Eastern Economy Edition

Third Edition

Basic VLSI Design

Douglas A. Pucknell
Kamran Eshraghian

PHI Learning

PHI

Chapter 5

Scaling of MOS Circuits

Little things are pretty.

— Proverb

Good things come in small packages.

— Proverb

OBJECTIVES

VLSI fabrication technology is still in the process of evolution which is leading to smaller line widths and feature size and to higher packing density of circuitry on a chip.

The scaling down of feature size generally leads to improved performance and it is important therefore to understand the effects of scaling. There are also future limits to scaling down which may well be reached in the next decade.

Although this chapter may be seen by some to interrupt the flow of the text toward actual VLSI design, the authors considered this an appropriate topic following the previous chapters dealing with basic parameters and characteristics which, of course, are all affected by scaling.

Microelectronic technology may be characterized in terms of several indicators, or figures of merit. Commonly, the following are used:

- Minimum feature size
- Number of gates on one chip
- Power dissipation
- Maximum operational frequency
- Die size
- Production cost.

Many of these figures of merit can be improved by shrinking the dimensions of transistors, interconnections and the separation between features, and by adjusting the doping levels and supply voltages. Accordingly, over the past decade, much effort has been directed toward the upgrading of process technology and the resultant scaling down of devices and feature size.

In the design processes postulated by Mead and Conway and used for most examples in this text, it has been the practice to dimension all layouts in terms of λ. A value may then be allocated to λ, prior to manufacture, which is in line with the capabilities of the silicon foundry or is determined by current technology and/or meets the specifications which have been set out for the circuit. One benefit of this approach lies in the fact that the design rules have been formulated in such a way as to allow *limited* direct scaling of the dimensions of circuits, so that today's design is not automatically outdated when line widths are reduced (i.e. the value allocated to λ is reduced) by advances in tomorrow's technology.

Scaling is therefore an important factor, and it is essential for the designer to understand the implementation and the effects of scaling. In writing this chapter, the authors gratefully acknowledge the useful contributions made by Dr A. Osserain and Dr B. Hochet, both of the Swiss Federal Institute of Technology, Lausanne, Switzerland.

This chapter discusses scaling and its effect on performance and indicates some problems and ultimate limitations.

5.1 SCALING MODELS AND SCALING FACTORS

The most commonly used models are the constant electric field scaling model and the constant voltage scaling model. They both present a simplified view, taking only first degree effects into consideration, but are easily understood and well suited to educational needs. Recently, a combined voltage and dimension scaling model has been presented (Bergmann, 1991).

In this chapter, the application of each of the three models will be illustrated. To assist in visualization, it is useful to refer to Figure 5.1 which indicates the device dimensions and substrate doping level which are associated with the scaling of a transistor.

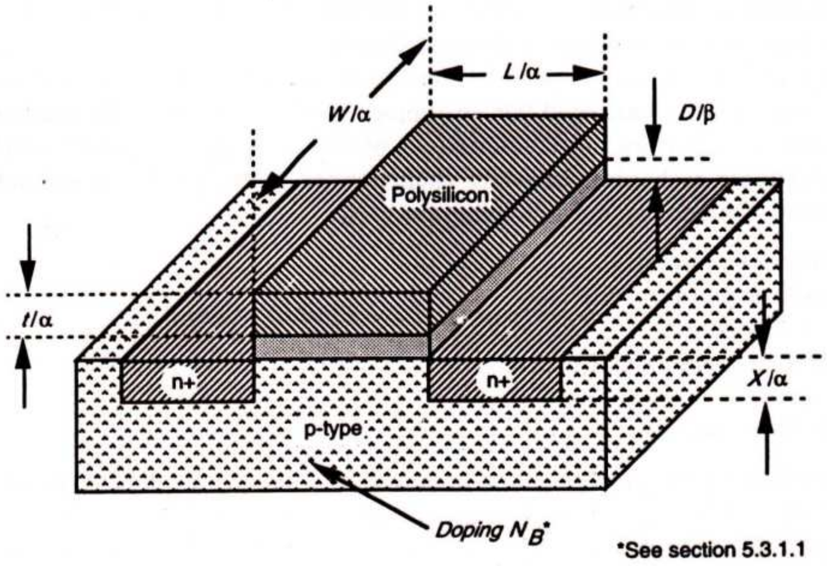


FIGURE 5.1 Scaled nMOS transistor (pMOS similar).

In order to accommodate the three models, two scaling factors—$1/\alpha$ and $1/\beta$—are used. $1/\beta$ is chosen as the scaling factor for supply voltage V_{DD} and gate oxide thickness D, and $1/\alpha$ is used for all other linear dimensions, both vertical and horizontal to the chip surface. For the constant field model and the constant voltage model, $\beta = \alpha$ and $\beta = 1$ respectively are applied.

5.2 SCALING FACTORS FOR DEVICE PARAMETERS

In this section, simple derivations and calculations reveal the effects of scaling.

5.2.1 Gate Area A_g

$$A_g = L.W.$$

where L and W are the channel length and width respectively. Both are scaled by $1/\alpha$.

Thus A_g is scaled by $1/\alpha^2$

5.2.2 Gate Capacitance Per Unit Area C_0 or C_{ox}

$$C_0 = \frac{\varepsilon_{ox}}{D}$$

where ε_{ox} is the permittivity of the gate oxide (thinox) [$= \varepsilon_{ins.}\varepsilon_0$] and D is the gate oxide thickness which is scaled by $1/\beta$

Thus C_0 is scaled by $\dfrac{1}{1/\beta} = \beta$

5.2.3 Gate Capacitance C_g

$$C_g = C_0 L.W.$$

Thus C_g is scaled by $\beta\dfrac{1}{\alpha^2} = \dfrac{\beta}{\alpha^2}$

5.2.4 Parasitic Capacitance C_x

C_x is proportional to $\dfrac{A_x}{d}$

where d is the depletion width around source or drain which is scaled by $1/\alpha$, and A_x is the area of the depletion region around source or drain which is scaled by $1/\alpha^2$.

Thus C_x is scaled by $\frac{1}{\alpha^2} \cdot \frac{1}{1/\alpha} = \frac{1}{\alpha}$

5.2.5 Carrier Density in Channel Q_{on}

$$Q_{on} = C_0 \cdot V_{gs}$$

where Q_{on} is the average charge per unit area in the channel in the 'on' state. Note that C_0 is scaled by β and V_{gs} is scaled by 1/β.

Thus Q_{on} is scaled by 1

5.2.6 Channel Resistance R_{on}

$$R_{on} = \frac{L}{W} \frac{1}{Q_{on}\mu}$$

where μ is the carrier mobility in the channel and is assumed constant.

Thus R_{on} is scaled by $\frac{1}{\alpha} \frac{1}{1/\alpha} 1 = 1$

5.2.7 Gate Delay T_d

T_d is proportional to $R_{on} \cdot C_g$

Thus T_d is scaled by $\frac{1.\beta}{\alpha^2} \frac{\beta}{\alpha^2}$

5.2.8 Maximum Operating Frequency f_0

$$f_0 = \frac{W}{L} \frac{\mu C_0 V_{DD}}{C_g}$$

or, f_0 is inversely proportional to delay T_d.

Thus f_0 is scaled by $\frac{1}{\beta/\alpha^2} = \frac{\alpha^2}{\beta}$

5.2.9 Saturation Current I_{dss}

$$I_{dss} = \frac{C_0\mu}{2} \frac{W}{L} (V_{gs} - V_t)^2$$

noting that both V_{gs} and V_t are scaled by $1/\beta$, we have

$$I_{dss} \text{ is scaled by } \beta(1/\beta)^2 = 1/\beta$$

5.2.10 Current Density J

$$J = \frac{I_{dss}}{A}$$

where A is the cross-sectional area of the channel in the 'on' state which is scaled by $1/\alpha^2$

So, J is scaled by $\dfrac{1/\beta}{1/\alpha^2} = \dfrac{\alpha^2}{\beta}$

5.2.11 Switching Energy Per Gate E_g

$$E_g = \frac{1C_g}{2}(V_{DD})^2$$

So, E_g is scaled by $\dfrac{\beta}{\alpha^2} \cdot \dfrac{1}{\beta^2} = \dfrac{1}{\alpha^2\beta}$

5.2.12 Power Dissipation Per Gate P_g

P_g comprises two components such that

$$P_g = P_{gs} + P_{gd}$$

where the static component

$$P_{gs} = \frac{(V_{DD})^2}{R_{on}}$$

and the dynamic component

$$P_{gd} = E_g f_0$$

It will be seen that both P_{gs} and P_{gd} are scaled by $1/\beta^2$

So, P_g is scaled by $1/\beta^2$

5.2.13 Power Dissipation Per Unit Area P_a

$$P_a = \frac{P_g}{A_g}$$

So, P_a is scaled by $\dfrac{1/\beta^2}{1/\alpha^2} = \alpha^2/\beta^2$

5.2.14 Power-speed Product P_T

$$P_T = P_g \,.\, T_d$$

So, P_T is scaled by $\dfrac{1}{\beta^2} \cdot \dfrac{\beta}{\alpha^2} = \dfrac{1}{\alpha^2\beta}$

5.2.15 Summary of Scaling Effects

It is useful to summarize the scaling effects in a convenient form. Table 5.1 sets out scaling effect for various key parameters of MOS FET devices and for the three scaling models mentioned earlier.

TABLE 5.1 Scaling effects

Parameters		*Combined V and D*	*Constant E*	*Constant V*
V_{DD}	Supply voltage	$1/\beta$	$1/\alpha$	1
L	Channel length	$1/\alpha$	$1/\alpha$	$1/\alpha$
W	Channel width	$1/\alpha$	$1/\alpha$	$1/\alpha$
D	Gate oxide thickness	$1/\beta$	$1/\alpha$	1
A_g	Gate area	$1/\alpha^2$	$1/\alpha^2$	$1/\alpha^2$
C_0(or C_{ox})	Gate C per unit area	β	α	1
C_g	Gate capacitance	β/α^2	$1/\alpha$	$1/\alpha^2$
C_x	Parasitic capacitance	$1/\alpha$	$1/\alpha$	$1/\alpha$
Q_{on}	Carrier density	1	1	1
R_{on}	Channel resistance	1	1	1
I_{dss}	Saturation current	$1/\beta$	$1/\alpha$	1
A_c	Conductor X-section area	$1/\alpha^2$	$1/\alpha^2$	$1/\alpha^2$
I	Current density	α^2/β	α	α^2
V_g	Logic 1 level	$1/\beta$	$1/\alpha$	1
E_g	Switching energy	$1/\alpha^2.\beta$	$1/\alpha^3$	$1/\alpha^2$
P_g	Power dispn per gate	$1/\beta^2$	$1/\alpha^2$	1
N	Gates per unit area	α^2	α^2	α^2
P_a	Power dispn per unit area	α^2/β^2	1	α^2
T_d	Gate delay	β/α^2	$1/\alpha$	$1/\alpha^2$
f_0	Max. operating frequency	α^2/β	α	α^2
P_T	Power-speed product	$1/\alpha^2.\beta$	$1/\alpha^3$	$1/\alpha^2$

Constant E:$\beta = \alpha$; Constant V:$\beta = 1$

5.3 SOME DISCUSSION ON AND LIMITATIONS OF SCALING

Although scaling down does have many desirable effects, some of the associated effects may cause problems which eventually become severe enough to prevent further miniaturization.

5.3.1 Substrate Doping

So far, in discussing the various effects, we have neglected the built-in (junction) potential V_B, which in turn depends on the substrate doping level, and this is acceptable so long as V_B is small compared with V_{DD}. However, when this no longer holds, then the effects of V_B must be included.

Furthermore, substrate doping impinges on many of the characteristics of transistors fabricated on it. Thus further discussion is warranted.

5.3.1.1 Substrate doping scaling factors

As the channel length of a MOS transistor is reduced, the depletion region widths must also be scaled down to prevent the source and drain depletion regions from meeting. Depletion region width d for the junctions is given by

$$d = \sqrt{\frac{2\varepsilon_{si}\varepsilon_0 V}{qN_B}}$$

where

ε_{si} = relative permittivity of silicon ($\div 12$)
ε_0 = permittivity of free space (= 8.85×10^{-14} F/cm)
V = effective voltage across the junction = $V_a + V_B$
q = electron charge
N_B = doping level of substrate
V_a (maximum value = V_{DD}) = applied voltage
V_B = built-in (junction) potential

and

$$V_B = \frac{kT}{q}\ln\left(\frac{N_B N_D}{n_i^2}\right)$$

where N_D is the source or drain doping, and n_i is the intrinsic carrier concentration in silicon.

In, say, 5 μm technology, V_B is in the region of 500 mV whilst applied voltage $V_a(= V_{DD})$ is commonly 5 V so that V_B may be neglected for scaling considerations. Under these circumstances,

$$d = \sqrt{\frac{2\varepsilon_{si}\varepsilon_0 V_{DD}}{qN_B}}$$

If V_{DD} is scaled by $1/\beta$ and d by $1/\alpha$, then N_B can be scaled by α^2/β (Bergmann, 1991).

For some more recent technologies, N_B is increased to reduce d so that V_B is also enlarged. (For example, if $N_B = 10^{15}$ cm^{-3} and $N_D = 10^{20}$ cm^{-3} then $V_B = 0.88$ V). At the same time, V_{DD} is also scaled down, and is thus no longer large compared with V_B so that V_B must be taken account of in scaling.

Thus, for the combined voltage and dimension scaling model applied to a transistor for which we have a known V_a, we may write

$$V_a = mV_B$$

where m is a real number, so that

$$V = V_a + V_B = mV_B + V_B$$

Now if we scale V_a by $1/\beta$ we have

$$V_s = \frac{mV_B}{\beta} + V_B \text{ so that scaling factor } = \frac{\beta + m}{\beta(m+1)}$$

where V_s is the effective scaled voltage across the depletion region. Consequently, N_B should be scaled by

$$\frac{\alpha^2(\beta + m)}{(m+1)}$$

so that d scales by $1/\alpha$.

This model not only expresses the effects of the relationship between V_a and V_B, but also shows their relation to the scaling factor β. Where m is large and β is small, the scaling factor for N_B reverts to α^2/β, but in other cases this model becomes significant,

5.3.1.2 Depletion width

In the previous discussion, N_B is increased to reduce the depletion width, but this also increases the threshold voltage V_t which is against the required trends for scaling down.

In [Hoen] N_B must be kept below 1.3×10^{19} cm^{-3}. At higher values of N_B, the maximum electric field which can be applied to the gate oxide is insufficient to invert the substrate so that no channel can be formed.

However, the technology of deep channel implantation increases N only near the source and drain to substrate junctions. Thus, N_B can be maintained at a satisfactory level in the channel region and this problem is thus reduced. Nonetheless, depletion width d and built-in potential V_B will impose limitations on scaling,

It can be shown (Grove, 1967) that

$$E_{max} = \frac{2V}{d}$$

where E_{max} is the maximum electric field induced in the one-sided step junction.

When N_B is increased by α and if $V_a = 0$, then V_B is increased by $\ln \alpha$ and d is decreased by

$$\sqrt{\frac{\ln \alpha}{\alpha}}$$

Therefore, the electric field E across the depletion region is increased by $\sqrt{\alpha/\ln \alpha}$ and will thus reach the critical level E_{crit} with increasing N_B.

Figure 5.2(a) shows the depletion width d as a function of substrate concentration N_B and supply voltage V_{DD}. The dashed line indicates the maximum depletion width for $E_{max} = E_{crit}$. Substituting into the equation for d, we have

$$d = \sqrt{\frac{2\varepsilon_{si}\varepsilon_0}{qN_B}\left(\frac{E_{crit}.d}{2}\right)}$$

whence

$$d = \frac{\varepsilon_{si}\varepsilon_0(E_{crit})}{qN_B}$$

The area of Figure 5.2(a) above the dashed line is the region where the increased electric field will induce breakdown. Thus, the point at which the dashed line and the $V_a = 0$ line intersect indicates the maximum allowable substrate doping level, which is about $N_B = 3 \times 10^{17}$ cm^{-3} (for $N_D = 3 \times 10^{20}$ cm^{-3}). At higher values of N_B junction tunneling will occur. Therefore allowable values for d fall below the dashed line and above the $V_a = 0$ line.

Figure 5.2(b) shows the maximum electric field in the depletion layer versus N_B. Any applied voltage greater than $V_a = 0$ will cause breakdown to occur at lower values of N_B.

In the foregoing discussions, the effects of N_D have been assumed to be negligible.

5.3.2 Limits of Miniaturization

The minimum size of a transistor is determined by both process technology and the physics of the device itself. The reduction of device geometry currently depends mainly on alignment accuracy and on the resolution of photolithographic technology; the limit on feature size is now at 0.3 µm, but the increasing availability of direct write E-beam technology will allow this limit to be further reduced.

The size of a transistor is usually defined in terms of its channel length L. As the channel length is scaled down, the edge of the depletion region around the source comes closer to that around the drain. In order to prevent punch-through and maintain transistor action, it can be shown that the channel length L must be at least $2d$. Therefore, L is in turn determined by the substrate concentration N_B and supply voltage V_{DD} (which determines V_a).

Applying the conclusions from the previous section, we may estimate the minimum possible channel length as 0.14 µm. The minimum transit time for an electron to travel from source to drain can also be calculated. From (Sze, 1985),

$$v_{drift} = \mu E$$

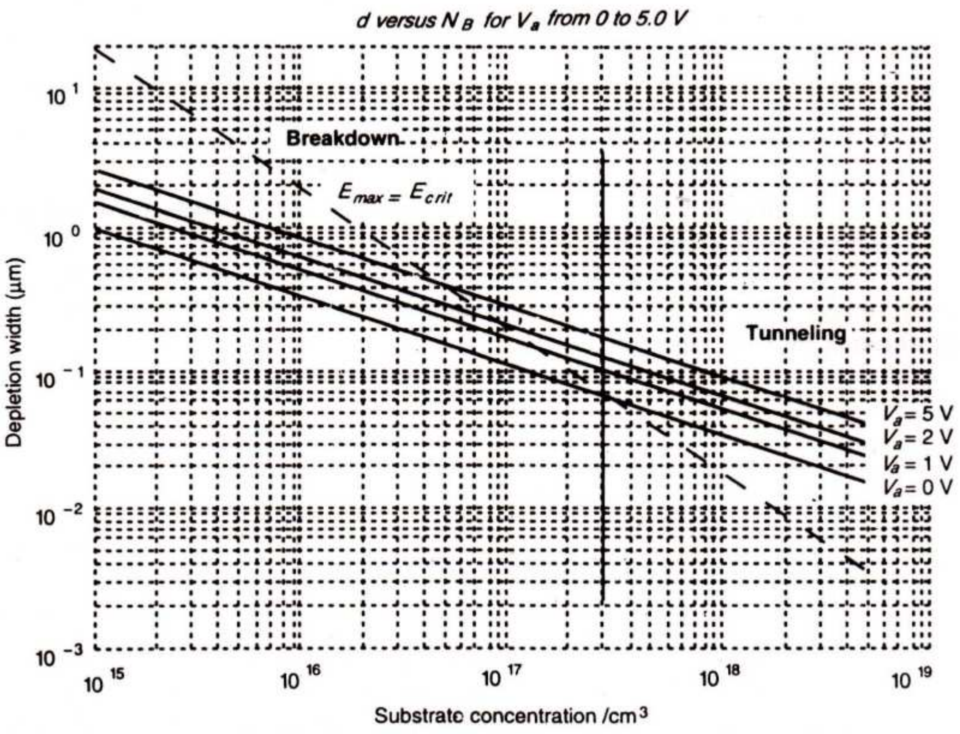


FIGURE 5.2(a) Substrate concentration/cm^3.

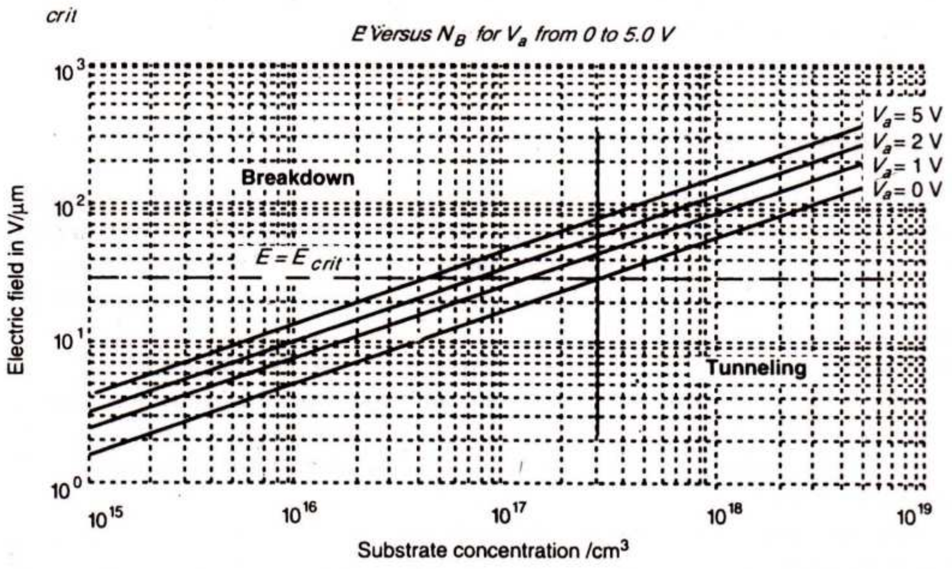


FIGURE 5.2(b) Depletion width *d* and electric field *E* versus substrate doping N_B.

where v_{drift} is the carrier drift velocity, and

$$L = 2d$$

so that transit time τ is given by

$$\tau = \frac{L}{v_{drift}} = \frac{2d}{\mu E}$$

The maximum carrier drift velocity is approximately equal to v_{sat} where saturation velocity $v_{sat} = 1 \times 10^7$ cm/sec (Sze, 1985), regardless of the supply voltage. Therefore the minimum transit time may be assumed to occur for a minimum size transistor when V_a is approximately 0 V. Transit times may be assessed from Figure 5.3. Note that Figure 5.3(a) assumes a transistor of size $L = 2d$ with zero space between source and drain depletion regions.

5.3.3 Limits of Interconnect and Contact Resistance

Since the width, thickness and spacing of interconnects are each scaled by $1/\alpha$, cross-section areas must scale by $1/\alpha^2$. Thus, for short distance interconnections the conductor length is also scaled by $1/\alpha$, so that resistance is increased by α. For constant field scaling, current I is also scaled by $1/\alpha$ so that IR drop remains constant as a device is scaled, and thus represents a higher proportion of the supply voltage V_{DD} which is also scaled by $1/\alpha$. Thus driving capability and noise margins are degraded.

With decreasing device dimensions, we are also seeing further increases in the levels of integration and consequent increases in die size. This lengthens the interconnections from one side of the chip to the other and, therefore, both resistance and capacitance of the

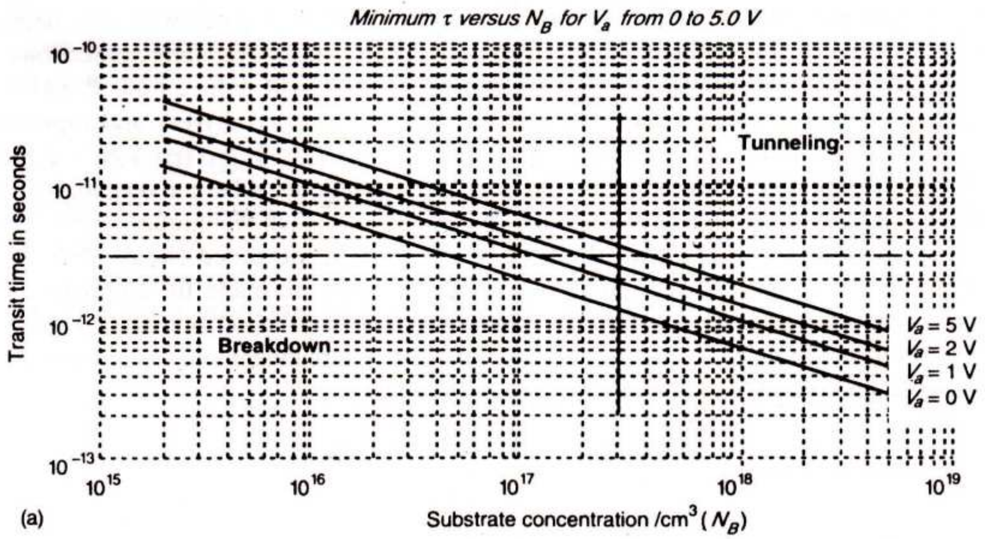


FIGURE 5.3(a) Transit time τ versus substrate concentration/cm³.

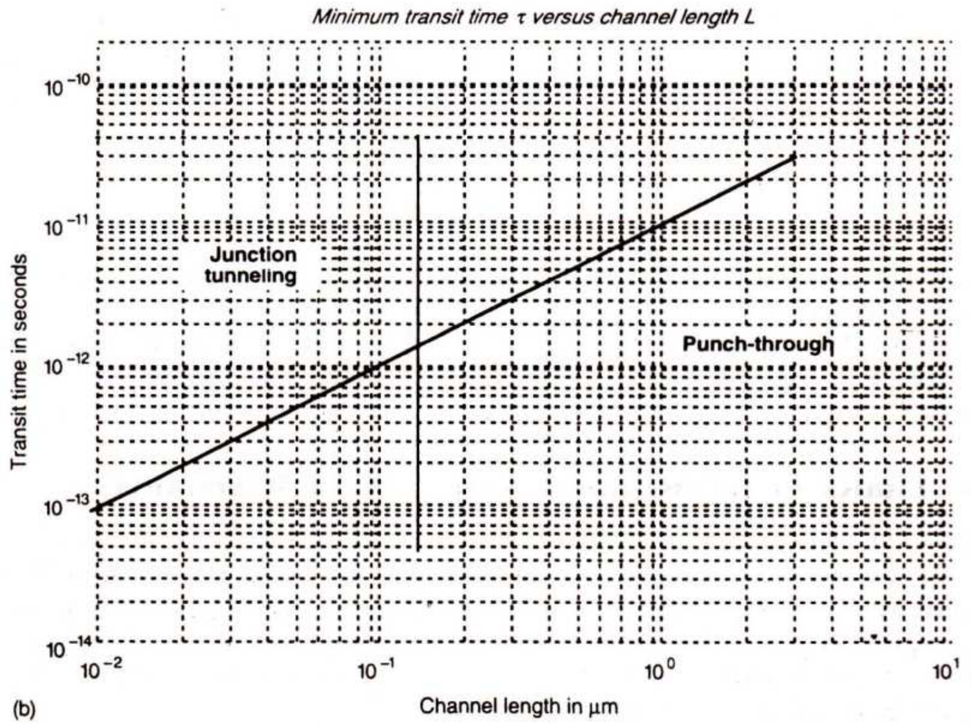


FIGURE 5.3(b) Transit time τ versus *L*.

interconnects are increased, producing much larger time constant values. Thus the effects of increased propagation delays, signal decay, and clock skew will decrease maximum achievable operating frequency, even though the smaller transistors produce gates with less delay.

One solution to this problem has been to make use of multilayer interconnections with thicker, wider conductors and thicker separating layers. This will reduce both *R* and *C* and also reduce die size. Other measures include the use of cascaded drivers and repeaters to reduce the effects of long interconnects.

A further option is to use optical interconnection techniques where a very high level of integration is required for high speed circuits. In order to use such techniques, optical fibers, laser diodes, receivers, and amplifiers must be included in the integrated circuit. Performance will vary with the materials used, but rough estimations can be made for comparison with metal interconnects. To start our considerations, a model may be set out as in Figure 5.4.

The propagation delay T_p along a single aluminum interconnect can be calculated from the following approximate equation (Sakurai, 1983):

$$T_p = R_{int}C_{int} + 2.3\ (R_{on}C_{int} + R_{on}C_L + R_{int}C_L)$$

whence

$$T_p \doteqdot (2.3\ R_{on} + R_{int})C_{int}$$

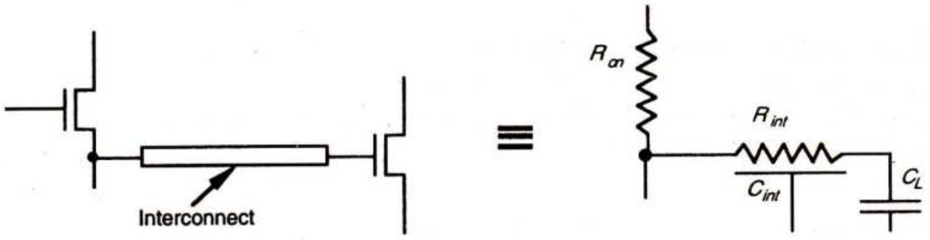


FIGURE 5.4 Model of metal interconnect.

Now

$$R_{int} = \rho \frac{L}{HW}$$

$$C_{int} = \varepsilon_{ox}\ [1.15.W/t_{ox} + 2.28\ (H/t_{ox})^{0.222}]L$$

where

R_{on} is the ON resistance of the transistor
R_{int} is the resistance of the interconnect
C_{int} is the capacitance of the interconnect
t_{ox} is the thickness of the dielectric oxide
ρ is the resistivity of the interconnect
L, W, H are the length, width and height (thickness) of the interconnect

$$\varepsilon_{ox} = 3.4515 \times 10^{-5}\ \text{pF}/\mu\text{m—the permittivity of SiO}_2$$

If we use a value of ρ = 3 μΩcm for aluminum (Bakoglu and Meindl, 1985), and if we choose t_{ox} = 0.8 μm for thick oxide with interconnect L = 1 cm, W = 3 μm and H = 1 μm, we then have the propagation delay T_p given by

$$T_p = (2.3 \times 5\ \text{k}\Omega + 0.1\ \text{k}\Omega)2.5\ \text{pF} = 29\ \text{nsec}$$

Optical fibers can be used to replace metal interconnects in critical applications, and Figure 5.5 shows this in schematic form. R_{int} and C_{int} may be assumed to be zero, and the time needed for the output driver to transfer a logic state is given by

$$T_p = 2.3\ R_{on}C_L + t_{laser} + t_{int} + t_{rec}$$

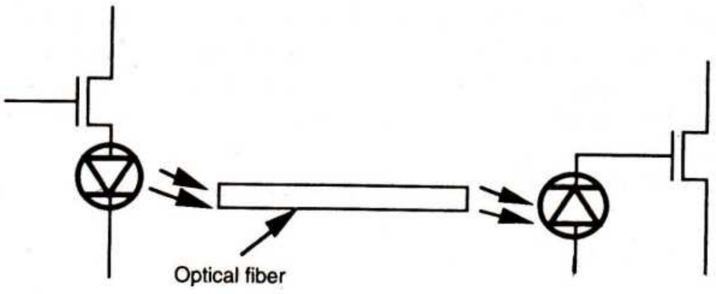


FIGURE 5.5 Electro-optical interconnection.

where

C_L is the input capacitance of the laser diode
t_{laser} is the delay time through the laser diode
t_{int} is the propagation delay along the optical fiber interconnnect
t_{rec} is the receiver delay time

and

$$t_{int} = \frac{nL}{c}$$

where

n is the refractive index for the optic fiber material
L is the length of the fiber
c is the free space speed of light ($c = 3 \times 10^8$ m/sec).

Since laser diodes and receivers can work at frequencies above 10 GHz, each of them presents a relatively short delay—typically around 100 psec. The capacitance of a discrete laser diode is about 1 pF (Hutcheson, 1987) and the refractive index of commonly used material for fiber optics is between 1.5 and 2.0.

Evaluating the propagation delay we have

$$T_p = 2.3 \times 5 \times 10^3 \times 1 \times 10^{-12} + 1 \times 10^{-10} + \frac{2 \times 10^{-4}}{3 \times 10^8} + 1 \times 10^{-10} = 11.7 \text{ nsec}$$

Delay time versus line length and width may be assessed from Figure 5.6. It is obvious that the longer the interconnect, the more speed advantage arises from the use of fiber optics. In considering delay time versus line width, it may be shown that R_{on} is the dominant factor for aluminum whilst R_{int} contributes the major component for poly.

The performance of laser diodes and receivers can be improved if they are formed as part of an integrated circuit. GaAs is a material which allows this integration since it can accommodate both electronic components and optical interconnections in the one chip.

5.4 LIMITS DUE TO SUBTHRESHOLD CURRENTS

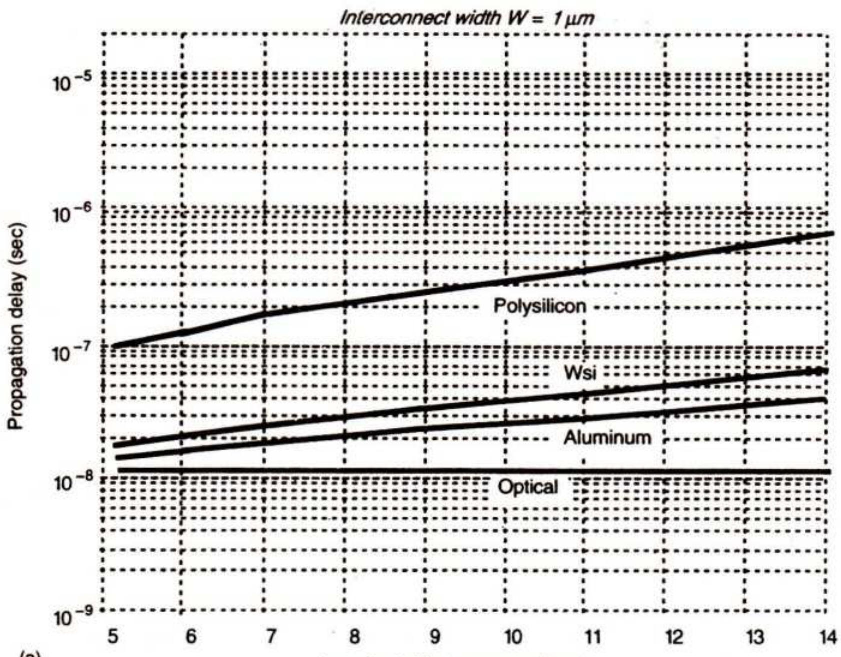
One of the major concerns in the scaling of devices is the effect on subthreshold current I_{sub} which is directly proportional to $\exp(V_{gs} - V_t)q/kT$:

When a transistor is in the off state, then the value of $V_{gs} - V_t$ is negative and should be as large as possible to minimize I_{sub}. As voltages are scaled down, the ratio of $V_{gs} - V_t$ to kT will reduce so that subthreshold current increases quite dramatically. For this reason it may be desirable to scale both V_{gs} and V_t together with V_{DD} by factor $1/b$ rather than $1/a$, since a is generally greater than b. However, this increases electric field strengths and thus lowers breakdown voltages.

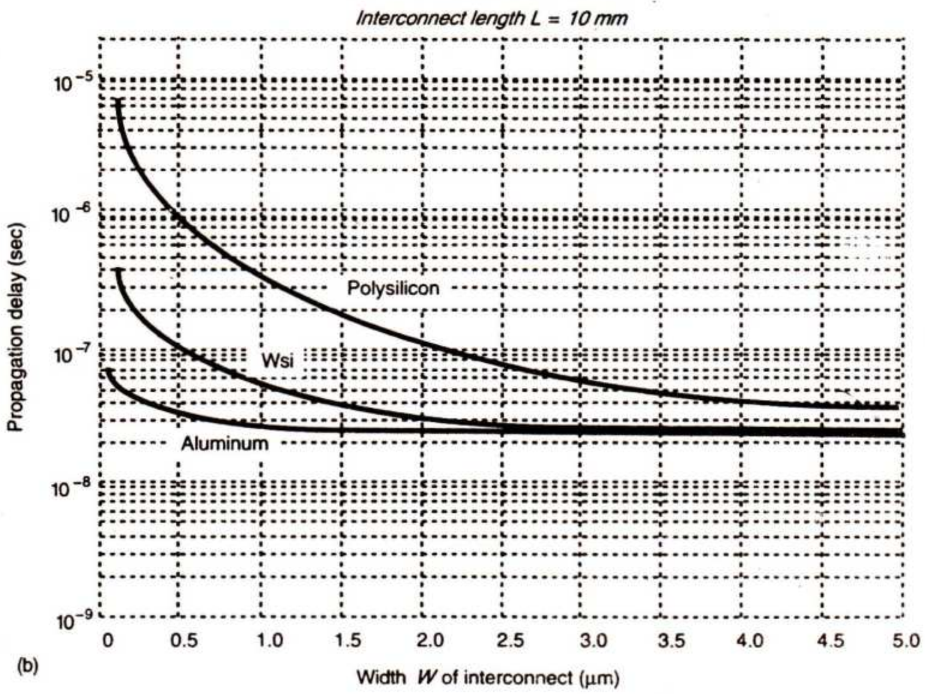
The maximum electric field across a depletion region is given by (Grove, 1967):

$$E_{max} = \frac{2(V_a - V_B)}{d}$$

This applies to a one-sided step junction.



(a)



(b)

FIGURE 5.6 Interconnect delay versus width and length.

R_{on} = 5 kΩ, $H = W/3$; $t_{ox} = W/3$; ρ_{Al} = 3 μΩcm; ρ_{wsi} = 30 μΩcm; ρ_{poly} = 500 μΩcm.

As discussed previously, $(V_a + V_B)$ is scaled by $(\beta + m)/\beta(m + 1)$ and d is scaled by $1/\alpha$. Therefore, E_{max} is scaled by $\alpha(\beta + m)/\beta(m + 1)$. Again, if α is greater than β, then more electric field stress will be applied across depletion regions of scaled-down transistors.

At the same time, the junction breakdown voltage BV must be considered. BV is given by (Grove, 1967):

$$BV = \frac{\varepsilon_{si}\varepsilon_0 (E_{crit})^2}{2qN_B}$$

It will be seen that BV is thus scaled by $\beta(m + 1)/\alpha^2 (\beta + m)$ and will decrease. Extra care is therefore required in estimating the breakdown voltage for scaled devices. It should be noted that electric fields are greater and BV is greater at the corners of diffusion regions underlying or abutting silicon dioxide.

5.5 LIMITS ON LOGIC LEVELS AND SUPPLY VOLTAGE DUE TO NOISE

Major advantages in the scaling of devices are smaller gate delay time, that is, higher operating frequencies and lower power dissipation. However, the decreased inter-feature spacing and greater switching speeds inevitably result in noise problems. Noise may also be amplified and is thus a major concern.

The mean square current fluctuation in the channel is given by

$$(i^2) = 4kTR_{n}g_{m}\Delta f$$

where R_n is the equivalent noise resistance at the input and Δf is the bandwidth.

F.M. Klaassen and J. Prins (1966) have investigated the thermal noise in a MOS transistor over a range of substrate doping levels N_B from 10^{14} to 10^{17} cm^{-3}. When a transistor works in saturation, g_m is no longer proportional to the gate voltage V_g, and can be expressed as

$$g_m \div BV_p$$

where

$$B = \frac{\mu W C_{ox}}{L}$$

and V_p is the pinch off voltage given by

$$V_p = V_g' - \frac{1}{2}\left(\frac{a}{C_{ox}}\right)^2 \left[\frac{(1 + 4V_g'C_{ox})^{1/2}}{a} - 1\right]$$

where

$$V_g' = V_g - V_t + V_B$$

$$a = (2\varepsilon_{si}q_{NB})^{1/2}$$

V_B is the junction (built-in) potential.

Then, the equivalent noise resistance R_n is given by

$$R_n = \left(\frac{1V'_g}{2V'_p} + \frac{1}{6}\right) g_m^{-1}$$

where

$$V'_p = V_p + V_B$$

Since V_p is a monotonically decreasing function of the gate oxide thickness t_{ox} and substrate doping N_B, R_n is also a monotonically decreasing function of the same parameters.

Consequently, the main factor of the thermal noise $R_n g_m$ is given by

$$R_n g_m = \frac{1}{2}\left(\frac{V_g - V_t + V_B}{V_p + V_B}\right) + \frac{1}{6}$$

This indicates that $R_n g_m$ is strongly and directly dependent on t_{ox} and N_B and also, to a lesser extent, on V_g. Experimental results, as in Figure 5.7, support this theory (Klaassen and Prins, 1966).

Considering current technology in which t_{ox} is scaled, the effect of N_B is smaller (Sah, Wu and Hielscher, 1966), and V'_p and V'_g are replaced by V_p and V_g respectively. Thus, the expression for $R_n g_m$ becomes

$$R_n g_m = \frac{1}{2}\left[\frac{V_g}{V_g - \frac{1}{2}\left(\frac{a}{C_{ox}}\right)^2 \left(\frac{1 + 4V'_g C_{ox}}{a}\right)^{1/2} - 1}\right] + \frac{1}{6}$$

When constant field scaling is applied, V_g is scaled by $1/\alpha$, C_{ox} and N_B are scaled by α. Consequently, $R_n g_m$ is only slightly reduced owing to the increased value of C_{ox}. Thus, the ratio of logic level to thermal noise is degraded by almost the same factor.

Flicker noise has been the subject of many studies since A.L. McWhorter introduced his model in 1956. The noise was observed and explained as the result of fluctuations of carriers trapped in the channel by surface states.

As a conclusion of the investigations by F.M. Klaassen, the change in the number of trapped carriers dn_t due to the change in the number of induced free carriers d_n presents a current fluctuation Δi at the output, such that

$$\Delta i^2 \approx \frac{q\mu s I V_d}{L^2 f}$$

where

$s = dn_t/d_n$—the surface state efficiency

I = the DC drain current
f = the frequency
V_d = the applied drain voltage.

Usually the output noise is represented by an equivalent noise voltage source ΔV at the input (Klaassen, 1971), such that

$$\Delta V^2 = \frac{qs(V_d - 0.5V_d)}{C_g f}$$

When the transistor operates in saturation, then $V_d \div V_g$, so that

$$\Delta V^2 = \frac{1}{2}\frac{qsV_g}{C_g f}$$

where
V_g = the effective applied gate voltage
C_g = the gate capacitance.

Since s is a process dependent factor, the flicker noise is scaled by one for constant field scaling or by α^2/β^2 for the combined scaling model.

In addition to noise sources already considered, other noise inputs are due to mutual inductive and mutual capacitive coupling, and these alone could impose practical limitations on the lowest usable operating voltages.

Considering the cross-talk between two parallel signal lines on a chip, the coupling model presented (Watts, 1989) shows that capacitive noise is proportional to $C.dV/dt$, where C is the inter-line capacitance, and dV/dt is approximately equal to V_a/t_r, where t_r is the rise time of the coupled signal. The inductive noise is related to LdI/dt, where L is the mutual inductance and $dI/dt \approx I_{sat}/t_r$. Therefore, cross-talk noise increases as operating frequency increases and t_r is reduced.

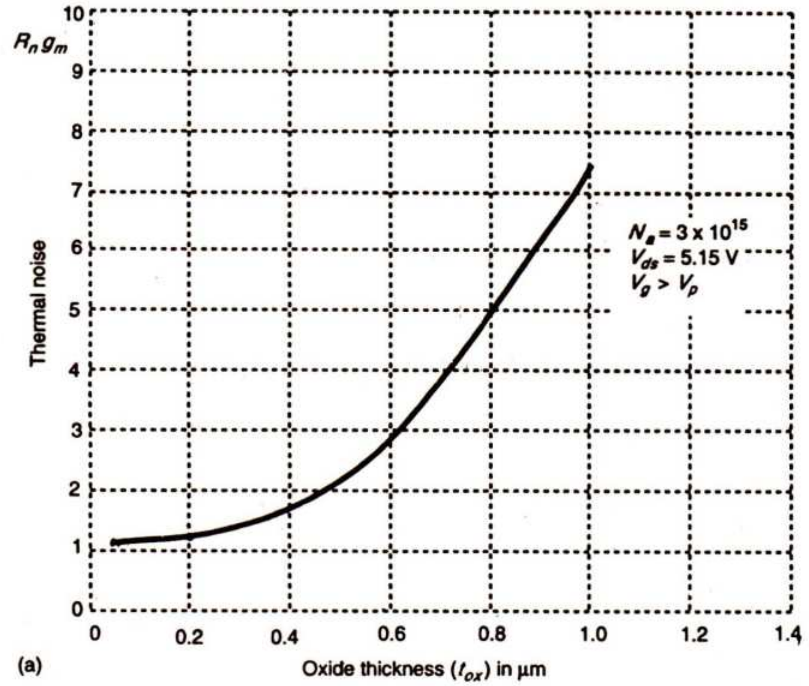
There are also other noise sources due to external influences, such as radio frequency signals, voltage spikes or voltage drops on power lines or ground connections, unterminated signal lines and lines with non-uniform impedance characteristics.

A typical peak to peak noise of at least 100 mV may be observed on the power and ground lines of even well-designed multilayer PC boards (Long and Butner, 1989).

Scaling down exacerbates the effect of both internally and externally generated noise and this degrades both the production yield and the reliability of high density chip layouts.

In order to assess the effects of noise on the probability of failure P_F in a circuit, we may consider a situation where, as in Figure 5.7, the minimum signal to noise ratio (SNR) for satisfactory operation is assumed to be four, and the mean noise is assumed to be zero with a standard deviation $\Sigma = 100$ mV. P_F may be estimated by using the Gaussian distribution

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty \left(\frac{e^{-\mu^2/2}}{2}\right) du$$



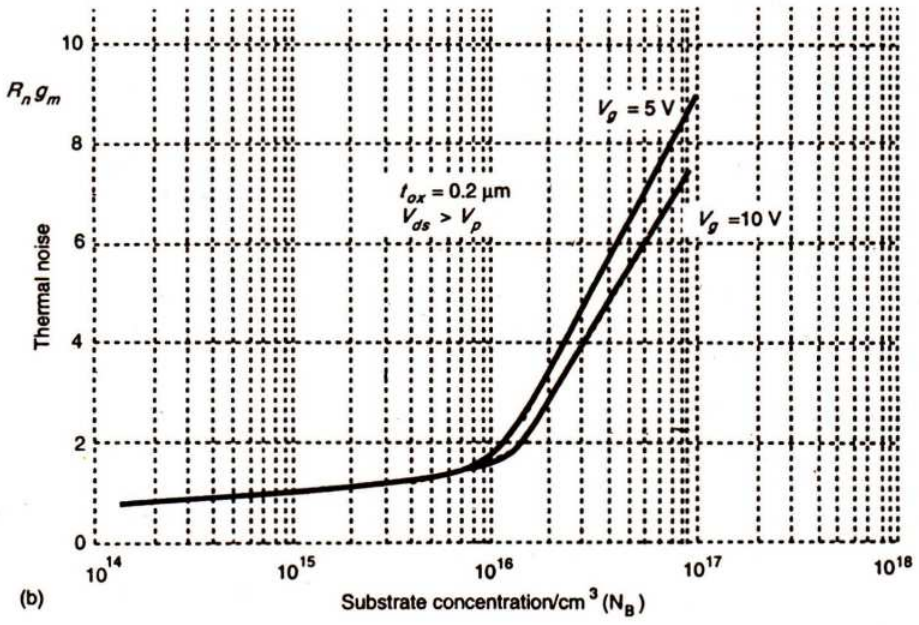


FIGURE 5.7 Thermal noise versus oxide thickness (a) and substrate doping (b).

The P_F values for a range of supply rail voltages and for the conditions specified are derived by integrating the appropriate area at one end of the Gaussian curve and are shown in Figure 5.8.

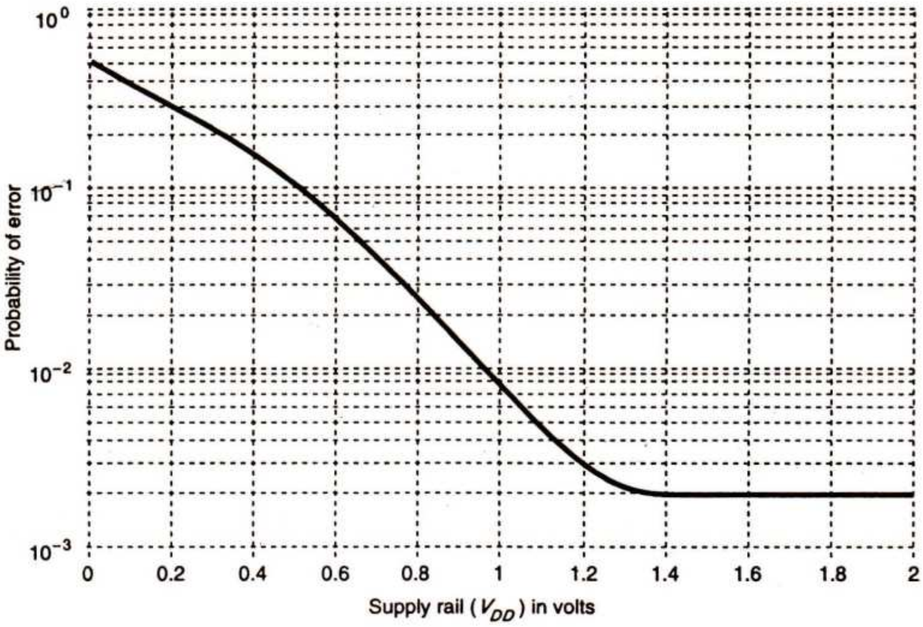


FIGURE 5.8 Probability of error versus supply voltage.

5.6 LIMITS DUE TO CURRENT DENSITY

High purity aluminum seems the most attractive, and is thus the most widely used, material for forming interconnections in VLSI chips. However, the scaling down of dimensions also increases the current density in interconnects by the same factor if constant field scaling is applied. When the current density in aluminum approaches 10^6 Amps/cm^2 (10 mA/μm^2), the interconnects are likely to be burned off owing to metal migration. Thus, allowable current densities are set well below this limit and figures of J = 1 to 2 mA/μm^2 are commonly used.

5.7 OBSERVATIONS

Scaling has not only been developed theoretically, but has also been widely applied in fabrication facilities as equipment improves. This has provided a direct and simple way of making smaller, faster chips. Current designs are often scaled down by 10% to 20% in linear dimensions as a way of providing better performance whilst faster smaller chip designs are completed.

The contributions made by scaling are significant. In particular, power dissipations are reduced, switching speeds are increased and chip size is reduced, which in turn improves production costs. However, the continual scaling down of dimensions is now pushing the technology toward both technological and ultimately, real physical limits.

Recent history has demonstrated that a new generation of microelectronic products emerges about every four years and that the device dimensions are scaled down by about 20% for each new generation. If this pace is maintained, then the ultimate physical limits on device size for silicon chips will be reached within the next decade. For further progress in the direction of high speed circuits, one must look to the development of alternative materials such as gallium arsenide (GaAS) and to super conductors.

5.8 REFERENCES

Bakoglu, H.B., and Meindl, J.D. (1985, May) 'Optimal interconnection circuits for VLSI', *IEEE Transactions on Electronic Devices*, Vol. ED-32, No.5.

Bergmann, N.W. (1991) 'A combined voltage and dimension scaling model for VLSI circuits', Proceedings of Microelectronics Conference, Melbourne, 24–25 June 1991, 101–4.

Grove, A.S. (1967) *Physics and Technology of Semiconductor Devices*, John Wiley and Sons Inc., New York.

Hutcheson, L.D. (1987) *Integrated Optical Circuits and Components: Design and Applications*, Marcel Dekker, New York.

Klaassen, F.M., and Prins, J. (1966) 'Thermal noise in MOS Transistors', *Philips Research Report*, Vol. 22.

Klaassen, F.M. (1971) 'Characterisation of Low l/*f* Noise in MOS Transistors', *IEEE Transactions on Electronic Devices*, Vol. ED-18, No. 10, 887–91.

Long, S.I., and Butner, S.E. (1989), *Gallium Arsenide Digital Integrated Circuit Design*, McGraw-Hill, USA.

Meindl, J.D. (1986) 'Interconnection limits on ultra large scale integration', *Proceedings VLSI '85*, North Holland, 13–19.

Sah, C.T., Wu, S.Y., and Hielscher, F.H. (1966) 'The effects of fixed bulk charge on the thermal noise in metal-oxide-semiconductor transistors', *IEEE Transactions on Electronic Devices*, Vol. ED-13, 410–14.

Sakurai, T. (1983, August) 'Approximation of wiring delay in MOSFET LSI', *IEEE Journal of Solid State Circuits*, Vol. SC-18, 418–26.

Sze, S.M. (1985) *Semiconductor Devices: Physics and Technology*, Bell Telephone Laboratories, USA.

Watts, R.K. (1989) *Submicron Integrated Circuits*, John Wiley and Sons Inc., New York.

Chapter 7
Subsystem Design Processes

One of the pleasantest things in the world is going on a journey.

— Sir John Harrington

The longest journey starts with a single step.

— Mao Zedong

OBJECTIVES

This chapter and the following two carry through the design of a digital system using a top-down approach. The complete system environment is that of a 4-bit microprocessor which is readily envisaged as an interconnection of four major architectural blocks—ALU, Control Unit, I/O Unit and Memory.

The design developed in this text is that of the ALU or data path, which itself divides readily into four subsystems. This chapter concentrates our attention on the design of one of the ALU subsystems—the Shifter.

The whole design process clearly illustrates the step-by-step nature of structured design and the inherently regular nature of properly conceived subsystem architecture. A general design process is developed and set out in this chapter.

7.1 SOME GENERAL CONSIDERATIONS

The first question to ask about any design methodology is the time-honored 'What's in it for me? Is it going to be worthwhile investing the time to learn?'. To answer the second part first, remarkably little time is needed to learn the rudiments of VLSI design. This is largely thanks to the Mead and Conway methodology which originally brought VLSI design within the scope of the ordinary electronics engineer. In fact, the average undergraduate student of electrical or electronic engineering can acquire a basic level of competence in VLSI design for an investment of about 40 hours of lectures spread over one or more academic terms or semesters. Similarly, a 10-day full-time continuing education course can quite readily bring practicing professional engineers or computer scientists up to a similar standard. A basic level of competence is taken as the ability to apply the design methodology and make use of design tools and procedures to the point where a chip design of several hundred transistors (or higher for regular structures) can be tackled.

The first part of the question—'What's in it for me?'—may be quite simply answered as: Providing better ways of tackling some problems, providing a way of designing and realizing systems that are too large, too complex, or just not suited to 'off-the-shelf' components and providing an appreciation and understanding of IC technology.

'Better' may include:

1. *Lower unit cost* compared with other approaches to the same requirement. Quantity plays a part here but even small quantities, if realized through cooperative ventures such as the multiproject chip (MPC) or multiproduct wafer (MPW), can be fabricated for as little as $200 (MPC) or $500 (MPW) per square millimetre of silicon, including the bonding and packaging of five or six chips per customer.
2. *Higher reliability*. High levels of system integration usually greatly reduce interconnections—a weak spot in any system.
3. *Lower power dissipation, lower weight, and lower volume* compared with most other approaches to a given system.
4. *Better performance*—particularly in terms of speed power product.
5. *Enhanced repeatability*. There are fewer processes to control if the whole system or a very large part of it is realized on a single chip.
6. *The possibility of reduced design/development periods* (particularly for more complex systems) if suitable design procedures and design aids are available.

7.1.1 Some Problems

Some of the problems associated with VLSI design are:

1. How to design large complex systems in a reasonable time and with reasonable effort. This is a problem shared with other approaches to system design.
2. The nature of architectures best suited to take full advantage of VLSI and the technology.
3. The testability of large/complex systems once implemented in silicon.

Problems 1 and 3 are greatly reduced if two aspects of standard practice are accepted:

- Approach the design in a top-down manner and with adequate computer-aided tools to do the job. Partition the system sensibly, aiming for simple interconnection between subsystems and high regularity within subsystems. Generate and then verify each section of the design.
- Design testability into the system from the outset and be prepared to devote a significant proportion (e.g. up to 30%) of the total chip area to test and diagnostic facilities.

These problems are the subject of considerable research and development activity at this time.

In tackling the design of a system, we must bear in mind that topological properties are generally far more significant than the logical operations being performed. It may be said that it is better to duplicate (or triplicate, etc.) rather than communicate. This is indeed the case, and it is an approach which seems wrong to more traditional designers. In fact, even in relatively straightforward designs, as much as 40–50% of the chip may be taken up with interconnections, and it is true to say that interconnections generally pose the most acute problems in the design of large systems. Communications must therefore be given the highest priority early in the design process and a *communications strategy* should be evolved and adhered to throughout that process.

Accordingly, the architecture should be carefully chosen to allow the design objectives to be realized *and* to allow high regularity in realization.

7.2 AN ILLUSTRATION OF DESIGN PROCESSES

- Structured design begins with the concept of hierarchy.
- It is possible to divide any complex function into less complex subfunctions. These may be subdivided further into even simpler subfunctions and so on—the bottom level being commonly referred to as 'leaf-cells'.
- This process is known as top-down design.
- As a system's complexity increases, its organization changes as different factors become relevant to its creation.
- Coupling can be used as a measure of how much submodules interact. Clever systems partitioning aims at reducing implicit complexity by minimizing the amount of interaction between subparts; thus independence of design becomes a reality.
- It is crucial that components interacting with high frequency be physically proximate, since one may pay severe penalties for long, high-bandwidth interconnects.
- Concurrency should be exploited—it is desirable that all gates on the chip do useful work most of the time.
- Because technology changes so fast, the adaptation to a new process must occur in a short time. Thus a technology-independent description becomes important.

In representing a design, several approaches may be used at different stages of the design process; for example:

- conventional circuit symbols;
- logic symbols;
- stick diagrams;
- any mixture of logic symbols and stick diagrams that is convenient at a particular stage;
- mask layouts;
- architectural block diagrams;
- floor plans.

We will illustrate various representations during the course of the following design exercise to illustrate design processes.

7.2.1 The General Arrangement of a 4-bit Arithmetic Processor

The 4-bit microprocessor has been chosen as a design example because it is particularly suitable for illustrating the design and interconnection of common architectural blocks.

Figure 7.1 sets out the basic architecture of most, if not all, microprocessors. At this stage we will consider the design of the data path only, but matters relevant to other blocks will follow in later chapters.

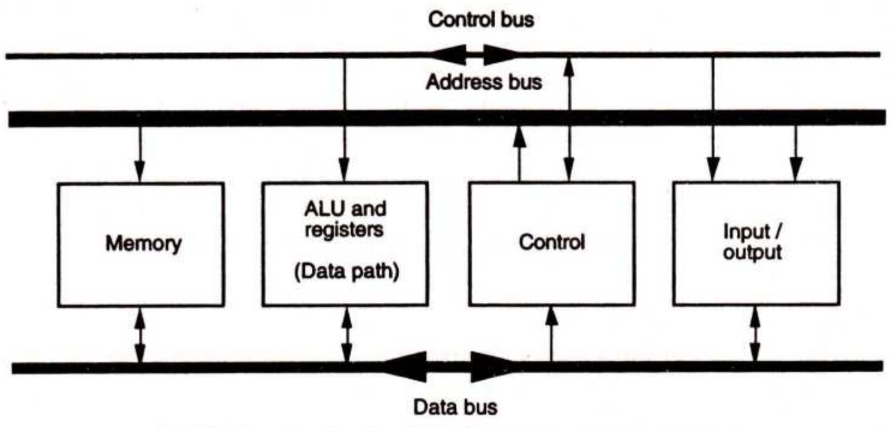


FIGURE 7.1 Basic digital processor structure.

The data path has been separated out in Figure 7.2 and it will be seen that the structure comprises a unit which processes data applied at one port and presents its output at a second port. Alternatively, the two data ports may be combined as a single bidirectional port if storage facilities exist in the data path. Control over the functions to be performed is effected by control signals as indicated.

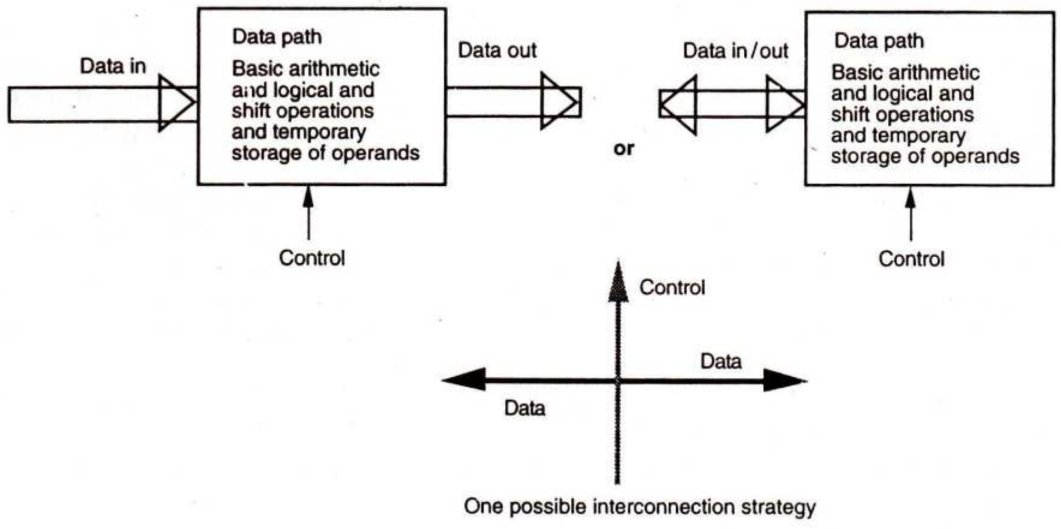

Data in
Data path
Basic arithmetic and logical and shift operations and temporary storage of operands
Data out
Control
or
Data in/out
Data path
Basic arithmetic and logical and shift operations and temporary storage of operands
Control
Control
Data
Data
One possible interconnection strategy

FIGURE 7.2 Communications strategy for data path.

At this early stage it is essential to evolve an interconnections strategy (as shown) to which we will then adhere.

Now we will decompose the data path into a block diagram showing the main subunits. In doing this it is useful to anticipate a possible *floor plan* to show the planned relative disposition of the subunits on the chip and thus on the mask layouts. A block diagram is presented in Figure 7.3.

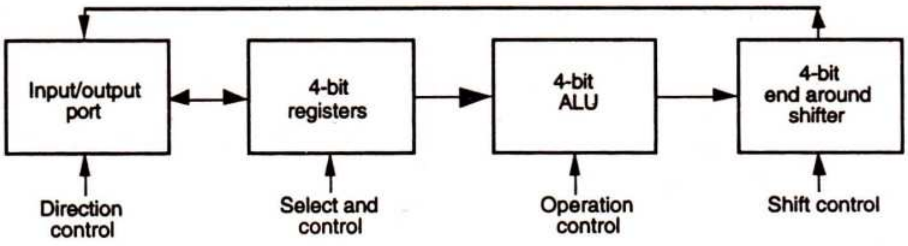


FIGURE 7.3 Subunits and basic interconnections for data path.

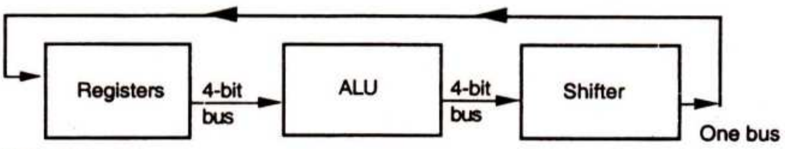
A further decision must then be made about the nature of the bus architecture linking the subunits. The choices in this case range from one-bus, two-bus or three-bus architecture. Some of the possibilities are shown in Figure 7.4.

In pursuing this particular design exercise, it was decided to implement the structure with a two-bus architecture. In our planning we can now extend on our interconnections strategy by planning for power rails and notionally making some basic allocation of layers on which the various signal paths will be predominantly run. These additional features are illustrated in Figure 7.5, together with a tentative floor plan of the proposed design which includes some form of interface (I/O) to the parent system data bus (see Figure 7.1).

The proposed processor will be seen to comprise a register array in which 4-bit numbers can be stored, either from an input/output port or from the output of the ALU via a shifter. Numbers from the register array can be fed in pairs to the ALU to be added (or subtracted, etc.) and the result can be shifted or not, before being returned to the register array or possibly out through the I/O port. Obviously, data connections between the I/O port, ALU, and shifter must be in the form of 4-bit buses. Simultaneously, we must recognize that each of the blocks must be suitably connected to control lines so that its function may be defined for any of a range of possible operations.

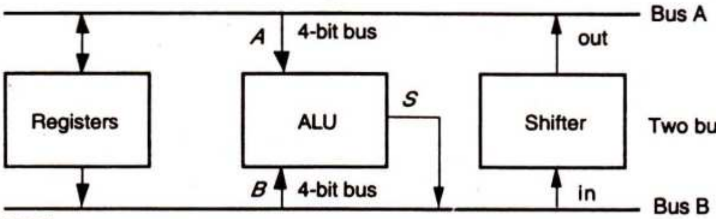
The required arrangement has been turned into a very tentative floor plan, as in Figure 7.5, which indicates a possible relative disposition of the blocks and also indicates an acceptable and sensible interconnection strategy indicated by the lines showing the preferred direction of data flow and control signal distribution. At this stage of learning, floor plans will be very tentative since we will not as yet be able to accurately assess the area requirements, say for a/4-bit register or a 4-bit adder.

Overall interconnection strategy having been determined, stick diagrams for the circuits comprising sections of the various blocks may be developed, conforming to the required strategy. An interactive process of modification may well then take place between the various stages as the design progresses. During the design process, and in particular when defining



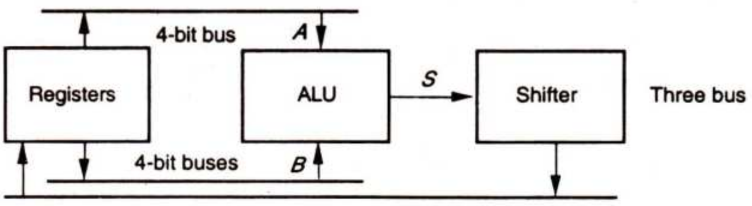
Sequence:

(i) First operand from registers to ALU. Operand is stored there.
(ii) Second operand from registers to ALU. Operands are added (etc.) and the result is, say, stored in the ALU.
(iii) The result is passed through shifter and stored in the registers.



Sequence:

(i) Two operands (*A* and *B*) are sent from register(s) to ALU and are operated upon and the result (*S*) is stored in ALU.
(ii) Result is passed through the shifter and stored in the registers.



Sequence:

The two operands (*A* and *B*) are sent from the registers, operated upon, and the shifted result (*S*) returned to another register *all in the same clock period.*

FIGURE 7.4 Basic bus architectures.

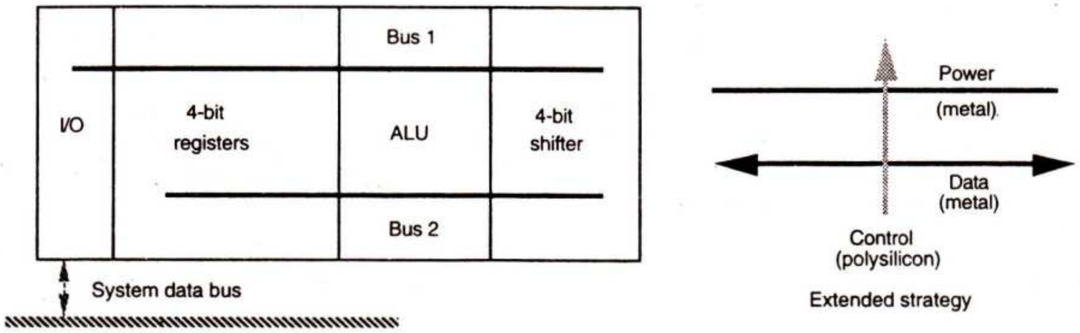


FIGURE 7.5 Tentative floor plan for 4-bit data path.

the interconnection strategy and designing the stick diagrams, care must be taken in allocating the layers to the various data or control paths. We must remember that:

1. Metal can cross polysilicon or diffusion without any significant effect (with some reservations to be discussed later).
2. Wherever polysilicon crosses diffusion a transistor will be formed. This includes the second polysilicon layer for processes that have two.
3. Wherever lines touch on the same level an interconnection is formed.
4. Simple contacts can be used to join diffusion or polysilicon to metal.
5. To join diffusion and polysilicon we must use either a buried contact or a butting contact (in which case all three layers are joined together at the contact) or two contacts, diffusion to metal then metal to polysilicon.
6. In some processes, a second metal layer is available. This can cross over any other layers and is conveniently employed for power rails.
7. First and second metal layers may be joined using a *via*.
8. Each layer has particular electrical properties which must be taken into account.
9. For CMOS layouts, p- and n-diffusion wires must not directly join each other, nor may they cross either a p-well or an n-well boundary.

With these factors in mind, we may now adopt suitable tactics to meet the strategic requirements when we approach the design of each subunit in turn.

7.2.2 The Design of a 4-bit Shifter

Any general purpose *n*-bit shifter should be able to shift incoming data by up to $n - 1$ places in a right-shift or left-shift direction. If we now further specify that all shifts should be on an 'end-around' basis, so that any bit shifted out at one end of a data word will be shifted in at the other end of the word, then the problem of right shift or left shift is greatly eased. In fact, a moment's consideration will reveal, for a 4-bit word, that a 1-bit shift right is equivalent to a 3-bit shift left and a 2-bit shift right is equivalent to a 2-bit shift left, etc. Thus we can achieve a capability to shift left or right by zero, one, two, or three places by designing a circuit which will shift right only (say) by zero, one, two, or three places.

The nature of the shifter having been decided on, its implementation must then be considered. Obviously, the first circuit which comes to mind is that of the shift register in Figures 6.38, 6.39 and 6.40. Data could be loaded from the output of the ALU and shifting effected; then the outputs of each stage of the shift register would provide the required parallel output to be returned to the register array (or elsewhere in the general case).

However, there is danger in accepting the obvious without question. Many designers, used to the constraints of TTL, MSI, and SSI logic, would be conditioned to think in terms of such standard arrangements. When designing VLSI systems, it pays to set out exactly what is required to assess the best approach. In this case, the shifter must have:

- input from a four-line parallel data bus;
- four output lines for the shifted data;
- means of transferring input data to output lines with any shift from zero to three bits inclusive.

In looking for a way of meeting these requirements, we should also attempt to take best advantage of the technology; for example, the availability of the switch-like MOS pass transistor and transmission gate.

We must also observe the strategy decided on earlier for the direction of data and control signal flow, and the approach adopted should make this feasible. Remember that the overall strategy in this case is for data to flow horizontally and control signals vertically.

A solution which meets these requirements emerges from the days of switch and relay contact based switching networks—the *crossbar switch*. Consider a direct MOS switch implementation of a 4 × 4 crossbar switch, as in Figure 7.6.

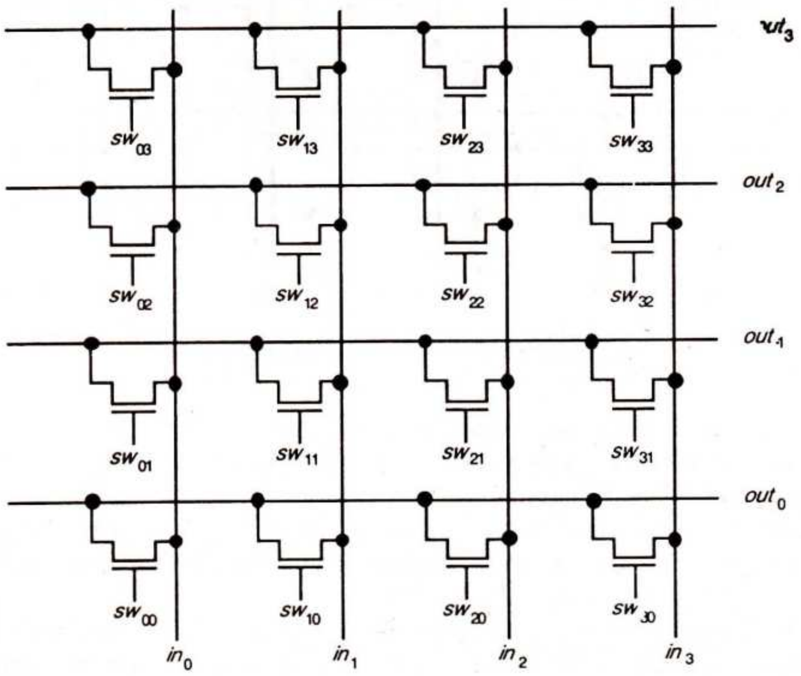


FIGURE 7.6 4 × 4 crossbar switch.

The arrangement is quite general and may be readily expanded to accommodate *n*-bit inputs/outputs. In fact, this arrangement is an overkill in that any input line can be connected to any or all output lines—if all switches are closed, then all inputs are connected to all outputs in one glorious short circuit. Furthermore, 16 control signals (sw_{00}–sw_{15}), one for each transistor switch, must be provided to drive the crossbar switch, and such complexity is highly undesirable. An adaptation of this arrangement recognizes the fact that we can couple the switch gates together in groups of four (in this case) and also form four separate groups corresponding to shifts of zero, one, two and three bits. The arrangement is readily adapted so that the in-lines also run horizontally (to conform to the required strategy).

The resulting arrangement is known as a *barrel shifter* and a 4 × 4-bit barrel shifter circuit diagram is given in Figure 7.7. The interbus switches have their gate inputs connected

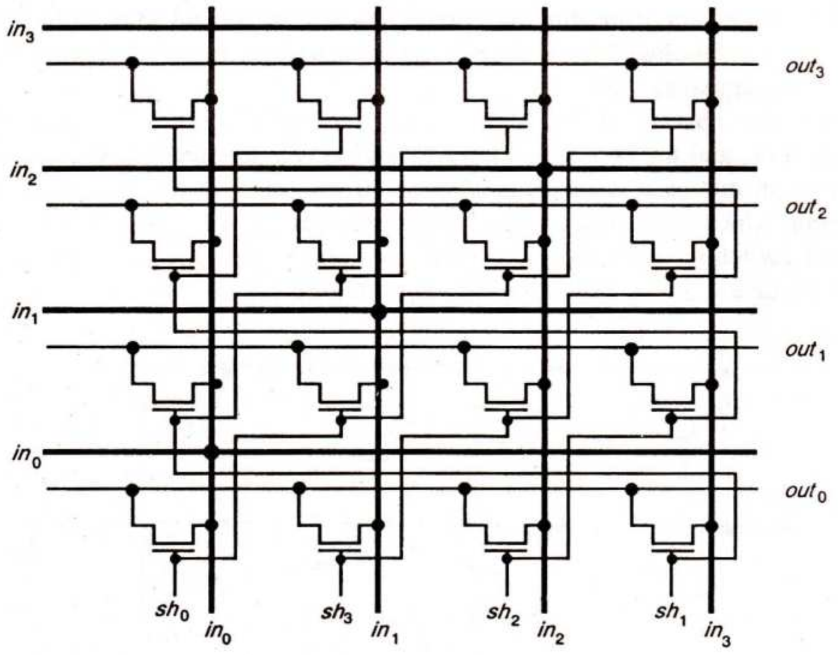


FIGURE 7.7 4 × 4 barrel shifter.

in a staircase fashion in groups of four and there are now four shift control inputs which must be mutually exclusive in the active state. CMOS transmission gates may be used in place of the simple pass transistor switches if appropriate.

The structure of the barrel shifter is clearly one of high regularity and generality and it may be readily represented in stick diagram form. One possible implementation, using simple n-type switches, is given in Figure 7.8.

The stick diagram clearly conveys regular topology and allows the choice of a standard cell from which complete barrel shifters of any size may be formed by replication of the standard cell. It should be noted that standard cell boundaries must be carefully chosen to allow for butting together side by side and top to bottom to retain the overall topology. The mask layout for standard cell number 2 (arbitrary choice) of Figure 7.8 may then be set out as in Figure 7.9. Once the standard cell dimensions have been determined, then any $n \times n$ barrel shifter may be configured and its outline, or bounding box, arrived at by summing up the dimensions of the replicated standard cell. The use of simple n-type switches in a CMOS environment might be questioned. Although there will be a degrading of logic 1 levels through n-type switches, this generally does not matter if the shifter is followed by restoring circuitry such as inverters or gate logic. Furthermore, as there will only ever be one n-type switch in series between an input and the corresponding output line, the arrangement is fast.

The minimum size *bounding box* outline for the 4 × 4-way barrel shifter is given in Figure 7.10. The figure also indicates all inlet and outlet points around the periphery together with the layer on which each is located. This allows ready placing of the shifter within the

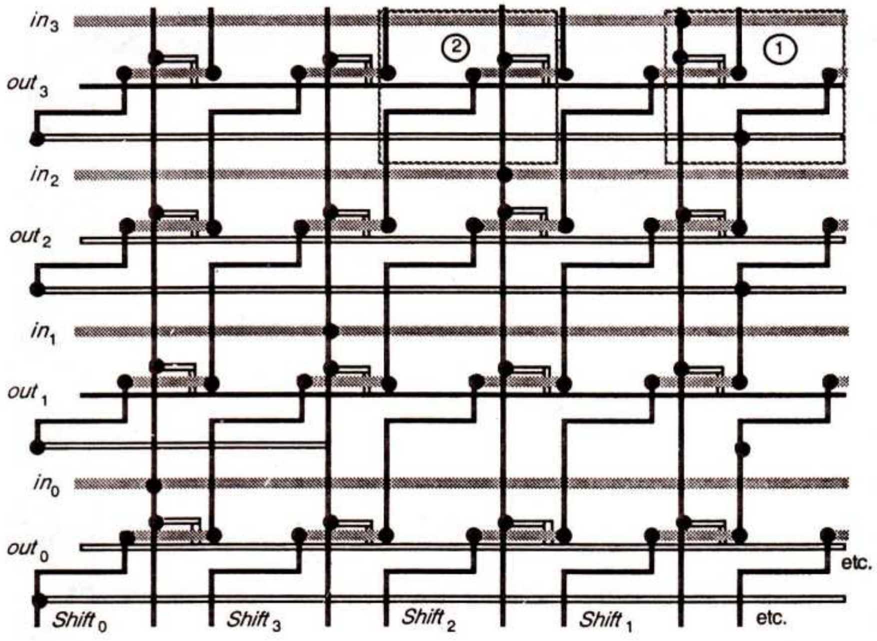


FIGURE 7.8 One possible stick diagram for a 4 × 4 barrel shifter.

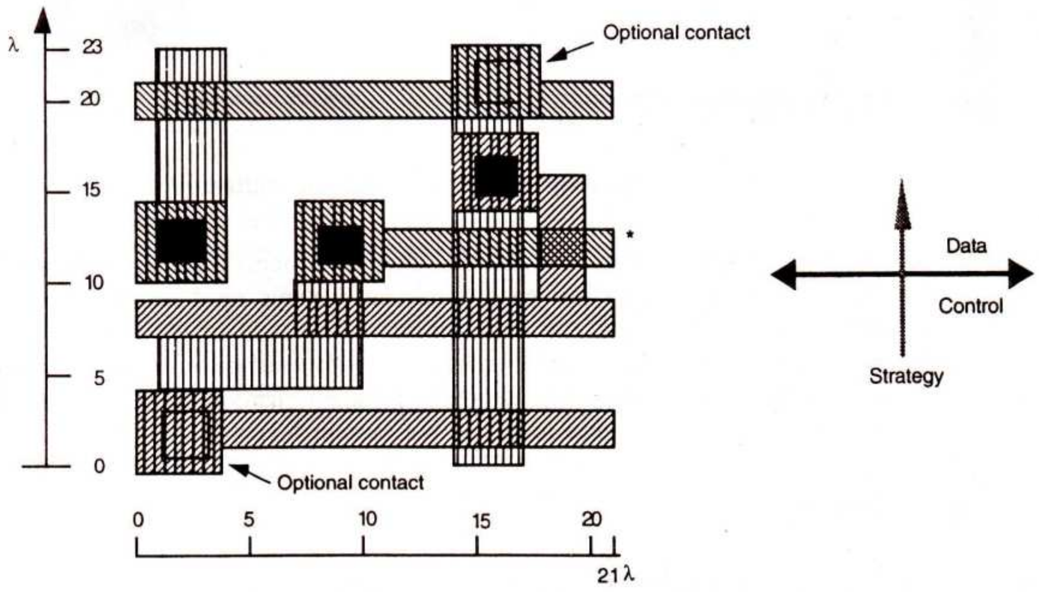


FIGURE 7.9 Barrel shifter standard cell 2—mask layout.

* If this particular cell is checked for design rule errors in isolation, then an error will be generated owing to insufficient extension of polysilicon over thinox where shown. This error will not be present when cells are butted together. This effect is caused by the particular choice of cell boundaries and care must be taken [illegible]

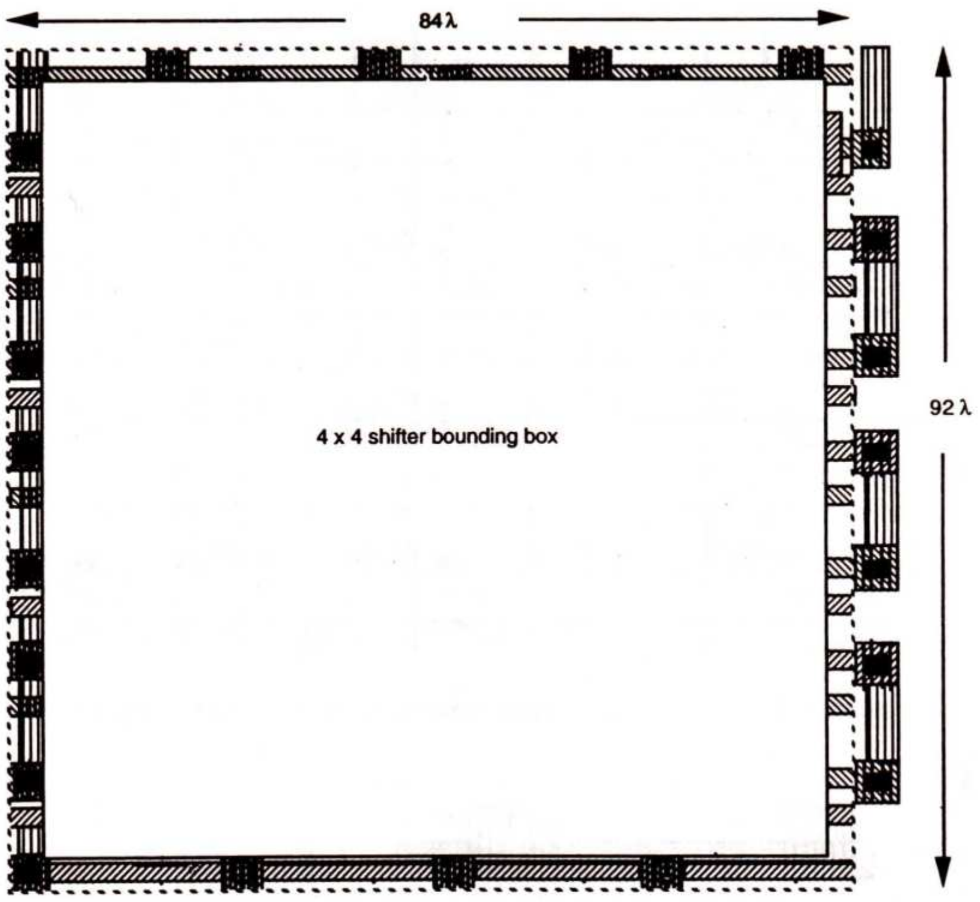


FIGURE 7.10 Bounding box for 4 × 4 barrel shifter.

floor plan (Figure 7.5) and its interconnection with the other subsystems forming the datapath. It also emphasizes the fact that, as in this case, many subsystems need external links to complete their architecture. In this case, the links shown on the right of the bounding box must be made and must be allowed for in interconnections and overall dimensions. This form of representation also allows the subsystem geometric characterization to be that of the bounding box alone for composing higher levels of the system hierarchy.

7.3 OBSERVATIONS

At this stage it is convenient to examine the way we have approached the design of a system and of a particular subsystem in detail. The steps involved may be set out as follows:

1. Set out a specification together with an architectural block diagram.
2. Suitably partition the architecture into subsystems which are, as far as possible, self-contained and which give simple interconnection requirements, if possible.

3. Set out a tentative floor plan showing the proposed physical disposition of subsystems on the chip.
4. Determine interconnection strategy.
5. Revise 2, 3 and 4 interactively as necessary.
6. Choose layers on which to run buses and the main control signals.
7. Take each subsystem in turn and conceive a regular architecture to conform to the strategy set out in 4. Set out circuit and/or logic diagrams as appropriate. Remember that MOS switch-based logic is such that both the logic 1 and logic 0 conditions of an output must be deliberately satisfied (not as in TTL logic, where if logic 1 conditions are satisfied then logic 0 conditions follow automatically).
8. Develop stick diagrams adopting suitable tactics to observe the overall strategy (4) and choice of layers (6). Determine suitable *standard cell(s)* from which the subsystem may be formed.
9. Produce mask layouts for the standard cell(s), making sure that cells can be butted together, side by side and top to bottom, without design rule violation or waste of space. Determine overall dimension of the standard cell(s).
10. Cascade and replicate standard cells as necessary to complete the desired subsystem. This may now be characterized in *bounding box* form with positions and layers of inlets and outlets. External links etc. *must* be allowed for.

7.4 TUTORIAL EXERCISES

1. (a) Set out the mask layout for a 4-way multiplexer using transmission gate switches.
   (b) Determine the overall dimensions (in terms of lambda) for your design.
   (c) Compare the attributes of this design with those of the n-type pass transistor version given in chapter 6.
2. Using a block diagram (symbolic) form of representation for the 4-way multiplexer, draw up an interconnection diagram showing four such multiplexers configured as a 4-bit shift left/right shifter subsystem. The shifter should meet the same overall logical requirements as the barrel shifter designed in this chapter. You should carefully specify the control signals needed for this shifter design.
3. Estimate the area that will be occupied by your design of question 2, assuming the use of the multiplexer design of question 1. Compare this with the barrel shifter design developed in this chapter.

Illustration of the Design Process—Computational Elements

Chapter **8**

Progress is a comfortable disease.

— E.E. CUMMINGS

Beneath this slab
John Brown is stowed
He watched the ad(d)s
And not the road.

— OGDEN NASH

OBJECTIVES

In this chapter we progress to the arithmetic subsystem of the 4-bit data path. Once again, high regularity should be the aim of the designer. If the subsystems are regular and therefore composed of relatively few actual leaf-cell circuits, then the designer can concentrate on the main problem of VLSI design—the routing of interconnections and communication paths. It is hoped that this fact is beginning to emerge as this design progresses. Properly conceived communications—both at leaf-cell and at system levels—are the key to good design.

The arithmetic circuitry required here is relatively simple but does lead into a further consideration of adder circuitry and a fairly comprehensive survey of arrangements for multiplication.

8.1 SOME OBSERVATIONS ON THE DESIGN PROCESS

The design of the shifter, as the first subsystem of the proposed 4-bit data path, has illustrated some important features:

1. First and foremost, try to put requirements into words (an *if, then, else* approach often helps you do this) so that the most appropriate architecture or logic can be evolved.

2. If a standard cell (or cells) can be arrived at, then the actual detailed design work is confined to relatively small areas of simple circuitry (leaf-cells). Such cells can usually have their performance simulated with relative ease so that an idea of the performance of the complete subsystem may be deduced.
3. If generality as well as regularity is achieved then, for example, any size of shifter can be built up by simple replication and butting together of the standard cell(s).
4. Design is largely a matter of the topology of communications rather than detailed logic circuit design.
5. Once standard cell layouts are designed, overall area calculations can be precisely made (*not* forgetting to allow for any necessary links or other external terminations). Thus, accurate floor plan areas may be allocated.
6. VLSI design methodology for MOS circuits is not hard to learn.
7. The design rules are simple and straightforward in application.
8. A structured and orderly approach to system design is highly beneficial and becomes essential for large systems.

8.2 REGULARITY

So far we have used regularity as a qualitative parameter. Regularity should be as high as possible to minimize the design effort required for any system. The level of any particular design as far as this aspect is concerned may be measured by quantifying regularity as follows:

$$\text{Regularity} = \frac{\text{Total number of transistors on the chip}}{\text{Number of transistor circuits that must be designed in detail}}$$

The denominator of this expression will obviously be greatly reduced if the whole chip, or large parts of it, can be fabricated from a few standard cells, each of which is relatively simple in structure.

For the 4 × 4-bit barrel shifter just designed, the regularity factor is given by

$$\text{Regularity} = \frac{16}{1} = 16$$

However, an 8 × 8-bit shifter, for example, would require no more detailed design and would have a regularity factor of 64.

Good system design can achieve regularity factors of 50 or 100 or more, and inherently regular structures, such as memories, achieve very high figures indeed.

8.3 DESIGN OF AN ALU SUBSYSTEM

Having designed the shifter, we may now turn our attention to another subsystem of the 4-bit data path (as in Figure 8.1). A convenient and appropriate choice is the ALU.

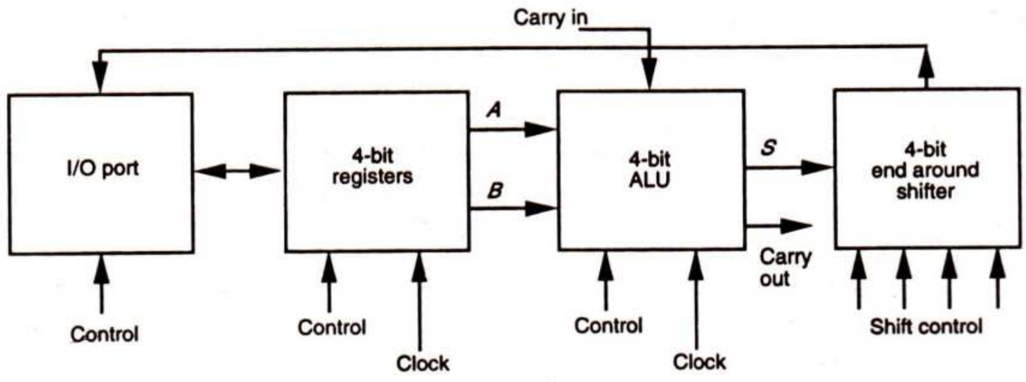


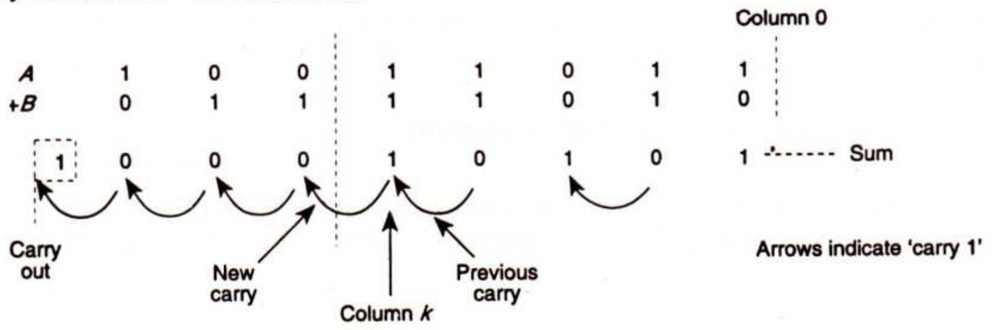
FIGURE 8.1 4-bit data path for processor (block diagram).

The heart of the ALU is a 4-bit adder circuit and it is this which we will actually design, indicating later how it may be readily adapted to subtract and perform logical operations. Obviously, a 4-bit adder must take the sum of two 4-bit numbers, and it will be seen we have assumed that all 4-bit quantities are presented in parallel form and that the shifter circuit has been designed to accept and shift a 4-bit parallel sum from the ALU.

Let us now specify that the sum is to be *stored* in parallel at the output of the adder from where it may be fed through the shifter and back to the register array. Thus, a single 4-bit data bus is needed from the adder to the shifter and another 4-bit bus is required from the shifted output back to the register array (since the shifter is merely a switch array with no storage capability). As far as the input to the adder is concerned, the two 4-bit parallel numbers to be added are to be presented in parallel on two 4-bit buses. We can also decide on some of the basic aspects of system timing at this stage and will assume clock phase ϕ_1 as being the phase during which signals are fed along buses to the adder input and during which their sum is stored at the adder output. Thus clock signals are required by the ALU as shown. The shifter is unclocked but must be connected to four shift control lines. It is also necessary to provide a 'carry out' signal from the adder and, in the general case, to provide for a possible 'carry in' signal, as indicated in Figure 8.1.

8.3.1 Design of a 4-bit Adder

In order to derive the requirements for an n-bit adder, let us first consider the addition of two binary numbers A + B as follows:



It will be seen that for any column k there will be *three* inputs—the corresponding bits of the input numbers, A_k and B_k, and the 'previous carry'—*carry in* (C_{k-1}). It will also be seen that there are two outputs, the *sum* (S_k) and a *new carry* (C_k).

We may thus set out a truth table for the k column of any adder, as in Table 8.1.

TABLE 8.1 Truth table for binary adder

Inputs			*Outputs*	
A_k	B_k	*Previous carry* C_{k-1}	*Sum* S_k	*New carry* C_k
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Conventionally, and assuming that we are not implementing a 'carry look ahead' facility, we may write *standard adder equations*, which fully describe the entries in Table 8.1.

One form of these equations is:

Sum $$S_k = H_k\overline{C}_{k-1} + \overline{H}_k C_{k-1}$$

New carry $$C_k = A_k B_k + H_k C_{k-1}$$

where

Half sum $$H_k = \overline{A}_k B_k + A_k \overline{B}_k$$

Previous carry is indicated as C_{k-1} and $0 \le k \le n - 1$ for n-bit numbers.

These equations may be directly implemented as *And-Or* functions or, most economically, S_k and H_k can be directly implemented with *Exclusive-Or* gates. However, for VLSI custom implementation there are none of the standard logic packages which are the delight of the TTL logic designer. It may be advantageous, then, to restate the requirements in another way.

8.3.1.1 Adder element requirements

Table 8.1 reveals that the *adder requirements* may be stated thus:

If $A_k = B_k$ then $S_k = C_{k-1}$

else $S_k = \overline{C}_{k-1}$

and for the carry C_k

If $\quad A_k = B_k \quad$ then $\quad C_k = A_k = B_k$*

else $\quad C_k = C_{k-1}$

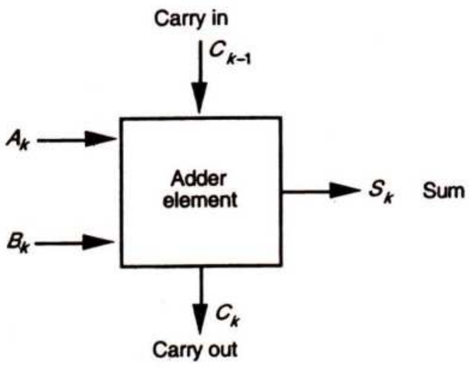
*This relationship could also have been stated as:

Carry $\quad C_k = 1 \quad$ when $\quad A_k = B_k = 1$

or $\quad C_k = 0 \quad$ when $\quad A_k = B_k = 0$

8.3.1.2 A standard adder element

A 1-bit adder element may now be represented as in Figure 8.2. Note that any number of such elements may be cascaded to form any size of adder and that the element is quite general.



n such elements would be cascaded to form an *n*-bit adder.

FIGURE 8.2 Adder element.

Note also that this standard adder element may itself be composed from a number of replicated subcells. Regularity and generality must be aimed at in all levels of the architecture.

Using multiplexers is an implementation of the logic circuitry for the adder element that is easy to follow, resulting directly from the way in which the requirements are stated in words (see section 8.3.1.1). This approach is illustrated in Figures 8.3 and 8.4 and it may be seen that the words used to describe the adder logic are directly implemented by the various paths through the multiplexers.

In these figures the multiplexers form C_k and $\overline{S}_k$ (*not* S_k) to allow single inverter storage or buffering of S_k if this is needed. In fact, one design actually implemented in silicon (see Figure 10.9) uses an nMOS multiplexer-based version of the adder. The basic logic requirements of this adder element, or bit-slice, are thus readily met in nMOS or CMOS technology. However, some practical factors must now be taken into account.

In order to form an *n*-bit adder, *n* adder elements must be cascaded with *carry out* of one element connecting to *carry in* of the next more significant element. Thus, the carry

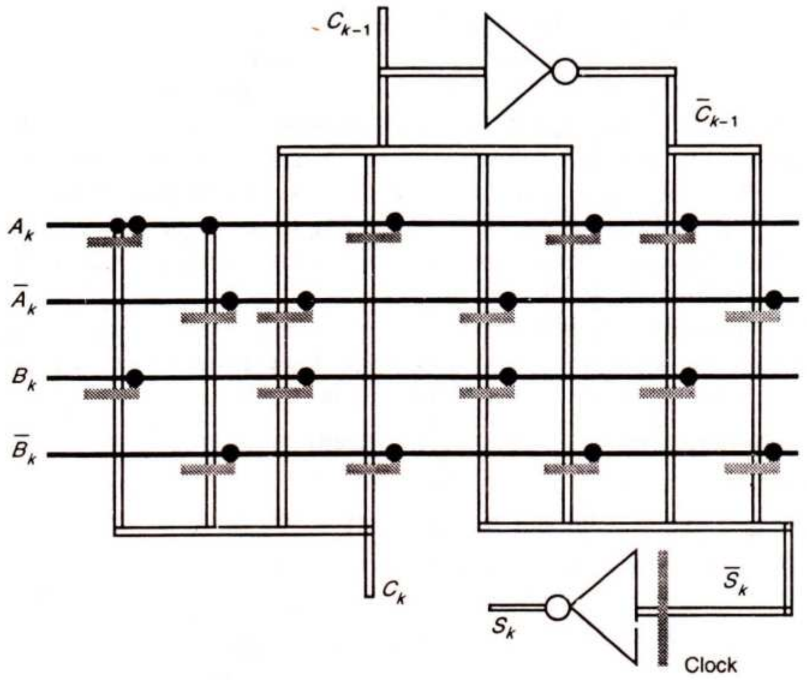


FIGURE 8.3 Multiplexer (*n*-switches)-based adder logic with stored and buffered sum output.

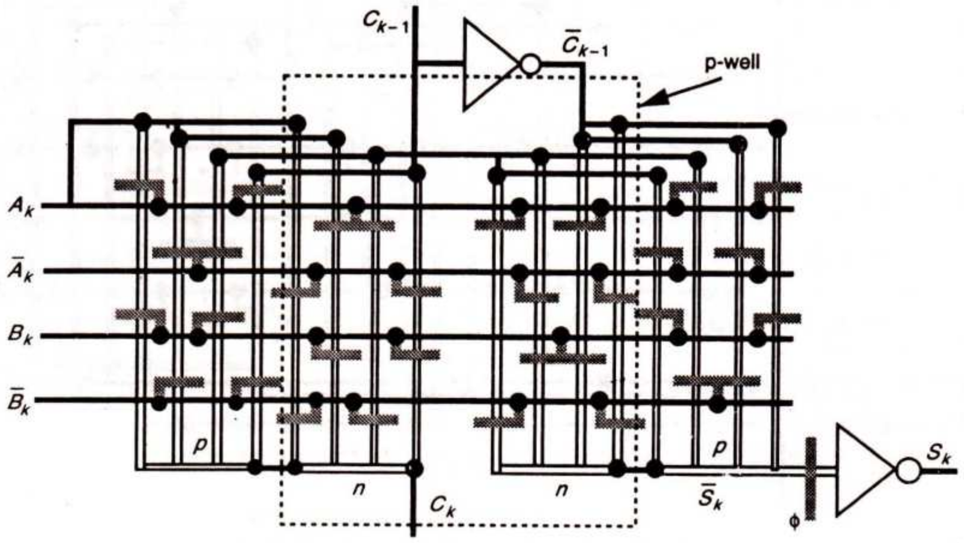


FIGURE 8.4 CMOS version of adder logic.

chain as a whole will consist of many pass transistors in series. This will give a very slow response and the carry line must therefore be suitably buffered between adder elements. (Remember, no more than four pass transistors in series—see section 4.9). Also, we have assumed that both complements, $\bar{A}_k$ and $\bar{B}_k$, of the incoming bits are available. This may not be the case. Furthermore, signals A_k and B_k are to be derived from buses interconnecting the register with the ALU and may thus be taken off the bus through pass transistors. If this is the case, then these signals could not be used directly to drive the pass transistors of the multiplexers (see section 6.2.1). Finally, we must allow for storing the sum at the output of the adder, as discussed earlier in this section.

More practical general arrangements are shown in Figure 8.5. It will be seen that the adder element now contains all necessary buffering (at the expense of increased area). Seven inverter stages are required, deployed as follows (from top to bottom of Figure 8.5):

- Two inverters to form $\bar{C}_{k-1}$ and C_{k-1} (buffered)
- Two inverters to form $\bar{A}_k$ and A_k (buffered)
- Two inverters to form $\bar{B}_k$ and B_k (buffered)
- One inverter to act as a dynamic store for S_k.

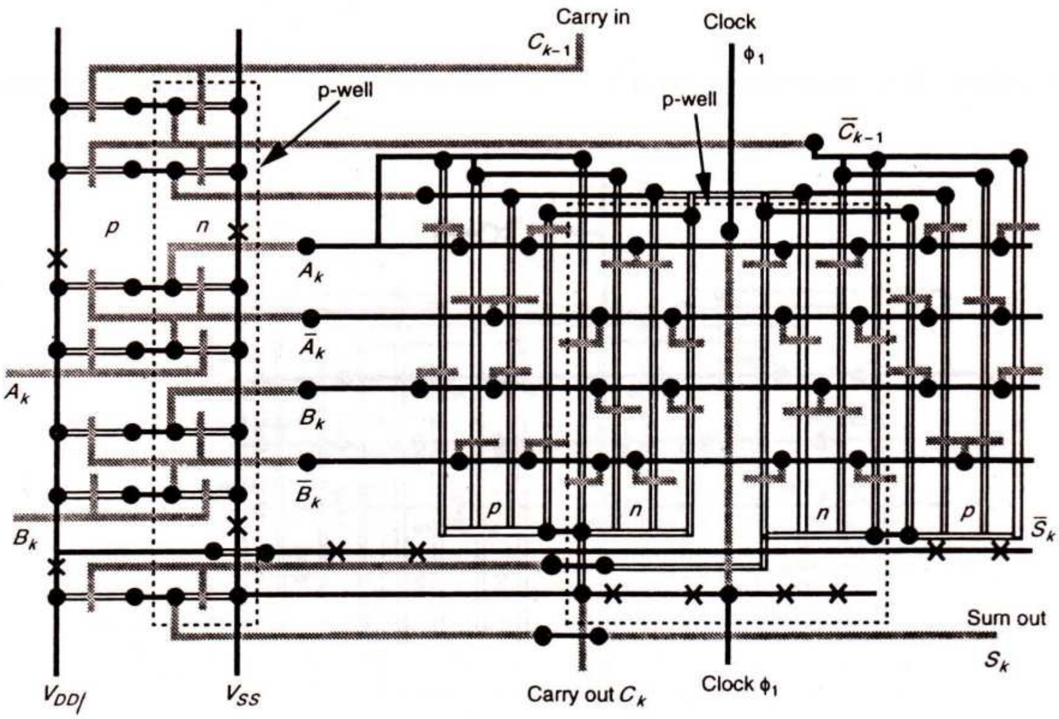


FIGURE 8.5 CMOS adder element.

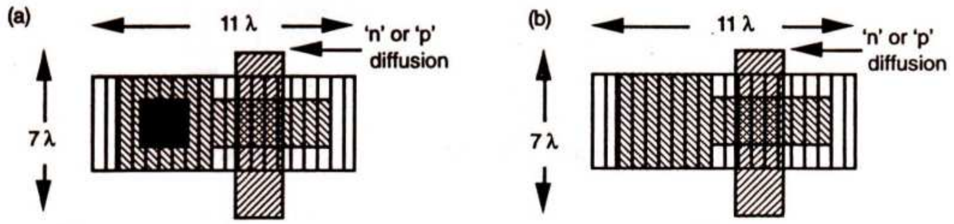
Note that only one inverter is required to store the sum digit S_k *provided* that $\bar{S}_k$ rather than $\bar{S}_k$ is formed by the multiplexer. The observant reader will have already noted that the logic is configured to form the complement $\bar{S}_k$.

8.3.1.3 Standard cells required to be designed for the adder element

The stick diagram of Figure 8.5 shows that the adder consists of three parts:

1. the multiplexers (nMOS or CMOS);
2. the inverter circuits (4:1 and 8:1 ratio nMOS or CMOS);
3. the communication paths.

The first choice is that of technology—for example, nMOS or CMOS—and this in turn decides the detailed nature of the multiplexer and inverters. Both technologies lend themselves well to a replicated standard cell approach, only two standard cells being required for the complete adder element. The first cell, given in Figure 8.6, is the very simple cell from which multiplexers are formed. Note that the one cell design may appear with a transistor (includes poly. and contact) or without a transistor (omits contact cut and/or poly.). The second cell required is an inverter.



FIGURES 8.6 Multiplexer cell with or without cut.

For nMOS, two versions of an inverter are needed—one for an 8:1 ratio and a second version for a 4:1 ratio. However, only one standard inverter cell design is actually needed with a choice of widths for the pull-down channel as shown in Figures 8.7 and 8.8. For completeness, a butting contact based inverter design is shown as Figure 8.7 since these contacts were used at one time by a number of nMOS fabricators.

The use of a 'standard' nMOS inverter with choice of width for the pull-down channel is a common practice. However, note that the narrow channel for the 4:1 configuration in Figure 8.7 has been placed so that its edges are on *whole λ boundaries*, not half λ boundaries as would be the case if narrowing had been carried out symmetrically. *Always* design mask layouts having edges on whole λ boundaries. Some design rule checking software and some fabrication processes might not accept half λ edges.

More commonly, buried contacts would be used to join diffusion and poly. layers in nMOS fabrication and suitable buried contact based inverter designs are given in Figure 8.8.

In this case the vertical dimension is larger than that of Figure 8.7, but there are occasions where the lack of any metal regions in the center of the inverter is a positive advantage. For the layout shown, two metal bus lines could be run through the cell and across the inverter from side to side. This might prove a considerable advantage in saving space in certain layouts, such as register or memory arrays where data buses must run through each storage element. This could not be done when using a compact butting contact design because of the need to maintain 3λ metal to metal separation from the rails and the metal layer 'cover' on the butting contact.

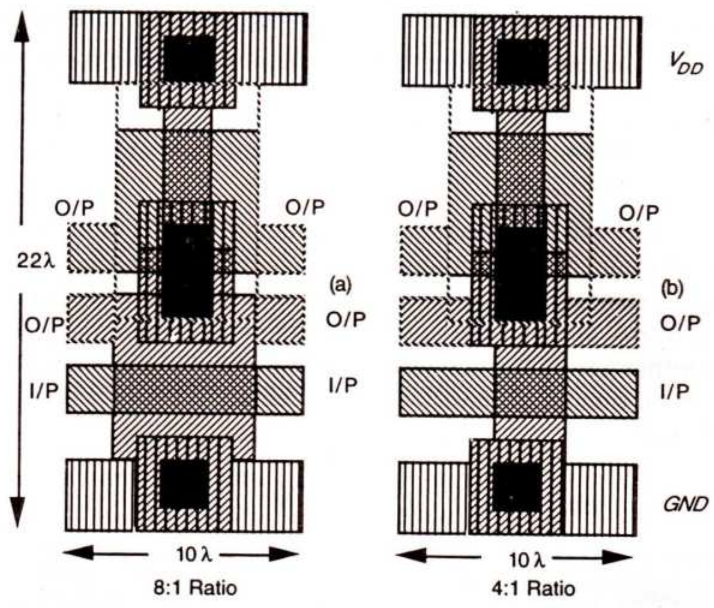


FIGURE 8.7 nMOS (butting contact) inverters.

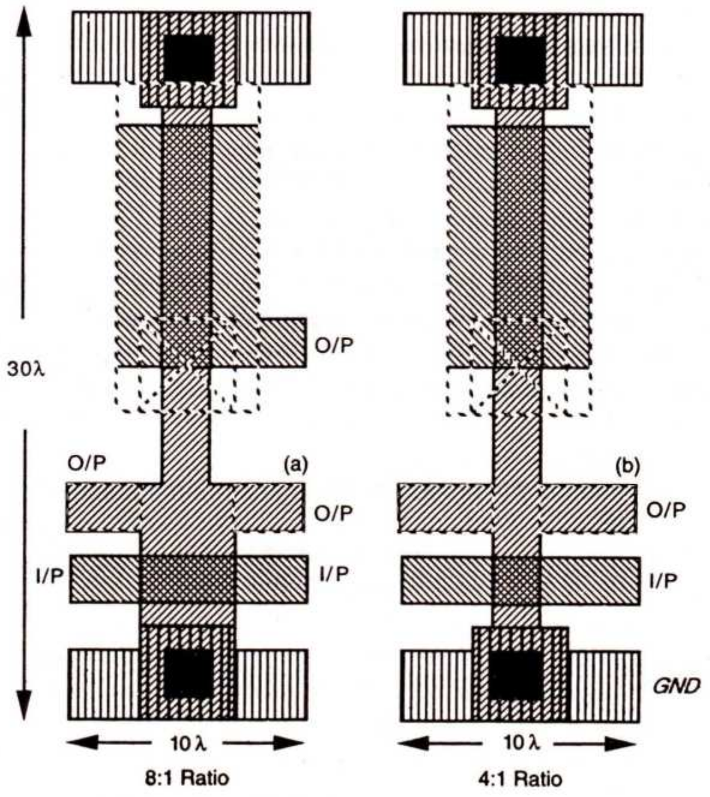


FIGURE 8.8 nMOS (buried contact) inverters.

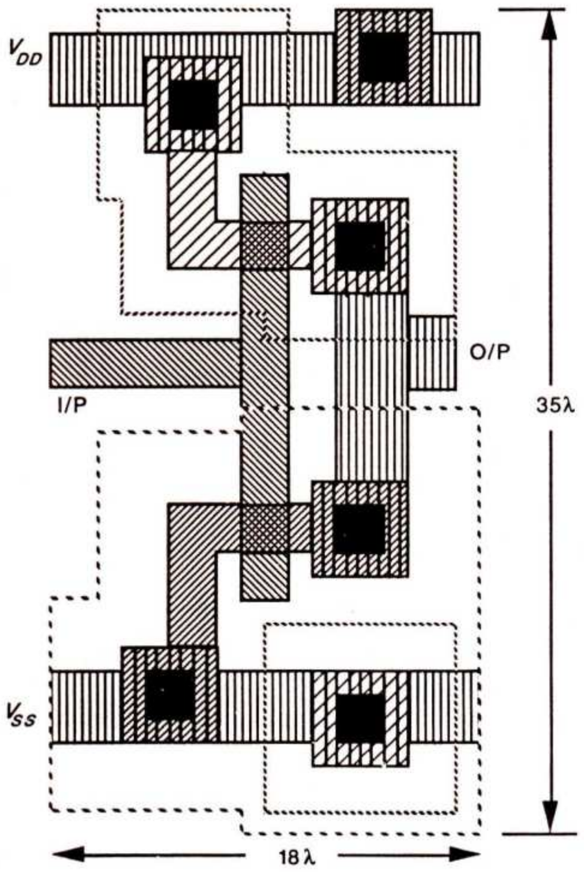


FIGURE 8.9 A CMOS inverter design.

For CMOS-based designs, as set out in Figure 8.5, we normally need a complementary CMOS inverter. A possible mask layout is shown in Figure 8.9.

8.3.1.4 Adder element bounding box

We may now assess the area requirements for, say, the CMOS adder element as in Figure 8.5. First estimate the bounding box for the multiplexer area of the adder. Each standard multiplexer cell (Figure 8.6) is $7\lambda \times 11\lambda$ and there are 16 such elements side by side 'horizontally' and four stacked 'vertically'. We must also allow at least an additional 6λ width for the metal to metal spacings required by the clock bus passing through the center. In the vertical direction we must allow spacings for the interconnections between the tops of the multiplexers (an estimated additional 30λ) and a further 10λ for the connection out from $\overline{S}_k$ and C_k at the bottom. Thus, the bounding box must be at least $16 \times 7\lambda + 6\lambda = 118\lambda$ 'wide' and $4 \times 11\lambda + 30\lambda + 10\lambda = 84\lambda$ in 'height', as shown in Figure 8.10.

To complete an assessment of the approximate area to be occupied by the CMOS adder element we need to allow for the seven inverters shown in Figure 8.5. We have already

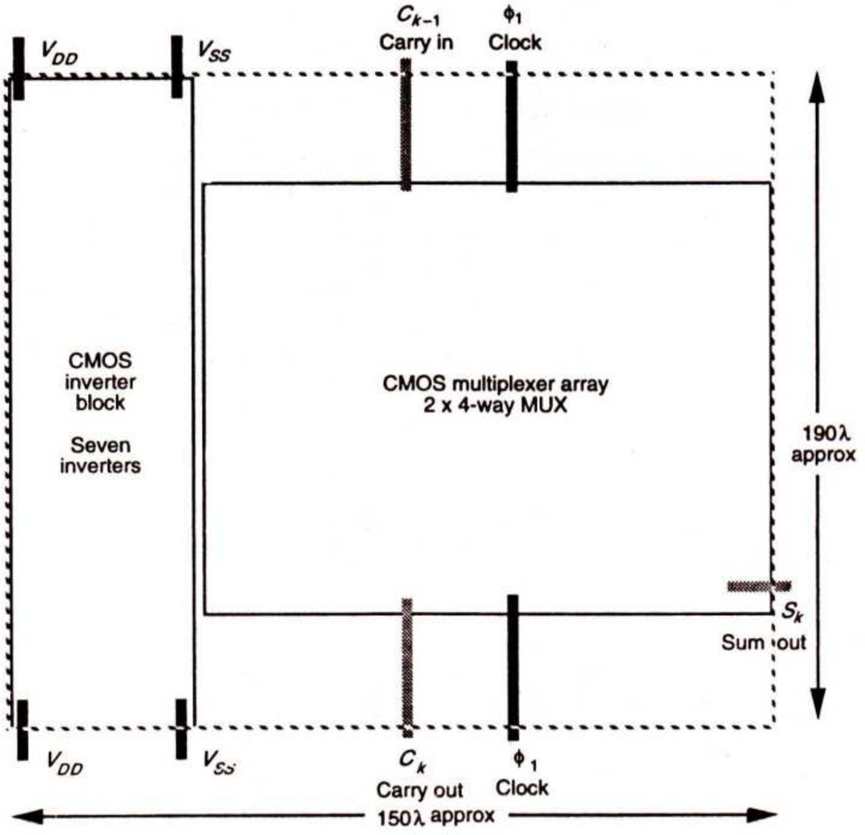


FIGURE 8.10 Approximate bounding box and floor plan for CMOS adder element.

determined a bounding box outline for a suitable CMOS inverter circuit (see Figure 8.9) and it will be seen that each inverter occupies a rectangle measuring 18λ 'wide' and 35λ 'high'. Thus, seven inverters alone placed side by side will occupy an area of 126λ × 35λ and, allowing, say, an additional 50% width for space between each for connections, we have an overall area requirement of about 190λ × 35λ for the inverters. Thus, the overall bounding box (or *floor plan outline*) for a complete adder element will be approximately that given as the overall outline in Figure 8.10. Note that vertical distribution of power is required by this layout, but the direction of global power distribution may be reviewed as the design of the complete processor—floor plan as in Figure 7.5—progresses. Details of inlet/outlet points on the inverter block and overall adder element bounding boxes will be worked out as part of the next tutorial exercise.

The 4-bit adder is then formed by cascading four adder elements as indicated in Figure 8.11(a) and an initial assessment of the minimum floor plan area requirement follows from the 4-bit adder bounding box of Figure 8.11(b). This is the second subsystem of the floor plan of Figure 7.5, the first being the barrel shifter of Figure 7.10.

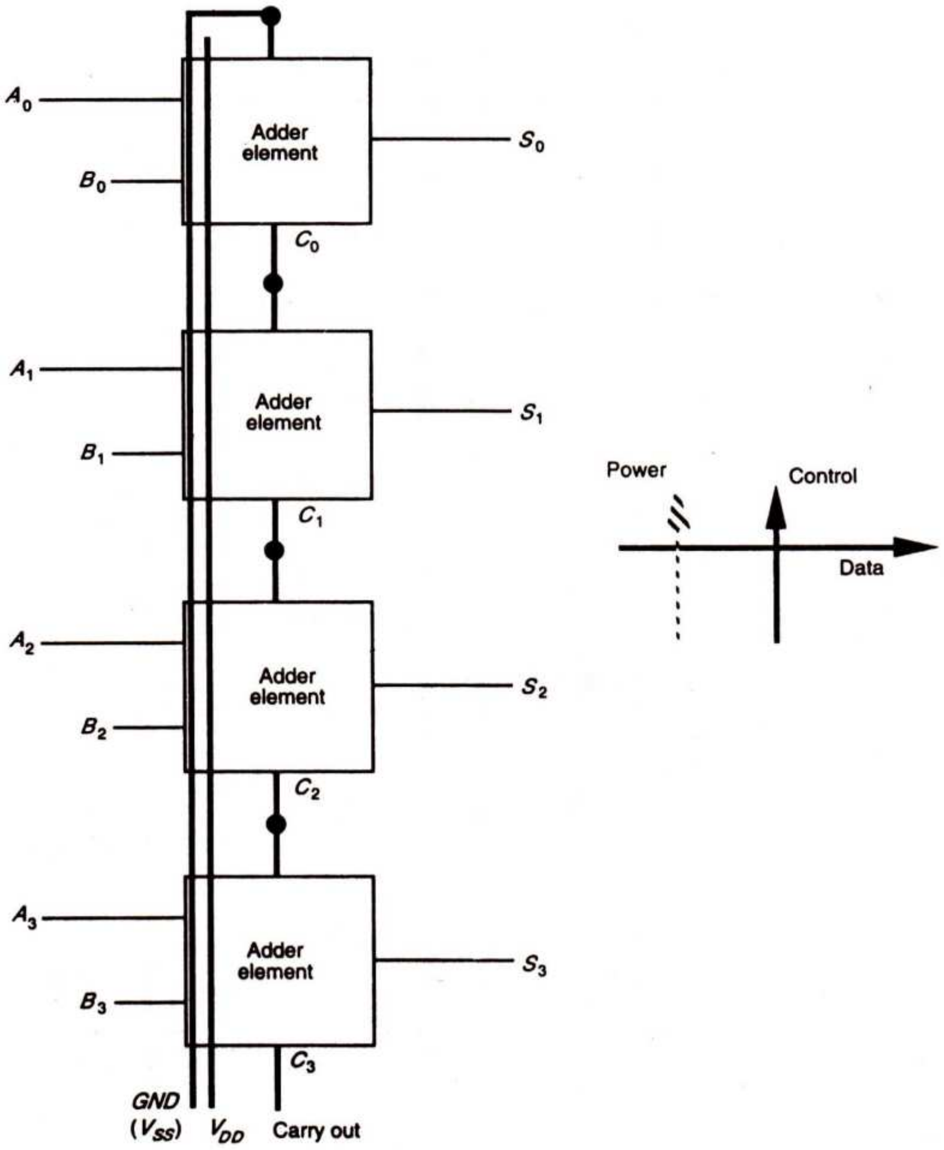


FIGURE 8.11(a) 4-bit adder.

8.3.2 Implementing ALU Functions with an Adder

An arithmetic and logical operations unit (ALU) must, obviously, be able to *add* two binary numbers ($A + B$), and must also be able to *subtract* ($A - B$).

From the point of view of logical operations it is essential to be able to *And* two binary words (A.B). It is also desirable to *Or* ($A + B$) and perhaps also detect *Equality*, and of course we also need an *Exclusive-Or* function.

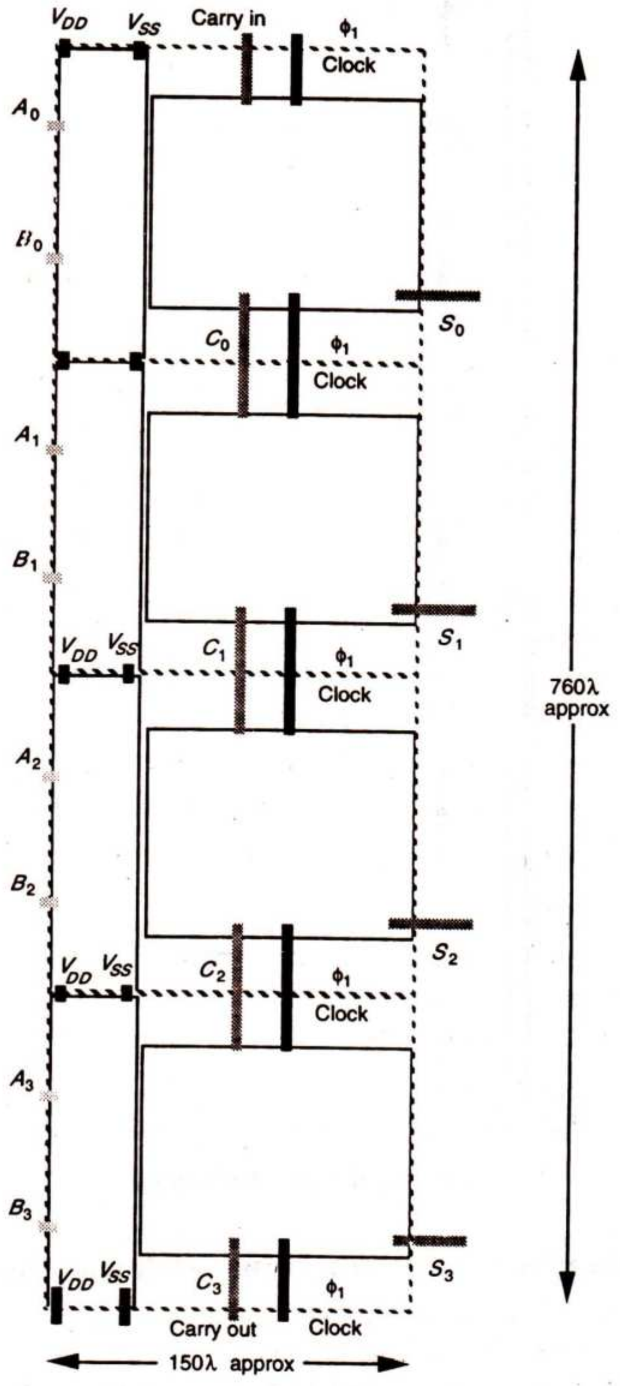


FIGURE 8.11(b) 4-bit adder outline.

Subtraction by an adder is an easy operation provided that the binary numbers A and B are presented in *twos complement* form. In this case, to find the difference $A - B$ it is only necessary to complement B (exchange 1 for 0 and vice versa for all bits of B), add 1 to the number thus obtained, and then *add* this quantity to A using the standard addition process discussed earlier. The output of the adder will then be the required difference in twos complement form. Note that the complement facility necessary for subtraction can also serve to form the *logical complement* (which is indeed exchanging 0 for 1 and vice versa).

It is highly desirable to keep the architecture of the ALU as simple as possible, and it would be nice if the adder could be made to perform logical operations as readily as it performs subtraction. In order to examine this possibility, consider the standard adder equation set out in section 8.3.1 and reproduced here:

Sum $\quad S_k = \bar{H}_k C_{k-1} + H_k \bar{C}_{k-1}$

New carry $\quad C_k = A_k B_k + H_k C_{k-1}$

where Half sum $\quad H_k = \bar{A}_k B_k + A_k \bar{B}_k$

Consider, first, the *Sum* output if C_{k-1} is held at logical 0, then

$$S_k = H_k.1 + \bar{H}_k.0 = H_k$$

that is $\quad S_k = H_k = A_k B_k + A_k B_k$—An *Exclusive-Or* operation

Now, hold C_{k-1} at logical 1, then

$$S_k = H_k.0 + \bar{H}_k.1 = \bar{H}_k$$

that is

$S_k = \bar{H}_k = \bar{A}_k \bar{B}_k + A_k B_k$—An *Exclusive-Nor* (*Equality*) operation

Next, consider the *carry* output of each element, first if C_{k-1} is held at logical 0.

Then

$C_k = A_k.B_k + H_k.0 = A_k.B_k$— An *And* operation

Now, if C_{k-1} is held at logical 1, then

$$C_k = A_k.B_k + H_k.1 = A_k.B_k + \bar{A}_k.B_k + A_k.\bar{B}_k$$

Therefore

$C_k = A_k. + B_k$ — An *Or* operation

Thus it may be seen that suitable switching of the carry line between adder elements will give the ALU logical functions. A possible arrangement of the adder elements for both arithmetic and logical functions is suggested in Figure 8.12.

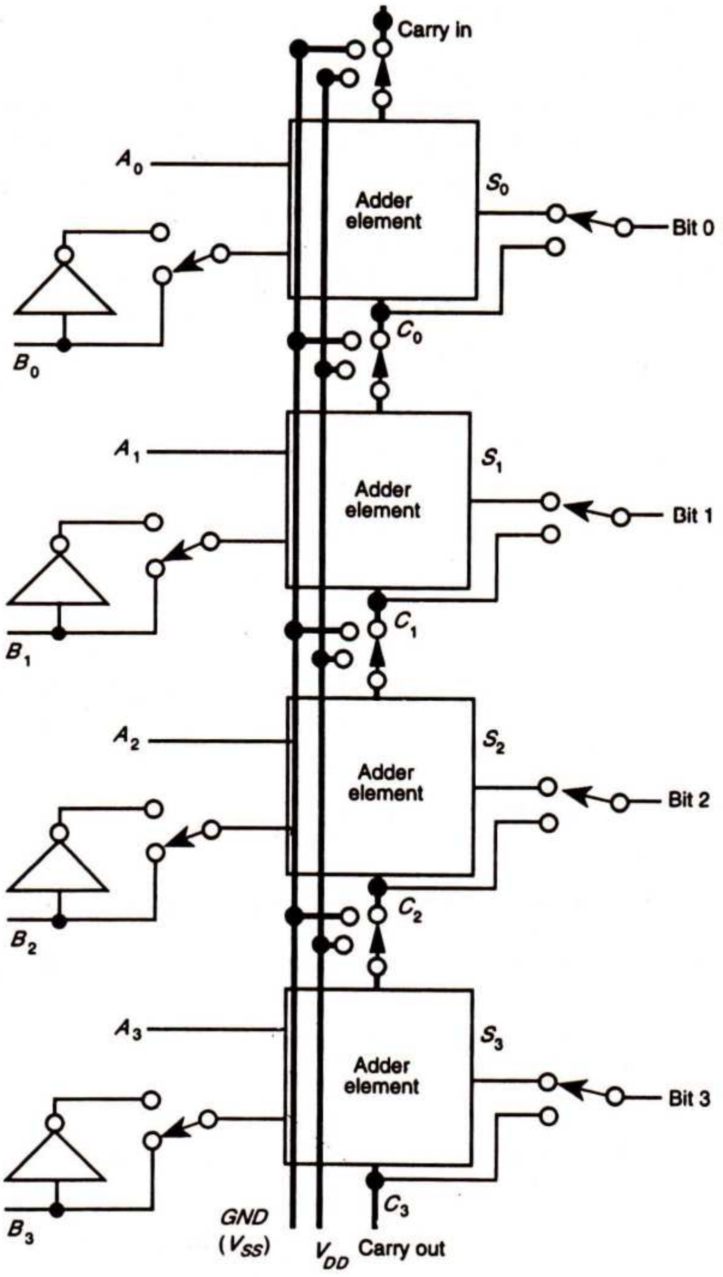


FIGURE 8.12 4-bit ALU.

8.4 A FURTHER CONSIDERATION OF ADDERS

A further consideration of aspects of adder circuitry is desirable since adders are the basic elements of all arithmetic processes. Also, so far, we have taken a very simple and direct approach to implementing the adder equations and have not considered refinement or optimization of performance.

In order to broaden the scope of our discussion, let us first consider some of the commonly used alternative forms of the adder equations introduced in section 8.3.1 and repeated here for convenience.

Sum $$S_k = \bar{H}_k C_{k-1} + H_k \bar{C}_{k-1}$$

New carry $$C_k = A_k B_k + H_k C_{k-1}$$

where

Half sum $$H_k = \bar{A}_k B_k + A_k \bar{B}_k$$

The expressions may also make use of lowercase letters. New carry may also be expressed in terms of the previous carry c_{k-1} with a *propagate* signal p_k and *generate* signal g_k, where

$$p_k(=H_k) = a_k \oplus b_k \text{ and } g_k = a_k . b_k$$

Then we may write,

new carry $$c_k = p_k \cdot c_{k-1} + g_k$$

or $$c_k = (a_k + b_k)c_{k-1} + a_k . b_k$$

and sum $$s_k = a_k \oplus b_k \oplus c_{k-1}$$

The sum may also be expressed in terms of the carry in c_{k-1} and carry out signals c_k together with the input bits a_k and b_k as follows:

$$s_k = \bar{c}_k .(a_k + b_k + c_{k-1}) + a_k . b_k . c_{k-1}$$

Such manipulations lead, for example, to the complementary CMOS logic circuit in Figure 8.13.

However, an alternative and perhaps more direct realization, which leads to the concept of a carry chain, is set out in Figure 8.14. This in turn, when considering carry circuits alone, leads to a popular arrangement known as the *Manchester carry-chain.*

8.4.1 The Manchester Carry-chain

Instead of the carry passing through a complete transmission gate as in Figure 8.14, the carry path is precharged by the clock signal and the carry path may then be gated by a single n-type pass transistor as shown in Figure 8.15.

Although individual Manchester carry cells are fast, care must be taken when cascading them since this effectively connects pass transistors in series. We have already seen that the delay goes up as the square of n where n is the number connected in series. Obeying the rules set out earlier to cover this situation, we must buffer after every four carry chain cells as shown in Figure 8.16.

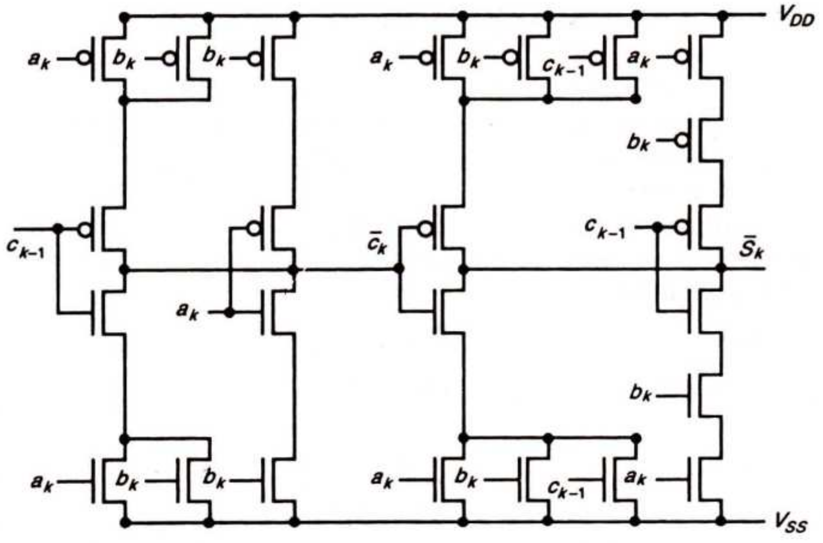


FIGURE 8.13 One possible (symmetrical) adder cell arrangement.

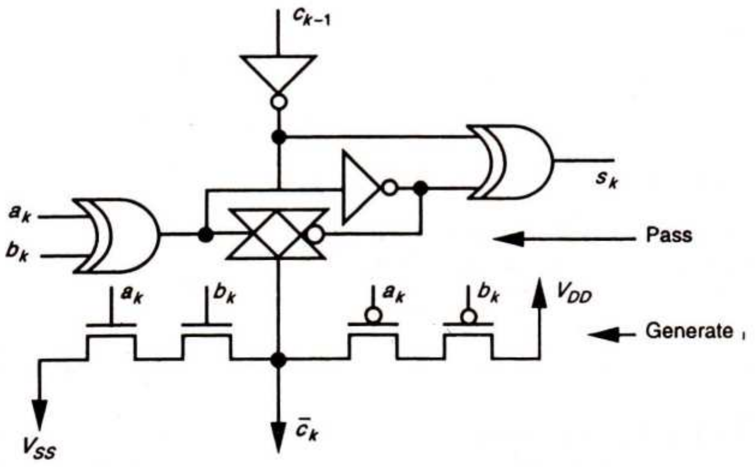


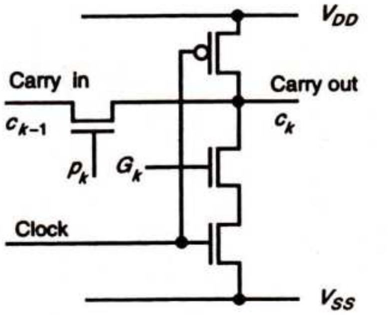
FIGURE 8.14 An adder element based on the pass/generate concept.

In BiCMOS technology it is possible to implement this arrangement and achieve speed improvement by a factor of two over the CMOS arrangement. However, this approach functions with lower input voltage swings to achieve the full speed advantage (Hotta et al., 1986).

8.4.2 Adder Enhancement Techniques*

In the case of small adders ($n < 8$-bits), it is generally advantageous to adopt the relatively

* This section is based on material provided by Dr B. Hochet of the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. The authors gratefully acknowledge this contribution.



Note in this case, $P_k = a_k \oplus b_k$ as before
but $G_k = \bar{a}_k . \bar{b}_k$

FIGURE 8.15 Manchester carry-chain element.

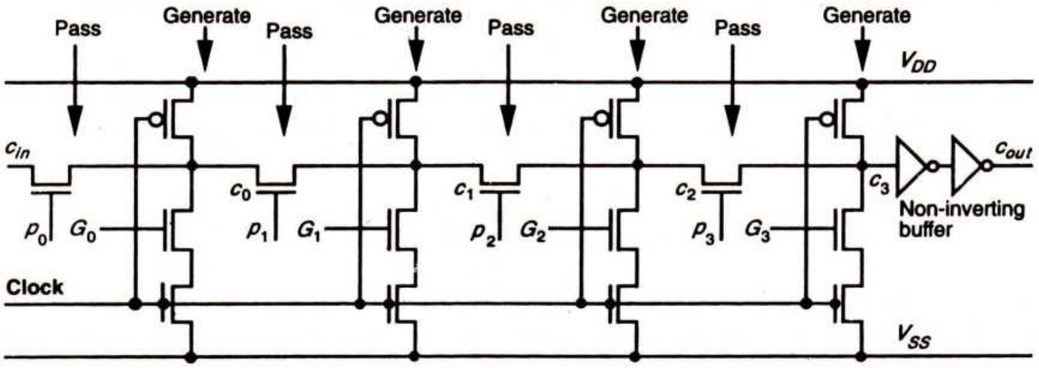


FIGURE 8.16 Cascaded Manchester carry-chain elements with buffering.

simple hardware of the ripple through carry. Thus, the carry completion time is clearly directly proportional to n. On the other hand, large adders (up to say $n = 64$ or even $n = 128$-bits) cannot afford to wait for the long completion time of a large ripple through carry line. Thus special techniques must be adopted to improve addition time. This improvement is possible only through some increase in complexity and, in consequence, at the expense of increased area in silicon. The next subsections discuss three techniques for effecting faster carry generation and each approach is characterized by a different area/performance ratio.

8.4.2.1 Carry select adders

For this arrangement—also referred to as a conditional sum adder—the adder is divided into blocks. Each block is composed of two adders, one with a logical 0 *carry in* and the other with a logical 1 *carry in*. The *sum* and *carry out* generated are then selected by the actual *carry in* which comes from the *carry out* output of the previous block as shown in Figure 8.17.

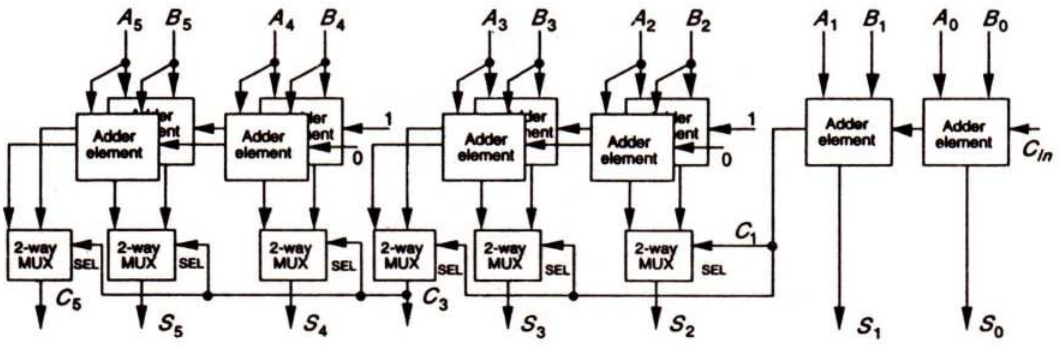

A_5 B_5 A_4 B_4 A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0
Adder element
1
0
C_{in}
2-way MUX
SEL
C_1
C_5 S_5 S_4 C_3 S_3 S_2 S_1 S_0

FIGURE 8.17 Carry select adder structure (6-bit).

8.4.2.1.1 Optimization of the carry select adder

Let us consider an n-bit ripple carry adder. The computation time T is given by:

$$T = k_1 n$$

where k_1 is the delay through one adder cell.

If we now divide the adder into blocks, each with two parallel paths, then the completion time T becomes

$$T = k_1 . \frac{n}{2} + k_2$$

where k_2 is the time needed by the multiplexer of the next block to select the actual output carry. A decision now has to be made on the size, in bits, of each adder block and clearly this could be 1-bit, in which case the number of multiplexers is a maximum, or two or more bits resulting in fewer multiplexers. If there are many multiplexers, then the ripple through effects occur in the multiplexer chain rather than in the carry chain through the blocks. Consequently, an optimum value must be sought for the block size.

Suppose the n-bit adder is divided into M blocks, and that each block contains P adder cells in series, and considering the arrangement of Figure 8.17, we may see that the completion time T for the overall carry output signal is composed of two parts:

- the propagation delay through the first block;
- the propagation delay through the multiplexers.

so that,

$$T = Pk_1 . + (M - 1)k_2$$

noting that $n = M.P$, the minimum value for T is reached when

$$M = \sqrt{(n.k_1/k_2}$$

As a further improvement, each succeeding block may be extended by one or more stages to account for the delay in the multiplexer. For instance, if the delay in the multiplexer is equal to the cell delay, then the size P of the succeeding block should be increased by one. On the other hand, if the multiplexer delay is twice that for the cell delay, then each block may have two more adder cells than the previous one; that is, P can be increased by two from one block to the next. The actual optimum increase in P from one block to its successor depends on the ratio between k_1 and k_2. However, care must be taken to properly allow for the multiplexer delay which will also depend on the number of inputs, that is, on P, increasing as P increases.

It should also be noted that the adder blocks do not have to be ripple carry adders but may use any of the available enhancement techniques, such as carry look-ahead or carry skip techniques. In such cases, the optimization requirements may be different from those discussed here.

8.4.2.2 Carry skip adders

When computing an addition with a ripple through adder, the completion time will sometimes be small since the carries, generated at several positions, are formed simultaneously as shown (e.g. with three carries) in Figure 8.18.

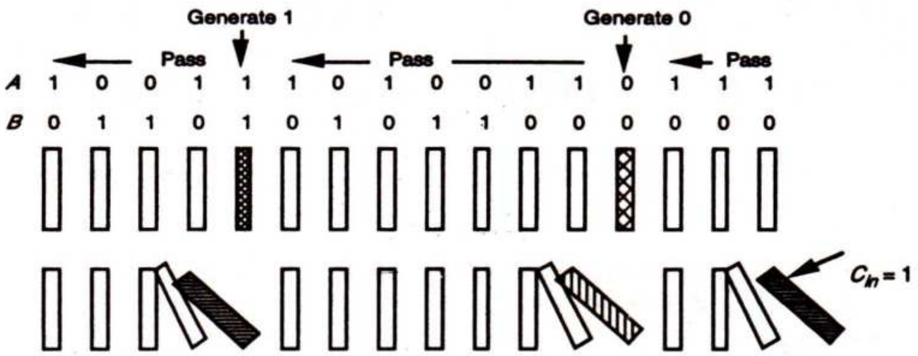

Generate 1
Generate 0
Pass
Pass
Pass
A 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1
B 0 1 1 0 1 0 1 0 1 1 0 0 0 0 0 0
$C_{in} = 1$

FIGURE 8.18 Diagrammatic representation of carry skip adder.

In this case, the carry propagation may be likened to the domino principle, where, if one falls, then each successive stage is knocked over in turn up to the next point at which a different carry is formed.

In this example, assuming the input *carry in* = 1, three simultaneous carry propagation chain reactions occur. It may be seen that the longest chain is the second one, which takes seven cell delays (from the fourth bit to the 11th bit). Thus, the addition time for these two numbers is determined by the longest chain, and in this case will be given by

$$T = 7.k + k'$$

where k is the cell delay and k' is the time needed to compute the 11th bit sum using the carry in to the 11th bit.

If, for a ripple carry adder, the input bits A_i and B_i are different for all bit positions, then the input carry is propagated at all bit positions and never generated. The addition is thus only completed after the carry has propagated along the entire adder. In this case, the computation delay must be *nk,* and although it may be less than this quite frequently, the worst case must be assumed in all cases when using the adder in, say, high speed or real-time or other time-critical applications.

Carry skip adders take advantage of both the generation and the propagation of the carry signal. They may be divided into blocks where, for each block, a special circuit is used to detect the condition when A and B bits differ in all bit positions in the block (that is $p_i = 1$ for all 'i' in the block). The output signal from such a circuit is called the *block propagation signal*. If the *block propagation signal* = 1, then the carry signal entering the block can bypass it and be transmitted through a multiplexer to the next block. Figure 8.19 sets out the schematic structure of a 24-bit carry skip adder, subdivided into four blocks and based on this approach.

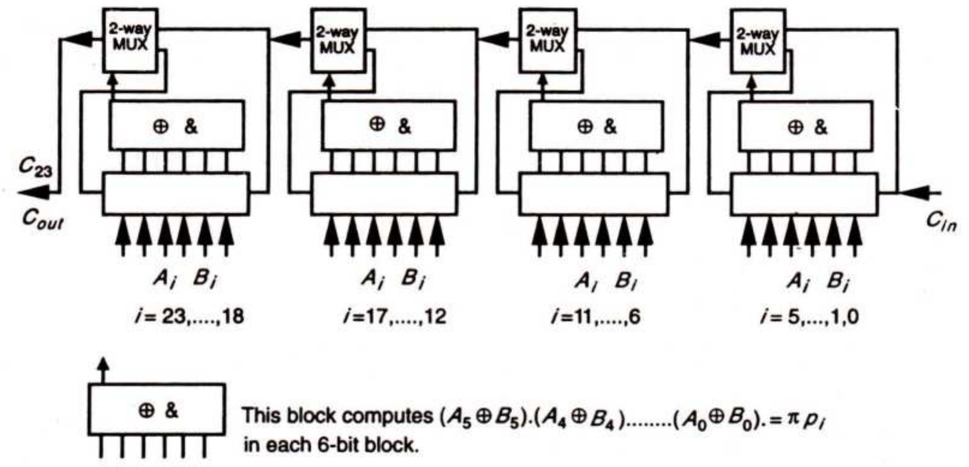


FIGURE 8.19 Structure of a 24-bit (for example) carry skip adder.

8.4.2.2.1 *Optimization of the carry skip adder*

Once again there will be factors which determine the optimum block size for this arrangement and in this case we assume equal size blocks. Let k_1 denote the time needed by the carry signal to propagate through the adder cell, and k_2 the time needed for a carry to skip over a block. Further, let us divide the n-bit carry skip adder into M blocks—each block containing P adder cells. Since, as was the case for the ripple carry adder, the actual computing time depends on the configuration of the input numbers, the completion time may well be small but may also reach the worst case. We must thus evaluate and optimize the worst case conditions as depicted in Figure 8.20.

The total (worst case) propagation delay time T is given by

$$T = 2(P - 1).k_1 + (M - 2)k_2$$

where

$$P = n/M$$

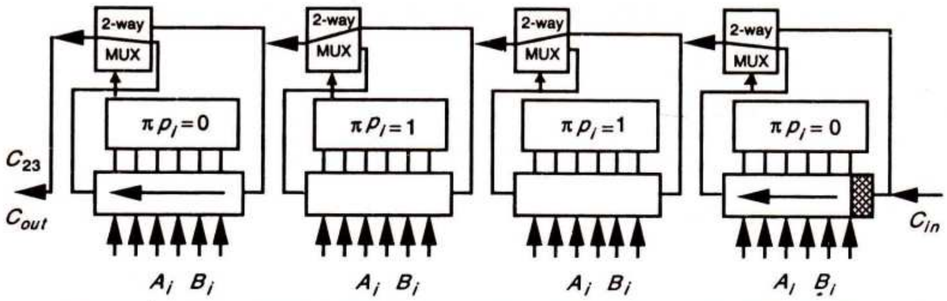


FIGURE 8.20 Worst case carry propagation for carry skip adder.

The minimum value of T is reached when

$$M = \sqrt{(2n.k_1/k_2}$$

As for the carry select adders, a further improvement may be achieved if the adder is divided into blocks of differing sizes (Guyot et al., 1987).

Finally, Figure 8.21 shows a possible arrangement for a block, complete with its multiplexer and block propagation signal generating circuit. This particular realization leads to good regularity and thus to a high density layout in silicon.

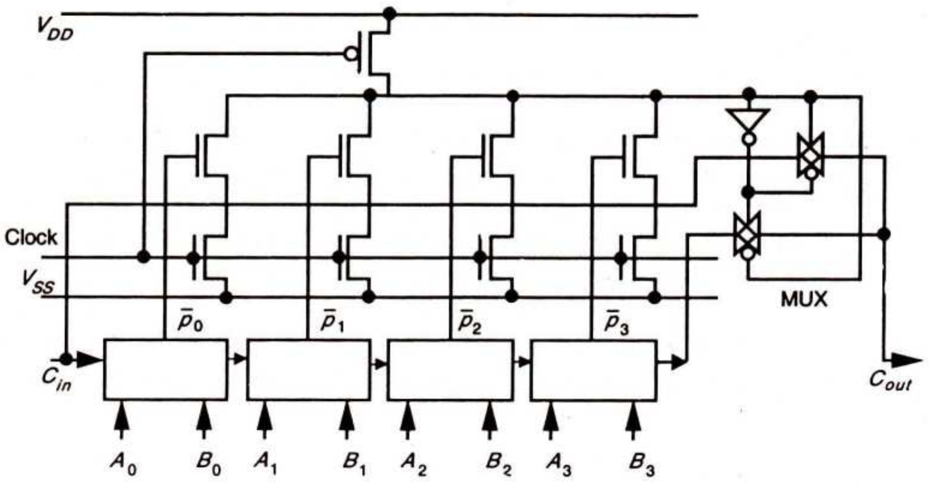


FIGURE 8.21 Possible implementation of the block propagation concept.

8.4.2.3 Carry look-ahead (CLA) adders

We have considered some other methods of improving adder throughput time and may now turn to algebra to seek a general solution to this problem. This is to be found in rearranging the expressions for the adder (given in section 8.3.1), in particular the expression for carry

$$C_k = A_kB_k + H_k.C_{k-1}$$

noting that $H_k = A_kB_k + A_kB_k$ the expression can be rearranged into the form

$$C_k = A_k.B_k + (A_k + B_k).C_{k-1}$$

Thus for C_0 we may write

$$C_0 = A_0.B_0 + (A_0 + B_0)C_{in}$$

which allows for an input carry; and, therefore

$$C_1 = A_1.B_1 + (A_1 + B_1)C_0$$

may then be written as

$$C_1 = A_1.B_1 + (A_1. + B_1).A_0.B_0. + (A_1. + B_1.).(A_0 + B_0).C_{in}$$

and, similarly

$$C_2. = A_2.B_2. + (A_2 + B_2).A_1.B_1 + (A_2. + B_2).(A_1 + B_1).A_0.B_0. + (A_2 + B_2).(A_1 + B_1).(A_0 + B_0).C_{in}$$

The next stage would be

$$C_3 = A_3.B_3 + (A_3 + B_3).A_2.B_2 + (A_3 + B_3).(A_2 + B_2).A_1.B_1 + (A_3 + B_3).(A_2 + B_2).(A_1 + B_1).A_0.B_0 + (A_3 + B_3).(A_2 + B_2).(A_1. + B_1).(A_0 + B_0).C_{in}$$

and so on for further stages.

If there is no input carry, then C_{in} becomes 0 and the last term in each expression for carry will be eliminated.

Although these expressions become very lengthy as the bit significance increases, each expression is only three logic levels deep, so the delay in forming the carry is constant irrespective of bit position. However, the logic does rapidly become over-cumbersome and also presents problems in 'fan-out' and 'fan-in' requirements on the gates used. A compromise, usually adopted, is a combination of 'carry look-ahead' and 'ripple through' as indicated in Figure 8.22. The 3-bit groups shown were arbitrarily chosen to illustrate the approach.

Following this particular approach, we may now write carry look-ahead expressions in terms of the generate g_k and propagate p_k signals defined earlier. The general form for the carry signal c_k thus becomes

$$c_k = g_k + p_k \cdot g_{k-1} + p_k \cdot p_{k-1} g_{k-2} + \ldots\ldots + p_k \ldots\ldots p_1 \cdot g_0 + p_k \ldots\ldots p_0 \cdot c_{in}$$

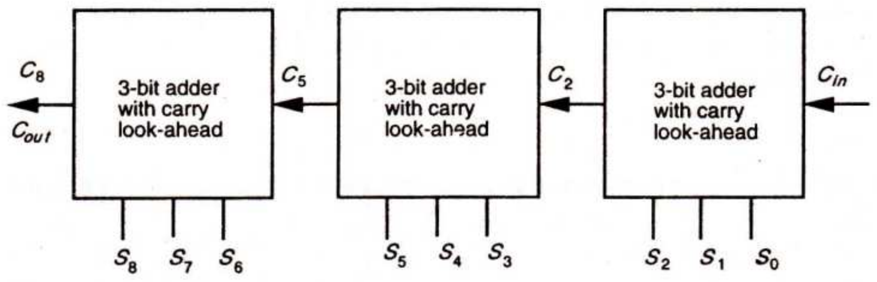
Considering a CLA-based adder divided into blocks of 4-bits, as in Figure 8.23, we may write the expressions for the carry circuits in one block as follows:

$$c_0 = g_0 + p_0 \cdot c_{in}$$

$$c_1 = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_{in}$$

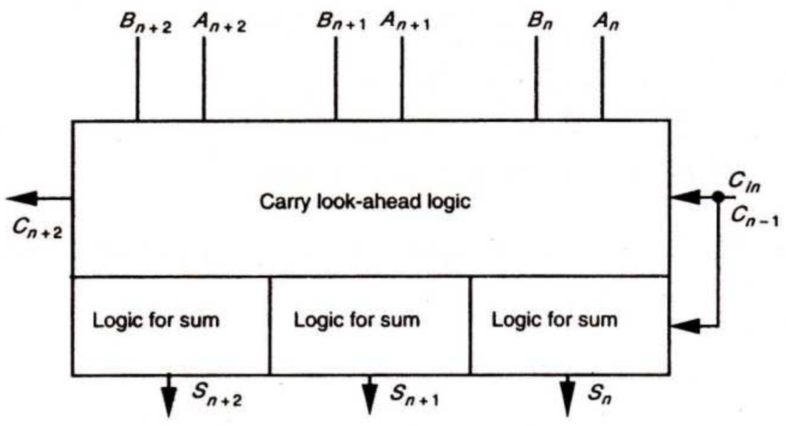
$$c_2 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_{in}$$

$$c_3 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_{in}$$

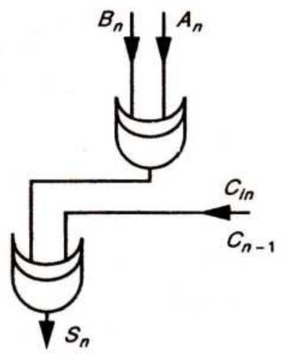


C_8
C_{out}
3-bit adder with carry look-ahead
C_5
3-bit adder with carry look-ahead
C_2
3-bit adder with carry look-ahead
C_{in}
S_8 S_7 S_6
S_5 S_4 S_3
S_2 S_1 S_0

(a) Partial carry look-ahead adder structure

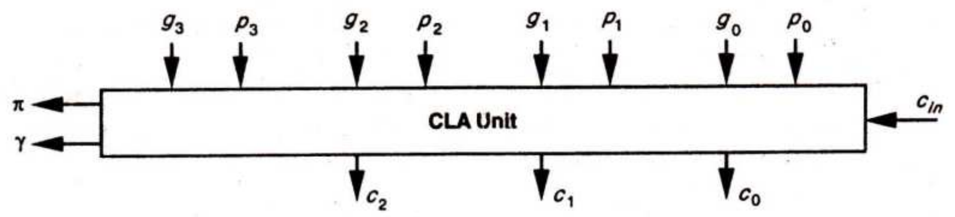


(b) Basic 3-bit adder cell with look-ahead



(c) Logic for sum

FIGURE 8.22 Carry look-ahead and ripple through compromise.



$\pi = p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_{in}$ $\qquad$ $\gamma = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 p_0$

FIGURE 8.23 4-bit block CLA unit.

In order to avoid a sequential propagation of carry signals between the blocks, we may generate additional signals π and γ such that

$$\pi = p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_{in} \text{ and } \gamma = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0$$

An important property of these signals is that c_3, the *carry out* of the block, is

$$c_3 = \gamma + \pi$$

This concept allows CLA techniques to be applied to the carry generation between blocks and for overall carry out as shown in Figure 8.24, which is the overall arrangement of a 16-bit CLA adder.

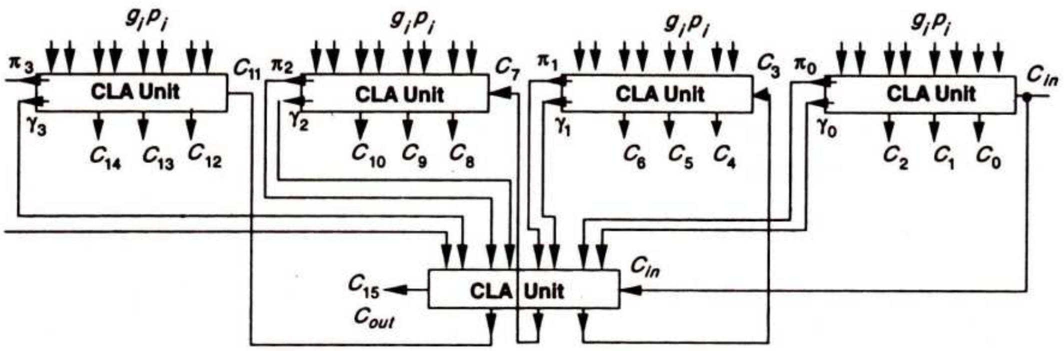


FIGURE 8.24 A 16-bit, 4 × 4 block CLA adder.

Further algebraic manipulation allows the expressions for carries within a four-bit block to be written

$$c_0 = g_0 + p_0.c_{in}$$
$$c_1 = g_1 + p_1.(g_0 + p_0c_{in})$$
$$c_2 = g_2 + p_2.(g_1 + p_1.g_0 + p_1.p_0.c_{in})$$
$$c_3 = g_3 + p_3.(g_2 + p_2.g_1 + p_2.p_1.g_0 + p_2.p_1.p_0.c_{in})$$

When implementing these circuits in silicon, each carry may be formed by one simple and very regular arrangement as indicated by Figure 8.25, which shows the formation of c_3. For each 4-bit CLA block, four such cells must be implemented, one for each carry c_0 to c_3, and an additional similar circuit is required to form γ.

In order to reduce this complexity, it is possible to use a dynamic logic technique known as 'Multiple Output Domino Logic'. Figure 8.26 illustrates the approach and is, in fact, a four-cell Manchester carry-chain.

8.4.3 A Comparison of Adder Enhancement Techniques

This section compares the three enhancement techniques we have discussed from the point of view of area occupied combined with performance. For the purpose of our study, we will compare three 32-bit adders—one carry select, one carry skip and one carry look-ahead. For convenience the carry select and carry skip adders will be assumed to be subdivided into equal size blocks. This must be so as a graduated sizing of blocks relies on an accurate knowledge of the gate delays—information which we do not have for this comparison. The adder cell to be used is required in two versions, one as in Figure 8.14 and a second version—with inverted inputs and carry output—as in Figure 8.27. In both cases, the delay between carry in and carry out is denoted by k_1 (the delay through one adder cell).

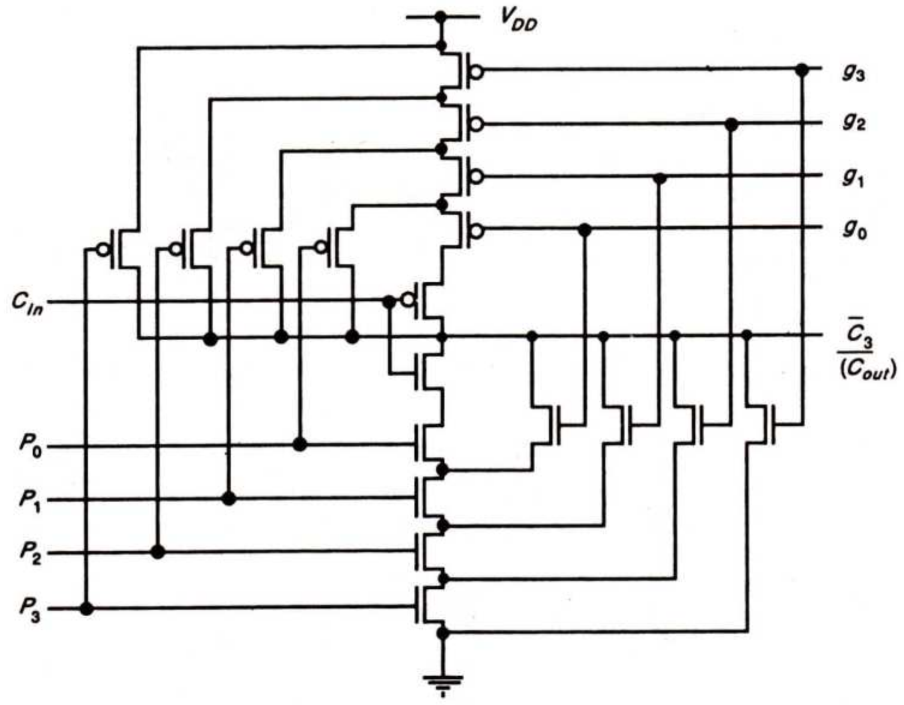


FIGURE 8.25 Generation of carry out (from 4-bits and carry in).

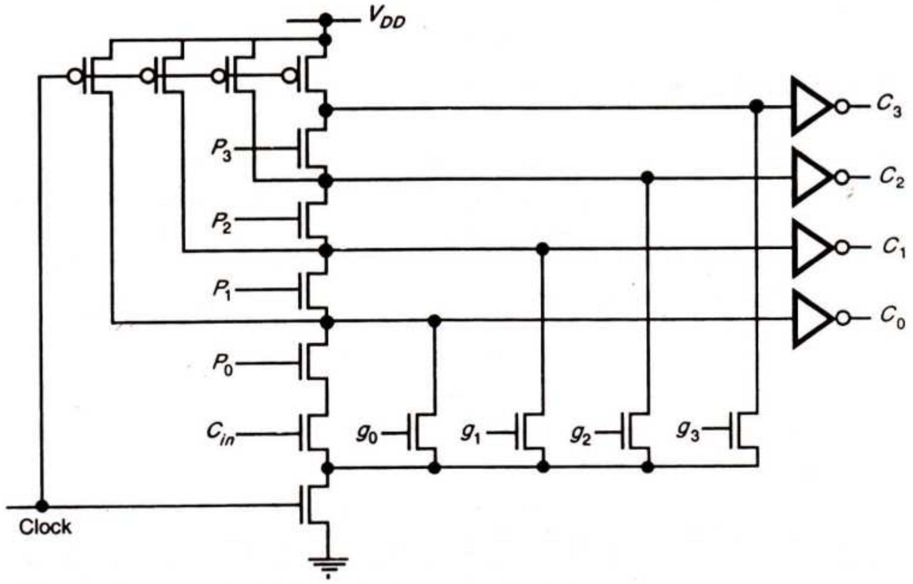


FIGURE 8.26 Four-cell Manchester carry-chain.

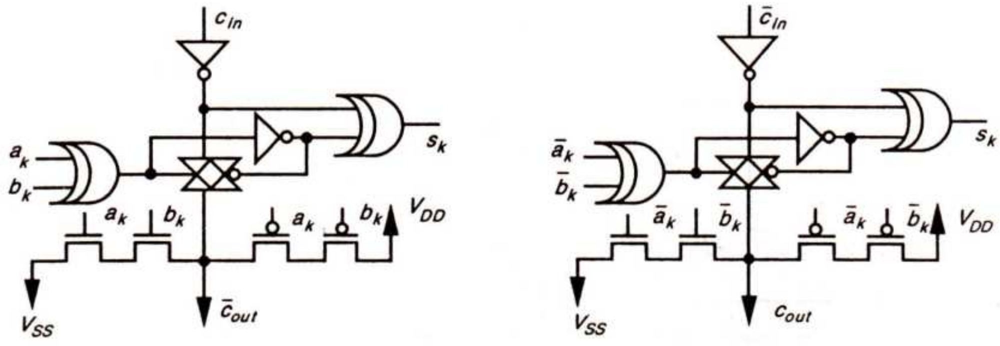


FIGURE 8.27 Adder cells with alternative input/output arrangement.

8.4.3.1 A 32-bit carry select adder assessment

The multiplexers to be used invert the signal and are based on a simple cell comprising one inverter and one transmission gate as shown for the 2-way multiplexer of Figure 8.28.

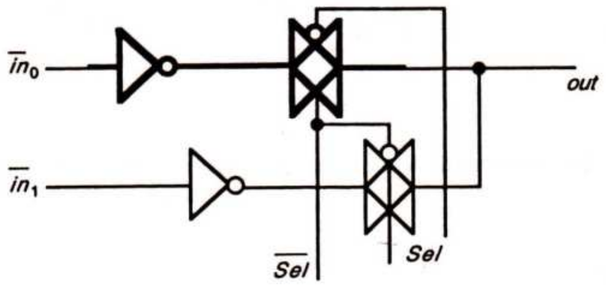


FIGURE 8.28 2-way multiplexer showing 'multiplexer cell' (bold lines).

Comparing this with the proposed adder cell, we may see that the multiplexer delay k_2 is the same as that for the adder cell so that $k_1 = k_2$ and, in consequence, the optimum block size evaluates as six. This does not divide exactly into 32, but we may choose to use four blocks of five cells and the remaining two blocks will then have six cells each as shown in Figure 8.29. The adder completion time is thus:

$$T = 5k_1 + 5k_2 = 10k$$

where $k = k_1 = k_2$.

The area of this 32-bit carry select adder is roughly twice that of a 32-bit ripple carry adder.

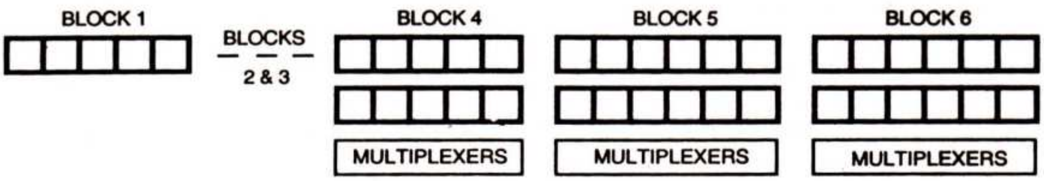


FIGURE 8.29 Arrangement of a 32-bit carry-select adder.

8.4.3.2 A 32-bit carry skip adder assessment

Once again, the cell delay k_1 and the multiplexer (as in Figure 8.28) delay k_2 may be assumed to be equal. In order to simplify the propagation time assessment, we will neglect the time taken to compute all the generate and propagate, as well as the block propagation signals, since they are all computed simultaneously and may be represented as an overhead $= k \approx k_1$.

Care must be taken to allow for the inversion of the carry signal, both in the adder cell and in the multiplexer. For this reason, the block size must be an even number of bits. Again, since the ratio between the cell delay and the multiplexer delay is assumed to be 1:1, we may write

$$k_1 = k_2 = k$$

also, since the ratio $k_1 = k_2$, the optimum block size is four cells so that there will be eight blocks of equal size.

The adder completion time is thus:

$$T = 4k_1 + 4k_2 + 6k_2 + k = 15k$$

where $k = k_1 = k_2$.

The area of this 32-bit carry skip adder is roughly one and a half that of a 32-bit ripple carry adder.

8.4.3.3 A 32-bit carry look-ahead adder assessment

Figure 8.30 represents the structure of a 32-bit carry look-ahead adder. For reasons of simplicity in presentation, each heavy interconnect line represents the interconnection of two signals, $(g_k.p_k)$ and $(\gamma_k.\pi_k)$. The fine interconnect lines are the carry signals.

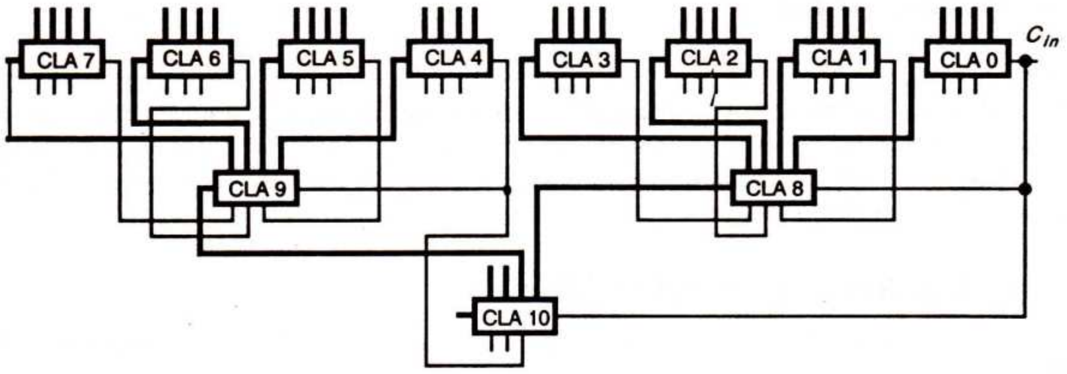


FIGURE 8.30 Arrangement of a 32-bit carry look-ahead adder.

Let the delay time of a CLA unit be k_3, then the completion time of the adder may be assessed as follows:

At time k_3 : $(\gamma_k \pi_k)$ for CLA 0–7 are set.

At time $2k_3$: $(\gamma_k.\pi_k)$ for CLA 8 and 9 are set; c_4, c_8, and c_{12} are set by CLA 8.

At time $3k_3$: c_{16} is set by CLA 10; using c_4, c_8, and c_{12}, CLA 1, CLA 2 and CLA 3 set their carry out.

At time $4k_3$: c_{20}, c_{24}, and c_{28}, are set by CLA 9.

At time $5k_3$: Using c_{20}, c_{24}, and c_{28}, CLA 5, CLA 6 and CLA 7 set their carry out.

Therefore, overall time $T = 5k_3$.

The exact value of k_3 depends on the actual CLA adder element arrangement and on the layout used, but, allowing for three levels of logic, it could be conservatively estimated as $1.5k_1$ to $2.0k_1$, where k_1 is the delay of the simple adder cell used before in the carry select and skip adders. If this is a reasonable assumption then, in comparison with the other evaluations, overall time T is given by

$$T = 7.5k \text{ to } 10k$$

However, noting the unused inputs of CLA 10 (Figure 8.30), it may be seen that a 64-bit CLA adder could be accommodated within the same overall time delay. Since the CLA cells are considerably more complex than the adder cells used in the carry select and carry skip adders, there will be a penalty in the area occupied. This is difficult to evaluate without detailed design work, but the area occupied will be several times greater than for a 32-bit ripple carry adder.

This concludes the consideration of adder circuitry. In the design of ALUs and digital processors generally, the adder is the most important circuit and is able to directly accommodate additions, subtractions and comparisons, together with a range of logical operations. Another common arithmetic requirement is for multiplication and it will be seen that the adder has an important role to play in the architecture of many multipliers.

8.5 MULTIPLIERS

A study of computer arithmetic processes will reveal that the most common requirements are for addition and subtraction, but that there is also a significant need for a multiplication capability. Thus, a brief overview of some common approaches to this problem is given in this section. Although division is obviously useful, it is a much less common requirement and will not be dealt with in this text.

8.5.1 The Serial-parallel Multiplier

This multiplier is the simplest one, the multiplication being considered as a succession of additions.

If $A = (a_n\, a_{n-1}\, a_{n-2} \,.\,.\,.\,.\,.\,.\,.\,.\,.\,.\, a_0)$ and

$B = (b_n\, b_{n-1}\, b_{n-2} \,.\,.\,.\,.\,.\,.\,.\,.\,.\,.\, b_0)$

then the product $A.B$ may be expressed as

$$A.B = (A.2^n.b_n + A.2^{n-1}.b_{n-1} + A.2^{n-2}.\, b_{n-2} \,.\,.\,.\,.\,.\,.\,.\,.\,.\,.\, A.2^0.b_0)$$

A possible form of this adder for multiplying four-bit quantities, based on this expression, is set out in Figure 8.31. Note that *D* indicates a *D* flip-flop simple and *FA* indicates a full adder—or adder bit slice. Number *A* is entered in the right-most 4-bits of the top row of *D* flip-flops which are connected to three further *D* flip-flops to form a 7-bit shift register to allow the multiplication of number *A* by 2^1, 2^2 ... 2^n, thus forming the *partial product* at each stage of the process.

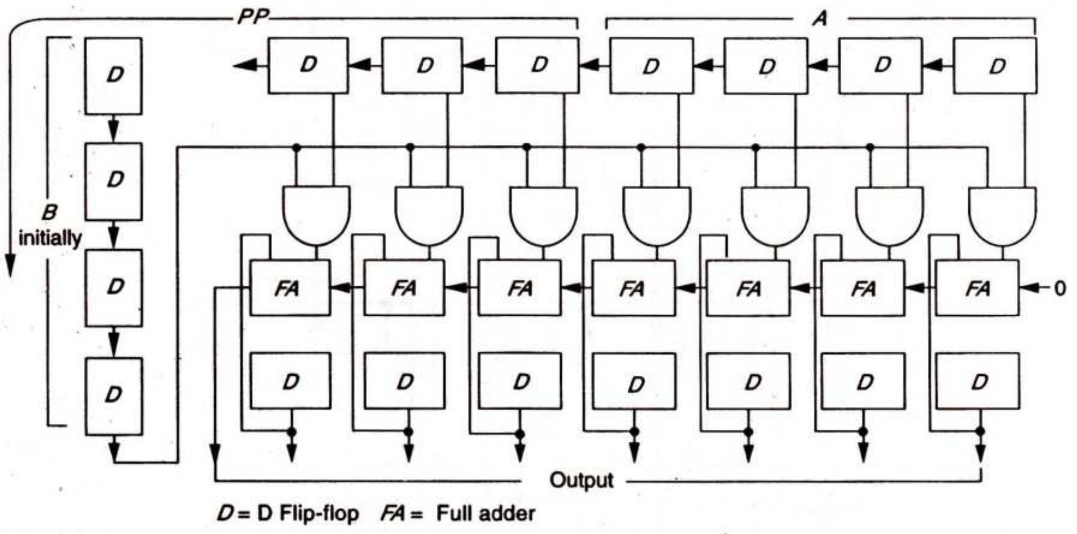


FIGURE 8.31 Arrangement of a 4-bit serial-parallel multiplier.

In some cases, it may be easier to right shift the contents of the *Accumulator*—(bottom row of *D* flip-flops) rather than left shifting *A*. This approach can be used to eliminate the least significant bits of the product if so desired.

A further reduction in hardware can result from noting that the three most significant bits of the partial product are set to zero initially, and are used only one by one as the shifting of *A* proceeds. These three bits can therefore be used to hold three bits of number *B* initially, thus saving three *D* flip-flops.

The structure under discussion here is suited only to positive or unsigned operands. If the operands are negative and twos complement encoded, then:

1. The most significant bit of *B* will have a negative weight and so a subtraction must be performed as the last step.
2. The most significant bit of *A* must be replicated since operand *A* must be expanded to 2*N* bits.

8.5.2 The Braun Array

A relatively simple form of parallel adder is the Braun array (see Figure 8.32). All partial products $A.b_k$ are computed in parallel, then collected through a cascaded array of carry save

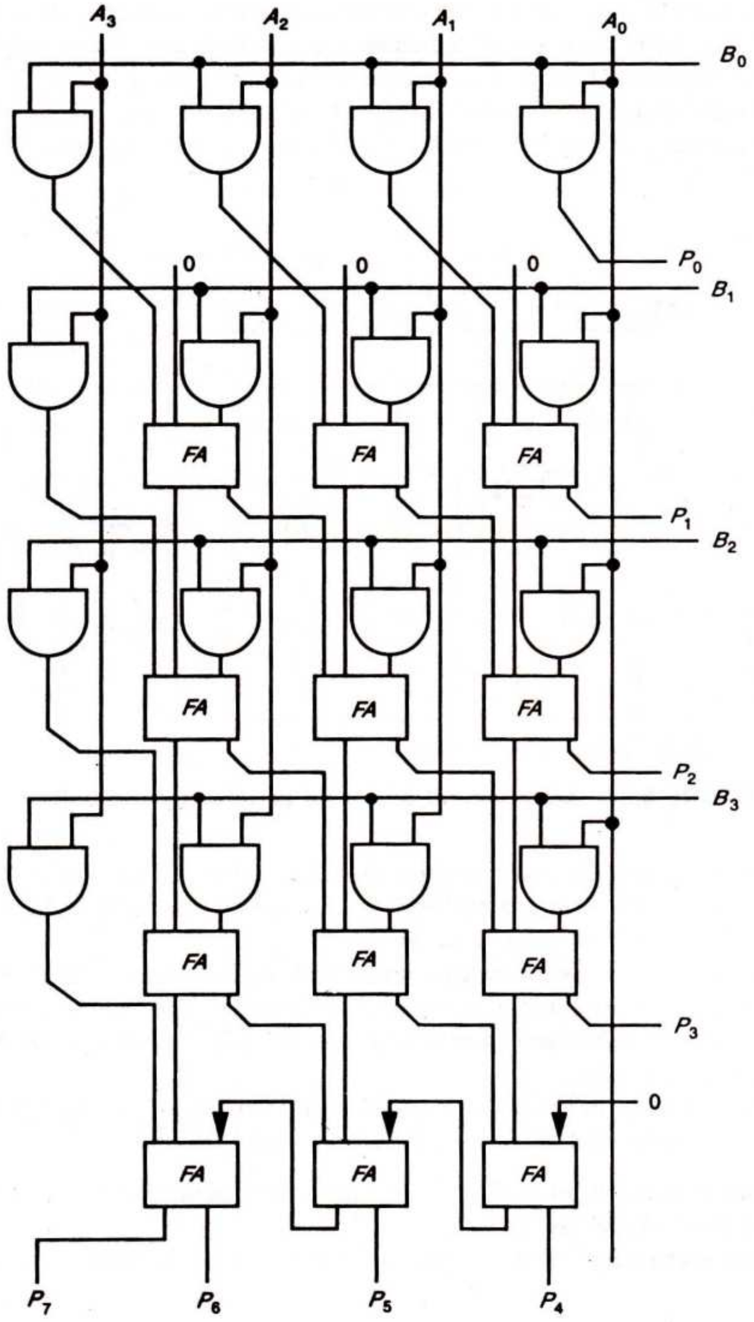


FIGURE 8.32 A 4-bit Braun multiplier.

adders. At the bottom of the array, an adder is used to convert the carry save form to the required form of output. Completion time is fixed by the depth of the array, and by the carry propagation characteristics of the adder. Notice that this multiplier is suited only to positive operands. Negative operands can be handled, for example, by the Baugh-Wooley multiplier which now follows.

8.5.3 Twos Complement Multiplication Using the Baugh-Wooley Method

This technique has been developed to design multipliers that are regular in structure and suited for twos complement numbers.

Let us consider two numbers A and B:

$$A = (a_{n-1} \ldots\ldots a_0) = -a_{n-1}.2^{n-1} + \sum_0^{n-2} a_i.2^i$$

$$B = (b_{n-1} \ldots\ldots b_0) = -b_{n-1}.2^{n-1} + \sum_0^{n-2} b_i.2^i$$

The product $A.B$ is given by:

$$A.B = a_{n-1}.b_{n-1}.2^{n-2} + \sum_0^{n-2}\sum_0^{n-2} a_i.b_j.2^{i+j} - a_{n-1}\sum_0^{n-2} b_i.2^{n+i-1} - b_{n-1}\sum_0^{n-2} a_i.2^{n+i-1}$$

If we use this form, it may be seen that subtraction operations are needed as well as addition. However, the negative terms may be rewritten, for example:

$$a_{n-1}\sum_0^{n-2} b_i.2^{n+i-1} = a_{n-1}.\left(-2^{n-2} + 2^{n-1} + \sum_0^{n-2} \overline{b}_i.2^{n+i-1}\right)$$

Using this approach, $A.B$ becomes

$$A.B = a_{n-1}.b_{n-1}.2^{n-2} + \sum_0^{n-2}\sum_0^{n-2} a_i.b_i.2^{i+j} + b_{n-1}\left(-2^{n-2} + 2^{n-1} + \sum_0^{n-2} \overline{a}_i.2^{n+i-1}\right)$$

$$+ a_{n-1}\left(-2^{n-2} + 2^{n-1} + \sum_0^{n-2} \overline{b}_i.2^{n+i-1}\right)$$

This equation may be put in a more convenient form by recognizing that

$$-(b^{n-1} + a^{n-1}).2^{2n-2} = -2^{2n-1} + (\overline{a_{n-1}} + \overline{b_{n-1}}).2^{2n-2}$$

Thus, AB is given by

$$A.B = 2^{2n-1} + (\overline{a_{n-1}} + \overline{b_{n-1}} + a^{n-1}.b^{n-1}).2^{2n-2}$$

$$+ \sum_{0}^{n-2}\sum_{0}^{n-2} a_i.b_j.2^{i+j} + (a_{n-1} + b_{n-1}).2^{n-1}$$

$$+ \sum_{0}^{n-2} b_{n-1}.\overline{a}_i.2^{n+1-j} + \sum_{0}^{n-2} a_{n-1}.\overline{b}_i\, 2^{n+i-1}$$

Since A and B are n-bit operands, their product may extend to $2n$-bits. The first, most significant, bit is taken into account by the first term -2^{2n-1} which is fed to the multiplier as a 1 in the most significant cell. The Baugh-Wooley arrangement is set out in Figure 8.33.

In serial-parallel multipliers there are as many idle clock cycles as there are 0s in the multiplicand and the same situation applies in Braun and Baugh-Wooley arrays. For this reason, it may be useful to introduce pipelining concepts between successive lines of the array. The clock speed of the pipeline is limited by the speed of the output adder, but it is possible to introduce further pipelining between the adder cells giving rise to the systolic array multiplier.

8.5.4 A Pipelined Multiplier Array*

Many parallel multipliers are iterative arrays. Some of these are carry-ripple structures with no storage elements, in which a given result must be output before new data words can be input. Such multipliers can be pipelined by introducing latches at appropriate positions in the array.

An example is a parallel multiplier based on *systolic array principles* as in Figure 8.34. It comprises a diamond-shaped array of latched, gated full adder cells, connected only to immediately adjacent cells. This has practical advantages as no broadcasting of data right across the multiplier array occurs.

With multiplicand X, multiplier Y and product P, the kth bit of each partial product $X_{k-i}.\ y_i$ is formed in one of the cells in the kth vertical column of the array.

The kth bit of the product

$$P_k = \sum_{i=0}^{k} X_{k-i} \cdot y_i$$

is formed by letting these components accumulate as p_k passes down the column. Carries generated at each stage in the array are passed to the left (next most significant column).

The residual carry bits passing across the lower left-hand boundary of the diamond must be added into the partial product sum to complete the multiplication. This is achieved with half of the above array placed at the lower left-hand boundary, retaining the iterative structure.

* J.V. McCanny and J.G. McWhirter, 'Completely iterative, pipelined multiplier array suitable for VLSI', *IEE Proc*, vol. 129, pt. G, no. 2, 40–46. This structure was designed by P. Evans as part of a VLSI course at the University of Adelaide, South Australia.

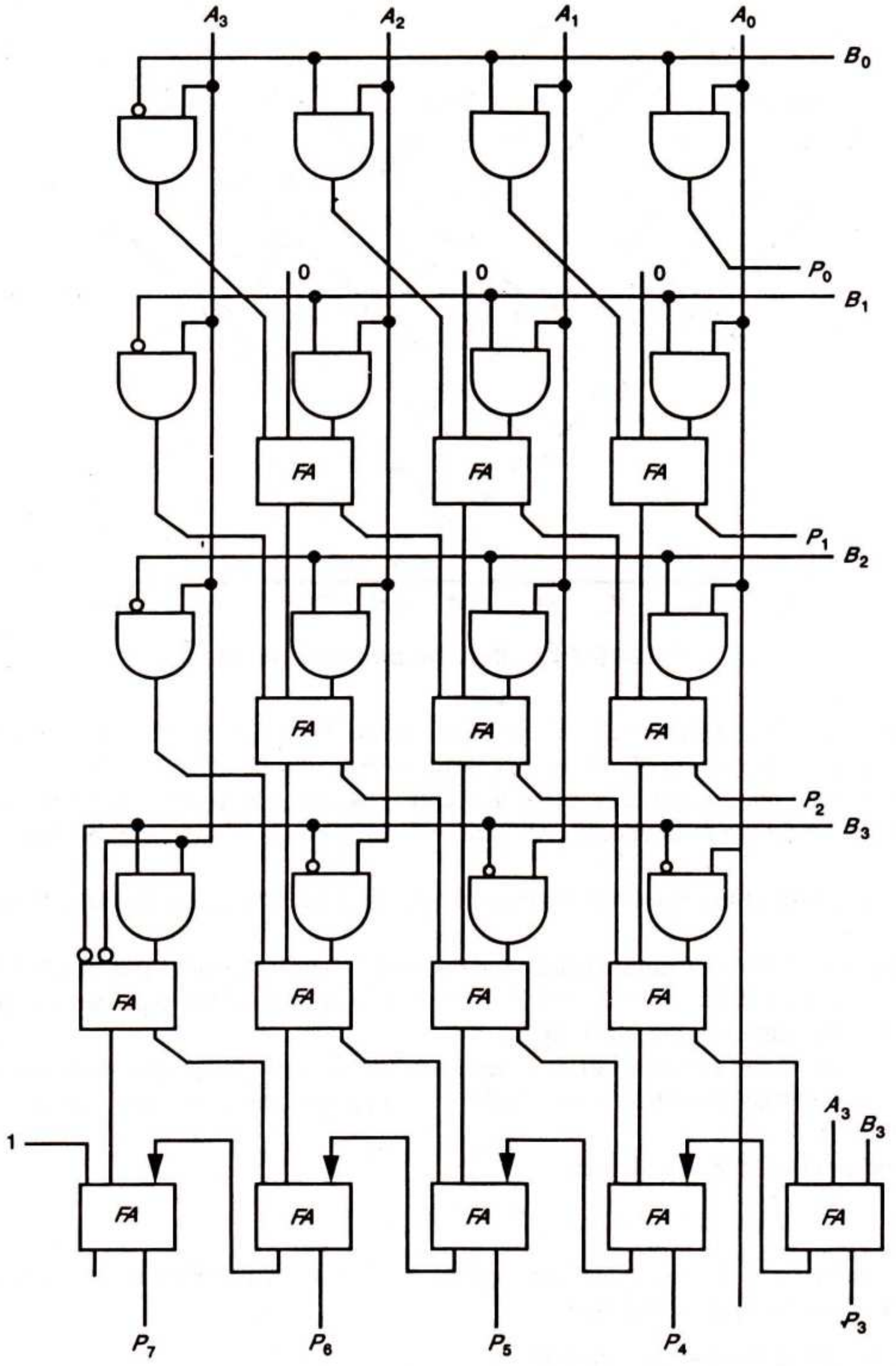


FIGURE 8.33 A 4-bit Baugh-Wooley multiplier.

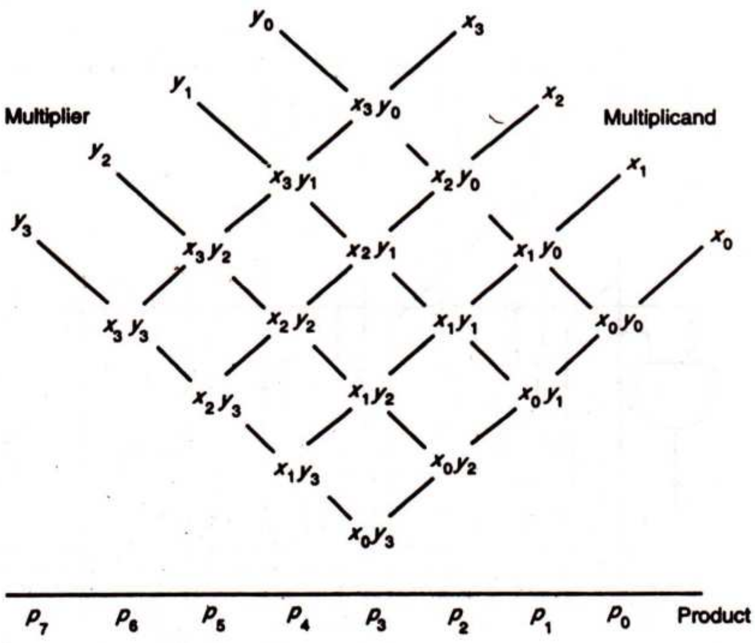


FIGURE 8.34 Systolic array multiplier.

This gives the general structure shown in Figure 8.35. For an n-bit $\times$ n-bit multiplier, $^1/_2(3n + 1)n$ cells are required. There is a further requirement of $3n^2$ latches to skew and deskew the input and output data. Note that each cell connects to six other cells, provided that it is not on the array boundary. All sum and carry inputs at the array boundary are set to zero.

The structure of the basic cell is shown in Figure 8.36. The gating function for unsigned numbers is $x.y$.

The delay of one operation through the pipeline is $3n$ clock cycles (i.e. it takes $3n$ clock cycles to obtain a product after X and Y are input). However, if the pipeline is kept full, a product will be output every clock cycle.

The clock period can be short as it must account for only the propagation time through one cell. The multiplier is thus a very high throughput structure (i.e. low average time per multiplication).

If the product $X.Y$ is rewritten

$$XY = x_{n-1}.2^{n-1}.\overline{Y} + x_{n-1}.2^{n-1} + \tilde{x}.Y$$

where $\tilde{x}$ is the $(n - 1)$ least significant bits of X, then the structure can be used for twos complement numbers, provided that:

1. The gating function is replaced by

$$(y \oplus d).x$$

where $d = 1$ for all cells on the upper left-hand boundary and $d = 0$ elsewhere.

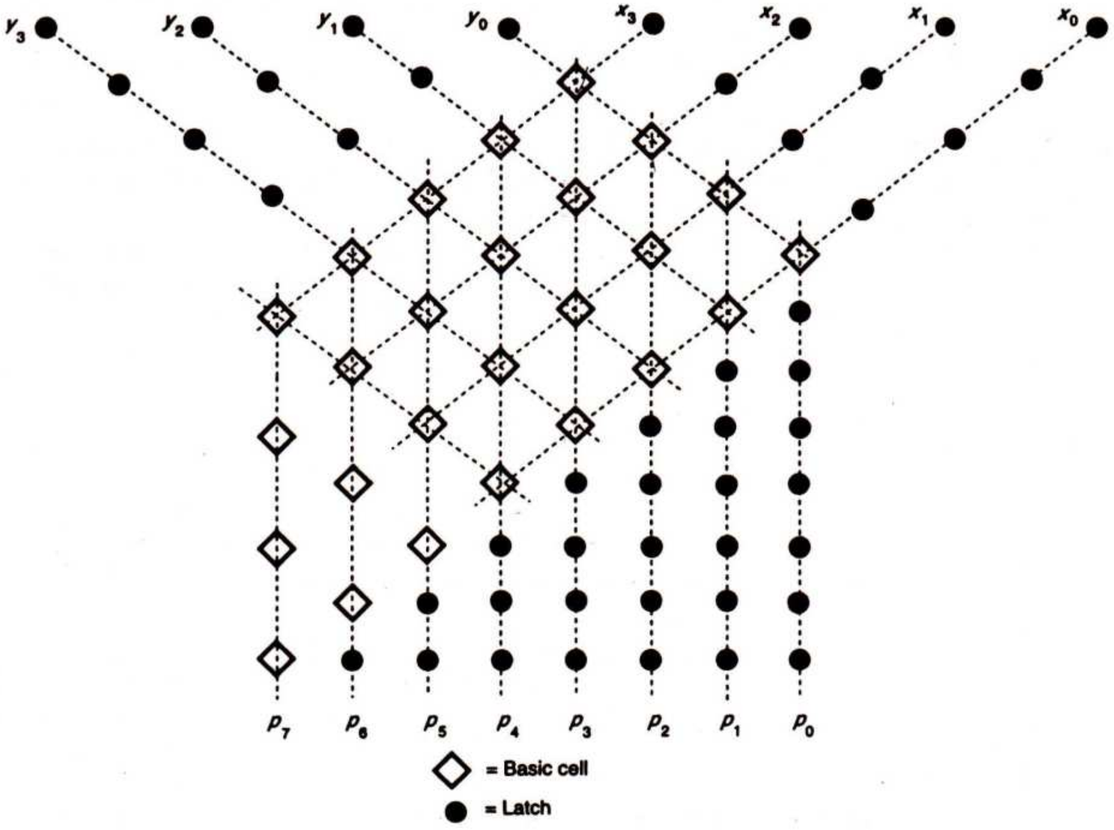
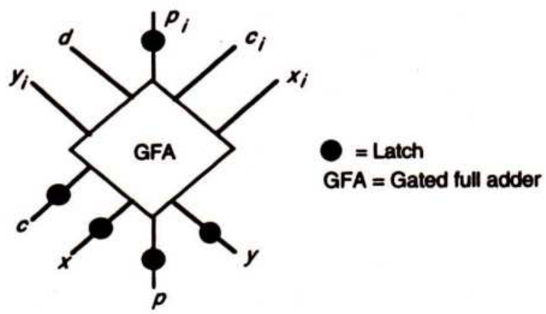


FIGURE 8.35 Multiplier structure.



Note: Where p_i = partial product sum in
p = partial product sum out
c_i = carr~~y in~~
c = carry out
d = line required for twos complement operation

FIGURE 8.36 Basic cell.

2. The value of x_{i-1} is fed to the carry input c_i as well as to the normal input x_i of the cell in the top row of the array.
3. Y is sign extended and suitably delayed sign extensions are input to left boundary y_i inputs.

The full adder chosen was a transmission gate adder because of its speed and because it generates the sum and the carry in equal time. The latches chosen were dynamic shift registers as the structure will be continuously clocked.

The timing diagram (Figure 8.37) illustrates the performance of the 8-bit version. After the initial delay of about 1.2 μsec, the output products are available at 50 nsec intervals.

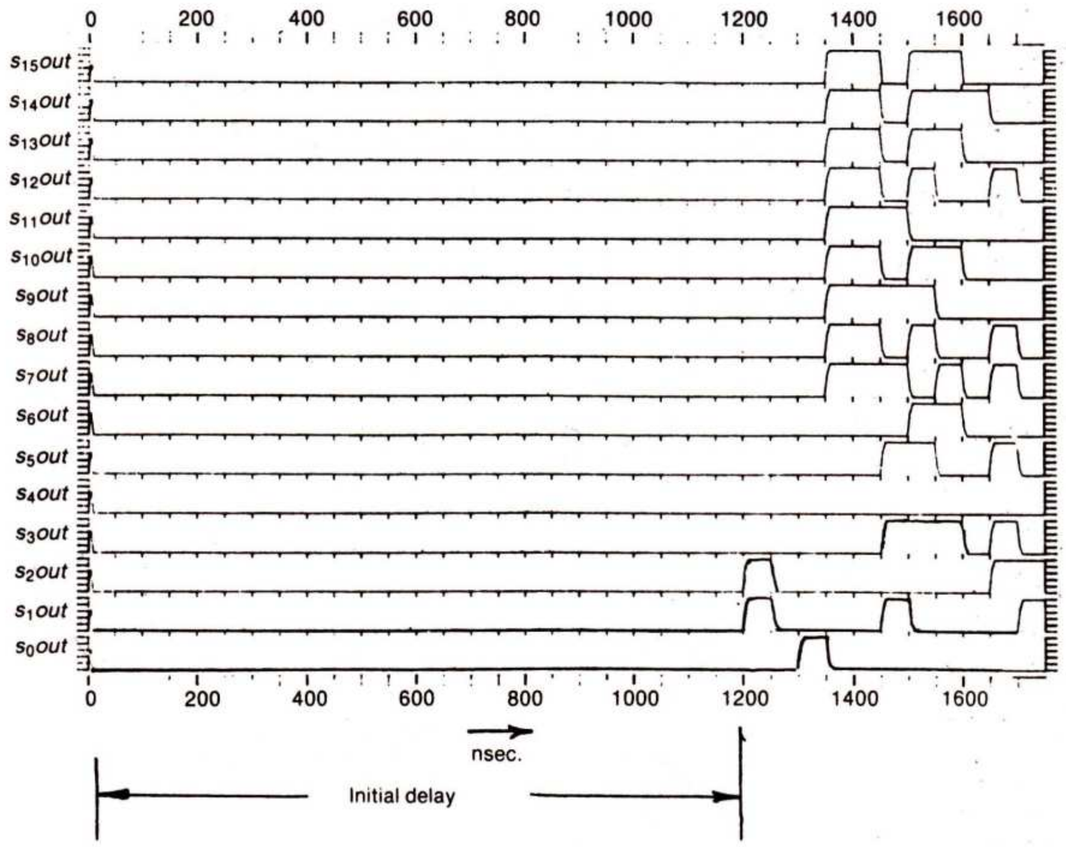


FIGURE 8.37 Performance of an 8-bit multiplier.

8.5.5 The Modified Booth's Algorithm

Another approach which avoids having many idle cells in a cellular multiplier as well as reducing the number of cycles compared with the serial-parallel multiplier is the use of the so-called modified Booth's algorithm. In principle, the modified algorithm requires rewriting the multiplicand in such a way that half the bits are 0. Clearly, this is possible only by using a special number system.

This converts a signed standard twos radix number into a number system where the digits are in the set {–1, 0, 1}. In this system any number may be written in several forms, that is, the system has redundancies.

Let us consider a number $B = b_{n-1}\, b_{n-2}\, ...\, b_1 b_0$ written in twos complement form:

$$B = -b_{n-1}.2^{n-1} + \sum_{k=0}^{n-2} b_k.2^k$$

which may be rewritten as

$$B = \sum_{k=0}^{n-2-1} (b_{2k-1} + b_{2k} + 2b_{2k+1})2^k$$

with $b_1 = 0$.

In this equation, the term in the brackets is in the set {–2, –1, 0, 1, 2}, so it cannot be equal to 3 or –3. In other words, after rewriting B through the modified algorithm, each pair of digits can only take the following forms: [–1, –1], [0, –1], [0, 0], [0, 1], [1, 1], that is (–2, –1, 0, 1, 2). Another consequence of the modified Booth's algorithm is that the sign of the numbers is implicitly taken into account.

8.5.5.1 Application to multiplication

Consider two numbers A and B. Encoding B through the modified algorithm converts its form to B' with digits –2, –1, 0, 1, 2. In this form there will be half the number of digits in B in B'. The digits of B' are scanned, and at each step, A is multiplied by –2, –1, 0, 1, or 2. The different cases are given in Table 8.2. For example, if bit b_{2k} of B is 0 and bits b_{2k+1} and b_{2k-1} are 1 and 0 respectively, then we must add $-2A$ to the sum forming the product in the accumulator.

TABLE 8.2 Modified Booth's multiplication

b_{2k+1}	b_{2k}	b_{2k-1}	*A multiplied by*
0	0	0	0
0	0	1	+1
0	1	0	+1
0	1	1	+2
1	0	0	–2
1	0	1	–1
1	1	0	–1
1	1	1	0

One possible implementation of a circuit to implement the requirements of Table 8.2 is set out in Figure 8.38.

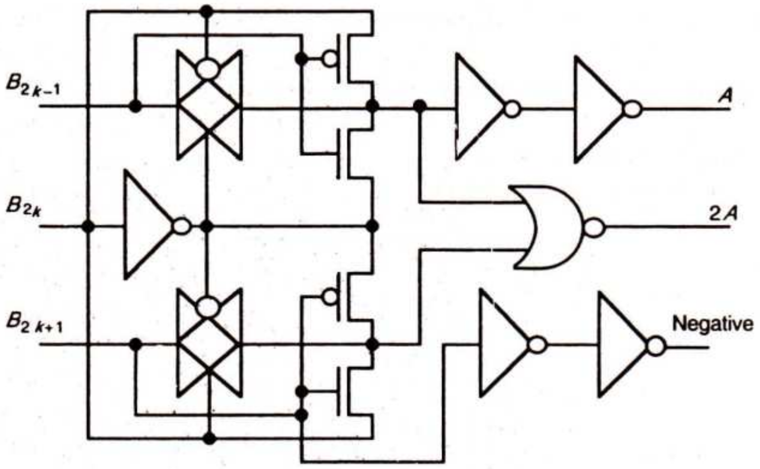


FIGURE 8.38 Booth encoder.

8.5.6 Wallace Tree Multipliers

Wallace trees were first introduced in 1964 (Wallace, 1964) in order to design multipliers whose completion time grows as the logarithm of the number of bits to be multiplied. The simplest Wallace tree is the full adder cell (three inputs—two outputs). More generally, an n input Wallace tree, as in Figure 8.39, is an n-input operation with $log_2(n)$ outputs, such that the value of the output word is equal to the number of '1'*s* in the input word (consider the full adder in this context). The input bits and the least significant bit of the output word have the same weight, as shown in Figure 8.39.

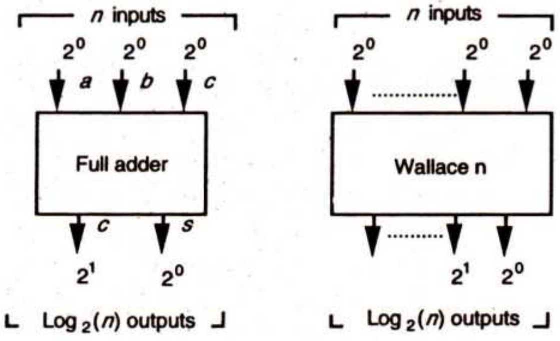


FIGURE 8.39 Wallace tree elements.

An important property of Wallace trees is that they may be constructed from adder cells. Furthermore, the number of adder cells needed grows as the logarithm $log_2(n)$ of the number of input bits n. In a Braun or a Baugh-Wooley multiplier with a ripple carry adder, the completion time for multiplication is proportional to $2n$. If the collection of the partial products is made through Wallace trees then the completion time for getting a result, in carry save notation, should be proportional to $log_2(n)$.

Figure 8.40 shows a seven-input adder for each weight and Wallace trees are used until only two bits of each weight remain. These bits are then added using the classical two-input adder. Wallace trees may be applied to multipliers in several ways.

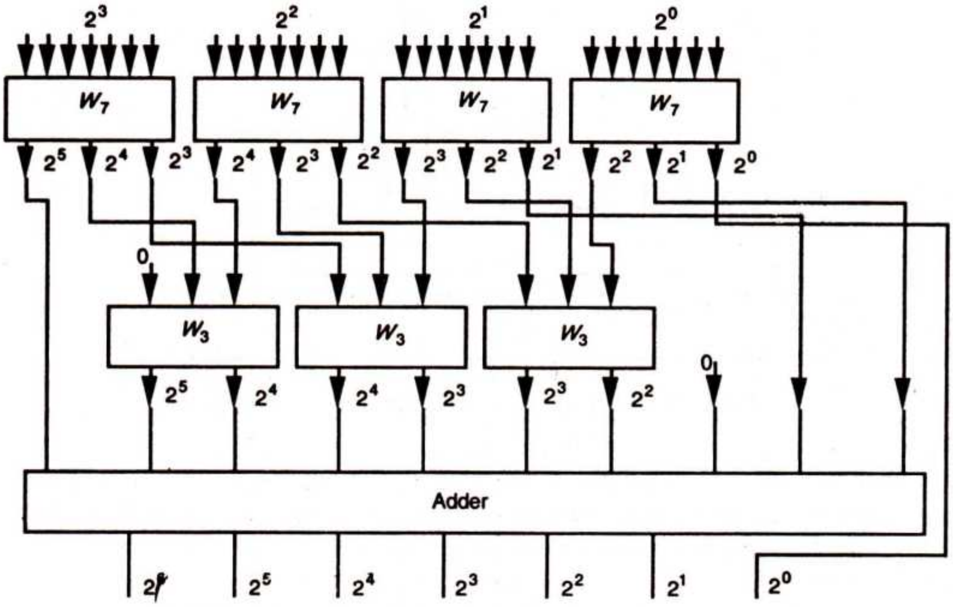


FIGURE 8.40 Example of the Wallace tree approach.

8.5.7 Recursive Decomposition of the Multiplication

One method, based on recursive decomposition of the multiplication, consists of partitioning the operands. For instance, if A and B are $2p$-bit numbers, then A (also B) may be cut into two parts A_0 and A_1 respectively, so that

$$A = 2^p . A_1 + A_0$$

$$B = 2^p . B_1 + B_0$$

The product $A.B$ is

$$A.B = 2^{2p}.A_1B_1 + 2^{6p}.(A_1.B_0 + A_0.B_1) + A_0.B_0$$

Using this method, four p-bit multipliers are used to compute $A_1.B_1.A_0.B_1.A_1.B_0$ and $A_0.B_0$. The results are collected through Wallace trees. The arrangement of a multiplier of this type, with 8-bit input words, is shown in Figure 8.41; the interconnections have been simplified for clarity. A_0, B_0, A_1 and B_1, are in fact 4-bit numbers and the outputs of the multiplier are 8-bit products. In this figure it has been assumed that the multipliers each contain an adder so that each result is not in carry save notation and thus eight adder cells (three-input Wallace

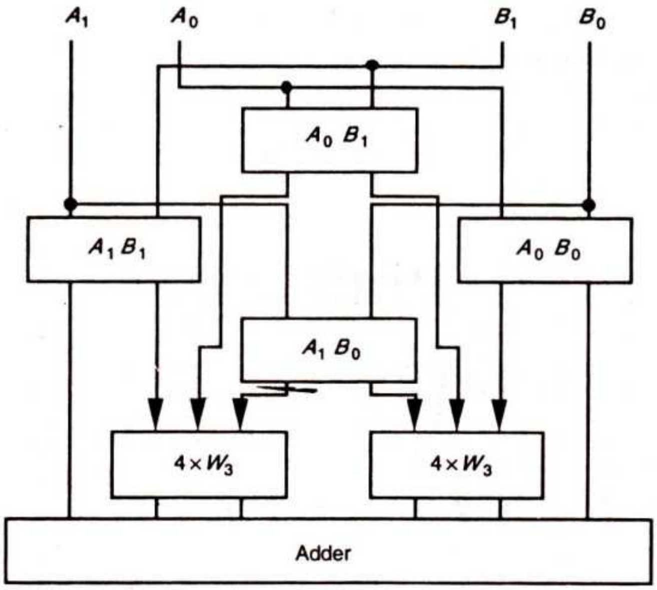


FIGURE 8.41 8-bit input word multiplier arrangement.

trees) are used to collect bits of the same weight. For instance, the multipliers denoted $A_0.B_1$, $A_1.B_0$ and $A_0.B_0$ give bits of weight 4, 5, 6, and 7. For each of these weights, three bits (as many as there are multipliers) must be added, and thus an adder cell must be used to reduce the number of bits of the same weight to two.

8.5.8 Dadda's Method

Another approach consists in computing all the partial products—like the Braun array—and then collecting all the bits of the same weight through Wallace trees. This is equivalent to partitioning the input operands to work with 1-bit multipliers (i.e. *And* gates). In 1965 L. Dadda developed a technique to build the Wallace layer using the minimum number of adder cells.

Consider k bits of the same weight i coming from k partial products. When adding these k bits by a k-input Wallace tree, bits of weights $i + 1$, $i + 2$, ... etc. appear which must in turn be added to the bits of weights $i + 1$, $i + 2$, ... coming from other partial products. Dadda's method consists in handling all bits in the collecting Wallace layer so as to minimize the number of adder cells as well as the critical path between the partial product generation and the final addition. All the developments of this technique may be found in the reference (Dadda, 1965). In conclusion, Wallace tree multipliers should be used only for large operands and where the performance is critical since the arrangement results in poor regularity due to the routing area needed to collect the partial products.

8.6 OBSERVATIONS

This chapter has provided possible designs for the arithmetic subsystem forming part of the complete data path we are designing. Both the subsystems so far designed have comprised only combinational logic with the exception of possible storage requirements at the 'Sum' output of the adder. The third subsystem, to be designed next, will introduce a need for memory or storage and this leads to a review of some possible memory elements and relevant characteristics.

8.7 TUTORIAL EXERCISES

1. Referring to Figure 8.12, design switches and other logic as necessary to implement the functions performed by the mechanical switches drawn in Figure 8.12. Work out the control lines needed to enable the ALU to perform add, subtract, logical *And*, logical *Or*; logical *Exclusive-Or*; and logical *Equality* operations.
2. Draw a bounding box representation with all inlet and output points shown (as in Figure 8.10) for the *logic circuitry* of an adder, using CMOS multiplexers (Figure 8.4) and CMOS inverters as suggested in Figure 8.9. You may wish to proceed as follows.

   Continue the design of a standard CMOS adder element (as represented in stick diagram form in Figure 8.5) by working out a layout for the complete inverter block and then representing it as a bounding box with inlet and outlet points indicated by layer and position. *Hint:* Design a suitable mask layout for the CMOS inverters and then represent each inverter circuit in bounding box form—with inlet and outlet points—so that only one inverter needs to be drawn in detail in setting out your layout.

   Interconnect the inverter block bounding box with CMOS multiplexer-based adder logic (as in Figure 8.4). Work out an accurate bounding box representation for the complete adder element showing inlet and outlet points, etc., by position and layer.
3. What are the overall dimensions of a 4-bit CMOS adder? Using the bounding box representations draw an accurate floor plan of the whole 4-bit adder (as in Figure 8.11) showing position and layer of inlet and outlet points.
4. Carry out the design of a 4-bit CMOS carry look-ahead adder up to stick diagram form. Then determine what standard cells are needed and design a mask layout for each.

8.8 REFERENCES

Dadda, L. (1965, March) 'Some schemes for parallel multipliers', *Alta Frequenza*, Vol. 19.

Guyot, A., Hochet, B., and Muller, J.M. (1987, October) 'A way to build efficient carry-skip adders', *IEE Trans. on Computers,* Vol. C–36.

Hotta, T. et al., (1986, October) 'CMOS/Bipolar circuit for 60 MHz digital processing', *IEEE Journal of Solid State Circuits*, 803–13, Vol. 21, No. 5.

Muller, J.M. (1989) *Arithmétique des Ordinateurs*, Editions Masson, Collection Etudes et Recherches en Informatique, Paris.

Wallace, C.S. (1964, February) 'A suggestion for a fast multiplier', *IEEE Trans. on Electronic Computers*, 14–17.

Wang, I.S. and Fisher, A.L. (1989, April) 'Ultrafast compact 32-bit CMOS adders in multiple-output domino logic', *IEEE Journal of Solid State Circuits*, Vol. 24.