

FUTURE VISION BIE

**One Stop for All Study Materials
& Lab Programs**



Future Vision

By K B Hemanth Raj

Scan the QR Code to Visit the Web Page



Or

Visit : <https://hemanthrajhemu.github.io>

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

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

Contact: MAIL: futurevisionbie@gmail.com

INSTAGRAM: www.instagram.com/hemanthraj_hemu/

INSTAGRAM: www.instagram.com/futurevisionbie/

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

Third Edition

Eastern
Economy
Edition

Basic VLSI Design



Douglas A. Pucknell
Kamran Eshraghian



<https://hemanthrajhemu.github.io>

5.3	Some Discussion on and Limitations of Scaling	119
5.3.1	Substrate Doping	119
5.3.2	Limits of Miniaturization	121
5.3.3	Limits of Interconnect and Contact Resistance	123
5.4	Limits Due to Subthreshold Currents	126
5.5	Limits on Logic Levels and Supply Voltage Due to Noise	128
5.6	Limits Due to Current Density	132
5.7	Observations	132
5.8	References	133

Chapter 6 Subsystem Design and Layout**134–179***Objectives* 134

6.1	Some Architectural Issues	134
6.2	Switch Logic	135
6.2.1	Pass Transistors and Transmission Gates	136
6.3	Gate (restoring) Logic	137
6.3.1	The Inverter	137
6.3.2	Two-input nMOS, CMOS and BiCMOS <i>Nand</i> Gates	138
6.3.3	Two-input nMOS, CMOS and BiCMOS <i>Nor</i> Gates	143
6.3.4	Other Forms of CMOS Logic	145
6.4	Examples of Structured Design (Combinational Logic)	151
6.4.1	A Parity Generator	151
6.4.2	Bus Arbitration Logic for n-line Bus	153
6.4.3	Multiplexers (Data Selectors)	157
6.4.4	A General Logic Function Block	159
6.4.5	A Four-line Gray Code to Binary Code Converter	160
6.4.6	The Programmable Logic Array (PLA)	162
6.5	Some Clocked Sequential Circuits	162
6.5.1	Two-phase Clocking	162
6.5.2	Charge Storage	166
6.5.3	Dynamic Register Element	168
6.5.4	A Dynamic Shift Register	169
6.6	Other System Considerations	170
6.6.1	Bipolar Drivers for Bus Lines	170
6.6.2	Basic Arrangements for Bus Lines	170
6.6.3	The Precharged Bus Concept	172
6.6.4	Power Dissipation for CMOS and BiCMOS Circuits	173
6.6.5	Current Limitations for V_{DD} and GND (V_{SS}) Rails	174
6.6.6	Further Aspects of V_{DD} and V_{SS} Rail Distribution	175
6.7	Observations	177
6.8	Tutorial Exercises	178

Subsystem Design and Layout

Logic is simply the architecture of human reason.

— EVELYN WAUGH

Tall oaks from little acorns grow.

— DAVID EVERETT

OBJECTIVES

Having now covered the basic MOS and BiCMOS technologies, the behavior of components formed by MOS layers, the basic units which help to characterize behavior, and a set of design rules, we are in a position to undertake the design of some of the subsystems (leaf-cells) from which larger systems are composed.

The most basic leaf-cells are the common logic gate arrangements and these are dealt with in nMOS, CMOS and BiCMOS forms. In the case of CMOS gates there are several ways in which the logic may be configured, most of which are dealt with in this chapter.

The concepts of structured design, which leads to system designs of high ‘regularity’, are introduced through examples. A highly regular design is to be sought for VLSI systems since high regularity implies the detailed design of relatively few leaf-cells which are then replicated many times and interconnected to form the system.

Other commonly applied concepts, such as two-phase clocks and buses for the interconnection paths between leaf-cells and subsystems, are also introduced and illustrated. Important aspects of power distribution on chip and associated limitations are discussed.

The chapter also includes an introduction to common subsystem designs such as multiplexers and shift registers.

6.1 SOME ARCHITECTURAL ISSUES

In all design processes, a logical and systematic approach is essential. This is particularly so in the case of the design of a VLSI system which could otherwise take so long as to render

the whole system obsolete before it is off the drawing board. Take, for example, the case of a relatively straightforward MSI logic circuit comprising, say, 500 transistors. A reasonable time to allocate to the design and proving of such a circuit could be some two engineer-months. Consider now the design of a 500,000 transistor VLSI system. Even if a linear relationship exists between complexity and design time, the required design time would be 2000 engineer-months or 170 engineer-years. In fact, design time tends to rise exponentially with increased complexity. Obviously, then, we must adopt design methods which allow the handling of complexity in reasonable periods of time and with reasonable amounts of labor.

Certainly we are not about to tackle 500,000 transistor designs in this text, but some sensible concepts applied even at the subsystem (leaf-cell) level can be most worthwhile and can also be directly compatible with larger system design requirements. Guidelines may be set out as follows:

1. Define the requirements (properly and carefully).
2. Partition the overall architecture into appropriate subsystems.
3. Consider communication paths carefully in order to develop sensible interrelationships between subsystems.
4. Draw a floor plan of how the system is to map onto the silicon (and alternate between 2, 3 and 4 as necessary).
5. Aim for regular structures so that design is largely a matter of replication.
6. Draw suitable (stick or symbolic) diagrams of the leaf-cells of the subsystems.
7. Convert each cell to a layout.
8. Carefully and thoroughly carry out a design rule check on each cell.
9. Simulate the performance of each cell/subsystem.

The whole design process will be greatly assisted if considerable care is taken with:

1. the *partitioning* of the system so that there are clean and clear subsystems with a minimum interdependence and complexity of interconnection between them.
2. the *design simplification* within subsystems so that architectures are adopted which allow the exploitation of a cellular design concept. This allows the system to be composed of relatively few standard cells which are replicated to form highly regular structures.

In designing digital systems in MOS technology there are two basic ways of building logic circuits, which will now be discussed.

6.2 SWITCH LOGIC

Switch logic is based on the 'pass transistor' or on transmission gates. This approach is fast for small arrays and takes no static current from the supply rails. Thus, power dissipation of such arrays is small since current only flows on switching.

Switch (pass transistor) logic is similar to logic arrays based on relay contacts in that the path through each switch is isolated from the signal activating the switch. In consequence, the designer has a considerable amount of freedom in implementing architectural features compared with bipolar logic-based designs.

A number of texts on switching theory, some dating from the 1950s and 1960s, have sections on relay/switch logic and the reader is referred to such material for generating ideas for implementation in MOS switch logic. An example is Marcus, *Switching Circuits for Engineers*, Prentice Hall, 1962; 3rd edn, 1975.

Basic *And* and *Or* connections are set out in Figure 6.1, but many combinations of switches are possible.

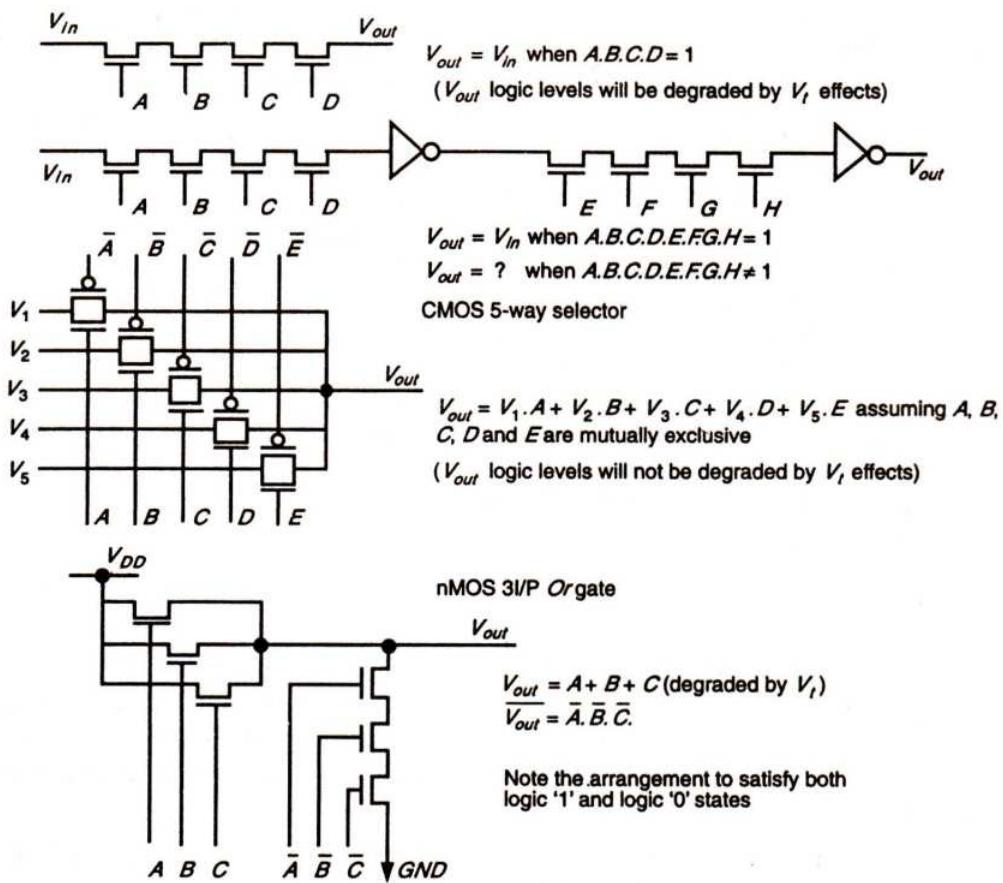


FIGURE 6.1 Some switch logic arrangements.

6.2.1 Pass Transistors and Transmission Gates

Switches and switch logic may be formed from simple n- or p-pass transistors or from transmission gates (complementary switches) comprising an n-pass and a p-pass transistor in parallel as shown in Figure 6.2. The reason for adopting the apparent complexity of the transmission gate, rather than using a simple n-switch or p-switch in most CMOS applications, is to eliminate the undesirable threshold voltage effects which give rise to the loss of logic levels in pass transistors as indicated in Figure 6.2. No such degradation occurs with the

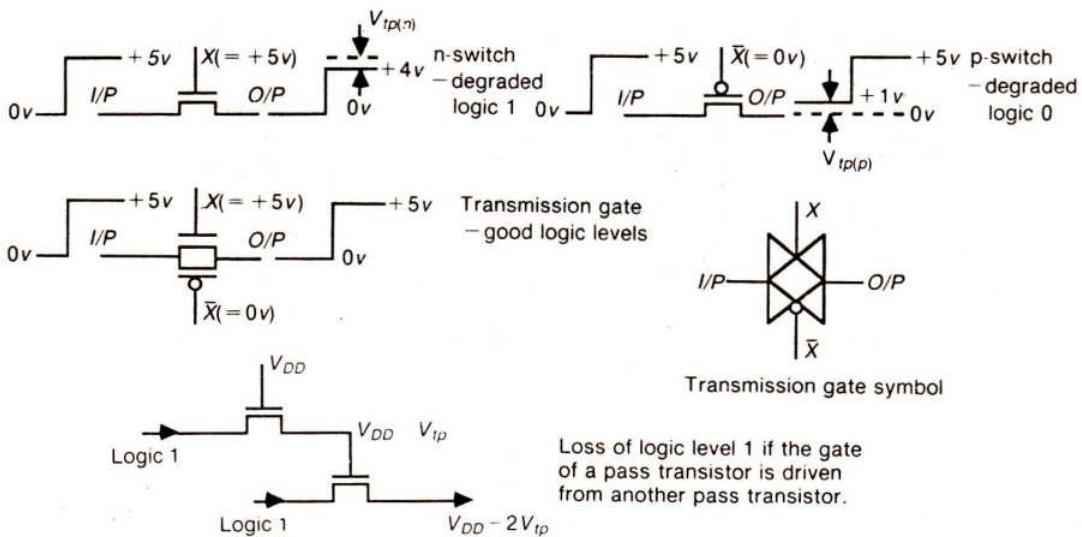


FIGURE 6.2 Some properties of pass transistors and transmission gates.

transmission gate, but more area is occupied and complementary signals are needed to drive it. ‘On’ resistance, however, is lower than that of the simple pass transistor switches.

When using nMOS switch logic, there is one restriction which must always be observed: *no* pass transistor gate input may be driven through *one or more pass transistors* (see Figure 6.2). As shown, logic levels propagated through pass transistors are degraded by threshold voltage effects. Since the signal out of pass transistor T_1 does not reach a full logic 1, but rather a voltage one transistor threshold below a true logic 1, this degraded voltage would not permit the output of T_2 to reach an acceptable logic 1 level.

6.3 GATE (restoring) LOGIC

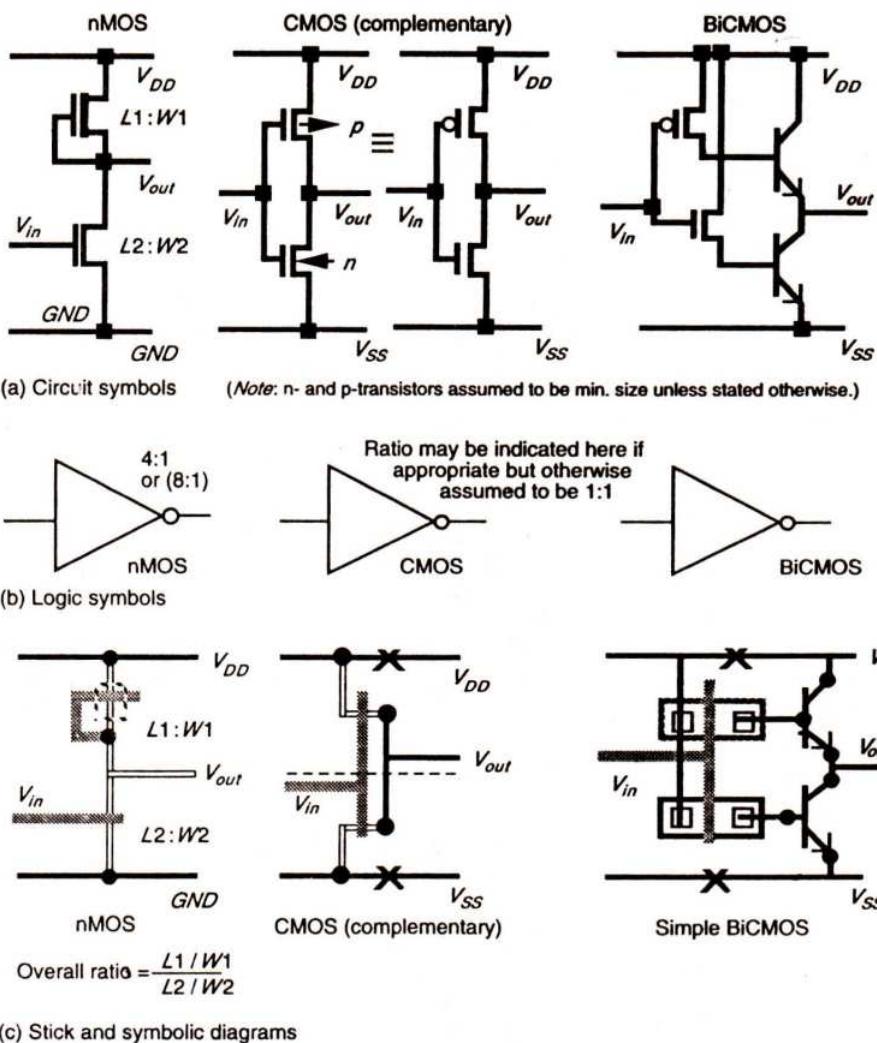
Gate logic is based on the general arrangement typified by the inverter circuits (the inverter being the simplest gate).

Both *Nand* and *Nor* and, with CMOS, *And* and *Or* gate arrangements are available. Inverters are also employed to complement and restore logic levels that have been degraded (e.g. because they have passed through pass transistors).

6.3.1 The Inverter

Some of the most commonly used inverter circuit diagrams—the inverter symbol, and the corresponding stick and symbolic diagrams—should be familiar by now. An assortment is reproduced here in Figure 6.3. Note that it is often useful to indicate the nMOS inverter $Z_{p.u.}/Z_{p.d.}$ ratio and/or the channel length to width ratio for each MOS transistor as shown.

In achieving the desired pull-up to pull-down ratio, several possibilities emerge, two of which are illustrated in Figures 6.4 and 6.5 for an 8:1 nMOS inverter. Note the effect that



FIGURES 6.3 nMOS, CMOS and BiCMOS inverters.

the different approaches have on power dissipation P_d and on the area occupied by the inverter. Also note the resistance and capacitance values. The CMOS inverter carries no static current and thus has no power dissipation unless switching. The reader must not, however, imagine that CMOS circuits have no dissipation problems. The switching dissipation for fast CMOS logic circuits will be considerable.

6.3.2 Two-input nMOS, CMOS and BiCMOS Nand Gates

Two-input Nand gate arrangements are given in Figure 6.6. The nMOS (and pseudo-nMOS) $L:W$ ratios should be carefully noted since they must be chosen to achieve the desired overall $Z_{p.u.}/Z_{p.d.}$ ratio (where $Z_{p.d.}$ is contributed in this case by *both* input transistors in series).

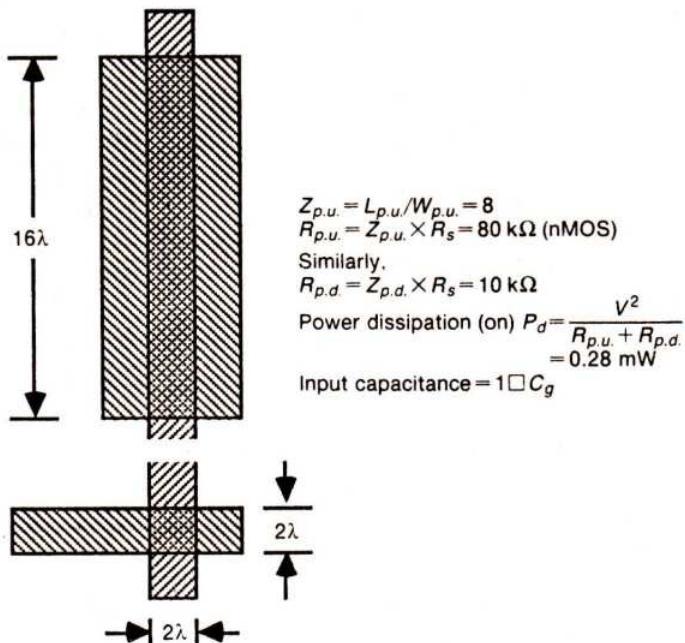
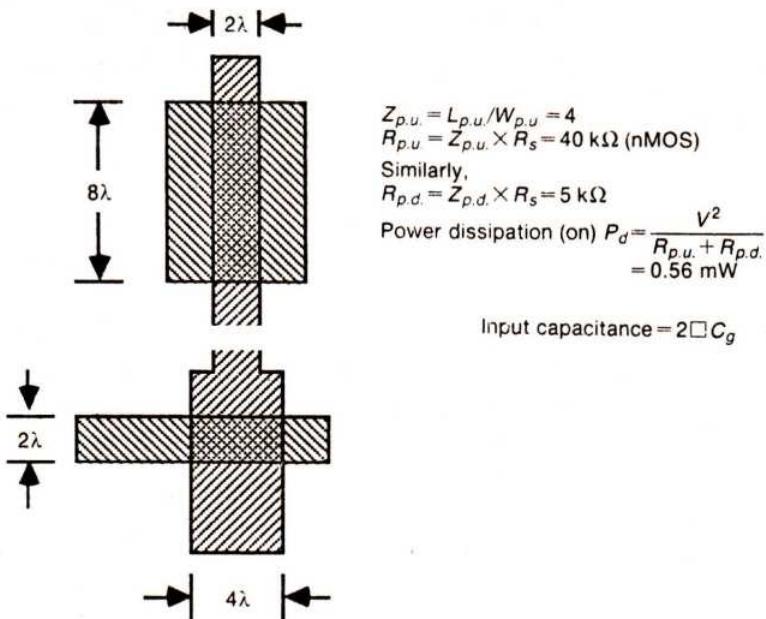
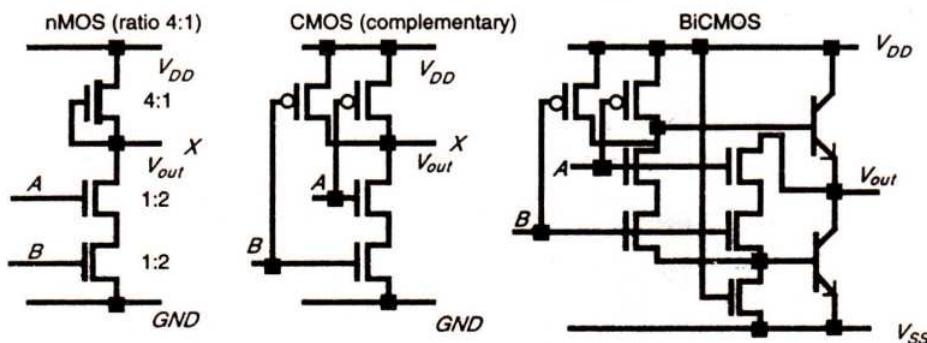


FIGURE 6.4 8:1 nMOS inverter (minimum size p.d.)



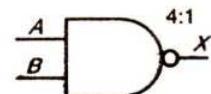
Note: A 4:1 inverter is formed if the p.d. width is halved.

FIGURE 6.5 An alternative 8:1 nMOS inverter.

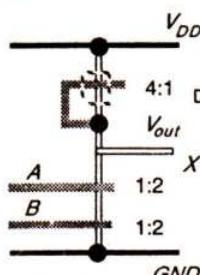
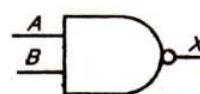


(a) Circuit diagrams

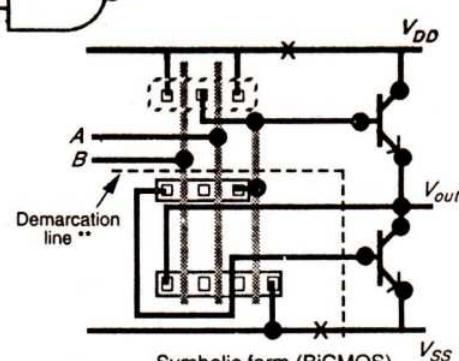
Note: n- and p- transistors assumed to be minimum size unless stated otherwise.



(b) Logic symbols



(c) Stick diagrams (nMOS and CMOS)



Note: The natural 2.5:1 asymmetry of the CMOS inverter is improved to 1.25:1 (or better) owing to the two n-type pull-down transistors in series for the two I/P Nand.

** Demarcation line (edge of n-well) may be shown if required.

FIGURE 6.6(a)–(c) nMOS, CMOS and BiCMOS 2-input Nand gates.

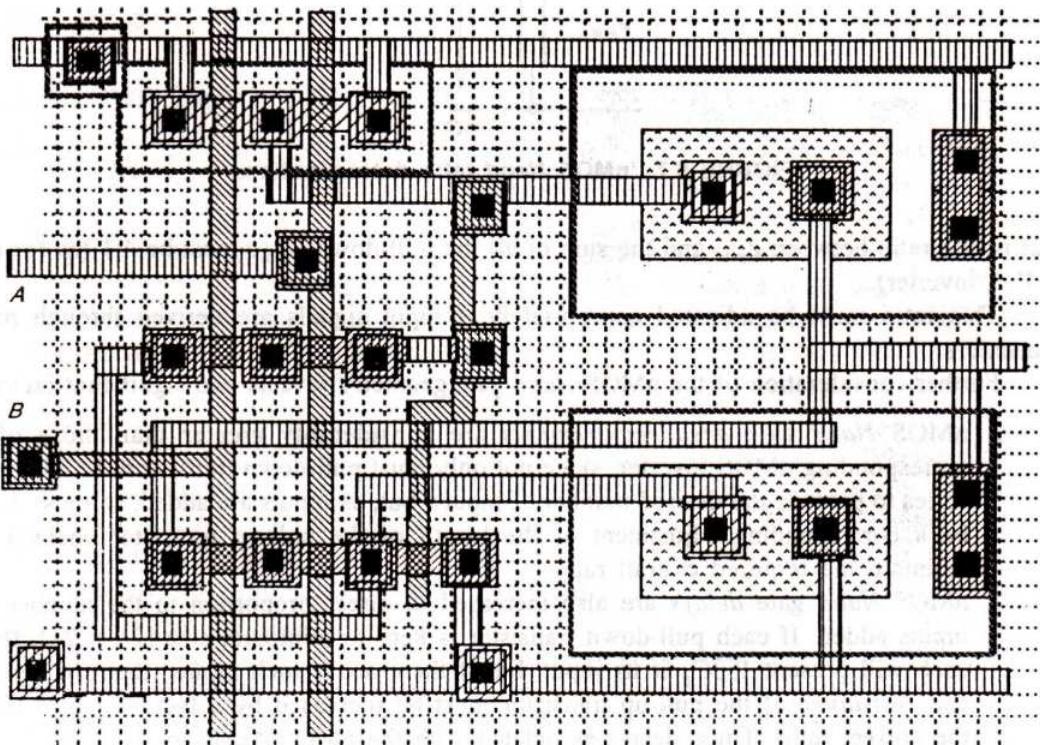
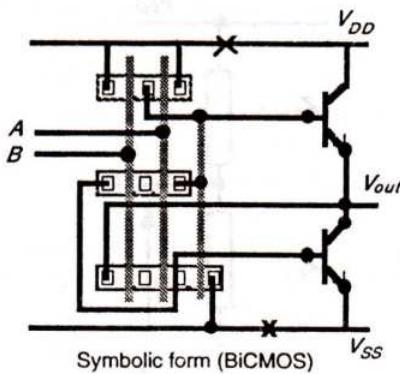
In order to arrive at the required $L:W$ ratios for an nMOS (or pseudo-nMOS) Nand gate with n inputs, it is only necessary to consider the very simple circuit model of the gate in the condition when all n pull-down transistors are conducting as in Figure 6.7.

The critical factor here is that the output voltage V_{out} must be near enough to ground to turn off any following inverter-like stages, that is

$$V_{out} \leq V_t = 0.2V_{DD}$$

Thus

$$\frac{V_{DD} \times nZ_{p.d.}}{nZ_{p.d.} + Z_{p.u.}} \leq 0.2V_{DD}$$



FIGURES 6.6(d) A BiCMOS two-input Nand gate.

where $Z_{p.d.}$ applies for any one pull-down transistor. The boundary condition then is

$$\frac{nZ_{p.d.}}{nZ_{p.d.} + Z_{p.u.}} = 0.2$$

whence nMOS Nand ratio $= \frac{nZ_{p.d.}}{nZ_{p.d.}} = \frac{4}{1}$

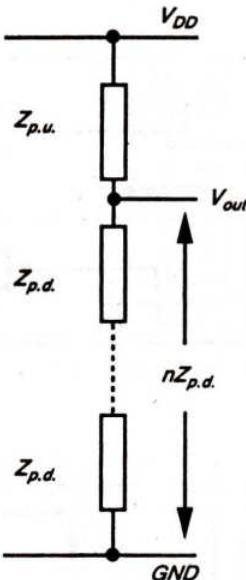


FIGURE 6.7 nMOS Nand ratio determination.

that is, the ratio between $Z_{p.u}$ and the sum of all the pull-down $Z_{p.d}$'s must be 4:1 (as for the nMOS inverter).

This ratio must be adjusted appropriately if input signals are derived through pass transistors.

Further consideration of the nMOS Nand gate geometry reveals two significant factors:

1. nMOS Nand gate *area requirements* are considerably greater than those of a corresponding nMOS inverter, since not only must pull-down transistors be added in series to provide the desired number of inputs, but, as inputs are added, so must there be a corresponding adjustment of the length of the pull-up transistor channel to maintain the required overall ratio.
2. nMOS Nand gate *delays* are also increased in direct proportion to the number of inputs added. If each pull-down transistor is kept to minimum size ($2\lambda \times 2\lambda$), then each will present $1/\square C_g$ at its input, but if there are n such inputs, then the length and *resistance* of the pull-up transistor must be increased by a factor of n to keep the correct ratio. Thus, delays associated with the nMOS Nand are

$$\tau_{Nand} = n\tau_{inv}$$

where n is the number of inputs and τ_{inv} is the corresponding nMOS inverter delay. (The alternative approach of keeping $Z_{p.u}$ constant and widening the pull-down channels has the same effect, since in this case C_g for each pull-down transistor will be increased to $n/\square C_g$).

Furthermore, the rise time of the nMOS Nand output is dependent on the actual input(s) on which the ∇ transition takes place.

In consequence of these properties, the nMOS *Nand* gate is used *only* where absolutely necessary and the number of inputs is restricted.

The CMOS *Nand* gate has no such restrictions but, bearing in mind the remarks on asymmetry (Figure 6.6), it is necessary to allow for extended fall-times on capacitive loads owing to the number of n-transistors in series forming the pull-down. Some adjustment of transistor geometry may be necessary for this reason and to keep the transfer characteristic symmetrical about $V_{DD}/2$.

The BiCMOS gate shown is a practical version and is thus more complex than the simple intuitive version. However, it has considerable load-driving capabilities and is most useful where a large fan-out is required or where there is some other form of high capacitance load on the output. A typical mask layout for this gate, using OrbitTM 2 μm design rules, is given in monochrome form in Figure 6.6(d) and in color as Color plate 8(a).

The relatively easy conversion from symbolic form to mask layout for the BiCMOS 2-input *Nor* gate is illustrated by Figure 6.6(d) and Color plate 8(a).

6.3.3 Two-input nMOS, CMOS and BiCMOS *Nor* Gates

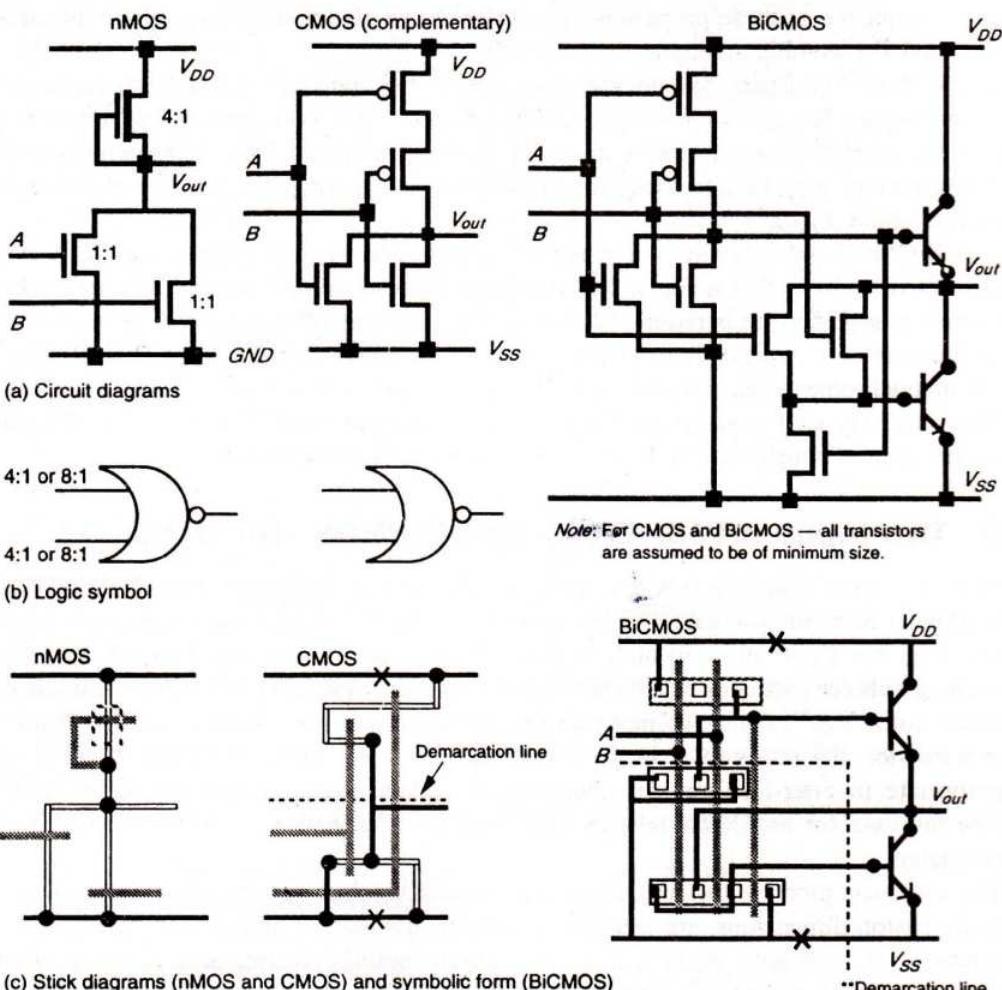
Two-input *Nor* gate arrangements are given in Figure 6.8; note here that the nMOS (or pseudo nMOS) form of *Nor* gate can be expanded to accommodate any reasonable number of inputs (e.g. see Color plate 9) and, in those technologies, is preferred to the *Nand* gate when there is a choice (which is usually the case if logical expressions are suitably manipulated).

Since both ‘legs’ of the two-input nMOS *Nor* gate provide a path to ground from the pull-up transistor, the ratios must be such that any one conducting pull-down leg will give the appropriate inverter-like transfer characteristic. Thus, each leg has the same ratio as would be the case for an nMOS inverter. This applies irrespective of the number of inputs accommodated.

The area occupied by the nMOS (or pseudo-nMOS) *Nor* gate is reasonable since the pull-up transistor dimensions are unaffected by the number of inputs accommodated. In consequence, the *Nor* gate is as fast as the corresponding inverter and is the preferred inverter-based nMOS (or pseudo-nMOS) logic gate when a choice is possible.

Obviously, the ratio between $Z_{p.u.}$ and $Z_{p.d.}$ of any one leg must be appropriate to the source from which that input is driven for nMOS designs (namely 4:1 driven from another inverter-based circuit or 8:1 if driven via one or more pass transistors) but will be uniformly 3:1 for a pseudo-nMOS design where any series switching is by transmission gate.

The CMOS *Nor* gate (see Color plate 9) consists of a pull-up p-transistor-based structure, which implements the logic 1 conditions and a complementary n-transistor arrangement to implement the logic 0 conditions at the output. In the case of the *Nor* gate, the p-structure consists of transistors in series, one for each input, while the n pull-down arrangement has as many transistors in parallel as there are inputs to the *Nor* gate. Thus, the already predominant resistance of the p-devices is aggravated in its effect by the number connected in series. Rise-time and fall-time asymmetry on capacitive loads is thus increased and there will also be a shift in the transfer (V_{in} vs V_{out}) characteristic which will reduce noise immunity. For these reasons, CMOS (complementary logic) *Nor* gates with more than two inputs may require adjustment of the p- and/or n-transistor geometries ($L:W$ ratios).



**Demarcation line (edge of n-well) may be shown if required.

FIGURES 6.8(a)–(c) nMOS, CMOS and BiCMOS two-input Nor gate.

The CMOS *Nand* gate, on the other hand, benefits from the connection of p-transistors in parallel, but once again the geometries may require thought when several inputs are required.

The BiCMOS *Nor* gate shown is a practical version and, as for the BiCMOS *Nand*, is more complex than a simple intuitive version. However, it also has considerable capacitive load-driving capabilities and is most useful where a large fan-out is required or where there is some other form of high capacitance load on the output such as in the I/O region of a chip.

The relatively easy conversion from symbolic form to mask layout is illustrated for the BiCMOS two-input *Nor* gate in Figure 6.8(d) and Color plate 8(b). The mask layout has been drawn using the OrbitTM 2 μm BiCMOS rule set.

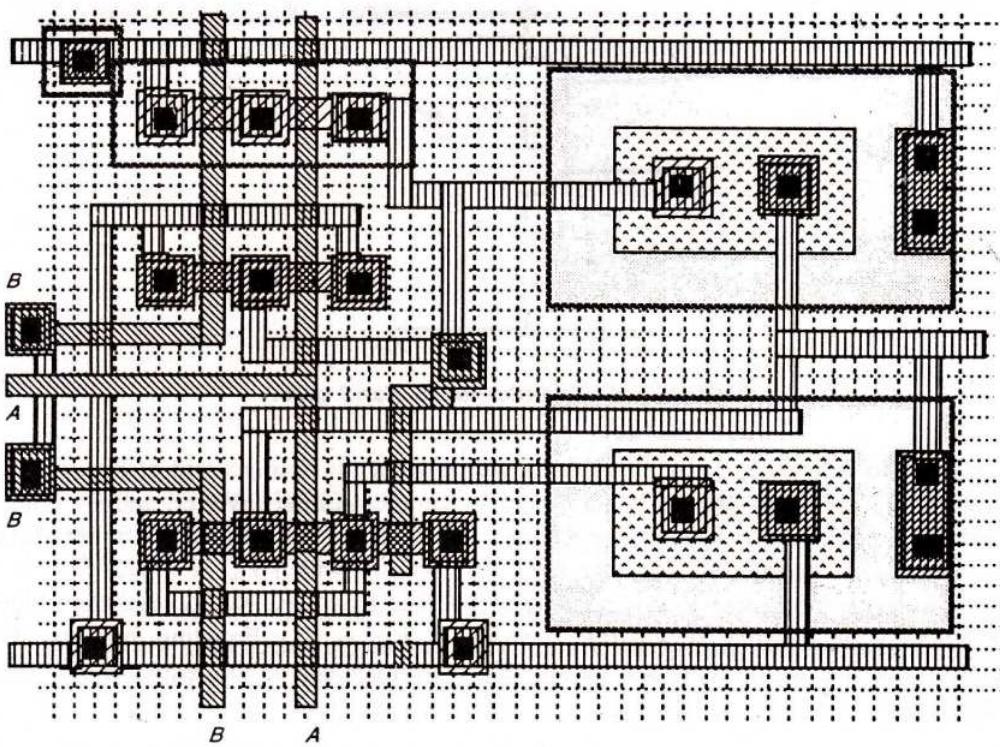
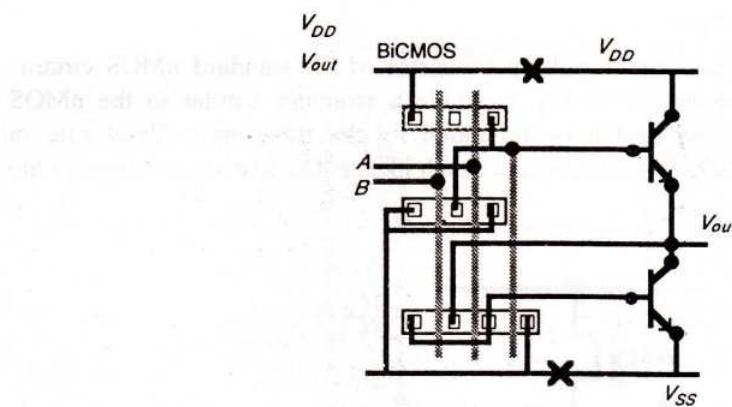


FIGURE 6.8(d) A BiCMOS two-input *Nor* gate.

6.3.4 Other Forms of CMOS Logic

The availability of both n- and p-transistors makes it possible for the CMOS designer to explore and exploit various alternatives to inverter-based CMOS logic.

6.3.4.1 Pseudo-nMOS logic

Clearly, if we replace the depletion mode pull-up transistor of the standard nMOS circuits with a p-transistor with gate connected to V_{SS} , we have a structure similar to the nMOS equivalent. This approach to logic design is illustrated by the three-input *Nand* gate in Figure 6.9. The circuit arrangements look and behave much like nMOS circuits and appropriate ratio rules must be applied.

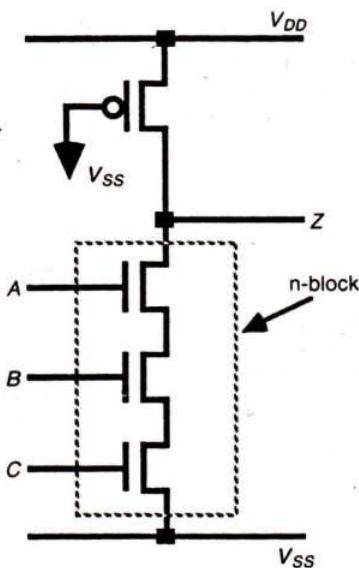


FIGURE 6.9 Pseudo-nMOS *Nand* gate.

In order to determine the required ratio, we consider the arrangement of Figure 6.10 in which a pseudo-nMOS inverter is being driven by another similar inverter, and we consider the conditions necessary to produce an output voltage of V_{inv} for an identical input voltage. As for the nMOS analysis, we consider the conditions for which $V_{inv} = V_{DD}/2$.

At this point the n-device is in saturation (i.e. $0 < V_{gsn} - V_{tn} < V_{dsn}$) and the p-device is operating in the resistive region (i.e. $0 < V_{dsp} < V_{gsp} - V_{tp}$). Equating currents of the n-transistor and the p-transistor, and by suitable rearrangement of the resultant expression, we obtain

$$V_{inv} = V_{tn} + \frac{(2\mu_p/\mu_n)^{1/2} [(-V_{DD} - V_{tn})V_{dsp} - V_{dsp}^2]^{1/2}}{(Z_{p.u.}/Z_{p.d.})^{1/2}}$$

where

$$Z_{p.u.} = L_p/W_p$$

and

$$Z_{p.d.} = L_n/W_n$$

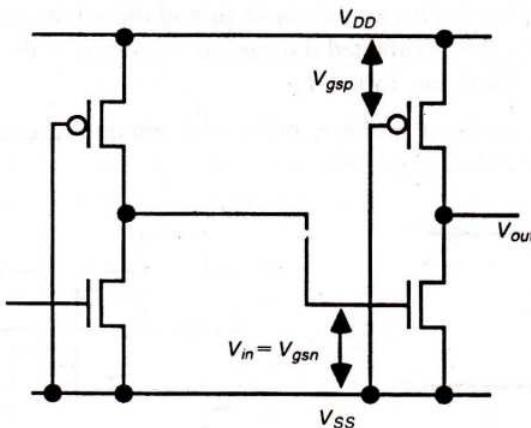


FIGURE 6.10 Pseudo-nMOS Inverter when driven from a similar Inverter.

With

$$V_{inv} = 0.5V_{DD}$$

$$V_{tn} = |V_{tp}| = 0.2V_{DD}$$

$$V_{DD} = 5 \text{ V}$$

$$\mu_n = 2.5 \mu_p$$

we obtain

$$\frac{Z_{p.u.}}{Z_{p.d.}} = \frac{3}{1}$$

A transfer characteristic, V_{out} vs V_{in} , can be drawn and, as for the nMOS case, the characteristic will shift with changes of $Z_{p.u.}/Z_{p.d.}$ ratio.

Two points require comment:

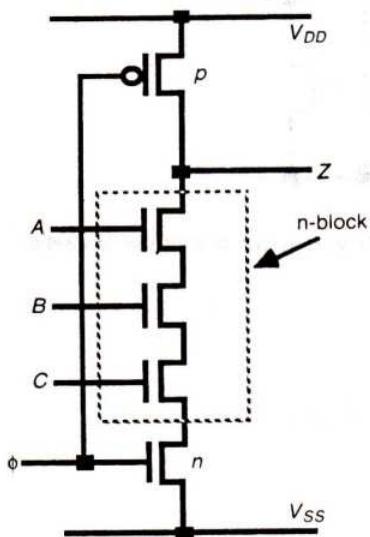
1. Since the channel sheet resistance of the p-pull-up is about 2.5 times that of the n-pull-down, and allowing for the ratio of 3:1, the pseudo-nMOS inverter presents a resistance between V_{DD} and V_{SS} which is, say, 85 kΩ compared with 50 kΩ for a comparable 4:1 nMOS device. Thus, power dissipation is reduced to about 60% of that associated with the comparable nMOS device.
2. Owing to the higher pull-up resistance, the inverter pair delay is larger by a factor of 8.5:5 than the 4:1 minimum size nMOS inverter.

6.3.4.2 Dynamic CMOS logic

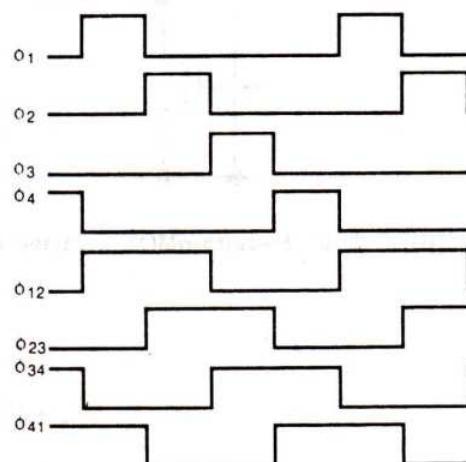
The actual logic (see Figure 6.11(a) for the schematic arrangement) is implemented in the inherently faster nMOS logic (the n-block); a p-transistor is used for the non-time-critical precharging of the output line 'Z' so that the output capacitance is charged to V_{DD} during the

off period of the clock signal ϕ . During this same period the inputs are applied to the n-block and the state of the logic is then evaluated during the on period of the clock when the bottom n-transistor is turned on. Note the following:

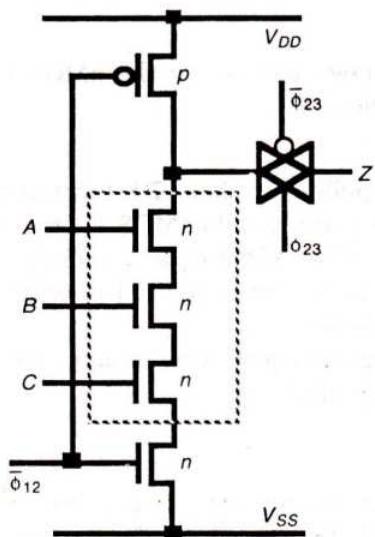
1. Charge sharing may be a problem unless the inputs are constrained not to change during the on period of the clock.



(a) Schematic



(b) Possible 4o and derived clocks



(c) Type 3 arrangement

2. Single phase dynamic logic structures cannot be cascaded since, owing to circuit delays, an incorrect input to the next stage may be present when evaluation begins, so that its output is inadvertently discharged and the wrong output results.

One remedy is to employ a four-phase clock in which the actual signals used are the derived clocks ϕ_{12} , ϕ_{23} , ϕ_{34} , and ϕ_{41} , as illustrated in Figure 6.11(b).

The basic circuit of Figure 6.11(a) is modified by the inclusion of a transmission gate as in Figure 6.11(c), the function of which is to sample the output during the ‘evaluate’ period and to hold the output state while the next stage logic evaluates. For this strategy to work, the next stage must operate on overlapping but later clock signals. Clearly, since there are four different derived clock signals which are used in sequential pairs (e.g. ϕ_{12} and ϕ_{23} in Figure 6.11(c)), there are four different gate clocking configurations. These configurations are usually identified by a type number which reflects the last of the clock periods activating the gate. For example, the gate shown would be identified as ‘type 3’ since the output Z is precharged during ϕ_2 and is evaluated during ϕ_3 (the transmission gate is clocked by ϕ_{23}). In order to avoid erroneous evaluations, the gates must be connected in allowable sequences as set out in Table 6.1.

TABLE 6.1 Dynamic logic types and sequences

Gate type	Evaluate clock	Transmission gate clock	Allowable next types
Type 1	$\bar{\phi}_{34}$	ϕ_{41}	Types 2 or 3
Type 2	$\bar{\phi}_{41}$	ϕ_{12}	Types 3 or 4
Type 3	$\bar{\phi}_{12}$	ϕ_{23}	Types 4 or 1
Type 4	$\bar{\phi}_{23}$	ϕ_{34}	Types 1 or 2

6.3.4.3 Clocked CMOS (C^2MOS) logic

The general arrangement may be made clearer by Figure 6.12. The logic is implemented in both n- and p-transistors in the form of a pull-up p-block and a complementary n-block pull-down structure (Figure 6.12(a)), as for the inverter-based CMOS logic discussed earlier. However, the logic in this case is evaluated (connected to the output) only during the on period of the clock. As might be expected, a clocked inverter circuit forms part of this family of logic as shown in Figure 6.12(b). Owing to the extra transistors in series with the output, slower rise-times and fall-times can be expected.

6.3.4.4 CMOS domino logic

An extension to the dynamic CMOS logic discussed earlier is set out in Figure 6.13. This modified arrangement allows for the cascading of logic structures using only a single phase clock. This requires a static CMOS buffer in each logic gate.

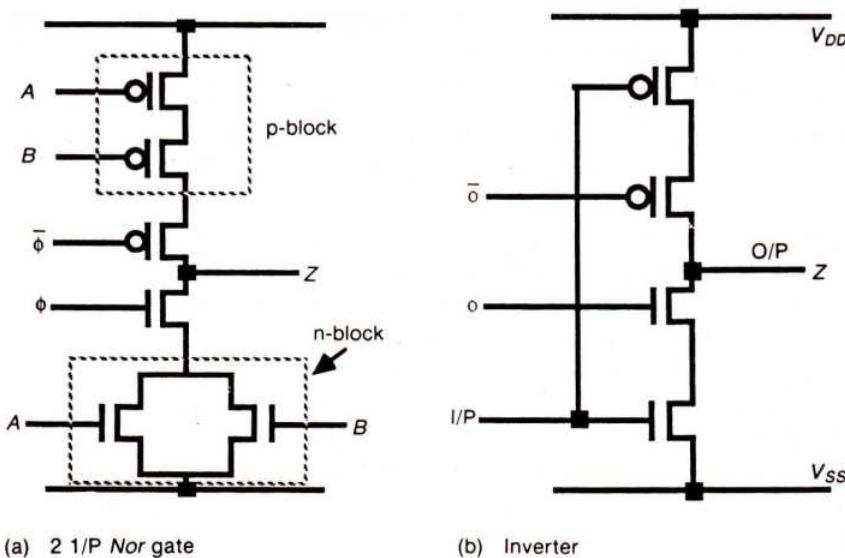
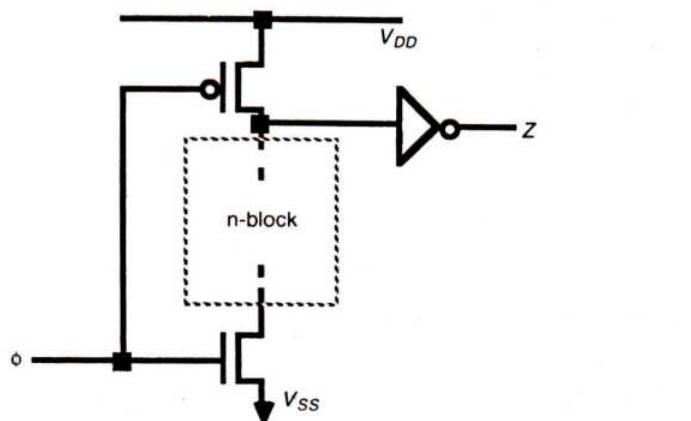
FIGURE 6.12 Clocked CMOS (C^2MOS) logic.

FIGURE 6.13 CMOS domino logic.

The following remarks will help to place this type of logic in the scheme of things:

1. Such logic structures can have smaller areas than conventional CMOS logic.
2. Parasitic capacitances are smaller so that higher operating speeds are possible.
3. Operation is free of glitches since each gate can make only one '1' to '0' transition.
4. Only non-inverting structures are possible because of the presence of the inverting buffer.
5. Charge distribution may be a problem and must be considered.

6.3.4.5 n-p CMOS logic

This is another variation of basic dynamic logic arrangement, in which the actual logic blocks are alternately 'n' and 'p' in a cascaded structure as in Figure 6.14. The precharge and evaluate transistors are fed from the clock ϕ and clockbar $\bar{\phi}$ alternately, and clearly the functions of the top and bottom transistors also alternate, between precharge and evaluate.

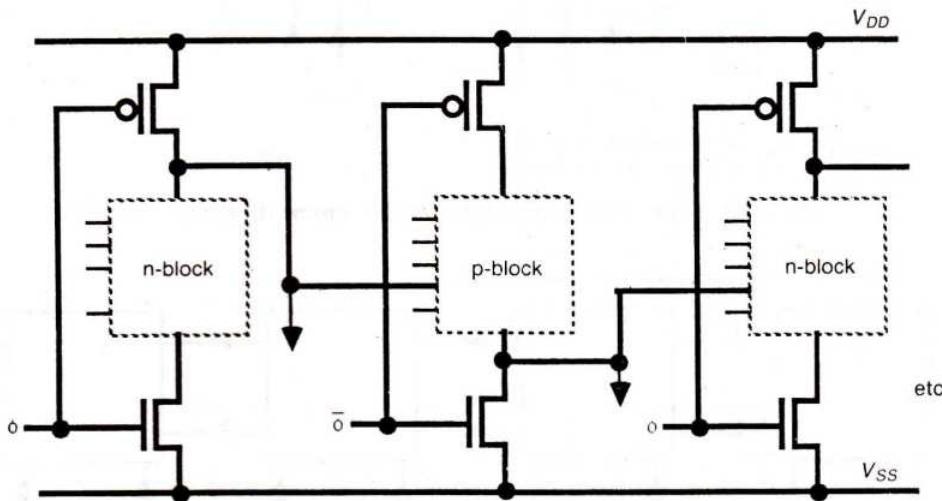


FIGURE 6.14 n-p CMOS logic.

Other forms of CMOS logic are also possible, but this text does not attempt to give an exhaustive treatment.

6.4 EXAMPLES OF STRUCTURED DESIGN (Combinational Logic)

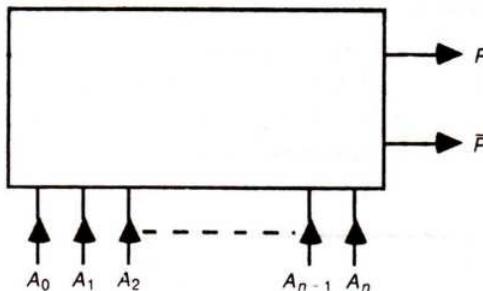
The best way to illustrate the nature of and approach to structured design is to work through some examples.

6.4.1 A Parity Generator

A circuit is to be designed to indicate the parity of a binary number or word. The requirement is indicated in Figure 6.15 for an $(n + 1)$ -bit input.

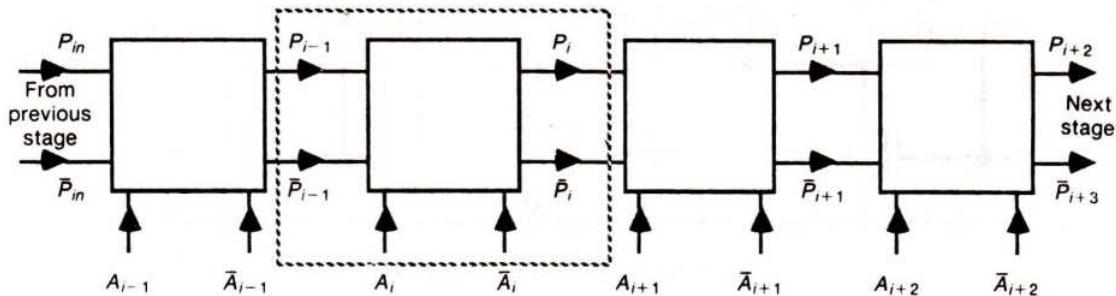
Since the number of bits is undefined, we must find a general solution on a cascadable bit-wise basis so that n can have any value. A suitably regular structure is set out in Figure 6.16. From this, we may recognize a standard or basic one-bit cell from which an n -bit parity generator may be formed. Such a cell is shown in Figure 6.17.

It will be seen that parity information is passed from one cell to the next and is modified or not by a cell, depending on the state of the input lines A_i and \bar{A}_i .



Note: $P = \begin{cases} 1 & \text{Even number of 1s at input} \\ 0 & \text{Odd number of 1s at input} \end{cases}$

FIGURE 6.15 Parity generator basic block diagram.



Note: Parity requirements are set at the left-most cell where $P_{in} = 1$ sets even and $P_{in} = 0$ sets odd parity.

FIGURE 6.16 Parity generator—structured design approach.

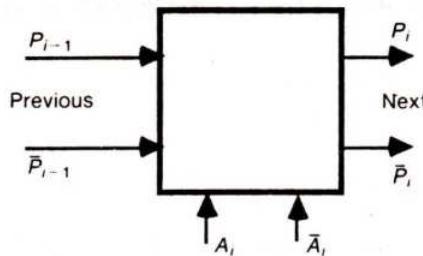


FIGURE 6.17 Parity generator—basic one-bit cell.

A little reflection will readily reveal that the requirements are:

$A_i = 1$ parity is changed, $P_i = \bar{P}_{i-1}$

$A_i = 0$ parity is unchanged, $P_i = P_{i-1}$

A suitable arrangement for such a cell is given in stick diagram form in Figure 6.18(a) (nMOS) and 6.18(b) (CMOS). The circuit implements the function

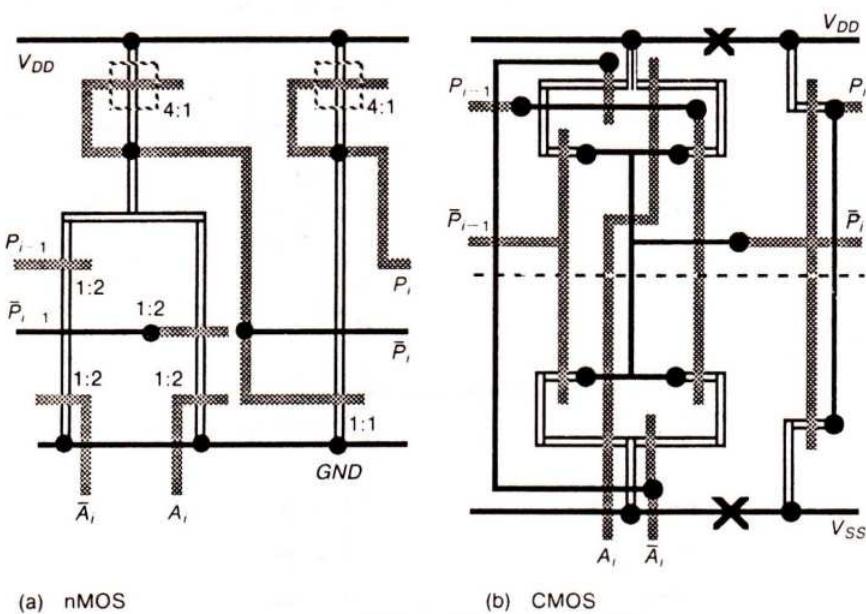


FIGURE 6.18 Stick diagrams (parity generator).

Note that a cell boundary may be chosen in each case so that cells may be cascaded at will.

When converting stick diagrams to layouts, care must be taken that the boundary is set so that *no design rule violations occur when cells are butted together*. Obviously, the boundary must also be chosen so that wastage of area is avoided and, where possible, so that *design rule errors are not present when a cell is checked in isolation*, although this may not always be possible.

Also, note that inlet and corresponding outlet points should match up both in *layer* and *position*, so that direct interconnection between cells is achieved when cells are butted.

6.4.2 Bus Arbitration Logic for n-line Bus

(This example and its solutions are similar to an example accredited to Professor John Newkirk in VTI course material.)

The functional requirements of this circuit are given by Figure 6.19 and associated truth table. If the highest priority line A_n is Hi (Logic 1), then output line A_n^P will be Hi and all other output lines Lo (Logic 0), irrespective of the state of the other input lines $A_1 \dots A_{n-1}$. Similarly, \bar{A}_{n-1}^P will be Hi only when A_{n-1} is Hi and A_n is Lo; again the state of all input lines of lower priority ($A_1 \dots A_{n-2}$) will have no effect and all other output lines will be Lo.

This requirement can be expressed algebraically as follows:

$$A_n^P = A_n$$

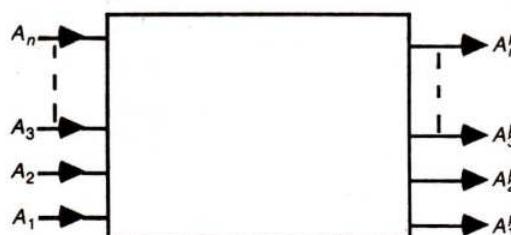
$$A_{n-1}^P = \bar{A}_n \cdot A_{n-1}$$

$$[\bar{A}_{n-1}^P = A_n + \bar{A}_{n-1}]$$

$$A_{n-2}^P = \bar{A}_n \cdot \bar{A}_{n-1} \cdot A_{n-2}$$

$$[\bar{A}_{n-2}^P = A_n + A_{n-1} + \bar{A}_{n-2}]$$

$$A_n^P = \bar{A}_n \cdot \bar{A}_{n-1} \cdot \bar{A}_{n-2} \dots \bar{A}_3 \cdot \bar{A}_2 \cdot A_1 \quad (\text{etc.})$$



Truth table

A_n	A_3	A_2	A_1	A_n^P	A_3^P	A_2^P	A_1^P
0	0	0	0	0	0	0	0
0		0	0	1	0	0	0	1
0		0	1	X	0		0	1	0
0		1	X	X	0		1	0	0
.
.
.
1	X	X	X	1	0	0	0

*X = Don't care

FIGURE 6.19 Bus arbitration logic and truth table.

A direct but unstructured implementation of these expressions may be readily envisaged and a suitable arrangement of switch (pass transistor) logic is given in Figure 6.20.

This implementation seems the obvious one, but it does suffer from the fact that as the input line under consideration moves down in significance so the complexity of the logic grows. For example, we have shown only the top three lines in Figure 6.20, but it will be seen that:

A_n requires one diffusion path and no switches

A_{n-1} requires two diffusion paths and two switches

A_{n-2} requires three diffusion paths and four switches

and so on.

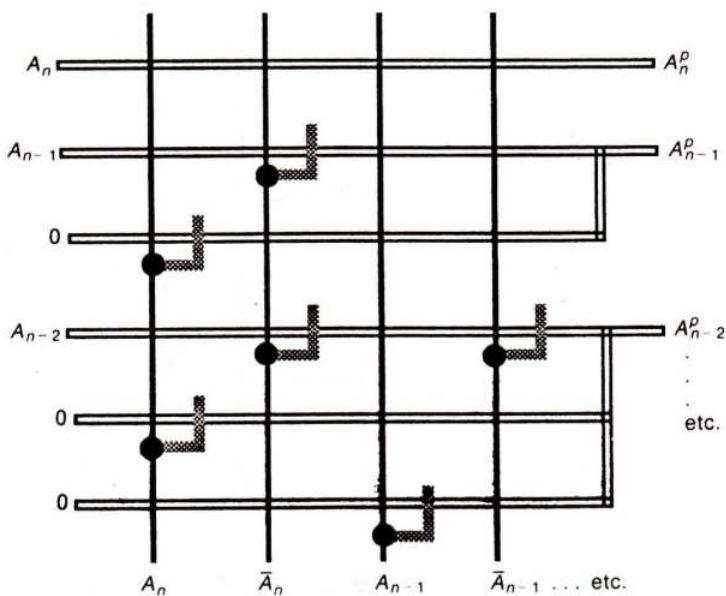


FIGURE 6.20 Stick diagram—bus arbitration logic.

This is not a regular structure and is not well suited for VLSI implementation. Therefore, we must take a cellular approach by setting out the requirements in alternative fashion as in Figure 6.21.

A regular structure having been arrived at, the requirements for each cell may be expressed as follows:

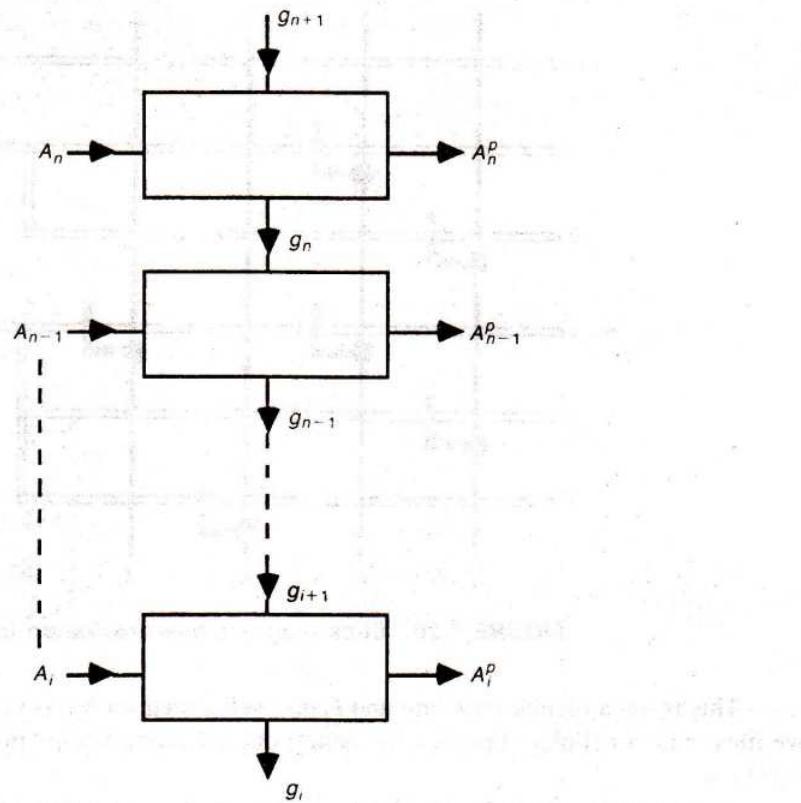
$$A_i^P = \begin{cases} g_{i+1} & \text{if } A_i = 1 \\ \text{or } 0 & \text{otherwise} \end{cases}$$

$$g_i = \begin{cases} 0 & \text{if } A_i = 1 \\ \text{or } g_{i+1} & \text{otherwise} \end{cases}$$

These requirements may be met by the circuit of Figure 6.22, but care must be taken not to cascade more than four cells without buffering the grant line.

The art of arriving at conveniently expressed relationships which allow a structured design is one which must be cultivated and it is often helped by adopting an ‘if, then, else (or otherwise)’ approach. The solution to the problem under consideration can be formulated after expressing the need of each cell in words:

$$\left. \begin{array}{l} \text{If } A_i = 1 \text{ then } A_i^P = g_{i+1}, \\ \text{else } A_i^P = 0 \text{ (if } A_i = 0) \end{array} \right\} \text{both } A_i^P \text{ and } g_i \text{ can be} \\ \left. \begin{array}{l} \text{If } A_i = 0 \text{ then } g_i = g_{i+1} \\ \text{else } g_i = 0 \text{ (if } A_i = 1) \end{array} \right\} \text{derived from } g_{i+1}$$



- Notes:
1. g = grant line.
 2. If grant line is 1, none of the lines above it wants priority.
 3. If $A_i = 0$, pass grant.

FIGURE 6.21 Bus arbitration logic—structured design.

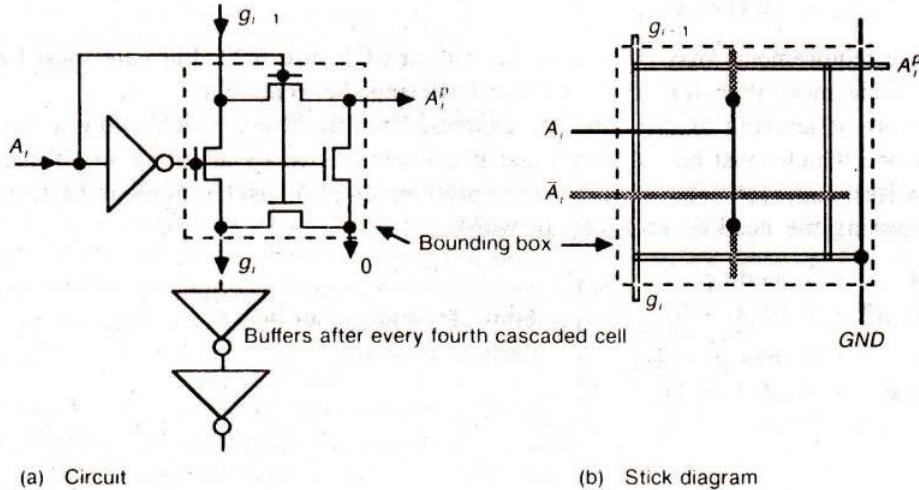


FIGURE 6.22 Bus arbitration logic—structured design.

From which we could deduce

$$\begin{aligned} A_i^P &= A_i \cdot g_{i+1} \\ g_i &= \bar{A}_i \cdot g_{i+1} \end{aligned}$$

However, there is a danger with expressions of the conventional Boolean type—a tendency to ignore the fact that MOS *switch* logic is such that not only must the logic 1 condition be satisfied, but it is also necessary to deliberately *satisfy* the logic 0 conditions. The TTL logic designer is used to working with logic circuits in which the output must be logic 0 if the logic 1 output conditions are not satisfied. However, some MOS switch-based logic circuits have the property that if the logic 1 output conditions are not met, then the output can be indeterminate or, if some storage capacitance is present (for example, input capacitance C_g of an inverter), then the output can remain at logic 1 even after the conditions which caused it no longer exist. Thus, it is necessary to deliberately implement the ‘else’ conditions. We must, therefore, write expressions for both the logic 1 and logic 0 conditions of the output lines, thus

$$\begin{aligned} A_i^P &= A_i \cdot g_{i+1}; \bar{A}_i^P = \bar{A}_i + \bar{g}_{i+1} \\ g_i &= \bar{A}_i g_{i+1}; \bar{g}_i = A_i + \bar{g}_{i+1} \end{aligned}$$

which is the circuit realized in Figure 6.22. This circuit is suitable for implementation in nMOS or in CMOS technology. Although in the CMOS case there is the possibility of replacing the n-type pass transistors by transmission gates, there is no advantage to be gained from this as the degrading of logic 1 level is counteracted by the presence of buffers after every fourth cell. There is clearly an area advantage in using simple n-type pass transistors and the only difference, therefore, between an nMOS and a CMOS design will be the type of buffer (inverter) stages.

6.4.3 Multiplexers (Data Selectors)

Multiplexers are widely used and have many applications. They are also commonly available in a number of standard configurations in TTL and other logic families. In order to arrive at a standard cell for multiplexers, we will consider a commonly used circuit, the four-way multiplexer.

The requirements and general arrangement of a four-way multiplexer are set out in Figure 6.23, from which we may write

$$Z = I_0 \cdot \bar{S}_1 \cdot \bar{S}_0 + I_1 \cdot \bar{S}_1 \cdot S_0 + I_2 \cdot S_1 \cdot \bar{S}_0 + I_3 \cdot S_1 \cdot S_0$$

where S_1 and S_0 are the selector inputs. Note that in this case we do not need to be concerned about undefined output conditions since, if S_1 and S_0 have defined logic states, output Z must always be connected to one of I_0 to I_3 .

Thus, a direct n-switch logic implementation follows which is given as Figure 6.24(a) in stick diagram form with a standard-cell-based mask layout following as Figure 6.25 and in Color plate 10.

A transmission-gate-based CMOS stick diagram is given in Figure 6.24(b). A mask

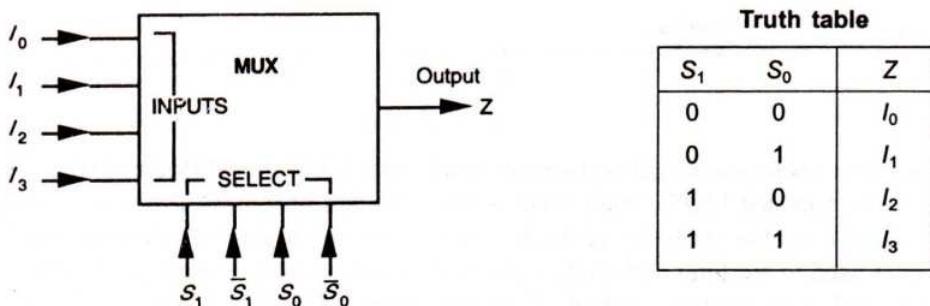


FIGURE 6.23 Selector logic circuit.

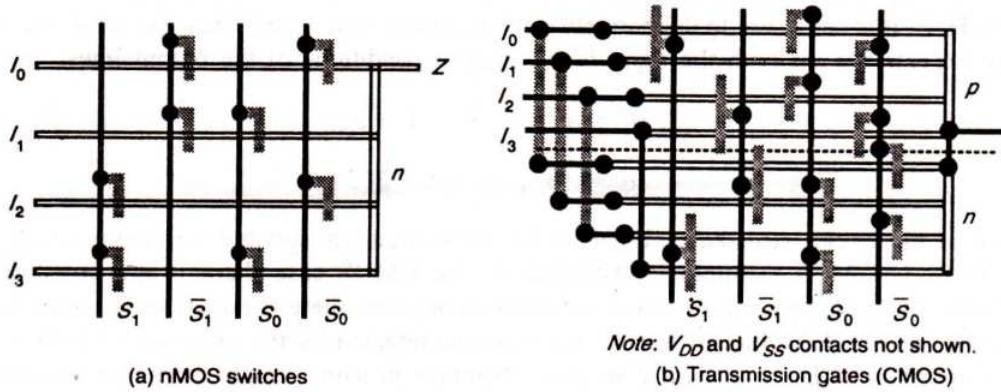


FIGURE 6.24 Switch logic implementations of a four-way multiplexer.

layout of this figure appears as Color plate 11 and it can be seen that all n-transistors are placed below the demarcation line and close to the V_{SS} rail to allow ready configuration of the p-well and V_{SS} contacts. The p-transistors are similarly placed above the notional demarcation line and close to V_{DD} . Note that logic 1 levels will not be degraded by this arrangement as those in the nMOS version are.

Now, if we can establish standard cells from which a four-way multiplexer can be composed, then we will also cover the case of the two-way multiplexer. Such a cell will, by inference, also be suitable for constructing an 8-way or a 16-way multiplexer.

For the nMOS case a standard cell is illustrated in Figure 6.25. The standard cell in this case measured $7\lambda \times 11\lambda$ and is shown in the dotted outline. Note that two versions of the cell are needed to complete the network, one version with a pass transistor as shown and the other version without. If computer-aided design tools are used, the two versions may be designed as one cell suitably parameterized to include or exclude the pass transistor. Note that in Figure 6.25 the dimensions do not include the end connection to Z .

Note also that this layout places the metal select lines over the top of pass transistors. This practice is acceptable in this situation where a transistor gate is actually driven from and connected to the particular metal line which runs across it. This method of economizing in

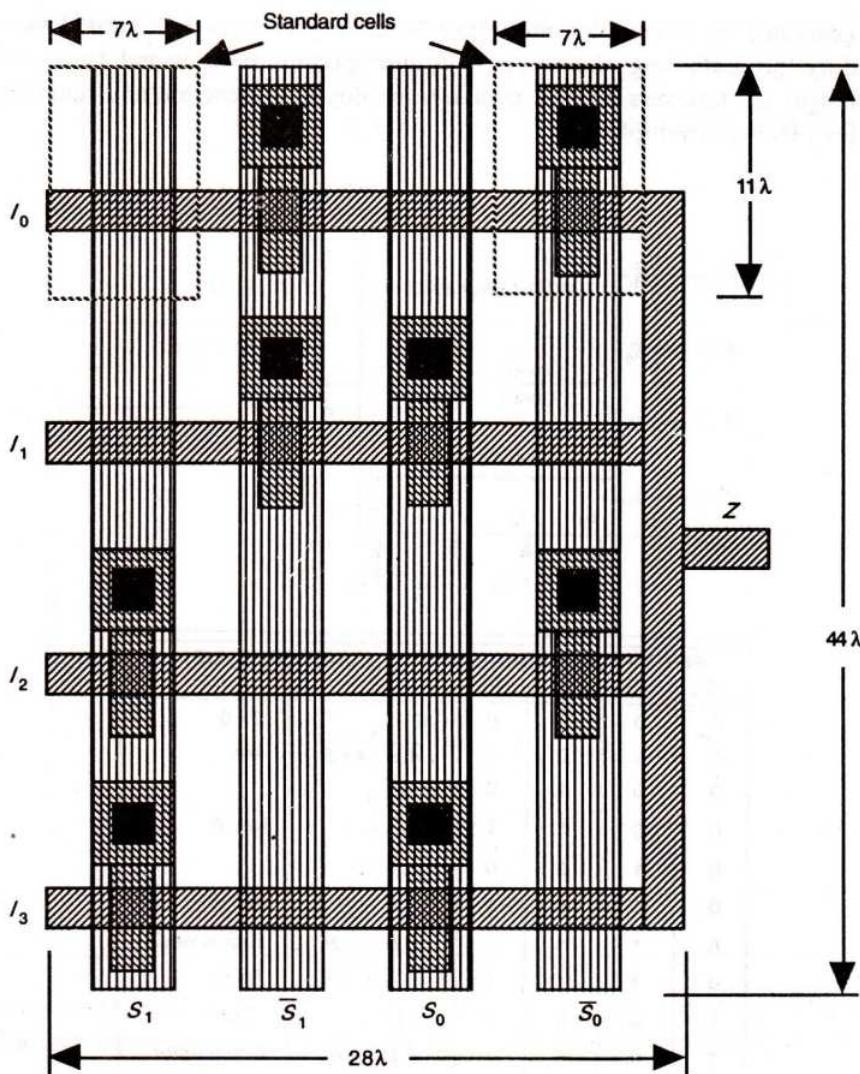


FIGURE 6.25 Four-way n-switch based multiplexer (MUX) layout (see also Color plate 10).

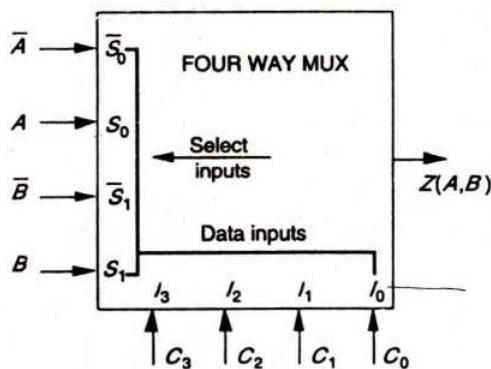
area must be used with caution when locating transistors under metal layers to which they are not connected, and may not be acceptable when the underlying transistors are used as storage points to hold a charge and retain a logic level.

6.4.4 A General Logic Function Block

An arrangement to generate any function of two variables (A, B) is readily formed from any form of four-way multiplexer.

The general approach is indicated in Figure 6.26. It will be seen that the required

function is generated by driving the multiplexer select inputs from the required two variables A and B and by ‘programming’ the inputs I_0-I_3 appropriately with 0s and 1s, as indicated in the figure. Larger multiplexers may be similarly employed to generate any function of up to four variables (16-way multiplexer).



INPUT PROGRAMMING				FUNCTION $Z(A,B)$	
C_3	C_2	C_1	C_0		
0	0	0	0	0	$Z = 0$
0	0	0	1	$\bar{A} \cdot \bar{B}; A + B$	<i>Nor</i>
0	0	1	0	$A \cdot \bar{B}$	
0	0	1	1	\bar{B}	<i>Not B</i>
0	1	0	0	$\bar{A} \cdot B$	
0	1	0	1	\bar{A}	<i>Not A</i>
0	1	1	0	$A \cdot \bar{B} + \bar{A} \cdot B$	<i>Exclusive-Or</i>
0	1	1	1	$\bar{A} + \bar{B}; \bar{A}B$	<i>Nand</i>
1	0	0	0	$A \cdot B$	<i>And</i>
1	0	0	1	$\bar{A} \cdot \bar{B} + A \cdot B$	<i>Comparator</i>
1	0	1	0	A	$O/P = A$
1	0	1	1	$A + \bar{B}$	
1	1	0	0	B	$O/P = B$
1	1	0	1	$\bar{A} + B$	
1	1	1	0	$A + B$	<i>Or</i>
1	1	1	1	1	$Z = 1$

FIGURE 6.26 General logic function block (two variables).

6.4.5 A Four-line Gray Code to Binary Code Converter

As a further exercise, which employs a very widely used logic arrangement (the *Exclusive-Or* gate), consider the requirement for code conversion from Gray to binary as set out in

By inspecting (or mapping from) Table 6.2, it will be seen that the following expressions relate the two codes:

$$\left. \begin{array}{l} A_0 = \bar{G}_0 \cdot A_1 + G_0 \cdot \bar{A}_1 \\ A_1 = \bar{G}_1 \cdot A_2 + G_1 \cdot \bar{A}_2 \\ A_2 = \bar{G}_2 \cdot A_3 + G_2 \cdot \bar{A}_3 \\ A_3 = G_3 \end{array} \right\} \quad \text{Exclusive-Or operations}$$

TABLE 6.2 Gray to binary code conversion

Gray code				Binary code			
G_3	G_2	G_1	G_0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

A suitable arrangement is set out in Figure 6.27, and the only detailed design required is that of a two input *Exclusive-Or* gate. Many arrangements are possible to implement this

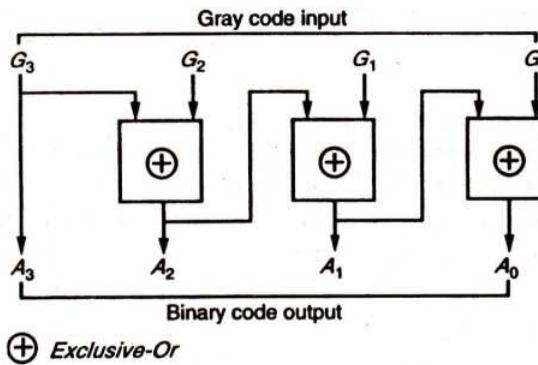


FIGURE 6.27 Gray to binary code converter.

operation, but let us consider an *Exclusive-Or* gate made up of standard logic gates, as in Figure 6.28.

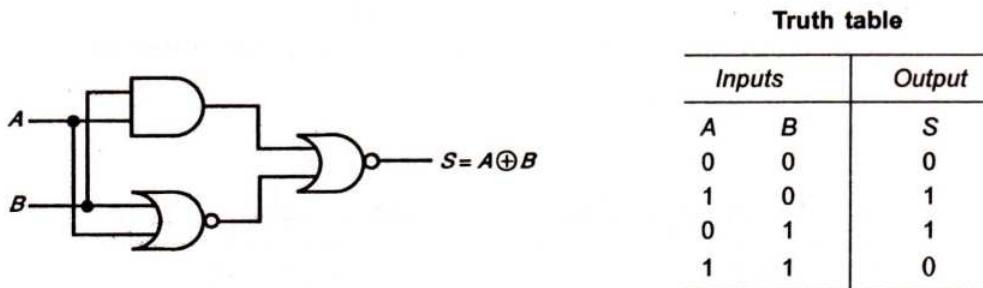


FIGURE 6.28 One possible arrangement for an *Exclusive-Or* gate.

A mask layout for this arrangement is presented in Figure 6.29; note that the p-well and p^+ mask outlines are included in the layout together with the p-well and substrate contacts. Simulation of the design yields the results given in Figure 6.30. The simulator used was the ISD* program PROBETM and the circuit extractor NETTM. Likely circuit delays can be seen quite plainly.

6.4.6 The Programmable Logic Array (PLA)

This arrangement provides a general, structured and regular way of mapping multiple output combinational logic expressions onto silicon. See Appendix C.

6.5 SOME CLOCKED SEQUENTIAL CIRCUITS

6.5.1 Two-phase Clocking

The clocked circuits to be considered here will be based on a two-phase non-overlapping clock signal as defined by Figure 6.31.

A two-phase clock offers a great deal of freedom in sequential circuit design if the clock period and the duration of the signals ϕ_1 and ϕ_2 are correctly chosen. If this is the case, data is allowed to become stable before any further transfer takes place and there is no chance of race conditions occurring.

Clocked circuitry is considerably easier to design than the corresponding asynchronous sequential circuitry. It does, however, usually pay the penalty of being slower. However, at this stage of learning VLSI design we will concentrate on two-phase clocked sequential circuits alone and thus simplify design procedures. When studying Figure 6.31, it is necessary to recognize the fact that ϕ_1 and ϕ_2 do not need to be symmetrical as shown. For a given clock

* ISD refers to Integrated Silicon Design Pty Ltd of Adelaide, South Australia. The use of their design tools is acknowledged with thanks and appreciation.

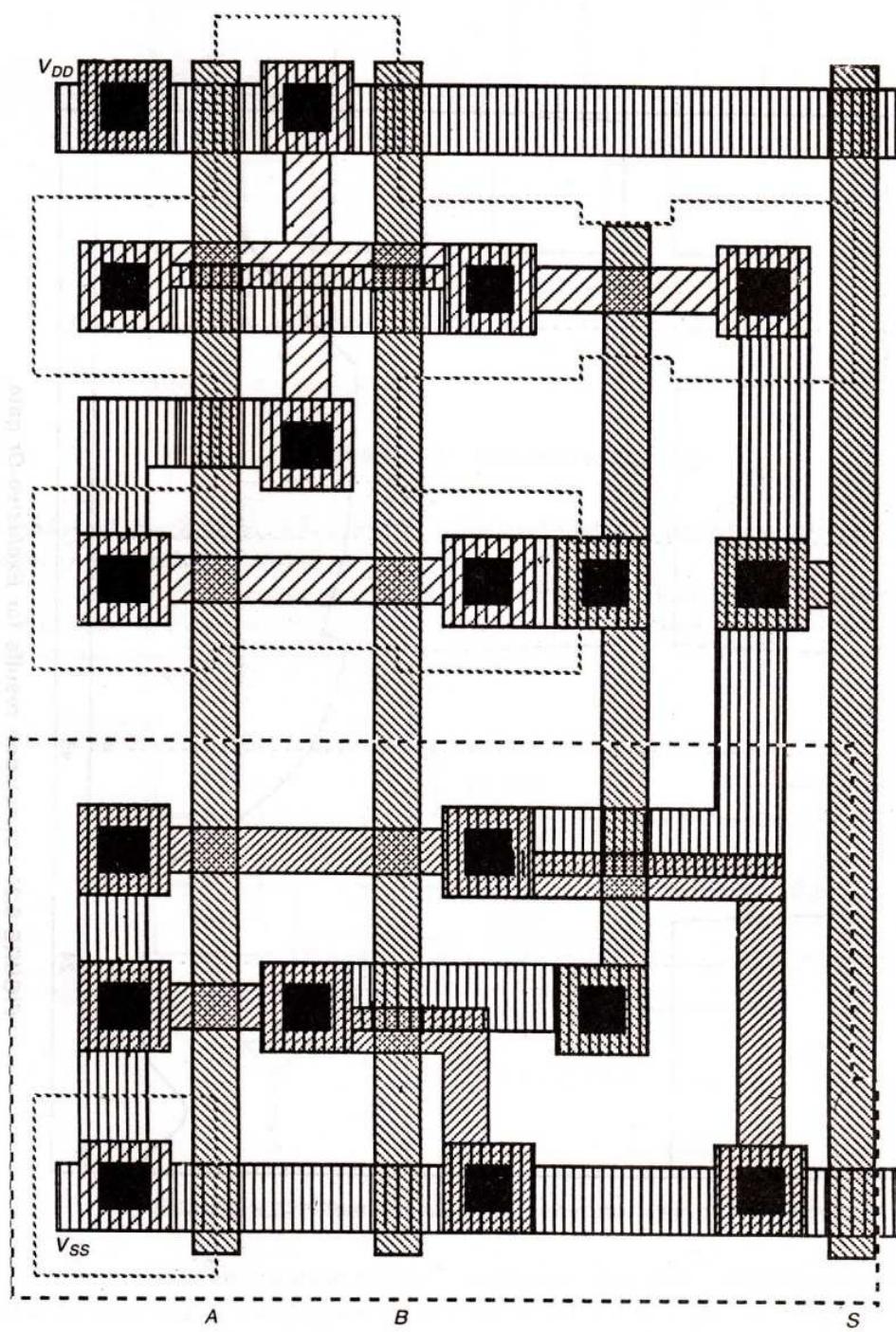
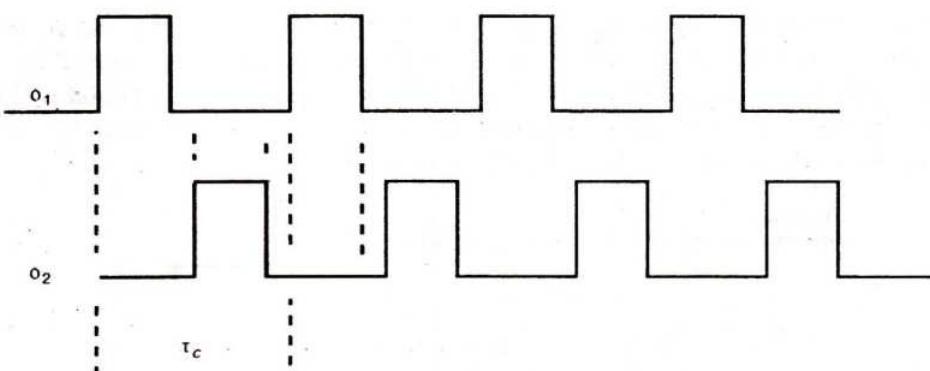


FIGURE 6.29 Mask layout for Exclusive-Or gate of Figure 6.28.



FIGURE 6.30 Simulation results for Exclusive-Or gate.



Notes:

1. τ_c = clock period
2. $o_1(t), o_2(t) = 0; \text{ all } t$

FIGURE 6.31 Two-phase clocking.

period, each clock phase period and its associated underlap period can be varied if the need arises in optimizing a design.

A number of techniques are used to generate the two clock phases. One popular method is illustrated in Figure 6.32 and it will be seen that the output frequency is one-quarter of that of the input clock.

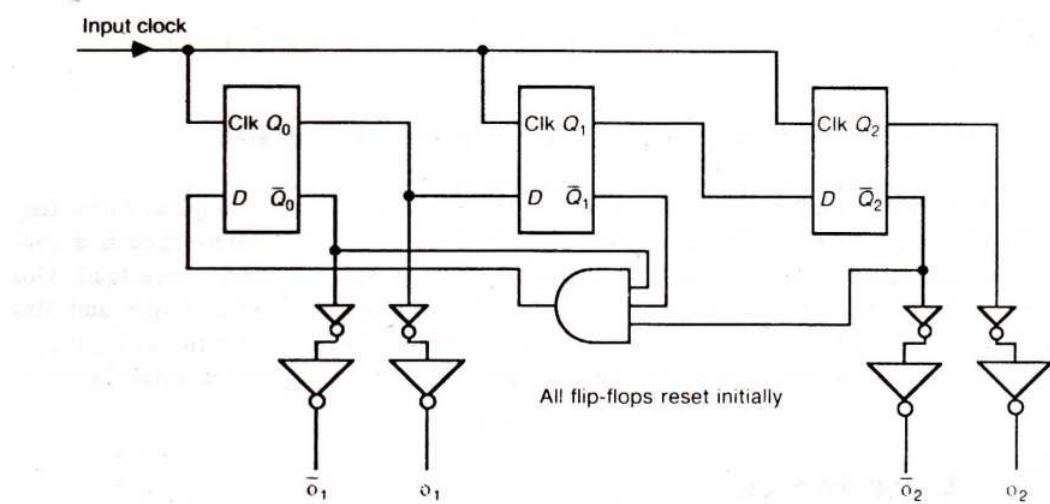


FIGURE 6.32 Two-phase clock generator using D flip-flops.

A very simple arrangement using combinational logic and generating a two-phase clock at the frequency of a single-phase input clock is set out in Figure 6.33(a). The input clock signal C is used to provide a delayed version of itself (CD) by passing it through an even number of inverters. The delay thus produced determines the underlap period for the two-

phase clock. Waveforms are as shown in Figure 6.33(b). The phase 1 signal ϕ_1 (PH1) is generated by *Anding C* with CD whilst the phase 2 signal ϕ_2 (PH2) is produced by *Noring C'* with CD' . Clearly, the minimum underlap period will be that generated by the delay through two inverters and this is also the increment by which the delay may be increased by adding further inverter pairs.

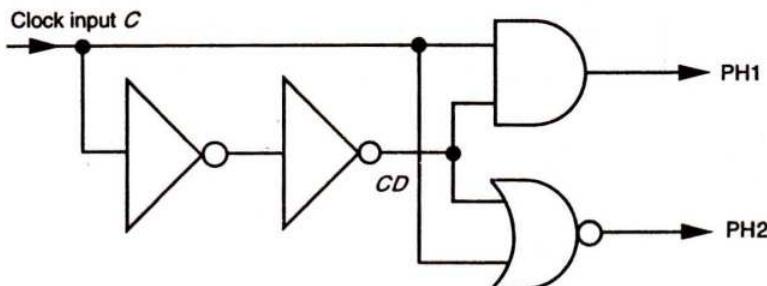


FIGURE 6.33(a) Simple two-phase clock generator circuit—basic form.

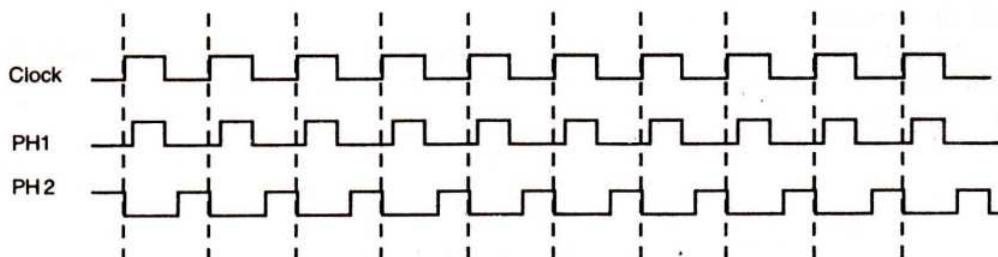


FIGURE 6.33(b) Waveforms for two-phase clock generator.

Since clock lines often feed many stages and are associated with long bus lines, they often present quite considerable capacitance to the clock line drivers. Here then is a case where a bipolar capability can be used to advantage to drive the high capacitance load. This approach is demonstrated in Figure 6.34, which uses bipolar-based output stages and also produces the complements of the two phases since complementary clocks are almost invariably required. Simulation waveforms are given in Figure 6.34(b) and a possible mask layout is presented as Color plate 12.

6.5.2 Charge Storage

A necessary feature of sequential circuits is a facility to remember or take account of previous conditions. An obvious area of application of such a facility is in memory elements, registers, finite state machines, etc.

MOS technology takes advantage of the excellent insulating properties of silicon dioxide layers on integrated circuits to store charges in capacitors, including the gate-channel capacitance of transistors. Such storage is known as *dynamic storage* since, in a reasonably short time,

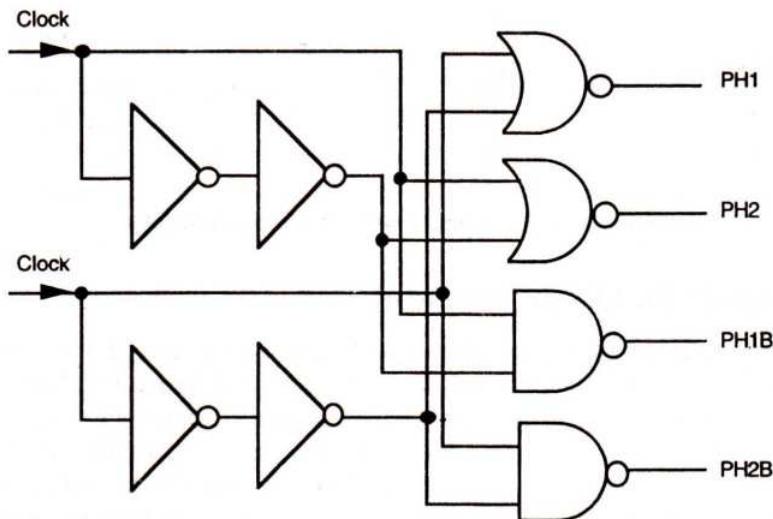


FIGURE 6.34(a) Two-phase clock generator (with complementary outputs) for BiCMOS logic implementation.

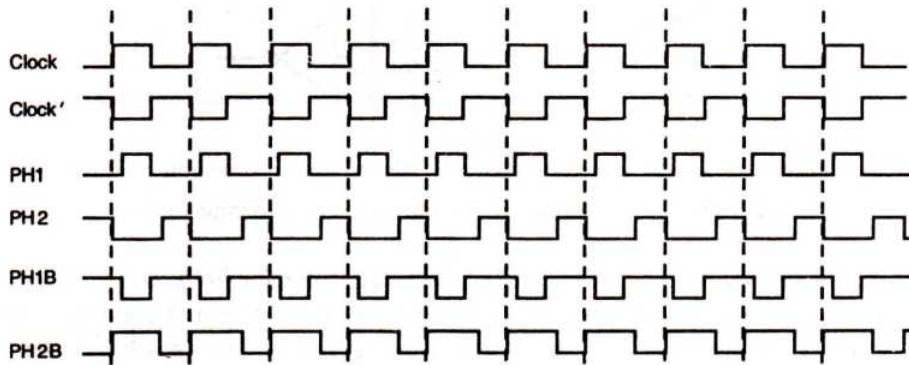


FIGURE 6.34(b) Waveforms for circuit of Figure 6.34(a).

stored charges will leak away and will have to be refreshed if data/conditions are to be retained.

Considering charges on the gate capacitance, the leaking away of the charge is mainly due to leakage currents I_s across the channel to substrate reversed biased diode. At room temperature and for typical 5 μm dimensions and typical voltages, this current is in the order of 0.1 nA, and so an approximate idea of holding time can be obtained from the simple circuit model of Figure 6.35 which considers $1 \square C_g$ initially charged to 5 volts.

This simple model indicates storage times of up to, say, 0.25 msec to discharge from V_{DD} to V_{inv} ($= 0.5V_{DD}$), but it should be noted that current I_s doubles for every 10°C rise in temperature so that the storage time is halved.

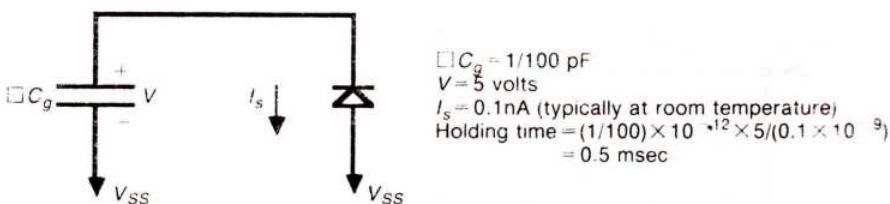
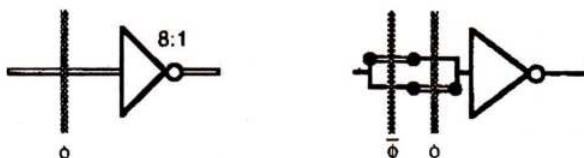


FIGURE 6.35 Simple stored charge model.

6.5.3 Dynamic Register Element

The basic dynamic register element is shown in Figure 6.36 in mixed stick/circuit notation and may be seen to consist of three transistors for nMOS and four for CMOS per stored bit in complemented form. The element's operation is simple to appreciate. $(V_{in})_t$ is clocked in by ϕ_1 (or ϕ_2) of the clock and charges the gate capacitance C_g of the inverter to V_{in} . If subscript t is taken to represent the time during which ϕ_1 (say) is at logic 1 and subscript $t + 1$ is taken to indicate the period during which ϕ_1 is at logic 0, then the available output will be $(\bar{V}_{in})_{t+1}$ which will be maintained by the stored charge on the gate until C_g discharges or until the next ϕ_1 signal occurs.



(a) nMOS pass transistor switched (b) CMOS transmission gate switched

FIGURE 6.36 Basic inverting dynamic storage cells.

If uncomplemented storage is essential, the basic element is modified as indicated in Figure 6.37 and will be seen to consist of six transistors for nMOS and eight for CMOS. Data clocked in on ϕ_1 is stored on C_{g1} and the corresponding output appears at the output of inverter 1. On ϕ_2 this value is clocked into and stored by C_{g2} and the output of inverter 2 then presents the 'true' form of the stored bit. Note that data read in on ϕ_1 is not available at the output until sometime following the next positive edge of the clock signal ϕ_2 .

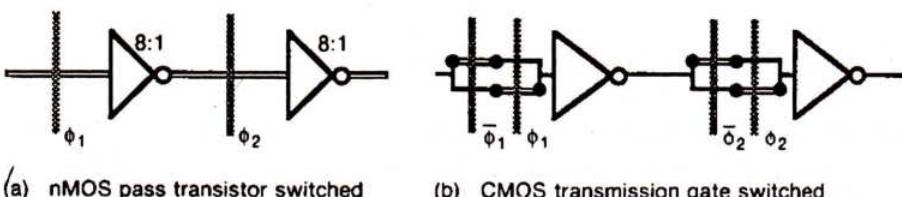


FIGURE 6.37 Non-inverting dynamic storage cells.

Dynamic storage elements and the corresponding register arrays are used in situations where signals are updated frequently (i.e. at < 0.25 msec intervals).

6.5.4 A Dynamic Shift Register

Cascading the basic elements of Figure 6.37 gives a serial shift register arrangement which may be extended to n bits. A four-bit serial right shift nMOS register is illustrated in Figure 6.38(a). Data bits are shifted in when $\phi_1.LD$ is present, one bit being entered on each ϕ_1 signal (provided that LD is logic 1). Each bit is stored in C_{g1} as it is entered, and then transferred complemented into C_{g2} during the next ϕ_2 . Thus, after a ϕ_1 followed by ϕ_2 signal, the stored bit is present at the output of inverter 2. On the next ϕ_1 , the next input bit is stored in C_{g1} and simultaneously the first bit stored is passed on to inverter pair 3 and 4 by being stored in C_{g3} , and so on. It will be seen that bits are thus clocked to the right along the shift register on each ϕ_1 followed by ϕ_2 sequence. Once four bits are stored, the data is available in parallel form at the outputs of inverters 2, 4, 6 and 8, and is also available in serial form from the output of inverter 8 when $\phi_1.RD$ is high as further clock sequences are received (where RD is the serial read control signal). The operation of the CMOS version (Figure 6.38(b)) is similar, transmission gates replacing inter-stage pass transistors and C_{in1} replacing C_{g1} , etc., as the storage capacitance.

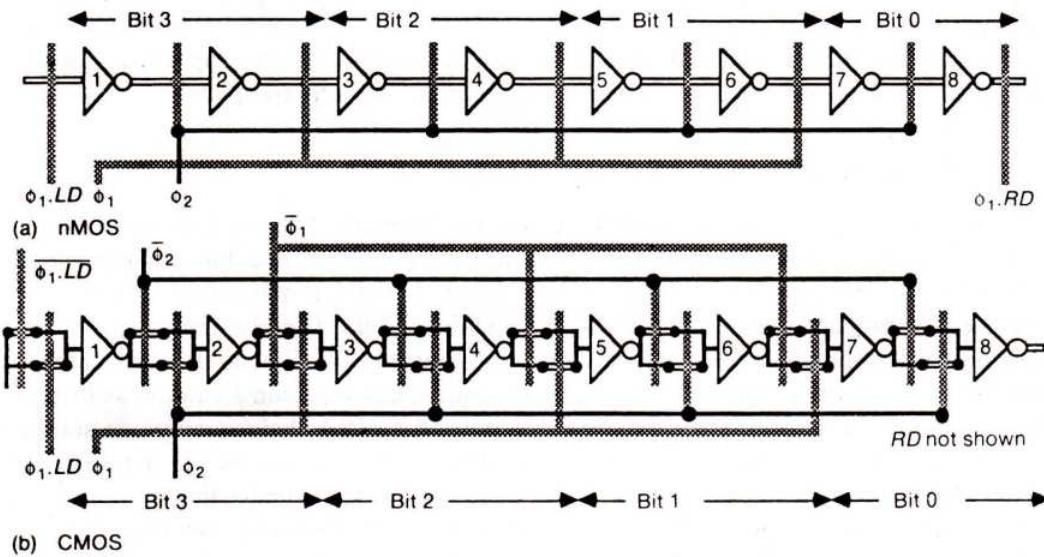


FIGURE 6.38 Four-bit dynamic shift registers (nMOS and CMOS).

Many variations of this basic arrangement are possible, but in general they are all based on the basic cell consisting of an inverter and a pass transistor or a transmission gate. Suitable standard cells are shown in stick diagram form in Figure 6.39 with the corresponding mask layouts in Figure 6.40. Note that two nMOS layouts are given (using butting and buried contacts respectively) and one possible CMOS layout is suggested (see also Color plate 7).

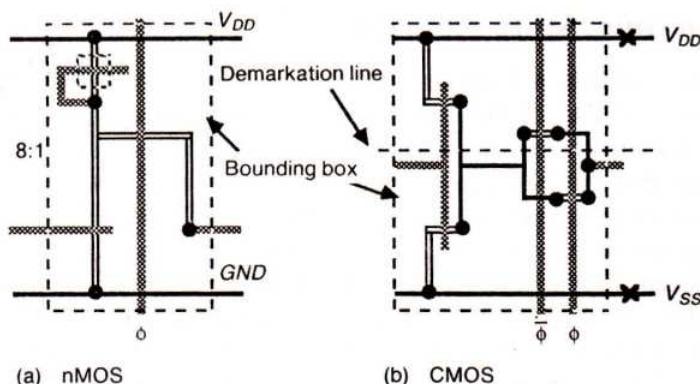


FIGURE 6.39 Stick diagrams for shift register cells.

6.6 OTHER SYSTEM CONSIDERATIONS

When designing at leaf-cell level, it is easy to lose sight of overall system requirements and restrictions. In particular, the use of buses to interconnect subsystems and circuits must always be most carefully considered; such matters and the current-carrying capacity of aluminum wiring used for V_{DD} and GND or V_{SS} rails are often overlooked completely.

6.6.1 Bipolar Drivers for Bus Lines

Bus structures carrying data or control signals are generally long and connected to and through a significant number of circuits and subsystems. Thus, the bus capacitances are appreciable and thought must be given to the manner in which any bus line is to be driven. Otherwise, the propagation of signals may be a slow process. Clearly, the capacitive load-driving properties of bipolar transistors in a BiCMOS process make bipolar drivers an attractive proposition for bus lines. However, this must be approached with some caution as the speed of bipolar drivers is only fully realized with bus lines for which there is only one source of drive, for example, as in the case of clock line drivers. Bipolar drivers are not so suitable where one or other of several sources drives a common bus since under those circumstances a series switch must be inserted between each source of drive and the bus. The series resistance of such a switch to a large extent negates the speed advantage. In such cases MOS transistor drivers are often used and the following basic approaches may be considered.

6.6.2 Basic Arrangements for Bus Lines

There are three classes of bus—passive, active, and precharged. A *passive* bus rail is a floating rail to which signals may be connected from drivers through series switches, for example, pass transistors, to propagate along the bus and from which signals may be taken, also through pass transistors (see Figure 6.41).

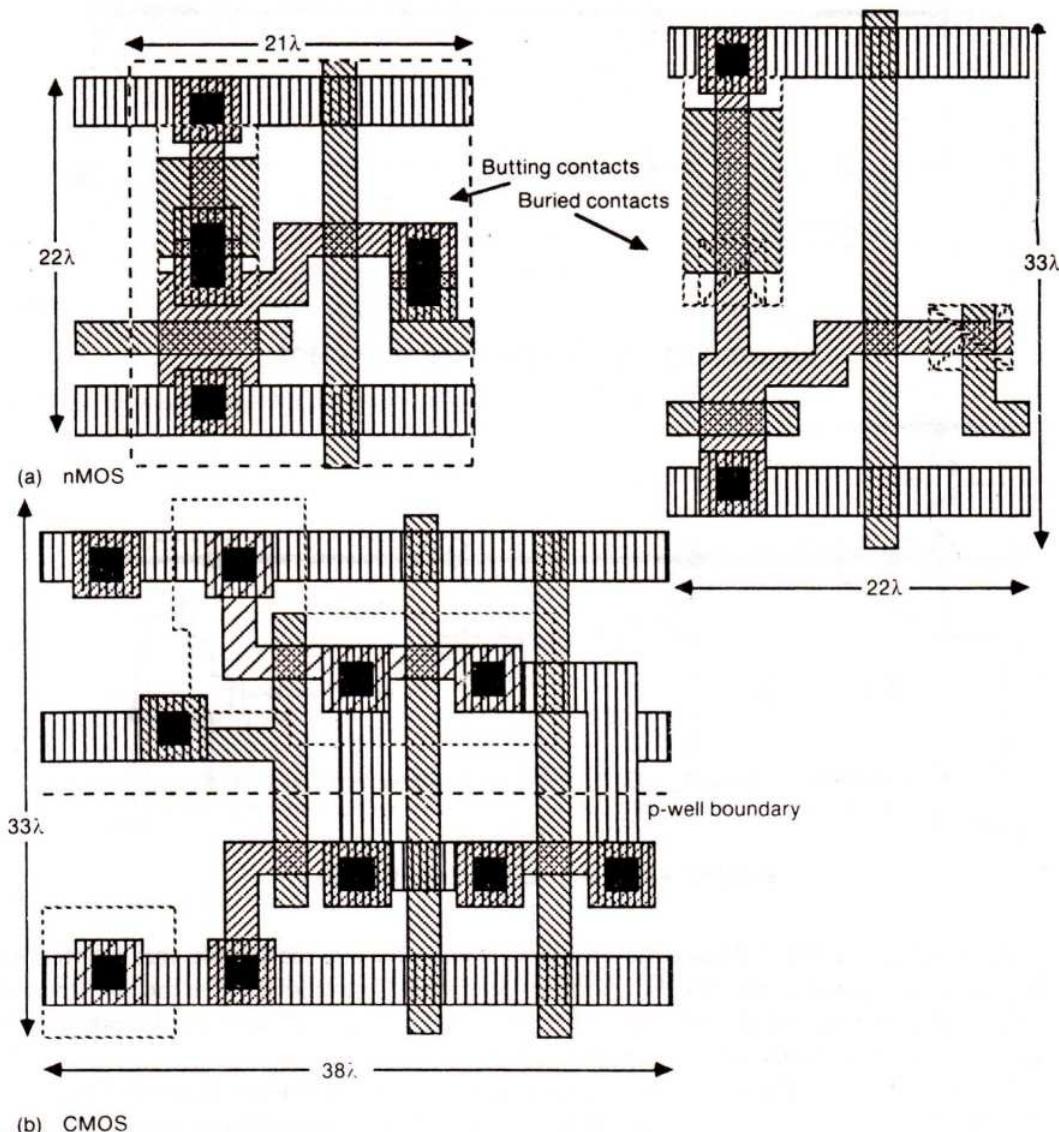
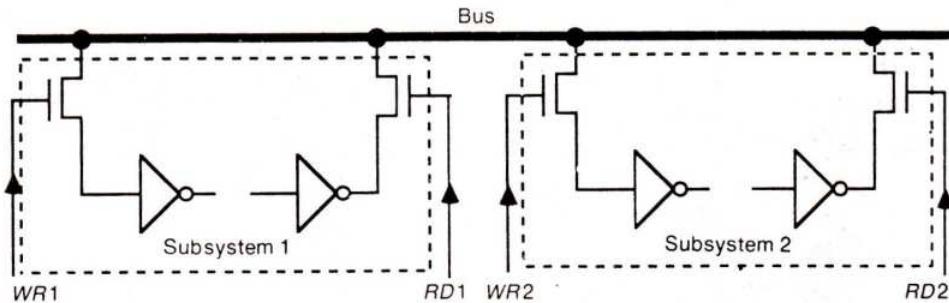


FIGURE 6.40 Mask layouts for nMOS and CMOS shift register cells (see also Color plate 7).

A form of *active bus* is to treat the bus rail as a wired *Nor* connection which has a common pull-up $R_{p.u.}$ and n-type pull-down transistors or series n-type transistor logic pull-downs where there are circuits which must be selected to drive the bus. Signals are taken off the bus in a similar manner and the general arrangement is given as Figure 6.42. This arrangement is not suited to complementary CMOS logic-based designs since it is based on pull-down logic only.



Note: For CMOS the pass transistors could become transmission gates.

FIGURE 6.41 Passive bus—nMOS or CMOS.

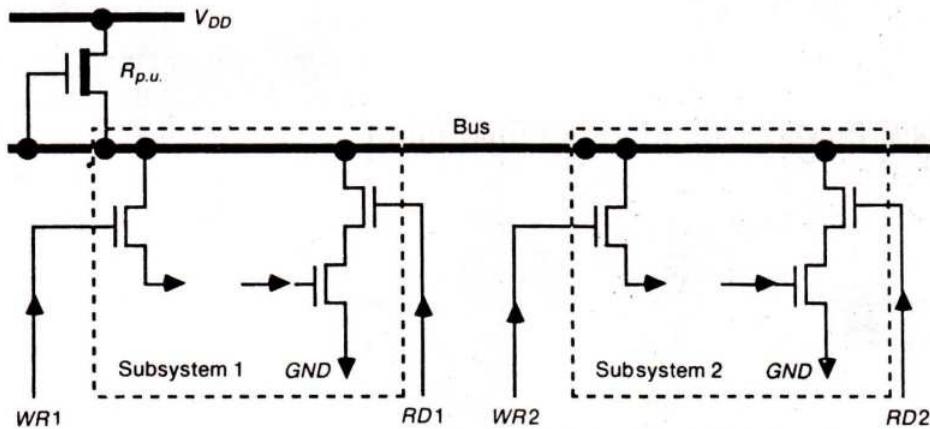


FIGURE 6.42 Active bus (not CMOS).

The passive bus suffers from ratio problems in that, for any reasonable area restrictions on the bus driver circuits, the bus will be slow to respond, particularly for the ΔV (logic 0 to 1) transitions, because of the relatively high value pull-up resistance of the drivers and the associated series pass transistor or transmission gate.

The active bus is better in that more time is available for the bus to charge to V_{DD} , since $R_{p.u.}$ is always connected to the bus and there are no series pass transistors between $R_{p.u.}$ and the bus. However, there are still ratio problems which limit the speed of the bus if reasonable area is to be occupied.

6.6.3 The Precharged Bus Concept

The *precharged* bus approach limits the effects of bus capacitance in that a single pull-up transistor which is turned on only during ϕ_2 (say) provides for the bus to charge during the ϕ_2 on period; the size of this transistor can be made relatively large (i.e. a low $L:W$ ratio) and, therefore, have a low resistance. There are no ratio problems between it and the bus drivers

since they are never turned on at the same time. The bus drivers merely pull down (or not) the precharged bus by discharging C_{Bus} . The arrangement is given at Figure 6.43 and, in effect, a ratioless precharged wired Nor circuit is formed by the bus system. However, care must be taken in nMOS systems when using logic 1 levels from the bus since the bus never reaches V_{DD} , due to threshold voltage effects in the precharging transistor.

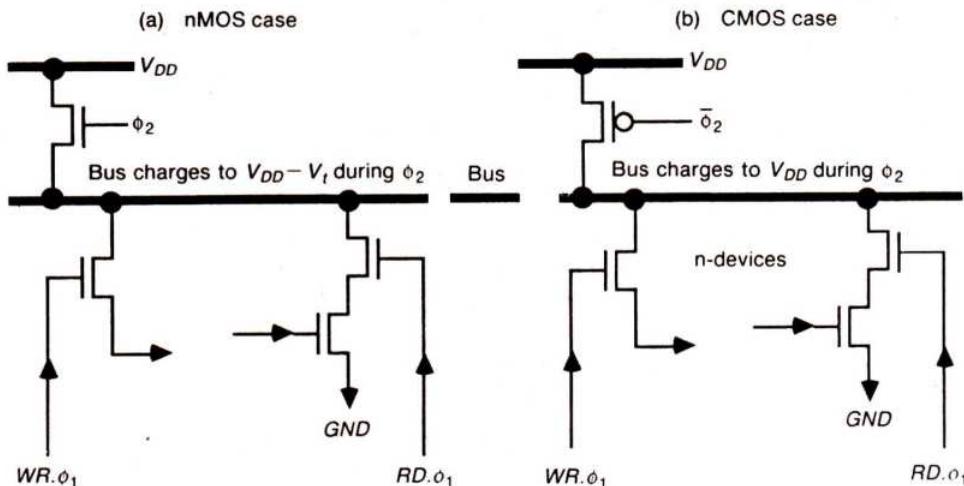


FIGURE 6.43 Precharged bus—nMOS and CMOS.

Cross-talk and delay factors are also of significance in bus design. For example, many signals on chip may be propagated for some considerable distance (in chip dimension terms) along metal buses. Now metal buses even of minimum width are relatively wide (e.g. $3\lambda = 7.5 \mu\text{m}$ in $5 \mu\text{m}$ technology), and thus have significant area capacitance to substrate (almost $2.5 \times 10^{-4} \text{ pF per } \mu\text{m length}$ for the example). This does not give rise to serious delay-line effects since the metal exhibits a low resistance (approximately $0.01 \Omega/\mu\text{m}$ for the example) but a metal bus of any length presents a significant capacitive load to the driver. For example, a bus 400λ ($1000 \mu\text{m}$) long will present a total $C = 0.25 \text{ pF}$.

Since metal also has appreciable thickness—typically $1 \mu\text{m}$ —the edge of a long bus represents a significant area. For the 400λ long bus considered above the area of each edge will be $1000 \mu\text{m}^2$. This may give rise to cross-talk noise between two or more buses which run side by side for any appreciable length. This problem is not as serious in silicon chip designs as in GaAs technology, for example, owing to the relatively low dielectric constant (approx. = 4) for the silicon dioxide which will form the dielectric between the edges of two parallel buses.

Bus structures are widely used and will be further discussed in following sections of this text.

6.6.4 Power Dissipation for CMOS and BiCMOS Circuits

For pseudo-nMOS type circuitry, current and power are readily determined in a manner similar to nMOS. However, for complementary inverter-based circuits we may proceed by

first recognizing that the very short current pulses which flow when circuits of this type are switching between states are generally negligible in comparison with charge and discharge currents of circuit capacitances. Then we may see that overall dissipation is composed of two terms:

1. P_1 the dissipation due to the leakage current I_1 through an ‘off’ transistor. Consequently, for n transistors, we have

$$P_1 = n.I_1 V_{DD}$$

where $I_1 = 0.1 \text{ nA}$, typically at room temperature.

2. P_s is the dissipation due to energy supplied to charge and discharge the capacitances associated with each switching circuit. Assuming that the output capacitance of a stage can be combined with the input capacitance(s) of the stage(s) it is driving and then represented collectively as C_L , then, for n identical circuits switched by a square wave at frequency f it may be shown that

$$P_s = C_L V_{DD}^2 f$$

The total power dissipation P_T is thus

$$P_T = P_1 + P_s$$

from which the average current may be deduced.

Power dissipation for bipolar devices can be simply modeled by

$$P = V_{cc} \times I_c$$

where V_{cc} is the supply voltage and I_c is the current through the device.

It may be seen that BiCMOS switching devices will exhibit a constant value for power dissipation, not frequency-dependent like CMOS.

6.6.5 Current Limitations for V_{DD} and $GND(V_{SS})$ Rails

A problem often ignored is that of metal migration for high current densities in metal conductors. If the current density exceeds a threshold value then one finds that metal atoms begin to move in the direction of the current. For aluminum conductors this threshold value is

$$J_{th} \doteq 1 \text{ to } 2 \text{ mA}/\mu\text{m}^2$$

The danger points occur where there is a narrowing or constriction in the conductor. At these points the current density is at its highest and metal is transported from the constricted regions which, in consequence, become even more constricted and eventually may blow like a fuse. The actual mechanism of atomic transport of metal in a thin film carrying relatively high currents is well understood, but the science of predicting the location and the time of such occurrences is not well developed.

By way of example, we may consider the question of how many nMOS 8:1 inverters (as in a dynamic shift register) can be driven by a minimum size conductor assuming lambda-

based rules and 5 μm technology. From the design rules, the metal is 3λ wide, which corresponds to 7.5 μm . The thickness of the conductor is about 1 μm as shown in Figure 6.44.

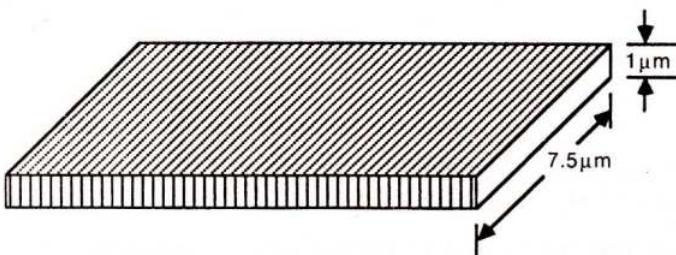


FIGURE 6.44 A minimum size metal path or wire.

For 8:1 inverter (e.g. 8:1 p.u. and 1:1 p.d.)

$$\begin{aligned} R &= (8 + 1) \times 10^4 \Omega \\ &= 90 \text{ k}\Omega \end{aligned}$$

Therefore

$$\text{Current } I = \frac{5}{90} = 0.06 \text{ mA per inverter.}$$

Now, with a wire cross-section of 7.5 μm^2 , the current density limitation $J_{th} = 1 \text{ mA}/\mu\text{m}^2$ implies that a current of 7.5 mA can be supplied. Thus about 125 inverters can be driven.

One approach that may be pursued to allow some increase in the current density above the specified critical limit is to take advantage of the ‘relaxation effect’ that occurs in the metal when electron flow occurs in short pulses rather than at a steady state level.

However, the important factor here is that a standard (minimum) width metal conductor can only support a subsystem of quite modest size. Thus, in a design of any complexity we must ensure that this fact is not overlooked and power rail distribution becomes an important and often complex issue.

6.6.6 Further Aspects of V_{DD} and V_{SS} Rail Distribution

Ideally, the power distribution rails (power distribution buses) for a chip should provide a constant and equal voltage supply to each and every device on the chip. Rails should also be able to supply the current required by every device. Clearly, these ideals are not achievable in practice and issues which determine the limitations are:

1. metal migration imposed current density restrictions—as already discussed in the preceding section.
2. the IR drop due to rail series resistance.
3. the series inductance of the rails.

The IR drops are readily calculated, provided that the currents in any bus section can be estimated since the metal bus cross-sectional area and length for that section are known.

For a parent bus supplying current to other uniformly distributed short bus branches along the length L of the parent bus, then the current at any distance x from the source is given by

$$I_x = I_L \left(\frac{1 - x}{L} \right)$$

where

I_L = the total load current supplied by the parent bus

x = the distance from the source.

The voltage drop at, say, the far end of the bus can be estimated from

$$\begin{aligned}\Delta V &= \rho \cdot \frac{I_L}{A} \left(\text{integral from 0 to } L \text{ of } \left(1 - \frac{x}{L} \right) dx \right) \\ &= \rho \cdot I_L \cdot \frac{L}{2A}\end{aligned}$$

where

ρ = resistivity of metal

A = cross-sectional area of metal bus.

However, the bus structure is not usually as regular as envisaged here so that estimation of the voltage drop at any point is not as simple a matter as implied above if accurate determination is required.

The transmission line nature of any wiring introduces the possibility of voltage transients due to its self-inductance L_0 . The transient changes in voltage due to the presence of self-inductance can be modeled by

$$\Delta V = L_0 \frac{dI}{dt}$$

where

dI/dt is the rate of change of line current.

Regarding the bus/oxide/substrate structure as a microstrip, the inductance L_0 is given by

$$L_0 = Z_0 \sqrt{\frac{\epsilon_{eff}}{c}}$$

where

Z_0 is the characteristic impedance

c is the velocity of light

ϵ_{eff} is the effective dielectric constant.

In general terms, line impedance Z_0 is given by

$$Z_0 = \left(\frac{L}{C} \right)^{1/2}$$

where

L and C are the values per unit length of the bus.

Clearly, transient voltages induced in either the V_{DD} or the $GND(V_{SS})$ rail may lead to noise margin problems for inverters and gates.

IR drops generally can give rise to deterioration in the noise margins. This can be visualized with the aid of Figure 6.45.

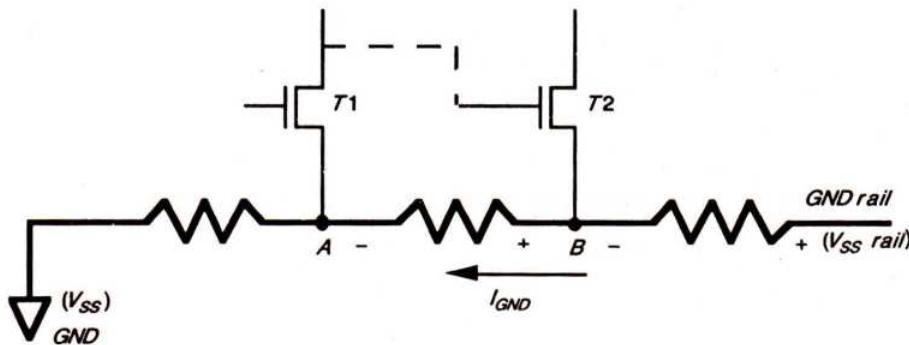


FIGURE 6.45 Ground (V_{SS}) rail noise.

It may be seen that if $T1$ is switched on, then any transient at point A and/or DC voltage induced in the GND (V_{SS}) rail from point B to GND will affect the noise margins for the next stage, $T2$.

6.7 OBSERVATIONS

The material presented up to this point has provided a basic toolkit and has introduced techniques with which to approach the task of VLSI system design. In this context, it is most important to learn by doing and the tutorial exercises included in the text are designed for that purpose. Instructors may wish to introduce further exercises from which students will undoubtedly benefit greatly. The best (perhaps the only) way to learn to design VLSI systems is to do it.

The next three chapters tackle the design of a four-bit data path system as part of a four-bit microprocessor.

6.8 TUTORIAL EXERCISES

1. (a) Construct a color-coded stick diagram to represent the design of the following integrated nMOS and CMOS structures and indicate pull-up/pull-down ratios in each case:
 - (i) three-input *Nand* gate;
 - (ii) three-input *Nor* gate;
 - (iii) 8:1 multiplexer circuit incorporating an enable control line;
 - (iv) a dual-serial shift register capable of holding and shifting (right) two 4-bit words;
 - (v) a selectively loadable dynamic register to hold one four-bit word (parallel).
 (b) For question 1(a)(iv) draw the corresponding transistor circuit diagrams.
2. Construct a stick diagram for an nMOS or CMOS parity generator as in Figure 6.46. The required response is such that $Z = 1$ if there is an even number (including zero) of 1s on the inputs and $Z = 0$ if there is an odd number. (Use simple color coding for stick diagrams.)

Configure your design in a modular expandable fashion so that the inputs could be increased to five or more quite readily, using the basic cell suggested in Figure 6.47.

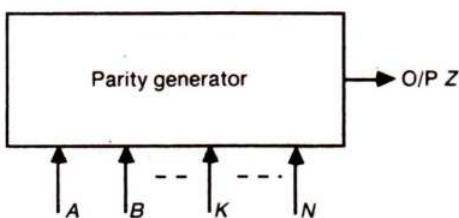


FIGURE 6.46 Parity generator outline.

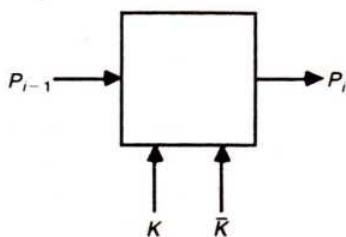


FIGURE 6.47 Parity generator cell.

3. (a) Construct a color-coded stick diagram to represent the design of an integrated nMOS structure to decode the three input lines E_0 , E_1 , and E_2 into eight output lines Z_0 , Z_1 ..., Z_7 , in accordance with the following truth table.

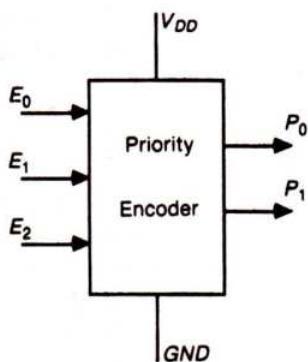
Truth table

E_2	E_1	E_0	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

- (b) Discuss the expandability or otherwise of your structure and the ease with which it would translate to CMOS.
4. A priority encoder is a combinational circuit in which each input is assigned a priority with respect to the other inputs, and the output code generated at any time is that associated with the highest priority input then present.
- Construct a color-coded stick diagram to implement such a structure as in the following table with Figure 6.48.

Truth table

E_2	E_1	E_0	P_1	P_0
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1



Truth table

E_2	E_1	E_0	P_1	P_0
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

FIGURE 6.48 Priority encoder.

5. Referring to section 6.4.6 and to the development of an *Exclusive-Or* gate, design an alternative form of two I/P *Exclusive-Or* using transmission gates and inverters only. Your design should include a stick diagram and a mask layout.