FUTURE VISION BIE

One Stop for All Study Materials

& Lab Programs

Future Vision

By K B Hemanth Raj

Scan the QR Code to Visit the Web Page

Or

Visit : https://hemanthrajhemu.github.io

Gain Access to All Study Materials according to VTU,
CSE – Computer Science Engineering,
ISE – Information Science Engineering,
ECE - Electronics and Communication Engineering
& MORE...

Join Telegram to get Instant Updates: https://bit.ly/VTU_TELEGRAM

Contact: MAIL: futurevisionbie@gmail.com

INSTAGRAM: www.instagram.com/hemanthraj_hemu/

INSTAGRAM: www.instagram.com/futurevisionbie/

WHATSAPP SHARE: https://bit.ly/FVBIESHARE

Eastern Economy Edition

Third Edition

Basic VLSI Design

Douglas A. Pucknell
Kamran Eshraghian

PHI

Chapter 9

Memory, Registers and Aspects of System Timing

Ay, now the plot thickens very much upon us.

— GEORGE VILLIERS, 2nd Duke of Buckingham

OBJECTIVES

The 4-bit data path design continues with the 4 × 4-bit register array. This raises the subject of memory/storage elements and techniques. Some of the possible dynamic and static memory cells are presented and key properties compared. The concept of an array of memory cells is extended to include RAM arrays and some of the needs for selection and control are explored.

Two of the subsystems of the 4-bit data path (as in Figures 7.3 and 8.1) having already been designed, it is now appropriate to consider the register arrangements in which the 4-bit quantities to be presented to the adder and shifter will be stored. The question of data storage is an important one which has already been mentioned a number of times. It raises the question of the choice of storage elements or memory cells as well as the questions of configuring arrays of such cells and the selection of a given cell or group of cells in an array.

Before looking at register arrangements, we should set out some ground rules for the design of the 4-bit processor. It is essential that such rules should be established early in the piece so that a uniform approach to 'reading, writing and refresh' is adhered to throughout. In practice, such rules would have been set out much earlier than this, but our progress through this text is such that in this case they are most effectively established here and would not have meant much earlier on.

9.1 SYSTEM TIMING CONSIDERATIONS

1. A two-phase non-overlapping clock signal is assumed to be available, and this clock alone will be used throughout the system.
2. Clock phases are to be identified as ϕ_1 and ϕ_2 where ϕ_1 is assumed to lead ϕ_2.

3. Bits (or data) to be stored are *written to* registers, storage elements, and subsystems on ϕ_1 of the clock; that is, write signals *WR* are *Anded* with ϕ_1.
4. Bits or data written into storage elements may be assumed to have settled before the immediately following ϕ_2 signal, and ϕ_2 signals may be used to *refresh* stored data where appropriate.
5. In general, delays through data paths, combinational logic, etc. are assumed to be less than the interval between the leading edge of ϕ_1 of the clock and the leading edge of the following ϕ_2 signal.
6. Bits or data may be *read* from storage elements on the next ϕ_1 of the clock; that is, read signals *RD* are *Anded* with ϕ_1. Obviously, *RD* and *WR* are generally mutually exclusive to any one storage element.
7. A general requirement for system stability is that there must be at least one clocked storage element in series with every closed loop signal path.

Strict adherence to a set of rules such as this will greatly simplify the task of the system designer and also help to avoid some of the disasters which will almost certainly occur if a haphazard approach is taken.

9.2 SOME COMMONLY USED STORAGE/MEMORY ELEMENTS

Everyone complains of his memory, but no one complains of his judgment.

— DUC DE LA ROCHEFOUCAULD

In order to make a comparative assessment of some possible storage elements, we will consider the following factors:

- area requirement;
- estimated dissipation per bit stored;
- volatility.

9.2.1 The Dynamic Shift Register Stage

One method of storing a single bit is to use the shift register approach previously introduced in section 6.5.4 (and also Figures 3.14, 3.17, 6.36, 6.37, 6.38, 6.39 and 6.40).

9.2.1.1 Area

This calculation applies to an nMOS design, as in Figure 6.40(a), with buried contacts. Allowing for the sharing of V_{DD} and *GND* rails between adjacent rows of register cells, each bit stored will require

$$(22\lambda \times 28\lambda) \times 2 \doteq 1200\lambda^2$$

For example, for $\lambda = 2.5\ \mu m$

$$\text{Area/bit} = 7500\ \mu m^2$$

To give an idea of what this implies, such area requirements would result in a maximum number of bits stored on a 4 mm × 4 mm chip area ≈ 2.1 k bits.

For a CMOS design, as in Figure 6.40(b), and allowing for the sharing of V_{DD} and V_{SS} rails between adjacent rows of register cells, each bit stored will require

$$(38\lambda \times 28\lambda) \times 2 \div 2100\lambda^2$$

For example, for $\lambda = 2.5\ \mu m$

$$\text{Area/bit} \approx 13{,}000\ \mu m^2$$

Such area requirements would result in a maximum number of bits stored on a 4 mm × 4 mm chip area ≈ 1.2 k bits.

9.2.1.2 Dissipation

In the case of CMOS designs, the static dissipation is very small and calculation at this stage will not be meaningful since only the switching dissipation will be significant (particularly at high speeds). This dynamic power consumption P_d can be written as

$$P_d = m.(C_L.V_{DD}^2.f)$$

where m is the duty cycle, C_L is effective load capacitance and f is the clock frequency.

In the nMOS case we can readily calculate the static dissipation, noting that in practice the switching dissipation would add to this. Each inverter stage has a ratio of 8:1 and if the layout of Figure 6.40(a) (buried contacts) is used, then, noting that one inverter of the pair must always be 'on',

$$Z_{p.u.} = 4R_s$$

and

$$Z_{p.d.} = \frac{1}{2}R_s$$

Therefore

$$\text{Current} = \frac{V_{DD}}{Z_{p.u.} + Z_{p.d.}} = \frac{5\text{ V} \times 10^6}{4.5 \times 10^4} = \frac{500}{4.5}\mu\text{A} \approx 110\ \mu\text{A}$$

Therefore

$$\frac{\text{Static dissipation}}{\text{Bit stored}} = V_{DD} \times \text{current} = 5\text{ V} \times 110\ \mu\text{A} = 550\ \mu\text{W}$$

Thus, 2.1 k bits on a single chip would dissipate

$$2.1 \times 10^3 \times 550 \times 10^{-6} = 1.15 \text{ watts}$$

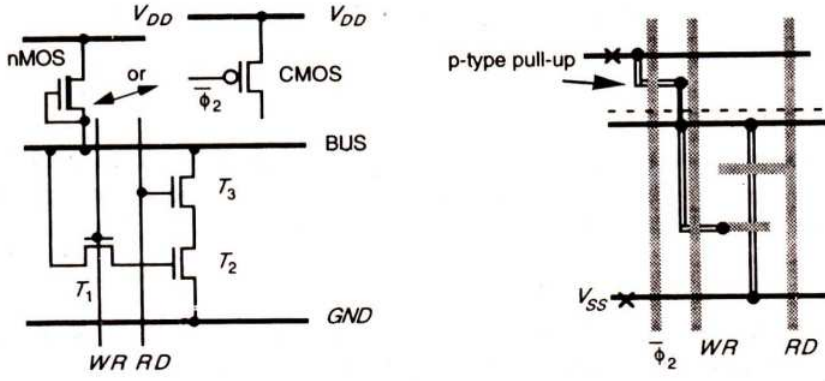
Dissipation can be reduced by using alternative geometry, but this is at the expense of increased area.

9.2.1.3 Volatility

Data is stored by the charge on the gate capacitance of each inverter stage, so that data storage time (without refresh) is limited to 1 msec or less.

9.2.2 A Three-transistor Dynamic RAM Cell

An arrangement which has been used in RAM (random access memory) and other storage arrangements is set out in Figure 9.1.



(a) Circuit

(b) CMOS stick diagram

Note: *WR* and *RD* are coincident with ϕ_1.

FIGURE 9.1 Three-transistor dynamic memory cell.

With regard to Figure 9.1(a), the action is as follows:

1. With the *RD* control line in the Lo state, then a bit may be read from the bus through T_1 by taking *WR* to the Hi state so that the logic level on the bus is communicated to the gate capacitance of T_2. Then *WR* is taken Lo again.
2. The bit value is then stored for some time by C_g of T_2 while both *RD* and *WR* are Lo.
3. To read the stored bit it is only necessary to make *RD* Hi and the bus will be pulled down to ground through T_3 and T_2 if a 1 was stored. Otherwise, T_2 will be non-conducting and the bus will remain Hi due to its pull-up arrangements.

Note that the complement of the stored bit is read onto the bus, but this presents few problems and can be taken care of at some common point in the memory array.

A stick diagram for the cell identified in Figure 9.1(a) is presented as Figure 9.1(b), and possible mask layouts follow in Figure 9.2. Note that this figure gives both nMOS and CMOS designs.

To return to our main theme, it is now appropriate to assess the three-transistor cells in the same manner as the previous one.

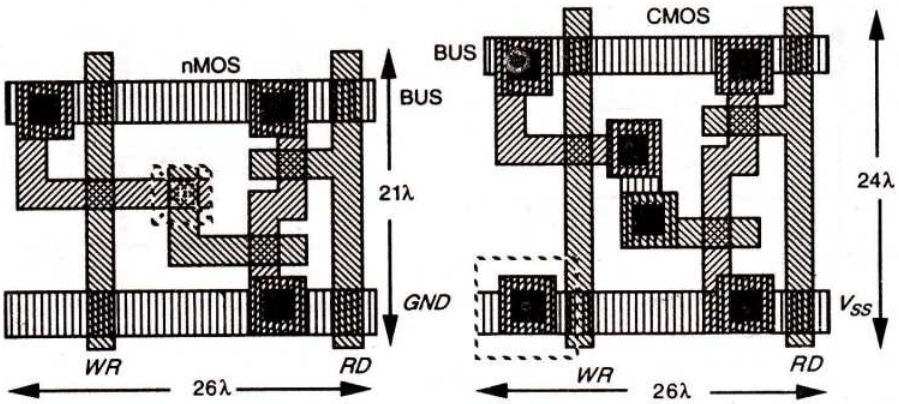


FIGURE 9.2 Mask layouts* for three-transistor (nMOS and CMOS) memory cell *(pull-ups not shown).

9.2.2.1 Area

From the layout it will be seen that an area of more than $500\lambda^2$ is required for each bit stored (less if *GND* (V_{SS}), and/or bus, and/or control lines are shared with other cells). Thus, for $\lambda = 2.5$ μm.

$$\text{Area/bit} \doteqdot 3000\ \mu m^2$$

Thus, to continue the previous example, the maximum number of bits which could be accommodated on a 4 mm × 4 mm silicon chip is > 4.8 k bits.

9.2.2.2 Dissipation

Static dissipation is nil since current flows only when *RD* is Hi and a 1 is stored. Thus, the actual dissipation associated with each bit stored will depend on the bus pull-up and on the duration of the *RD* signal and on the switching frequency.

9.2.2.3 Volatility

The cell is 'dynamic' and will hold data only for as long as sufficient charge remains on C_g (of T_2).

9.2.3 A One-transistor Dynamic Memory Cell

The area occupied by each bit stored in each of the previous cases is quite considerable, which clearly limits, say, the number of bits which could be stored on a single chip of reasonable size.

Various approaches have been taken to reduce the area per bit requirements and one such approach is the one-transistor cell as shown in Figure 9.3. The concept of the single transistor cell is quite simple, as may be seen from Figure 9.3(a). It basically consists of a capacitor C_m which can be charged during '*write*' from the *read/write* line, provided that the *row select* line is Hi. The state of the charge C_m can be read subsequently by detecting the

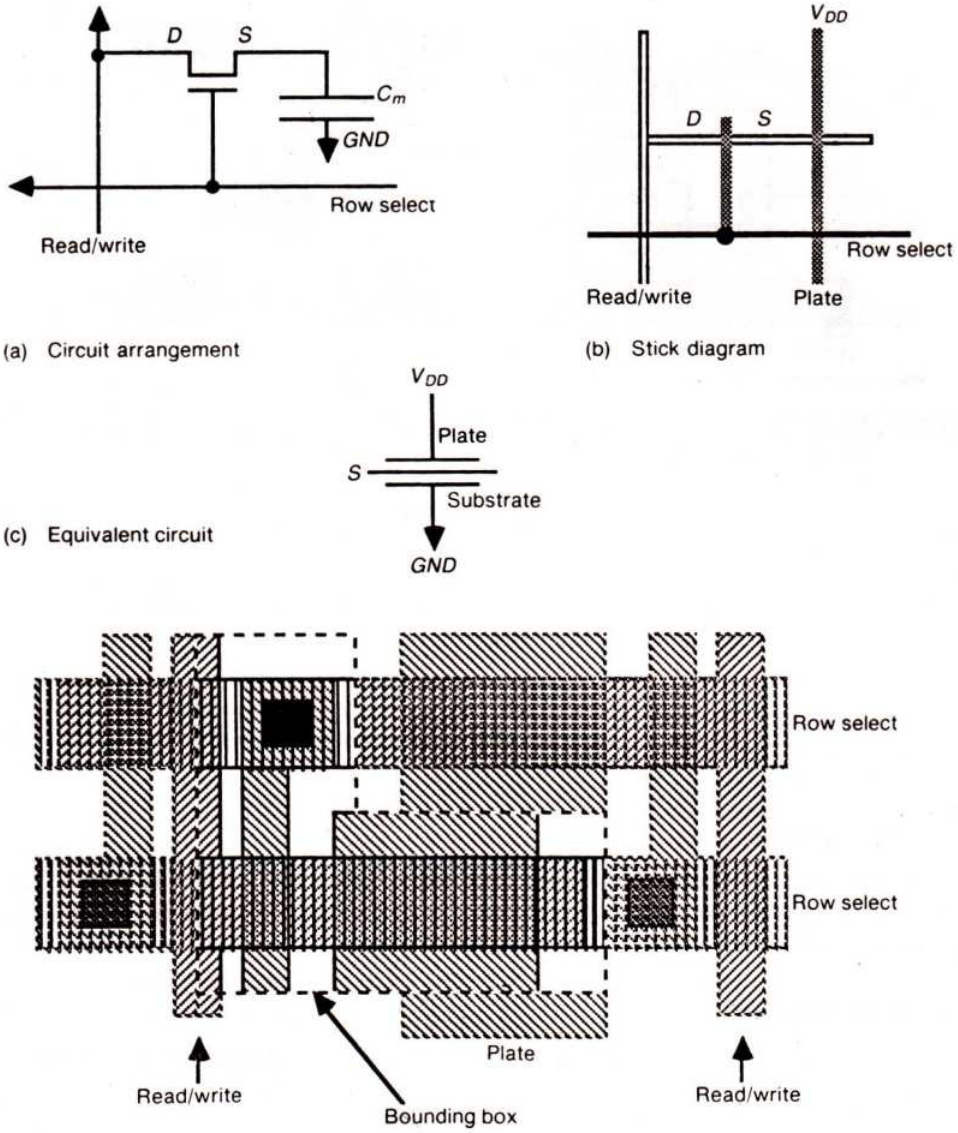


FIGURE 9.3 One-transistor memory cell.

state of the charge via the same *read/write* line with the *row select* line Hi again, and a sense amplifier of a suitable nature can be designed to differentiate between a stored 0 and a stored 1.

However, in practice the cell is slightly more complex than first considerations might suggest, since special steps must be taken to ensure that C_m has sufficient capacitance to allow ready detection of the stored content.

The most obvious and readily fabricated C_m in the structure under consideration would be to extend and enlarge the diffusion area comprising the source (S) of the pass transistor in Figure 9.3(b). We would then rely on the junction capacitance between the n-diffusion region and the p-substrate to form C_m. However, if we consult Table 4.2 (which gives capacitance values for a typical MOS process), we will see that the capacitance per unit area of diffusion is much less than the capacitance per unit area between gate and channel (i.e. between the channel under the thin gate oxide and the polysilicon gate area).

If we use the diffusion to substrate capacitance alone, a comparatively large area will be required to give any significant value of capacitance; for example, at least $16\lambda^2$ will be needed to give a capacitance equal to $1\square C_g$ (i.e. 0.01 pF in the 5 μm MOS process being considered). A solution is to create a much more significant capacitor by using a polysilicon plate (which is connected to V_{DD}) over the diffusion area. Thus, C_m is formed as a three-plate structure as indicated in Figure 9.3(c). For example, for the area given in Figure 9.3(d), $C_{Diff\text{-}Poly.} = 100\square C_g$ (= 0.1 pF), while the contribution from the diffusion region to the substrate will be much smaller but will add some 25% to this figure, giving a total C_m of 0.125 pF for the layout considered. Even so, careful design is necessary to achieve consistent readability.

9.2.3.1 Area

The area enclosed to indicate the standard cell in Figure 9.3(d) is $200\lambda^2$. Thus for $\lambda = 2.5$ μm, area/bit stored = $200\lambda^2 = 1250$ μm^2.

Therefore, the number of bits per 4 mm × 4 mm chip area is approximately 12 k bits (allowing some 'overheads' for sensing, etc.).

9.2.3.2 Dissipation

There is no static power associated with the cell itself, but there must be an allowance for switching energy while writing to and reading from the storage elements.

9.2.3.3 Volatility

Quite obviously, leakage current mechanisms will deplete the charge stored in C_m and thus the data will be held for only up to 1 msec or less. Therefore, periodic refresh operations must be provided. It will also be realized that reading the cell is a destructive operation and that the stored bit must be rewritten every time it is read.

9.2.4 A Pseudo-static RAM/register Cell

So far, all the storage elements considered have been volatile and thus have an implied need to be periodically refreshed. This is not always convenient and it is necessary to consider the design of a static storage cell which will hold data indefinitely. A common way of meeting this need is to store a bit in two inverter stages in series with feedback, say, on ϕ_2 to refresh the data every clock cycle. Circuit arrangements are shown in Figures 9.4(a) and 9.5(a) and it will be seen that a bit may be written to the cell from the bus by energizing the *WR* line. From our system timing consideration of section 9.1, we will assume *WR* to occur in coincidence

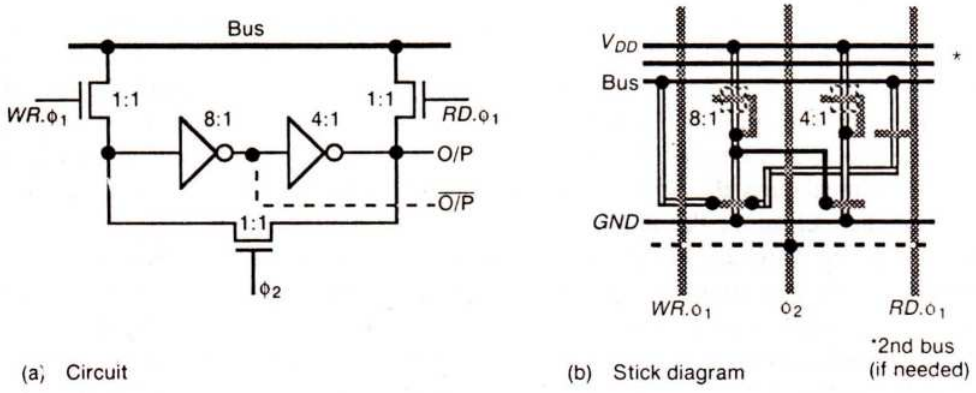


FIGURE 9.4 nMOS pseudo-static memory cell.

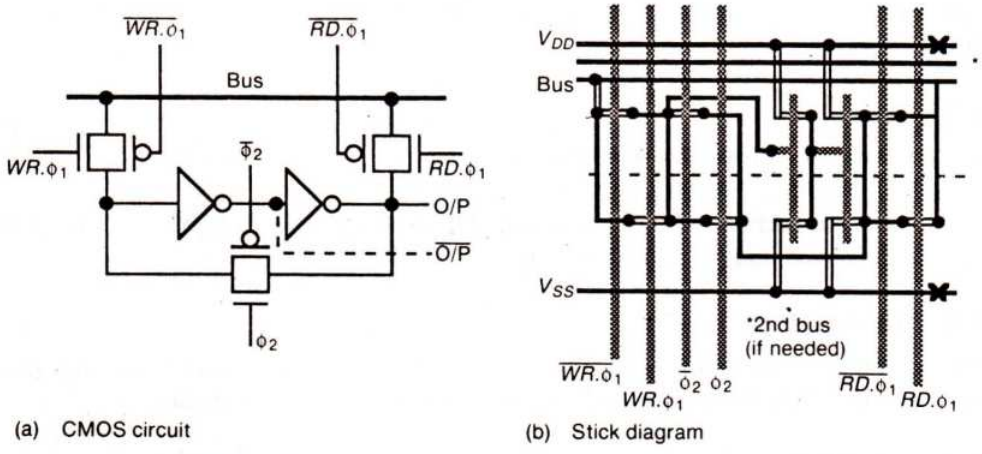


FIGURE 9.5 CMOS pseudo-static memory cell.

with ϕ_1 of the clock. Thus, the bit is stored on C_g of inverter 1 and will be reproduced complemented at the output of inverter 1 and true at the output of inverter 2. It will be seen that during every ϕ_2 of the clock the stored bit is refreshed through the gated feedback path from the output of inverter 2 to the input of inverter 1. Thus the bit will be held as long as ϕ_2 of the clock recurs at intervals less than the decay time of the stored bit. To read the state of the cell it is only necessary to energize the *RD* line, which is also assumed coincident with ϕ_1 of the clock, and the bit will be read onto the bus.

Note that:

1. *WR* and *RD* must be mutually exclusive (but are both coincident with ϕ_1).
2. If ϕ_2 is used for refresh, then the cell must not be read during ϕ_2 of the clock unless the feedback path is inhibited during *RD*. If an attempt is made to read the cell onto the bus during refresh, then charge sharing effects between the bus and input (C_g) capacitances may cause the destruction of the stored bit.

3. Cells must be stackable, both side by side and top to bottom. This must be carefully considered together with the overall strategy to be observed when the layout is drawn.
4. Allow for other bus lines to run through the cell so that register and memory arrays are readily configured.

With these factors in mind, it is possible to draw up stick diagrams as in Figures 9.4(b) and 9.5(b), which show the nMOS and CMOS basic cells.

Mask level layouts follow from this; a possible layout for an nMOS cell which can be written to from bus *A* and can be read onto bus *A* or bus *B* is given in Figure 9.6.

The mask layout shown in Figure 9.6 occupies an area of $59\lambda \times 45\lambda = 2655\lambda^2$, but if we are considering a single bus and a more compact layout then the area requirement can be reduced to about $1750\lambda^2$ or less. The CMOS version of this cell is *not* really a practical proposition other than for storing a few bits. Ten transistors are needed per bit stored, which makes the cell too demanding in area to be the basis of larger memories. We will therefore evaluate only the nMOS version of the cell and, to return to the original purpose, we may now set out the relevant parameters for this popular and useful pseudo-static storage cell in the same terms as have been used previously.

9.2.4.1 Area

A typical area for a nMOS single cell with single bus is in the region of $1750\lambda^2$. Therefore, for $\lambda = 2.5\ \mu m$

$$\text{Area/bit} \approx 10000\ \mu m^2$$

Thus, the maximum number of bits of storage per 4 mm × 4 mm chip is approximately 1.4 k bits.

9.2.4.2 Dissipation

The nMOS cell uses two inverters, one with an 8:1 and the other with a 4:1 ratio. Dissipation will depend on the current drawn and thus on the actual geometry of the inverters, but let us assume that inverters are based on minimum feature size gate areas so that the 8:1 stage will present a resistance of 90 kΩ and the 4:1 stage a resistance of 50 kΩ between the supply rails. Now when one stage is off, the other is on so that, say, each spends half-time in the conducting state. Therefore

$$\text{Average current} = 0.5\left(\frac{5\text{ V}}{90\text{ k}\Omega} + \frac{5\text{ V}}{50\text{ k}\Omega}\right) \approx 80\ \mu\text{A}$$

Therefore dissipation per bit stored = 80 μA × 5 V = 400 μW. Thus 1.4 kbits on a single chip would dissipate 560 mW.

9.2.4.3 Volatility

The cell is non-volatile provided that ϕ_2 signals are present.

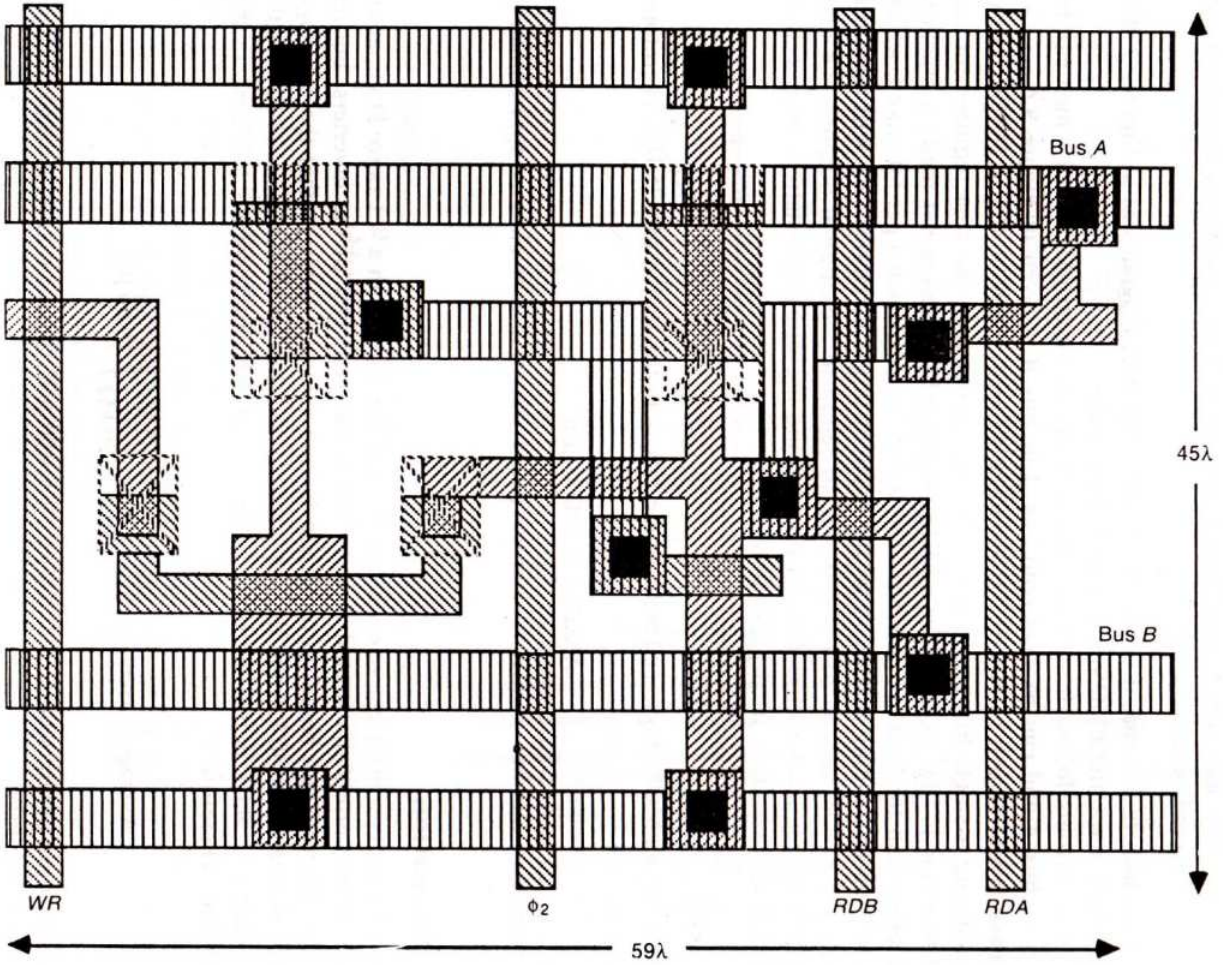


FIGURE 9.6 nMOS pseudo-static memory cell with read to either of two buses.

9.2.5 Four-transistor Dynamic and Six-transistor Static CMOS Memory Cells

Most of the preceding memory cells involved n-type transistors and can therefore be implemented in either nMOS or CMOS designs. The cells about to be described utilize both n-type and p-type transistors and are therefore intended for CMOS systems only (although the dynamic element can be readily adapted to nMOS-only implementation).

Both the dynamic and static elements, set out in Figure 9.7, use a two bus per bit arrangement so that associated with every bit stored there will be a *bit* and a $\overline{bit}$ bus as shown. In both cases the buses are precharged to logic 1 before read or write operations take place.

Figure 9.7(a) gives the arrangement for a four-transistor dynamic cell for storing one bit. Each bit is stored on the gate capacitance of two n-type transistors T_1 and T_2 and a description of the write and read operation follows.

9.2.5.1 Write operations

Both *bit* and $\overline{bit}$ buses are precharged to V_{DD} (logic 1) in coincidence with ϕ_1 of an assumed two-phase clock. Precharging is effected via the p-transistors T_5 and T_6 in the diagram. Now (with reference to Figure 9.7(c), the appropriate '*column select*' line is activated in coincidence with the clock phase ϕ_2 and either the *bit* or $\overline{bit}$ line is discharged by the logic levels present on the I/O *bus* lines, the I/O lines acting in this case as a current sink when carrying a logic 0. The '*row select*' signal is activated at the same time as '*column select*' and the bit line states are 'written in' via T_3 and T_4 and stored by T_1 and T_2 as charges on gate capacitances C_{g2} and C_{g1} respectively. Note that the way in which T_1 and T_2 are interconnected will force them into complementary states while the *row select* line is high. Once the select lines are deactivated, the bit stored will be remembered until the gate capacitances lose enough charge to drop the 'on' gate voltage below the threshold level for T_1 or T_2.

9.2.5.2 Read operations

Once again both *bit* and $\overline{bit}$ lines are precharged to V_{DD} via T_5 and T_6 during ϕ_1 so that both lines will be at logic 1. Now if, say, a 1 has been stored, T_2 will be on and T_1 will be off, and thus the $\overline{bit}$ line will be discharged to V_{SS} (logic 0) through T_2 and the stored bit thus reappears on the bit lines.

When such cells are used in RAM arrays, it is necessary to keep the area of each cell to a minimum and transistors will be of minimum size and therefore incapable of sinking large charges quickly. Thus it is important that the charges stored on the bit lines be modest and this may not be the case if they are directly paralleled by the I/O line and other associated capacitances through the column select circuitry. RAM arrays therefore generally employ some form of sense amplifier. A possible arrangement is shown in Figure 9.7(c) in which T_1, T_2, T_3 and T_4 form *a flip-flop circuit*. If we assume the sense line to be inactive, then the state of the bit lines is reflected in the charges present on the gate capacitances of T_1 and T_3 with respect to V_{DD} such that a 1 will turn off and a 0 turn on either transistor. Current flowing from V_{DD} through an on transistor helps to maintain the state of the bit lines and predetermines

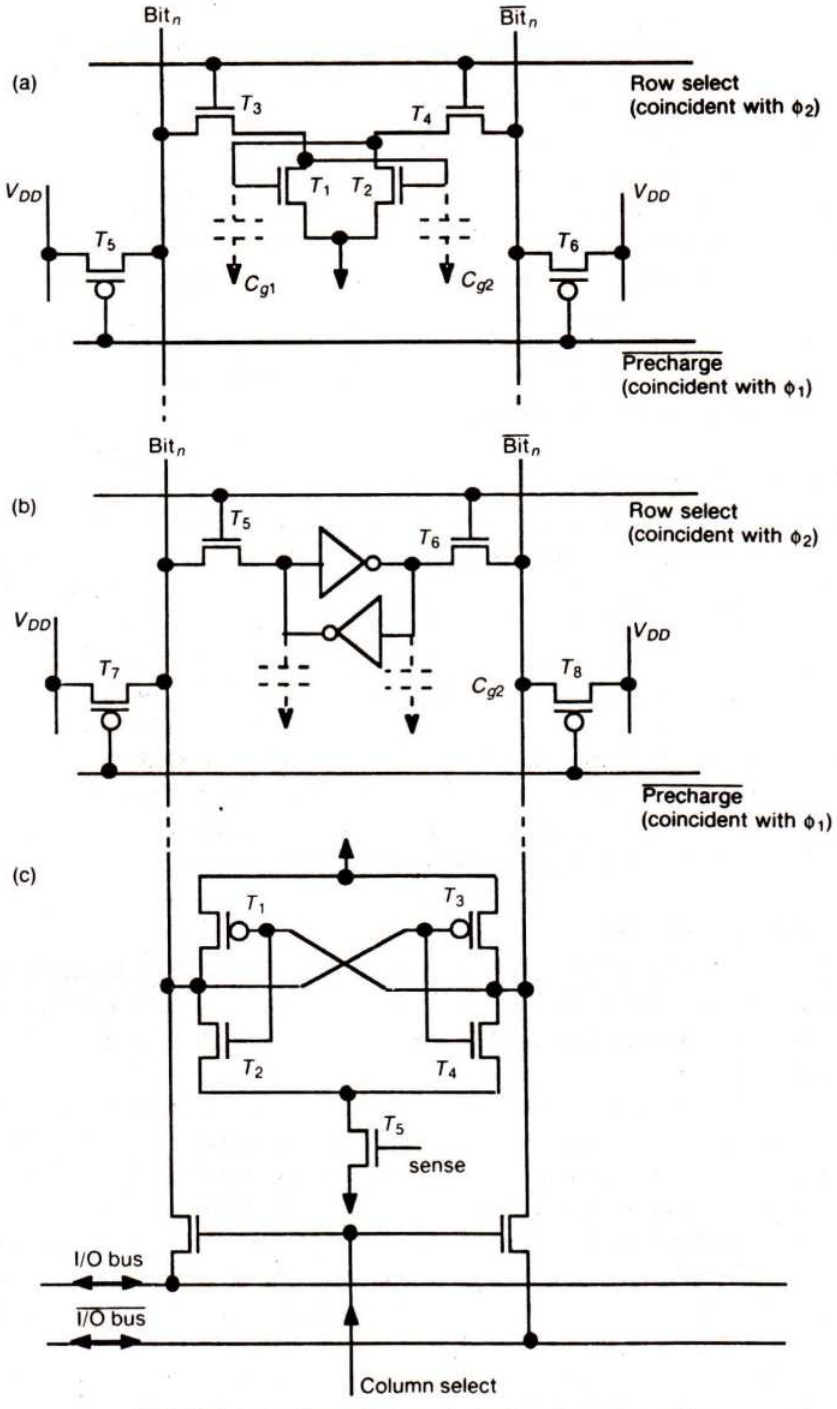


FIGURE 9.7 Dynamic and static memory cells.

the state which will be taken up by the sense flip-flop when the sense line is then activated. The geometry of the single sense amplifier per column will be such as to amplify the current sinking capability of the selected memory cell.

Figure 9.7(b) indicates an adaption of the basic dynamic cell, just considered, to form a static memory cell. At the expense of two additional transistors per bit stored, the transistors T_1 and T_2 of Figure 9.7(a) can each be replaced by an inverter as shown in Figure 9.7(b). This arrangement will clearly render the cell static in its data-storing capabilities.

The general arrangement of a RAM utilizing the circuits considered here appears later in this chapter (Figure 9.18).

9.2.6 JK Flip-flop Circuit

No consideration of memory elements would be complete without the JK flip-flop. The JK flip-flop is a particularly widely used arrangement and is an example of a static memory element. It is also most useful in that other common arrangements such as the D flip-flop and the T flip-flop are readily formed from the JK arrangement. Edge-triggered circuits are conveniently designed with an ASM (algorithmic state machine) approach—see C.Clare, *Designing Logic Systems Using State Machines*, McGraw-Hill, 1983)—and the design equations for a JK flip-flop, as in Figure 9.8, follow from an ASM chart setting out the requirements as in Figure 9.9. It should be noted that the flip-flop is assumed to have an asynchronous clear (*Clr*) input as well as the clocked J and K inputs, and that J and K are read in during the Hi level of the clock ϕ, and the data thus read is transferred to the output on the falling edge of ϕ.

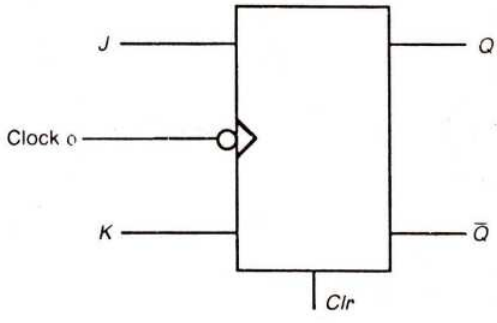


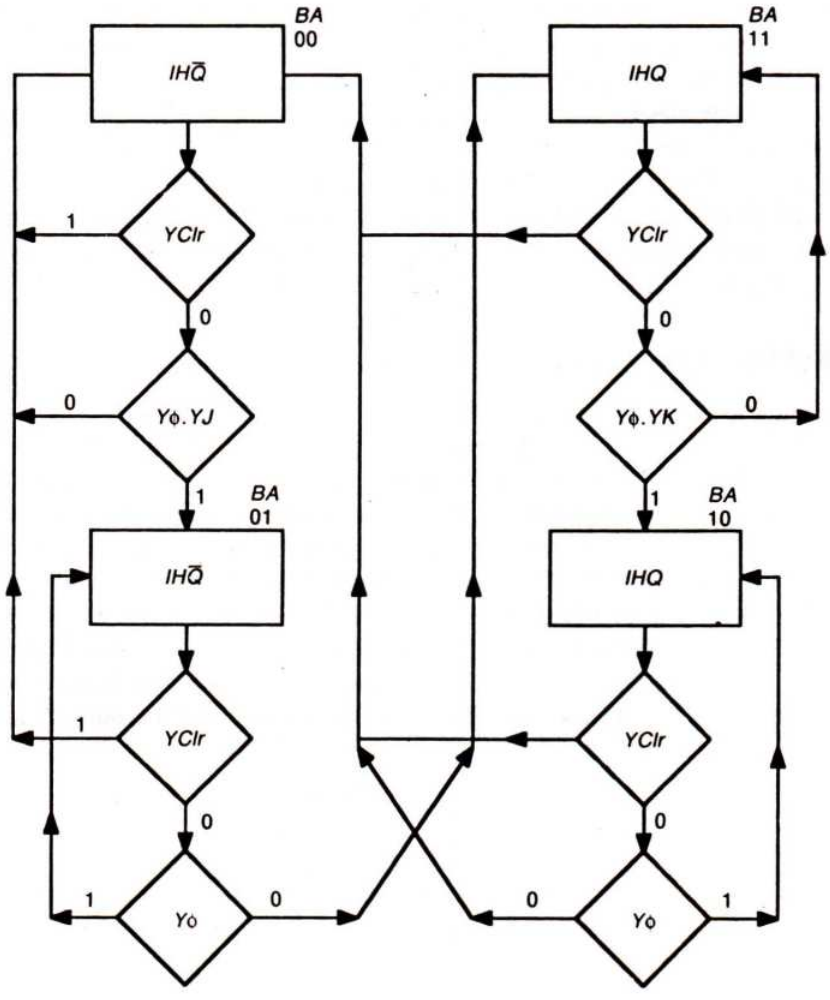
FIGURE 9.8 JK flip-flop.

Design equations are readily derived from the ASM chart of Figure 9.9 and, making the secondary variable assignments (AB in the figure), we may express the requirements as follows:

$$A = a.(\overline{Clr}).(\bar{b} + \bar{\phi} + \bar{K}) + \bar{b}.(\overline{Clr}).J.\phi$$

$$B = (\overline{Clr}).(a.\bar{\phi} + b.\phi)$$

where output $Q = B$, and a and b are the fed back state of the secondary variables A and B respectively.



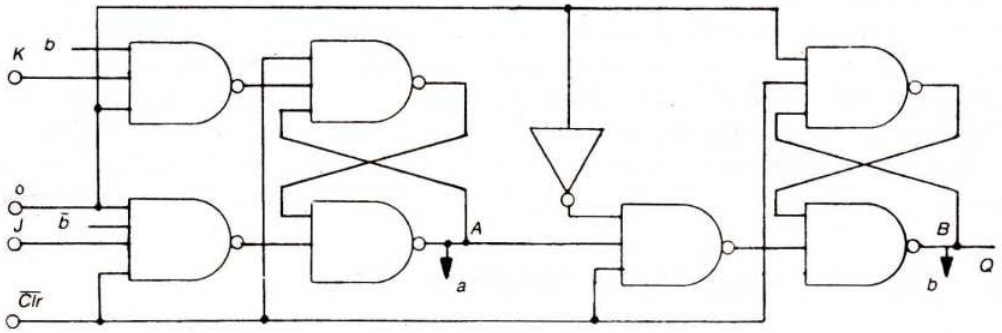
Note: *IHQ* = *Q* output 'Hi' (Immediate)
YClr = Yes clear
Yϕ.YJ = Yes clock and *J*, . . ., etc.

FIGURE 9.9 ASM chart for JK flip-flop.

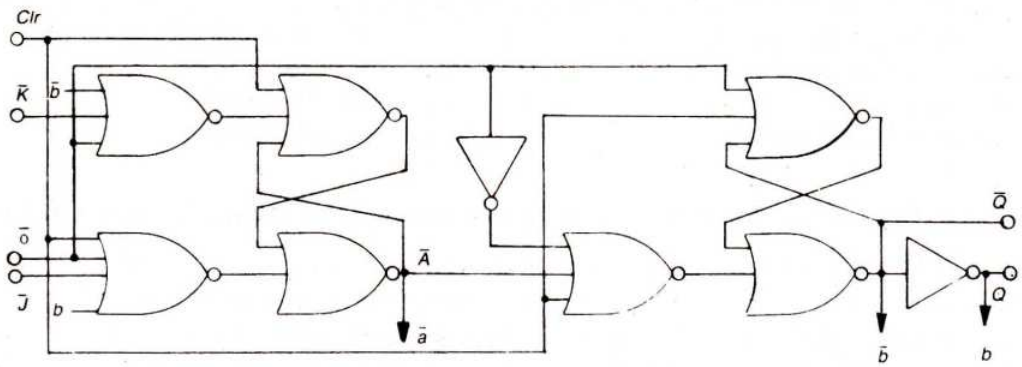
9.2.6.1 Logic gate implementations

We are now faced with a choice of implementations based on *Nand* or *Nor* or switch logic. The expressions for *A* and *B* are readily realized in *Nand* or *Nor* logic, as shown in Figure 9.10, and it will be seen that a master/slave arrangement applies in each case.

However, an initial consideration of each arrangement will reveal that, for nMOS, the *Nand arrangement is impractical*, owing to the relatively large number of gates requiring three or more inputs which will therefore be inherently large in area and slow in performance. The obvious nMOS alternative is a *Nor* gate arrangement which is a practical proposition and can be readily implemented.



(a) *Nand* gate version



(b) *Nor* gate version

FIGURE 9.10 Logic arrangement for JK flip-flop.

For CMOS, both *Nand* and *Nor* gates are suitable although the *Nor* gate is generally slower.

9.2.6.2 Switch logic and inverter implementation

In setting out an arrangement of n pass transistors to realize the logical requirements, we must bear in mind earlier considerations on the nature of switch logic networks: that is, there should be no more than four pass transistors in series (section 4.9); pass transistors are not to be used to drive the gates of other pass transistors; the logic 0 as well as the logic 1 transmission conditions are to be deliberately satisfied. Thus, we need to implement the expressions for $\bar{A}$ and $\bar{B}$ *as well as* the expressions for A and B given earlier in this section. The resulting arrangement is given at Figure 9.11 and is a realization of the JK flip-flop based on n-pass transistor logic and inverters only.

9.2.7 D Flip-flop Circuit

A D flip-flop is readily formed from a JK flip-flop by renaming the J input D and then replacing connections to K by $\bar{D}$ (see Figure 9.10). Similarly, a T(Toggle) flip-flop is formed from the JK by making $J = K = E$, where E is the toggle enabling input.

It should also be noted that the arrangements given may be simplified by the omission of the *Clr* input, or that a *Preset* input can be substituted for or added to the *Clr* input if required. Furthermore, the way in which clock activation takes place may be modified by a reshaping of requirements in the ASM chart of Figure 9.9 and a consequent reformulation of the JK flip-flop design equations given at the beginning of section 9.2.6 of this text.

However, a much simpler version of the D flip-flop is obtained from a pseudo-static approach, as in Figure 9.12 for CMOS. Clearly, an nMOS version is also readily configured.

9.3 FORMING ARRAYS OF MEMORY CELLS

The memory cells discussed in section 9.2 and others will most often be used in arrays of some form or other. Typical arrays are registers and random access memories (RAM) and these arrays will be used as examples in this section. We must not forget, however, that another common application is to use memory elements individually as 'flags' or 'status bits' in system design. In any event, there must be some means of *selecting* a particular cell or group of cells and some means of effecting *read* or *write* operations.

9.3.1 Building up the Floor Plan for a 4 × 4-bit Register Array

This will be the third subsystem to be considered for the 4-bit data path, the floor plan of which appeared as Figure 7.5; the first two subsystem minimum bounding box outline dimensions have been given as Figure 7.10 (4 × 4 barrel shifter) and Figure 8.11(b) (4-bit adder). The fourth and final subsystem—the I/O port facilities—will be left for the reader to consider as an exercise in completing a system design (prior to adding inlet and outlet pads through which a chip is bonded to the outside world).

Starting with a bounding box representation of the chosen memory cell—in this case we have presented typical dimensions and connections of a pseudo-static cell with two bus lines, as in Figure 9.13—we can arrive at a bounding box for a single 4-bit register and hence the floor plan for a 4 × 4-bit register array.

The bounding box representation of the cell is 'stacked' to form a 4-bit register as in Figure 9.14, the overall vertical dimension being about 180λ. Note how the cells stack 'vertically' to form a 4-bit word and note that although a 'vertical' distribution of power has been assumed at the input of the register, power distributes horizontally thereafter. Note that since only a single metal layer has been assumed throughout, short but wide diffusion 'duck unders' have then been used to allow the ground (or V_{SS}) rail to cross the V_{DD} rail. However, it must be stressed that V_{DD} and *GND* (V_{SS}) connections must *always* be made through metal rails, except where crossovers are unavoidable; such crossovers should then normally be by means of short diffusion 'duck unders' where there is no second metal layer.

The required architecture may then be built up by stacking complete registers, side by side in this case, to form the desired four-register array, the dimensions being around 180λ × 240λ. The floor plan is given in Figure 9.15 and the diagram clearly indicates the direction of data flow and control signal distribution. Note that this floor plan does *not* include the selection and control circuitry.

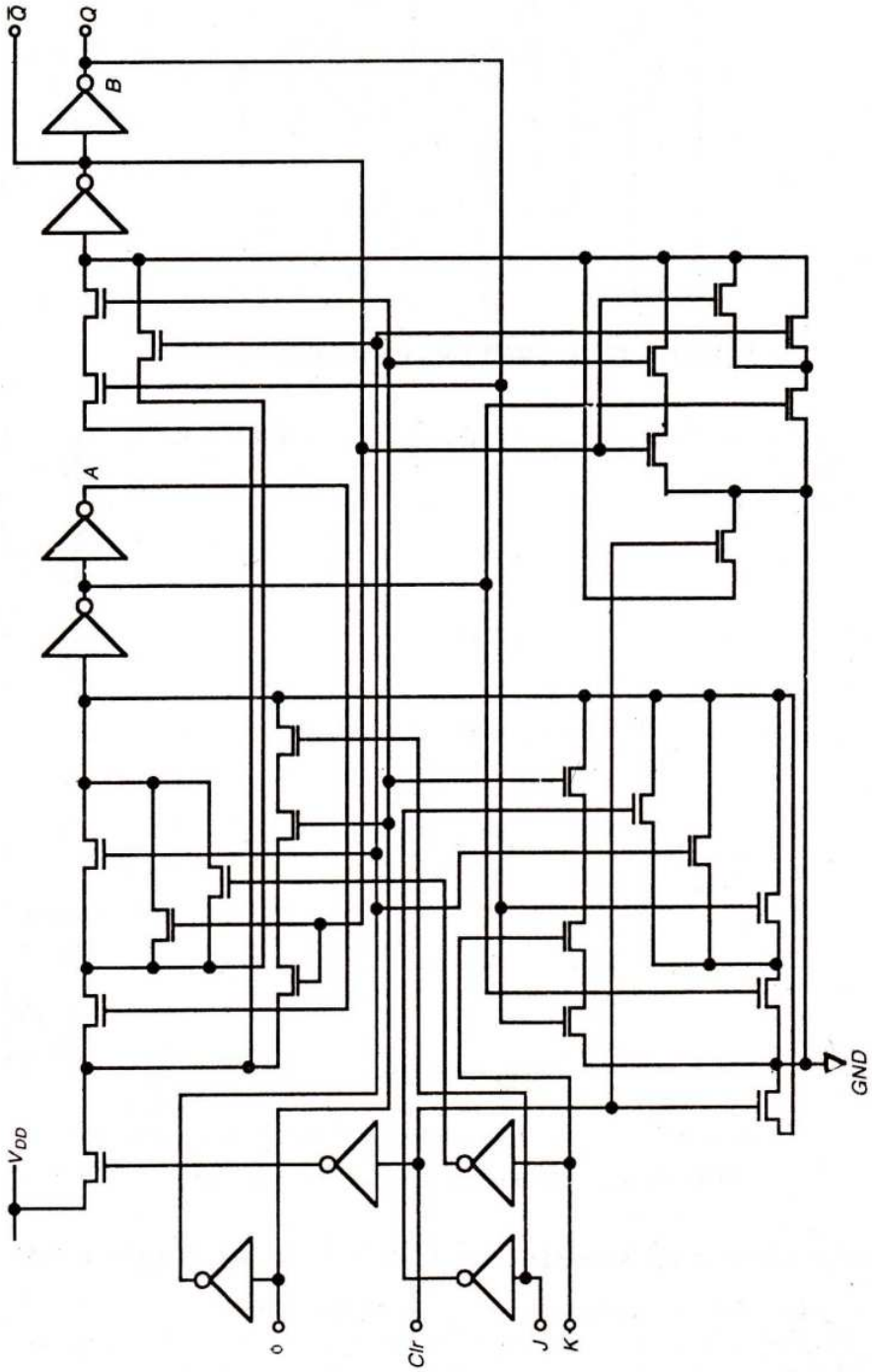


FIGURE 9.11 Switch logic implementation of JK flip-flop.

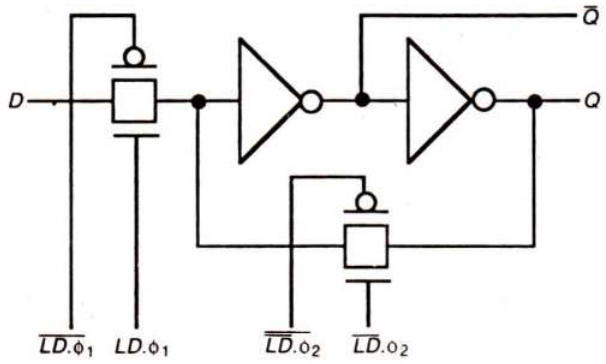


FIGURE 9.12 A CMOS pseudo-static D flip-flop.

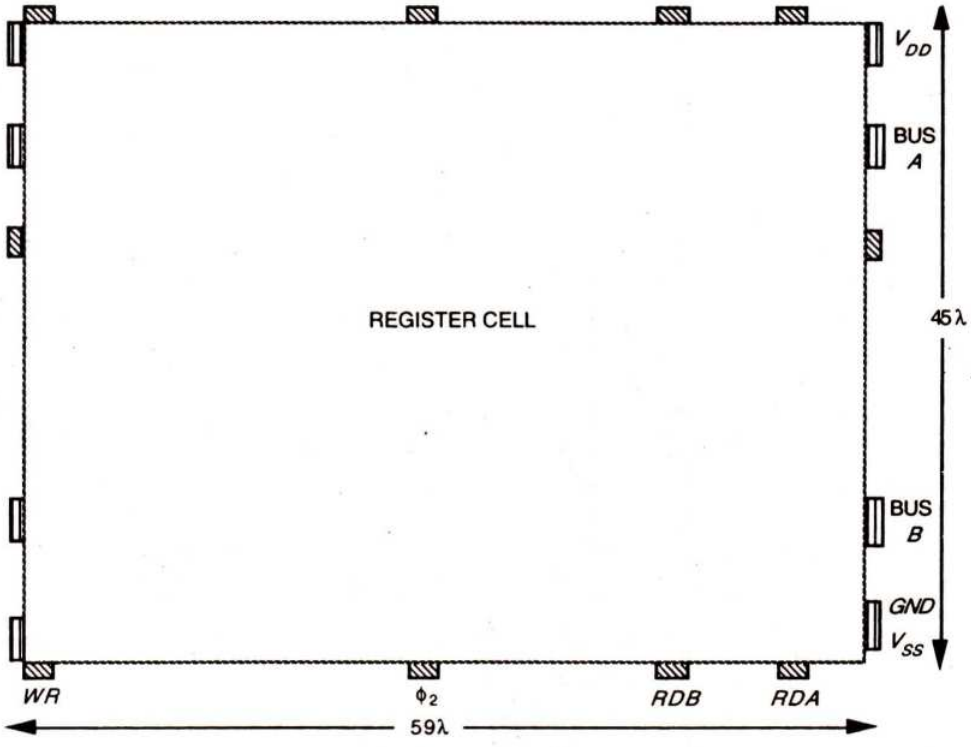


FIGURE 9.13 Bounding box for register cell.

9.3.2 Selection and Control of the 4 × 4-bit Register Array

Figure 9.15 shows that the register array must be provided with the control signals *WR0r–WR3r; RDB0r–RDB3r*; and *RDA0r–RDA3r*; derived from register select signals so that each register may be selected for read or write. We must also note *that we* need the capability to select two registers simultaneously for connection to the adder and that, in some cases, we

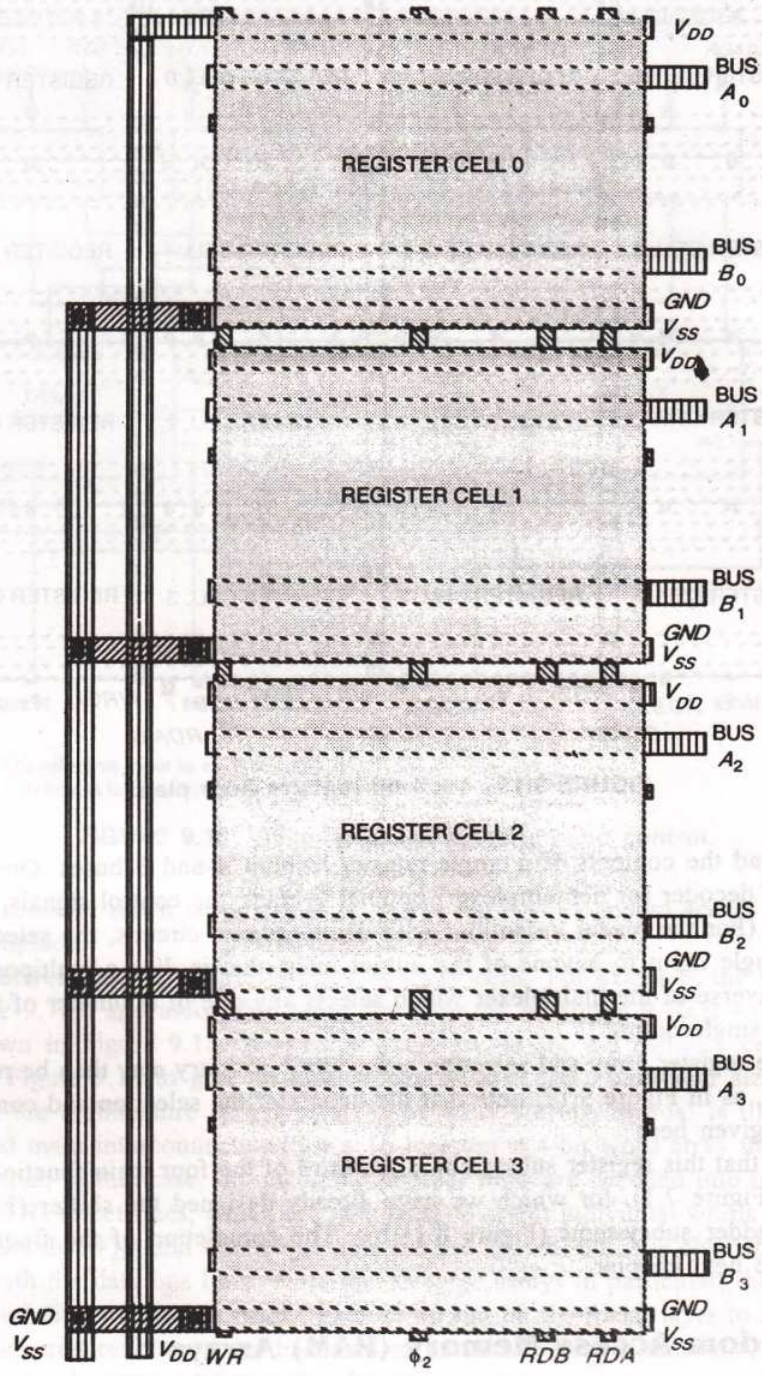


FIGURE 9.14 4-bit register floor plan.

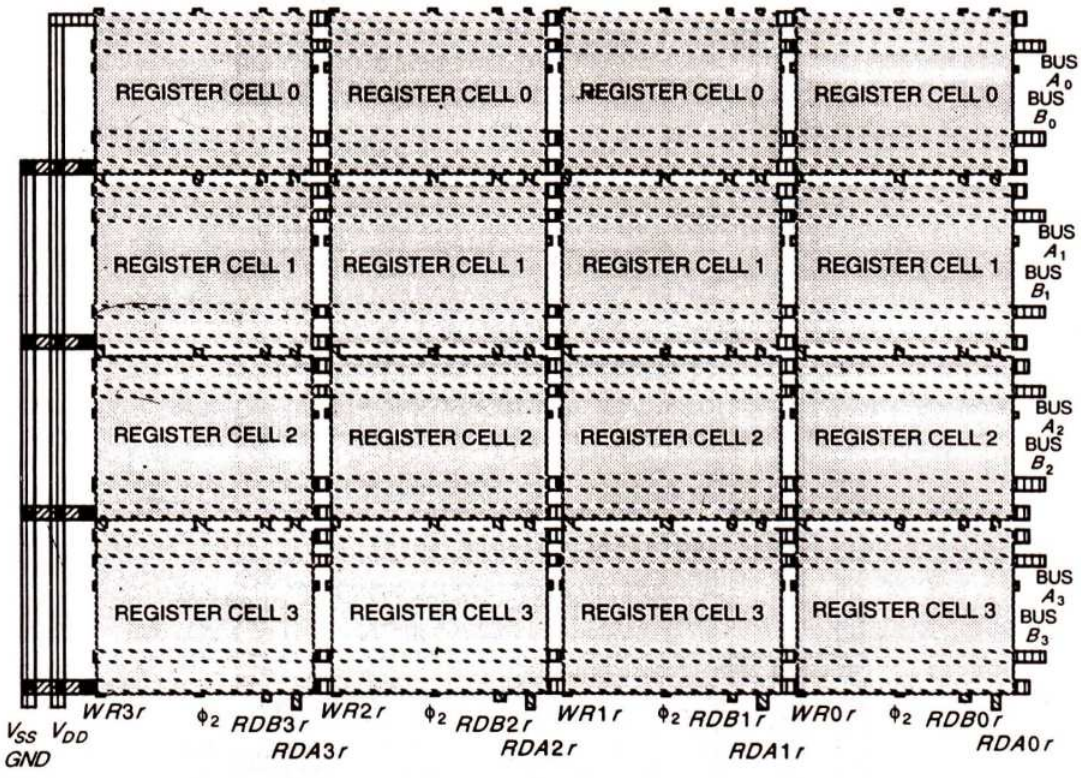


FIGURE 9.15 4 x 4-bit register floor plan.

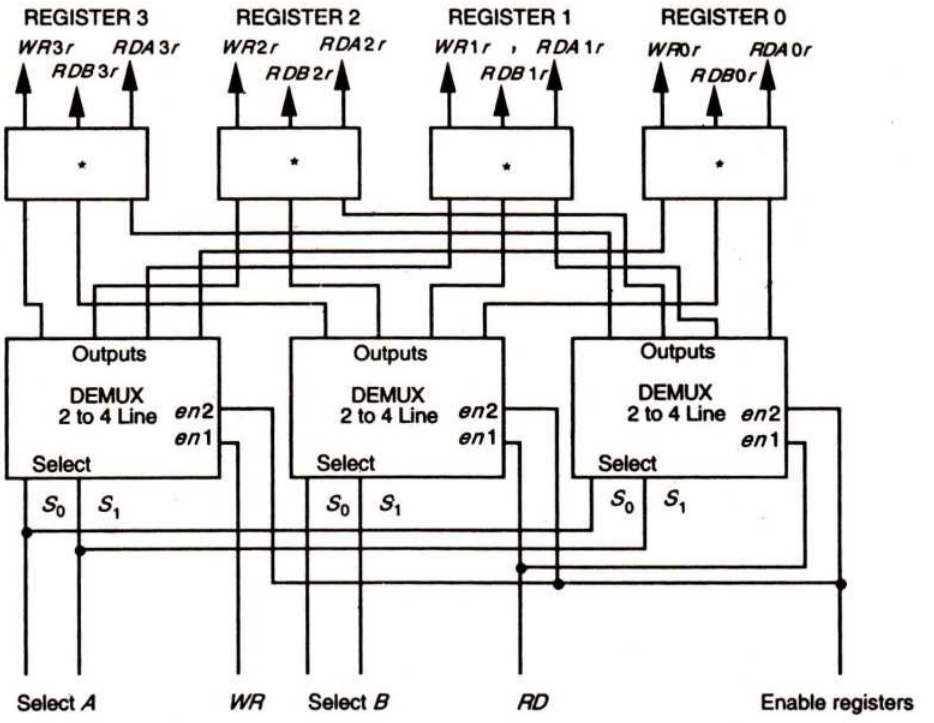
may wish to read the contents of a single register to both A and B buses. One approach is to make use of decoder (or demultiplexer) circuits to route the control signals, as suggested in Figure 9.16. (For the reader unfamiliar with demultiplexer circuits, the select lines allow routing of a single input to anyone of the output lines, that is, like a multiposition switch, and are the converse of the multiplexer which selects any one of a number of input lines to be routed to a single output.)

The whole register array and selection and control circuitry may then be represented in floor plan form, as in Figure 9.17; note that the details of the selection and control circuitry have not been given here.

Note also that this register subsystem is the third of the four main functional blocks of the data path (Figure 7.1), for which we have already designed the shifter (Figures 7.8 to 7.10) and the adder subsystems (Figure 8.11(b)). The completion of the floor planning is discussed in the next chapter.

9.3.3 Random Access Memory (RAM) Arrays

Now that we have considered individual memory cells, some of which are quite small in area and low in dissipation, and also the use of memory cells in an array of registers, it is not a



Note: The effective input to each decoder is en1.en2.
* Indicates buffers/drivers as required.

FIGURE 9.16 Decoder-based selection and control.

large step to consider much larger arrays of which the RAM (random access memory) is the most commonplace.

It is relatively easy to form arrays of memory cells. For example, the CMOS memory cell of Figure 9.7(b) and associated sense circuitry (as in Figure 9.7(c)) will form a RAM array as shown in Figure 9.18. A suitable mask layout for the memory cell used here is suggested in Figure 9.19 to give an idea of overall area for a particular size of array.

Finally, the architecture of a typical RAM array storing 'words' is illustrated by the floor plan and main interconnections for a 16-location × 4-bit word array in Figure 9.20. It will be seen that, in this case, the incoming address lines are decoded into row and column (with *RD* or *WR*) select lines, which are then used to select individual words in the memory. It will be noted that V_{DD} and $GND(V_{SS})$ rails are not shown, but it is clear that they may be interleaved with the data bus lines. Note that in large arrays in particular, the data bus lines will become relatively long and must therefore be run on the metal layer to avoid excessive capacitance or series resistance. As discussed earlier, the bus capacitance *and* the control line capacitances *must* be allowed for in the design.

To complete this section, Figure 9.21 is a plot of the *metal layer* only for a 16-location × 4-bit memory. The regularity of the memory array, the way in which the buses are run, and the various subsections of the floor plan are clearly evident.

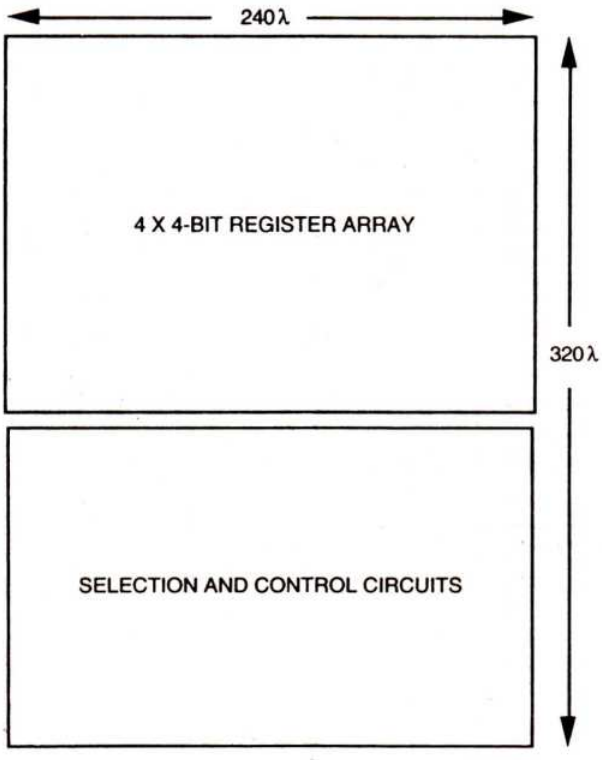


FIGURE 9.17 Overall floor-plan—register array and select/control circuits.

9.4 OBSERVATIONS

This chapter has completed our introduction to most of the techniques and many of the commonly used circuit arrangements for VLSI design in Silicon.

We have now completed the design of three of the four subsystems comprising the 4-bit data path which has been the 'vehicle' we have used to explore design processes.

We have also begun to see that communications are a very important aspect of design and this will be further emphasized in the next chapter.

9.5 TUTORIAL EXERCISES

1. Design a two-line to four-line decoder (demultiplexer) circuit to the mask layout level and determine its bounding box. Then work out the arrangement of, and area occupied by, the 4 × 4-bit register select and control circuit of Figure 9.16.
2. Taking a 16-location × 2-bit RAM arrangement as an example, suggest (with sketches only) how such an arrangement could be configured for two-port operation (i.e. data can be read or written to any location from either of two 2-bit data buses).

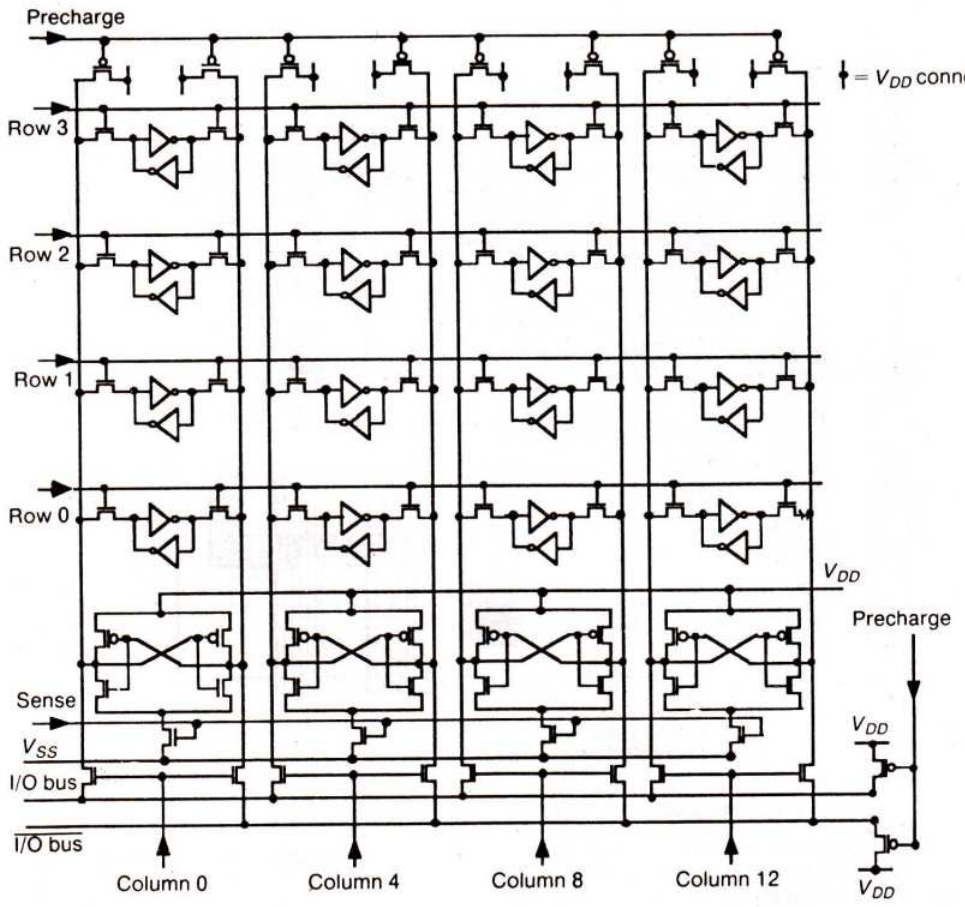


FIGURE 9.18 16-bit CMOS static memory array.

3. For the 4 × 4-bit register designed in this chapter, and using the select and control circuit suggested in question 1, work out all communication paths and interconnections between the three subsystems of the 4-bit data path we have so far designed. Use a bounding box representation for each subsystem and clearly indicate the layers on which interconnections are made. What is a suitable overall area for the processor as so far designed?
4. *nMOS:* Using a 4-bit word arrangement of Figure 9.22, and consulting the RAM arrangement of Figure 9.20, draw a mask level layout for the word select circuit associated with each 4-bit word and thus determine the overall area needed for each word stored. Next, design stick diagram level arrangements for the remaining blocks on the floor plan—that is, the row and column selection circuits and the input buffers and drivers, etc. Then, *estimate* the area needed for the 16 × 4-bit RAM as a whole (without I/O pads).

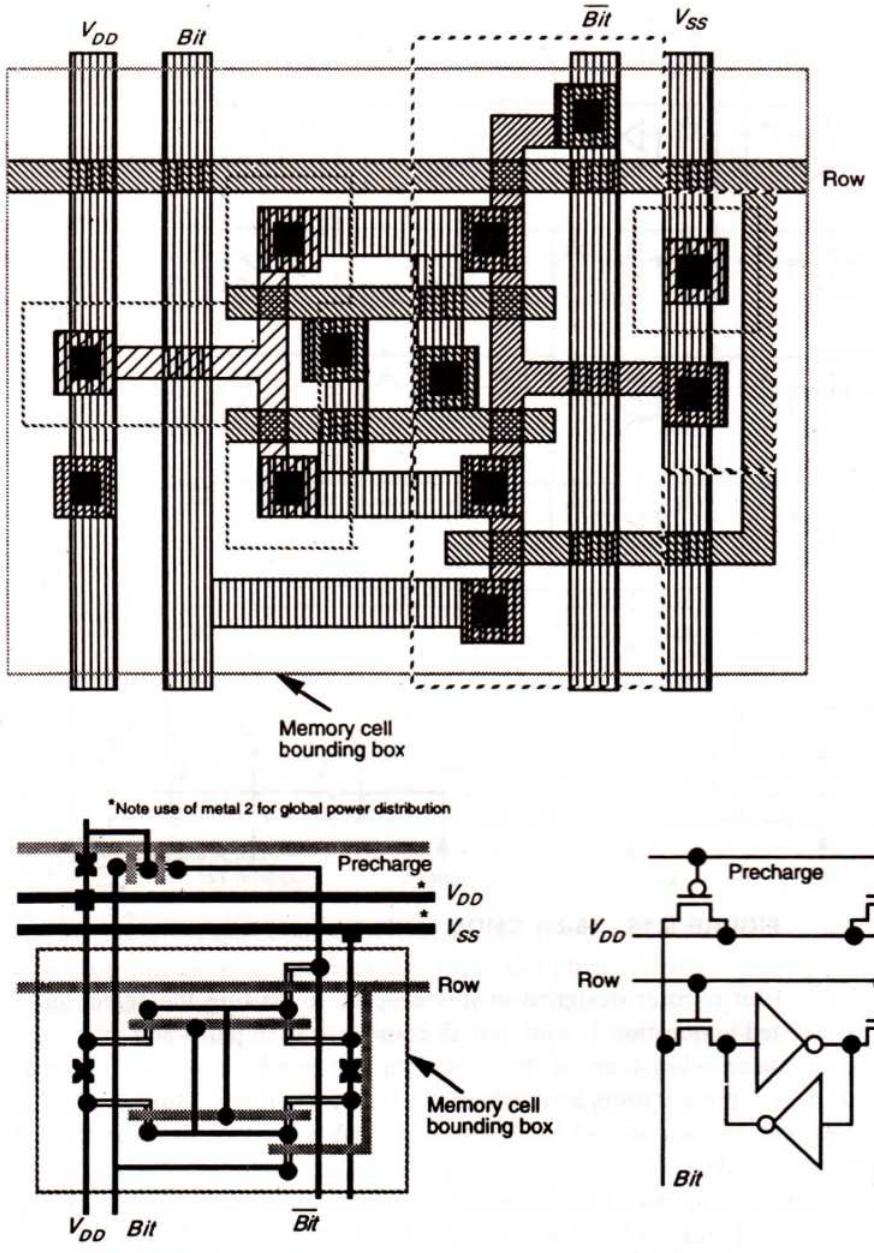


FIGURE 9.19 CMOS static memory cell-mask and stick layout.

5. *CMOS:* Starting with the 16-bit RAM array of Figure 9.18, design suitable decoding and control circuitry to allow row and column selection and read and write operation of the array. (You may find it useful to refer to section 9.2.5.)

   Design *one* memory cell as far as the mask layout and determine a suitable area per bit stored.

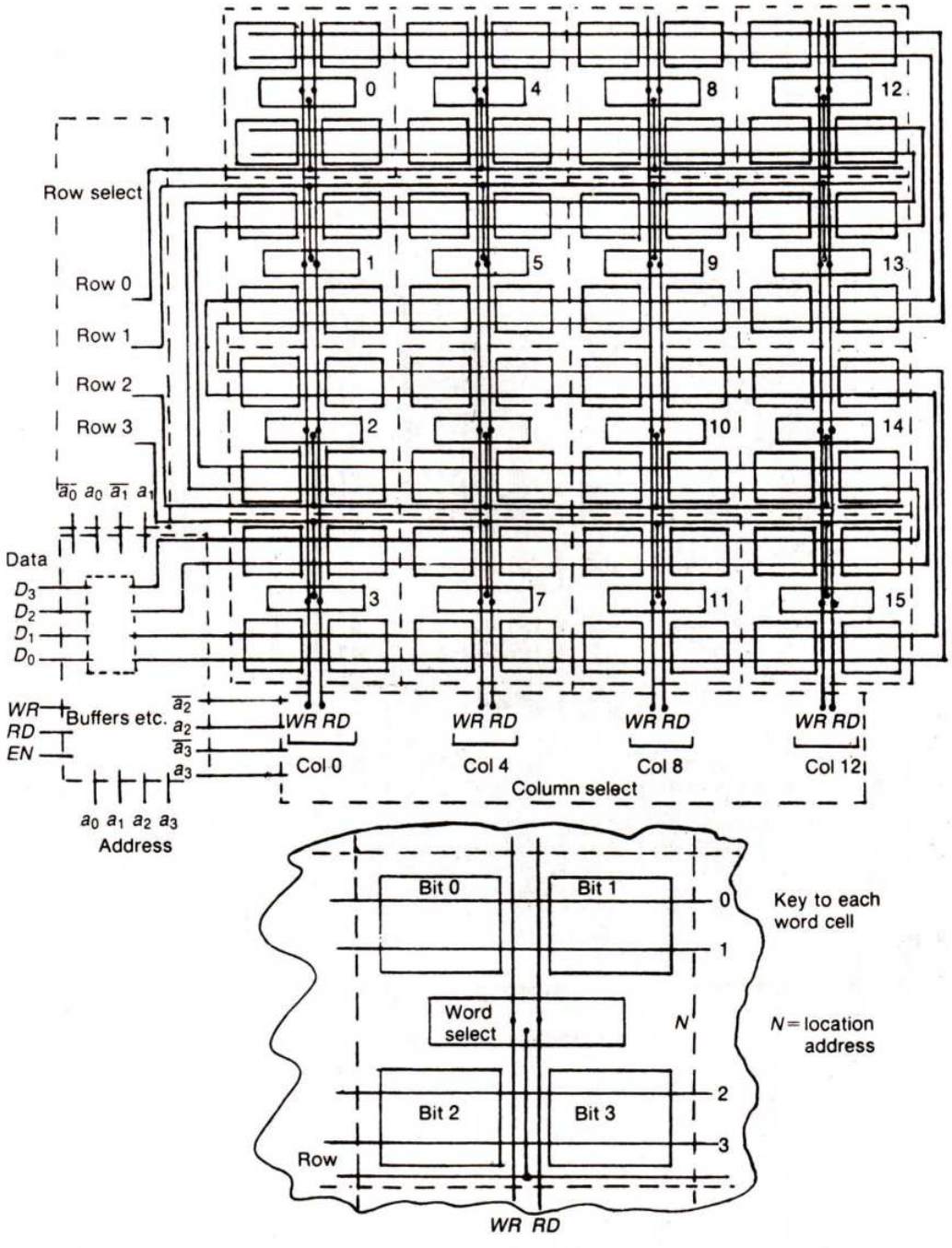


FIGURE 9.20 Floor plan of 16 × 4-bit RAM.

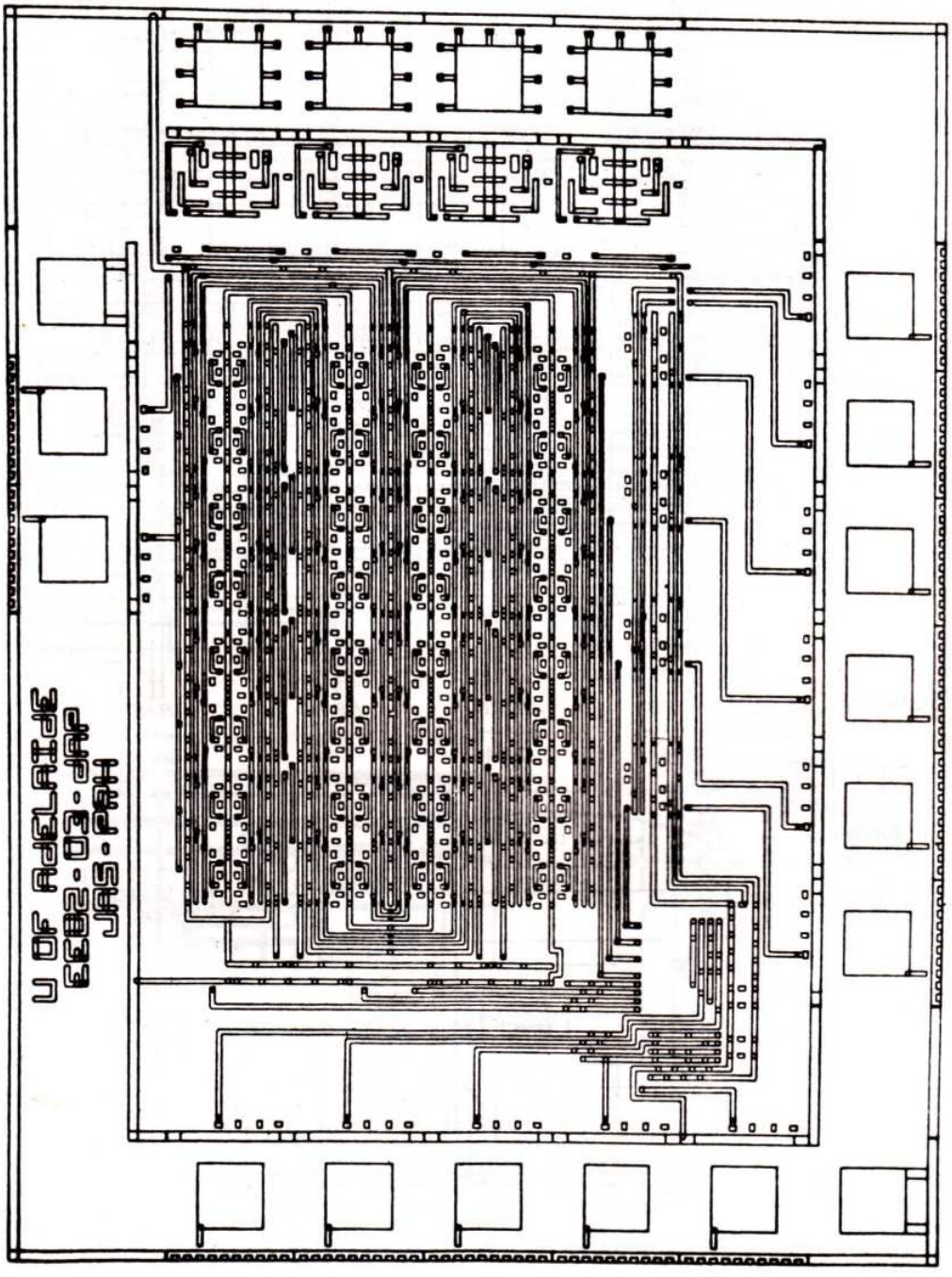


FIGURE 9.21 Metal layer only for 16 × 4-bit RAM.

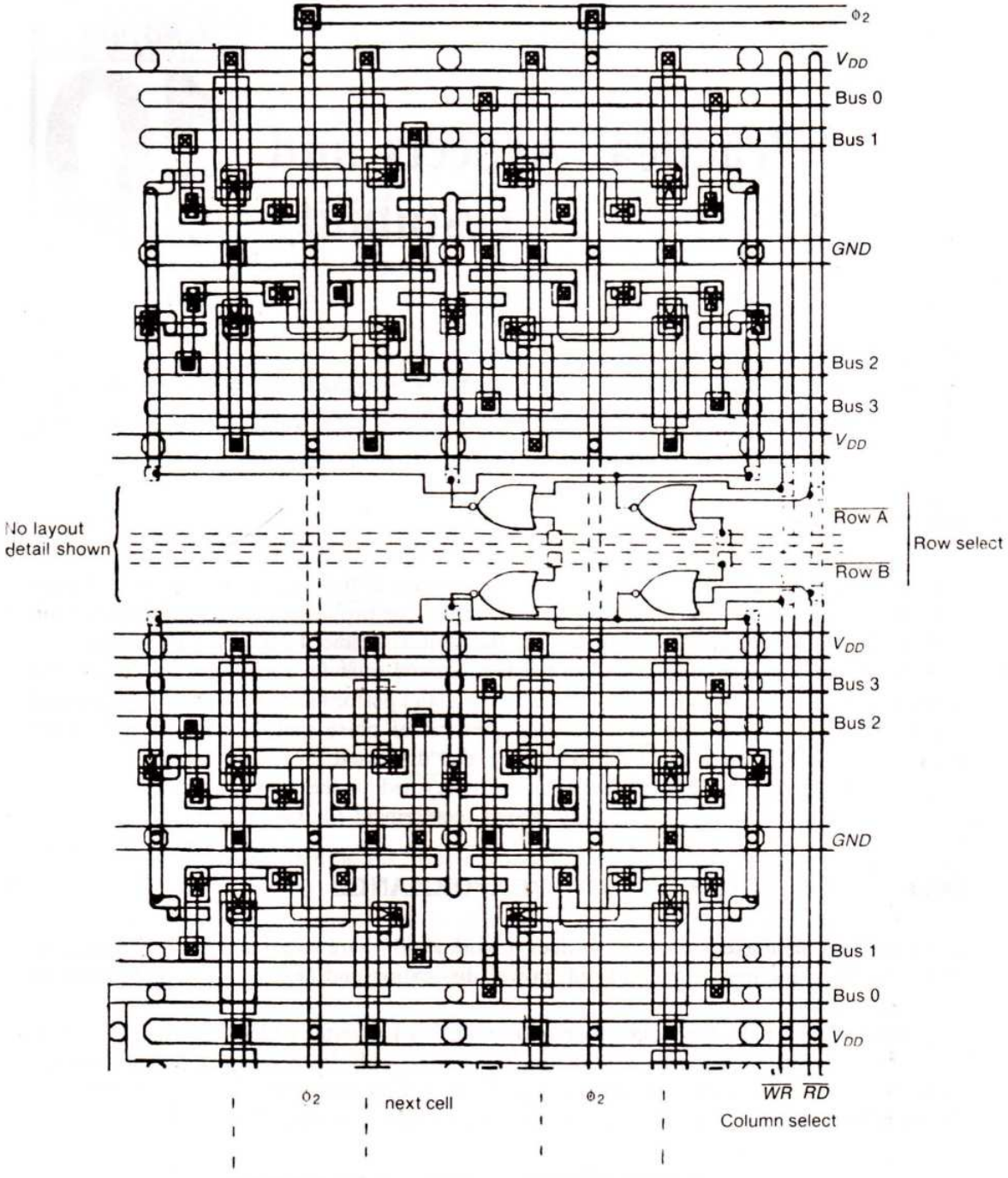


FIGURE 9.22 Two 4-bit words of nMOS RAM array.