 would be used with this sense amplifier? What is the initial voltage at the inputs to the sense amp?

(d) Given the size of the input transistors, what is the steady-state voltage at the gate node and the resulting size of M_5?

(e) Choose the sizes of M_3 and M_4 to establish a suitable steady-state output voltage.

DESIGN PROBLEM 1: Decoder Design in 0.18 μm Technology

Your task is to create a decoder for a small memory shown in Figure D8.1. The memory consists of 1K words, each containing 64 bits, arranged as a 256 $\times$ 256 array of 6T SRAM cells. A word is selected for reading by the row and column decoders. The row decoder asserts one of the 256 wordlines. Each Metal2 wordline selects a row of four 64-bit words. Four selected words feed the input of the 4:1 MUX controlled by the outputs of the column decoder.

The SRAM receives a 10-bit address. The inputs to the row decoder are the upper 8 bits of the address, while the lowest 2 bits of the address feed the column decoder. Each access transistor in the memory cell is 4λ in width. Each SRAM cell is $40\lambda \times 40\lambda$. The diagram in Figure D8.1 illustrates the SRAM array, with a blow-up of one of the SRAM cells in the forty-second word.

The spec allows a maximum of 5 fF of gate capacitance on any address input. You can assume that the clock period is 24FO4 delays. Your task is to minimize the delay from the address becoming stable to the wordline rising. The general topology for the decoder is provided in Figure D8.2. Your design goal is to implement an optimal row decoder that converts 8 address bits into 256 wordlines using logical effort.

(a) What is the total load on one wordline? Include both the loading of the SRAM cells and the capacitance of a Metal2 wordline. Assume a wire capacitance of 0.2 fF/μm. You can assume that gate capacitance is 2 fF/μm.

(b) Find the total path effort in the address decode path by estimating the fanout from input (address bit) to output (wordline), the branching factor, and the logical effort. Estimate the number of stages needed in the optimal design.

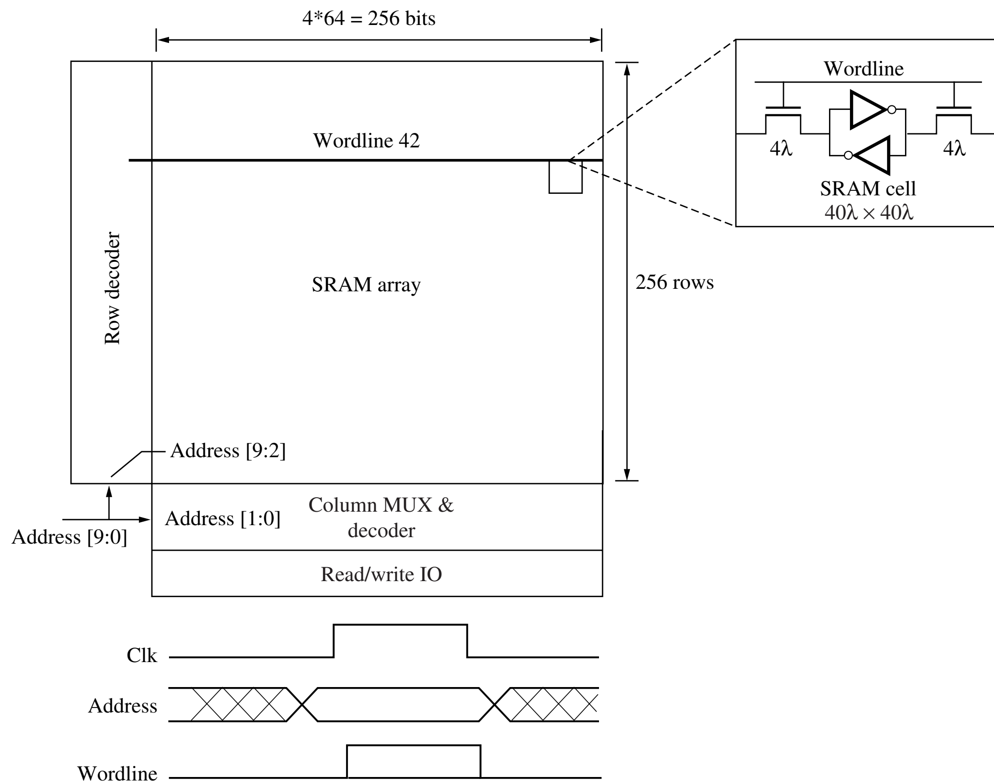


Figure D8.1
SRAM layout and timing information.

(c) You can implement the predecoder in either 2 or 4 stages as shown in Figure D8.2. Which of the two approaches is better? Using logical effort, decide which architecture you are going to use. For both cases, the final decode stage is a NAND2-INV combination. Ignore any sideload capacitances.

(d) Next choose the correct design for the decoder, and size the gates to optimize performance. What is your estimate of the delay of the decoder in terms of FO4 delay (include the delay from the parasitics of the gates in your estimate)?

(e) Now include the sideload at the outputs of the predecoder stage. The sideload is due to the wire running vertically in Figure D8.2. Compute the actual sizes when the sideload is included.

(f) Compare the hand-calculated delay against SPICE.

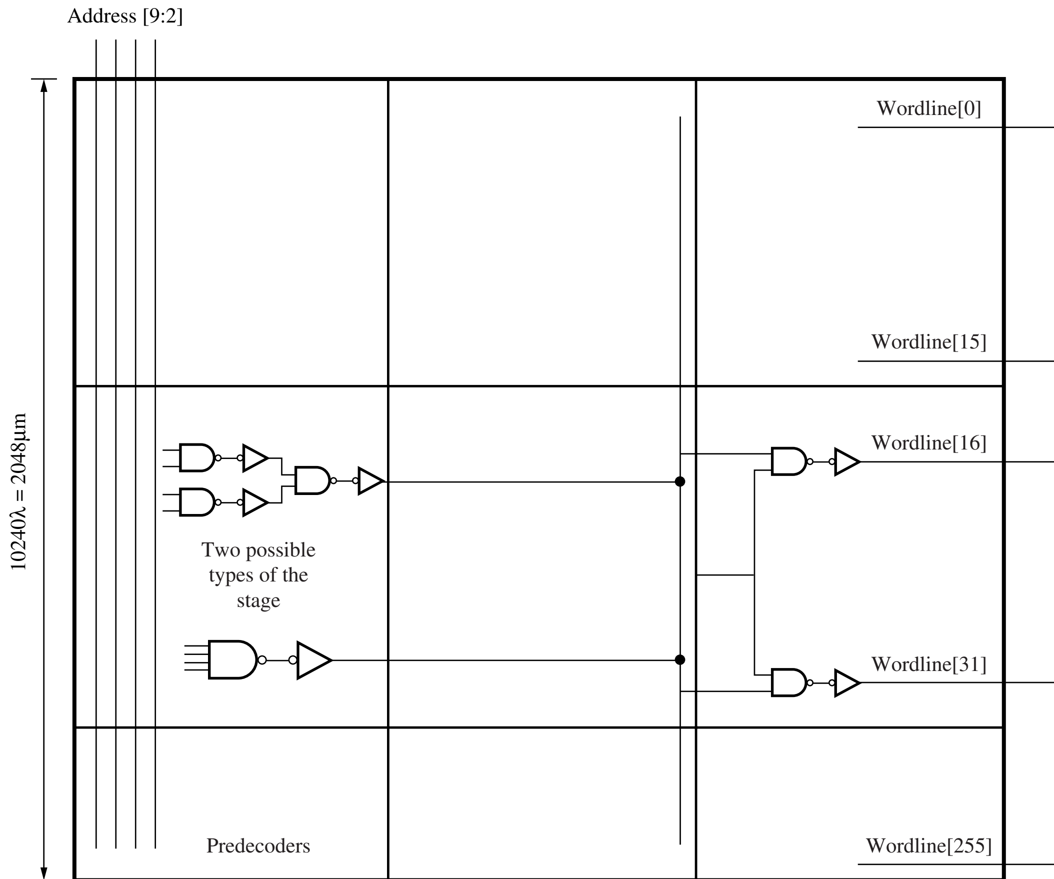


Figure D8.2
Decoder topology, branching, and wire loads.

DESIGN PROBLEM 2: Design of SRAM Cell and Read/Write Circuitry in 0.18 μm Technology

The problem explores cell and I/O design issues for a 64K SRAM based on the previous problem. Since there are 256 columns in the SRAM and 64 bits/word, it requires 64 sense amplifiers (1 per bit of output) and each one is driven by 1 of 4 possible columns (one column from each interleaved 64-bit word).

(a) First consider the cell design. Recall that in Design Problem 1, the access transistor of the SRAM cell was given as 4λ. Size the rest of the transistors to satisfy the following requirements: (1) During a read, the internal voltage should not rise above V_{TN}, (2) During a write, the access transistor must be able to pull the internal node to at least 0.8 V (i.e., well below V_S

of the inverter), and (3) Make the transistors as small as possible. One assumption you could make is that the bitline is around 0 V during the write operation. For the read operation, assume that the bitline acts like a supply voltage at V_{DD}. Rules of thumb are available for the design but you may need to adjust these values to make the cell work properly. All transistors should be specified in integer units of λ. The minimum channel length/width is 2λ.

(b) Estimate V_S of the inverters used in the cell. Why is this number important in the cell design? Use SPICE to verify the dc parameters you calculated. Provide both V_{OL} for the read operation and V_S as measured in HSPICE. Adjust any device sizes as necessary to achieve the design specifications.

(c) Using hand calculations, estimate the current that the cell can draw from the bitline when the wordline goes high. Use an average between the initial value and the steady-state value to estimate this current.

(d) Next, calculate the capacitance of the bitline. There are 256 rows of cells and, from Design Problem 1, each cell is $40\lambda \times 40\lambda$.The column pull-up transistors are each 50λ. Assume that each contact has a capacitance of 1 fF.

(e) Next compute the time required to discharge one of the bitlines by 180 mV.

(f) The last part of the read cycle involves the sense amplifier. The design of the sense amp has been provided in P8.8 using Figure 8.20. Compute the discharge time assuming that the bias current is 300 μA and the output capacitance is 50 fF. Assume a voltage swing at the output of the sense amp of 0.9 V.

(g) The next step of the design is the write circuit of Figure 8.18. To write the cell, the bitline has to be driven low enough so that the cell switches. Ideally, the write time should be about the same speed as the read time. If we work to make it faster, we are wasting our effort because the read will be the limiting factor. If it is slower than the read, then the write is in the critical path and that is not desirable either. Using a simple *RC* model, determine the effective resistance needed to get the bitlines to swing down to $V_{DD}/5$ in the same amount of time it takes to do the read. Size M_{13}, M_{14}, and M_{15} so that they have this effective resistance.

(h) As a last step, we need to set the values of the column pull-up transistors of Figure 8.18. They need to pull the bitlines up after a write but before the end of the precharge phase. How big do they need to be? The final difference of the bitlines should be less than 10% of the desired read swings to avoid confusing the sense amp. Typically, we make them twice the size of the pull-down devices for a fast precharge. Size these transistors accordingly.

(i) Check your results using SPICE.