3 *Fabrication and Design Rules*

Following the circuit design, the next level down in the structured design hierarchy is that of geometric layout. This is the specification of the geometric patterns required for each level of the fabrication process, so that when the shapes are superimposed they perform the desired circuit function.

The minimum size of a shape allowed on a layer and its relation to shapes on other levels are given by a set of design rules for the process. These rules ensure that the layout is within the capabilities of the fabrication process and that the designer does not inadvertently introduce additional unwanted features into the circuit such as parasitic transistors, short circuits or capacitances!

Since the basis for the design rules and the function of the shapes at each level of the process require an appreciation of the fabrication process, the principal stages in MOS processing will be considered first.

3.1 The MOS Process

Integrated circuits are manufactured on a wafer of silicon having many chip positions; for example, a circular wafer of 100 mm diameter can accommodate of the order of 150 6 mm x 6 mm chips. The fabrication of an MOS wafer usually requires between six and eleven patterning levels. These patterns are usually generated from a computer data file containing a low level description of the user's desired geometric layout. This description is combined with other users' descriptions if more than one chip design is to be included on the wafer.

Normally each layer pattern leads to the production of a mask by photographic (or electron beam) techniques. There are two types of mask. The first type allows light through the defined shapes but blocks light from passing through other areas, while the second type only allows light through areas external to the shapes; both mask types are likely to be used during fabrication to create a patterned level corresponding to the geometric shape or its reverse.

At each patterning level in the process, a set of similar operations has to be performed on the wafer and these are summarised in figure 3.1. First of all, new material is formed on or in the surface of the wafer (figure 3.1a). Here, a controlled amount of material is introduced as a surface coating. This is usually by means of evaporation, or by means of thermal growth where suitable gases flow over the wafer at high temperature, or by means of ion implantation where

charged impurity atoms are injected by firing them at the wafer surface. A further stage may be required where the wafer is heated to 'drive-in' a dopant to the required depth.

The next operation is to define the desired layer pattern in the grown material. The entire wafer is covered with photoresist which is a light-sensitive liquid film. The photoresist is exposed to ultra-violet light through the appropriate mask (figure 3.1b). Either positive or negative resist can be used, and figure 3.1c shows negative resist where areas exposed to the light are hardened; positive resist has the opposite effect.

Unhardened photoresist is removed by a solvent, exposing the underlying material in these regions. This material is now etched away, as shown in figure 3.1d; the hardened photoresist protects its underlying material from the etch. Finally, the hardened photoresist is removed by etching to leave the desired pattern (figure 3.1e).

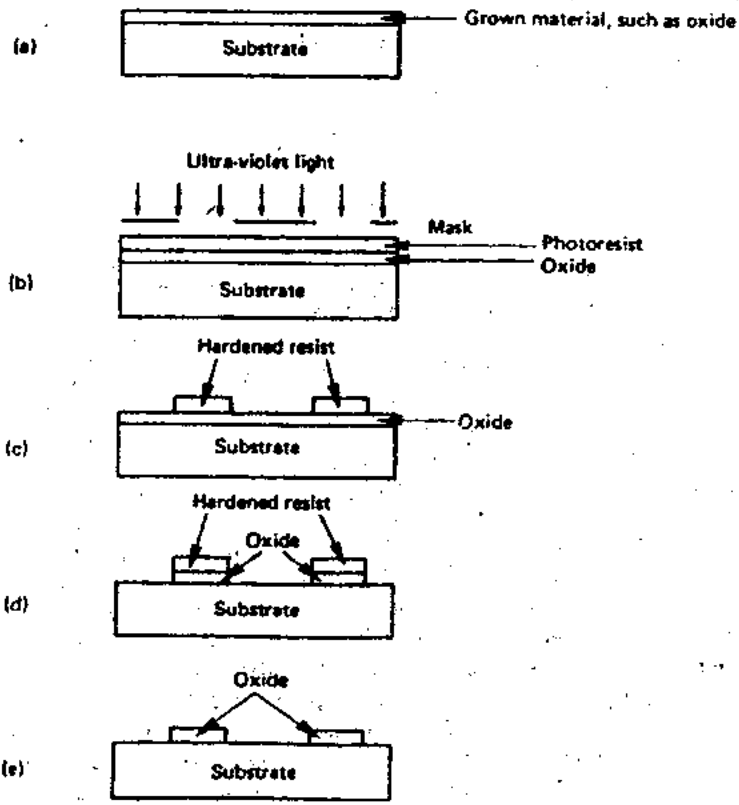


Figure 3.1 MOS processing: (a) material growth, (b) layer patterning, (c) removal of unexposed resist, (d) oxide etch, (e) final layer pattern

The mask-making stage can be dispensed with by directly 'writing' the desired layer pattern on to the wafer. However, at the present time, the cost of the equipment needed to produce fine geometrical lines using this technique is too expensive for it to be in widespread use.

3.2 NMOS Processing

The fabrication process is most easily understood by considering the implementation of a simple circuit, and figure 3.2 shows the geometric layout at each level of an NMOS inverter with a depletion load; the layout obeys the process's design rules.

A transistor gate is formed wherever polysilicon crosses diffusion (semiconductor) with oxide between these layers. This leads to transistors at these points since diffusion regions surrounding the gate areas are doped with an n^+ impurity and thus form the transistor drains and sources. A depletion implant surrounds the NMOS depletion transistor and this patterning level is used to alter the device's threshold so that a depletion transistor rather than an enhancement device is formed.

It will be seen from figure 3.2 that there are three points at which polysilicon crosses diffusion and that the middle point is surrounded by a buried contact. This buried contact removes the oxide between the polysilicon and diffusion so that these two conducting materials contact one another; it should be noted that no transistor is formed here because the oxide has been removed. This polysilicon-to-diffusion connection effects the gate-to-source connection of the depletion transistor.

The 5 V and 0 V power lines are implemented in metal because of its very low resistance. The 5 V metal line connects to the drain of the depletion transistor and the 0 V to the source of the enhancement device via contact cuts. These contact cuts are holes down to the diffusion level so that metal can flow into the hole, thereby allowing metal and diffusion to contact.

Figure 3.3 outlines the principles of NMOS processing as a series of cross-sections along the arrowed centre line shown in figure 3.2. It will be appreciated that, in practice, the process is more complex and that the exact sequence of operations depends upon the particular process.

The starting material of the wafer is a lightly doped p-type silicon substrate (figure 3.3a). Mask 1 defines all diffusion regions, called active areas; these include all transistor areas (source, gate and drain) plus any diffusion lines used to interconnect circuits. Areas external to the active regions are covered with a thick isolating oxide (figure 3.3b).

Mask 2 defines the depletion implant regions. The areas defined by this mask are given an n-type implant (figure 3.3c). This is normally followed by an unmasked p-type implant and drive-in which sets the depletion and enhancement thresholds throughout the active regions. The field oxide prevents penetration in the isolation areas.

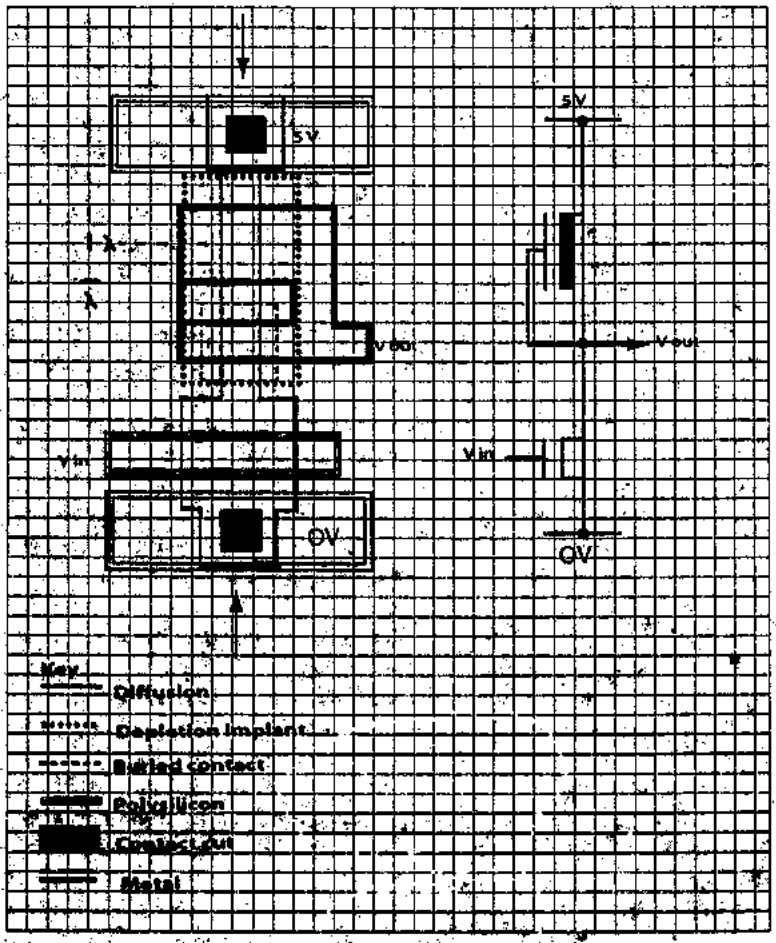


Figure 3.2 Geometric layout of an NMOS inverter with a deplication load

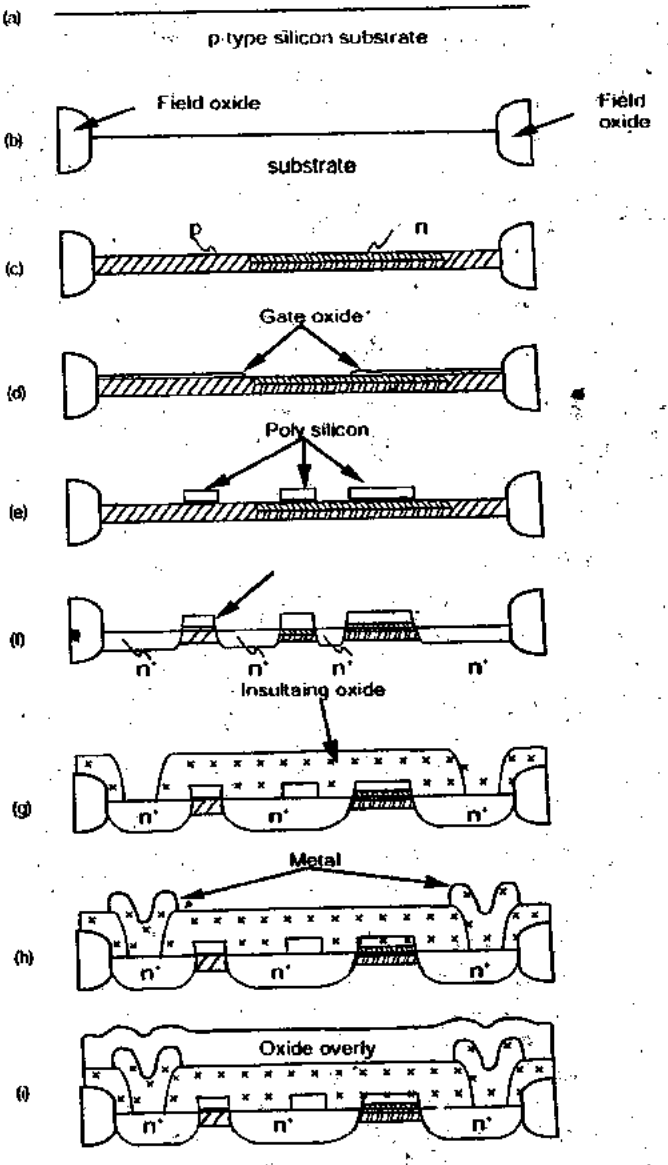


Figure 3.3 NMOS processing: (a) starting material, (b) active region definition - mask 1, (c) thershold implants -mask 2, (d) buried contact area - mask 3, (e) polysilicon definition - mask 4, (f) source and drain diffusion, (g) contact cuts -mask 5, (h) metal definition -mask 6, (i) final cross-section of NMOS inverter

The entire wafer surface is covered with a thin layer of (gate) oxide and mask 3, the buried contact mask, defines regions where the oxide is to be removed (figure 3.3d). The surface is now covered with polysilicon and mask 4 specifies the areas where polysilicon is to remain. This includes all gate areas, all polysilicon to diffusion connections and all polysilicon interconnections (figure 3.3e).

An unmasked n^+ diffusion now defines the source and drain regions (figure 3.3f). Note that, since the edges of the gate define the start of the transistor's source and drain, these features are self-aligned relative to the position of the gate. The wafer surface is covered with an oxide which will insulate the polysilicon and diffusion from the metal layer. The wafer is heated to provide a smooth surface and to drive-in the n^+ regions.

Mask 5 defines the contact cuts in this insulating oxide where metal is to be connected to diffusion or polysilicon (figure 3.3g). The wafer is covered with aluminium and mask 6 specifies regions where the aluminium is to remain; this includes all metal interconnections and all metal to diffusion and polysilicon connections (figure 3.3h).

An oxide overlay is grown to protect the surface (figure 3.3i). Mask 7 defines the areas where the overlay is etched away to allow contact between the aluminium of the input and output pads of the chip and external circuitry.

3.3 The CMOS Process

Here there are two approaches. Either the starting material of the substrate is n-type, in which case a p-type well is made for the fabrication of the NMOS device, or the starting material is p-type and an n-well is created in which a PMOS device is made. In the past, the former method was chosen as it was easier to form a p-well than an n-well (as the n-type substrate needed to be less heavily doped than the p-type substrate). However, the emphasis is now on an n-well process as this allows a combination of NMOS and CMOS devices to be more efficiently fabricated on the same wafer or within the same chip.

Figure 3.4 shows the cross-sectional structure of an n-well CMOS inverter. The well has to be connected to the most positive voltage available so that the pn junctions of the PMOS device are always reverse-biased. Figure 3.4 shows the 5 V line connecting to both the p^+ diffusion, forming the source of the PMOS transistor, and also to the n-well via an n^+ diffusion. Similarly, the p-type substrate (via a p^+ diffusion) and the source of the NMOS device are connected to the most negative available voltage – that is, 0 V; a local connection to 0 V for the substrate is advisable to reduce the possibility of the parasitic transistors present within the CMOS structure turning on.

A geometric layout for a CMOS inverter is shown in figure 3.5. Although the depletion implant and buried contact masks required for the NMOS process are not required, CMOS fabrication requires the definition of the n-well areas in

addition to differentiation between active regions to be doped p^+ and n^+; this makes CMOS fabrication more complex than NMOS. The power and 0 V connections to the CMOS inverter are normally effected by butting contacts as shown in the layout and it can be seen that here the metal connects a region of n^+ diffusion to an adjacent p^+ diffusion area. This method of connecting the rails is preferred as it minimises the silicon area occupied.

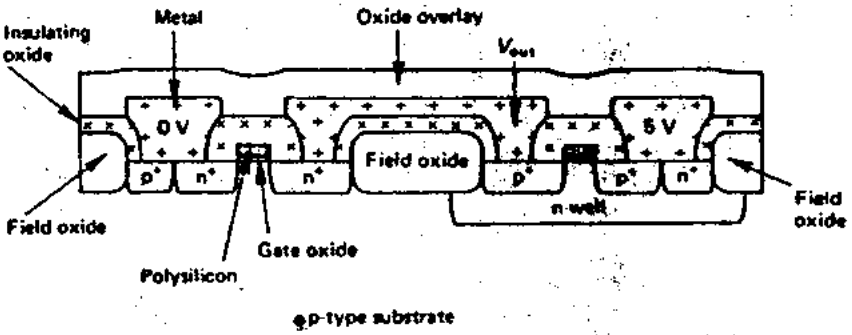


Figure 3.4 Final cross-section of a CMOS inverter

The starting material for the CMOS process is a p-type substrate. The area defined by mask 1 is converted to n-type and forms the n-well for the PMOS devices. Mask 2 defines all the diffusion regions for the PMOS and NMOS devices and any diffusion interconnections. Regions external to these active areas are covered with an isolating field oxide. Mask 3 surrounds the n-well area and the regions external to it are subjected to a threshold implant.

Surface oxidisation with a thin layer of oxide is followed by coating the wafer surface with polysilicon. Mask 4 defines where the polysilicon and underlying gate oxide are to remain. A p^+ diffusion using mask 5 defines the source and drain regions of the PMOS devices and the substrate connection area. An n^+ diffusion using the reverse of mask 5 now defines the source and drain regions of NMOS devices and the n-well connection area. Note that again a device's source and drain areas are self-aligned with respect to the gate.

The remaining processing is very similar to that for NMOS. Isolation oxide is grown over the wafer and mask 6 defines the position of contact cuts. The surface is covered with aluminium and mask 7 defines where metal is to remain. Finally, the surface is covered with an overlay oxide and mask 8 defines the contact cuts in the overlay for the input and output pad positions. The final cross-section of the inverter, corresponding to the arrowed line on the layout, is shown in figure 3.4.

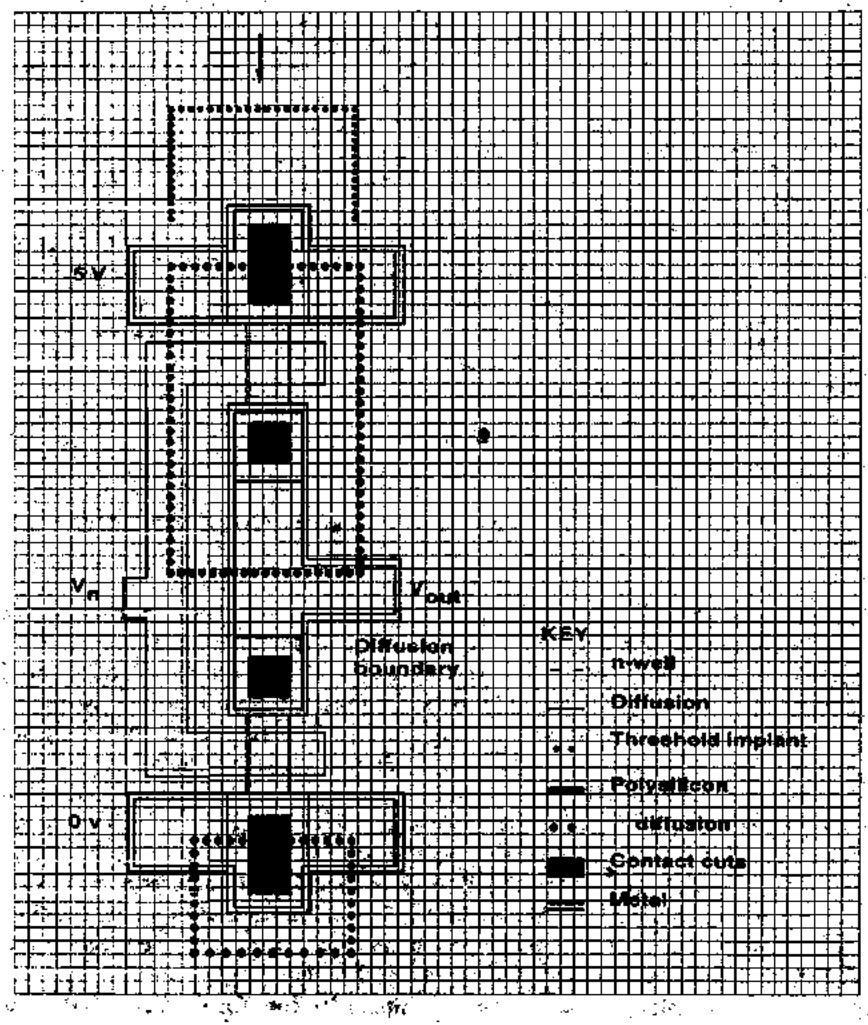


Figure 3.5 Geometric layout of a CMOS inverter

3.4 Yield

The successful fabrication of a chip to the user's specification depends upon the accurate control of the many operations within the process. It also depends

upon factors such as imperfections in gases and materials used, and mis-alignment of masks causing mis-registration between the patterning layers. Processing faults normally cause unwanted short circuits between layers or cause open circuits owing to breaks in the conducting layers or poor contact between them; such faults make the design function in a different manner to that expected.

Process control is monitored by parametric test circuits on the wafer. These are normally implemented by placing test chips, called drop-ins, at random chip positions on the wafer or by allocating some area on each chip design for a test circuit; the former method is preferable as the user does not lose valuable silicon area. After processing, these test areas are exercised and their electrical parameters observed. For example, device threshold voltages, $\epsilon\mu_n/D$, resistances, capacitances and test circuit speeds might be monitored.

Wafers which have a reasonable percentage of test circuits satisfying the specified processing tolerances for the fabrication line are suitable for functional testing. The wafer is either cut (scribed) and a selection of chips from random points on the wafer are packaged for the user to evaluate, or the wafer is tested prior to scribing so that only working chips are packaged and given to the user. Clearly, testing prior to packaging is preferable although it involves the use of specialised test equipment.

It is hardly surprising, in view of the complexity of fabrication, that the percentage of functionally working circuits on a wafer (called the 'yield') is not high and that a yield of 30 per cent is considered to be good. It should also be noted that, in general, the greater the silicon area occupied by a design, the smaller the resulting yield, and for large designs this factor should be taken into consideration at the system design stage when error detection, correction or fail-safe strategies are determined.

3.5 Electrical Parameters

Each process has parameter values associated with the particular line. These include not only figures for the transistor parameters but also values for the resistance and capacitance of the conducting layers. The parameter values are directly related to the system performance and circuit design. Table 3.1 shows typical values for a process with a 6 μm minimum line width.

The transistor parameters allow the user to calculate transistor aspect ratios and several examples of their use have been given in chapter 2. The capacitance values combined with the geometric layout enable the approximate magnitude of the capacitance at any point in a circuit to be estimated; these can subsequently be used in circuit simulation to determine the speed of operation.

For example, consider the layout of the NMOS inverter in figure 3.2. The input capacitance C_{in} of the inverter is the gate capacitance C_{gate} of the enhancement transistor plus the capacitance C_{poly} of the other polysilicon driven by the input. If figure 3.2 is drawn to a scale of 3 μm per division (that is, 9 μm^2 per

Table 3.1 Typical process parameters for a 6 μm line

Transistor parameters	
NMOS enhancement device	$V_{te} = 1$ V at $V_{sb} = 0$ V
	$\epsilon\mu_n/D = 30\ \mu A/V^2$
NMOS depletion device	$V_{td} = -4$ V at $V_{sb} = 0$ V
	$\epsilon\mu_n/D = 25\ \mu A/V^2$
PMOS enhancement transistor	$V_{tep} = 1$ V at $V_{sb} = 0$ V
	$\epsilon\mu_p/D = 15\ \mu A/V^2$
Capacitances	
Diffusion	0.00013 pF/μm²
Gate	0.0004 pF/μm²
Polysilicon	0.00005 pF/μm²
Metal	0.00003 pF/μm²
Resistances	
Diffusion	5-15 Ω/square
Polysilicon	20-80 Ω/square
Metal	0.03 Ω/square

square) corresponding to a minimum line width of 6 μm, then the gate area attached to V_{in} is 108 μm² (6 μm x 18 μm) and the remaining polysilicon also occupies 108 μm². So

$$C_{gate} = 108 \times 0.0004 \text{ pF} = 0.043 \text{ pF}$$

and

$$C_{poly} = 108 \times 0.00005 \text{ pF} = 0.005 \text{ pF}$$

giving an input capacitance, C_{in}, of the order of 0.05 pF

To calculate the output capacitance of the inverter, it is necessary to evaluate the capacitance of all features electrically connected to V_{out}. Thus C_{out} comprises the gate capacitance of the depletion transistor, C_{gate}, plus the capacitance of the remaining polysilicon connected to V_{out} (including the buried contact area), C_{poly}, plus the capacitance of the diffusion between the driver and load transistors, C_{diff}. Using the layout to give the number of squares relevant to each term

$$C_{gate} = 8 \times 9 \times 0.0004 \text{ pF} = 0.029 \text{ pF}$$

$$C_{poly} = 48 \times 9 \times 0.00005 \text{ pF} = 0.022 \text{ pF}$$

C_{diff} = 24 x 9 x 0.00013 pF = 0.028 pF

giving a total C_{out} of 0.08 pF. It should be appreciated that this is only an approximation, as capacitance parameters have a wide tolerance (typically ± 15 per cent) and are voltage dependent. In addition, secondary effects such as the gate–drain capacitance of the pull-down transistor contribute to C_{out} so that a value of about 0.1 pF is more realistic.

The resistance of a conducting material of constant depth is proportional to its length/width and is therefore measured in ohms per square since the resistance is independent of the size of the square. The resistance values for the process when combined with the capacitance figures determine the effect of interconnecting signals between a circuit output and input.

For example, consider an interconnection of length L μm and width W μm having a resistance of R Ω/square and a capacitance of C pF/μm². The line resistance R_l is LR/W Ω and the line capacitance C_l is LWC pF. The line delay R_lC_l is therefore $L^2CR/1000$ ns.

In a 6 mm x 6 mm silicon chip, the maximum line length arises between two diagonally opposite corners of the chip. Assuming only vertical and horizontal connections are allowed, the maximum line length is 12 mm. Using the above formula for the line delay with the capacitance and average resistance values given for polysilicon, diffusion and metal, the delay down a 12 mm line is about 360 ns for a polysilicon interconnection, 187 ns for a diffusion line and only 0.13 ns for a metal line.

Thus the delay down diffusion and polysilicon lines can be very significant. Often, an output is connected to more than one gate input and if these inputs are spaced along the output line then there will be a time difference between the arrival of the output signal at the different inputs. The reader should be aware that this timing difference or skew can be considerable for polysilicon and diffusion lines.

It is clearly advantageous to have as many long interconnections as possible in metal and otherwise to use diffusion. However, most circuit inputs (and outputs in NMOS) are in polysilicon. This, combined with the problem of interconnecting many points and the availability of only a single metal layer, imposes a different routing scheme. As a result, it is normal practice to run metal lines in one direction, say vertically, and to place polysilicon or diffusion interconnections at right angles to this – that is, horizontally.

Where long lines in polysilicon or diffusion are unavoidable, the line delay can be significantly reduced by inserting gates in the signal path since the delay is proportional to the square of the line length. For example, a gate placed halfway down a 12 mm polysilicon line reduces the delay to 180 ns (90 ns for each 6 mm section).

It should be noted that two layers of metal would greatly ease the problem of delays down lines; unfortunately, this is not at present generally available for full custom design.

3.6 Scaling

Electrical parameters relating to a 6 μm process have been presented. These results can also be used to estimate the characteristics of finer geometry processes by scaling features. The effects of scaling are most easily considered by assuming that all geometric dimensions (horizontal and vertical) and voltages are reduced by a constant factor a. Thus

$$\text{new width } W' = \frac{\text{old width}}{\text{scale factor}} = \frac{W}{a}$$

$$\text{new length } L' = L/a$$

$$\text{new thickness } D' = D/a$$

$$\text{new supply voltage } V'_p = V_p/a$$

and

$$\text{new enhancement device threshold } V'_{te} = V_{te}/a$$

Applying the current equation (2.3) to a scaled transistor, its saturation current I' is

$$I' = \frac{\epsilon\mu_n W'}{2L'D'} (V'_{gs} - V'_t)^2 = \frac{I}{a}$$

Thus the current per transistor decreases by a factor a. However, since a factor of a^2 more scaled devices can be placed on a similar sized chip, the current drawn from the supply increases by a factor a. The scaled supply voltage is V'_p/a, so that the power supplied to a similar sized chip is unaltered by scaling.

Circuit capacitances are reduced by a factor a since

$$C' = \frac{\epsilon L'W'}{D'} = \frac{C}{a}$$

Relating this to the gate delay of the scaled circuit

$$\text{gate delay}' \propto \frac{C'_{out}}{W'/L'}$$

Hence

$$\text{gate delay}' = \frac{\text{gate delay}}{a}$$

and the gate speed is increased by a factor a. Now the gate power is

$$V_p' I' = \frac{V_p I}{a^2}$$

so the speed-power product is

$$\text{speed-power product}' = \text{gate delay}' \, V_p' \, I' = \frac{\text{speed-power product}}{a^3}$$

It can be seen that, apart from the increase in current density on the chip, the other effects of reducing features and voltages are all advantageous. However, another unwanted effect arises when considering the delay down lines interconnecting gates. Here the length does not scale as the chip is assumed to be of similar area. Hence line lengths will remain constant. (Indeed lengths are likely to increase as improvements in the technology allow increased chip sizes.)

Assuming an interconnection length L, the line capacitance remains the same since

$$C_i' = \frac{\epsilon L W'}{D'} = C_i$$

but the line resistance R_i' scales up by a factor a^2 as

$$R_i' \propto \frac{L}{W' T'}$$

where T' is the new conductor depth. This gives

$$R_i' = a^2 R_i$$

The delay down an interconnection line is proportional to $R_i' C_i'$ and thus scales up by a factor a^2. Taking a 12 mm line length again and a scaling factor of 10, the delay is 36.0 μs for a polysilicon interconnection, 18.7 μs for a diffusion line and 13 ns for a metal line. Thus delays in polysilicon and diffusion become unacceptably large and the delay down metal lines is no longer negligible. This suggests that it will not be sensible to scale all features by an identical factor.

3.7 Design Rules

Design rules for the geometric layout have allowed the design of silicon chips to be undertaken by those with little electronic engineering background, since their use has removed the necessity for a detailed understanding of fabrication. Design rules also have the advantages of shortening the design process by allowing the user to translate from a circuit diagram to a layout in a relatively short time (with practice!) and of enabling the layout to be checked for violations.

The design rules have to take into account the tolerances encountered during processing. Factors such as under or over etching cause features to expand or contract, while mis-registration of features on one layer with respect to features on another layer cause shapes to deviate from their specified position.

It is usual to specify the design rules in terms of the processing tolerance for the fabrication line, λ. λ represents the maximum shift from the theoretical position on the layout between features on two different layers. Thus the minimum length or width of a feature on any layer is 2λ to allow for shape contraction. Similarly, the separation of features on a layer is a minimum of 2λ to ensure adequate continuity of the intervening material. Currently values of λ are between 1 μm and 3 μm, depending on the age of the line, and it is usual to draw layouts such as those in figures 3.2 and 3.5 to a scale of 1λ per division.

Typically, the minimum width for a polysilicon and diffusion line is 2λ. Metal lines run over a more uneven surface than the other conducting layers and are therefore a minimum of 3λ wide to ensure their continuity. Polysilicon lines can be spaced 2λ apart, as can metal lines. Diffusion lines have to be spaced 3λ apart to avoid the possibility of their associated depletion regions overlapping and conducting current.

Where a diffusion line runs parallel to a polysilicon line, the lines are separated by λ to prevent the lines overlapping to form an unwanted capacitor. Metal lines can pass over both diffusion and polysilicon without electrical effect. However, because of the uneven surface for metal, it is the recommended practice to leave λ between a metal edge and a polysilicon or diffusion line to which it is not electrically connected (that is, the metal is unrelated to the polysilicon or diffusion). These rules relating to the minimum width and separation of conducting lines are illustrated in figure 3.6; again, the length of each division on the diagram is λ.

A transistor is formed where polysilicon crosses diffusion with thin oxide between these layers. It should be noted that the design rules for the minimum line width of polysilicon and diffusion define a transistor gate, source and drain to have a minimum length and width of 2λ. In addition, the polysilicon of the gate extends 2λ beyond the gate area on to the field oxide to prevent the drain and source from shorting. These rules can be observed in figures 3.5 and 3.6.

In NMOS, a depletion implant is used to form a depletion transistor. An implant surrounding the transistor by 2λ ensures that no part of the transistor remains in the enhancement mode, and similarly a separation of 2λ from the

get of an enhancement transistor aviods affecting this device. Implants are seprated by 2λ to prevent them form (*see figure 3.6*)

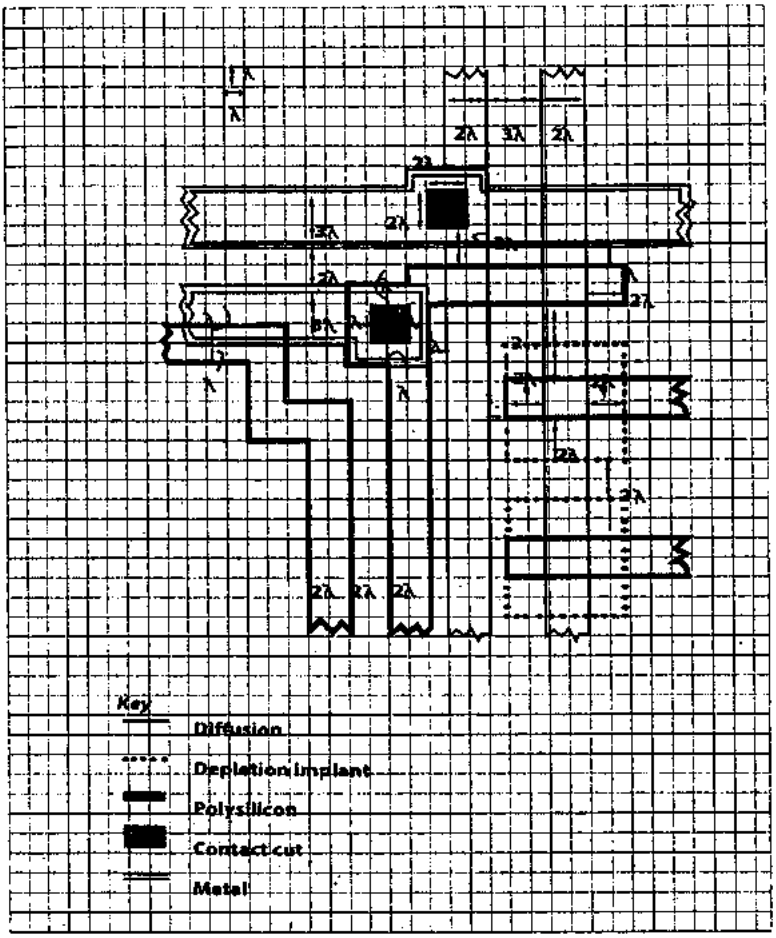


Figure 3.6 Examples of the NMOS design rules

The gate and source of a deplication device are contected together to from the load in NMOS circuts. This connection can be made by a connection known as a 'butting conrtact; where metal makes contact to both the diffusion forming

the source of the depletion transistor and to the polysilicon forming this device's gate. The advantage of the butting contact is that it removes the need for the buried contact mask and its associated processing. However, considering the cross-section of such a contact, shown in figure 3.7, it can be seen that the metal descending the hole has a tendency to fracture at the polysilicon corner, causing an open circuit.

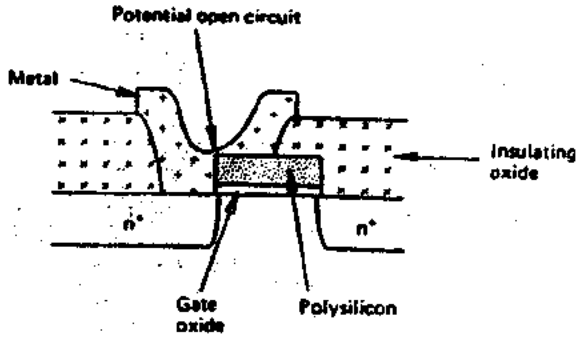


Figure 3.7 NMOS butting contact

For this reason, the buried contact is the preferred method of connecting diffusion to polysilicon in NMOS technology. As previously explained, the buried contact windows define areas where thin oxide is to be removed so that polysilicon connects directly to diffusion. The contact area between polysilicon and diffusion must be a minimum of $2\lambda \times 2\lambda$ to ensure an adequate contact area and the buried contact window must surround this contact area by λ in all directions to avoid any part of this area forming a (parasitic) transistor. Similarly, a buried contact window must be separated from its related transistor gate by λ to prevent the gate area from being reduced. Figure 3.2 illustrates the rules appertaining to buried contacts.

A more compact method of buried contact connection, shown in figure 3.8, can be used where the channel length is long or is not critical. Here the gate length is dependent upon the alignment of the buried contact mask relative to the polysilicon and can therefore vary by $\pm\lambda$ from its nominal value.

Metal connects to polysilicon or diffusion via contact cuts. Again the contact area must be $2\lambda \times 2\lambda$ to ensure an adequate contact. The metal and polysilicon or diffusion must overlap this contact area by λ so that the two desired conductors encompass the contact area despite any mis-alignment between the conducting layers and the contact hole. Contact holes are spaced 2λ from any gate regions to ensure that no contact to any part of the gate is attempted. These rules for contact cuts are shown in figures 3.2, 3.5 and 3.6. The minimum separation

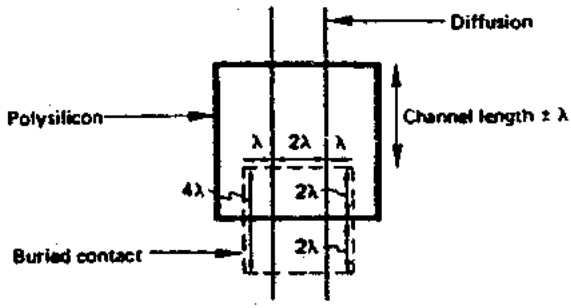


Figure 3.8 Alternative buried contact connection

of holes is 2λ to prevent holes from merging. Finally, where large areas of metal are to be connected to large areas of diffusion or polysilicon, several contact cuts should be used; this causes the current flow between the two layers to be more evently distributed and reduces the resistance of the bulk diffusion or polysilicon area.

The rules for CMOS layouts are similar to those for NMOS, except that the rules for depletion implants and buried contacts do not apply. The additional rules for CMOS are shown in figure 3.5 and concern the definition of the n-well area, the threshold implant for the two types of transistor and the definition of the source and drain regions for the PMOS and NMOS devices.

To ensure the separation of the PMOS and NMOS devices, the n-well supporting a PMOS device is spaced 6λ from the active area (diffusion) of the NMOS transistor; this avoids any overlap of their associated depletion regions. The n-well must completely surround the PMOS device's active area, necessitating an overlap of 2λ. The threshold implant mask covers all n-wells and surrounds the n-well by λ. The p^+ diffusion mask defines the areas to receive a p^+ diffusion. It is thus coincident with the threshold mask surrounding the PMOS transistor but excludes the n-well region to be connected to the supply. In addition, a p^+ diffusion is required to effect the ground connection to the substrate. This diffusion mask therefore also defines this substrate region, and the mask should surround the conducting material of this contact area by λ.

It will be noted from figure 3.5 that butting connections are used to connect adjacent areas of semiconductor. As usual, the contact area to each type of semiconductor is 2λ × 2λ, making a total contact area of 2λ × 4λ. Again, the conducting materials must surround the contact area by λ all round. Figure 3.4 shows that both semiconductor regions of the butting contact are at the same level and there is thus no likelihood of the metal fracturing as in NMOS.

Neither NMOS nor CMOS usually allow contact cuts to the gate of a transistor, because of the danger of etching away part of the gate. This represents no real restriction as a relatively large gate area would be required under the design rules to support such a connection! In fact it should be noted that contact cuts increase silicon area, as it is necessary to increase conduction widths to 4λ to encompass them.

In theory, the λ convention and similar design rules allow designs to be fabricated on many processing lines. However, in practice the requirements of individual processing lines usually differ sufficiently to require a redesign of some layers of the geometric layout. For example, a layout based on a CMOS n-well process cannot be directly ported to a p-well process. Similarly, an NMOS layout using buried contacts will require some redesign to run on a process not supporting such contacts. It is therefore advisable that the user knows which fabrication line is to be used before the layout is commenced.

3.8 Stick Diagrams

A direct translation of a complex circuit diagram of many transistors to a geometric layout can be difficult, and may well require a few attempts before a compact layout complying with the design rules is obtained. Progressing to a geometric layout is greatly aided if the circuit is represented in stick diagram form.

The stick diagram is a representation of the circuit in terms of the lines and connections required on each mask level. The diagrams are drawn with the symbols (or colours) associated with the different patterning layers and if, in addition, all transistor aspect ratios are specified then the stick diagram corresponds directly with the lines and connections of the layout. Thus the stick diagram of the NMOS inverter of figure 3.2 is shown in figure 3.9.

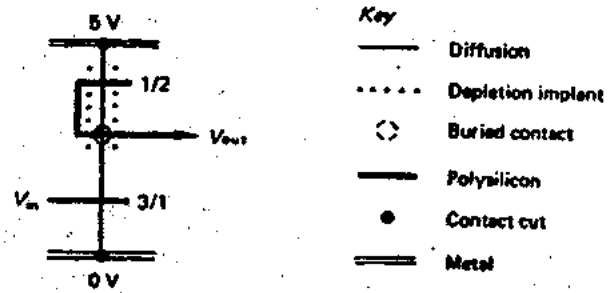


Figure 3.9 Stick diagram of the NMOS inverter with a depletion load

The advantage of a stick diagram is that it allows a circuit to be translated directly and easily into a geometric layout with the aid of the design rules. Furthermore, the stick diagram can be combined with the layout rules to estimate the size of a circuit without the need to draw a full geometric layout.

For example from figure 3.9, the width of the NMOS inverter can be estimated to be 10λ, comprising 6λ for the width of the enhancement mode transistor (since the minimum line length is 2λ and the device's aspect ratio is 3/1) and 2λ on either side of the gate region to ensure that the device's drain and source do not short by reaching round the gate area.

Stick diagrams are a very positive aid to translating from a circuit to a layout and their use prior to the layout stage is recommended.

3.9 Further Reading

T. W. Griswold, 'Portable design rules for bulk CMOS', *VLSI Design*, September (1982) pp. 62–7.

J. Mavor, M. A. Jack and P. B. Denyer, *Introduction to MOS LSI Design*, Addison-Wesley, 1983.

C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

4 MOS Logical Circuit Design

Circuit design is the realisation of the required logic for a system in terms of transistor circuits. The design objective of this stage is to produce a circuit which optimises the often conflicting requirements of minimum silicon area, minimum power consumed and maximum circuit speed. In addition, the difficulty of managing the design of a large system on a chip necessitates the adoption of repetitive, simple structures; this increases the chances of getting a working chip at the first attempt and greatly reduces the design time compared with an *ad hoc* approach.

4.1 Combinational and Sequential Logic

Logic design normally comprises combinational logic and memory elements. No storage is associated with combinational logic circuits. Thus the output of such a circuit can be regarded as responding to its inputs according to the logic function being performed. If the delay through the circuit from applying an input to the output responding is T_d, then the output at time $T + T_d$ is a function of the inputs at time T.

There are two approaches to implementing combinational logic: these are random logic and the transistor array. In the former, a desired logic function is directly implemented as a circuit. The implementation may be a single circuit or the function may be subdivided and a (simpler) circuit designed for each part; this latter method has the advantage of producing circuits which can normally be used to realise other logic functions in the design. Thus combinational logic designed in this way tends to comprise a set of special-purpose functional cells. While such an *ad hoc* method tends to maximise the circuit speed, the design time is usually large compared with a more structured approach.

The alternative method of designing combinational logic is that of a transistor array. Here a two-dimensional, regular array of transistor positions is used to implement functions by placing transistors at the appropriate positions. The array is a general-purpose circuit whose structure is simple as well as regular. The design time can therefore be short and such a technique is highly suited to Computer Aided Computer Design (CACD) assistance.

A memory element output is a function of its inputs and sometimes its output at a previous time. The two main methods for implementing such devices are

static and dynamic circuits. Storage elements based upon a static technique rely on feedback to maintain the output state (indefinitely) until it is altered. Dynamic circuits store states as charge upon a capacitor and since charge can leak from the capacitor, the data has to be periodically refreshed.

New data is normally entered into a memory element when a timing pulse or clock is applied to it. The timing strategy is usually determined at the system level and clearly the use of dynamic circuits imposes a minimum frequency on the system. Despite the need to refresh data and the fact that the clock-driving circuits for dynamic elements tend to be more complex than those for static circuits because of loading considerations, dynamic elements have the advantage that they use fewer transistors per memory bit, occupy less silicon area and consume less power than a static circuit. For this reason, both static and dynamic elements are used in a system, depending upon the application.

This chapter examines how the fundamental circuits of chapter 2 can be developed to implement combinational and memory functions using the techniques outlined in this section.

4.2 Random Logic

Any system can be constructed from nand or nor gates and figure 4.1a depicts a two-input nand gate in terms of voltage-controlled switches. Here, a high level on A and B closes switches S1 and S2, causing V_{out} to be connected to 0 V. If either A or B or both are low, then either S1 or S2 or both switches are open and there is no connection between V_{out} and 0 V; no current flows in the load and thus V_{out} is high at 5 V.

Figure 4.1b shows the realisation of this nand gate in NMOS technology. A high level on A and B turns T1 and T2 on and the output level depends on the aspect ratios chosen for the transistors. Logic families are usually based upon a standard basic circuit so that the different functions of the family have the same output characteristics, such as output voltage levels and current capability. If this convention is adopted for NMOS logic and functions are designed to display similar output features to those of the NMOS inverter of section 2.9, then the nand gate has a low level output voltage of 0.3 V and a pull-up transistor aspect ratio of 1/2.

For an n-input nand gate, the drain-source voltage of each pull-down transistor is approximately $0.3/n$ V compared with 0.3 V for the inverter. It is thus necessary to increase the aspect ratio of all pull-down transistors in the nand gate to $3n/1$. A 6/1 aspect ratio is therefore required for T1 and T2 in figure 4.1b.

It should be noted that the input and output capacitance of the nand gate are greater than those for the inverter. The additional output capacitance is due to an increase in the diffusion area connected to V_{out}, arising from the increase in the width of the pull-down transistors. More importantly, the capacitance of each input is increased by a factor of approximately n compared with the inverter

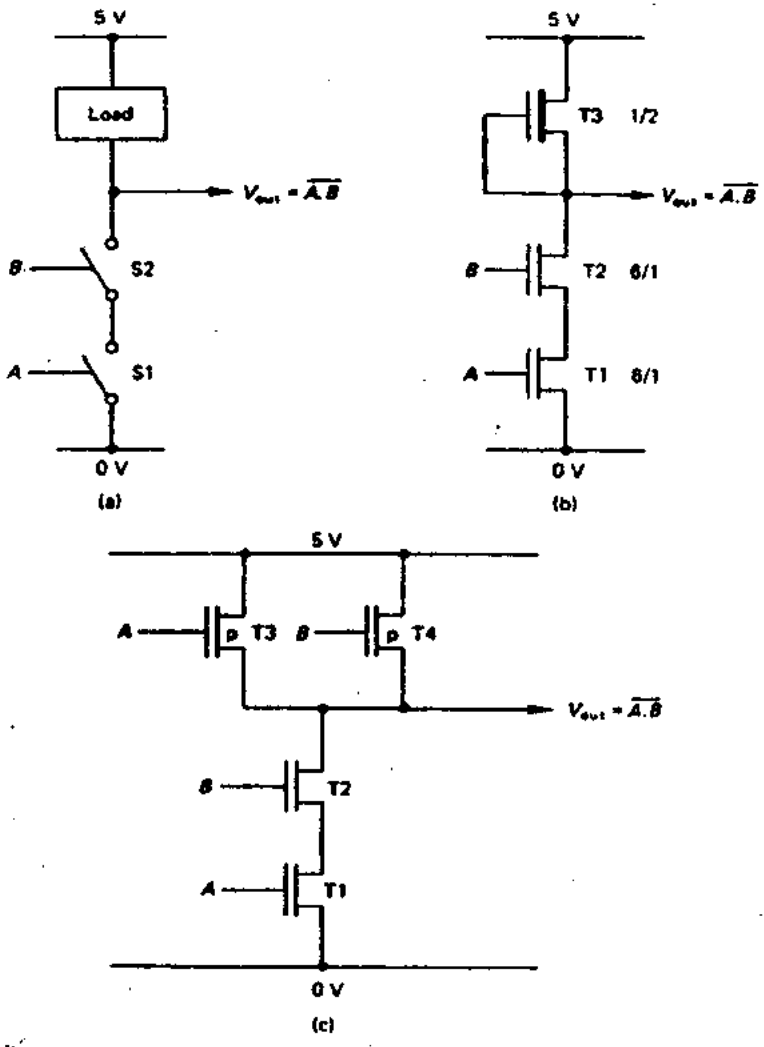


Figure 4.1 A two-input nand gate: (a) switches, (b) NMOS technology, (c) CMOS circuit

input, owing to the increased gate area. This adversely affects the input and output edge times of the nand gate. As a result, it is inadvisable to use nand gates with a fan-in greater than 4.

A CMOS nand gate is obtained by replacing the load with PMOS transistors placed in parallel, as shown in figure 4.1c. If A and B are high, T1 and T2 are on, and T3 and T4 are off; thus V_{out} is low. For all other combinations of A and B, at least one of T1 and T2 is off, and at least one of T3 and T4 is on, connecting V_{out} to 5 V.

A CMOS circuit requiring n pull-down transistors has n pull-up transistors. In general, n-channel transistors connected in series in the pull-down circuit have their associated p-channel transistors connected in parallel in the pull-up circuit and vice versa. This can lead to some unwieldy and large structures compared with the equivalent NMOS circuit which only requires $n + 1$ transistors (see figure 4.3). This is particularly true if the transistor aspect ratios in CMOS circuits are chosen so as to give similar edge times to the CMOS inverter. It is therefore usual to use minimum geometry transistor sizes for all transistors in CMOS circuits, and to accept the resulting slower edge times plus the disparity between rising and falling edge times arising from the difference between hole and electron mobility.

Figure 4.2a shows a two-input nor gate in the form of switches. If A is high, switch S1 closes while switch S2 closes if B is high; in either case V_{out} is low. Thus V_{out} is high only if both A and B are low so that S1 and S2 are both open.

The NMOS two-input nor gate is shown in figure 4.2b. Again the circuit has output characteristics similar to those for the NMOS inverter. It should be noted that V_{out} is 0.3 V or less if at least one input is high. Hence the aspect ratio of the pull-down transistors is 3/1. The output is 0.3 V if only one input to the nor gate is high. If A and B are both high, T1 and T2 are on and the saturation current from the pull-up transistor T3 splits between T1 and T2, causing V_{out} to become approximately 0.15 V. In general, an n-input nor gate with m high inputs has a V_{out} of $0.3/m$ V.

The output capacitance of the nor gate is increased compared with that for the NMOS inverter as a result of the additional diffusion area in the pull-down circuit. However, the capacitance of each input is identical to that for the inverter. Thus it is advantageous in terms of the load on the driving logic to use nor gates rather than nand gates, and nor gates should be used in preference to nand gates in designs wherever possible.

The CMOS equivalent of a two-input nor gate is obtained by replacing the load with two PMOS transistors in series, as shown in figure 4.2c. If A and B are low then T3 and T4 are on, and T1 and T2 are off; thus V_{out} is high. For all other combinations of A and B, at least one of T1 and T2 is on, and at least one of T3 and T4 is off, making V_{out} low.

A system constructed from just nor or nand elements tends to contain a large number of elements and constrains the user to thinking in terms of primitive functions. It does not take advantage of the possibilities offered by the technology. It is thus far better, if a random logic approach is adopted, to think in terms of the functions required in a design and to implement these; this is particularly applicable if such special-purpose functions can be used many times.

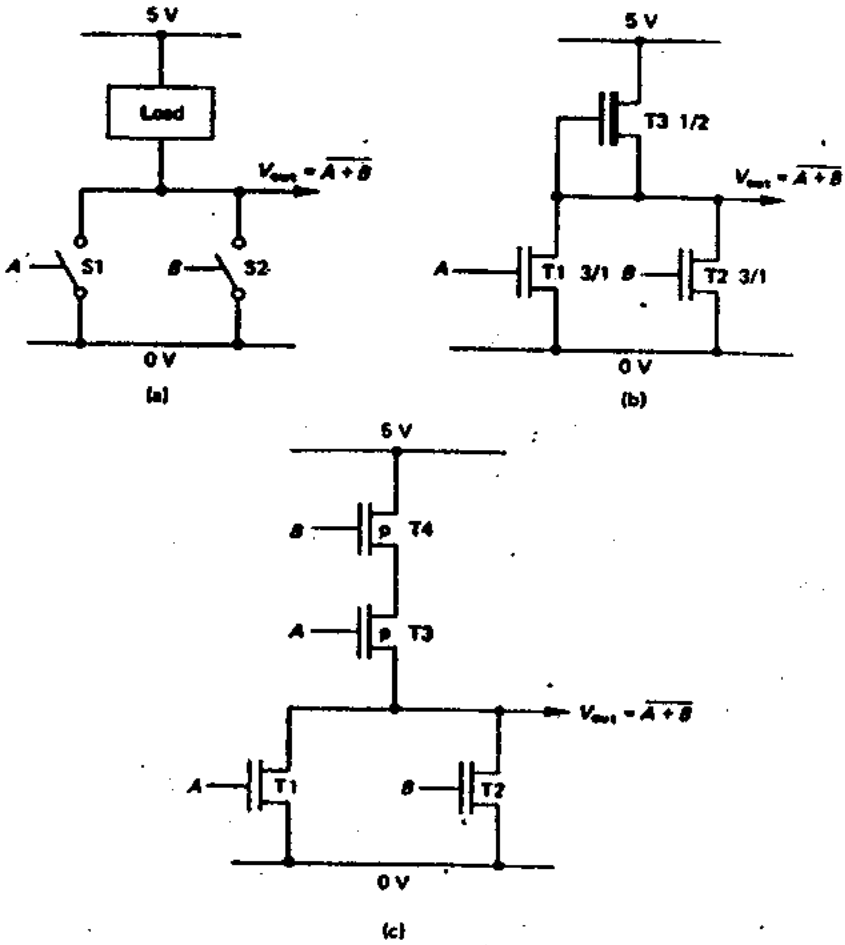


Figure 4.2 A two-input nor gate: (a)-switches, (b) NMOS technology (c) CMOS circuit

For example, the and-or-not function

$$V_{out} = \overline{A + B.C + D.E.F}$$

can be far more efficiently implemented as a single gate than as a set of nor and nand gates. Figure 4.3, parts a and b, illustrate the NMOS and CMOS implementation of this complex logic function. Each branch in the pull-down circuit effects the and operation while the connection of the branches to V_{out} performs the or-not (that is, nor) function.

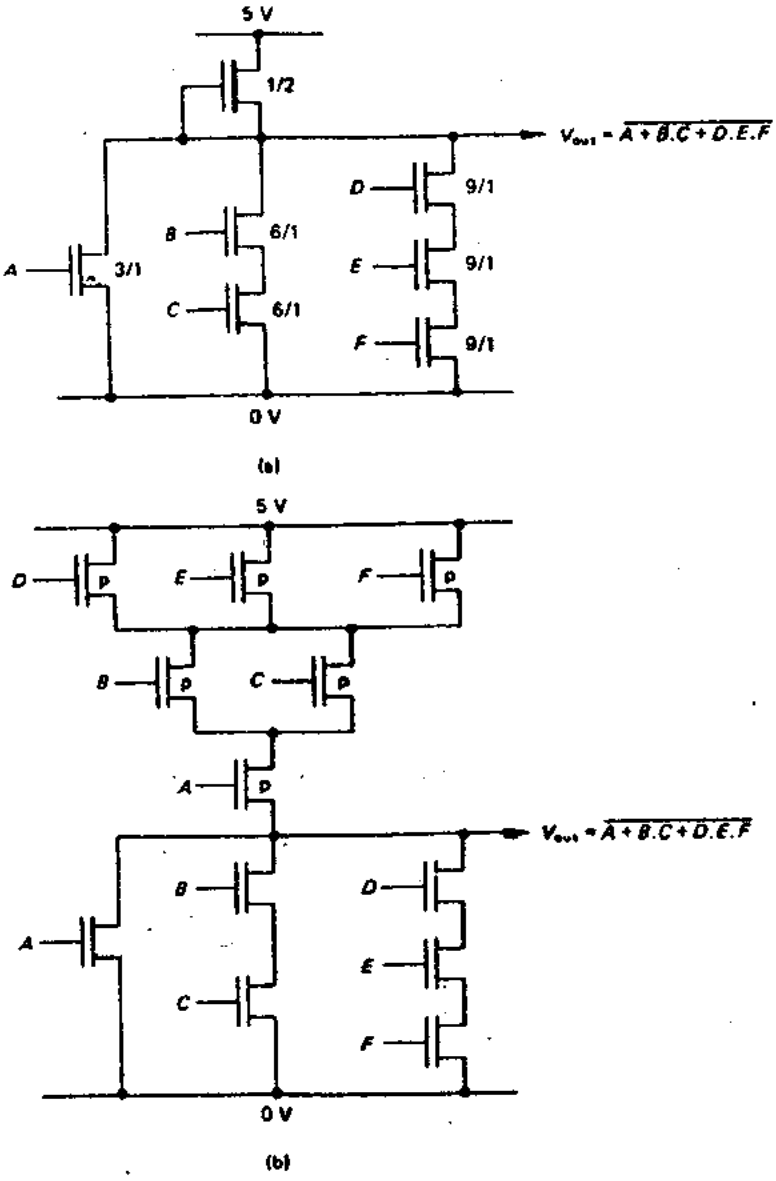


Figure 4.3 A complex logic function: (a) NMOS circuit, (b) CMOS circuit

4.3 Pass Transistor Array

A regular array of transistors can be implemented with pass transistors, and figure 4.4a illustrates the principle of such an array in stick diagram form for NMOS technology. The circuit is a two-to-one multiplexor with an enable. It performs the logic function

$$V_{out} = \text{Enable}.(\overline{\text{Select}}.A + \text{Select}.B)$$

If Enable is high, then V_{out} becomes equal to A or B, depending upon the state of the control input, Select; if Select is low, A is transmitted to V_{out} and if high, V_{out} equals B. When the circuit is not enabled, V_{out} is low.

Since the threshold of a depletion device is −4 V, these transistors can always be considered on with little or no drain–source voltage drop. The output is therefore determined by the enhancement transistors which act as voltage-controlled switches.

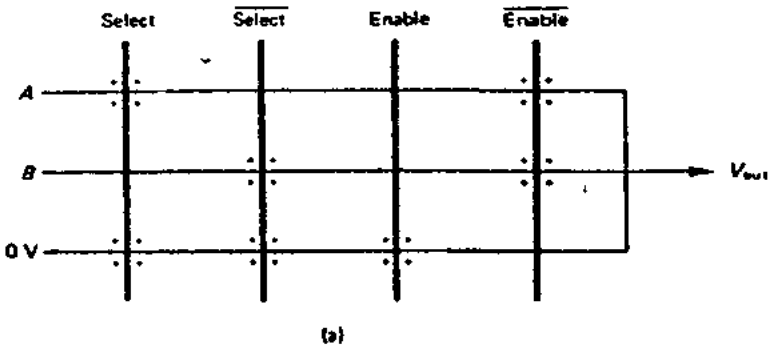
The array as shown performs the and-or function, although the technique can be used to form one or more and functions. Both phases of the input variables normally have to be formed and it is necessary to form an output for every input combination, so that the output is always at a defined level. In general, the array has to be designed so that only one row of the array is activated at any time; thus no path exists between the input to a row and the input to any other row, preventing interference and current flow between inputs.

The 3 x 4 transistor matrix shown in figure 4.4a encompasses all possible input combinations. The top row is activated when Enable is high and Select is low, causing the output to be A. The middle row is on when Enable and Select are high, and B is passed to V_{out}. The bottom row is activated if Enable is low and 0 V is transmitted to V_{out}. Thus one row (and only one row) is always activated.

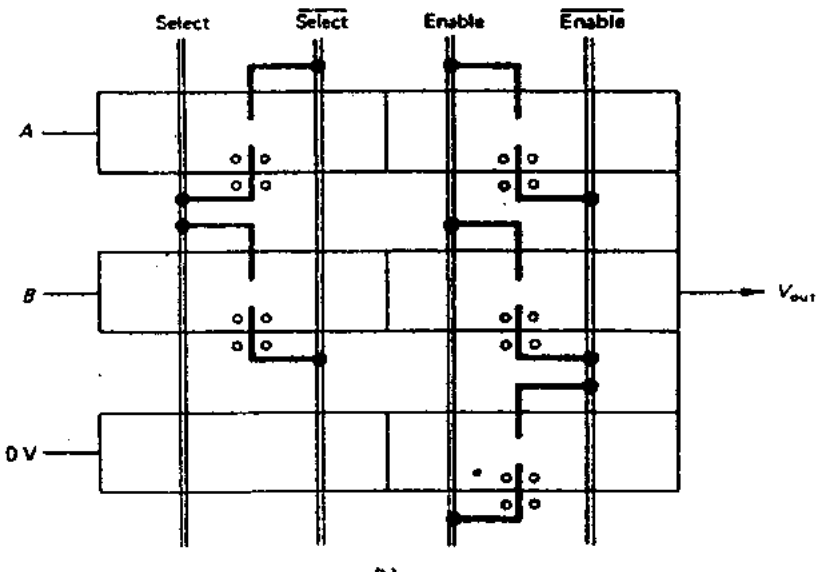
Clearly, the array is well suited to implementing functions where the control inputs applied to the enhancement transistor gates are common to several rows. Thus functions such as the complex expression illustrated in figure 4.3 would be inefficient to implement with this type of array.

A CMOS pass transistor array performing the same function is shown in figure 4.4b. As depletion devices are not available in this technology, input variables positioned vertically are in metal so that they can pass over input variables running horizontally in diffusion without effect. Alternate rows of diffusion contain NMOS and PMOS devices, and polysilicon stubs from the metal cross the diffusion paths to create pass transistors where required.

The advantages of the NMOS pass transistor array are the small amount of silicon area required since the transistors are minimum geometry with no contact cuts, and very low power consumption as there is no connection between power and ground. However, a depletion device tends to cut off when its gate voltage



(a)



(b)

Key

- Diffusion
- Depletion implant
- n-well, threshold implant, p⁺ diffusion
- Polysilicon
- Contact cut
- Metal

Figure 4.4 Pass transistor array: (a) NMOS, (b) CMOS

is low and its drain and source voltages are high. This effect combined with the delay through a chain of enhancement pass transistors (section 2.13) usually gives this circuit a significantly longer delay time than the equivalent random logic circuit. The delays associated with the depletion transistors can be avoided by organising the array in a similar manner to the CMOS array of figure 4.4b. Here, the depletion transistors are replaced by metal overpasses running over the diffusion. The layout is clearly not as compact as that in figure 4.4a.

The speed–power product of a circuit is often used as a comparison measure between different design styles. This product for the pass transistor array usually compares unfavourably with that for the equivalent random logic circuit. Thus, unless the user's primary requirement is to minimise the area occupied and the power consumed, an alternative transistor array should be used.

4.4 Programmable Logic Array

A programmable logic array or PLA consists of two transistor arrays which combine to form the sum of products function (see figure 4.5). The first array, called the AND plane, performs the and function as required on its N inputs and outputs P product terms. These are input to the second plane, called the OR plane, which performs the or function on the product terms as specified. The S outputs from this plane are thus the sum of products.

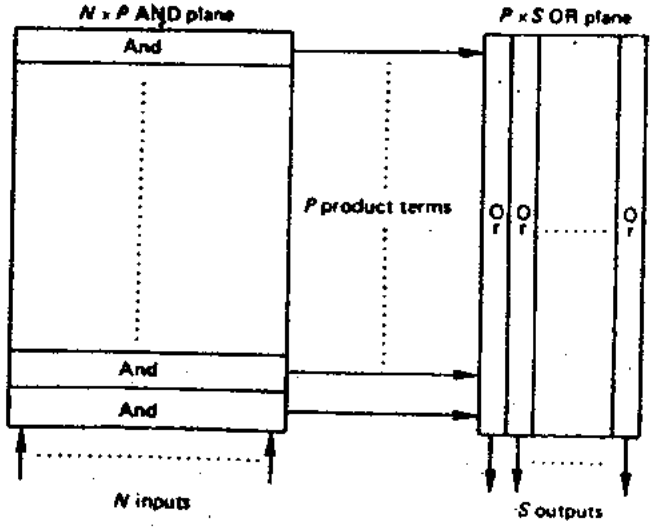


Figure 4.5 General structure of programmable logic array

In practice, since nor gates are used in preference to nand gates, the two planes are implemented as nor arrays and are thus identical in structure except that the OR plane is rotated clockwise by 90° with respect to the AND plane. Since the second array performs the nor function rather than or, it is necessary to invert all outputs from the OR plane. Alternatively, these output inverters can be omitted if the two planes are designed to form the inverse of the inverse sum of products expression. It is also usual to buffer the input signals to the PLA because of the capacitive loading on the AND plane and because an inverter is often required here in any case to provide the inverse of an input signal.

As an example of a PLA design, consider a three-line priority encoder which indicates the highest priority line activated as a two-bit number if the circuit is enabled. The truth table for this circuit is shown in table 4.1. *D0* is the top priority line and *D2* the bottom priority. *A* and *B* both low indicates that the circuit is not enabled or that all priority lines are '0'. If a priority line is activated by taking it to a '1', then *A* and *B* correspond to the highest priority line present.

Table 4.1 Truth table for three-line priority encoder

	Inputs			*Outputs*	
Enable	*D0*	*D1*	*D2*	*A*	*B*
0	X	X	X	0	0
1	0	0	0	0	0
1	1	X	X	1	1
1	0	1	X	1	0
1	0	0	1	0	1

X = 0 or 1 – that is, don't care.

Inspection of the truth table reveals that

$$A = \text{Enable}.D0 + \text{Enable}.D1$$

$$B = \text{Enable}.D0 + \text{Enable}.\overline{D1}.D2$$

Thus the AND plane forms three product terms *P0*, *P1* and *P2* where

$$P0 = \text{Enable}.D0 = \overline{\overline{\text{Enable}} + \overline{D0}}$$

$$P1 = \text{Enable}.D1 = \overline{\overline{\text{Enable}} + \overline{D1}}$$

$$P2 = \text{Enable}.\overline{D1}.D2 = \overline{\overline{\text{Enable}} + D1 + \overline{D2}}$$

P0, *P1* and *P2* are combined in the OR plane to form the sum of products terms *S0* and *S1* where

$$S0 = \overline{P0 + P1} = \overline{\text{Enable}.D0 + \text{Enable}.D1}$$

$$S1 = \overline{P0 + P2} = \overline{\text{Enable}.D0 + \text{Enable}.\overline{D1}.D2}$$

Finally $S0$ and $S1$ have to be inverted to form the desired A and B outputs.

Figure 4.6 shows the PLA implementation of the priority encoder. It can be seen that the arrays are effectively a set of multi-input nor gates whose outputs are distributed along the entire width of a plane. The input lines to a plane run at 90° to the output lines and thus any input can be connected to any output line as required.

Comparing the characteristics of this PLA circuit with those of the equivalent random logic circuit, the and-nor function is implemented in two (nor) stages of logic whereas it is a single level in random logic. The PLA can therefore be expected to operate more slowly than the random logic circuit. The power consumption of this PLA is also more than that for the random logic circuit, owing to the fact that in the PLA a larger number of circuits are involved in forming the output. If the circuit speed is expressed in terms of the circuit delay, then it is apparent that the speed-power product of the PLA will be greater than that of its random logic equivalent. It should also be apparent that large PLAs are impractical because of the high capacitance associated with the input and output lines of the array.

The area occupied by a PLA is more efficiently utilised when inputs are common to several product terms, and product terms are common to several sum of product outputs. Thus figure 4.6 illustrates a function that is efficient in area to implement since the Enable input is common to all three product terms and the product term $P0$ is common to A and B. Even so, only 11 out of 21 possible pull-down transistor positions are used.

A sparse array is inefficient in terms of the space it occupies and the designer should investigate alternative circuit arrangements. A reduction in the space may result from the use of just one plane with the addition of some random logic or by the implementation of a set of (smaller) PLAs arising from the partitioning of the function. Alternatively, the PLA size may be reduced by externally combining some inputs so that a smaller number of inputs and product terms are required.

Another technique to utilise space more effectively is to fold the array. This approach takes advantage of sparseness in rows and columns and is demonstrated in figure 4.7 for the priority encoder example. By suitably arranging the input and output lines in addition to splitting a row and column, advantage has been taken of unused transistor positions in the original arrays. The top row of figure 4.7 can be split because the product term Enable.$D1$ is not used in the formation of B and Enable.$\overline{D1}$.$D2$ is not used in A's formation. The split in the leftmost column is possible as $D0$ only appears in one product term. As a result of this compression, 11 out of 14 pull-down positions are used.

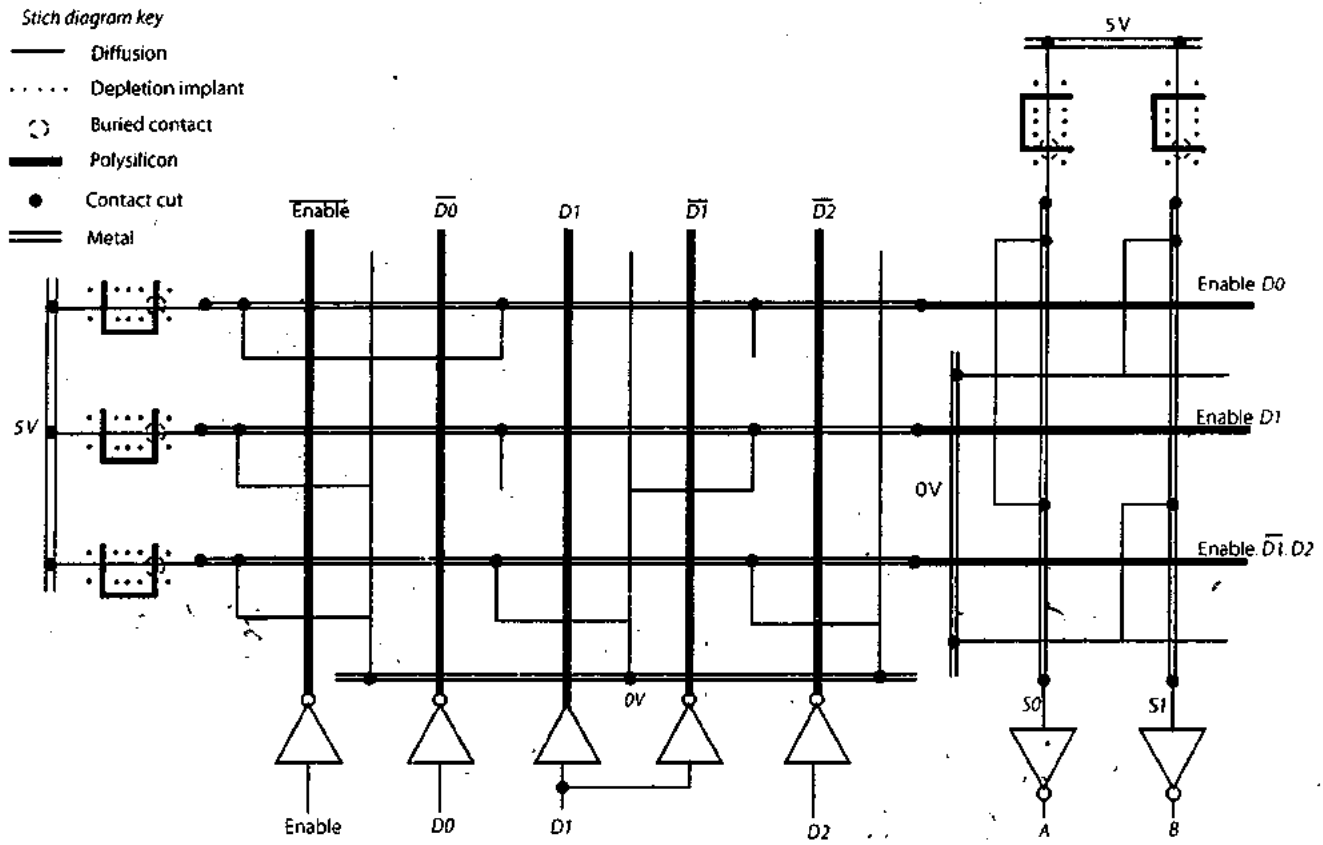


Figure 4.6 PLA for a three-line priority encoder

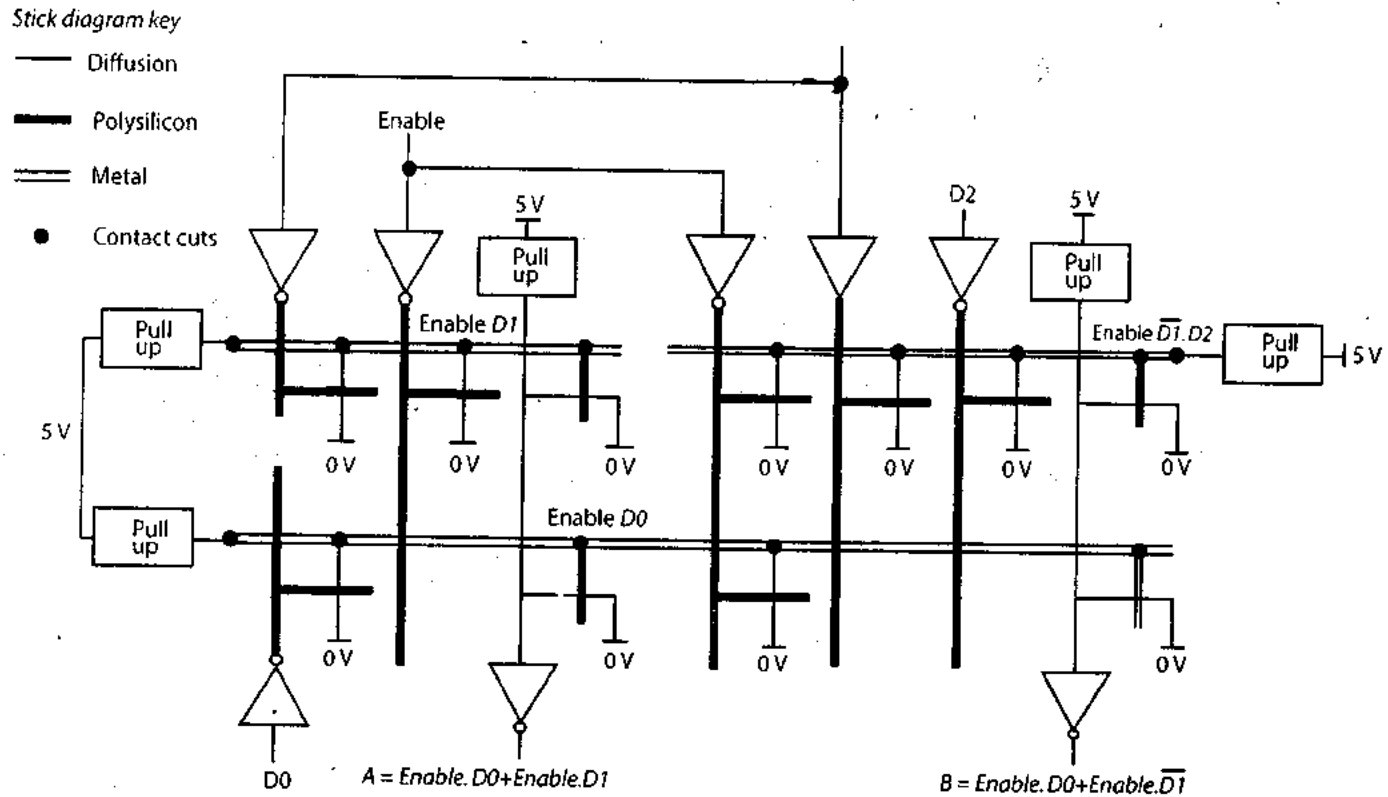


Figure4.7 An Example of a folded PLA

The advantage of the PLA is that it is very easy to design arrays for any logic function and the geometric layout can be implemented quickly, even if it has been hand designed. Such a regular and simple structure lends itself to CACD techniques. This is true at all levels of a PLA's design since a set of rules can easily be formulated which translates the user's Boolean expressions of output functions into a geometric layout. This translation process could also include logic minimisation on the expressions supplied, and possibly fold the PLA. The PLA is likely to function correctly at the first deisgn attempt, particularly if CACD aids are used; this is not so easily achievable with an *ad hoc* approach.

CMOS technology is not so elegantly implemented as a PLA structure because of the nature of its pull-up circuit. Since the arrays implement the nor function, the pull-up circuit consists of PMOS transistors placed in series between an array output and the power rail (see figure 4.2c). For each transistor in the pull-down circuit, there is a corresponding PMOS transistor in the pull-up circuit and thus inputs to the pull-down transistors in both arrays have to be connected to their associated pull-up transistor. Clearly, the CMOS pull-up circuit will significantly increase the area occupied by a PLA compared with its NMOS equivalent.

Figure 4.8 shows how each nor gate in each plane of a CMOS PLA can be modified to avoid the implementation of the conventional CMOS pull-up circuit. The pull-up circuit now consists of a PMOS transistor T1 and the pull-down circuit has an additional NMOS transistor T2 connected in series with the standard pull-down circuit, comprising transistors T3 to T*X*. During the pre-charge period, Pre-charge is low so T1 is on and T2 is off. Thus, regardless of the input levels to the other pull down transistors, there is no connection between the nor gate output and 0 V; the output charges up to 5 V via T1.

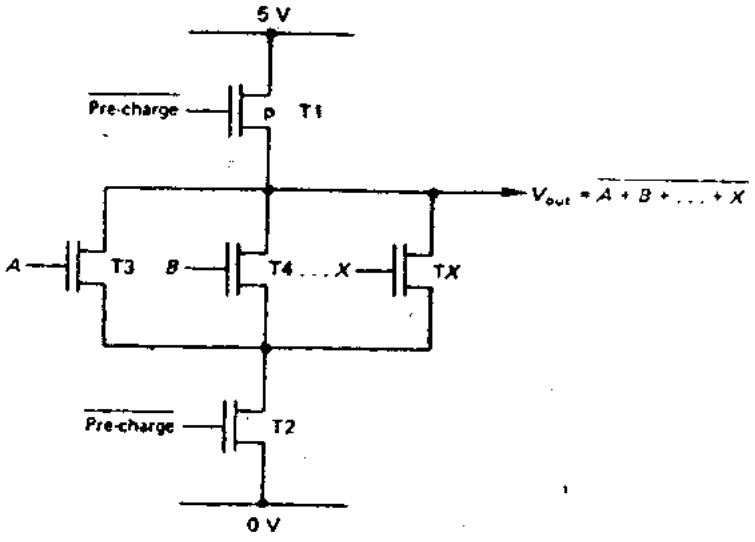


Figure 4.8 A pre-charged nor gate

When Pre-charge is taken high, T1 turns off and T2 turns on. If the gate input to any other pull-down transistor is high, then there is a series path between the array plane output and 0 V, causing the output to discharge to 0 V. However, if the inputs of transistors T3 to T*X* are all low, there is no current path to ground and the output remains at 5 V. Since the pull-up circuit is active when the pull-down circuit is off and vice versa, the circuit operates correctly, regardless of aspect ratios; all transistors can be minimum geometry. The circuit is essentially dynamic and in the case of a high output there is no connection to either rail; the voltage is maintained by the charge on the output capacitance of the nor gate. The two distinct phases of operation required cause this type of PLA to be slower than conventional PLAs and clearly the output is not valid during the pre-charge period.

4.5 Static Flip Flops

A flip flop is a memory device with two stable states, one of which represents the storage of a '0' and the other the storage of a '1'. The basis of a static flip flop arises from cross-coupling two inverters, as shown in figure 4.9 for NMOS technology. Transistors T1 and T3 form one inverter, and T2 and T4 the other inverter.

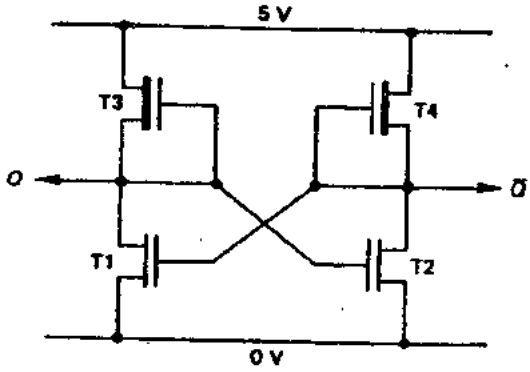


Figure 4.9 Basis of static (NMOS) flip flop

If the gate of T1 is high at 5 V, then T1 is on and Q is low at about 0.3 V. Thus T2 is off, causing $\bar{Q}$ to be high at 5 V. Thus the feedback connections of T1's drain to T2's gate and T2's drain to T1's gate maintain and reinforce this existing state. The other flip flop state arises if the gate of T2 is high at 5 V.

Here, T2 is on and $\bar{Q}$ is low. $\bar{Q}$ holds T1 off so Q is high. Again the feedback maintains the existing state.

By adding additional pull-down transistors in parallel with T1 and T2, the different types of commonly encountered flip flops can be constructed. Figure 4.10a shows that with the addition of transistors T5 and T6, a Set-Reset flip flop is obtained. The circuit is now that of two cross-coupled nor gates, as shown in figure 4.10b.

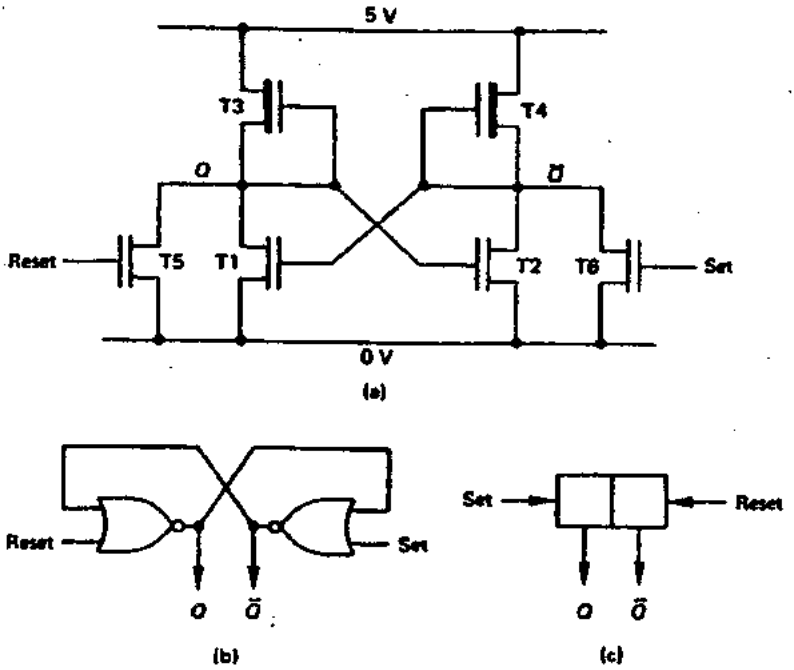


Figure 4.10 (NMOS) Set-Reset flip flop: (a) circuit diagram, (b) logic diagram, (c) logic symbol

Normally Reset and Set are low so T5 and T6 are off and the existing state is maintained by the feedback. Consider that the flip flop is in the reset state with Q low and $\bar{Q}$ high. To switch the flip flop to the set state (Q high, $\bar{Q}$ low), Set is taken high. This turns T6 on, forcing $\bar{Q}$ low. With Reset low, both T1 and T5 are off, so Q goes high, turning T2 on. This reinforces the low on $\bar{Q}$. When Set is removed (that is, goes low), the feedback between T1 and T2 maintains this state. To change back to the reset state, Reset is taken high while Set remains

low. This turns T5 on, forcing Q low which in turn switches T2 off. Since T2 and T6 are off, $\bar{Q}$ rises, turning T1 on so that the reset state continues when Reset returns to '0'.

It should be noted that if Set and Reset are high simultaneously, then Q and $\bar{Q}$ become low at this time. If both inputs are removed together, then the final state of the flip flop cannot be predicted. Such an indeterminate state should, of course, be avoided in designs.

The storing of data in a flip flop is often synchronised to a timing signal. Such clocked flip flops are either edge or level triggered, depending upon the circuit design. In the former type, the flip flop is clocked when the clock level changes. If this change is from a low to a high level the device is said to be positive edge triggered, while a negative edge triggered flip flop enters new data on a high to low clock change.

Data to the flip flop must be valid and remain constant before and after the clocking edge for times referred to as the set-up and hold time respectively. These times, which are usually of the order of a few nanoseconds, ensure that the flip flop fully switches when the trigger edge is applied. If the inputs are changed during the set-up or hold time then, although the flip flop starts to respond to this change, at the end of the hold time the outputs may be in a partially switched state.

On the expiry of the hold time, the data inputs and clock are effectively isolated from the circuitry. The circuit thus reverts to the basic flip flop of figure 4.9. In the case of partially switched outputs, the feedback operates to magnify any voltage difference between the outputs, causing the flip flop to eventually switch to a valid logic state. The time for a flip flop to settle under these circumstances depends upon the initial voltage difference between the outputs. This time gets progressively longer as this voltage difference gets smaller and, for the case of equal outputs, the flip flop can remain indefinitely in its partially switched state. Even if a partially switched device does settle, the final logic state cannot be predicted.

When the input data and clock are asynchronous, partial switching can occur and it is usual in these circumstances for a system to allow a settling time after the clock edge for the flip flop outputs to become valid. This time is usually chosen to be significantly longer than the propagation delay so that a flip flop very rarely fails to reach a valid logic state. Even so, it should be appreciated that such a system will be subject to occasional failure. Such timing conflicts can be avoided by the use of synchronous timing techniques, which for this reason are recommended for use in chip designs.

Level-triggered flip flops enter new data into the device when the clock is at a '1' level. Again a set-up and hold time are associated with the clock but here the times refer to the time before and after the negative clock edge, since this is the time the clocking input to the flip flop is removed. Again, if the data is changed during the set-up or hold time then the flip flop can switch to an indeterminate state which may or may not eventually settle to a valid logic state.

In addition to the ability to clock in new data, many flip flops have a set and/or reset facility which is not usually synchronised to the clock. Depending upon the exact details of the circuit, set and reset may override the clock or vice versa.

The two most common types of clocked flip flop are the *D*-type and the *J-K*. In the *D*-type, the data or *D* input is copied to the *Q* output when the clock is applied. Figure 4.11 shows the circuit for a level-triggered *D*-type flip flop with an overriding set and reset input. The truth table for its operation is given in table 4.2.

Table 4.2 Truth table for *D*-type flip flop

Inputs				*Outputs*	
Set	*Reset*	*Clock*	*D*	*Q*	$\bar{Q}$
0	0	0	X	*Q*	$\bar{Q}$
0	0	1	0	0	1
0	0	1	1	1	0
1	0	X	X	1	0
0	1	X	X	0	1
1	1	X	X	0	0 – indeterminate

X = 0 or 1

Q is low if the gate voltage on transistors T1 or T5 or T7 and T9 and T11 is high while $\bar{Q}$ is low if the gate voltage on T2 or T6 or T8 and T10 and T12 is high. Thus with Set, Reset and Clock low, the existing state is maintained by the feedback between T1 and T2.

To set the flip flop, Set is taken high. This turns T6 on and forces $\bar{Q}$ low. Since $\bar{Q}$, Reset and Set are low, T1, T5 and T11 are off so all pull-down branches connected to *Q* are off. Thus *Q* is high and T2 is on, reinforcing the low level on $\bar{Q}$. The feedback between T1 and T2 maintains this state when Set is removed. Note that Set connected to T11's gate is necessary to ensure that the flip flop is set, regardless of the clock level.

Similarly, the flip flop is reset by taking Reset high and leaving Set low. This turns T5 on, forcing *Q* low, and since no pull-down branch on $\bar{Q}$ is on, $\bar{Q}$ is high. T1 and T2 maintain this state when Reset is removed. Again, the flip flop enters an indeterminate state if Set and Reset are simultaneously high.

Set and Reset are inoperative when they are low. In this mode, data is entered into the flip flop when Clock is high. A high level on *D* in these circumstances turns the series chain of transistors T8, T10 and T12 on, forcing $\bar{Q}$ low while no pull-down branch on *Q* is on since T1, T5 and T9 are off. Thus *Q* is high. Alternatively, a low level on *D* causes T7, T9 and T11 to be on, forcing *Q* low while $\bar{Q}$ is high, since T2, T6 and T10 are off. Thus with Set and Reset inactive, the *D* input is copied to the *Q* output when Clock is high.

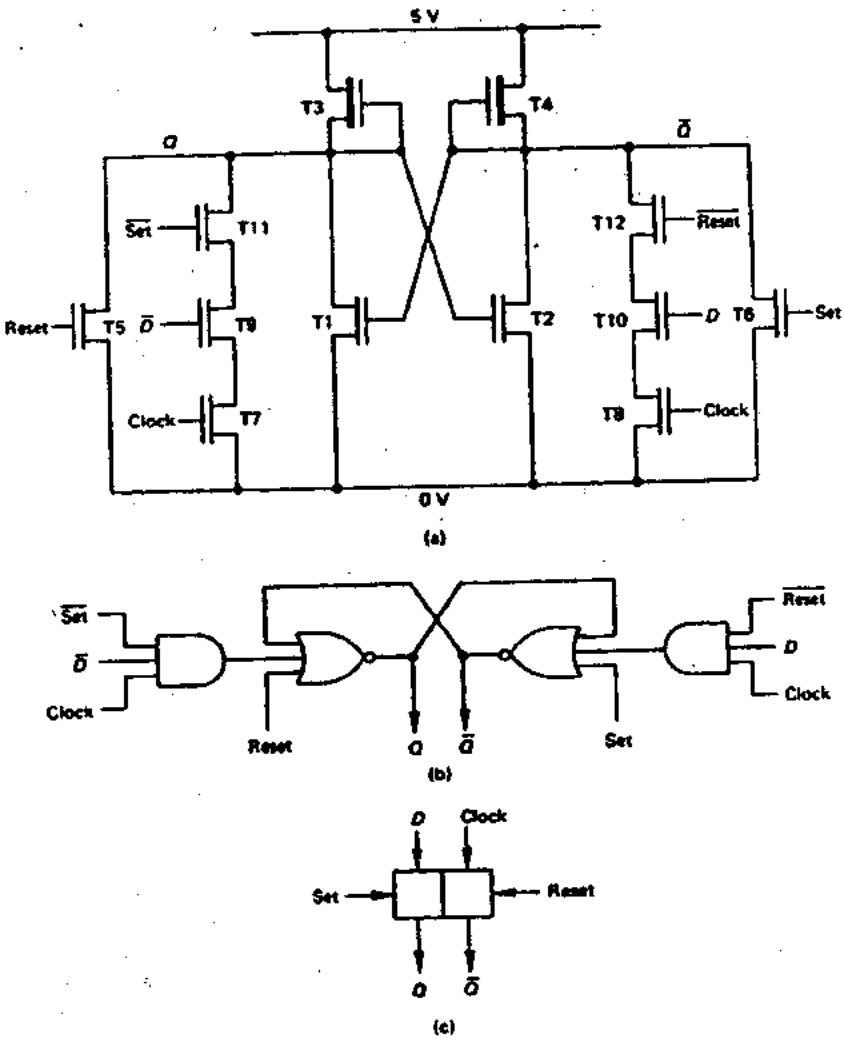


Figure 4.11 *D*-type flip flop with overriding Set and Reset: (a) circuit diagram, (b) logic diagram, (c) logic symbol

The circuit can be simplified if restrictions are placed on the flip flop's operation. For example, T11 and T12 can be omitted if the flip flop is only set or reset when the clock is '0'. Alternatively, if set and reset facilities are not required, T5, T6, T11 and T12 are unnecessary.

A *J–K* flip flop is an edge-triggered device whose operation is determined by the *J* and *K* inputs when the clock is high. Its outputs indicate any new state when the clock is removed and its truth table is given in table 4.3.

Table 4.3 Truth table for *J–K* flip flop

Inputs			*Outputs*		
J	*K*	*Clock*	*Q*	$\bar{Q}$	
0	0	1 → 0	*Q*	$\bar{Q}$	
1	0	1 → 0	1	0	
0	1	1 → 0	0	1	
1	1	1 → 0	1 → 0	0 → 1	Change state
			or 0 → 1	1 → 0	

It can be seen from the truth table that the use of the *J* and *K* inputs allows the flip flop to continue with its current state (*J* = '0', *K* = '0'), to be set (*J* = '1', *K* = '0'), to be reset (*J* = '0', *K* = '1') or to toggle – that is, to change state – (*J* = '1', *K* = '1') on the negative clock edge.

The last case is an example of the flip flop outputs in one time interval determining the outputs in the next time period and in order to implement this feature, the *J–K* device consists of two flip flops known as the 'master' and 'slave'. The operation of these two flip flops is mutually exclusive. Thus the master only clocks new data in when the slave is maintaining its outputs constant. Similarly, the slave only clocks data in when the master's outputs are constant.

This can be implemented by applying a clock to the master and generating its inverse which is applied to the slave. However, there is some overlap where both the master and slave clocks are high and as a result, correct operation is dependent upon the (master) flip flop propagation delay exceeding the overlap. To avoid the possibility of timing conflicts where the master and slave simultaneously clock in data, non-overlapping clocks are used (see figure 4.12). The master and slave clocks, $\phi 1$ and $\phi 2$, are generated from the top waveform shown. When the master clock is high, the slave clock is low and vice versa. Furthermore, when one of these clocks is removed, both are low for a period before the other clock is applied.

Figure 4.13 shows the NMOS circuit of a *J–K* flip flop. The slave outputs, *Q* and $\bar{Q}$, are inputs to the master flip flop and the master outputs, Q_m and $\bar{Q}_m$, are inputs to the slave. The master operates when $\phi 1$ is high and new data can be entered into it according to the levels of the *J* and *K* inputs and the slave outputs. The slave operates when $\phi 2$ is high and the master outputs are copied into the slave. Note that the *J–K* flip flop outputs are taken from the slave and that, since the master outputs are constant during $\phi 2$, the slave only changes state when $\phi 2$ is first applied. Thus the *J–K* flip flop is effectively edge triggered.

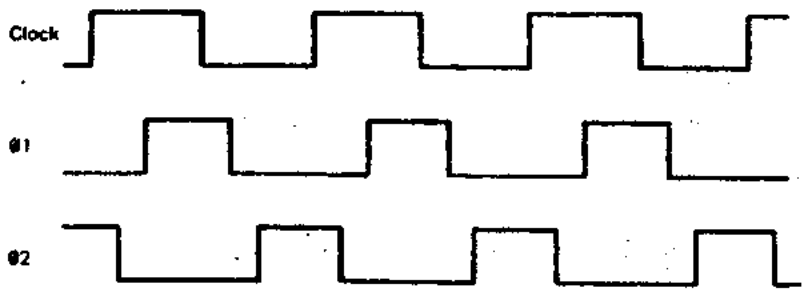


Figure 4.12 Two-phase non-overlapping clocks

On $\phi 1$, Q_m is forced low if the gate inputs to transistors T5 and T7 and T9 are all high, while $\bar{Q}_m$ is forced low if T6 and T8 and T10 turn on. Thus if J and K are low, T8 and T7 are off, so regardless of the slave outputs, the existing states of Q_m and $\bar{Q}_m$ are maintained by the feedback between T1 and T2.

If J is high and K is low, then T6, T8 and T10 only all turn on during $\phi 1$ if the current state of the master (as recorded by the slave) is the reset state (Q_m and Q low); in this case $\bar{Q}_m$ is forced low and since all pull-down branches connected to Q_m are off, Q_m rises. If the master is already in the set state when $\phi 1$ is applied, then T7 and T10 are off and the existing state continues. Similarly if K is high and J low on $\phi 1$, T5, T7 and T9 only all turn on if the master is in the set state. In this case, Q_m falls and $\bar{Q}_m$ then rises since T2 and T8 are off.

If J and K are both high when $\phi 1$ is applied, then one of the T5, T7 and T9 or T6, T8 and T10 branches turns on. If the slave Q output is high (indicating that Q_m is currently high), then T5, T7 and T9 turn on since their gate inputs are all high; this causes Q_m to fall to a low level and $\bar{Q}_m$ then to rise. Alternatively, if Q is low, then T6, T8 and T10 turn on, forcing $\bar{Q}_m$ low and Q_m high. Thus with J and K high, the feedback from the slave outputs has been used to change the state of the master.

The slave operates on $\phi 2$. If Q_m is high at this time, then T16 and T18 turn on, forcing $\bar{Q}$ low. Since T17 is off if T18 is on, Q adopts a high level. Alternatively, if Q_m is low when $\phi 2$ goes high, T15 and T17 turn on, forcing Q low; T18 is off and thus $\bar{Q}$ is high. Thus the master outputs are copied into the slave when $\phi 2$ is applied. When $\phi 2$ is removed, the feedback between T11 and T12 maintains the levels on Q and $\bar{Q}$.

For applications such as shifting and counting where flip flop outputs are connected to other flip flop inputs and outputs change simultaneously, a master and slave flip flop must be used for each bit to ensure correct operation. Such an arrangement allows the next required state to be entered into the masters while the current state is maintained constant by the slaves. This next state is then

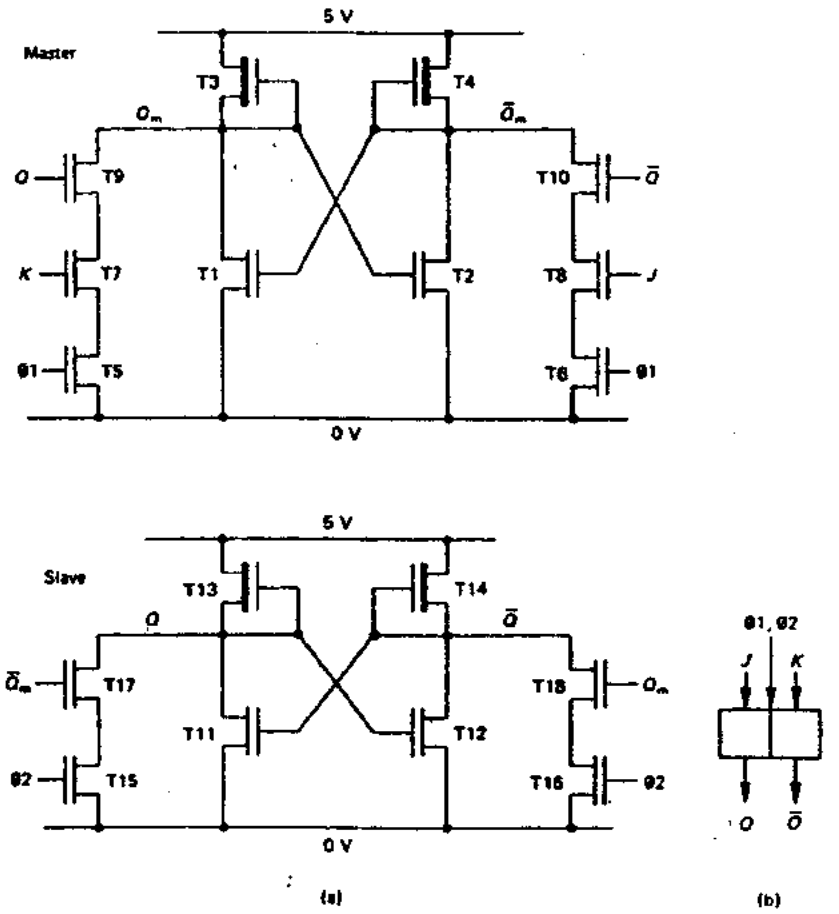


Figure 4.13 An NMOS *J-K* flip flop: (a) circuit diagram, (b) logic symbol

copied into the slaves on the slave clock. If a master and slave implementation is not adopted, then outputs are liable to change at the same time as they are being clocked as data into other flip flops; this can result in the new output state being erroneously clocked in.

Counting and shifting applications require either one *J-K* or two *D*-type flip flops per bit. However, when implementing a register, only one flip flop per bit is necessary so a *J-K* flip flop is needlessly complex here. Hence, the designer should use the flip flop type best suited to a particular section of the design. Furthermore the inclusion of redundant facilities should be avoided as this will minimise the number of transistors and hence the silicon area occupied.

The remarks in this section apply equally well to CMOS circuits and it will be appreciated that all the flip flop designs presented can be converted to a CMOS version by replacing the depletion transistor pull-up in NMOS by a CMOS pull-up circuit.

4.6 Dynamic Flip Flops

The basis of a dynamic flip flop is the combination of a pass transistor and an inverter, as shown in figure 4.14 for NMOS technology. A high level on the Clock input causes the pass transistor T1 to turn on and the voltage level on the Data input determines the voltage at the gate of T2. A low input voltage causes this voltage to be passed to T2's gate while a high input of V_p suffers a threshold voltage loss of V_{te} in transmission to the input of T2. When the Clock input is taken low, the level on T2's gate is maintained by the capacitance inherent in the circuit at this point, which is principally the gate capacitance of T2.

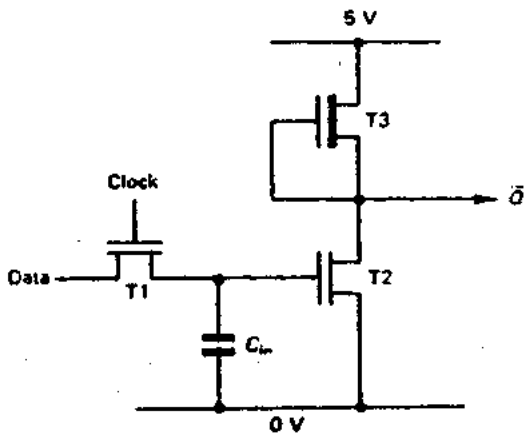


Figure 4.14 Basic dynamic flip flop

T2 and T3 act as an inverter with the aspect ratios chosen to restore the high level input voltage of T2 to the standard voltage level for a logic '0' output. Thus the flip flop is a level-triggered *D*-type device with an inverse output, as shown in table 4.4

Table 4.4 Truth table for dynamic flip flop

Inputs		*Output*
Clock	*Data*	$\bar{Q}$
0	X	$\bar{Q}$
1	0	1
1	1	0

X = 0 or 1.

The clock width must be sufficient to allow the inverter's input capacitance to charge to its high level voltage and this determines the maximum frequency of operation. The minimum operating frequency is determined by the need to retain the data stored, despite the loss of charge on the capacitor as a result of leakage currents. Therefore, dynamic flip flops are best suited to applications where data is regularly clocked rather than indefinitely held or infrequently clocked. Thus the use for such devices particularly arises in designs where data is continually shifted, as for example in the transmission of data streams, in serial-to-parallel or parallel-to-serial data conversion, in data pipelines and in shift register applications.

Figure 4.15a shows how data can be progressively passed through a chain of *n* flip flops. The devices are clocked using two-phase non-overlapping clocks with odd-numbered flip flops clocked on $\phi 1$ and even-numbered devices on $\phi 2$.

On $\phi 1$, the even-numbered flip flops are inactive and their constant data outputs are clocked into the succeeding odd-numbered flip flop. When $\phi 1$ is removed, this data is held constant at the outputs of the odd-numbered flip flops. On the next $\phi 2$ clock, the even-numbered devices are clocked and the data held in the preceding (odd) numbered flip flop is entered.

The progression of data entered into the chain is shown in the timing diagram of figure 4.15b. Initially, Data is a logic '0'. On the first $\phi 1$ clock, this is entered into flip flop 1, causing its output to be the inverse of its data input – that is, a logic '1'. The first $\phi 2$ clock enters the output from flip flop 1 into flip flop 2, so the output of flip flop 2 becomes a '0' at this time. The following $\phi 1$ clocks flip flops 1 and 3. New data is entered into flip flop 1 and the data output of flip flop 2 is clocked into flip flop 3. In this way, data entered into the chain propagates one stage down the chain at each clock pulse with the inverse form of the data stored in odd-numbered flip flops and the true phase in even-numbered devices.

An essential feature in the design of shifting circuits is the use of $\phi 1$ and $\phi 2$ for alternate flip flops in the chain. This clocking arrangement ensures that on a clock pulse, data stored in a device can only be shifted to the next stage in the chain. A common clock for all devices cannot be used as the data input of a stage would not remain constant during clocking.

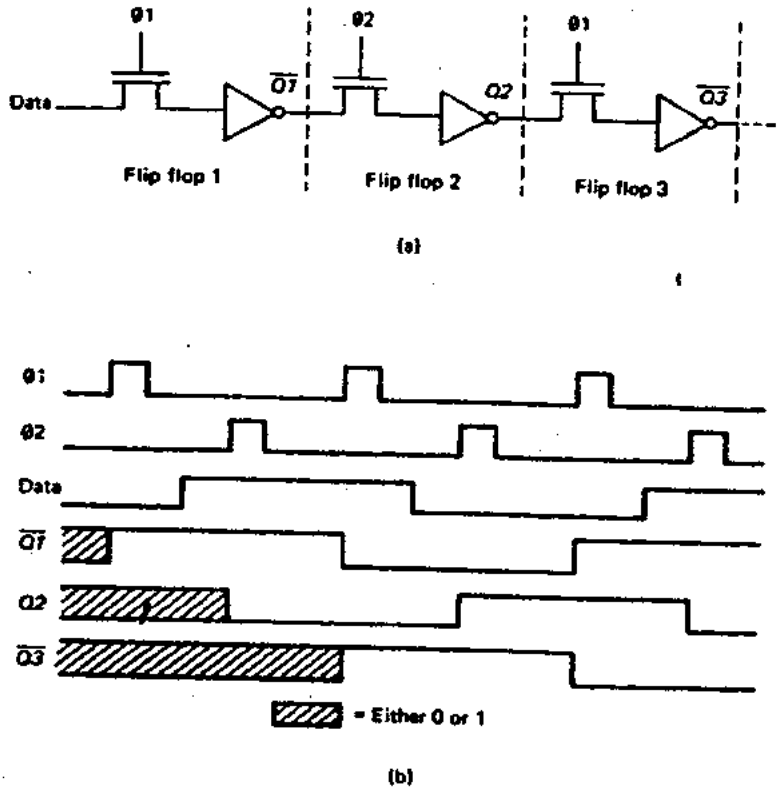


Figure 4.15 A shift chain using dynamic circuits: (a) circuit diagram, (b) timing

Usually the width of the data path is greater than the single bit depicted in figure 4.15a and an identical circuit would be used for each data bit. The flip flops for such an arrangement at a particular stage in the chain are collectively referred to as a 'register', and registers are often connected via combinational logic blocks; a general organisational schematic is shown in figure 4.16.

On ø1, the odd-numbered registers are loaded with data from the preceding (even-numbered) combinational logic block. This data is then operated upon by the succeeding (odd-numbered) logic block and these block outputs are clocked into the even-numbered registers on ø2. The timing of data through the pipeline has to allow for the delay through the slowest combinational logic block plus the register set-up and propagation times. The most effective use of the logic results if the delay time through each logic block is similar.

There is, of course, no reason why some of the outputs from registers or combinational logic blocks should not be input to earlier or later stages in the chain, and figure 4.17 illustrates some of the possibilities. It should be noted that the only requirement of such feedforward and feedback connections is that the set-up time of the register being clocked into is met. In effect, provided the interconnection time between any stage in the chain is not significant, then a signal which can be clocked into the succeeding stage on $\phi 1$ can be clocked into any other stage operating on $\phi 1$; similarly, data entered on $\phi 2$ can be clocked into any stage using $\phi 2$.

The logic blocks can be implemented using the random logic or transistor array techniques previously discussed. In particular, the combination of a logic block plus an input and output register can be very conveniently implemented using a PLA plus dynamic flip flops.

It can be seen from the PLA design shown in figure 4.6 that each input signal to the AND plane is buffered and that each output signal from the OR plane is inverted. Thus the addition of a minimum geometry pass transistor prior to each input buffer forms a dynamic inverting or non-inverting *D*-type flip flop. Similarly, a pass transistor placed between each OR plane output and its inverter forms the output register (see figure 4.18b). Thus not only can an input and output register be easily incorporated into the PLA design but their inclusion requires little additional silicon area.

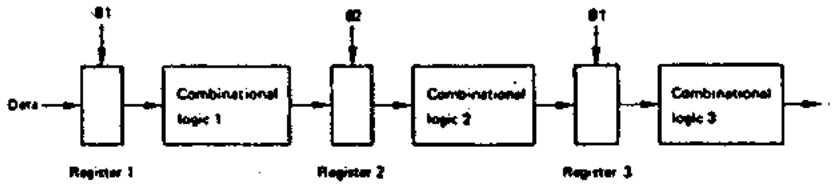


Figure 4.16 General schematic of a data pipeline

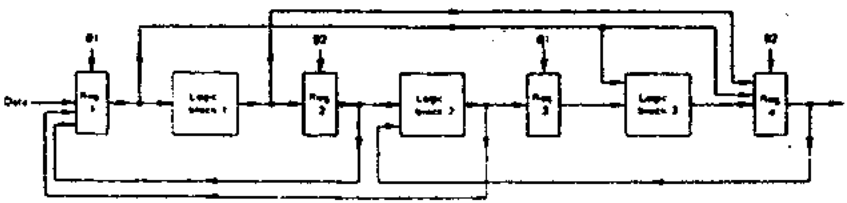


Figure 4.17 Examples of feedforward and feedback connections between pipeline stages

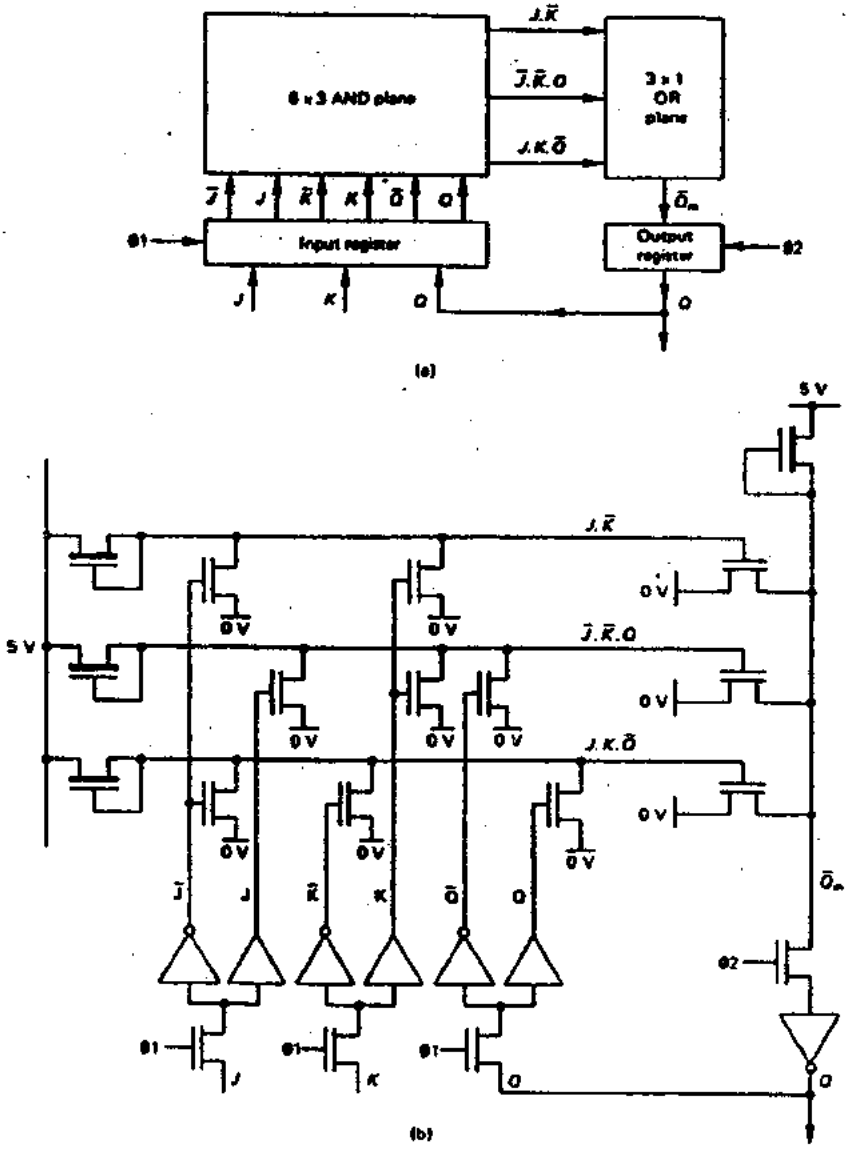


Figure 4.18 *J-K* flip flop: (a) logic diagram, (b) circuit diagram

A simple example illustrating the use of a feedback connection around a PLA with an input and output register is that of a *J-K* flip flop. In figure 4.18, *J* and *K* plus the output of the slave flip flop Q are clocked into the input register on $\phi 1$. Using inverting and non-inverting buffers, the true and inverse phase of the three input signals are formed. These six signals are operated upon by the AND plane which forms three product terms $\overline{J}.\overline{K}.Q$, $J.\overline{K}$ and $J.K.\overline{Q}$. The product terms are combined by the single nor gate in the OR plane. Thus the OR plane output $\overline{Q}_m$ is a logic '0' if *J* is '1' and *K* is '0', or if *J* and *K* are '0' with the slave output high, or if *J* and *K* are '1' and the slave output is low; all other input combinations on $\phi 1$ cause the OR plane output to be a '1'. On $\phi 2$, $\overline{Q}_m$ is clocked into the output register, yielding the slave (and flip flop) output Q.

Further reductions to the area and power required for a dynamic flip flop arise from the use of a ratioless inverter with a clocked load (see figure 4.19). Again the circuit comprises a pass transistor and an inverter. The area reduction results from the use of minimum geometry NMOS enhancement devices while the power reduction arises because power is only consumed when Clock is high; it is only during this time that the inverter load, T3, turns on, creating a current path between 5 V and 0 V.

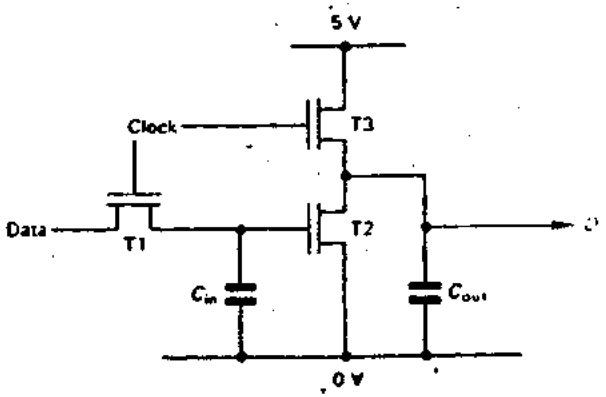


Figure 4.19 A clocked-load dynamic flip flop

When Clock becomes a '1', transistor T1 turns on and the input passes to the gate of T2. If Data is low, T2 is off. T3 is on and supplies T2's leakage current, causing $\overline{Q}$ to be high at $5 - V_{te}$. When Clock is removed, T1 and T3 turn off and T2 remains off. Thus the high level at $\overline{Q}$ is maintained. If, however, Data is high at 5 V when Clock is applied, then a level of $5 - V_{te}$ is transferred to the gate of T2. T2 and T3 are on and $\overline{Q}$ assumes a voltage which is not a valid logic level. Using the parameters previously adopted in examples and a high level input of 3.12 V on T2's gate, the level at $\overline{Q}$ can be calculated to be about 1.9 V. When

Clock becomes '0'. T1 and T3 turn off but T2 remains on and the capacitance at $\overline{Q}$, C_{out}, discharges to 0 V via T2. Clearly, the flip flop output is not valid until C_{out} has had time to discharge to a logic '0' level following the removal of Clock.

It should be noted that in the conventional dynamic flip flop of figure 4.14, the output voltage level is actively defined at all times by the inverter transistors T2 and T3. However, in the modified design of figure 4.19 only a low level on $\overline{Q}$ is actively defined (T2 on) when Clock is a '0'; the high level on $\overline{Q}$ is maintained solely by the charge on the output capacitance, since T2 and T3 are off when Clock is low.

This characteristic of the circuit is of importance if the output is passed to another dynamic circuit, since the charge stored will be shared between them. In figure 4.20, the alternate bits in the shift chain use $\phi 1$ and $\phi 2$. On $\phi 1$, flip flop 1 operates as previously described. Thus when $\phi 1$ is removed, $\overline{Q1}$ remains at $5 - V_{te}$ if Data was low and discharges to 0 V if Data was high. T4 is off and thus flip flop 2 is inactive during this time.

On $\phi 2$, the pass transistor T4 turns on. If $\overline{Q1}$ is low, then T2 is on and any voltage held on the input capacitance of T5 discharges to 0 V via T4 and T2. Thus T5 is off and T6 is on, causing $Q2$ to be $5 - V_{te}$; this level is maintained by the charge on C_{out} at $Q2$ when $\phi 2$ is taken low.

If $\overline{Q1}$ is high and the level stored on C_{in} of T5 is also high when $\phi 2$ is applied, then the level at T5's gate remains at $5 - V_{te}$. Here, T5 and T6 are on and $Q2$ assumes a non-standard voltage of 1.9 V. When $\phi 2$ is taken to '0', T5 remains on, discharging $Q2$ to 0 V. However, if $\overline{Q1}$ is high and the initial level on C_{in} of T5 is 0 V when $\phi 2$ is taken high, then charge is shared between C_{out} at $\overline{Q1}$ and C_{in} of T5. Here, the successful transfer of a high level to the gate of T5 is critically dependent upon the relative magnitude of C_{out} and C_{in}. Before T4 is turned on, the charge Q_{out} stored on C_{out} at $\overline{Q1}$ is

$$Q_{out} = C_{out}(V_p - V_{te})$$

When T4 turns on, this charge Q_{out} is shared between the two capacitors which are in parallel, and the new voltage level V_{out} across both capacitors is

$$V_{out} = \frac{Q_{out}}{C_{in} + C_{out}} = \frac{C_{out}(V_p - V_{te})}{C_{in} + C_{out}}$$

The new level of V_{out} is less than the high level voltage of $V_p - V_{te}$ held at $\overline{Q1}$ and must be greater than the threshold of T5, in order to discharge $Q2$ to 0 V when $\phi 2$ is removed. In fact, V_{out} has to significantly exceed V_{te} to turn on enough current in T5 (when $\phi 2$ is removed) to avoid incurring a long discharge time. Assuming $V_p - V_{te}$ is approximately 3 V and a minimum V_{out} of 2 V after the charge transfer, then $C_{out} > 2C_{in}$.

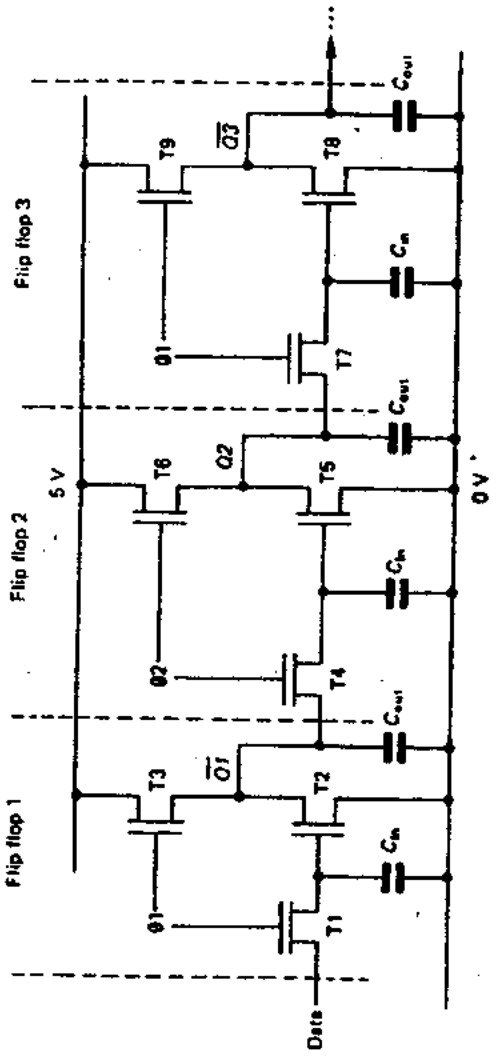


Figure 4.20 A clocked-load shift chain

4.7 Random Access Memory

A group of registers where only one register is accessed at a time and where the time to obtain or alter information in any register is similar is referred to as a Random Access Memory. The register or line to be read from or written to is specified by a unique binary address; thus a 2^n line store requires an n-bit address.

The address has to be decoded in order to select a particular register and 2^n n-input nand gates are necessary to provide a unique select signal for each register. For a store of any significant size, the amount of logic for the address decoding is prohibitive. As a result, the address is split into two parts usually referred to as the X and Y bits, and each part is separately decoded. Although this reduces the decoding logic to 2^X X-input plus 2^Y Y-input nand gates, where $X + Y = n$, there is only a total of $2^X + 2^Y$ select signals. It is therefore necessary to combine the signals obtained from the X and Y decoders in order to uniquely select one of 2^n registers.

Figure 4.21 shows a general schematic for reading from and writing to one bit of a store. The 2^n memory cells (one from each line) are organised as a two-dimensional matrix of 2^X rows and 2^Y columns.

If reading, the X decoder selects one row of cells and their outputs are enabled on to the Column Data Out lines; the outputs from all other cells are disabled. The Y decoder is used to select one of these data outputs which is then clocked into the output register. Thus, effectively, the intersection of an X and a Y select line determines the position of the selected memory cell within the matrix.

In some types of memory cell, reading the data from a cell causes it to be destroyed and it has therefore to be rewritten after reading. In this case, the Column Data Out line is connected back on to its Column Data In line via a coupling circuit. The X decoder is still selecting the same row of cells and the information on the Column Data In lines is written to these cells to restore their initial state.

Access to the memory is random and, if using dynamic elements as memory cells, it is necessary to refresh the data at periodic intervals to ensure that information is not lost. Refreshing is accomplished by using the X decoder to select a row of cells, reading out the data and then writing it back via the coupling circuits. It should be noted that a row of the matrix can be refreshed at a time and thus the entire memory can be restored in 2^X refresh operations. The refresh address to be presented to the X decoder is normally kept in a counter which is incremented by one every time the memory is accessed for refreshing.

Writing is accomplished by first reading from the row of cells selected by the X decoder. This data is now coupled back on to the Column Data In lines except that the input data to be written is superimposed on the Column Data In line selected by the Y decoder. Thus in the selected row, data is rewritten to all cells not selected by the Y decoder and new data is written to the selected cell.

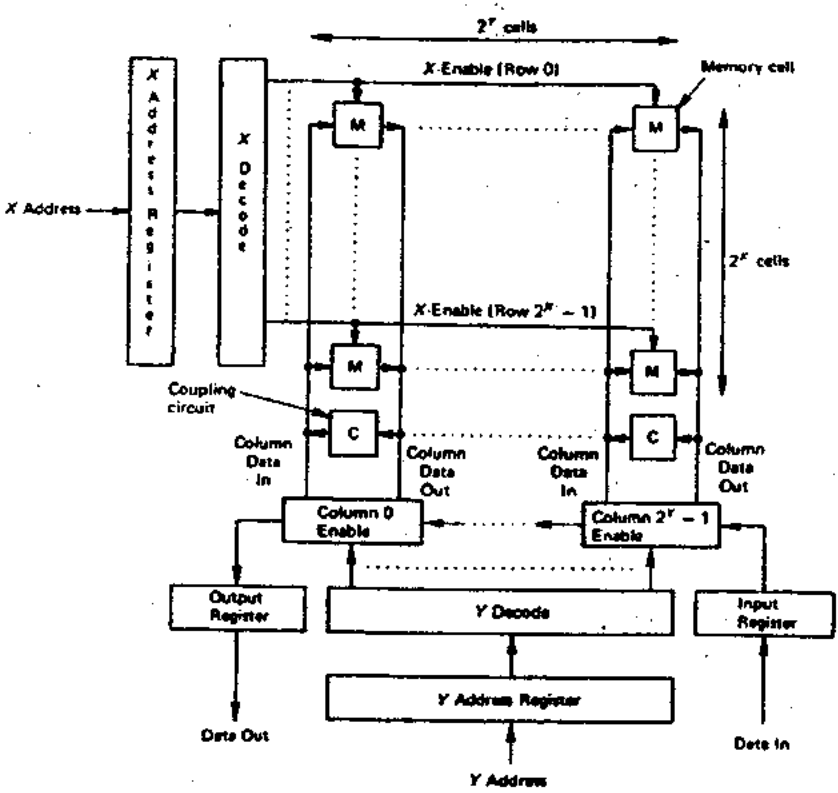


Figure 4.21 General schematic of a 2^n by one bit store

It is usual to include registers to hold the address and input and output data. This allows circuitry external to the memory to operate on the store's last output data and to calculate the next store address while the memory is operating on its current address.

A large number of signals are required to drive a store of significant size and it is often necessary to reduce them. Most often this reduction is effected by using the same lines for the X and Y address and by the use of common (bidirectional) data input/output lines. Sharing the address lines relies on the fact that a row of cells is first selected by the X decoder and then one of these cells is enabled for reading from or writing to by the Y decoder. Thus the X address is presented first and clocked into the X decode register. This allows the row decode and selection to operate while the Y address is presented and clocked into the Y decode register.

Figure 4.22 shows a three-transistor dynamic memory cell and its control signals. The state of the cell is stored on the capacitor C_s. To read its state, the X-Enable signal for the row is combined with the Read signal, turning T3 on. T4 is a clocked pull-up load for the Column Data Out line and is on at this time since Read is high. Thus the inverse of the voltage stored on the capacitor C_s appears on the Column Data Out line. The Y-Enable selects the column and turns on T9, allowing the data to be presented to the output register. Reading is not destructive, so there is no need to rewrite the data back after reading. Note that T3 isolates the memory cell from the Column Data Out line when the row is not selected.

Refreshing is indicated by a Refresh signal and performed by first reading data from a row and then rewriting its contents. During reading, T5 is on and thus data read out is transferred to the capacitor C_r. The Read signal is now removed (turning T3, T5 and T4 off) and Write is applied. The Refresh is used to prevent the Y decoder from selecting a column. Thus T9 and T10 are off and $\bar{Y}$-Enable is high, turning on T7 and the clocked load T8. This enables the inverse of the voltage stored on C_s to appear on the Column Data In line. Thus transistors T5, T6, T7 and T8 form the circuitry which couples the Column Data Out line to its Column Data In line. The Write signal combines with the X-Enable signal to turn on T1 for each cell in the selected row. Thus the voltage on each Column Data In line is passed via T1 to the storage capacitor C_s, restoring the data in these cells. It should be noted that establishing a voltage on C_s and C_r does not occur simultaneously, in order to avoid destroying the data held by the memory cell.

Writing is accomplished as a read operation followed by a write. Again after reading, the inverse of the data stored on C_s is stored on C_r. In the write phase, the Refresh signal is low and the Y decoders select a column turning its T10 on. T10 passes the information in the input register to the Column Data In line. T7 and T8 are off for the selected column since its $\bar{Y}$-Enable is low. The T1 transistors of the selected row are on, causing the write data to be transferred to C_s of the selected memory cell. As the Y-Enable for all unselected columns is low, the data read out of the cells in these columns is coupled back on to the Column Data In lines and rewritten during the write phase.

The ratioed design of figure 4.22 can be converted to a ratioless design by the addition of a pre-charge phase prior to a read, write or refresh operation. During the pre-charge phase, T4 and T8 are on (with their gates connected to the Pre-charge signal) and all other transistors are off. This causes all Column Data In and Column Data Out lines to be pre-charged high.

Operations now proceed in a similar manner to that previously described, except that the clocked loads, T4 and T8, are off. In the read phase, the inverse of the data stored on the C_s capacitors of the selected row are transferred to the Column Data Out lines and also transferred to the coupling circuit capacitance C_r. A high level on C_s causes the Column Data Out line to discharge to 0 V while a low level on C_s results in charge sharing between the Column Data Out

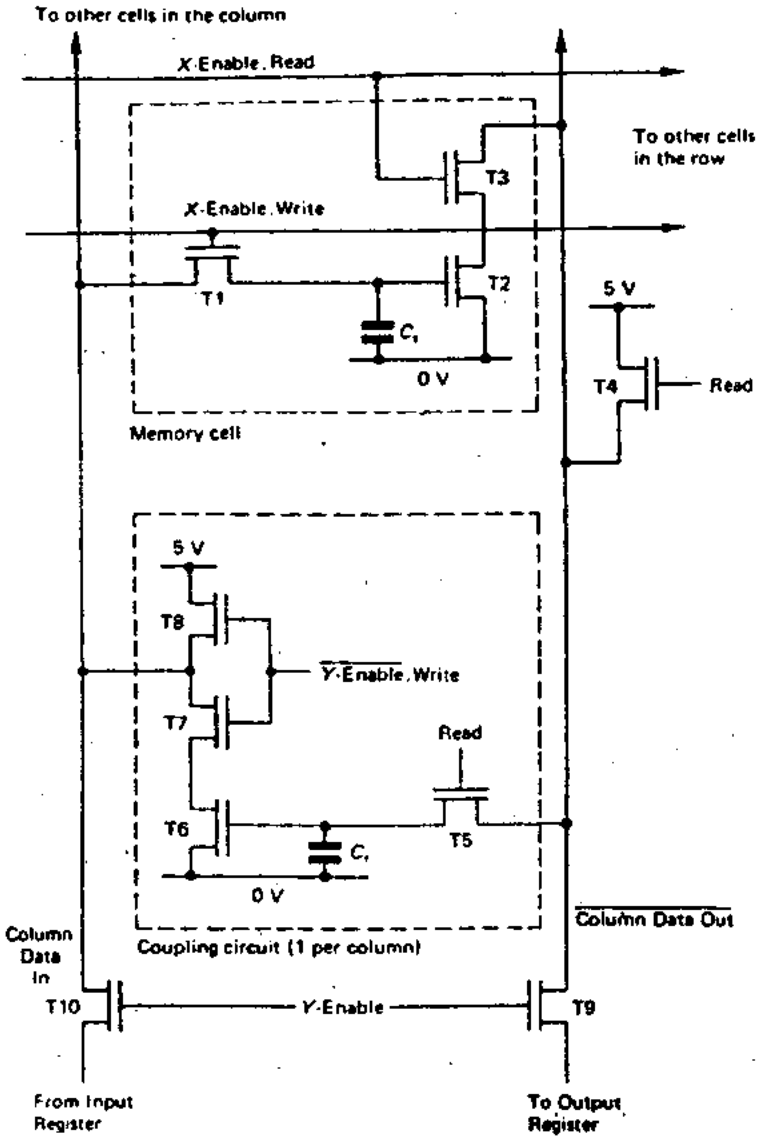


Figure 4.22 A three-transistor per bit memory cell and its control signals

line capacitance and C_r; in this latter case, a high level is transferred to C_r as the line capacitance is much greater than C_r.

The inverse of the voltage on C_r is transferred to the Column Data In lines and also to selected memory cells when rewriting. Again, a high voltage on C_r causes the Column Data In line to discharge. A low voltage on C_r results in a high level being transferred to the selected cell as a result of charge sharing between the Column Data In line and C_s.

A static memory cell is shown in figure 4.23. It consists of two cross-coupled inverters which form a flip flop, plus two pass transistors T5 and T6. If reading, the *X*-Enable signal is applied turning on T5 and T6. Thus the Q and $\bar{Q}$ flip flop outputs appear on the Column Data and Column Data lines respectively. The *Y*-Enable for the selected column turns on T7 and T8, so that the data can be clocked into the output register.

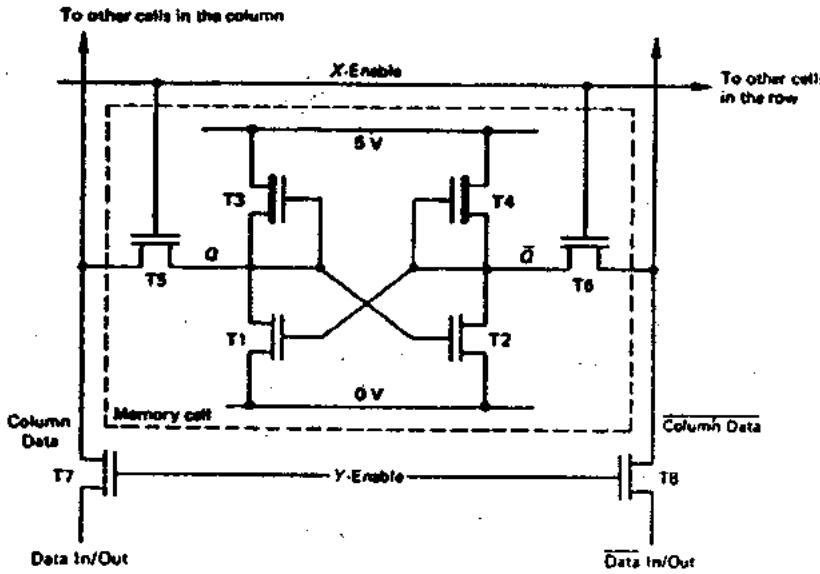


Figure 4.23 A six-transistor static memory cell

Reading is not destructive and it is not, of course, necessary to refresh the data in cells. Writing is performed by first reading out the data from cells in the selected row. Again the *Y*-Enable selects a column, turning its T7 and T8 on. This allows the write data and its inverse to be impressed upon the Column Data and Column Data lines. T5 and T6 are on and if the data to be written differs

from that stored by the cell, then the existing high output from the flip flop is pulled low by the incoming data. The cell feedback causes the other flip flop output to go high so that the cell contains the new data. In all unselected columns, the data read out of a cell is present on its Column Data and Column Data lines and it is this data which is rewritten, since T7 and T8 are off for these columns.

Large dynamic random access memories have storage cells consisting of a single transistor T and a (MOS) capacitor C_s, as shown in figure 4.24. Polysilicon forms the earth plate of the capacitor while the underlying semiconductor forms its other plate. In practice, it is usual for the capacitor's polysilicon to be formed on a different polysilicon layer to that of the transistor gate, as this leads to a smaller geometric layout.

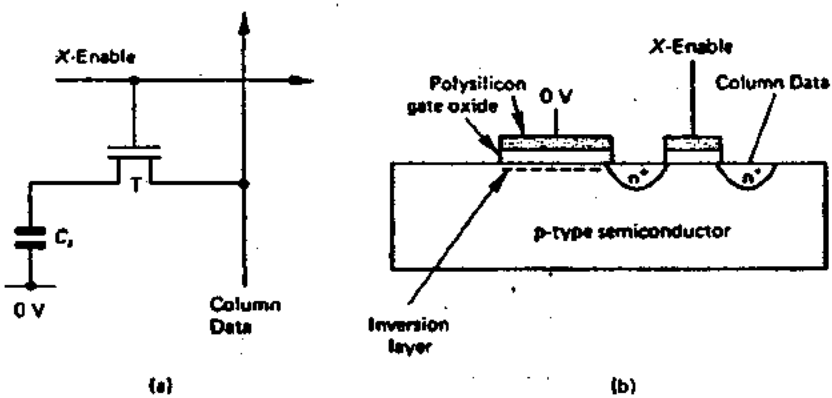


Figure 4.24 A one-transistor dynamic memory cell: (a) circuit, (b) structure

The data is held on the capacitor and is accessed via the pass transistor T. If writing, the data to be written is placed on the Column Data line (0 V or 5 V) and the *X*-Enable line is taken high. T turns on, resulting in the write data being stored on the capacitor.

If reading, again the *X*-Enable line is taken high. However, the Column Data line is not driven so that the capacitor C_s is connected to this line via T. The Column Data line has capacitance C_i which maintains the line's initial voltage of V_i. Turning T on in these circumstances causes charge sharing between C_s and C_i. This determines the final voltage attained by the Column Data line. If V_i lies between the two logic levels, then C_s stores a '1' if V_i rises and a '0' if V_i falls. However, since the Column Data line capacitance is far greater than the memory cell capacitance, charge sharing results in only a small change in the Column

Data line voltage. A further consequence is that the Column Data line voltage is transferred to C_s, destroying its stored data. These features cause additional complexity in the circuitry and timing necessary to sense and restore the data in a cell.

Figure 4.25 shows how a one-transistor per bit cell is incorporated into the memory. The sense amplifier consists of two NMOS inverters with enhancement loads (T1, T3 and T2, T4) and pass transistor T5; the inverters are cross-coupled to form a flip flop. Half the memory cells for a column are connected to one side of the flip flop while the other side is connected to the rest of a column's cells. Each flip flop output is also connected to a circuit referred to as a dummy cell. The dummy cell circuit is similar to that for a memory cell and has a pass transistor T6 in series with a capacitor C_d. In addition, the pass transistor T7 is used to establish a reference voltage across C_d. Thus the circuit is symmetrical and the load capacitance on each side of the flip flop, C_l, is identical.

Three phases of operation are necessary to read data from a cell. The first phase pre-charges the circuitry to pre-defined voltages. In this phase, $\phi 2$ is low, isolating all storage and dummy cell capacitors from the sense amplifier. $\phi 1$ and Pre-charge are applied, turning T3, T4 and T5 on. T5 connects Column Data and Column Data together and the current through T3 and T4 charges them to a voltage V_i. Simultaneous with this activity, the capacitor C_d in the dummy cell on the side opposite to that of the memory cell to be selected is charged to a voltage V_{ref} via T7; V_{ref} is usually halfway between the logic '0' and '1' voltage levels.

The data is sensed in the second phase. $\phi 1$ and Pre-charge are removed. $\phi 2$ is taken high and combines with the X decode signal to turn on the transistor, T, in the memory cells of the selected row. On the other side of the sense amplifier, $\phi 2$ is used to turn on transistor T6 of the dummy cell. Thus charge sharing now occurs between C_s and C_l on one side of the amplifier and between C_d and C_l on the other side.

Referring to figure 4.26 which shows the charge stored and the voltage levels established during the pre-charge phase, assuming that the selected cell is on the Column Data side

$$Q_s = V_s C_s$$

$$Q_l = V_l C_l$$

and

$$Q_d = V_{ref} C_d$$

Charge sharing in the sense phase between C_s and C_l results in a voltage $V_{\overline{data}}$ on the Column Data line of

$$V_{\overline{data}} = \frac{\text{total charge}}{\text{total capacitance}} = \frac{Q_s + Q_l}{C_s + C_l}$$

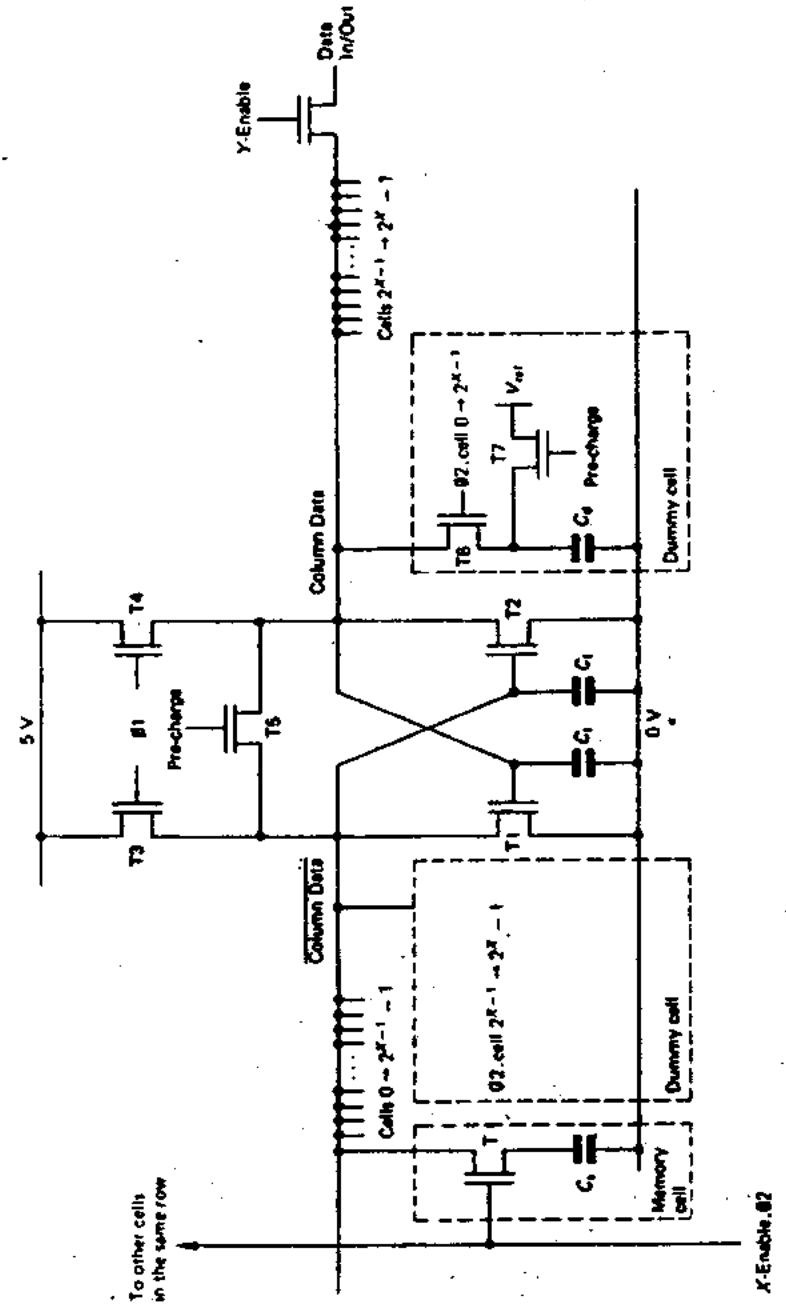


Figure 4.25 Sensing and restoring data to a one-transistor per bit memory cell

Substituting into this expression for Q_s and Q_i yields

$$V_{\overline{data}} = \frac{V_s C_s + V_i C_i}{C_s + C_i}$$

Similarly, charge sharing between C_d and C_i gives

$$V_{data} = \frac{Q_d + Q_i}{C_d + C_i} = \frac{V_{ref} C_d + V_i C_i}{C_d + C_i}$$

on the Column Data line. Further simplification is possible by letting C_s equal C_d and now

$$V_{data} = \frac{V_{ref} C_s + V_i C_i}{C_s + C_i}$$

The difference between V_{data} and $V_{\overline{data}}$ in the sense phase is therefore determined by the difference between V_s and V_{ref}. In the case illustrated, a low voltage of 0 V stored on a selected memory cell results in the dummy cell establishing a higher voltage on Column Data than Column Data, while a high memory cell voltage, V_s, greater than V_{ref} causes $V_{\overline{data}}$ to be greater than V_{data}.

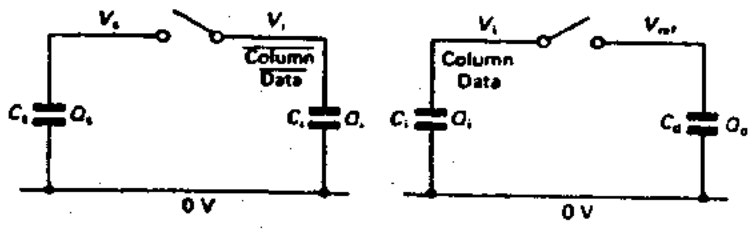


Figure 4.26 Charge and voltages established during pre-charge in a one-transistor per bit memory

Having established a voltage difference between the sense amplifier outputs, the third phase staticises the sensed outputs, restoring Column Data and Column Data to logic levels and restores the voltage in cells of the selected row. Also during this phase, data from the column selected by the Y decode circuitry is clocked into the output register.

In phase 3, $\phi 2$ remains high so that memory cells in the selected row remain enabled. $\phi 1$ is also taken high to provide a load for the inverters of the flip flop. The feedback between the inverters operates to reinforce the (small) voltage

difference between the two sides of the flip flop and switching continues until one inverter is off and the other on. This results in logic levels appearing on Column Data and Column Data and, since the memory cell is still selected, the charge is restored on the storage capacitor C_s at this time.

Writing information to a cell follows a similar sequence of events except that, during the sense phase, the data to be written is placed on the selected Column Data line to prime the flip flop. In the staticise and restore phase, $\phi 1$ is high to activate the flip flop and the inverse of the write data appears on Column Data. The pass transistor of the selected memory cell is on. Write data is written to a selected cell on the Column Data side while its inverse is stored in a selected cell on the Column Data side. Note that the inversion of data written to the Column Data side does not matter, as inversion occurs again at readback.

The size of a commercial one-transistor per cell dynamic random access memory is normally 2^n by 1 bits, and the technology has improved to the point where 2^{20} bits are anticipated.

4.8 Further Reading

D. J. Kinniment and J. V. Woods, 'Synchronisation and arbitration circuits in digital systems', *Proc. IEE*, 123, No. 10 (1976) pp. 961-96.

C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

K. U. Stein, A. Sihling and E. Doering, 'Storage array and sense/refresh circuit for single-transistor memory cells', *IEEE Journal of Solid-State Circuits*, SC-7, No. 5 (1972) pp. 336-40.