UNIT-I (MOS TRANSISTOR THEORY)

The MOS transistor- Current Voltage Relations- Threshold Voltage- Second order effects- Capacitances in MOSFET - Scaling of MOS circuits - Review of CMOS - DC characteristics - Dynamic behaviour- Power consumption.

Introduction :

The MOSFET – Metal Oxide FET

As well as the Junction Field Effect Transistor (JFET), there is another type of Field Effect Transistor available whose Gate input is electrically insulated from the main current carrying channel and is therefore called an **Insulated Gate Field Effect Transistor** or **IGFET**. The most common type of insulated gate FET which is used in many different types of electronic circuits is called the **Metal Oxide Semiconductor Field Effect Transistor** or **MOSFET** for short.

The **IGFET** or **MOSFET** is a voltage controlled field effect transistor that differs from a JFET in that it has a "Metal Oxide" Gate electrode which is electrically insulated from the main semiconductor n-channel or p-channel by a very thin layer of insulating material usually silicon dioxide, commonly known as glass. This ultra thin insulated metal gate electrode can be thought of as one plate of a capacitor. The isolation of the controlling Gate makes the input resistance of the **MOSFET** extremely high way up in the Mega-ohms (MΩ) region thereby making it almost infinite.

As the Gate terminal is isolated from the main current carrying channel "NO current flows into the gate" and just like the JFET, the MOSFET also acts like a voltage controlled resistor were the current flowing through the main channel between the Drain and Source is proportional to the input voltage. Also like the JFET, the MOSFETs very high input resistance can easily accumulate large amounts of static charge resulting in the **MOSFET** becoming easily damaged unless carefully handled or protected.

Characteristics of MOSFET :

1. Bilaterally Symmetric device
2. Unipolar device
3. High Input Impedance
4. Voltage Controlled
5. Self Isolated

MOS transistor performs very well as a switch; it introduces very few parasitic effects, simple integration density, simple manufacturing process which make it possible to produce large and complex circuits in an economical way.

MOS Transistor Types and Construction :

MOSFETs are three terminal devices with a Gate, Drain and Source and both P-channel (PMOS) and N-channel (NMOS) MOSFETs are available. Figure 1 represents the conduction characteristics of MOS transistors.

- n-Channel MOS : Majority carriers are electrons.
- p-Channel MOS : Majority carriers are holes.

- Positive/negative voltage applied to the gate (with respect to substrate) enhances the number of electrons/holes in the channel and increases conductivity between source and drain.

- V*t* defines the voltage at which a MOS transistor begins to conduct. For voltages less than V *t* (threshold voltage), the channel is cut off.

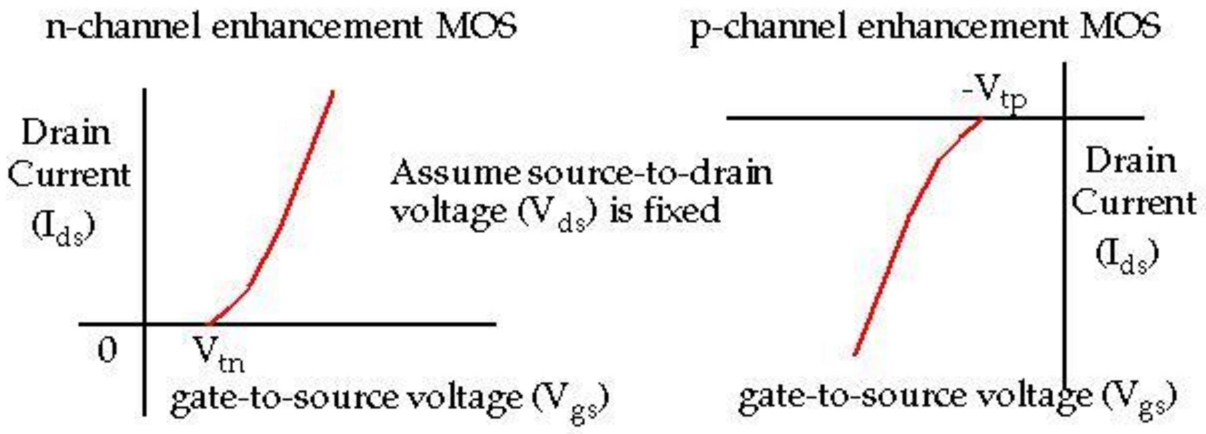


Figure 1. Conduction characteristics of MOS transistors.

The main difference this time is that MOSFETs are available in two basic forms:

- 1. Depletion Type – the transistor requires the Gate-Source voltage, (VGS) to switch the device "OFF". The depletion mode MOSFET is equivalent to a "Normally Closed" switch.
- 2. Enhancement Type – the transistor requires a Gate-Source voltage, (VGS) to switch the device "ON". The enhancement mode MOSFET is equivalent to a "Normally Open" switch.

The symbols and basic construction for both configurations of MOSFETs are shown below.

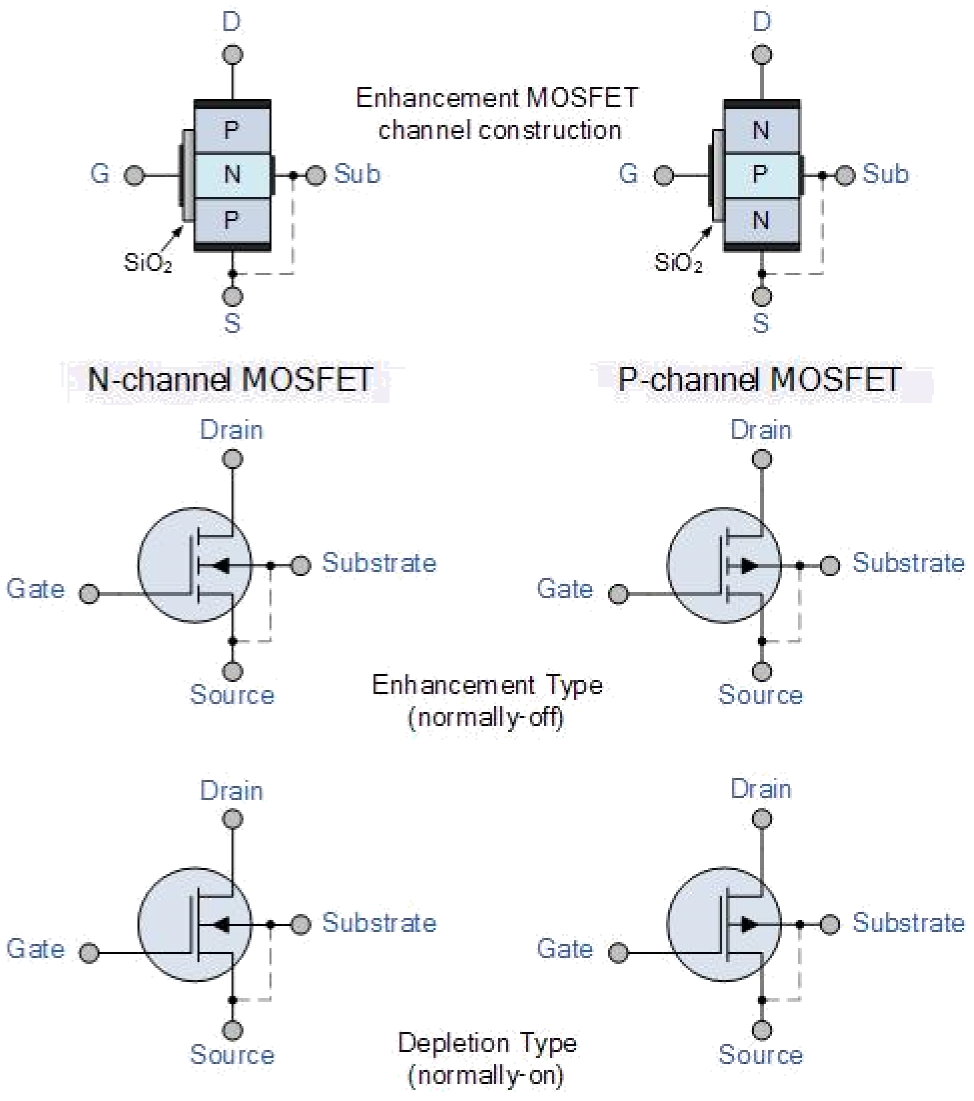

D
Enhancement MOSFET channel construction
P
N
P
G
Sub
SiO2
S
N
P
N
N-channel MOSFET
P-channel MOSFET
Drain
Gate
Substrate
Source
Enhancement Type (normally-off)
Depletion Type (normally-on)

Figure 2. Construction and symbols of MOS transistors

The four MOSFET symbols above show an additional terminal called the Substrate and it is not normally used as either an input or an output connection but instead it is used for grounding the substrate. It connects to the main semiconductive channel through a diode junction to the body or metal tab of the MOSFET. Usually in discrete type MOSFETs, this substrate lead is connected internally to the source terminal. As in enhancement types, it can be omitted from the symbol for clarification.

The line between the drain and source connections represents the semiconductive channel. If this is a solid unbroken line then this represents a “Depletion” (normally-ON) type MOSFET as drain current can flow with zero gate potential. If the channel line is shown dotted or broken it is an “Enhancement” (normally-OFF) type MOSFET as zero drain current flows with zero gate potential. The direction of the arrow indicates whether the conductive channel is a p-type or an n-type semiconductor device.

Basic MOSFET Structure and Symbol :

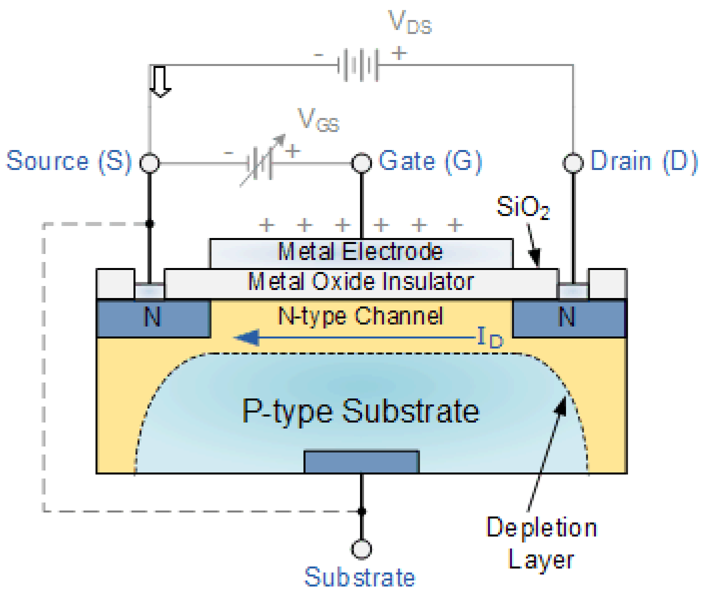


Figure 3. MOSFET Structure

The construction of the Metal Oxide Semiconductor FET is very different to that of the Junction FET. Both the Depletion and Enhancement type MOSFETs use an electrical field produced by a gate voltage to alter the flow of charge carriers, electrons for n-channel or holes for P-channel, through the semiconductive drain-source channel. The gate electrode is placed on top of a very thin insulating layer and there are a pair of small n-type regions just under the drain and source electrodes. Figure 3 represents the MOSFET structure.The gate of a junction field effect transistor, JFET must be biased in such a way as to reverse-bias the pn-junction. With a insulated gate MOSFET device no such limitations apply so it is possible to bias the gate of a MOSFET in either polarity, positive (+ve) or negative (-ve).This makes the MOSFET device especially valuable as electronic switches or to make logic gates because with no bias they are normally non-conducting and this high gate input resistance means that very little or no control current is needed as MOSFETs are voltage controlled devices. Both the p-channel and the n-channel MOSFETs are available in two basic forms, the **Enhancement** type and the **Depletion** type.

Enhancement-mode MOSFET:

The more common **Enhancement-mode MOSFET** or eMOSFET, is the reverse of the depletion-mode type. Here the conducting channel is lightly doped or even undoped making it non-conductive. This results in the device being normally “OFF” (non-conducting) when the gate bias voltage, VGS is equal to zero. The circuit symbol shown above for an enhancement MOS transistor uses a broken channel line to signify a normally open non-conducting channel.

For the n-channel enhancement MOS transistor a drain current will only flow when a gate voltage (VGS) is applied to the gate terminal greater than the threshold voltage (VTH) level in which conductance takes place making it a transconductance device. The application of a positive (+ve) gate voltage to n-type eMOSFET attracts more electrons towards the oxide layer around the gate thereby increasing or enhancing (hence its name) the thickness of the channel allowing more current to flow. This is why this kind of transistor is called an enhancement mode device as the application of a gate voltage enhances the channel.

Increasing this positive gate voltage will cause the channel resistance to decrease further causing an increase in the drain current, ID through the channel. In other words, for an n-channel enhancement mode MOSFET: +VGS turns the transistor "ON", while a zero or -VGS turns the transistor "OFF". Then, the enhancement-mode MOSFET is equivalent to a "normally-open" switch.The reverse is true for the p-channel enhancement MOS transistor. When VGS = 0 the device is "OFF" and the channel is open. The application of a negative (-ve) gate voltage to the p-type eMOSFET enhances the channels conductivity turning it "ON". Then for an p-channel enhancement mode MOSFET: +VGS turns the transistor "OFF", while -VGS turns the transistor "ON".

Enhancement-mode N-Channel MOSFET and circuit Symbols

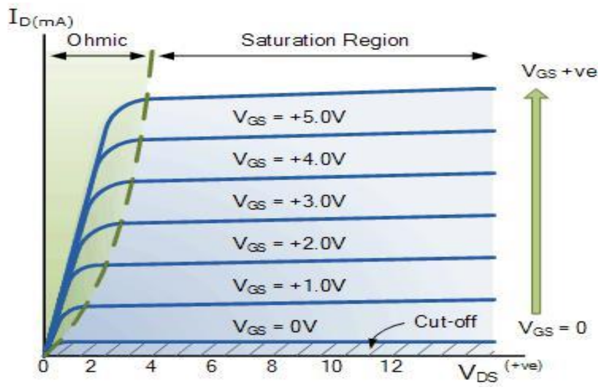


Figure 4. Drain characteristics of eMOSFET

Enhancement-mode MOSFETs make excellent electronics switches due to their low "ON" resistance and extremely high "OFF" resistance as well as their infinitely high input resistance due to their isolated gate. Enhancement-mode MOSFETs are used in integrated circuits to produce CMOS type Logic Gates and power switching circuits in the form of as PMOS (P-channel) and NMOS (N-channel) gates. CMOS actually stands for *Complementary MOS* meaning that the logic device has both PMOS and NMOS within its design. Figure 4 represents the drain characteristics of Enhancement mode transistor.

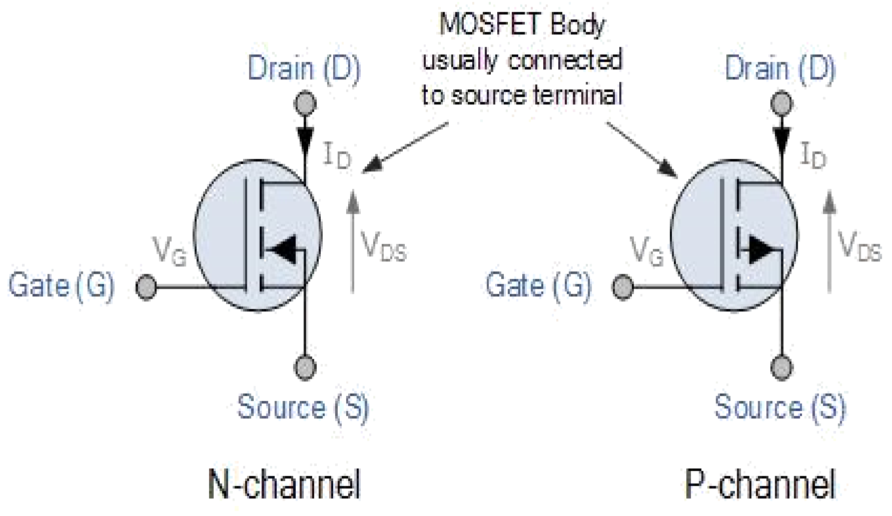


Figure 5. Symbols of eMOSFET transistors

<u>Depletion-mode MOSFET :</u>

The **Depletion-mode MOSFET**, which is less common than the enhancement mode types is normally switched "ON" (conducting) without the application of a gate bias voltage. That is the channel conducts when VGS = 0 making it a "normally-closed" device. The circuit symbol shown above for a depletion MOS transistor uses a solid channel line to signify a normally closed conductive channel.For the n-channel depletion MOS transistor, a negative gate-source voltage, -VGS will deplete (hence its name) the conductive channel of its free electrons switching the transistor "OFF". Likewise for p-channel depletion MOS transistor a positive gate-source voltage, +VGS will deplete the channel of its free holes turning it "OFF".

In other words, for an n-channel depletion mode MOSFET: +VGS means more electrons and more current. While a -VGS means less electrons and less current. The opposite is also true for the p-channel types. Then the depletion mode MOSFET is equivalent to a "normally-closed" switch.

Depletion-mode N-Channel MOSFET and circuit Symbols

The depletion-mode MOSFET is constructed in a similar way to their JFET transistor counterparts were the drain-source channel is inherently conductive with the electrons and holes already present within the n-type or p-type channel. This doping of the channel produces a conducting path of low resistance between the Drain and Source with zero Gate bias. The drain characteristics and the circuit symbols of depletion mode transistor are shown in Figure.6.

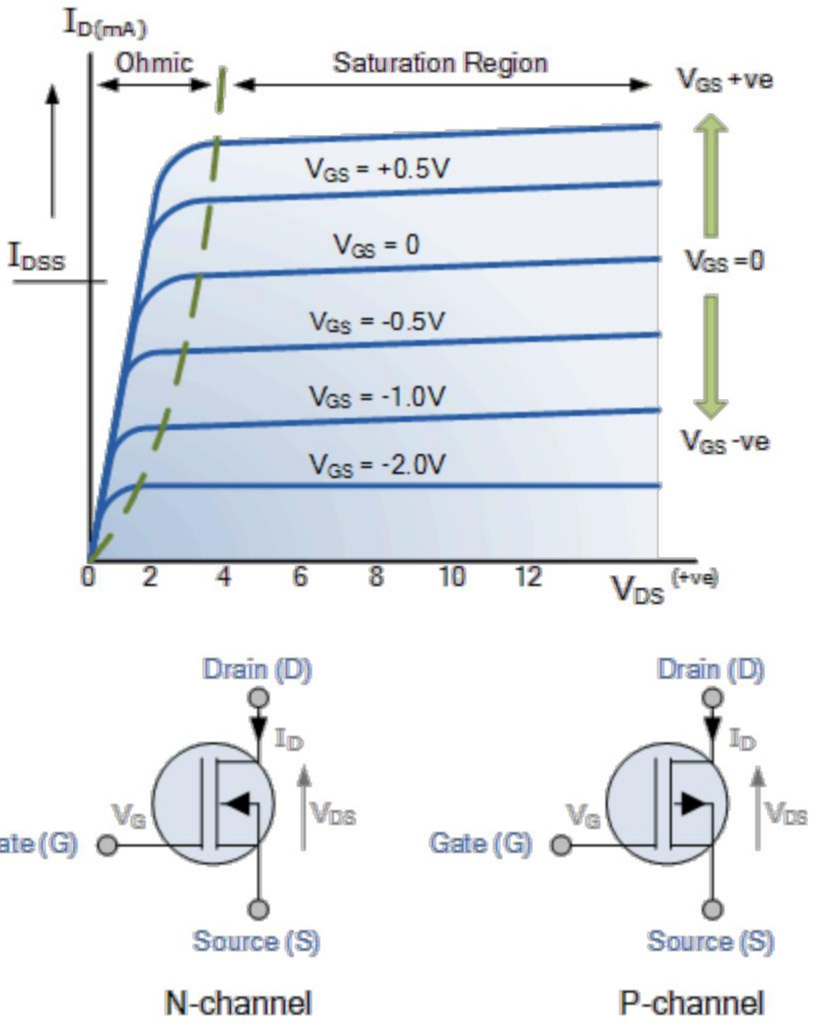
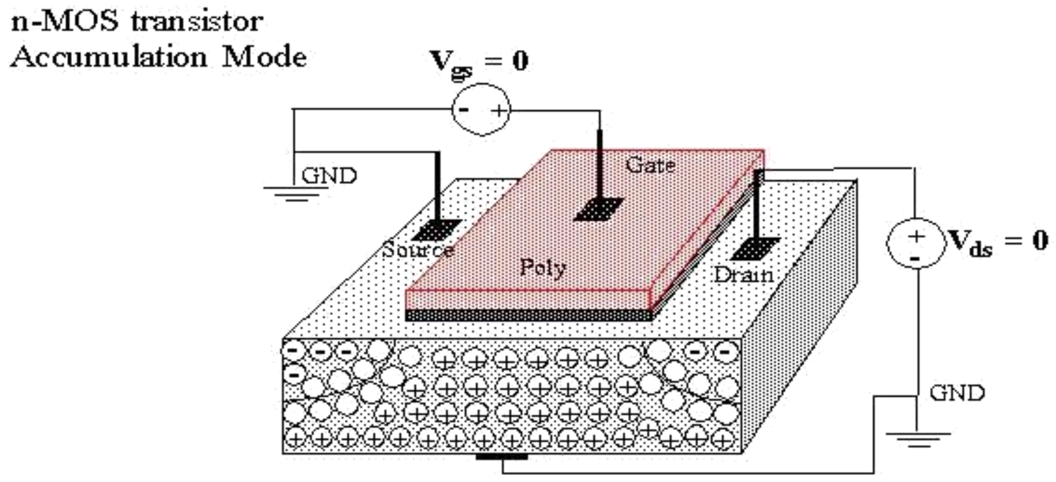


Figure 6. Drain characteristics and symbols of Depletion MOS

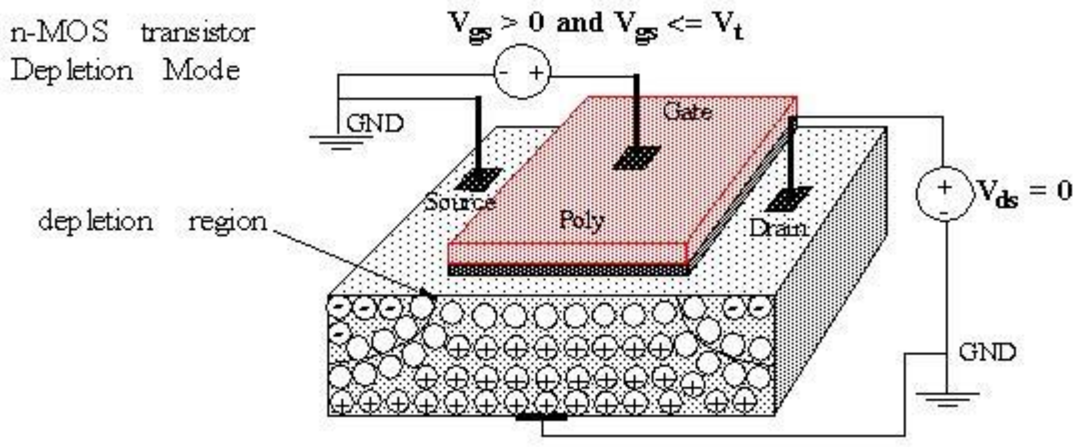
Operating modes of nMOS transistor:

Three sets, of D.C conditions are needed to apply for understanding the operating modes of nMOS transistor. There are three modes based on the magnitude of Vgs is shown in Figure.7 and named as,

1. accumulation, 2. depletion 3. inversion.



Depletion Mode



Inversion Mode

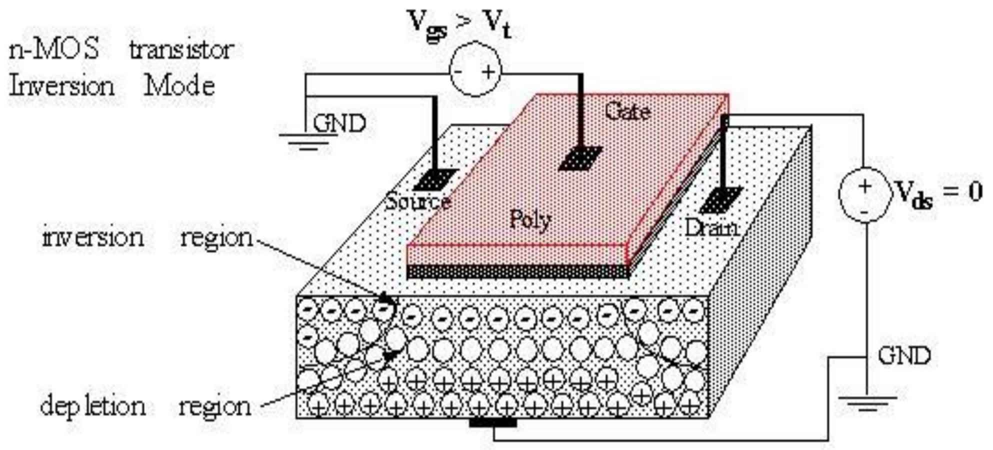


Figure 7. Operating modes of MOS transistor

Enhancement mode Transistor action:-

The device requires a voltage to be applied before the channel is formed is called as "Enhancement mode transistor". Three basic sets of dc conditions required for understanding the operating regions of MOS transistor. To establish the channel between the source and the drain a minimum voltage (Vt) must be applied between gate and source. This minimum voltage is called as Vth. There is no conducting channel present initially hence nMOS enhancement is normally called as "OFF device. It is known as "threshold voltage" that is required to form the channel to bring out transistor into conduction.

a) Vgs >Vt
Vds = 0

Since Vgs > Vt and Vds = 0 the channel is formed but no current flows between drain and source. This region is called **Cut-off region**.

b) Vgs > Vt

Vds < Vgs - Vt

This region is called the N**on-Saturation Region or linear region** where the drain current increases linearly with Vds. When Vds is increased the drain side becomes more reverse biased (hence more depletion region towards the drain end) and the channel starts to pinch. This is called as the pinch off point.

c) Vgs > Vt

Vds > Vgs - Vt

This region is called **Saturation Region** where the drain current remains almost constant. As the drain voltage is increased further beyond (Vgs-Vt) the pinch off point starts to move from the drain end to the source end. Even if the Vds is increased more and more, the increased voltage gets dropped in the depletion region leading to a constant current. The typical threshold voltage for an enhancement mode transistor is given by Vt = 0.2 * Vdd.

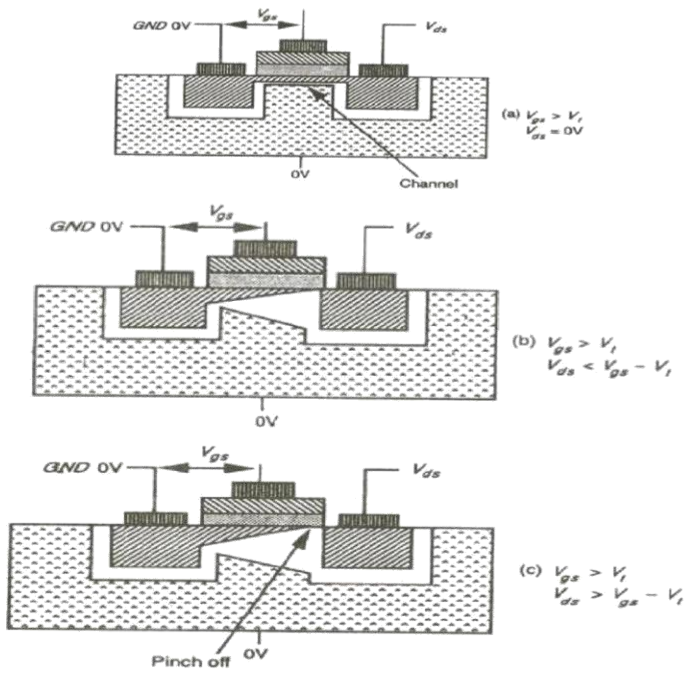

GND 0V
V_{gs}
V_{ds}
(a) $V_{gs} > V_t$, $V_{ds} = 0V$
0V
Channel
(b) $V_{gs} > V_t$, $V_{ds} < V_{gs} - V_t$
(c) $V_{gs} > V_t$, $V_{ds} > V_{gs} - V_t$
Pinch off

Figure.8 (a)(b)(c) Enhancement mode transistor with different Vds values

Current and Voltage Relationships of MOS transistor:

A **mathematical** description of enhancement MOSFET behavior is relatively straightforward with just **3 equations**. Specifically, the drain current *iD* can be expressed in terms of *vGS* and *vDS* for each of the **three MOSFET modes** (i.e., Cutoff, Triode, Saturation).The drain to source current is defined as the ratio between the charges induced in the channel and electron transit time. This section first derives the current-voltage relationships for various bias conditions in a MOS transistor. Although the subsequent discussion is centred on an nMOS transistor, the basic expressions can be

derived for a pMOS transistor by simply replacing the electron mobility μ_n by the hole mobility μ_p and reversing the polarities of voltages and currents.

As mentioned in the earlier section, the fundamental operation of a MOS transistor arises out of the gate voltage *VGS* (between the gate and the source) creating a channel between the source and the drain, attracting the majority carriers from the source and causing them to move towards the drain under the influence of an electric field due to the voltage *VDS* (between the drain and the source). The corresponding current *IDS* depends on both *VGS* and *VDS* .

Steps involved:

Let us consider the simplified structure of an nMOS transistor shown in Figure.9, in which the majority carriers such as electrons flow from the source to the drain.

The conventional current flowing from the drain to the source is given by

$$I_{DS} = -I_{SD} = (\text{charge induced in channel}) / (\text{electron transit time}) = Q_C / \tau_n$$

Now, transit time = τ_n (length of the channel) / (electron velocity) = *L / v* where

velocity is given by the electron mobility and electric field; or, $v = \mu_n E_{DS}$ Now, *EDS*

= *VDS/ L*, so that velocity $v = (\mu_n V_{DS}) / L$ Thus, the transit time is $\tau_n = L^2 / (\mu_n V_{DS})$

At room temperature (300 K), typical values of the electron and hole mobility are given by

$\mu_n = 650 \text{ cm}^2 / V - \text{sec}$, and $\mu_p = 240 \text{ cm}^2 / V - \text{sec}$

The current-voltage relationship can be derived separately for the linear (or non-saturated)

region and the saturated region of operation.

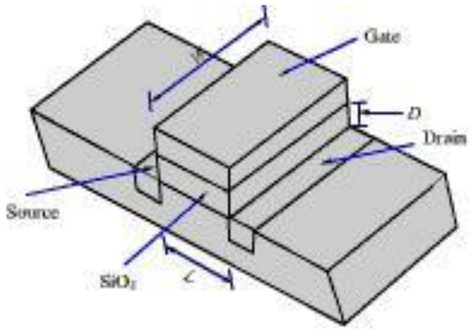


Figure 9. Geometrical structure of nMOS transistor

Linear region:

Note that this region of operation implies the existence of the uninterrupted channel between the source and the drain, which is ensured by the voltage relation *VGS* - *Vth* > *VDS* . In the channel, the voltage between the gate and the source varies ***linearly*** with the distance *x* from the source due to the IR drop in the channel.

Assume that the device is not saturated and the average channel voltage is *VDS* /2. The

effective gate voltage or *VG,eff = Vgs - Vth*

Charge per unit area = $E_g \varepsilon_{ins} \varepsilon_0$

where *Eg* average electric field from gate to channel, ε_{ins} : relative permittivity of oxide between gate and channel (~4.0 for SiO2), and ε_0 : free space permittivity (8.85 x 10 $^{-14}$ F/cm).
So, induced charge is expressed as equation,

$$Q_C = E_g \varepsilon_{ins} \varepsilon_0 WL$$

where *W is the* width of the gate and *L* is the length of channel.

$Q_C = E_g \varepsilon_{ins} \varepsilon_0 WL$ Substitute the formula for Eg = {(Vgs -Vt) – Vds /2 } / D D is the oxide thickness

$$O_C = WL\varepsilon_{ins}\varepsilon_0 / D\left\{\left(V_{GS} - V_{th}\right) - V_{DS} / 2\right\}$$

$$\tau_n = L^2 / \left(\mu_n V_{DS}\right)$$

Thus, the current from the drain to the source may be expressed as

$$I_{DS} = Q_C / \tau_n = \varepsilon_{ins}\varepsilon_0 \mu_n W / (LD)\left\{\left(V_{GS} - V_{th}\right) - V_{DS} / 2\right\} V_{DS}$$

Thus, in the non-saturated region, where $V_{DS} < V_{GS} - V_{th}$

$I_{DS} = (KW) / L\left\{\left(V_{GS} - V_{th}\right)V_{DS} - V^2{}_{DS} / 2\right\}$ ....(1)

where the parameter

$$K = \left(\varepsilon_{ins}\varepsilon_0 \mu_n\right) / D$$

Writing $\beta = (KW) / L$, where *W/L* is contributed by the geometry of the device,

$I_{DS} = \beta\left\{\left(V_{GS} - V_{th}\right)V_{DS} - V^2{}_{DS} / 2\right\}$ ....(2)

Since, the gate-to-channel capacitance is $C_G = (\varepsilon_{ins}\varepsilon_0 WL)/D$ (parallel plate capacitance), then $K = (C_G \mu_n)/(WL)$

, so that equation may be written as

$I_{DS} = (C_G \mu_n)/L^2 \{(V_{GS} - V_{th})V_{DS} - V^2{}_{DS}/2\}$ ....(3)

Denoting *CG = C0 / WL* where *C0* is the gate capacitance per unit area,

$I_{DS} = (C_0 \mu_n W)/L^2 \{(V_{GS} - V_{th})V_{DS} - V^2{}_{DS}/2\}$ .... (4)

Saturated region: Under the voltage condition *VGS - Vth = VDS* , a MOS device is said to be in saturation region of operation. In fact, saturation begins when *VDS = VGS - Vth* , since at this point, the resistive voltage drop (IR drop) in the channel equals the effective gate-to-channel voltage at the drain. One may assume that the current remains *constant* as *VDS* increases further. Putting *VDS = VGS - Vth* , the equations (1-4) under saturation condition need to be modified as

$I_{DS} = (KW)/L\{(V_{GS} - V_{th})^2/2\}$ .... (5)

$I_{DS} = \beta(V_{GS} - V_{th})^2/2$ .... (6)

$I_{DS} = \{C_G \mu_n (V_{GS} - V_{th})^2\}/(2L^2)$ .... (7)

$I_{DS} = \{C_0 \mu_n (V_{GS} - V_{th})^2\}/(2L)$ .... (8)

Note: The expressions derived for *IDS* are valid for both the enhancement and the depletion mode devices. However, the threshold voltage for the nMOS depletion mode devices (generally denoted as *Vtd*) is *negative* . From Figure of the typical current-voltage characteristics for nMOS enhancement as well as depletion mode transistors, the corresponding curves for a pMOS device may be obtained with appropriate reversal of polarity.

For an *n* -channel device with μ_n = 600 cm^2/ V.s, *C0* = 7 X 10^{-8} F/cm^2 , *W* = 20 m, *L* = 2 m and *Vth* = *VT0* = 1.0 V, let us examine the relationship between the drain current and the terminal voltages.

$$K = (C_G\mu_n)/(WL) = (C_0\mu_n W)/L = 600\,\text{cm}^2/\text{V.s} \times 7\times10^{-8}\,\text{F/cm}^2 \times 20\mu\text{m}/2\,\mu\text{m} = 0.42\,\text{mA/V}^2$$

Now, the current-voltage equation can be written as follows.

$$I_{DS} = 0.21\text{mA/V}^2\{2(V_{GS}-1.0)V_{DS} - V^2{}_{DS}\}$$

Threshold Voltage:

The threshold voltage Vth for a nMOS transistor is the minimum amount of the gate-to-source voltage VGS necessary to cause surface inversion so as to create the conducting channel between the source and the drain. For VGS< Vth , no current can flow between the source and the drain. For VGS> Vth , a larger number of minority carriers (electrons in case of an nMOS transistor) are drawn to the surface, increasing the channel current. However, the surface potential and the depletion region width remain almost unchanged as VGS is increased beyond the threshold voltage.

The physical components determining the threshold voltage are the following.

- work function difference between the gate and the substrate.
- gate voltage portion spent to change the surface potential.
- gate voltage part accounting for the depletion region charge.
- gate voltage component to offset the fixed charges in the gate oxide and the silicon-oxide boundary.

Although the following analysis pertains to an nMOS device, it can be simply modified to reason for a p-channel device. The work function difference ϕ_{GS} between the doped polysilicon gate and the p-type substrate, which depends on the substrate doping, makes up the first component of the threshold voltage. The externally applied gate voltage must also account for the strong inversion at the surface, expressed in the form of surface potential 2ϕ_F, where ϕ_F denotes the distance between the intrinsic energy level *EI* and the Fermi level *EF* of the p-type semiconductor substrate.

The factor 2 comes due to the fact that in the bulk, the semiconductor is p-type, where *EI* is above *EF* by ϕ_F, while at the inverted n-type region at the surface *EI* is below EF by ϕ_F, and thus the amount of the band bending is 2ϕ_F . This is the second component of the threshold voltage. The potential difference ϕ_F between *EI* and *EF* is given as

second component of the threshold voltage. The potential difference ϕ_F between *EI* and *EF* is given as

$$\phi_F = \frac{kT}{q} \ln\left(\frac{N_A}{n_i}\right)$$

where *k* is the Boltzmann constant, *T is the* temperature, *q* : electron charge *NA* : acceptor concentration in the p-substrate and ni is the intrinsic carrier concentration. The expression kT/q is 0.02586 volt at 300 K. The applied gate voltage must also be large enough to create the depletion charge. Note that the charge per unit area in the depletion region at strong inversion is given by

$$Q_{d0} = -2(\varepsilon_S q N_A \phi_F)^{1/2}$$

where ε_S is the substrate permittivity. If the source is biased at a potential *VSB* with respect to the substrate, then the depletion charge density is given by

$$Q_d = -2(\varepsilon_S q N_A(\phi_F + V_{SB}))^{1/2}$$

The component of the threshold voltage that offsets the depletion charge is then given by -Qd /*Cox* , where *Cox* is the gate oxide capacitance per unit area, or *Cox* = $\varepsilon_{ox} / t_{ox}$ (ratio of the oxide permittivity and the oxide thickness). A set of positive charges arises from the interface states at the Si-SiO2 interface. These charges, denoted as *Qi* , occur from the abrupt termination of the semiconductor crystal lattice at the oxide interface. The component of the gate voltage needed to offset this positive charge (which induces an equivalent negative charge in the semiconductor) is -*Qi* /*Cox*. On combining all the four voltage components, the threshold voltage *VTO*, for zero substrate bias, is expressed as

$$V_{T0} = \phi_{GS} - 2\phi_F - \frac{Q_{d0}}{C_{ox}} - \frac{Q_i}{C_{ox}}$$

For non-zero substrate bias, however, the depletion charge density needs to be modified to include the effect of VSB on that charge, resulting in the following generalized expression for the threshold voltage, namely

$$V_T = \phi_{GS} - 2\phi_F - \frac{Q}{C_{ox}} - \frac{Q}{C_{ox}}$$

The generalized form of the threshold voltage can also be written as

$$V_T = \phi_{GS} - 2\phi_F - \frac{Q_{d0}}{C_{ox}} - \frac{Q_i}{C_{ox}} - \frac{Q_d - Q_{d0}}{C_{ox}} = V_{T0} - \frac{Q_d - Q_{d0}}{C_{ox}}$$

Note that the threshold voltage differs from *VTO* by an additive term due to substrate bias.
This term, which depends on the material parameters and the source-to-substrate voltage *VSB*, is given by

$$\frac{Q_d - Q_{d0}}{C_{ox}} = -\frac{\sqrt{2qN_A\varepsilon_S}}{C_{ox}}\left(\sqrt{|2\phi_F + V_{SB}|} - \sqrt{|2\phi_F|}\right)$$

Thus, in its most general form, the threshold voltage is determined as

$$V_T = V_{T0} + \gamma\left(\sqrt{|2\phi_F + V_{SB}|} - \sqrt{|2\phi_F|}\right)$$ .... (1)

in which the parameter γ, known as the ***substrate-bias* (or *body-effect*)** coefficient is given by

$$\gamma = \frac{\sqrt{2qN_A\varepsilon_S}}{C_{ox}}$$ .... (2)

The threshold voltage expression given by (1) can be applied to n-channel as well as p-channel transistors. However, some of the parameters have opposite polarities for the pMOS and the nMOS transistors. For example, the substrate bias voltage *VSB* is positive in nMOS and negative in pMOS devices. Also, the substrate potential difference
ϕ_F is negative in nMOS, and positive in pMOS. Whereas, the body-effect coefficient γ is positive in nMOS and negative in pMOS. Typically, the threshold voltage of an enhancement mode n-channel transistor is positive, while that of a p-channel transistor is negative.

Second order effects:

The current-voltage equations in the previous section however are ideal in nature. These have been derived keeping various secondary effects out of consideration. The effects are, 1. Threshold voltage variations 2.Subthreshold region 3. Channel length modulation 4.Mobility variation 5. Fowler-Nordheim Tunneling 6. Drain Punch through 7.Impact Ionization

Threshold voltage and body effect: The threshold voltage *Vth* does vary with the voltage difference *Vsb* between the source and the body (substrate). Thus including this difference, the generalized expression for the threshold voltage is reiterated as

$$V_T = V_{T0} + \gamma\left(\sqrt{|2\phi_F + V_{SB}|} - \sqrt{|2\phi_F|}\right)$$ .... (3)

in which the parameter γ, known as the *substrate-bias* (or *body-effect*) coefficient is given by

$$\gamma = \frac{\sqrt{2qN_A\varepsilon_S}}{C_{ox}}$$

Typical values of γ range from 0.4 to 1.2. It may also be written as

$$\gamma = \frac{t_{ox}}{\varepsilon_{ox}}\sqrt{2qN_A\varepsilon_S} = \frac{1}{C_{ox}}\sqrt{2qN_A\varepsilon_S}$$

Drain punch-through:

In a MOSFET device with improperly scaled small channel length and too low channel doping, undesired electrostatic interaction can take place between the source and the drain known as *drain-induced barrier lowering* (DIBL) takes place. This leads to punch-through leakage or breakdown between the source and the drain, and loss of gate control. One should consider the surface potential along the channel to understand the punch-through phenomenon. As the drain bias increases, the conduction band edge (which represents the electron energies) in the drain is pulled down, leading to an increase in the drain-channel depletion width.

In a long-channel device, the drain bias does not influence the source-to-channel potential barrier, and it depends on the increase of gate bias to cause the drain current to flow. However, in a short-channel device, as a result of increase in drain bias and pull-down of the conduction band edge, the source-channel potential barrier is lowered due to DIBL. This in turn causes drain current to flow regardless of the gate voltage (that is, even if it is below the threshold voltage Vth). More simply, the advent of DIBL may be explained by the expansion of drain depletion region and its eventual merging with source depletion region, causing punch-through breakdown between the source and the drain. The punch-through condition puts a natural constraint on the voltages across the internal circuit nodes.

Sub-threshold region conduction:

The cutoff region of operation is also referred to as the sub-threshold region, which is mathematically expressed as IDS =0 ; VGS < Vth However, a phenomenon called *sub-threshold conduction* is observed in small-geometry transistors. The current flow in the channel depends on creating and maintaining an inversion layer on the surface. If the gate voltage is inadequate to invert the surface (that is, *VGS*< *VT0*), the electrons in the channel encounter a *potential barrier* that blocks the flow. However, in small-geometry MOSFETs, this potential barrier is controlled by both *VGS* and *VDS* .

If the drain voltage is increased, the potential barrier in the channel decreases, leading to *drain-induced barrier lowering* (DIBL). The lowered potential barrier finally leads to flow of electrons between the source and the drain, even if *VGS* < *VT0* (that is, even when the surface is not in strong inversion). The channel current flowing in this condition is called the *sub-threshold current*. This current, due mainly to diffusion between the source and the drain, is causing concern in deep sub-micron designs. The model implemented in SPICE brings in an exponential, semi-empirical dependence of the drain current on *VGS* in the *weak inversion region.* Defining a voltage *V* on as the boundary between the regions of weak and strong inversion,

$$I_D(weak\ inversion) = I_{on} . e^{(V_{GS}-V_{on}).\left(\frac{q}{nkT}\right)}$$

where *Ion* is the current in strong inversion for *VGS* =*Von* .

Channel length modulation :

so far one has not considered the variations in channel length due to the changes in drain-to-source voltage *VDS* . For long-channel transistors, the effect of channel length variation is not prominent. With the decrease in channel length, however, the variation matters. Figure shows that the inversion layer reduces to a point at the drain end when

VDS = *VDSAT* = *VGS* -*Vth* .

That is, the channel is *pinched off* at the drain end. The onset of saturation mode operation is indicated by the pinch-off event. If the drain-to-source voltage is increased beyond the saturation edge (*VDS* > *VDSAT*), a still larger portion of the channel becomes pinched off.

Let the effective channel (that is, the length of the inversion layer) be $L_{eff} = L - \Delta L$.
where *L* : original channel length (the device being in non-saturated mode), and ΔL : length of the channel segment where the inversion layer charge is zero. Thus, the pinch-off point moves from the drain end toward *VDS* the source with increasing drain-to-source voltage . The remaining portion of the channel between the pinch-off point and the drain end will be in depletion mode. For the shortened channel, with an effective channel voltage of *VDSAT* , the channel current is given by

$$I_{DS(SAT)} = \frac{\mu_n C_{ox}}{2} \cdot \frac{W}{L_{eff}} \cdot (V_{GS} - V_{T0})^2 \qquad \text{..................... (1)}$$

The current expression pertains to a MOSFET with effective channel length *Leff*, operating in saturation. The above equation depicts the condition known as *channel length modulation*, where the channel is reduced in length. As the effective length decreases with increasing *VDS*, the saturation current *IDS(SAT)* will consequently increase with increasing *VDS* . The current given by (1) can be re-written as

$$I_{DS(SAT)}=\frac{\mu_n C_{ox}}{2}\cdot\left(\frac{1}{1-\frac{\Delta L}{L}}\right)\frac{W}{L}\cdot(V_{GS}-V_{T0})^2$$
.................(2)

The second term on the right hand side accounts for the channel modulation effect. It can be shown that the factor channel length ΔL is expressible as

$$\Delta L \propto \sqrt{V_{DS}-V_{DSAT}}$$

One can even use the empirical relation between ΔL and *VDS* given as follows.

$$1-\frac{\Delta L}{L}\approx 1-\lambda V_{DS}$$

The parameter λ is called the *channel length modulation coefficient*, having a value in the range 0.02V -1 to 0.005V -1. Assuming that $\lambda V_{DS} << 1$, the modified saturation current expression can be written as

$$I_{DS(SAT)}=\frac{\mu_n C_{ox}}{2}\cdot\frac{W}{L_{eff}}\cdot(V_{GS}-V_{T0})^2\cdot(1+\lambda V_{DS})$$

This simplified equation points to a linear dependence of the saturation current on the drain-to-source voltage. The slope of the current-voltage characteristic in the saturation region is determined by the channel length modulation factor λ .

Impact ionization:

An electron traveling from the source to the drain along the channel gains kinetic energy at the cost of electrostatic potential energy in the pinch-off region, and becomes a “hot” electron. As the hot electrons travel towards the drain, they can create secondary electron-hole pairs by impact ionization. The secondary electrons are collected at the drain, and cause the drain current in saturation to increase with drain bias at high voltages, thus leading to a fall in the output impedance. The secondary holes are collected as substrate current. This effect is called *impact ionization.*

The hot electrons can even penetrate the gate oxide, causing a gate current. This finally leads to degradation in MOSFET parameters like increase of threshold voltage and decrease of transconductance. Impact ionization can create circuit problems such as noise in mixed-signal systems, poor refresh times in dynamic memories, or latch-up in CMOS circuits. The remedy to this problem is to use a device with lightly doped drain. By reducing the doping density in the source/drain, the depletion width at the reverse-biased drain-channel junction is increase and consequently, the electric filed is reduced. Hot carrier effects do not normally present an acute problem for *p* -channel MOSFETs. This is because the channel mobility of holes is almost half that of the electrons. Thus, for the same filed, there are fewer hot holes than hot electrons. However, lower hole mobility results in lower drive currents in *p* -channel devices than in *n* -channel devices.

Capacitances in MOSFET:

The parasitic transistor can be clearly seen in Figure as the N+ / P / N+ region between drain and source. If the distance the current travels from the enhanced region across the source N+ region is small, Rbe is negligible and the base collector junction of the parasitic transistor appears as a diode.

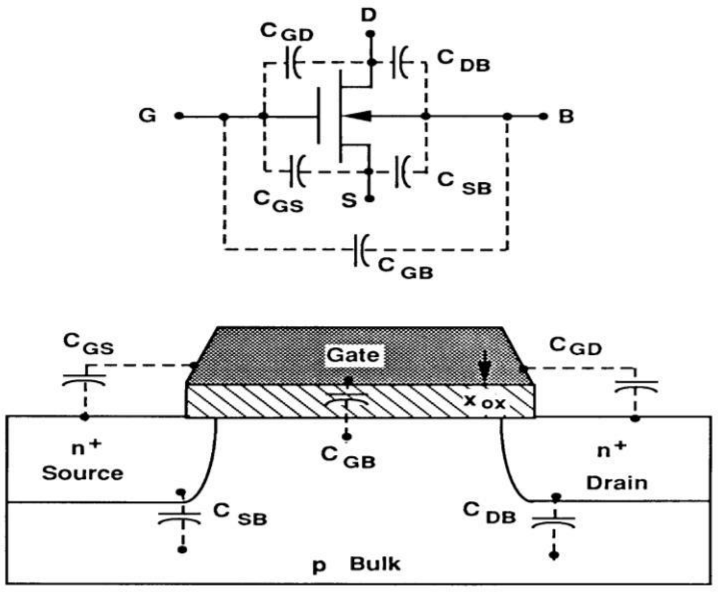


Figure 2.9: MOSFET Capacitance Model

Figure 10. MOSFET Capacitances

The equivalent circuit of an enhancement MOSFET is shown in Figure 10.

Two parasitic capacitances between gate to source and gate to drain will cause switching delays if the gate driver cannot support large initial currents. A further parasitic capacitance and transistor exist between drain and source but due to the internal structure the transistor appears as a diode and capacitor connected between drain and source as shown in Figure 6b. Unfortunately the parasitic diode does NOT have the structure of a fast diode and must be neglected and a separate fast diode used in a high speed switching circuit.

Gate Capacitances

- The build-up and removal of the channel and its associated charge is similar to charging and discharging a capacitor. In the case of the channel, this capacitor has an upper plate or electrode, that is the MOS gate, and a lower electrode mad of three plates, the source, the bulk (substrate) and the drain. Hence, charges can enter or leave the upper plate only through the gate terminal. For the lower plate, the charges can enter/leave through any of the three terminals (S, B and D).

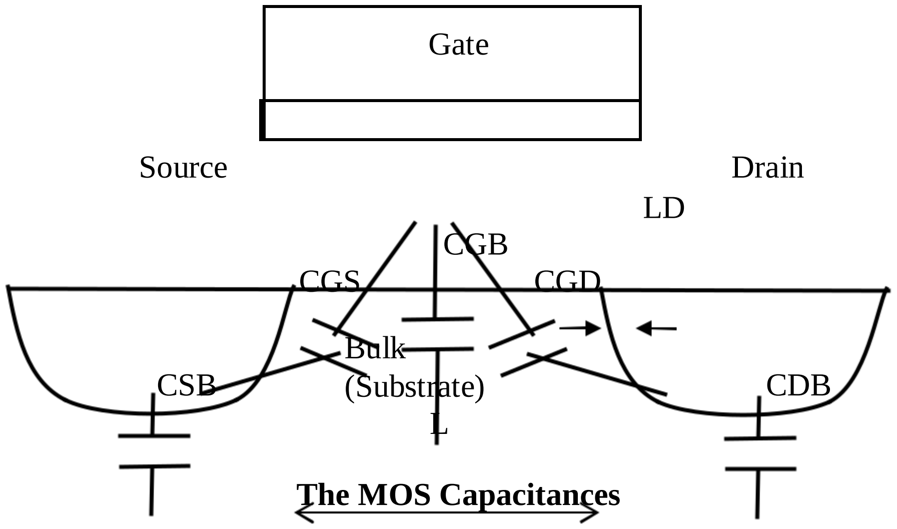
- Hence the channel charge is lumped (modeled) into three capacitances, as shown in the figure below, Gate-to-Bulk capacitance (CGB), Gate-to-Source capacitance (CGS), and Gate-to-Drain capacitance (CGD). These capacitances are not constant; their values depend on the region of operation. CGS and CGD have two components, called overlap capacitance, that are constant. They basically represent the capacitance between the gate and S/D regions in the overlap area, as shown in the figure below. In the cut-off region, where the channel region is in accumulation (of majority carriers), the gate capacitance is the same as Cox (times L*W), and it is all to the bulk (i.e. CGS and CGD = 0).

- When the device is on (i.e. channel is created and surface is in strong-inversion), the channel charge shields the bulk from the gate, i.e. CGB becomes zero and the gate capacitance is distributed between CGS and CGD. In linear region, the gate capacitance is distributed equally between CGS and CGD while in saturation, almost all of the channel charge is controlled by the source, i.e. CGD =0, while CGS =2/3 Cox* **L***W.

- The table below summarizes the values of the gate capacitances for the three different regions of operation as a function of the oxide capacitance Cox, the device length L and width W, and the overlap length LD between the gate and S/D regions. Also shown below the table, a graph of the gate capacitances versus VGS for the different regions of operation. For this graph, VDS is kept constant. The value of VGS that gives a minimum total gate capacitance is actually the threshold voltage.

S/D Junction Capacitances

- The source and drain forms diodes with the bulk. As seen before these diodes will have junction capacitances that are dependent on the voltage difference between their terminals. Hence, as the source or drain voltages change, these capacitances will be charged or discharged. These capacitances, named CSB and CDB are also shown in the Figure 11.below.



The MOS Capacitances

Region / Capacitance	Cut-Off	Saturation	Linear
CGB	Cox*W*L	0	0
CGS	LDWCox	2/3(CoxWL)+LDWCox	½(CoxWL)+LDWCox
CGD	LDWCox	LDWCox	½(CoxWL)+LDWCox

Values of Gate Capacitances at different regions

Scaling of MOS Circuits :

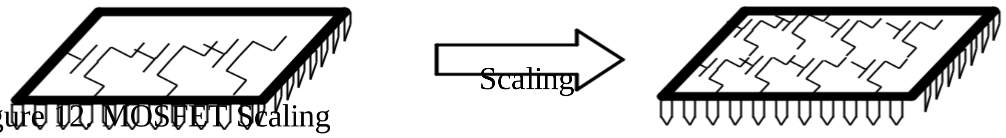


Figure 12. MOSFET Scaling

- To increase the number of devices per IC, the device dimensions had to be shrunk from one generation to another (i.e. scaled down)There are two methods of scaling:

1. Full-Scaling (also called Constant-Field scaling): In this method the device dimensions (both horizontal and vertical) are scaled down by 1/S, where S is the scaling factor. In order to keep the electric field constant within the device, the voltages have to be scaled also by 1/S such that the ratio between voltage and distance (which represents the electric field) remain constant. The threshold voltage is also scaled down by the same factor as the voltage to preserve the functionality of the circuits and the noise margins relative to one another. As a result of this type of scaling the currents will be reduced and hence the total power per transistor (P=IxV) will also be reduced, however the power density will remain constant since the number of transistors per unit area will increase. This means that the total chip power will remain constant if the chip size remains the same (this usually the case).

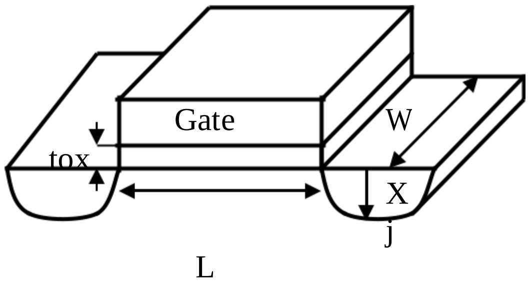


Figure 13. MOSFET Scaling

The table 1 below summarizes how each device parameter scales with S (S>1)

Parameter	**Before scaling**	**After scaling**
Channel length	L	L/S
Channel width	W	W/S
Oxide thickness	tox	tox/S
S/D junction depth	Xj	Xj/S
Power Supply	VDD	VDD/S
Threshold voltage	VTO	V_{TO} /S
Doping Density	N_A&N_D	N_A *S and N_D *S
Oxide Capacitance	Cox	S*Cox

Drain Current	IDS	I_{DS} /S
Power/Transistor	P	P/S^2
Power Density/cm^2	p	p

2. Constant-Voltage scaling (CVS): In this method the device dimensions (both horizontal and vertical are scaled by S, however, the operating voltages remain constant. This means that the electric fields within the device will increase (filed =Voltage/distance). The threshold voltages remain constant while the power per transistor will increase by S. The power density per unit area will increase by S^3! This means that for the same chip area, the power chip power will increase by S^3. This makes constant-voltage-scaling (CVS) very impractical. Table 2 represents the device parameter scales under constant voltage.

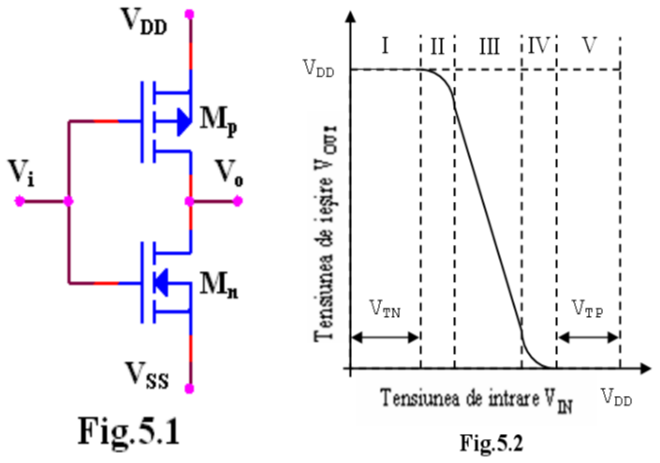
3. Also, the device doping has to be increased more aggressively (by S^2) than the constant-field scaling to prevent channel punch-through. Channel punch-through occurs when the Source and Drain Depletion regions touches one another. By increasing the doping by S^2, the depletion region thickness is reduced by S (the same ratio as the channel length). However, there is a limit for how much the doping can be increased (the solid solubility limit of the dopant in Silicon). Again, this makes the CVS impractical in most cases. The following table summarizes the changes in key device parameters under constant-voltage scaling: In almost all cases, the scaling is a combination of constant-field scaling and constant-voltage scaling, such that the number of devices is increased and the total power/chip does not increase much.

Parameter	**Before scaling**	**After scaling**
Channel length	L	L/S
Channel width	W	W/S
Oxide thickness	tox	tox/S
S/D junction depth	Xj	Xj/S
Power Supply	VDD	VDD
Threshold voltage	VTO	VTO
Doping Density	N_A&N_D	$N_A * S^2$ and $N_D * S^2$
Oxide Capacitance	Cox	S*Cox

Drain Current	IDS	I_{DS}*S
Power/Transistor	P	P*S
Power Density/cm^2	*p*	$p * S^3$

Review of CMOS Inverter:

In the figure, it is presented as a pair of MOS transistors, one channel n and one channel p, which represents a CMOS inverter. This is the fundamental element on which logic gates are realized and any other functions needed in CMOS circuit design.



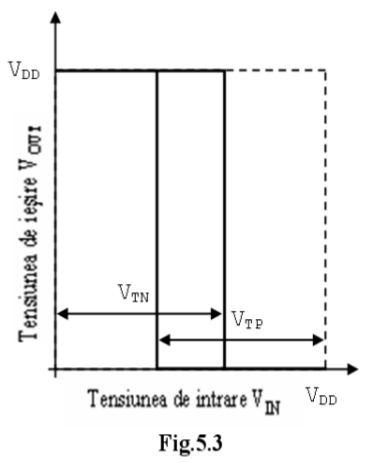


Figure 14. VTC of CMOS Inverter

When a positive direct voltage (+VDD) representing logic “1”, is applied on common gates terminal, the NMOS transistor Mn opens and the PMOS transistor, Mp will be blocked. That ends with the output at the low voltage (VSS) source, “0” logic. In the same way, if a low voltage is applied on the common gate, the PMOS transistor (Mp) will be blocked and the NMOS transistor (Mn) will be opened. In this case, the output voltage will be at a high voltage (+VDD), which is logic “1”.

DC Characteristics of CMOS Inverter:

CMOS inverters (Complementary NOSFET Inverters) are some of the most widely used and adaptable MOSFET inverters used in chip design. They operate with very little power loss and at relatively high speed. Furthermore, the CMOS inverter has good logic buffer characteristics, in that, its noise margins in both low and high states are large.

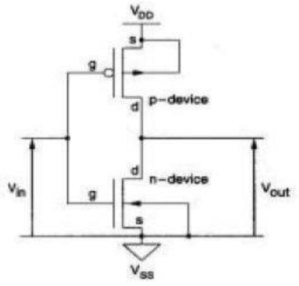


Figure 15. Logic diagram of CMOS Inverter

A CMOS inverter contains a PMOS and a NMOS transistor connected at the drain and gate terminals, a supply voltage VDD at the PMOS source terminal, and a ground connected at the NMOS source terminal, were VIN is connected to the gate terminals and VOUT is connected to the drain terminals.(given in diagram). It is important to notice that the CMOS does not contain any resistors, which makes it more power efficient that a regular resistor-MOSFET inverter. As the voltage at the input of the CMOS device varies between 0 and VDD, the state of the NMOS and PMOS varies accordingly. If we model each transistor as a simple switch activated by VIN, the inverter‘s operations can be seen very easily:The diagram given, explains when the each transistor is turning on and off. When VIN is low, the NMOS is "off", while the PMOS stays "on": instantly charging VOUT to logic high. When Vin is high, the NMOS is "on and the PMOS is "off": taking the voltage at VOUT to logic low. Before we study the DC characteristics of the inverter we should examine the ideal characteristics of inverter which is shown below. The characteristic shows that when input is zero output will high and vice versa.

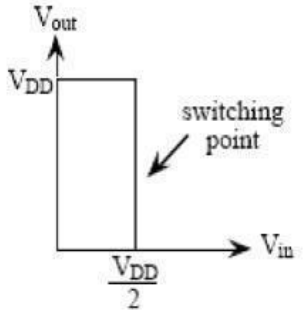


Figure.16 Ideal Characteristics of an Inverter.

The actual characteristic is also given here for the reference. Here we have shown the status of both NMOS and PMOS transistor in all the regions of the characteristics.

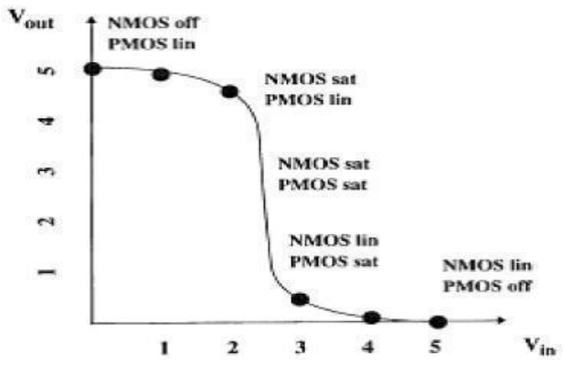


Figure.17 Actual Characteristics of an Inverter.

Figure shows five regions namely region A, B, C, D & E. also we have shown a dotted curve which is the current that is drawn by the inverter.

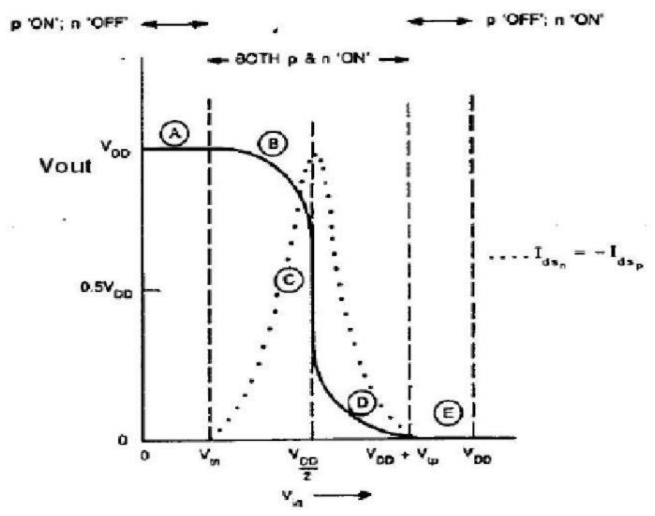


Figure.18 DC Characteristics of CMOS Inverter

Region A:

The output in this region is high because the P device is OFF and n device is ON. In region A, NMOS is cutoff region and PMOS is on, therefore output is logic high. We can analyze the inverter when it is in region B. the analysis is given below:

Region B:

The equivalent circuit of the inverter when it is region B is given below.

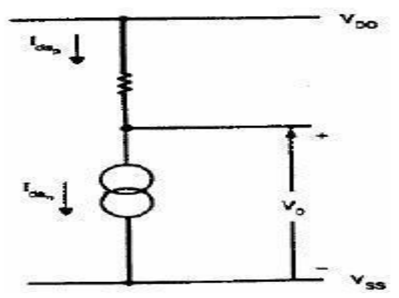


Figure.19 Equivalent circuit in Region B

In this region PMOS will be in linear region and NMOS is in saturation region. The expression for the NMOS current is

$$I_{ds_n} = \beta_n \frac{[V_{in} - V_{t_n}]^2}{2},$$

The expression for the PMOS current is

$$I_{ds_p} = -\beta_p\left[(V_{in} - V_{DD} - V_{t_p})(V_O - V_{DD}) - \frac{1}{2}(V_O - V_{DD})^2\right]$$

The expression for the voltage Vo can be written as

$$V_O = (V_{in} - V_{t_p}) + \left[(V_{in} - V_{t_p})^2 - 2\left(V_{in} - \frac{V_{DD}}{2} - V_{t_p}\right)V_{DD} - \frac{\beta_n}{\beta_p}(V_{in} - V_{t_n})^2\right]^{1/2}$$

Region C:

The equivalent circuit of CMOS inverter when it is in region C is given here. Both n and p transistors are in saturation region, we can equate both the currents and we can obtain the expression for the midpoint voltage or switching point voltage of a inverter. The corresponding equations are as follows:

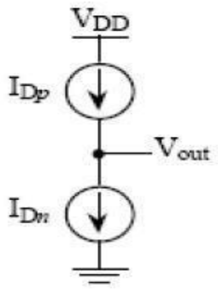


Figure.20 Equivalent circuit in Region C

The corresponding equations are as follows:

$$I_{ds_p} = \frac{1}{2}\beta_p(V_{in} - V_{DD} - V_{t_p})^2$$
$$I_{ds_n} = \frac{1}{2}\beta_n(V_{in} - V_{t_n})^2$$

By equating both the currents, we can obtain the expression for the switching point voltage as,

$$V_{in} = \frac{V_{DD} + V_{t_p} + V_{t_n}\sqrt{\frac{\beta_n}{\beta_p}}}{1 + \sqrt{\frac{\beta_n}{\beta_p}}}$$

Region D: The equivalent circuit for region D is given in the figure below. We can apply the same analysis what we did for region B and C and we can obtain the expression for output voltage.

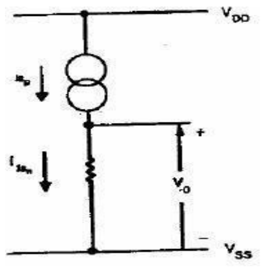


Figure.21 equivalent circuit in region D.

Region E:

The output in this region is zero because the P device is OFF and n device is ON.

Influence of βn / βp on the VTC characteristics:

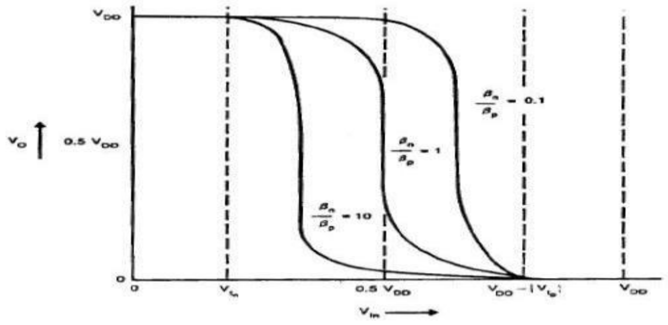


Figure.22 Effect of βn/βp ratio change on the DC characteristics of CMOS inverter.

The characteristics shifted left if the ratio of βn/βp is greater than1 (say 10). The curve shifts right if the ratio of βn / βp is lesser than 1(say 0.1). This is decided by the switching point equation of region C. The equation is repeated here the reference again.

$$V_m = V_{sp} = V_{DD} + V_{tp} + V_{tn}(\beta n/\beta p)1/2 \,/\, 1 + (\beta n/\beta p)1/2$$

Noise Margin:

Noise margin is a parameter related to input output characteristics. It determines the allowable noise voltage on the input so that the output is not affected. We will specify it in terms of two things:

- LOW noise margin
- HIGH noise margin

LOW noise margin: is defined as the difference in magnitude between the maximum Low output voltage of the driving gate and the maximum input Low voltage recognized by the driven gate.

NML=|VILmax – VOLmax|

HIGH noise margin: is defined difference in magnitude between minimum High output voltage of the driving gate and minimum input High voltage recognized by the receiving gate.

NMH = |Vohmin – VIHmin|

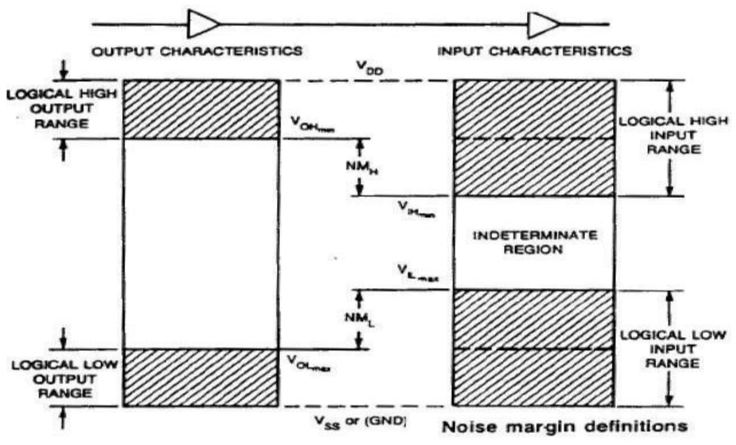


Figure.23 noise margin definitions

Figure shows how exactly we can find the noise margin for the input and output. We can also find the noise margin of a CMOS inverter. The following figure gives the idea of calculating the noise margin.

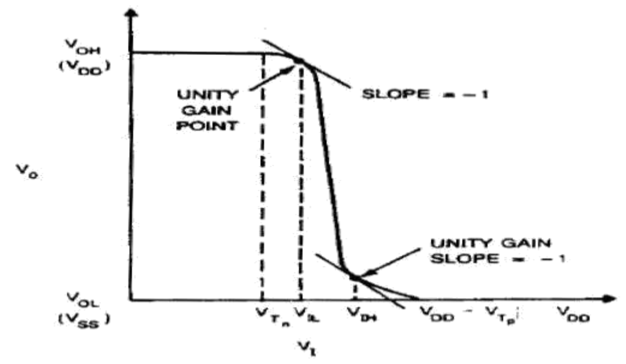


Figure.24 Noise Margin

Dynamic Behaviour of CMOS Inverter :

The dynamic of performance of a logic-circuit family is characterized by the propagation delay of its basic inverter. The inverter propagation delay (tP) is defined as the average of the low-to-high (tPLH) and the high-to-low (tPHL) propagation delays:

$$t_P = \frac{t_{PLH} + t_{PHL}}{2} \qquad (1)$$

Propagation delays tPLH and tPHL are defined as the times required for output voltage to reach the middle between the low and high logic levels, i.e. 50 % of VDD in our case of CMOS logic. The propagation delay of the CMOS inverter is determined by the time it takes to charge and discharge the capacitances present in the logic circuit. Figure 1b shows the circuit for analysis of the propagation delay of the inverter under condition that it is driving an identical inverter. To make the analysis tractable it is convenient to replace capacitances attached to the output node of primary inverter (Q1-Q2 in Figure) with equivalent capacitances between output node and ground. This is considerable simplification but individual consideration of every capacitor including nonlinear capacitances in the MOS transistor model makes a manual analysis virtually impossible. Hence, for our purposes we adopt this simplified model that is adequate for qualitative analysis and allows making estimation of CMOS inverter propagation delay. Figure shows the "inverter driving inverter circuit" where all capacitors are lumped together to form three equivalent capacitors connected between output and ground of the primary inverter. Propagation delay is calculated by taking the time difference of the 50% transition pints of the input and output waveforms. The delay expression is given as follows:

$$t_{pHL} \square \frac{t_r}{K_z} \square \log\left(\frac{C34}{\frac{1}{\square_n} \square \frac{V_{DD}}{2}}\right) \square \frac{t_r}{2} \qquad (2)$$

$$\text{Where } C34 \square [c_r \square V_{DD} \square (1 \square \frac{K_y \square (1 \square v_{thn})^{\square n \square 2}}{\square_n \square 2}) \frac{1}{\square \square_n} \square c_r \square V_{DD} \square v_{thn} \square K_y \square_n \square 1] \square \exp[K_z(1 \square v_{thn}) \square_n \square 1],$$

$$K_y \square \frac{\square t_r \square s \square \square_n}{(C_L \square C_M)(\square_n \square 1)} \square \square, \quad K_z \square \frac{\square t_r \square s \square \square_n}{C_L \square C_M} \square \square_n \square v_{thn} \square_n \quad (1)$$

t_r is the rise time of the input signal, $\square_n$ is the channel length modulation factor, VDD is the power supply voltage, v_{thn} is the normalized threshold voltage, $\square n$ represents the velocity saturation index, and C_L represents the load capacitance. C_M is the gate to drain coupling capacitance, which is proportional to the area of the inverter and reverse proportional to the oxide thickness T_{OX}. $\square_{sn} \square I_{D0} /(1\square v_{thn})^{\square n}$, where I_{D0} is the process related factor, which is proportional to the width over length ratio of a MOSFET.

Power Consumption of CMOS Inverter:

The power consumption of an inverter has two principle components: static power dissipation and dynamic power dissipation. Ideally the static power consumption of the CMOS inverter is equal to zero because the pMOS & NMOS devices are never ON simultaneously in steady state operation. But always a leakage current is flowing towards the reverse biased PN junction of the transistors located between the source or drain and the substrate. Two sources of leakage current are identified.

1. Reverse biased PN junction diode current
2. Subthreshold current

The logic diagram of CMOS inverter is shown in figure.25

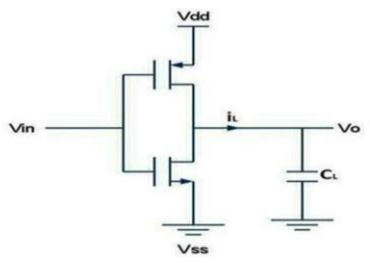


Figure 25. CMOS Inverter

The leakage current contribution is very small and can be ignored. It results in a power consumption of 0.5mW, which is not much of an issue. Another source of leakage current is potentially the subthreshold current of the transistors. MOS transistor can experience a Drain to Source current (Ids) even when Vgs is smaller than the threshold voltage. This current is called sub-threshold current. By including both sources of leakage,the resulting static power consumption is expressed as,

Pstatic = I leakage Vdd

The majority of the power is consumed during switching. Each time the capacitor is fully charged through pMOS, that is voltage rises from 0 to Vdd and a certain amount of energy is drawn from the power supply. Part of this energy is dissipated in the pMOS device while the remainder is stored on the load capacitor. During the HIGH to LOW transition, the output capacitor is discharged and the stored energy is dissipated in the nMOS transistor.

The energy drawn from the power supply during transition is,

EVdd = CL Vdd ^{2}

The energy Ec is stored on the capacitor at the end of the transition, Ec =

CL Vdd^{2} / 2

If the gate is switched ON and OFF by f times, then the dynamic power consumption is expressed as,

Pdyn = CL Vdd^{2} f

The power consumption due to direct path current is calculated from energy consumption per switching period. A direct current path exists between Vdd and Gnd for a short period of time during switching while nMOS and pMOS transistors are conducting simultaneously.

The power consumption is expressed as,

Pdp =(tr +tf /2) Vdd I peak f

Total power consumption of CMOS inverter is the sum of three components.

Ptotal = Pstatic + Pdyn + Pdp

Substitute the formulas for final expression, then it is

Ptotal = Vdd I leakage +CL Vdd f + Vdd Ipeak f (tr +tf) /

2

Small signal AC characteristics :

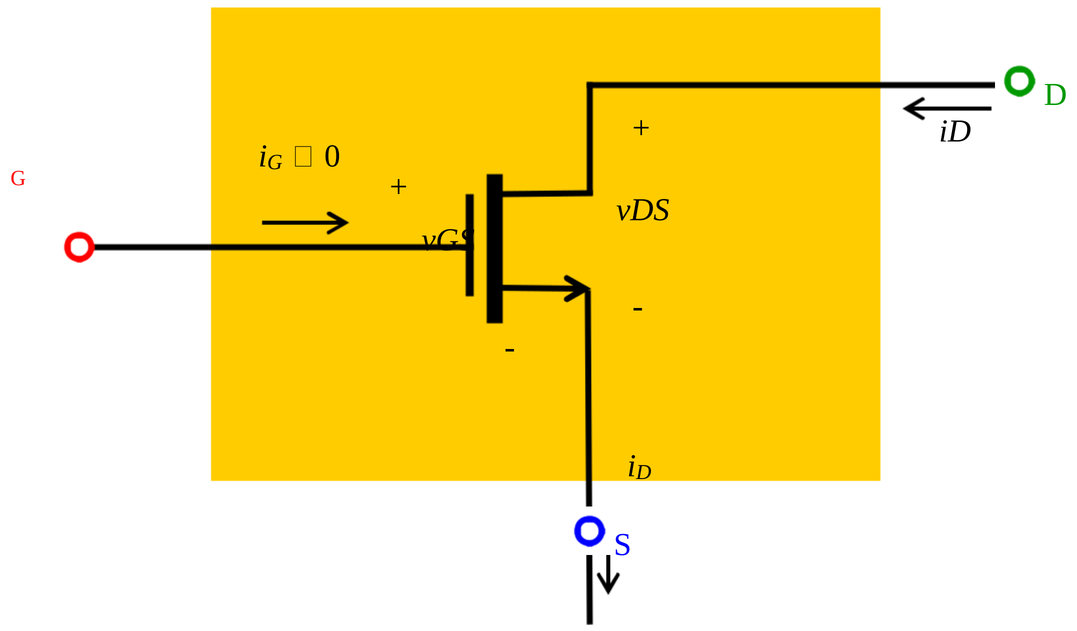
The design of most MOSFET amplifiers is divided into the separate tasks of biasing and small-signal modeling. Biasing is the adjustment of the "quiescent" (average) DC voltages and currents in the circuit so that the positive and negative excursions of the applied input signal do not cause the transistor to enter the triode or cutoff regions. In a linear amplifier, in which the output signal is intended to be a magnified replica of the input signal, it is necessary to keep the transistor operating in the saturation (constant-current) region at all times.

The biasing of an amplifier amounts to solving a DC problem, whereas small-signal modeling is an AC problem. The latter task involves analyzing how a circuit responds to incremental changes in the input voltage or current. It is therefore used to determine the signal amplification characteristics of circuits. The **small-signal circuit parameters of** MOSFET are transconductance and output

resistance.

$$g_m = 2K(V_{GS} - V_t) \qquad r_o = \frac{1}{\lambda K(V_{GS} - V_t)^2}$$

A **MOSFET** is a device with **three** terminals, called the gate, drain, and source. Its behavior is described in terms of current i_D and voltages v_{GS}, v_{DS}.



A MOSFET small-signal circuit model is:

A device with **three** terminals, called the gate, drain, and source. Its behavior is described in terms of current *id* and voltages v_{gs}, v_{ds}.

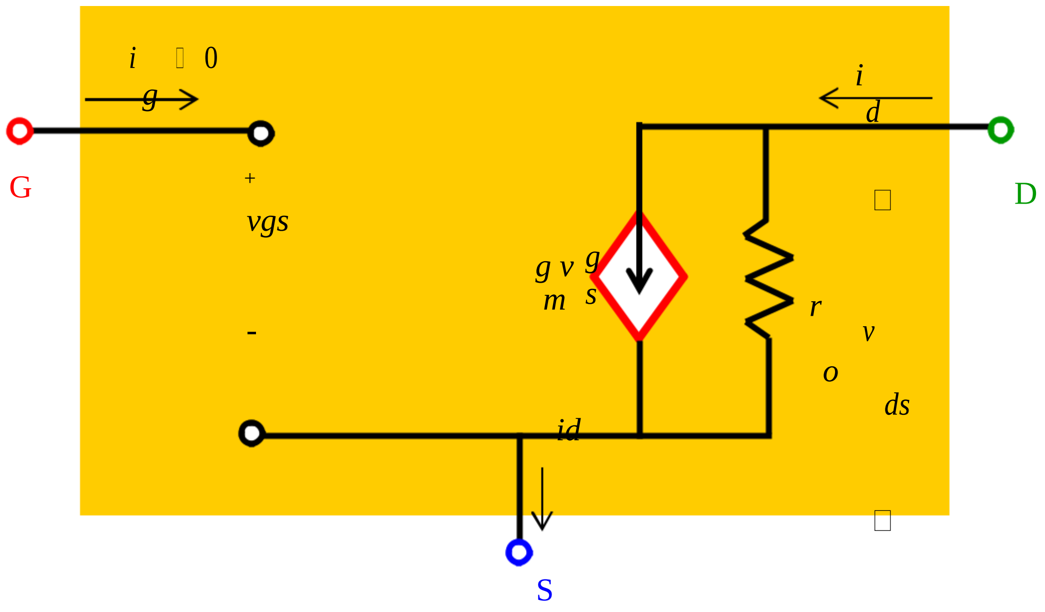


Figure 26. Small signal model of MOS transistor

QUESTIONS FOR PRACTICE

1. Explain the working of CMOS inverter in different region and derive the expression for W / L ratio.
2. Determine the expression for threshold voltage of nMOS transistor.
3. Derive the current voltage relationships of MOS transistor in non-saturation and saturation regions .
4. Draw the DC Characteristics of CMOS inverter and explain its operation.
5. Describe neatly about the dynamic behaviour of CMOS inverter.
6. Write short notes about secondary effects of MOS transistor.
7. With suitable equations, calculate the total power consumption of CMOS inverter.

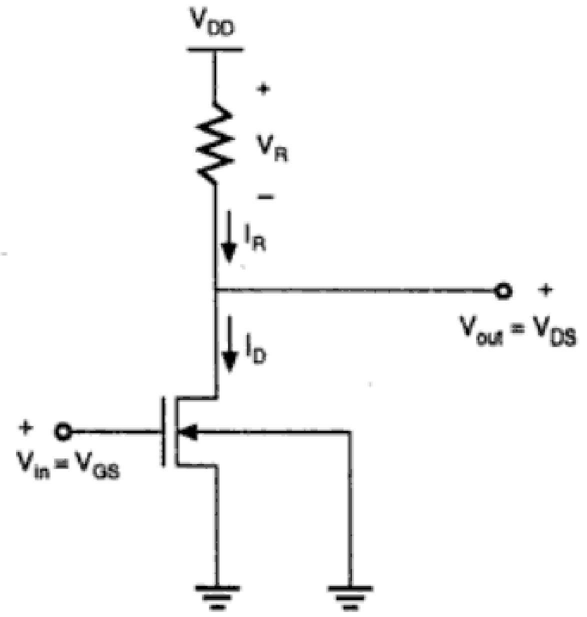
Unit –II

Combinational Logic Design

nMOS depletion load and Static CMOS design - Determination of Pull-up and Pull-down ratio-Design of Logic gates- Sizing of transistors -Stick diagrams-Lay out diagram for static CMOS - Pass transistor logic - Dynamic CMOS design - Noise considerations - Domino logic, np CMOS logic - Power consumption in CMOS gates - Multiplexers - Transmission gates design.

Resistive Load Inverter

The basic structure of a resistive load inverter is shown in the figure given below. Here, enhancement type nMOS acts as the driver transistor. The load consists of a simple linear resistor R_L. The power supply of the circuit is V_{DD} and the drain current I_D is equal to the load current I_R.



Circuit Operation

When the input of the driver transistor is less than threshold voltage V_{TH} ($V_{in} < V_{TH}$), driver transistor is in the cut – off region and does not conduct any current. So, the voltage drop across the load resistor is ZERO and output voltage is equal to the V_{DD}. Now, when the input voltage increases further, driver transistor will start conducting the non-zero current and nMOS goes in saturation region.

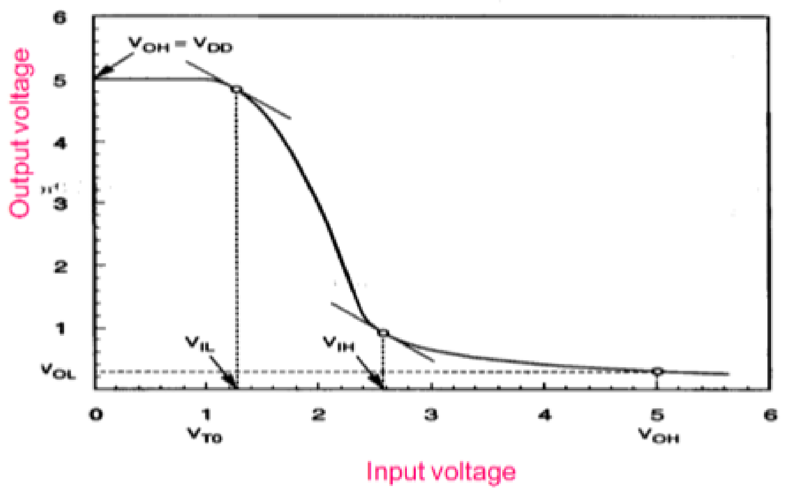
Mathematically,

$$I_{D} = \frac{K_{n}}{2}\left [V_{GS}-V_{TO} \right]^{2}$$

Increasing the input voltage further, driver transistor will enter into the linear region and output of the driver transistor decreases.

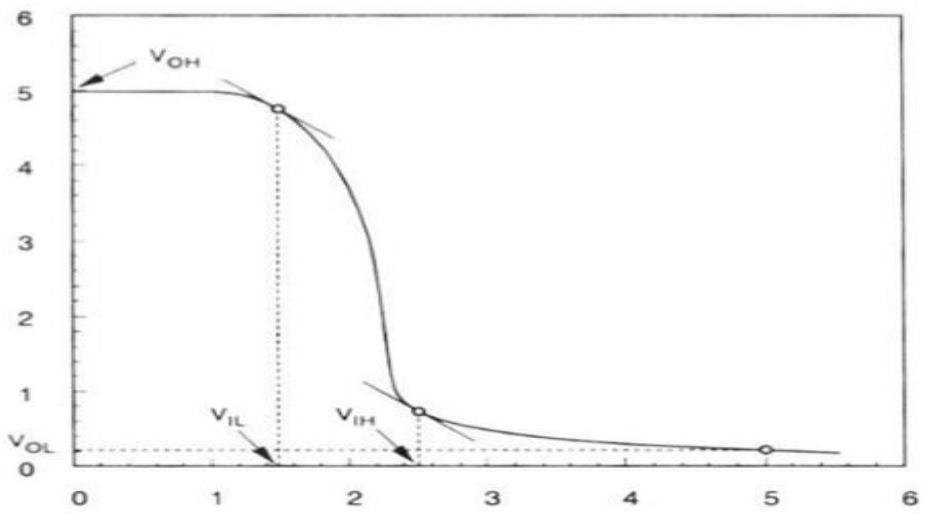
$$I_{D} = \frac{K_{n}}{2}2\left [V_{GS}-V_{TO} \right]V_{DS}-V_{DS}^{2}$$

VTC of the resistive load inverter, shown below, indicates the operating mode of driver transistor and voltage points.

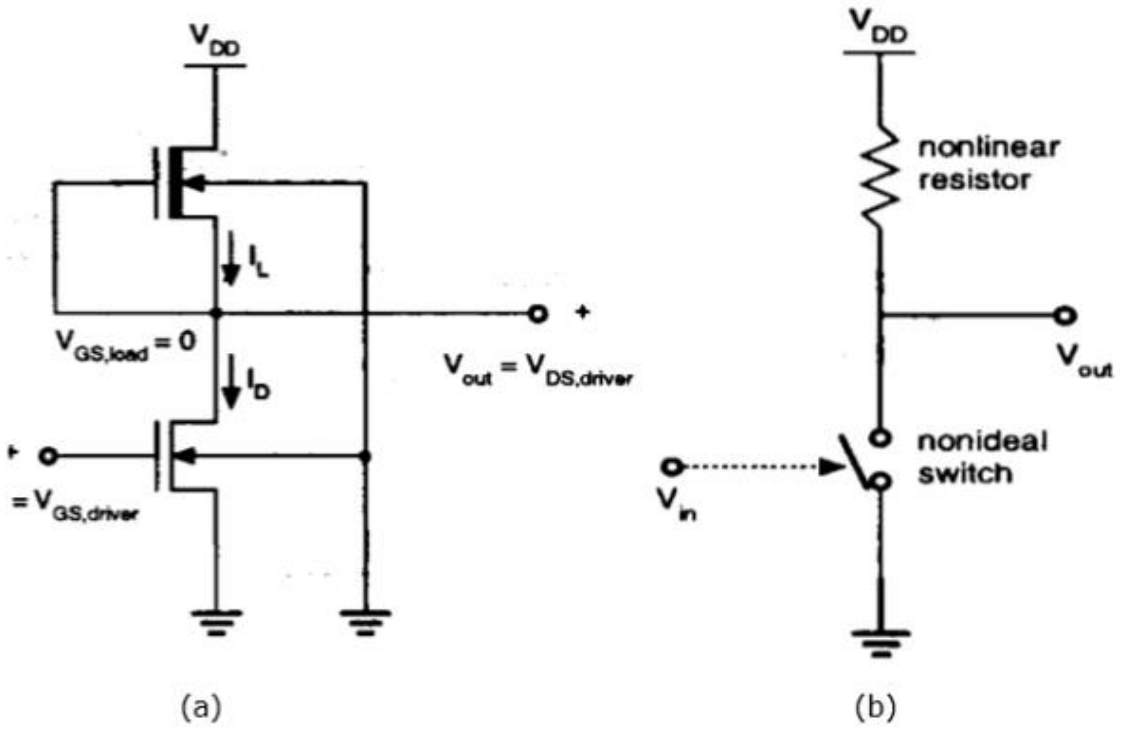


Inverter with N type MOSFET Load

The main advantage of using MOSFET as load device is that the silicon area occupied by the transistor is smaller than the area occupied by the resistive load. Here, MOSFET is active load and inverter with active load gives a better performance than the inverter with resistive load.



Depletion Load NMOS



Drawbacks of the enhancement load inverter can be overcome by using depletion load inverter. Compared to enhancement load inverter, depletion load inverter requires few more fabrication steps for channel implant to adjust the threshold voltage of load.

The advantages of the depletion load inverter are - sharp VTC transition, better noise margin, single power supply and smaller overall layout area.

As shown in the figure, the gate and source terminal of load are connected;

So, $V_{GS} = 0$. Thus, the threshold voltage of the load is negative. Hence,

$$V_{GS,load}> V_{T,load}$$ is satisfied

Therefore, load device always has a conduction channel regardless of input and output voltage level.

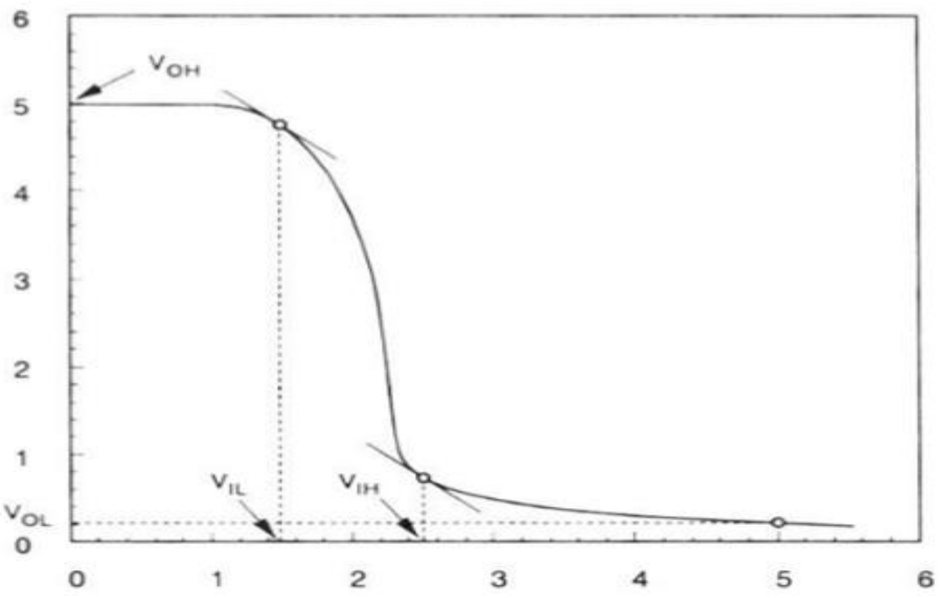
When the load transistor is in saturation region, the load current is given by

$$I_{D,load} = \frac{K_{n,load}}{2}\left [-V_{T,load}\left (V_{out} \right) \right]^{2}$$

When the load transistor is in linear region, the load current is given by

$$I_{D,load} = \frac{K_{n,load}}{2}\left [2\left | V_{T,load}\left (V_{out} \right) \right |.\left (V_{DD}-V_{out} \right)-\left (V_{DD}-V_{out} \right)^{2} \right]$$

The voltage transfer characteristics of the depletion load inverter is shown in the figure given below −



Static CMOS Design

Introduction - Complementary Static CMOS design offers low noise sensitivity, speed and low power consumption. Static CMOS design means that at any time, the output of the gate is directly connected to VSS or VDD. In comparison, Dynamic CMOS gates depend upon the device capacitance to temporarily hold charge at the output.

Properties of Complementary Static CMOS

6. Contains a pull up network (PUP) and pull down network (PDN)

7. PUP networks consist of PMOS transistors
8. PDN networks consist of NMOS transistors
9. Each network is the dual of the other network
10. The output of the complementary gate is inverted.

Pull up and Pull down networks

PDN Design - Any function is first designed by the Pull Down network. The Pull Up network can be designed by simply taking the dual of the Pull Down network, once it is found. Note: An NMOS gate will pass current between source and drain when 5 volts is presented at the gate.

- All functions are composed of either and'ed or or'ed sub-functions.
- The And function is composed of NMOS transistors in series .
- The Or function is composed of NMOS transistors in paralle l.

This key idea is completely opposite that of pull-up network.

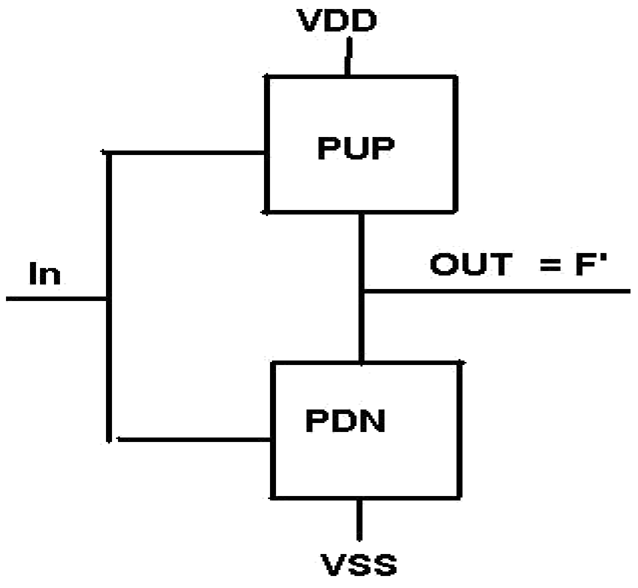


Figure. Static CMOS design

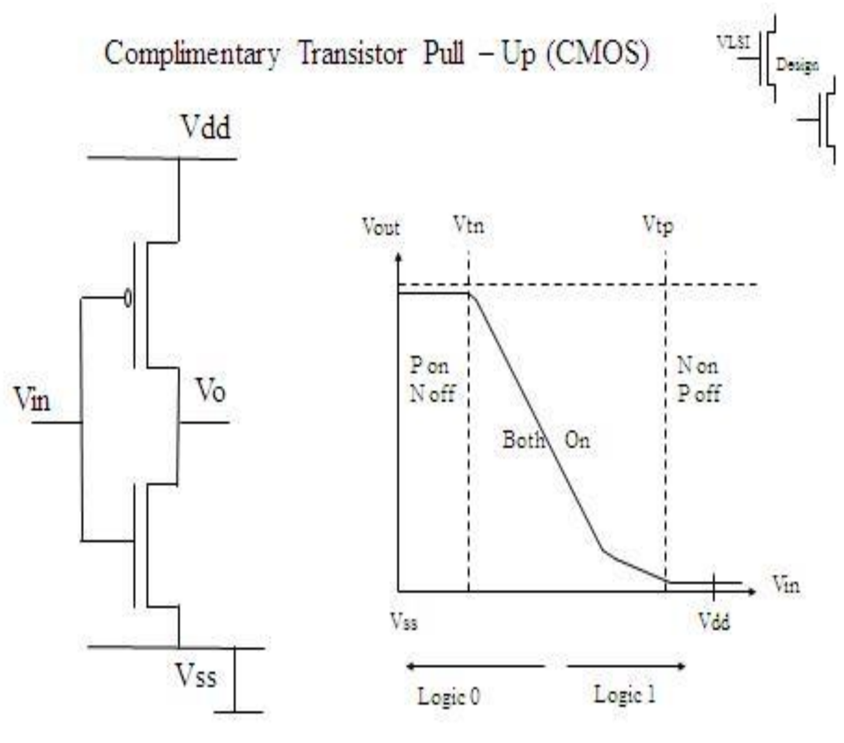


Figure. Static CMOS Inverter

Design of Logic gates and Stick Diagrams :

MOS circuits are formed on four basic layers:

> N-diffusion

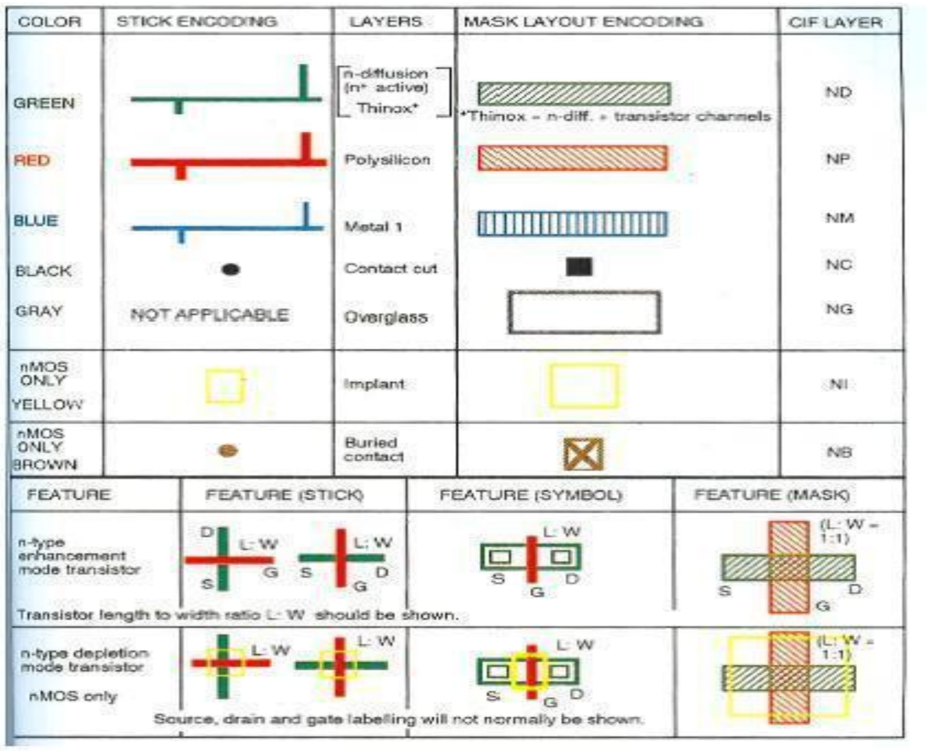
> P-diffusion

> Polysilicon

> Metal

These layers are isolated by one another by thick or thin silicon dioxide insulating layers. Thin oxide mask region includes n-diffusion / p-diffusion and transistor channel. Stick diagrams may be used to convey layer information through the use of a color code.

For example: n-diffusion--green poly--red blue-- metal yellow--implant black--
contact areas. Encodings for NMOS process:



COLOR	STICK ENCODING	LAYERS	MASK LAYOUT ENCODING	CIF LAYER
GREEN		n-diffusion (n⁺ active) Thinox*	▨ *Thinox = n-diff. + transistor channels	ND
RED		Polysilicon	▨	NP
BLUE		Metal 1	▥	NM
BLACK	●	Contact cut	■	NC
GRAY	NOT APPLICABLE	Overglass	□	NG
nMOS ONLY YELLOW	□	Implant	□	NI
nMOS ONLY BROWN	●	Buried contact	☒	NB

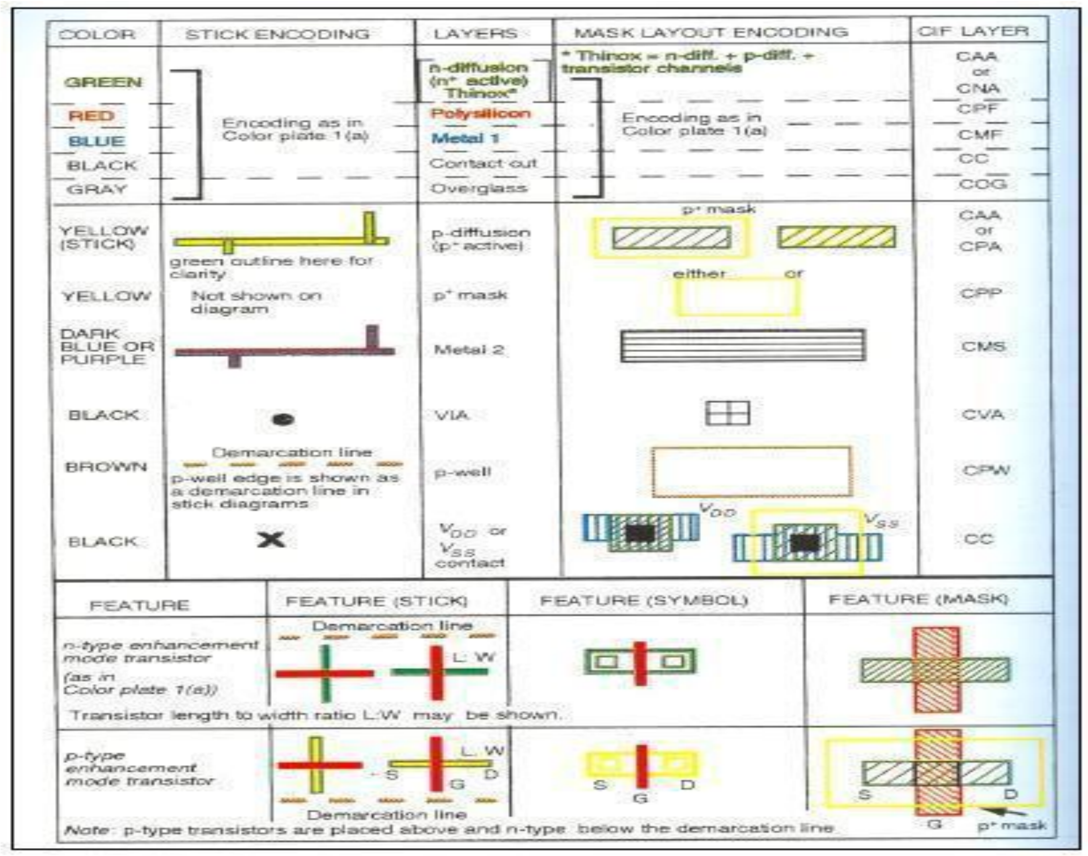
FEATURE	FEATURE (STICK)	FEATURE (SYMBOL)	FEATURE (MASK)
n-type enhancement mode transistor	D L:W G S; L:W S D G	L:W S G D	(L:W = 1:1) S D G
Transistor length to width ratio L:W should be shown.			
n-type depletion mode transistor nMOS only	L:W; L:W	L:W S G D	(L:W = 1:1)
	Source, drain and gate labelling will not normally be shown.		

Figure. NMOS encodings.

Figure shows the way of representing different layers in stick diagram notation and mask layout using nmos style.

Figure shows when a n-transistor is formed: a transistor is formed when a green line (n+ diffusion) crosses a red line (poly) completely. Figure also shows how a depletion mode transistor is represented in the stick format.

Encodings for CMOS process:



COLOR	STICK ENCODING	LAYERS	MASK LAYOUT ENCODING	CIF LAYER
GREEN	Encoding as in Color plate 1(a)	n-diffusion (n⁺ active) Thinox*	* Thinox = n-diff. + p-diff. + transistor channels; Encoding as in Color plate 1(a)	CAA or CNA
RED		Polysilicon		CPF
BLUE		Metal 1		CMF
BLACK		Contact cut		CC
GRAY		Overglass		COG
YELLOW (STICK)	green outline here for clarity	p-diffusion (p⁺ active)	p⁺ mask; either or	CAA or CPA
YELLOW	Not shown on diagram	p⁺ mask		CPP
DARK BLUE OR PURPLE		Metal 2		CMS
BLACK	●	VIA		CVA
BROWN	Demarcation line; p-well edge is shown as a demarcation line in stick diagrams	p-well		CPW
BLACK	✕	V_{DD} or V_{SS} contact	V_{DD} V_{SS}	CC

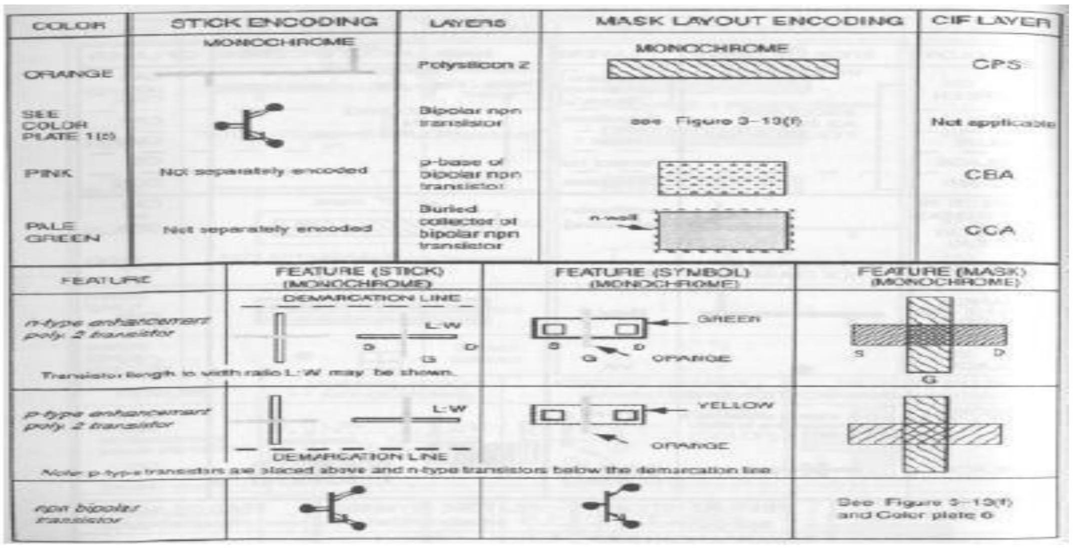
FEATURE	FEATURE (STICK)	FEATURE (SYMBOL)	FEATURE (MASK)
n-type enhancement mode transistor (as in Color plate 1(a))	Demarcation line; L:W		
Transistor length to width ratio L:W may be shown.			
p-type enhancement mode transistor	L:W; S G D; Demarcation line	S G D	S G D; p⁺ mask
Note: p-type transistors are placed above and n-type below the demarcation line			

Figure. CMOS encodings.

Figure shows when a n-transistor is formed: a transistor is formed when a green line (n+ diffusion) crosses a red line (poly) completely.

Figure also shows when a p-transistor is formed: a transistor is formed when a yellow line (p+ diffusion) crosses a red line (poly) completely.

Encoding for BJT and MOSFETs:



COLOR	STICK ENCODING	LAYERS	MASK LAYOUT ENCODING	CIF LAYER
	MONOCHROME		MONOCHROME	
ORANGE		Polysilicon 2	▨	CPS
SEE COLOR PLATE 1(f)		Bipolar npn transistor	see Figure 3-13(f)	Not applicable
PINK	Not separately encoded	p-base of bipolar npn transistor		CBA
PALE GREEN	Not separately encoded	Buried collector of bipolar npn transistor	n-well	CCA

FEATURE	FEATURE (STICK) (MONOCHROME)	FEATURE (SYMBOL) (MONOCHROME)	FEATURE (MASK) (MONOCHROME)
n-type enhancement poly 2 transistor	DEMARCATION LINE L:W S G D Transistor length to width ratio L:W may be shown.	GREEN S G D ORANGE	S D G
p-type enhancement poly 2 transistor	L:W DEMARCATION LINE Note p-type transistors are placed above and n-type transistors below the demarcation line	YELLOW ORANGE	
npn bipolar transistor			See Figure 3-13(f) and Color plate 6

Figure. BiCMOS encodings.

There are several layers in an nMOS chip:

_ a p-type substrate

_ paths of n-type diffusion

_ a thin layer of silicon
dioxide

_ paths of polycrystalline
silicon

a thick layer of silicon dioxide

_ paths of metal (usually aluminum)

_ a further thick layer of silicon dioxide

contact cuts through the silicon dioxide can be used wherever connections are required. The three layers carrying paths can be considered as independent conductors that only interact where polysilicon crosses diffusion to form a transistor.

These tracks can be drawn as stick diagrams with _ diffusion in green _ polysilicon in red _ metal in blue using black to indicate contacts between layers and yellow to mark regions of implant in the channels of depletion mode transistors.

With CMOS there are two types of diffusion: n-type is drawn in green and p-type in brown. These are on the same layers in the chip and must not meet. In fact, the method of fabrication required that they be kept relatively far apart. Modern CMOS processes usually support more than one layer of metal. Two are common and three or more are often available.

Actually, these conventions for colors are not universal; in particular, industrial (rather than academic) systems tend to use red for diffusion and green for polysilicon. Moreover, a shortage of colored pens normally means that both types of diffusion in CMOS are colored green and the polarity indicated by drawing a circle round p-type transistors or simply inferred from the context. Colorings for multiple layers of metal are even less standard.

There are three ways that an nMOS inverter might be drawn:

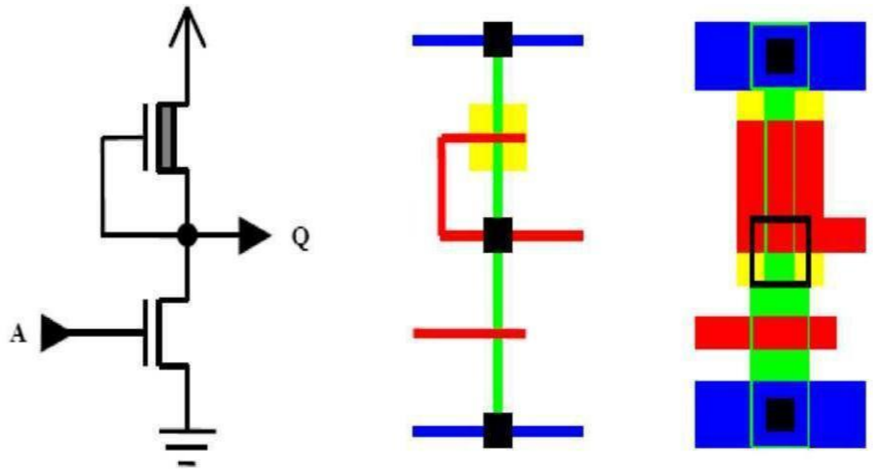


Figure. nMOS depletion load inverter.

Figure shows schematic, stick diagram and corresponding layout of nMOS depletion load inverter.

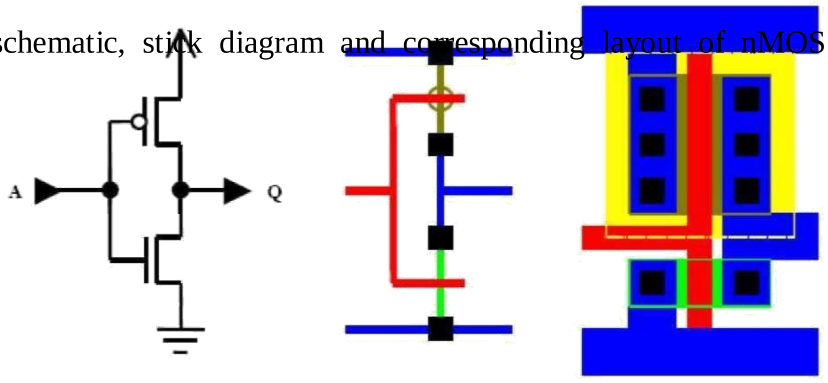


Figure. CMOS inverter

Figure shows the schematic, stick diagram and corresponding layout of CMOS inverter.

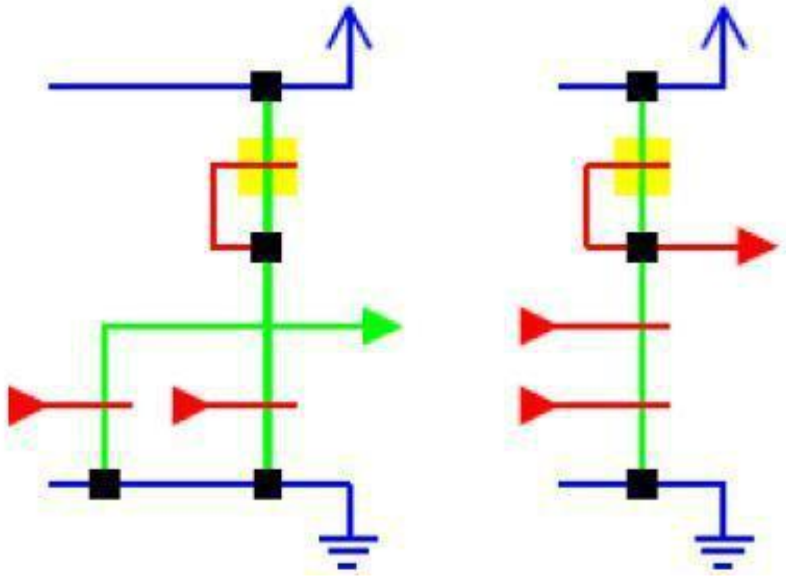


Figure. Stick diagrams for nMOS NOR and NAND.

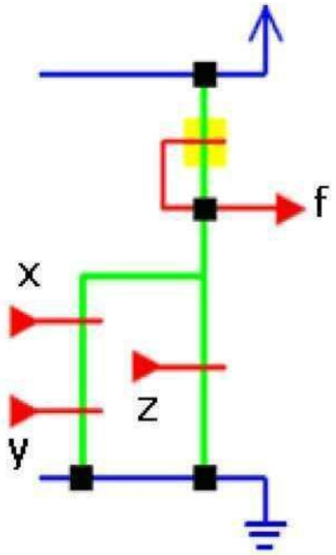


Figure. Stick diagram of a given function f.

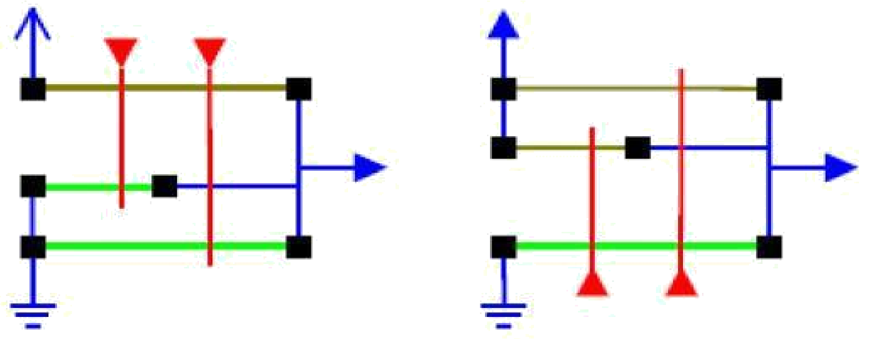


Figure. Stick diagram of nMOS implementation of the function f= [(xy) +z]'.

Figure shows the stick diagram CMOS NOR and NAND, where we can see that the p diffusion line never touched the n diffusion directly, it is always joined using a blue color metal line.

NMOS and CMOS Design style:

In the NMOS style of representing the sticks for the circuit, we use only NMOS transistor, in CMOS we need to differentiate n and p transistor, that is usually by the color or in monochrome diagrams we will have a demarcation line. Above the demarcation line are the p transistors and below the demarcation are the n transistors. Following stick shows CMOS circuit example in monochrome where we utilize the demarcation line.

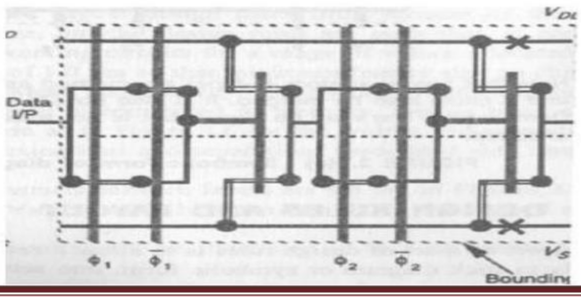


Figure. Stick diagram of dynamic shift register in CMOS style.

Figure shows the stick diagram of dynamic shift register using CMOS style. Here the output of the TG is connected as the input to the inverter and the same chain continues depending the number of bits.

Design Rules:

Design rules include width rules and spacing rules. Mead and Conway developed a set of simplified scalable X -based design rules, which are valid for a range of fabrication technologies. In these rules, the minimum feature size of a technology is characterized as 2 X. All width and spacing rules are specified in terms of the parameter X. Suppose we have design rules that call for a minimum width of 2 X, and a minimum spacing of 3 X . If we select a 2 um technology (i.e., X = 1 um), the above rules are translated to a minimum width of 2 um and a minimum spacing of 3 um. On the other hand, if a 1 um technology (i.e., X = 0.5 um) is selected, then the same width and spacing rules are now specified as 1 um and 1.5 um, respectively.

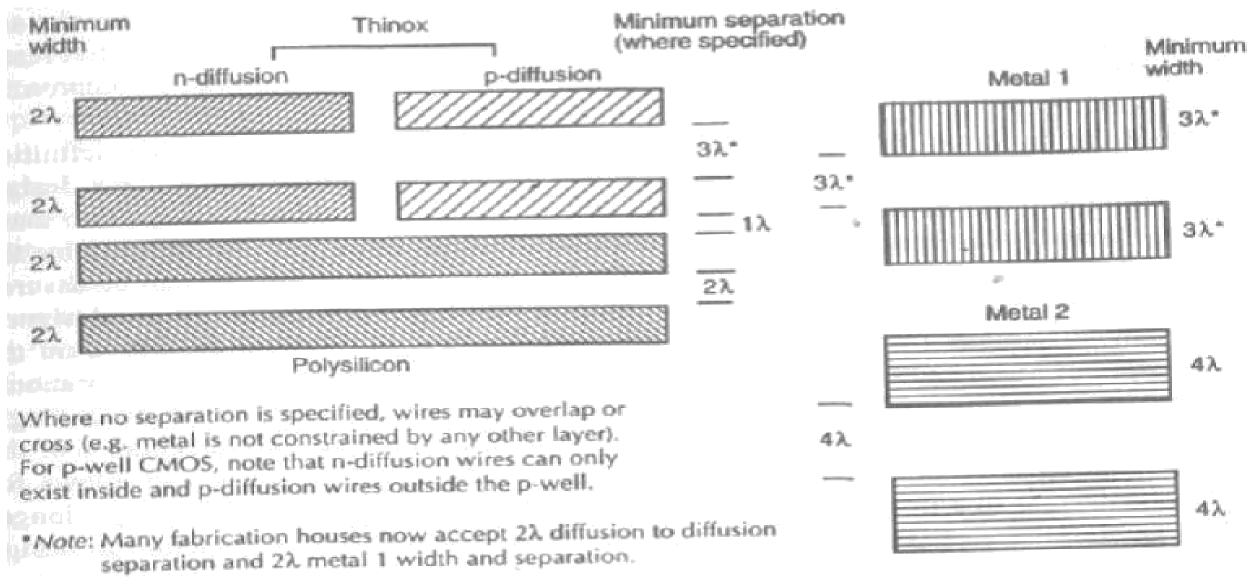

Minimum width
Thinox
Minimum separation (where specified)
Minimum width
n-diffusion
p-diffusion
Metal 1
2λ
3λ*
3λ*
3λ*
2λ
3λ*
1λ
2λ
2λ
Metal 2
2λ
Polysilicon
4λ
Where no separation is specified, wires may overlap or cross (e.g. metal is not constrained by any other layer). For p-well CMOS, note that n-diffusion wires can only exist inside and p-diffusion wires outside the p-well.
4λ
4λ
*Note: Many fabrication houses now accept 2λ diffusion to diffusion separation and 2λ metal 1 width and separation.

Figure. Design rules for the diffusion layers and metal layers.

Figure shows the design rule n diffusion, p diffusion, poly, metal1 and metal 2. The n and p diffusion lines is having a minimum width of 2λ and a minimum spacing of 3λ.Similarly we are showing for other layers.

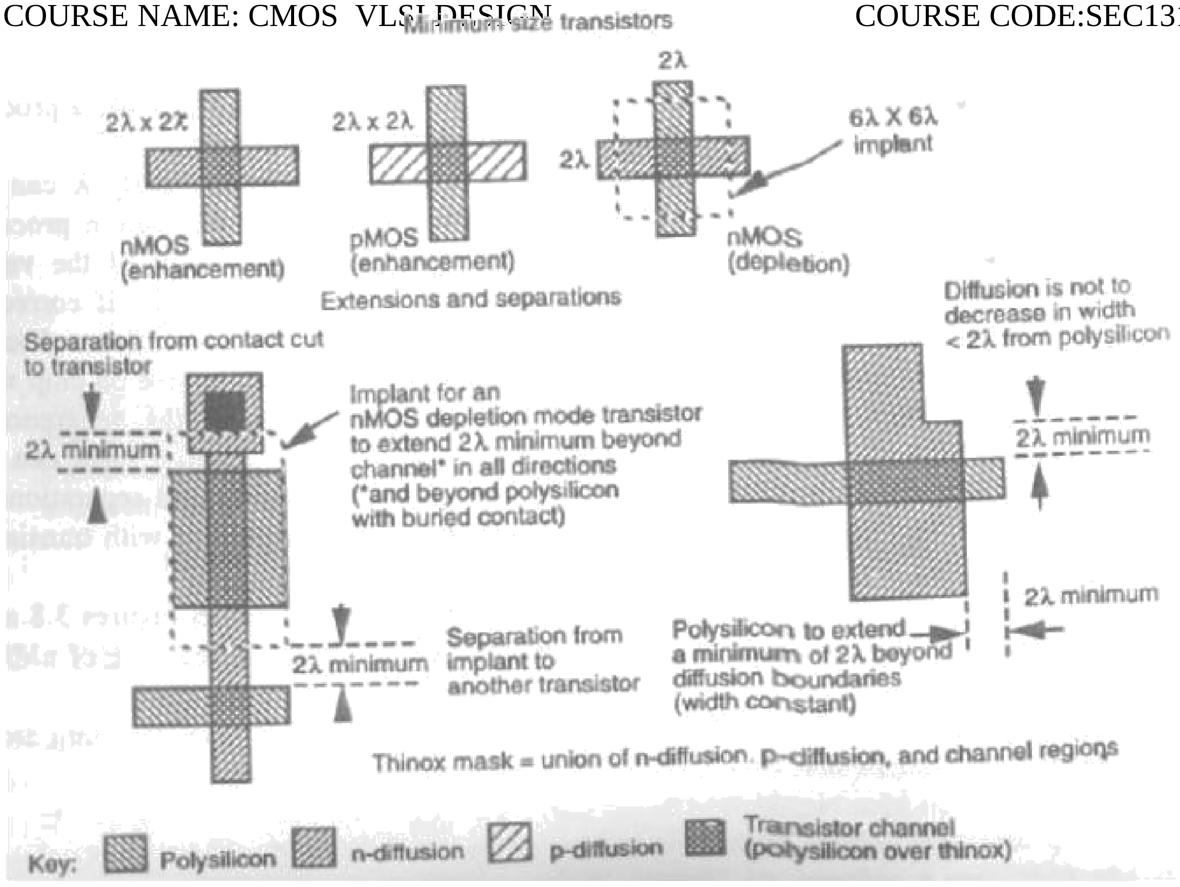

Minimum-size transistors
2λ x 2λ
2λ x 2λ
2λ
2λ
6λ X 6λ implant
nMOS (enhancement)
pMOS (enhancement)
nMOS (depletion)
Extensions and separations
Separation from contact cut to transistor
2λ minimum
Implant for an nMOS depletion mode transistor to extend 2λ minimum beyond channel* in all directions (*and beyond polysilicon with buried contact)
Diffusion is not to decrease in width < 2λ from polysilicon
2λ minimum
2λ minimum
Separation from implant to another transistor
Polysilicon to extend a minimum of 2λ beyond diffusion boundaries (width constant)
2λ minimum
Thinox mask = union of n-diffusion, p-diffusion, and channel regions
Key: Polysilicon n-diffusion p-diffusion Transistor channel (polysilicon over thinox)

Figure. Design rules for transistors and gate over hang distance.

Figure shows the design rule for the transistor, and it also shows that the poly should extend for a minimum of 7k beyond the diffusion boundaries. (gate over hang distance)

Via is used to connect higher level metals from metal connection.

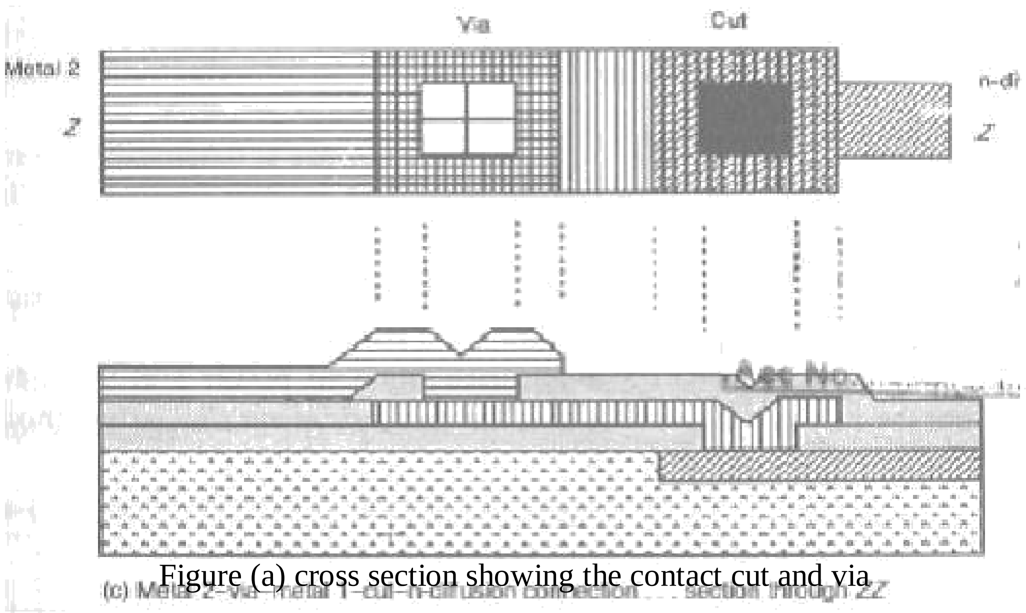

Via
Cut
Metal 2
Z
n-di
Z'
(c) Metal 2—via—metal 1—cut—n-diffusion connection — section through ZZ'

Figure (a) cross section showing the contact cut and via

Figure shows the design rules for contact cuts and Vias. The design rule for contact is minimum2λx2λ and same is applicable for a Via.

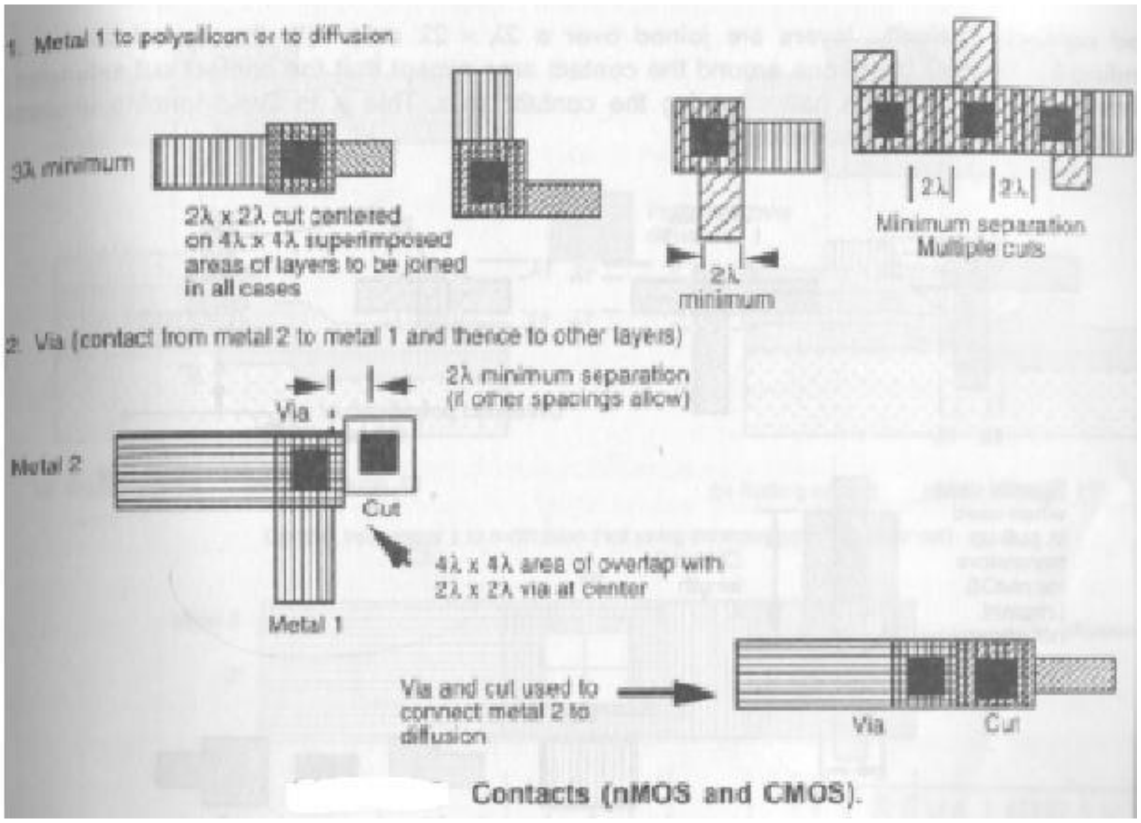

1. Metal 1 to polysilicon or to diffusion
3λ minimum
2λ x 2λ cut centered on 4λ x 4λ superimposed areas of layers to be joined in all cases
2λ minimum
2λ
2λ
Minimum separation Multiple cuts
2. Via (contact from metal 2 to metal 1 and thence to other layers)
Via
2λ minimum separation (if other spacings allow)
Metal 2
Cut
4λ x 4λ area of overlap with 2λ x 2λ via at center
Metal 1
Via and cut used to connect metal 2 to diffusion
Via
Cut
Contacts (nMOS and CMOS).

Figure (b). Design rules for contact cuts and vias

Buried contact: The contact cut is made down each layer to be joined and it is shown in figure.

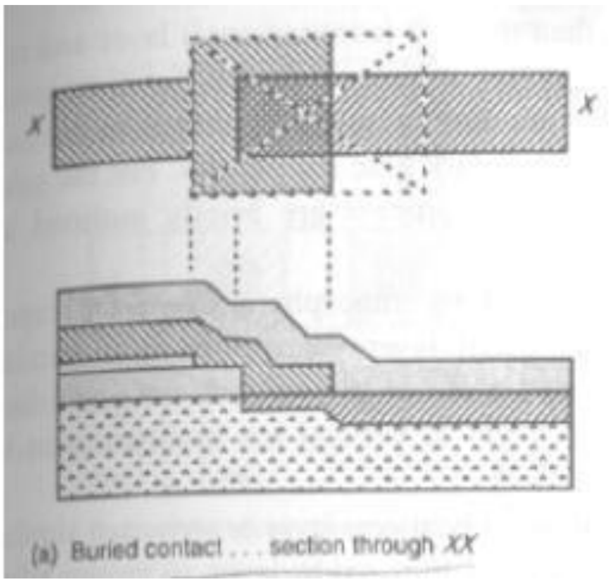


Figure. Buried contact.

Butting contact: The layers are butted together in such a way the two contact cuts become contiguous. We can better under the butting contact from figure 15.

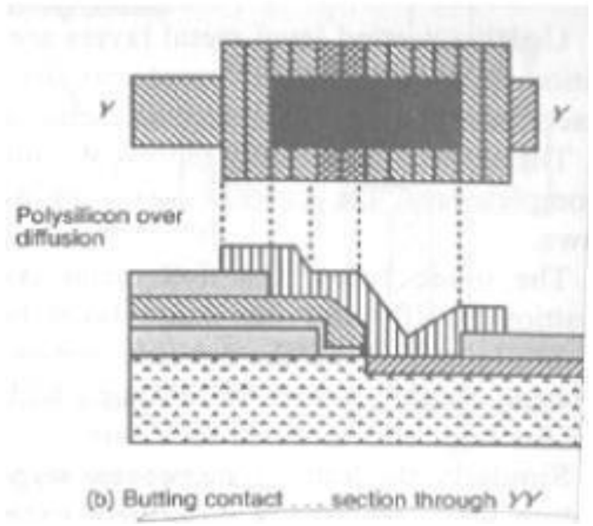

Y
Y
Polysilicon over diffusion
(b) Butting contact . . . section through YY

Figure. Butting contact.

CMOS LAMBDA BASED DESIGN RULES:

Till now we have studied the design rules wrt only NMOS, what are the rules to be followed if we have the both p and n transistor on the same chip will be made clear with the diagram. Figure shows the rules to be followed in CMOS well processes to accommodate both n and p transistors.

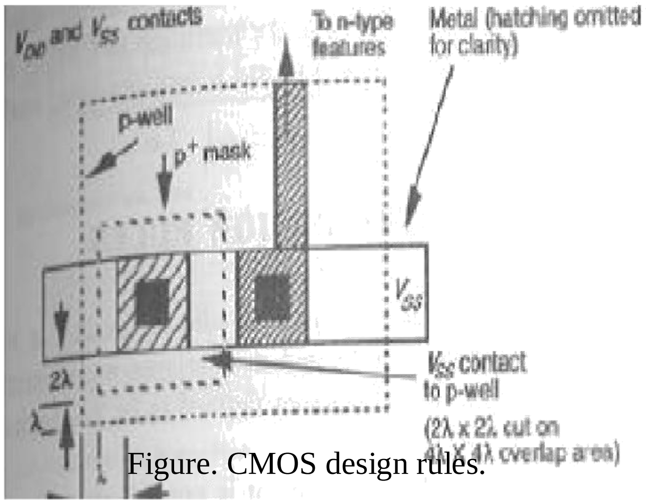

V_{DD} and V_{SS} contacts
To n-type features
Metal (hatching omitted for clarity)
p-well
p^+ mask
V_{SS}
V_{SS} contact to p-well
($2\lambda \times 2\lambda$ cut on $4\lambda \times 4\lambda$ overlap area)
2λ
λ

Figure. CMOS design rules.

Orbit 2μm CMOS process:

In this process all the spacing between each layers and dimensions will be in terms micrometer. The 2^m here represents the feature size. All the design rules whatever we have seen will not have lambda instead it will have the actual dimension in micrometer.

In one way lambda based design rules are better compared micrometer based design rules, that is lambda based rules are feature size independent.

Figure shows the design rule for BiCMOS process using orbit 2um process.

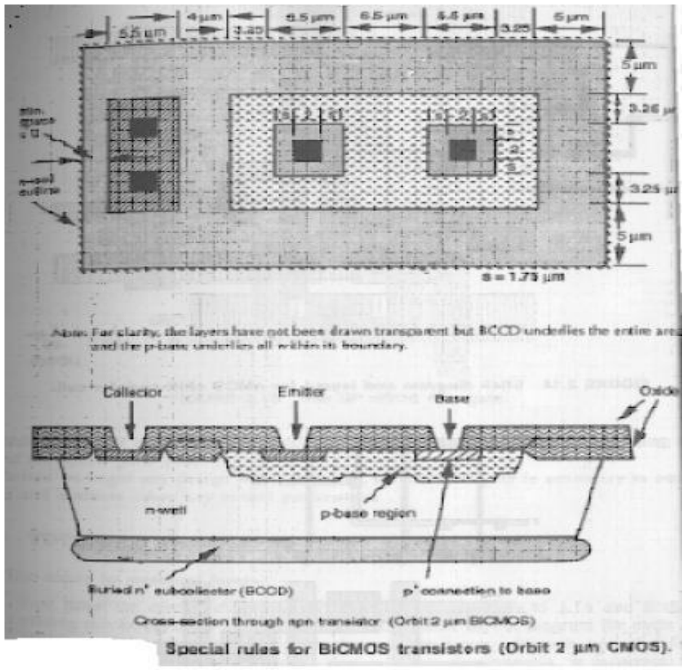

5.5 μm
4 μm
3.25
6.5 μm
6.5 μm
6.5 μm
3.25
5 μm
5 μm
3.25 μm
3.25 μm
5 μm
s = 1.75 μm
Note: For clarity, the layers have not been drawn transparent but BCCD underlies the entire area and the p-base underlies all within its boundary.
Collector
Emitter
Base
Oxide
n-well
p-base region
Buried n⁺ subcollector (BCCD)
p⁺ connection to base
Cross-section through npn transistor (Orbit 2 μm BiCMOS)
Special rules for BiCMOS transistors (Orbit 2 μm CMOS).

Figure. BiCMOS design rules.

The following is the example stick and layout for 2way selector with enable (2:1 MUX).

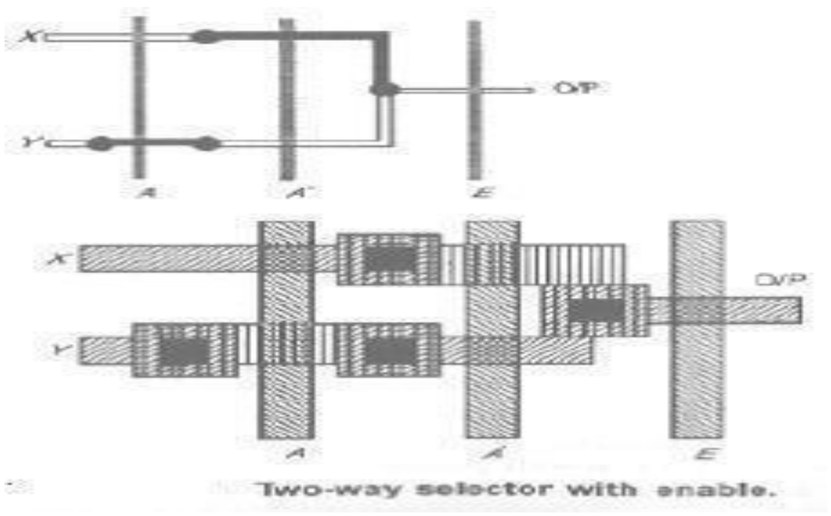


Figure. Two way selector stick and layout

BASIC PHYSICAL DESIGN AN OVERVIEW

The VLSI design flow for any IC design is as follows

1 .Specification	(problem definition)
2. Schematic (gate level design)	(equivalence check)
3. Layout	(equivalence check)

4. Floor Planning
5 .Routing, Placement
6. On to Silicon

When the devices are represented using these layers, we call it physical design. The design is carried out using the design tool, which requires to follow certain rules. Physical structure is required to study the impact of moving from circuit to layout. When we draw the layout from the schematic, we are taking the first step towards the physical design. Physical design is an important step towards fabrication. Layout is representation of a schematic into layered diagram.

This diagram reveals the different layers like ndiff, polysilicon etc that go into

formation of the device. At every stage of the physical design simulations are carried out to verify whether the design is as per requirement. Soon after the layout design the DRC check is used to verify minimum dimensions and spacing of the layers. Once the layout is done, a layout versus schematic check carried out before proceeding further. There are different tools available for drawing the layout and simulating it.

The simplest way to begin a layout representation is to draw the stick diagram. But as the complexity increases it is not possible to draw the stick diagrams. For beginners it easy to draw the stick diagram and then proceed with the layout for the basic digital gates. We will have a look at some of the things we should know before starting the layout. In the schematic representation lines drawn between device terminals represent interconnections and any no planar situation can be handled by crossing over. But in layout designs a little more concern about the physical interconnection of different layers. By simply drawing one layer above the other it not possible to make interconnections, because of the different characters of each layer. Contacts have to be made whenever such interconnection is required. The power and the ground connections are made using the metal and the common gate connection using the polysilicon. The metal and the diffusion layers are connected using contacts. The substrate contacts are made for same source and substrate voltage. Which are not implied in the schematic. These layouts are governed by DRC's and have to be atleast of the minimum size depending on the technology used. The crossing over of layers is another aspect which is of concern and is addressed next.

1. Poly crossing diffusion makes a transistor

2. Metal of the same kind crossing causes a short.

3. Poly crossing a metal causes no interaction unless a contact is made.

Different design tricks need to be used to avoid unknown creations. Like a combination of metal1 and metal 2 can be used to avoid short. Usually metal 2 is used for the global vdd and vss lines and metal1 for local connections.

SCHEMATIC AND LAYOUT OF BASIC GATES

1. CMOS INVERTER/NOT GATE SCHEMATIC

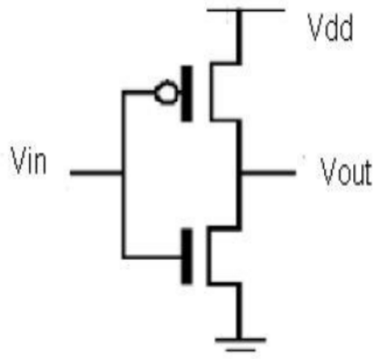


Figure. Inverter.

TOWARDS THE LAYOUT

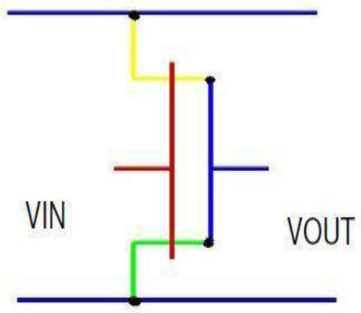


Figure. Stick diagram of inverter.

The diagram shown here is the stick diagram for the CMOS inverter. It consists of a Pmos and a Nmos connected to get the inverted output. When the input is low, Pmos (yellow) is on and pulls the output to vdd; hence it is called pull up device. When Vin =1, Nmos (green) is on it pulls Vout to Vss, hence Nmos is a pull down device. The red lines are the poly silicon lines connecting the gates and the blue lines are the metal lines for VDD (up) and VSS (down).The layout of the cmos inverter is shown below. Layout also gives the minimum dimensions of different layers, along with the logical connections and main thing about layouts is that can be simulated and checked for errors which cannot be done with only stick diagrams.

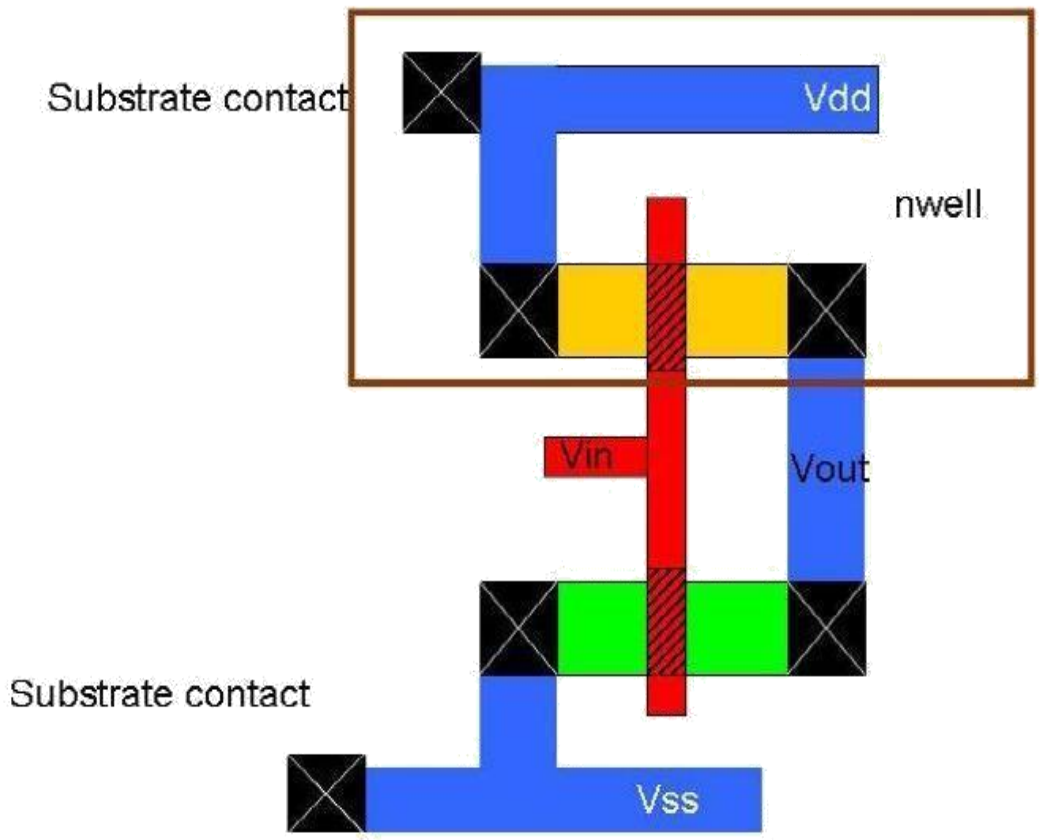


Figure. Layout of inverter.

The layout shown above is that of a CMOS inverter. It consists of a pdiff (yellow colour) forming the pmos at the junction of the diffusion and the polysilicon (red colour) shown hatched ndiff (green) forming the nmos(area hatched).The different layers drawn are checked for their dimensions using the DRC rule check of the tool used for drawing. Only after the DRC (design rule check) is passed the design can proceed further. Further the design undergoes Layout Vs Schematic checks and finally the parasitic can be extracted.

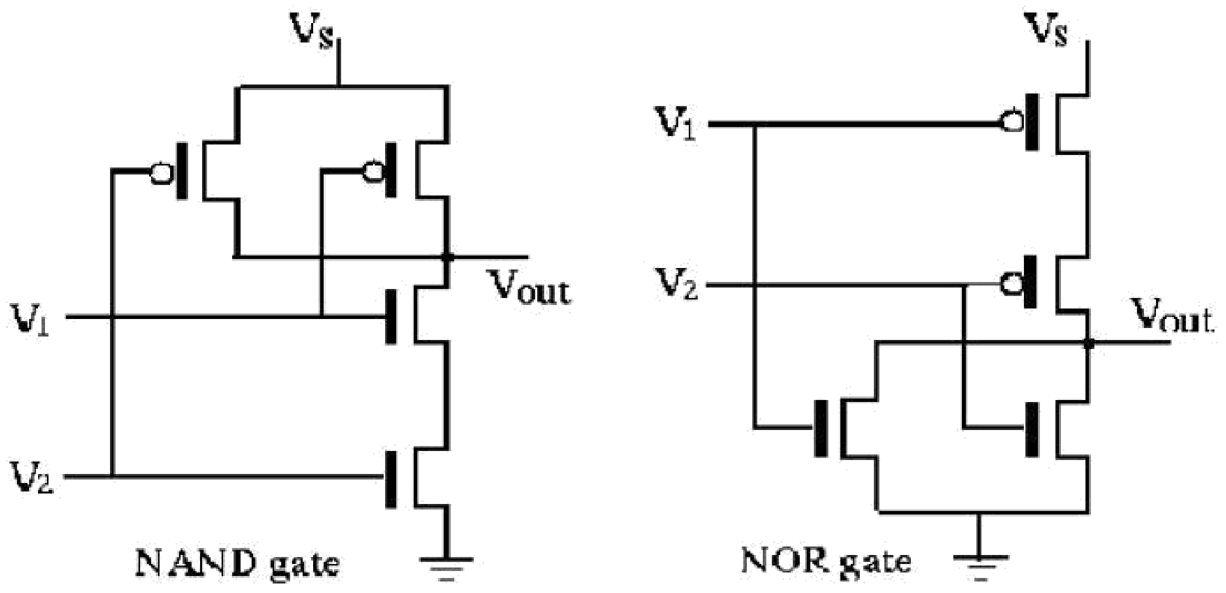


Figure. Schematic diagrams of nand and nor gate

We can see that the nand gate consists of two pmos in parallel which forms the pull up logic and two nmos in series forming the pull down logic. It is the complementary for the nor gate. We get inverted logic from CMOS structures. The series and parallel connections are for getting the right logic output. The pull up and the pull down devices must be placed to get high and low outputs when required.

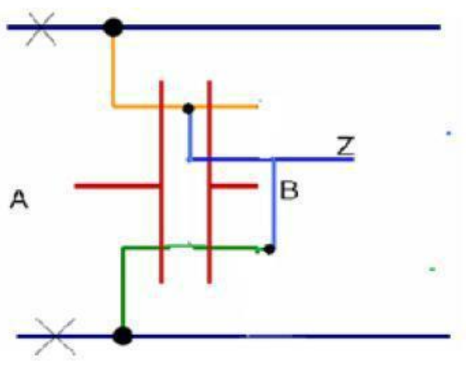


Figure. Stick diagrams of nand gate.

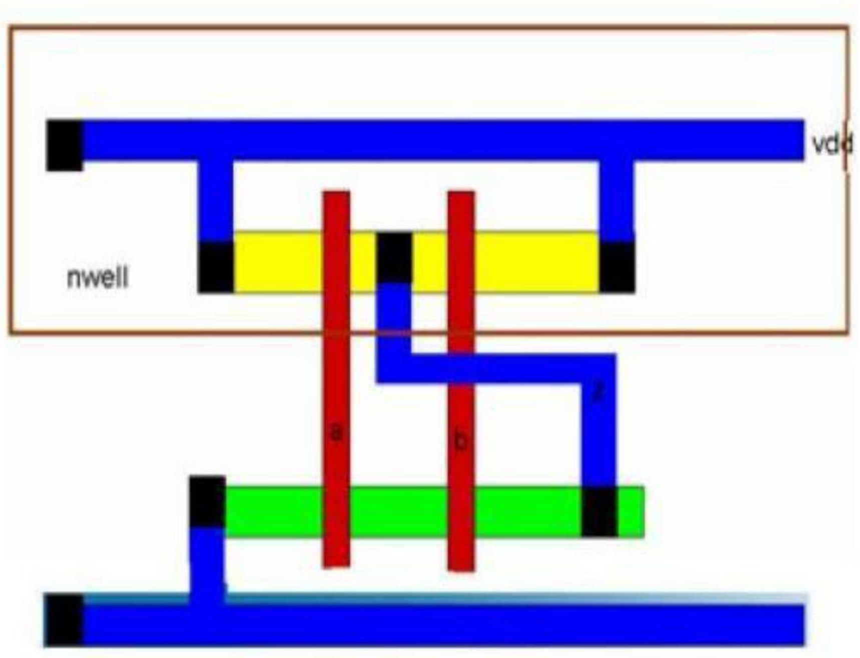


Figure. Layout of nand gate.

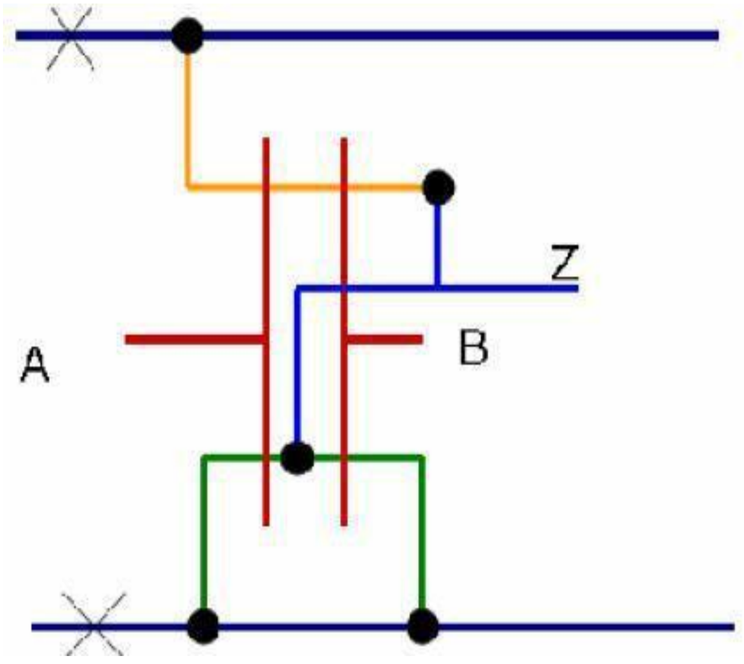


Figure. Stick diagram of NOR gate

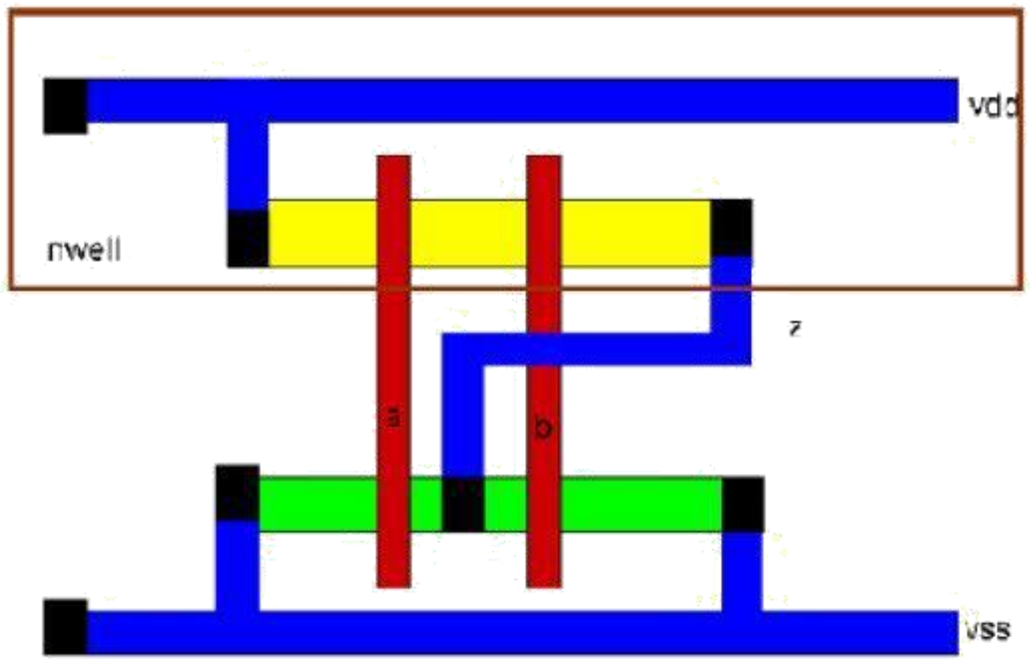


Figure. Layout of NOR gate.

Complementary Static CMOS Example :

F = ((A+B) C) + D

Static CMOS Example

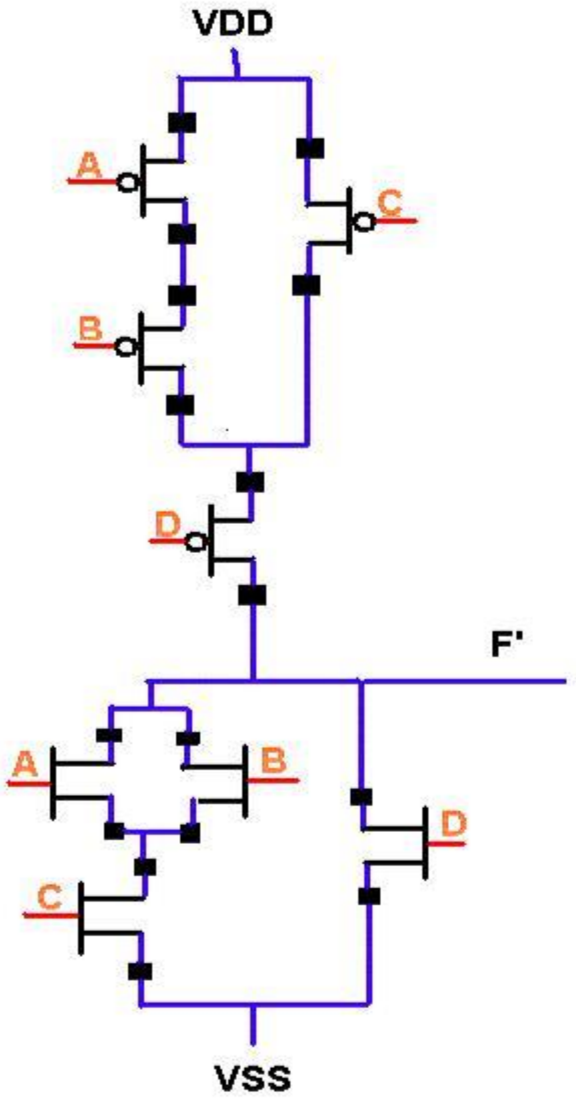
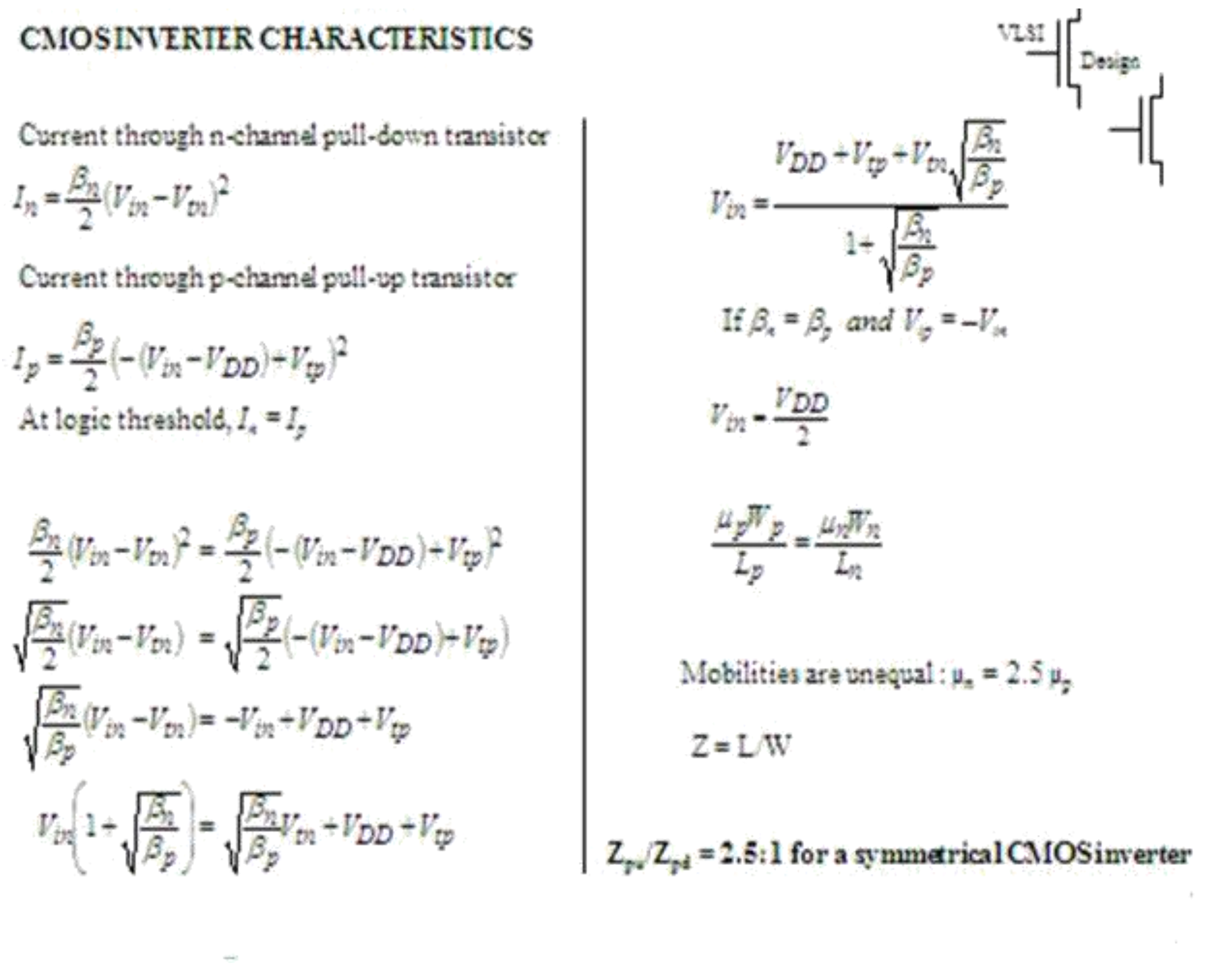


Figure. Design example



CMOS INVERTER CHARACTERISTICS

Current through n-channel pull-down transistor

$$I_n = \frac{\beta_n}{2}(V_{in} - V_{tn})^2$$

Current through p-channel pull-up transistor

$$I_p = \frac{\beta_p}{2}(-(V_{in} - V_{DD}) + V_{tp})^2$$

At logic threshold, $I_n = I_p$

$$\frac{\beta_n}{2}(V_{in} - V_{tn})^2 = \frac{\beta_p}{2}(-(V_{in} - V_{DD}) + V_{tp})^2$$

$$\sqrt{\frac{\beta_n}{2}}(V_{in} - V_{tn}) = \sqrt{\frac{\beta_p}{2}}(-(V_{in} - V_{DD}) + V_{tp})$$

$$\sqrt{\frac{\beta_n}{\beta_p}}(V_{in} - V_{tn}) = -V_{in} + V_{DD} + V_{tp}$$

$$V_{in}\left(1 + \sqrt{\frac{\beta_n}{\beta_p}}\right) = \sqrt{\frac{\beta_n}{\beta_p}}V_{tn} + V_{DD} + V_{tp}$$

$$V_{in} = \frac{V_{DD} + V_{tp} + V_{tn}\sqrt{\frac{\beta_n}{\beta_p}}}{1 + \sqrt{\frac{\beta_n}{\beta_p}}}$$

If $\beta_n = \beta_p$ and $V_{tp} = -V_{tn}$

$$V_{in} = \frac{V_{DD}}{2}$$

$$\frac{\mu_p W_p}{L_p} = \frac{\mu_n W_n}{L_n}$$

Mobilities are unequal : $\mu_n = 2.5\,\mu_p$

$Z = L/W$

$Z_{pu}/Z_{pd} = 2.5:1$ for a symmetrical CMOS inverter

CMOS COMPLEMENTARY LOGIC

CMOS logic structures of nand & nor has been studied in this unit. They were ratioed logic i.e. they have fixed ratio of sizes for the n and the p gates. It is possible to have ratio less logic by varying the ratio of sizes which is useful in gate arrays and sea of gates. Variable ratios allow us to vary the threshold and speed .If all the gates are of the same size the circuit is likely to function more correctly. Apart from this the supply voltage can be increased to get better noise immunity.

The increase in voltage must be done within a safety margin of the source -drain break down. Supply voltage can be decreased for reduced power dissipation and also meet the constraints of the supply voltage. Sometimes even power down with low power dissipation is required. For all these needs an on chip voltage regulator is required which may call for additional space requirement. A CMOS requires a n-block and a p-block for completion of the logic. That is for an n input logic, 2n gates are required. The variations to this circuit can include the following techniques reduction of noise margins and reducing the function determining transistors to one polarity.

PSEUDO NMOS LOGIC

This logic structure consists of the pull up circuit being replaced by a single pull up pmos whose gate is permanently grounded. This actually means that PMOS is all the time on and that now for a n input logic we have only n+1 gates. This technology is equivalent to the depletion mode type and preceded the CMOS technology and hence the name pseudo. The two sections of the device are now called as load and driver. The Gn/Gp (Gdriver / Gload) has to be selected such that sufficient gain is achieved to get consistent pull up and pull down levels. This involves having ratioed transistor sizes so that correct operation is obtained. However if minimum size drivers are being used then the gain of the load has to be reduced to get adequate noise margin.
There are certain drawbacks of the design which is highlighted next

- The gate capacitance of CMOS logic is two unit gates but for pseudo logic it is only one gate unit.
- Since number of transistors per input is reduced area is reduced drastically.

The disadvantage is that since the pMOS is always on, static power dissipation occurs whenever the nMOS is on. Hence the conclusion is that in order to use pseudo logic a tradeoff between size & load or power dissipation has to be made.

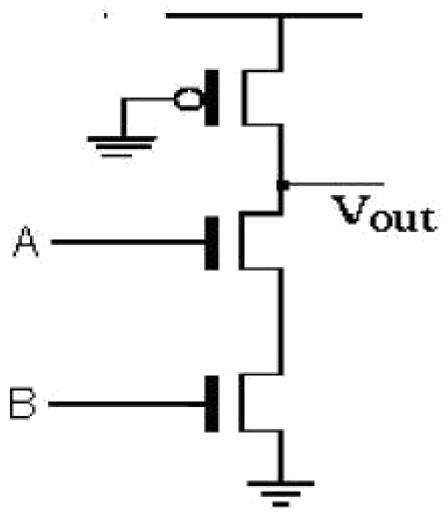


Figure. Pseudo NMOS

OTHER VARIATIONS OF PSEUDO NMOS

Multi drain logic

One way of implementing pseudo nmos is to use multi drain logic. It represents a merged transistor kind of implementation. The gates are combined in an open drain manner, which is useful in some automated circuits. Figure 4.

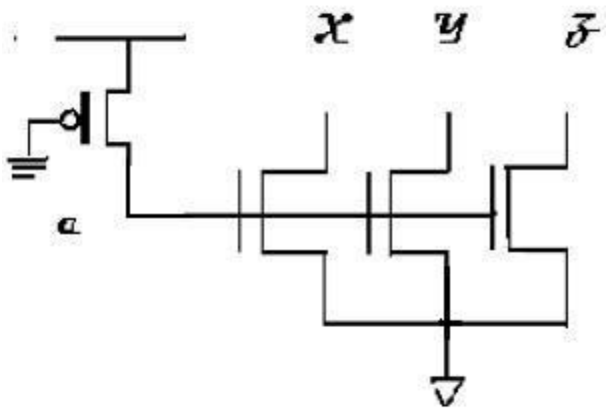


Figure: Multi drain logic

Pass transistors:

We have n and p pass transistors.

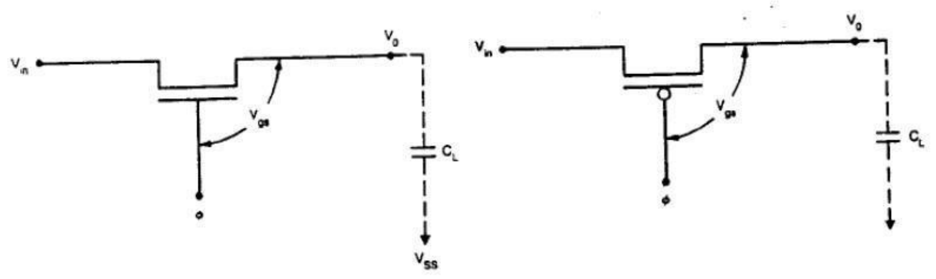
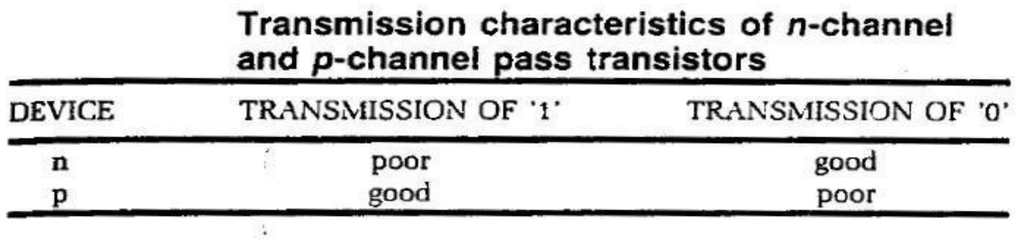


Figure. n and p pass transistors.

The disadvantage with the pass transistors is that, they will not be able to transfer the logic levels properly. The following table gives that explanation in detail.



Transmission characteristics of *n*-channel and *p*-channel pass transistors

DEVICE	TRANSMISSION OF '1'	TRANSMISSION OF '0'
n	poor	good
p	good	poor

If Vdd (5 volts) is to be transferred using nMOS the output will be (Vdd-Vtn).
POOR 1 or Weak Logic 1

If Gnd(0 volts) is to be transferred using nMOS the output will be Gnd.
GOOD 0 or Strong Logic 0

If Vdd (5 volts) is to be transferred using pMOS the output will be Vdd.
GOOD 1 or Strong Logic 1

If Gnd(0 volts) is to be transferred using pMOS the output will be Vtp.
POOR 0 or Weak Logic 0.

Transmission gates (TGs):

It‘s a parallel combination of pmos and nmos transistor with the gates connected to a complementary input. The disadvantages weak 0 and weak 1 can be overcome by using a TG instead of pass transistors.

Working of transmission gate can be explained better with the following equation. **When _="0"n and p device off, Vin=0 or 1, Vo="Z" When _="1" n and p device on, Vin=0 or 1, Vo=0 or 1 , where „Z" is high impedance.**

The resistance because two transistors will come in parallel and it is shown in the graph. The graph shows the resistance of n and p pass transistors, and resistance of TG which is lesser than the other two.

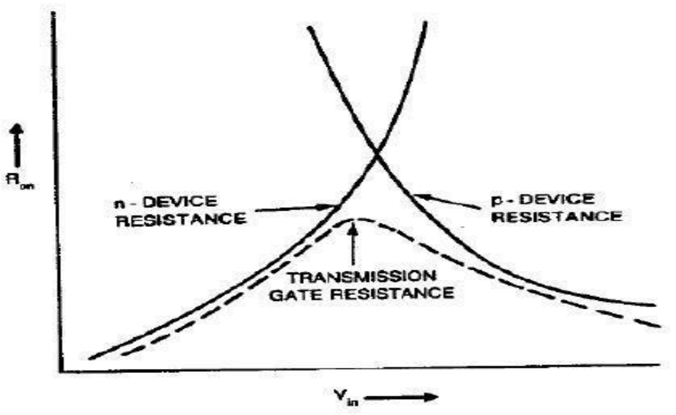


Figure. Graph of resistance vs. input for pass transistors and TG

DYNAMIC CMOS LOGIC:

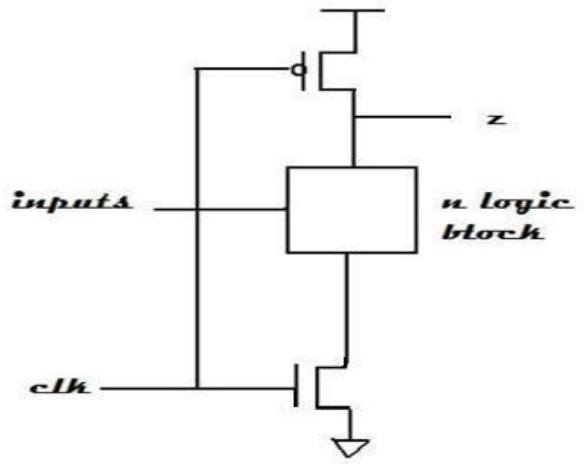


Figure. Dynamic CMOS logic

This logic looks into enhancing the speed of the pull up device by precharging the output node to vdd. Hence we need to split the working of the device into precharge and evaluate stage for which we need a clock. Hence it is called as dynamic logic. The output node is precharged to vdd by the pmos and is discharged conditionally through the nmos. Alternatively you can also have a p block and precharge the n transistor to vss. When the clock is low the precharge phase occurs. The path to Vss is closed by the nmos i.e. the ground switch. The pull up time is improved because of the active pmos which is already precharged. But the pull down time increases because of the ground switch.

There are a few problems associated with the design, like

b) Inputs have to change during the precharge stage and must be stable during the evaluate. If this condition cannot occur then charge redistribution corrupts the output node.

c) A simple single dynamic logic cannot be cascaded. During the evaluate phase the first gate will conditionally discharge but by the time the second gate evaluates, there is going to be a finite delay. By then the first gate may precharge

CMOS DOMINO LOGIC

The disadvantage associated with the dynamic CMOS is over come in this logic. In this we are able to cascade logic blocks with the help of a single clock. The precharge and the evaluate phases retained as they were. The change required is to add a buffer at the end of each stage. This logic works in the following manner. When the clk=0, ie during the precharge stage the output of the dynamic logic is high and the output of the buffer is low. Since the subsequent stages are fed from the buffer they are all off in the precharge stage. When the gate is evaluated in the next phase, the output conditionally goes low and the output of the buffer goes high. The subsequent gates make a transition from high to low.

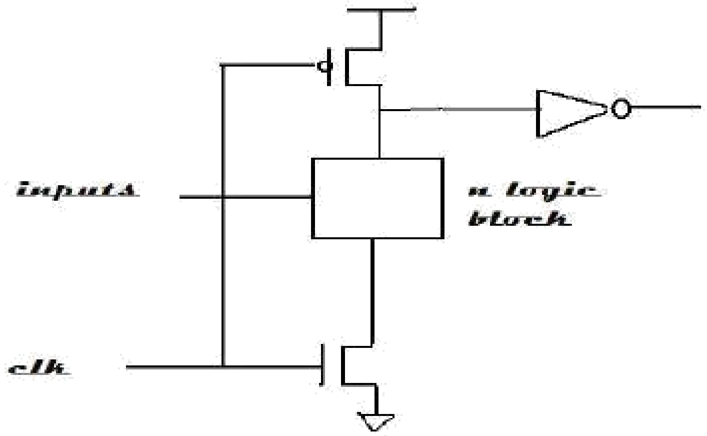


Figure. CMOS Domino logic.

Hence in one clock cycle the cascaded logic makes only one transition from 1 to 0 and buffer makes a transition from 0 to 1.In effect we can say that the cascaded logic falls like a line of dominos, and hence the name. The advantage is that any number of logic blocks can be cascaded provided the sequence can be evaluated in a single clock cycle. Single clock can be used to precharge and evaluate all the logic in a block. The limitation is that each stage must be buffered and only non- inverted structures are possible.

A further fine tuning to the domino logic can also be done. Cascaded logic can now consist of alternate p and n blocks and avoid the domino buffer. When clk=0,ie during the precharge stage, the first stage (with n logic) is precharged high and the second a p logic is precharged low and the third stage is high. Since the second stage is low, the n transistor is off. Hence domino connections can be made.

The advantages are we can use smaller gates, achieve higher speed and get a smooth operation. Care must be taken to ensure design is correct.

NP CMOS LOGIC (ZIPPER CMOS)

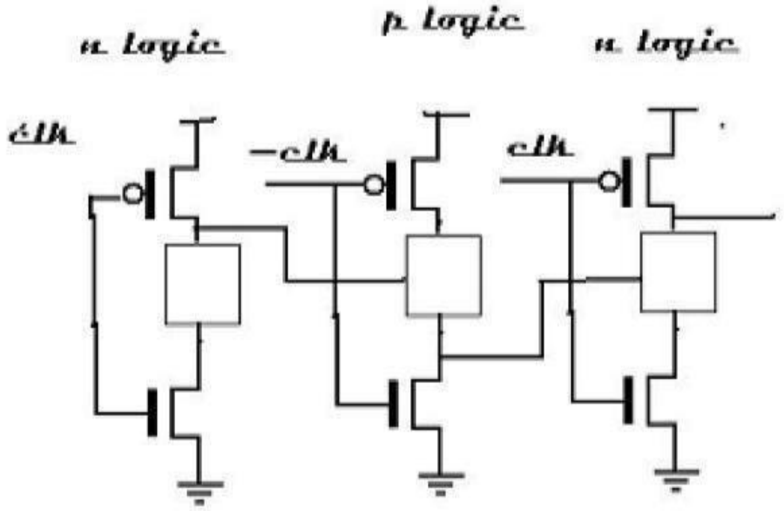
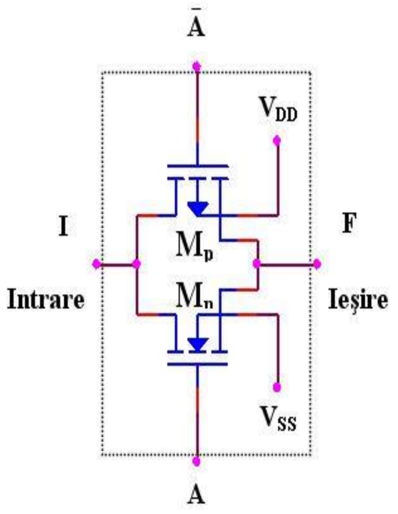
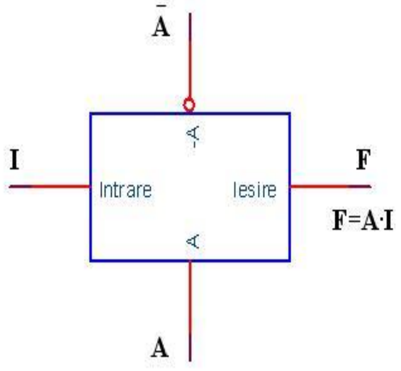


Figure. NP domino logic.

MULTIPLEXERS AND TRANSMISSION GATES

A **multiplexer** (or mux) is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A **multiplexer** of 2ninputs has n select lines, which are used to select which input line to send to the output. Another fundamental element in the CMOS construction is the transmission gate. It consists from a pair of complementary MOS transistor connected in parallel, shown below.





The circuit acts like a switch, the logic variable A being the control input. When the control input A is in logic “1” and Ā in logic “0” the transmission gate is open, and between the input and output appears a small resistance which lets the current flow in any direction.

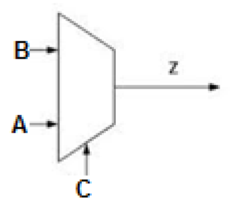
The value of the input voltage must be positive related to VSS and negative related to VDD. When A is in logic “0” and Ā is in logic “1”, the transmission gate is blocked, and there is a big resistance between the input and the output of the circuit. The truth table and logic symbol of Multiplexer is shown in figure.

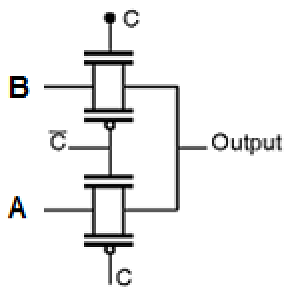
Truth Table

S	D1	D0	Y

2X1 MUX using Transmission Gates

2:1 MUX using transmission gate



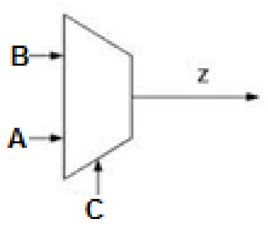


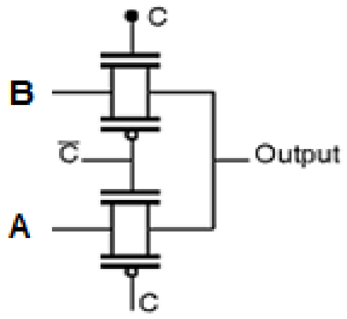
Select Signal (C)	Output (Z)
0	A
1	B

A,B- MUX Inputs,
C- Mux Select signal,
Output- Mux output

2X1 MUX using Transmission Gates

2:1 MUX using transmission gate

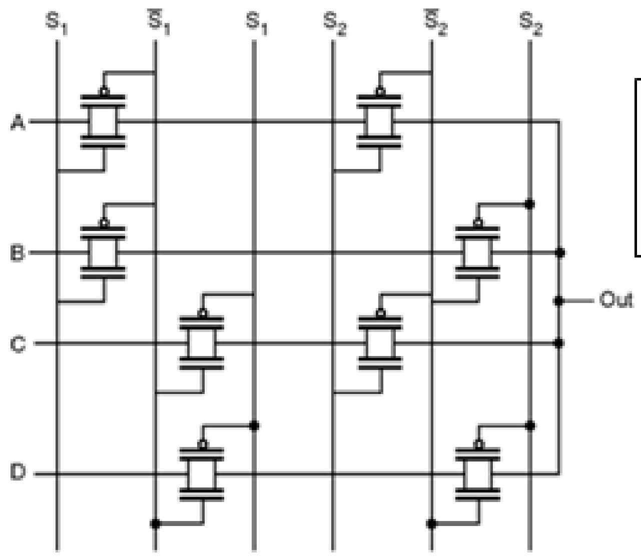




Select Signal (C)	Output (Z)
0	A
1	B

A,B- MUX Inputs,
C- Mux Select signal,
Output- Mux output

4X1 MUX using Transmission Gates



A,B,C,D- MUX Inputs,
S1,S2- Mux Select signal,
Out- Mux output

4 : 1 MUX using transmission gate

QUESTIONS FOR PRACTICE

1.Explain the working of CMOS inverter in different region and derive the expression for w/l ratio.

2.Implement , sizing and draw the stick diagram using static CMOS logic.

a.i. Two input nand gate
a.ii. Three input nor gate
a.iii. Two input exnor gate

3.Derive the expression of w/l ratio for depletion load nmos inverter.

4.Implement ,sizing and draw the stick diagram using depletion load logic a.iv. Two input nand gate
a.v. Three input nor gate
a.vi. Two input exnor

5..What is pass transistor logic its advantages and disadvantages.

6.Implement the full adder,halfadder,halfsubtractor and full subtractor using pass transistor logic.

7..Explain the working of dynamic CMOS design and implement the design y=(ab+cd)’

8.Explain the working of domino and npcmos logic.

9.With example implement the 2:1,4:1 mux and Demux using transmission gate.

UNIT 3 SEQUENTIAL LOGIC DESIGN

Introduction - Static sequential circuits- CMOS static flip-flop - Dynamic sequential circuits - Pseudo static latch- Dynamic two phase flip-flop - clocked CMOS logic - Pipelining - NORA CMOS logic -True single phase clocked logic - Realization of D-FF in TSPC logic.

Introduction

11. Unlike combinational logic circuits, the output of sequential logic circuits not only depends on current inputs but also on the past sequence of inputs.
12. Sequential circuits are constructed using combinational logic and a number of memory elements with some or all of the memory outputs fed back into the combinational logic forming a feedback path or loop.

A very simple sequential circuit with no inputs created using inverters to form a feedback loop:

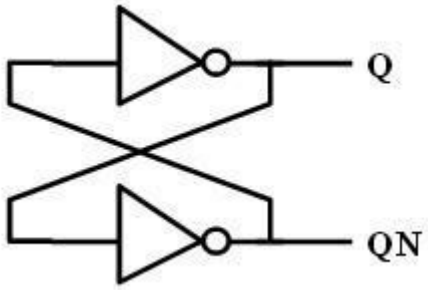


Figure 3.1 Basic Flip-Flop

When this circuit is powered up it randomly outputs Q = 0 or Q =1

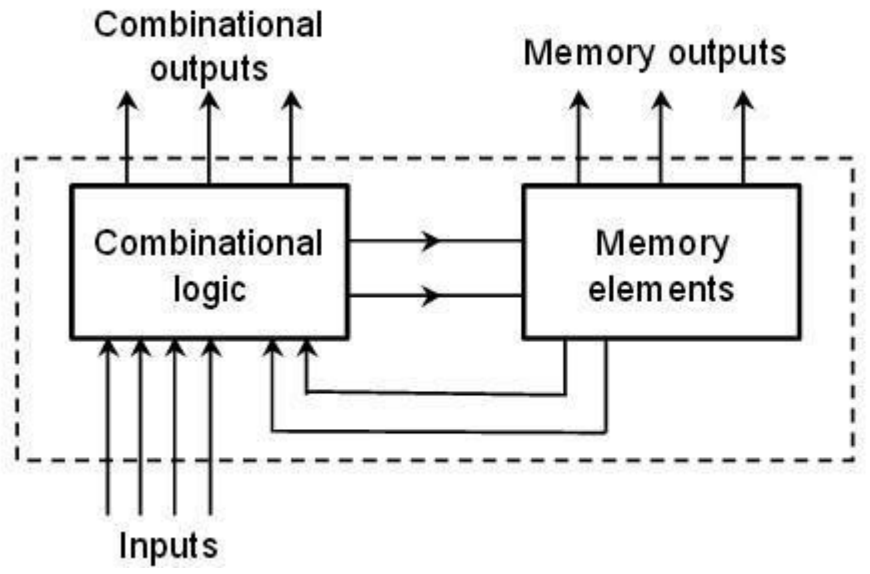


Figure 3.2. Block diagram of Sequential Logic

Sequential circuit = Combinational logic + Memory Elements
Current State of A sequential Circuit: Value stored in memory elements (value of state variables).
State transition: A change in the stored values in memory elements thus changing the sequential circuit from one state to another state.

Combinational circuits is one in which the output is a function of the current inputs. The sequential circuits is one in which the output depends on previous as well as current inputs; such circuits are said to have state. Finite state machines and pipelines are two important examples of sequential circuits. Sequential circuits are usually designed with flip-flops or latches, which are some-times called memory elements that hold data called tokens. The purpose of these elements is not really memory; instead, it is to enforce sequence, to distinguish the current token from the previous or next token. Therefore, we will call them sequencing elements .Without sequencing elements, the next token might catch up with the previous token, garbling both. Sequencing elements delay tokens that arrive too early, preventing them from catching up with previous tokens. Unfortunately, they inevitably add some delay to tokens that are already critical, decreasing the performance of the system. This extra delay is called sequencing overhead. Static circuits refer to gates that have no clock input, such as complementary CMOS, pseudo-nMOS or pass transistor logic.

Dynamic circuits refer to gates that have a clock input, especially domino logic. To complicate terminology, sequencing elements themselves can be either static or dynamic. A sequencing element with static storage employs some sort of feedback to retain its output value indefinitely. An element with dynamic storage generally maintains its value as charge on a capacitor that will leak away if not refreshed for a long period of time. The choices of static or dynamic for gates and for sequencing elements can be independent.

<u>Static sequential circuits</u>

Static memories use positive feedback to create a bistable circuit — a circuit having two stable states that represent 0 and 1. The basic idea is shown in Figure 3.3a, which shows two inverters connected in cascade along with a voltage-transfer characteristic typical of such a circuit. Also plotted are the VTCs of the first inverter, that is, Vo1 versus Vi1 , and the second inverter (Vo2 versus Vo1). The latter plot is rotated to accentuate that Vi2 = Vo1 . Assume now that the output of the second inverter Vo2 is connected to the input of the first Vi1 , as shown by the dotted lines in Figure 3.3a. The resulting circuit has only three possible operation points (A, B, and C), as demonstrated on the combined VTC. The following important conjecture is easily proven to be valid:

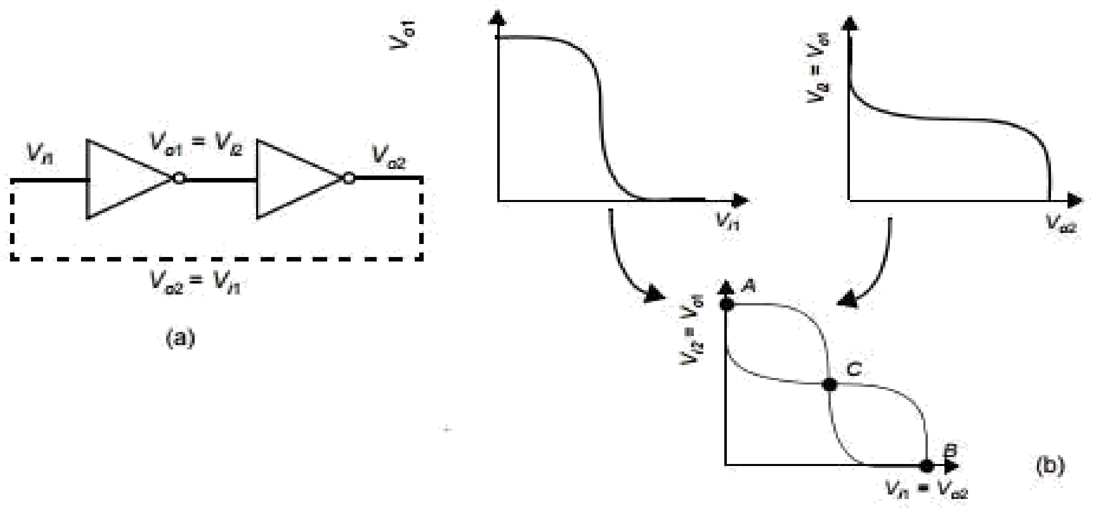


Figure 3.3 VTC of Flip-Flop

Under the condition that the gain of the inverter in the transient region is larger than 1, only A and B are stable operation points, and C is a metastable operation point. Suppose that the cross-coupled inverter pair is biased at point C. A small deviation from this bias point, possibly caused by noise, is amplified and regenerated around the circuit loop. This is a consequence of the gain around the loop being larger than 1. The effect is demonstrated in Figure 3.3a. The bias point moves away from C until one of the operation points A or B is reached. In conclusion, C is an unstable operation point. Every deviation (even the smallest one) causes the operation point to run away from its original bias. The chance is indeed very small that the cross-coupled inverter pair is biased at C and stays there. Operation points with this property are termed as meta-stable.

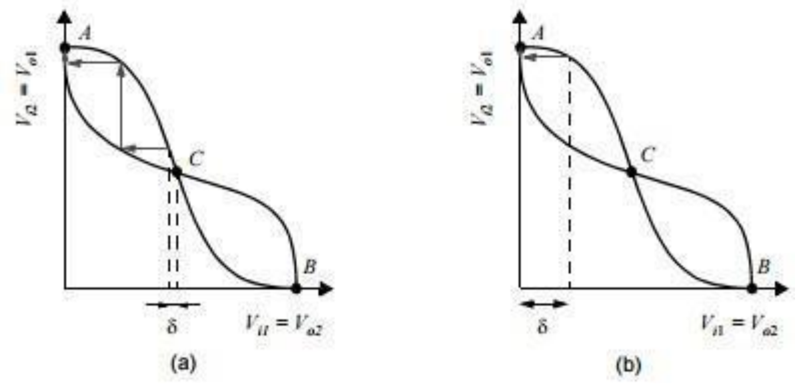


Figure 3.4. Combined VTC

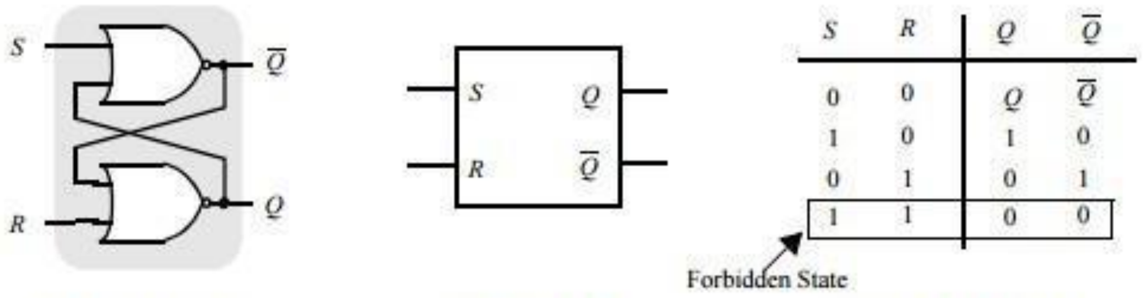
On the other hand, A and B are stable operation points, as demonstrated in Figure 3.4b. In these points, the loop gain is much smaller than unity. Even a rather large deviation from the operation point is reduced in size and disappears. Hence the cross-coupling of two inverters results in a bistable circuit, that is, a circuit with two stable states, each corresponding to a logic state. The circuit serves as a memory, storing either a 1 or a 0 (corresponding to positions A and B).. A trigger pulse must be applied to change the state of the circuit. Another common name for a bistable circuit is flip-flop (unfortunately, an edge-triggered register is also referred to as a flip-flop).

CMOS static flip-flop :

SR-Flip flop

The cross-coupled inverter pair shown in the previous section provides an approach to store a binary variable in a stable way. However, extra circuitry must be added to enable control of the memory states. The simplest incarnation accomplishing this is the well know SR —or set-reset— flip-flop, an implementation of which is shown in Figure 3.5a. This circuit is similar to the cross-coupled inverter pair with NOR gates replacing the inverters. The second input of the NOR gates is connected to the trigger inputs (S and R), that make it possible to force the outputs Q and Q to a given state. These outputs are complimentary (except for the SR = 11 state).

When both S and R are 0, the flip-flop is in a quiescent state and both outputs retain their value (a NOR gate with one of its input being 0 looks like an inverter, and the structure looks like a cross coupled inverter). If a positive (or 1) pulse is applied to the S input, the Q output is forced into the 1 state (with Q going to 0). Vice versa, a 1 pulse on R resets the flip-flop and the Q output goes to 0. These results are summarized in the characteristic table of the flip-flop, shown in Figure c. The characteristic table is the truth table of the gate and lists the output states as functions of all possible input conditions. When both S and R are high, both Q and Q' are forced to zero. Since this does not correspond with our constraint that Q and Q must be complementary, this input mode is considered to be forbidden. An additional problem with this condition is that when the input triggers return to their zero levels, the resulting state of the latch is unpredictable and depends on whatever input is last to go low. Finally, Figure 3.5 shows the schematics symbol of the SR flip-flop.



S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
1	1	0	0

(a) Schematic diagram (b) Logic symbol (c) Characteristic table

Figure 3.5. SR Flip-flop

The SR flip-flops discussed so far are asynchronous, and do not require a clock signal. Most systems operate in a synchronous fashion with transition events referenced to a clock. One possible realization of a clocked SR flip-flop— a level-sensitive positive latch— is shown in Figure 3.6. It consists of a cross-coupled inverter pair, plus 4 extra transistors to drive the flip-flop from one state to another and to provide clocked operation. Observe that the number of transistors is identical to the implementation, but the circuit has the added feature of being clocked. The drawback of saving some transistors over a fully-complimentary CMOS implementation is that transistor sizing becomes critical in ensuring proper functionality. Consider the case where Q is high and an R pulse is applied. The combination of transistors M4, M7, and M8 forms a ratioed inverter. In order to make the latch switch, we must succeed in bringing Q below the switching threshold of the inverter M1 -M2. Once this is achieved, the positive feedback causes the flip-flop to invert states. This requirement forces us to increase the sizes of transistors M5 , M6 , M7 , and M8 .

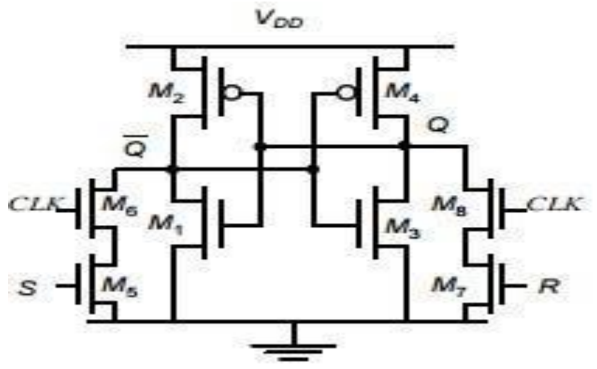


Figure 3.6 CMOS clocked SR FLIPFLOP

Dynamic sequential circuits

Storage in a static sequential circuit relies on the concept that a cross-coupled inverter pair produces a bistable element and can be used to memorize binary values. This approach has the useful property that a stored value remains valid as long as the supply voltage is applied to the circuit, hence the name static. The major disadvantage of the static gate, however, is its complexity. When registers are used in computational structures that are constantly clocked such as pipelined datapath, the requirement that the memory should hold state for extended periods of time can be significantly relaxed. This results in a class of circuits based on temporary storage of charge on parasitic capacitors. The principle is exactly identical to the one used in dynamic logic — charge stored on a capacitor can be used to represent a logic signal. The absence of charge denotes a 0, while its presence stands for a stored 1. No capacitor is ideal, unfortunately, and some charge leakage is always present. A stored value can hence only be kept for a limited amount of time, typically in the range of milliseconds. If one wants to preserve signal integrity, a periodic refresh of its value is necessary. Hence the name dynamic storage. Reading the value of the stored signal from a capacitor without disrupting the charge requires the availability of a device with a high input impedance.

There are three types : 1. Pseudostatic latch 2. Dynamic two-phase flipflo[p 3. C ^{2}MOS register.

The Pseudostatic latch

It is possible to reduce the clock load to two transistors by using implement multiplexers using NMOS only pass transistor as shown in Figure 3.7. The advantage of this approach is the reduced clock load of only two NMOS devices. When CLK is high, the latch samples the D input, while a low clock-signal enables the feedback-loop, and puts the latch in the hold mode. While attractive for its simplicity, the use of NMOS only pass transistors result in the passing of a degraded high voltage of VDD-Vtn to the input of the first inverter. This impacts both noise margin and the switching performance, especially in the case of low values of VDD and high values of Vtn. It also causes static power dissipation in first inverter. Since the maximum input-voltage to the inverter equals VDD-Vtn, the PMOS device of the inverter is never turned off, resulting is a static current flow.

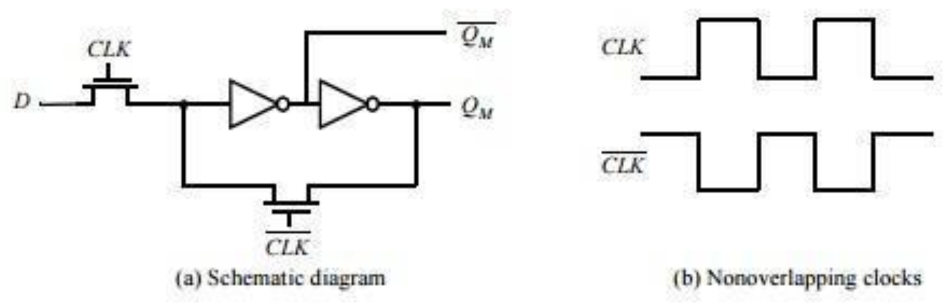


Figure 3.7 Pseudo-static latch

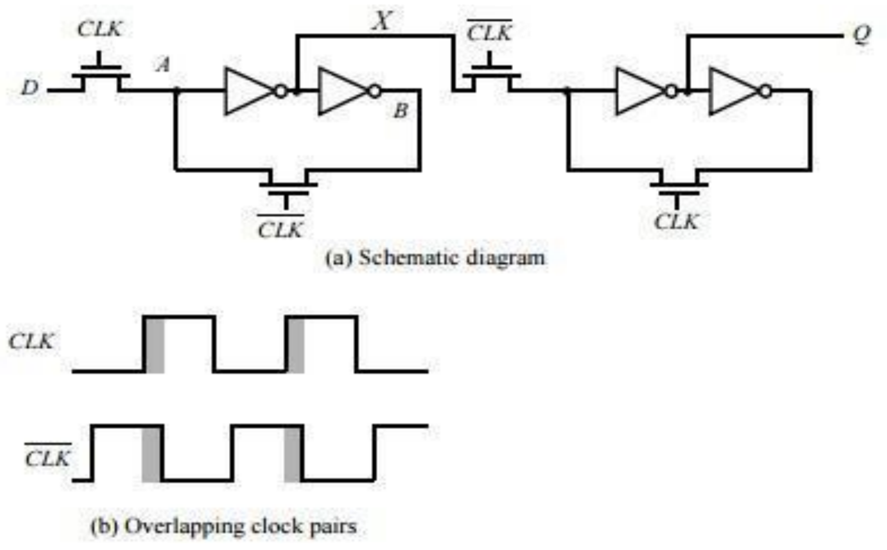


Figure 3.8 Master-slave register based NMOS pass transistor

So far, we have assumed that CLK is a perfect inversion of CLK, or in other words, that the delay of the generating inverter is zero. Even if this were possible, this would still not be a good assumption. Variations can exist in the wires used to route the two clock signals, or the load capacitances can vary based on data stored in the connecting latches. This effect, known as clock skew is a major problem, and causes the two clock signals to overlap as is shown in Figure 3.8b. Clock-overlap can cause two types of failures, as illustrated for the NMOS-only negative master-slave register of Figure 3.8a.

When the clock goes high, the slave stage should stop sampling the master stage output and go into a hold mode. However, since CLK and CLK are both high for a short period of time (the overlap period), both sampling pass transistors conduct and there is a direct path from the D input to the Q output. As a result, data at the output can change on the rising edge of the clock, which is undesired for a negative edge triggered register. The is known as a race condition in which the value of the output Q is a function of whether the input D arrives at node X before or after the falling edge of CLK. If node X is sampled in the meta-stable state, the output will switch to a value determined by noise in the system.

The primary advantage of the multiplexer-based register is that the feedback loop is open during the sampling period, and therefore sizing of devices is not critical to functionality. However, if there is clock overlap between CLK and CLK, node A can be driven by both D and B, resulting in an undefined state. Those problems can be avoided by using two non-overlapping clocks PHI1 and PHI2 instead (Figure 3.9), and by keeping the nonoverlap time tnon_overlap between the clocks large enough such that no overlap occurs even in the presence of clock-routing delays. During the non-overlap time, the FF is in the high-impedance state— the feedback loop is open, the loop gain is zero, and the input is disconnected. Leakage will destroy the state if this condition holds for too long a time. Hence the name pseudostatic: the register employs a combination of static and dynamic storage approaches depending upon the state of the clock.

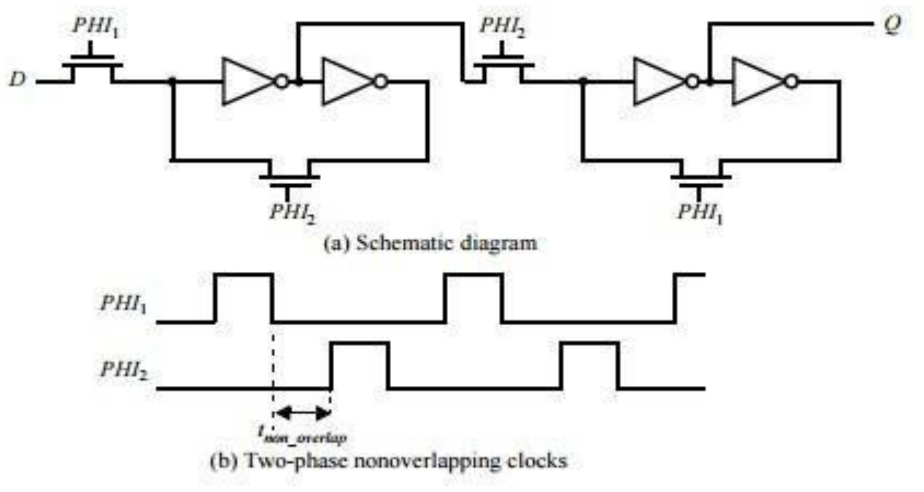


Figure 3.9 Pseudostatic two-phase D register

Dynamic two-Phase Flip-Flop

Dynamic Transmission-Gate Edge-triggered Registers

A fully dynamic positive edge-triggered register based on the master-slave concept is shown in Figure 3.10. When CLK = 0, the input data is sampled on storage node 1, which has an equivalent capacitance of C1 consisting of the gate capacitance of I1 , the junction capacitance of T1 , and the overlap gate capacitance of T1 . During this period, the slave stage is in a hold mode, with node 2 in a high-impedance (floating) state. On the rising edge of clock, the transmission gate T2 turns on, and the value sampled on node 1 right before the rising edge propagates to the output Q (note that node 1 is stable during the high phase of the clock since the first transmission gate is turned off). Node 2 now stores the inverted version of node 1. This implementation of an edge-triggered register is very efficient as it requires only 8 transistors. The sampling switches can be implemented using NMOS-only pass transistors, resulting in an even-simpler 6 transistor implementation. The reduced transistor count is attractive for high-performance and low-power systems.

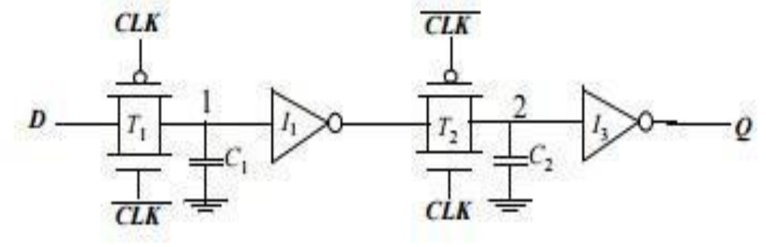


Figure 3.10 Dynamic edge triggered register

The set-up time of this circuit is simply the delay of the transmission gate, and corresponds to the time it takes node 1 to sample the D input. The hold time is approximately zero, since the transmission gate is turned off on the clock edge and further inputs changes are ignored. The propagation delay (t c-q) is equal to two inverter delays plus the delay of the transmission gate T2 . One important consideration for such a dynamic register is that the storage nodes (i.e., the state) has to be refreshed at periodic intervals to prevent a loss due to charge leakage, due to diode leakage as well as sub-threshold currents. In datapath circuits, the refresh rate is not an issue since the registers are periodically clocked, and the storage nodes are constantly updated. Clock overlap is an important concern for this register. Consider the clock waveforms shown in Figure 3.11. During the 0-0 overlap period, the NMOS of T1 and the PMOS of T2 are simultaneously on, creating a direct path for data to flow from the D input of the register to the Q output. This is known as a race condition. The output Q can change on the falling edge if the overlap period is large — obviously an undesirable effect for a positive edge-triggered register. The same is true for the 1-1 overlap region, where an input-output path exists through the PMOS of T1 and the NMOS of T2 . The latter case is taken care off by enforcing a hold time constraint. That is, the data must be stable during the high-high overlap period. The former situation (0-0 overlap) can be addressed by making sure that there is enough delay between the D input and node 2 ensuring that new data sampled by the master stage does not propagate through to the slave stage. Generally the built in single inverter delay should be sufficient and the overlap period constraint is given as:

$$t_{overlap0-0} < t_{T1} + t_{I1} + t_{T2}$$

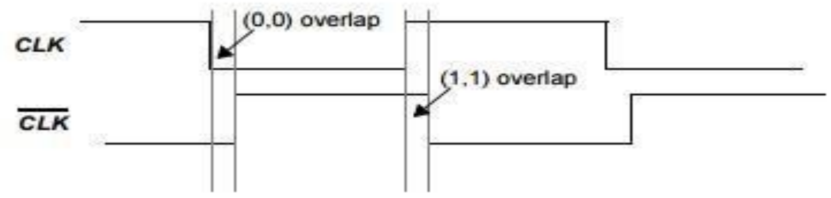


Figure 3.11 Impact of non-overlapping clock

Clocked CMOS logic : C^{2}MOS logic

Figure 3.12 shows an ingenious positive edge-triggered register based on the master-slave concept which is insensitive to clock overlap. This circuit is called the C2MOS (Clocked CMOS) register [Suzuki73]. The register operates in two phases.

 CLK = 0 (CLK = 1): The first tri-state driver is turned on, and the master stage acts as an inverter sampling the inverted version of D on the internal node X. The master stage is in the evaluation mode. Meanwhile, the slave section is in a high-impedance mode, or in a hold mode. Both transistors M7 and M8 are off, decoupling the output from the input. The output Q retains its previous value stored on the output capacitor CL2.

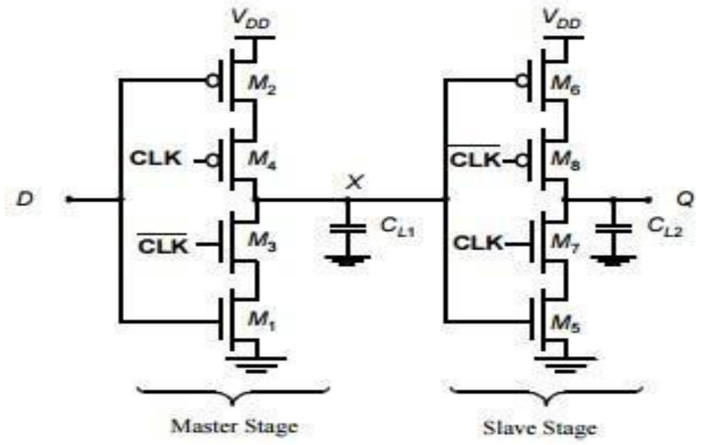


Figure 3.12 C^{2}MOS master-slave positive edge triggered

 The roles are reversed when CLK = 1: The master stage section is in hold mode (M3 - M4 off), while the second section evaluates (M7 -M8 on). The value stored on CL1 propagates to the output node through the slave stage which acts as an inverter.

The overall circuit operates as a positive edge-triggered master-slave register — very similar to the transmission-gate based register presented earlier. However, there is an important difference: A C^{2}MOS register with CLK-CLK clocking is insensitive to overlap, as long as the rise and fall times of the clock edges are sufficiently small.

To prove the above statement, we examine both the (0-0) and (1-1) overlap cases (Figure 3.12). In the (0-0) overlap case, the circuit simplifies to the network shown in Figure 3.13a in which both PMOS devices are on during this period. The (1-1) overlap case, where both NMOS devices M3 and M7 are turned on, is somewhat more contentious.

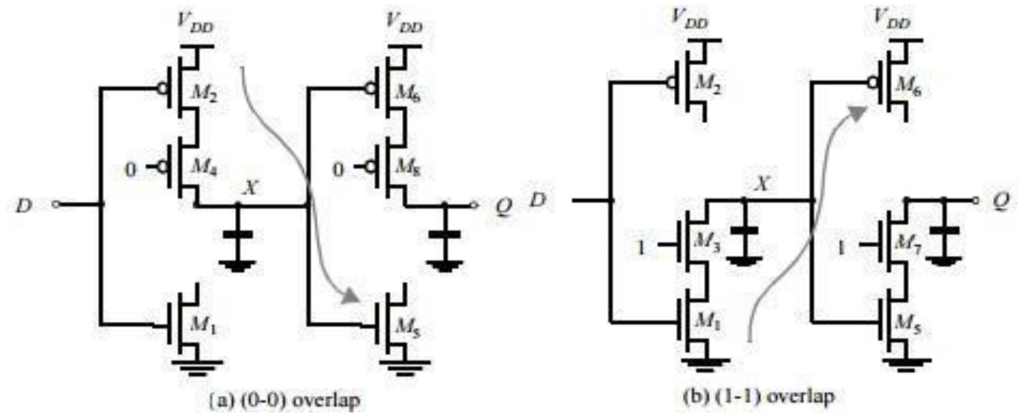


Figure 3.13 C^{2}MOS during overlap period

Pipelining

Pipelining is a popular design technique often used to accelerate the operation of the data paths in digital processors. The idea is easily explained with the example of Figure 3.14. The goal of the presented circuit is to compute log (|a - b|), where both a and b represent streams of numbers, that is, the computation must be performed on a large set of input values. The minimal clock period Tmin necessary to ensure correct evaluation is given as:

$$T_{min} = t_{c\text{-}q} + t_{pd,\text{logic}} + t_{su}$$

where t c-q and tsu are the propagation delay and the set-up time of the register, respectively. We assume that the registers are edge-triggered D registers. The term of delay tpd,logic stands for the worst-case delay path through the combinatorial network, which consists of the adder, absolute value, and logarithm functions. In conventional systems, the latter delay is generally much larger than the delays associated with the registers and dominates the circuit performance. Assume that each logic module has an equal propagation delay. We note that each logic module is then active for only 1/3 of the clock period (if the delay of the register is ignored). For example, the adder unit is active during the first third of the period and remains idle— this is, it does no useful computation— during the other 2/3 of the period. Pipelining is a technique to improve the resource utilization, and increase the functional throughput. Assume that we introduce registers between the logic blocks, as shown in Figure 3.15. This causes the computation for one set of input data to spread over a number of clock periods, as shown in Table

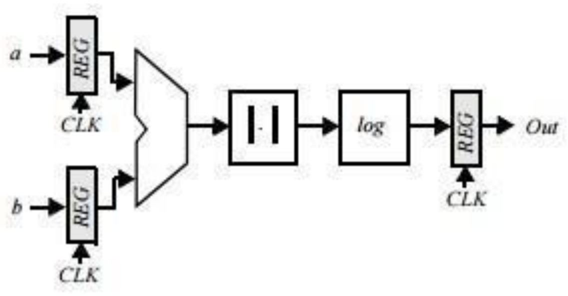


Figure 3.14 Non pipelined version

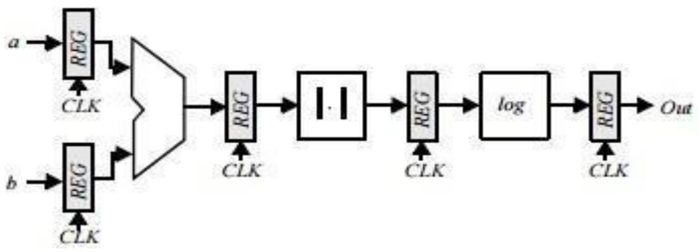
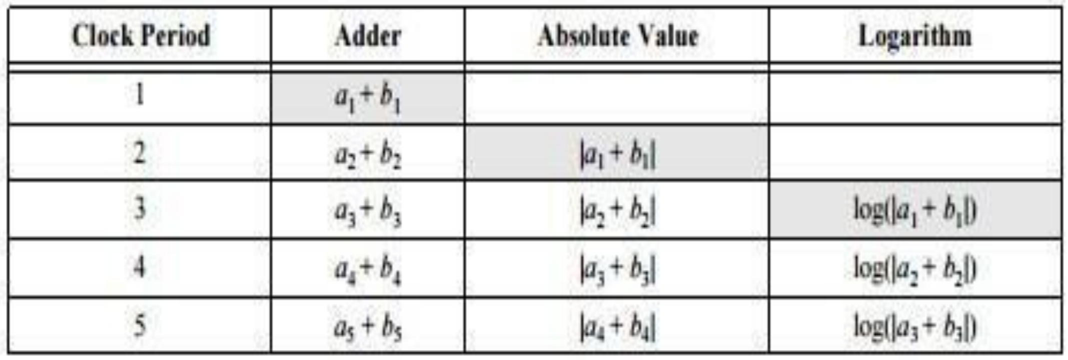


Figure 3.15 Pipelined version

The result for the data set (a1, b1) only appears at the output after three clock-periods. At that time, the circuit has already performed parts of the computations for the next data sets, (a2,b2) and (a3,b3). The computation is performed in an assembly-line fashion, hence the name pipeline.



Clock Period	Adder	Absolute Value	Logarithm
1	$a_1 + b_1$		
2	$a_2 + b_2$	$\lvert a_1 + b_1\rvert$	
3	$a_3 + b_3$	$\lvert a_2 + b_2\rvert$	$\log(\lvert a_1 + b_1\rvert)$
4	$a_4 + b_4$	$\lvert a_3 + b_3\rvert$	$\log(\lvert a_2 + b_2\rvert)$
5	$a_5 + b_5$	$\lvert a_4 + b_4\rvert$	$\log(\lvert a_3 + b_3\rvert)$

The advantage of pipelined operation becomes apparent when examining the minimum clock period of the modified circuit. The combinational circuit block has been partitioned into three sections, each of which has a smaller propagation delay than the original function. This effectively reduces the value of the minimum allowable clock period:

$$T_{min,\text{pipe}} = t_{c\text{-}q} + \max(t_{pd,\text{add}}, t_{pd,\text{abs}}, t_{pd,\text{log}})$$

Suppose that all logic blocks have approximately the same propagation delay, and that the register overhead is small with respect to the logic delays. The pipelined network output performs the original circuit by a factor of three under these assumptions, or then Tmin,pipe= Tmin / 3. The increased performance comes at the relatively small cost of two additional registers, and an increased latency. This explains why pipelining is popular in the implementation of very high-performance datapaths.

NORA CMOS logic

NORA CMOS combines C^2MOS pipeline registers and NORA dynamic logic function blocks. Each module consists of a block of combinational logic that can be a mixture of static and dynamic logic, followed by a C^2MOS latch. Logic and latch are clocked in such a way that both are simultaneously in either evaluation, or hold (precharge) mode. A block that is in evaluation during CLK = 1 is called a CLK-module, while the inverse is

called a $\overline{\text{CLK}}$ -module. Examples of both classes are shown in Figure 3.16 a and b, respectively. The operation modes of the modules are summarized in Table 3.1.

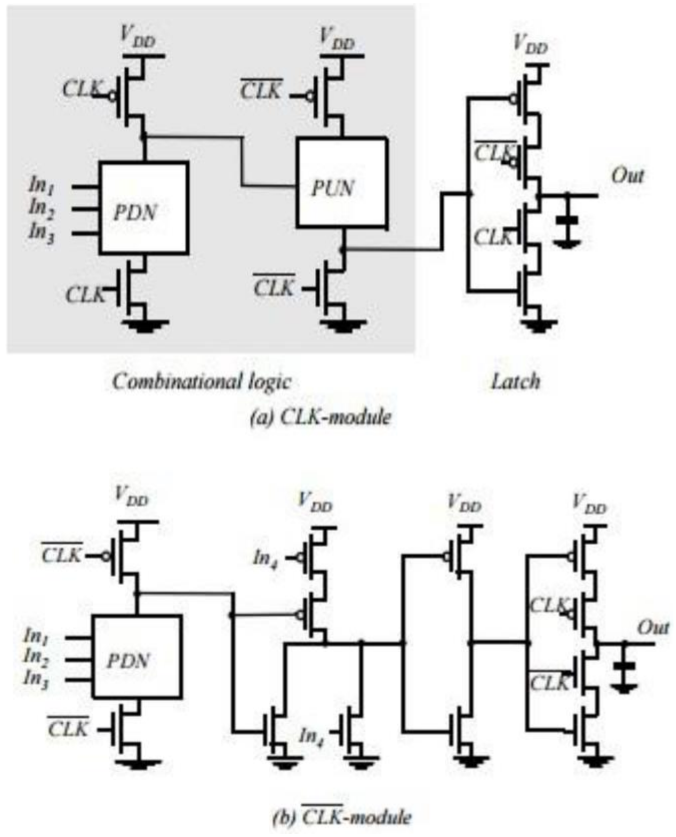
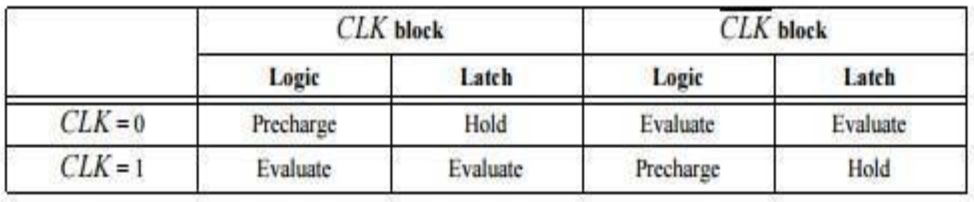


Figure 3.16 Example of NORA-CMOS module

Table 3.1 Operation mode of NORA CMOS logic module



	CLK block		$\overline{CLK}$ block	
	Logic	Latch	Logic	Latch
$CLK = 0$	Precharge	Hold	Evaluate	Evaluate
$CLK = 1$	Evaluate	Evaluate	Precharge	Hold

A NORA datapath consists of a chain of alternating CLK and CLK modules. While one class of modules is precharging with its output latch in hold mode, preserving the previous output value, the other class is evaluating. Data is passed in a pipelined fashion from module to module. NORA offers designers a wide range of design choices. Dynamic and static logic can be mixed freely, and both CLKp and CLKn dynamic blocks can be used in cascaded or in pipelined form. With this freedom of design, extra inverter stages, as required in DOMINO-CMOS, are most often avoided.

True single phase clocked logic

In the two-phase clocking schemes described above, care must be taken in routing the two clock signals to ensure that overlap is minimized. While the C^2MOS provides a skew-tolerant solution, it is possible to design registers that only use a single phase clock. The True Single-Phase Clocked Register (TSPCR) proposed by **Yuan and Svensson** uses a **single clock** (without an inverse clock)

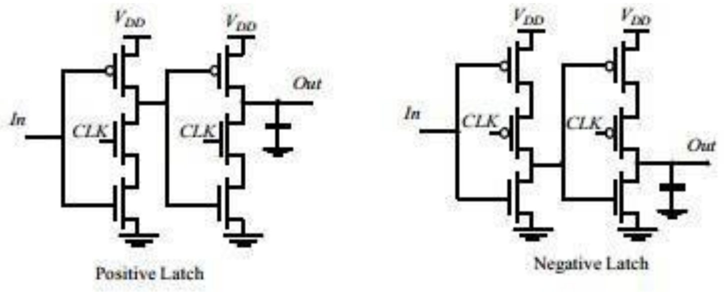


Figure 3.17 Doubled C^2MOS latches

The basic single -phase positive and negative latches are shown in Figure 3.17. For the positive latch, when *CLK* is high, the latch is in the *transparent mode* and corresponds to two cascaded inverters; the latch is non-inverting, and propagates the input to the output. On the other hand, when *CLK* = 0, both inverters are disabled, and the latch is in *hold-mode*. Only the pull-up networks are still active, while the pull-down circuits are deactivated. As a result of the dual-stage approach, no signal can ever propagate from the input of the latch to the output in this mode. A register can be constructed by cascading positive and negative latches. The clock load is similar to a conventional transmission gate register, or C^2MOS register. The main advantage is the use of a single clock phase. The disadvantage is the slight increase in the number of transistors— 1 2 transistors are required. TSPC offers an additional advantage: the possibility of embedding logic functionality into the latches.

This reduces the delay overhead associated with the latches. Figure 3.17 outlines the basic approach for embedding logic, while Figure 3.17 shows an example of a positive latch that implements the AND of*In*1 and *In*2 in addition to performing the latching function. While the *set-up time* of this latch has increased over the one shown in Figure 3.17, the overall performance of the digital circuit (that is, the clock period of a sequential circuit)has improved: the increase in *set-up time* is typically smaller than the delay of an AND gate. This approach of embedding logic into latches has been used extensively in the design of the EV4 DEC Alpha microprocessor and many other high performance processors.

The TSPC latch circuits can be further reduced in complexity as illustrated in Figure 3.17, where only the first inverter is controlled by the clock. Besides the reduced number of transistors, these circuits have the advantage that the clock load is reduced by half. On the other hand, not all node voltages in the latch experience the full logic swing. For instance, the voltage at node *A* (for *Vin* = 0 V) for the positive latch maximally equals *VDD –V Tn*, which results in a reduced drive for the output NMOS transistor and a loss in performance. Similarly, the voltage on node *A* (for *Vin* = *VDD*) for the negative latch is only driven down to |*VTp*|. This also limits the amount of *VDD* scaling possible on the latch. Figure 3.17 shows the design of a specialized *single-phase edge-triggered register*. When *CLK* = 0, the input inverter is sampling the inverted *D* input on node *X*. The second (dynamic) inverter is in the precharge mode, with *M*6 charging up node *Y* to *VDD*. The third inverter is in the *hold* mode, since *M*8 and *M*9 are *off*. Therefore, during the low phase of the clock, the input to the final(static) inverter is holding its previous value and the output *Q* is stable. On the rising edge of the clock, the dynamic inverter*M*4-*M*6 evaluates. If *X* is high on the rising edge, node *Y* discharges. The third inverter*M*7-*M*8 is on during the high phase, and the node value on *Y* is passed to the output *Q*. On the positive phase of the clock, note that node *X* transitions to a low if the *D* input transitions to a high level. Therefore, the input must be kept stable till the value on node *X* before the rising edge of the clock propagates to *Y*. This represents the *hold time* of the register (note that the *hold time* less than 1 inverter delay since it takes 1 delay for the input to affect node *X*). The *propagation delay* of the register is essentially three inverters since the value on node *X* must propagate to the output *Q*. Finally, the *set-up time* is the time for node *X* to be valid, which is one inverter delay.

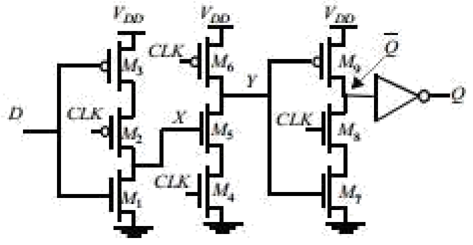


Figure 3.18 Adding logic to TSPC approach

Realization of D-FF in TSPC logic

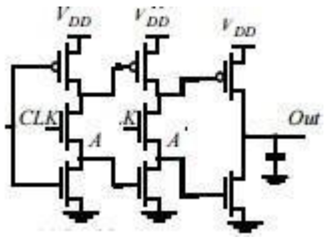


Figure 3.19 Positive edge-triggered D-Flip-Flop using split-output latches

The basic TSPC latches can be combined in many different ways to implement all essential sequential components. For instant, a number of implementations of an edge-triggered D FF using TSPC are shown in the figure 3.19 uses five transistor split-output latch. The resulting gate achieves almost the same speed as the first two, but reduces the clock load substantially(two clock connection instead of four).The later benefit turns out to be of major important, especially for circuits employing many registers such as large shift-registers.

QUESTION FOR PRACTICE

1. Explain the working of C^2MOS.
2. Discuss the advantage of True single phase clocked logic and explain with an example
3. Give the working principle of NORA CMOS and list its advantages
4. Explain the pipelining architecture of CMOS LOGIC.
5. List the advantage and working of dynamic two phase flip-flop.
6. What are Static sequential circuits?
7. List the characteristics of SR Flip-flop and implement using CMOS Logic
8. Write about Pseudo static latch.
9. Realize D-Flip-Flop using True single phase clocked logic
10. Compare static sequential and dynamic sequential circuits. List its advantage with an example.

UNIT 4 SUBSYSTEM DESIGN

Introduction-Designing Static and Dynamic Adder circuits - The Array Multiplier - Multiplier structures-Baugh Wooly - Booth Multiplier - Barrel shifter - Memory structures - SRAM and DRAM design - Design approach of Programmable logic devices - PLA, PAL and FPGA.

THE ADDER

The adder is one of the most critical components of a processor, as it is used in the Arithmetic Logic Unit (ALU), in the floating-point unit and for address generation in case of cache or memory access (John Rabaey (2003)).Increasing demand for mobile electronic devices such as cellular phones and laptop computers requires the use of power efficient VLSI circuits.

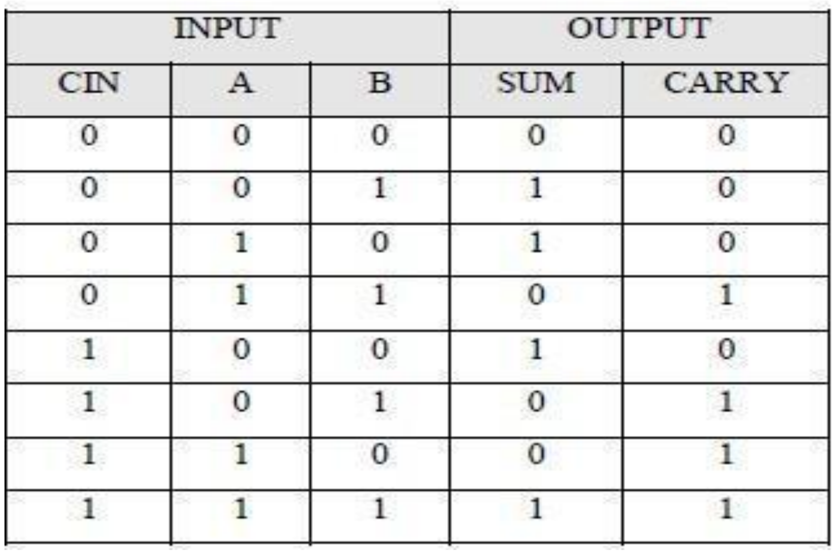
THE BINARY ADDER

The full adder operation can be stated as follows: Given the three 1- bit inputs A, B, and Cin, it is desired to calculate the two 1-bit outputs Sum and Carry, where

Sum = (A xor B) xor Cin

Carry = A and B + Cin (A xor B)

Table.1.Truth Table of Full Adder



INPUT			OUTPUT	
CIN	A	B	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Conventional Full Adder :

Probably the simplest approach to designing an adder is to implement gates to yield the required majority logic functions.

SUM = A.B.Cin + A.B.Cin + A.B.Cin + A.B.Cin

SUM = Cin (A.B+A.B) + Cin (A.B+A.B)

13. A (XOR) B (XOR)

Cin CARRY = A.B + A. Cin + B.

Cin CARRY = A.B + Cin (A+B)

The gate schematic for the direct implementation of equation is shown in figure.

This implementation uses a 3-input XOR gate. A transistor level implementation is shown in figure. This uses a total of 32 transistors.

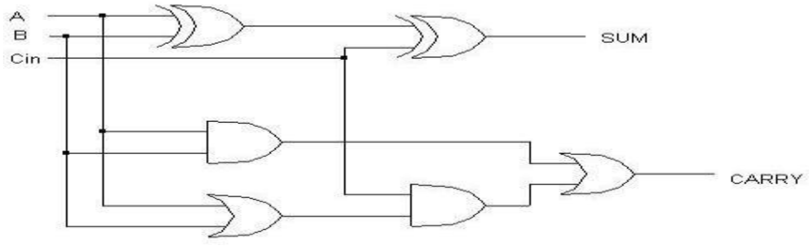


Figure.4.1 Gate schematic for conventional full adder

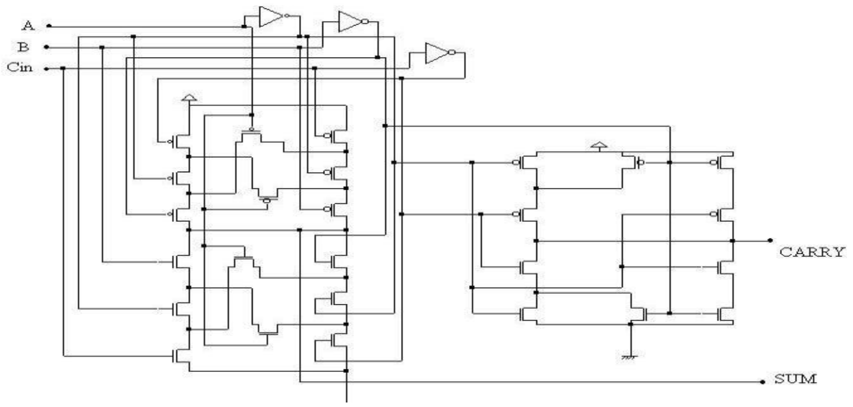


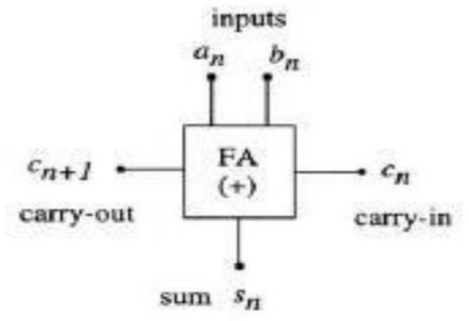
Figure 4.2 Transistor level schematic for conventional full adder

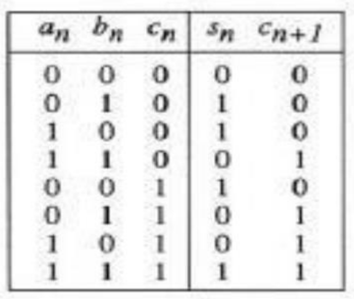
Adder Circuits:

Binary adders provide the basic connection between Boolean operations and arithmetic. They are commonly used for comparing different technologies or design styles since they are of reasonable importance and complexity. Figure shows the basic symbol and function table for a full-adder circuit that uses the inputs A and B and Cin a carry-in bit to produce the sum bit S and the carry-out bit Cn+1 The most common SOP expressions obtained directly from the table entries are given by

$$s_n = a_n \oplus b_n \oplus c_n$$

$$c_{n+1} = a_n b_n + c_n (a_n \oplus b_n)$$





a_n	b_n	c_n	s_n	c_{n+1}
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

Figure 5.45 Full adder operation

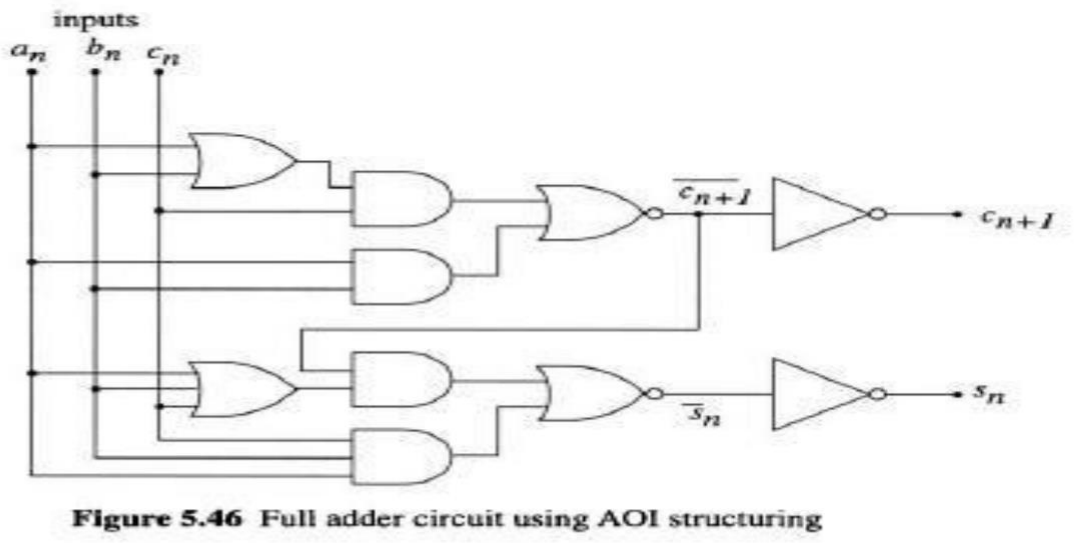


Figure 5.46 Full adder circuit using AOI structuring

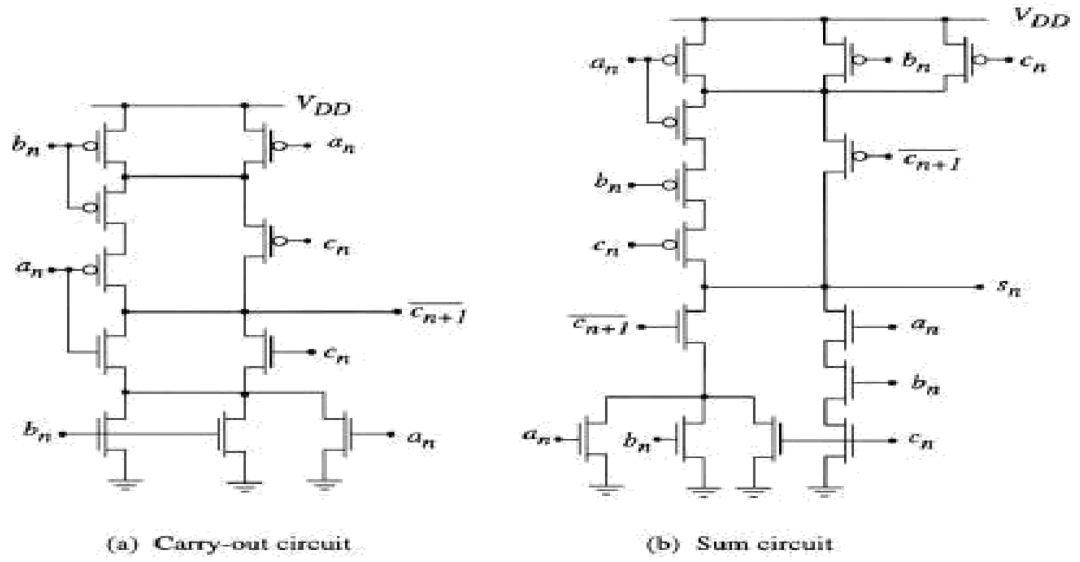


Figure 5.47 CMOS circuits using AOI structuring

A n Array Multiplier

Logical effort is very effective for comparing large structures to select one of several quite different overall organizations. Rather than completing a detailed design of each alternative, we can use the method of logical effort to estimate the performance of a design sketch. In this extended example, we explore several designs for a multiplier. We make no claim that we find “the best multiplier design” for any situation; we offer it only to illustrate the application of logical effort.

The example illustrates many of the techniques of logical effort. The reader is assumed to be familiar with logical effort applied to static gates, asymmetric gates, forks, and branches. An alternative design using domino logic is explored briefly. A multiplier is an interesting design example because it affords a rich set of design choices. We can build a multiplier that uses a large array of adders or an alternative that cycles a smaller array of adders several times to complete the product. We will use logical effort to seek a design with minimum delay for a given array configuration. Our exploration of the design proceeds in several steps:

 Define the context of the multiplier array, its inputs and outputs, and its basic structure. This step identifies key critical paths and structural elements, such as an *adder cell*.

 Evaluate different designs for the adder cell and its interconnection into an array. Here logical effort is vital for quick evaluation of a set of combinatorial alternatives. We will estimate the least delay through the cell without doing a detailed design or picking transistor- size

 Evaluate the design of the parts of the multiplier other than the adder cell to estimate overall performance. This step uncovers a property of the design that suggests altering the overall structure slightly.

 Complete the design of the adder cell by determining transistor sizes and other implementation details. As the design proceeds, we will face many design decisions, often exchanging space and speed. Not all of the reasonable choices are worked out in detail.

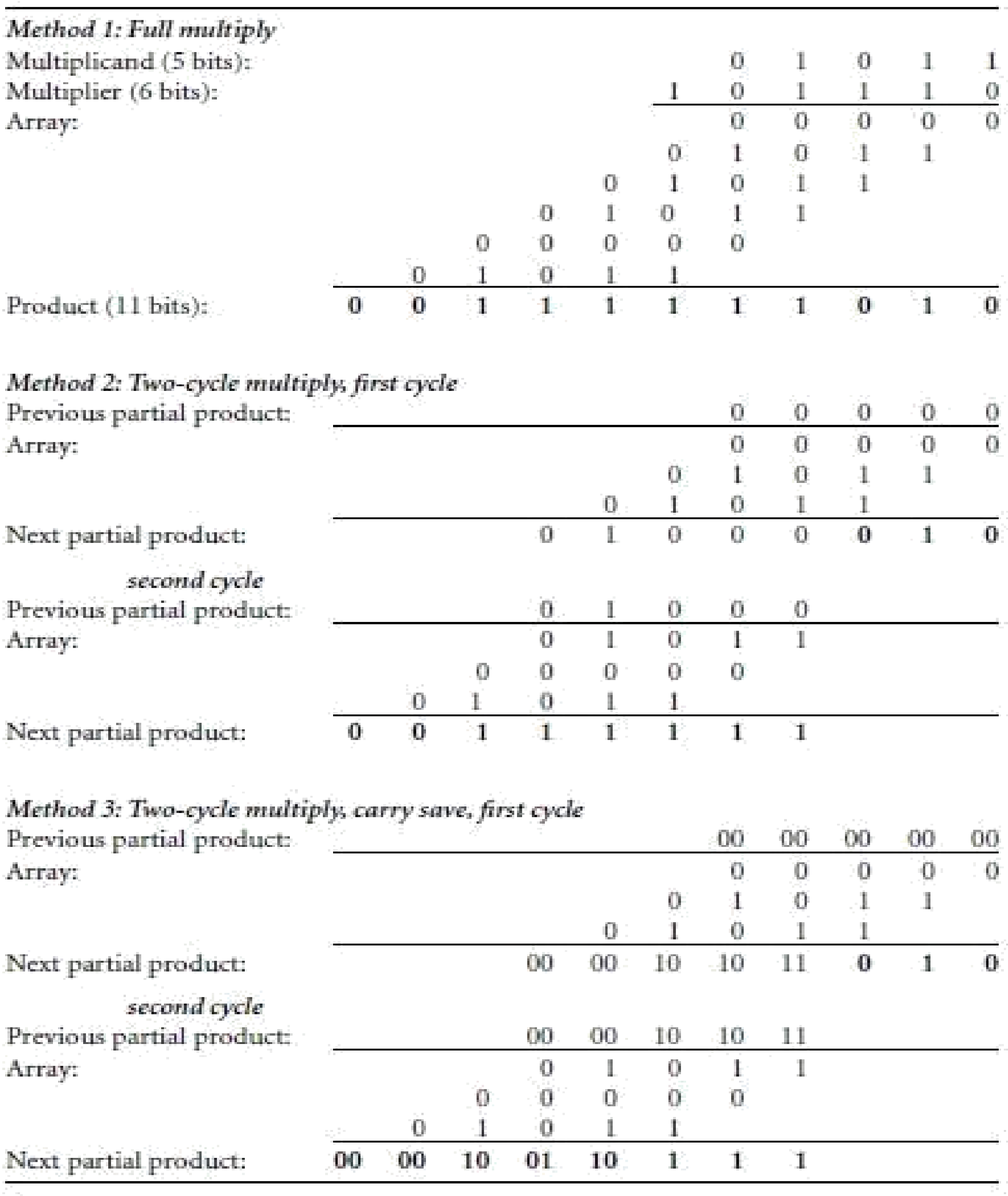
Multiplier Structure

There are many different structures that can be used to implement a multiplier. Method 1 in Table 1 shows diagrammatically the simple method for multiplication we were all taught in school, modified so that all the numbers use binary notation. In the illustration, a 5-bit multiplicand is multiplied by a 6-bit multiplier to obtain an 11-bit product. Each row of the array is the product of the multiplicand and a single bit of the multiplier. Because multiplier bits are either 0 or 1, each row of the array is either 0 or a copy of the multiplicand. The array is laid out so that each row is shifted to the left to account for the increasing bi- nary significance of the multiplier bits. We sum the rows of the array to obtain the product. You may wish to verify that the result is correct: in base ten, the multiplicand is 11, the multiplier 46, and the product 506.

This simple method can be implemented directly in hardware by using five separate 2-input, 5-bit adders to add the six rows of the array shown in the table. But this design suffers in two ways. First, it is large because there are a lot of adders. And second, the delay will be substantial because bits must propagate through all five adders and because each adder has a carry path to resolve a 5-bit carry.

A small multiplier can be constructed by iteratively adding values to a partial product. The partial product is held in a register initialized to 0. During each cycle, the multiplicand is added to the partial product if the low-order bit of the multiplier is 1, the multiplier is shifted one bit to the right, and the multiplicand is shifted one bit to the left. Because this method uses only a single adder, it is much smaller than the full adder array. In our example, this method would require six cycles to complete because we require a 6-bit multiplier. Instead of either of these extremes, we will study a design that lies in between. It will use two cycles to form the product, and each cycle will form the partial product governed by three bits of the multiplier. Method 2 in Table 1 shows how this technique is related to the full array: the first cycle computes the same answer as the top three rows of the full array, and the second cycle computes the same answer as the bottom three rows of the full array.

Table 1—Three methods for multiplying a 5-bit multiplicand by a 6-bit multiplier.

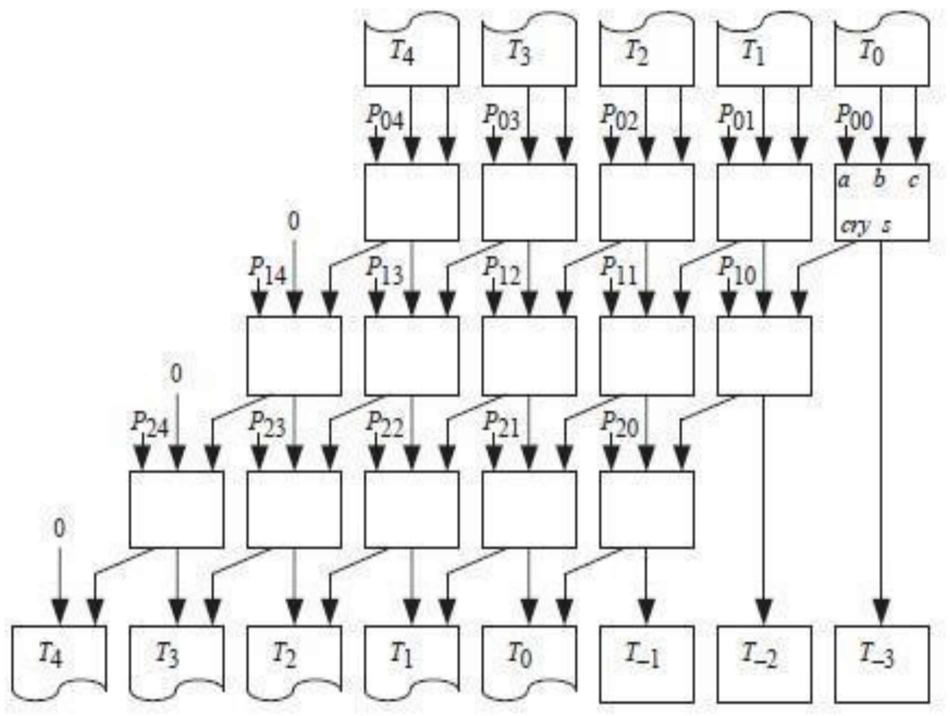


Method 1: Full multiply											
Multiplicand (5 bits):							0	1	0	1	1
Multiplier (6 bits):						1	0	1	1	1	0
Array:							0	0	0	0	0
						0	1	0	1	1	
					0	1	0	1	1		
				0	1	0	1	1			
			0	0	0	0	0				
		0	1	0	1	1					
Product (11 bits):	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**0**
Method 2: Two-cycle multiply, first cycle											
Previous partial product:							0	0	0	0	0
Array:							0	0	0	0	0
						0	1	0	1	1	
					0	1	0	1	1		
Next partial product:				0	1	0	0	0	**0**	**1**	**0**
second cycle											
Previous partial product:				0	1	0	0	0			
Array:				0	1	0	1	1			
			0	0	0	0	0				
		0	1	0	1	1					
Next partial product:	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**			
Method 3: Two-cycle multiply, carry save, first cycle											
Previous partial product:							00	00	00	00	00
Array:							0	0	0	0	0
						0	1	0	1	1	
					0	1	0	1	1		
Next partial product:				00	00	10	10	11	**0**	**1**	**0**
second cycle											
Previous partial product:				00	00	10	10	11			
Array:				0	1	0	1	1			
			0	0	0	0	0				
		0	1	0	1	1					
Next partial product:	**00**	**00**	**10**	**01**	**10**	**1**	**1**	**1**			

The remaining five bits become the “previous partial product” for the second cycle. This two-cycle method may still suffer from long carry paths in the adders, especially if the number of bits in the multiplicand (the length of the carry path) exceeds the number of rows in the multiplier array. By using *carry-save form*, we can speed up the multiplier array by deferring carry resolution until after the two cycles have finished. The essence of carry-save form is to represent a partial
product by remembering two bits for each

binary position. To obtain a conventional bi- nary number from its carry-save form, we must sum the two bits corresponding to each binary position and propagate any carries that are generated. Method 3 in Table 1 shows the double-bit partial product representation. You may wish to verify that the “next partial products” in carry-save form have exactly the same values as their counterparts in Method 2.

There is a price for this structure, however: after the multiplier array is finished, five bits of the final product remain in carry-save form. To obtain a carry-resolved result, the carry-save result must be fed to a 2-input, 5-bit full adder, which has a 5-bit carry path. Although carry lookahead techniques can be used for speed, the time required for this carry resolution step may be critical to the overall design. But the virtue of the structure shown in Method 3 is that the carry resolution is done only once, not once per cycle of the multiplier array.



Fig~~ur~~e 4.7.The multiplier array of adder cells.

A partial product enters at the top, the Pij values are added, and a new partial product emerges at the bottom, in carry-save form. In an actual layout, successive rows would probably be shifted to the right so that the adder cells would form a perfect rectangular array.But how does carry-save form avoid carry paths in the array? The answer is illustrated in Figure 1, laid out to resemble the format shown in Table 1, Method 3. The previous partial product enters at the top, in carry-save form, from register cells *Tj*

.

That is, each bit of the partial product is represented by two signals, which must be added to determine the binary value. Both signals from *Tj* have binary weight 2^j in the result. The next partial product is produced at the bottom of the figure. Note that

three bits of the result (*T*−1, *T*−2, and *T*−3) are produced in carry-resolved rather than carry-save form and become a part of the final product. The remaining five bits of partial product are routed to the inputs of the registers *Tj* , *j* = 0 . . . 4 to become the partial product used as input in the next iteration of the multiplier. To simplify the diagram, the drawing of each register *Tj* is split: its outputs appear at the top of the
figure and its inputs appear at the bottom.

The array consists of three rows of five adder cells each. Each row is responsible for adding a shifted form of the multiplicand to the partial product and passing the partial product to the row below. Each adder cell contains a 1-bit full adder with three inputs of equal binary value, labeled *a*, *b*, and *c*. Each cell has two outputs representing the sum (*s*) and carry (*cry*) that result from adding together the three inputs. The sum output represents the same binary weight as each of the three inputs. The carry output represents twice the binary weight of the sum output. As you can see from the figure, each cell combines a sum and a carry input from cells earlier in the array with a product bit, labeled *Pij*. Note that the longest path through the array is three adder cells, corresponding to the number of rows in the array.

The product bits *Pij* are generated by combining multiplicand and multiplier bits using an

and gate, $Pij = Qi \wedge Rj$, where *Qi* are bits of the multiplier and *Rj* are bits of the multiplicand, again using the notation that bit *j* has weight 2^j . The effect is that a row of
cells adds 0 to the partial product if the corresponding bit of the multiplier is 0 and adds the multiplicand if it is 1. The structure of the array causes the multiplicand to shift to the left, corresponding to the weight of the multiplier bit. The multiplier *Q* and multiplicand *R* are both stored in registers operated at the same time as the partial-product register *T*. Of course, at the end of the first cycle, the multiplier must be shifted
three bits to the right so that in the second cycle the high-order three bits of the multiplier are used to control the array.

Baugh-Wooley Multiplier

2.1 Multiplying two 2's compliment numbers

The Baugh-Wooley multiplication algorithm is an efficient way to handle the sign bits. This technique has been developed in order to design regular multipliers, suited for 2's-complement numbers. Let us consider two n-bit numbers, A and B, to be multiplied. A and B can be represented as

$$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i \qquad (1)$$

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

Where the a_i's and b_i's are the bits in A and B, respectively, and a_{n-1} and b_{n-1} are the sign bits.

The product, $P = A \times B$, is then given by the following equation:

$$\begin{aligned} P &= A \times B \\ &= \left(-a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i\right) \times \left(-b_{n-1}2^{n-1} + \sum_{j=0}^{n-2} b_j 2^j\right) \\ &= a_{n-1}b_{n-1}2^{2n-2} + \sum_{i=0}^{n-2}\sum_{j=0}^{n-2} a_i b_j 2^{i+j} \\ &\quad -2^{n-1}\sum_{i=0}^{n-2} a_i b_{n-1} 2^i - 2^{n-1}\sum_{j=0}^{n-2} a_{n-1} b_j 2^j \qquad (3) \end{aligned}$$

Equation (3) indicates that the final product is obtained by subtracting the last two *positive terms* from the first two terms.

2.2 Baugh-Wooley multiplication algorithm

Rather than do a subtraction operation, we can obtain the 2's complement of the last two term and add all terms to get the final product.

The last two terms are $n-1$ bits each that extend in binary weight from position 2^{n-1} up to 2^{2n-3}. On the other hand, the final product is $2n$ bits and extends in binary weight from 2^0 up to 2^{2n-1}.

We pad each of the last two terms in Equation (3) with zeros to obtain a $2n$-bit number to be able to add them to the other terms. The padded terms extend in binary weight from 2^0 up to 2^{2n-1}.

Assuming X is one of the last two terms we can represent it with zero padding as

$$X = -0 \times 2^{2n-1} + 0 \times 2^{2n-2} + 2^{n-1} \sum_{i=0}^{n-2} x_i 2^i + \sum_{j=0}^{n-2} 0 \times 2^j \qquad (4)$$

The above equation gives the *value* of X due to the fact that a negative value is associated with the MSB.

When we *store* X in a register, the negative sign at MSB is not used since X is stored as a *binary pattern*. Thus partial product X is, therefore, represented by

bit position	$2n-1$	$2n-2$	$2n-3$	$2n-4$	$\cdots$	n	$n-1$	$n-2$	$n-3$	$\cdots$	0
bit value	0	0	x_{n-2}	x_{n-3}	$\cdots$	x_1	x_0	0	0	$\cdots$	0

The two's complement of X is obtained by complimenting all bits in the above equation and adding '1' at the LSB:

bit position	$2n-1$	$2n-2$	$2n-3$	$2n-4$	$\cdots$	n	$n-1$	$n-2$	$n-3$	$\cdots$	0
bit value	1	1	$\bar{x}_{n-2}$	$\bar{x}_{n-3}$	$\cdots$	$\bar{x}_1$	$\bar{x}_0$	1	1	$\cdots$	$1+1$

Adding the '1' at LSB will result in the new pattern for $-X$ as

bit position	$2n-1$	$2n-2$	$2n-3$	$2n-4$	$\cdots$	n	$n-1$	$n-2$	$n-3$	$\cdots$	0
bit value	1	1	$\bar{x}_{n-2}$	$\bar{x}_{n-3}$	$\cdots$	$\bar{x}_1$	$\bar{x}_0+1$	0	0	$\cdots$	0

Assuming the last two terms are expressed as X and Y, then adding $-X$ to $-Y$ amounts to adding the following two bit patterns:

bit position	$2n-1$	$2n-2$	$2n-3$	$2n-4$	$\cdots$	n	$n-1$	$n-2$	$n-3$	$\cdots$	0
$-X$	1	1	x_{n-2}	x_{n-3}	$\cdots$	x_1	x_0+1	0	0	$\cdots$	0
$+(-Y)$	1	1	y_{n-2}	y_{n-3}	$\cdots$	y_1	y_0+1	0	0	$\cdots$	0

The '1' pattern at most significant bits transforms into

bit position	$2n-1$	$2n-2$
	1	1
+	1	1
	1	0

Similarly, the '1' pattern at position $n-1$ becomes

bit position	n	$n-1$
		1
+		1
	1	0

The final product $P = A \times B$ in Equation (3) becomes:

$$\begin{aligned} P &= a_{n-1}b_{n-1}2^{2n-2} + \sum_{i=0}^{n-2}\sum_{j=0}^{n-2} a_i b_j 2^{i+j} + \\ &\quad 2^{n-1}\sum_{i=0}^{n-2}\overline{b_{n-1}a_i}2^i + 2^{n-1}\sum_{j=0}^{n-2}\overline{a_{n-1}b_j}2^j + \\ &\quad -2^{2n-1} + 2^n \end{aligned} \tag{5}$$

Let us assume that A and B are 4-bit binary numbers, then the product $P = A \times B$ is 8-bits long and is given by

$$\begin{aligned} P &= a_3b_3 2^6 + \sum_{i=0}^{2}\sum_{j=0}^{2} a_i b_j 2^{i+j} + \\ &\quad 2^3\sum_{i=0}^{2}\overline{b_3 a_i}2^i + 2^3\sum_{j=0}^{2}\overline{a_3 b_j}2^j + \\ &\quad -2^7 + 2^4 \end{aligned} \tag{6}$$

Figure 1 shows the implementation of the Baugh-Wooley multiplier.

The basic Baugh-Wooley cells are shown in Figure 2.

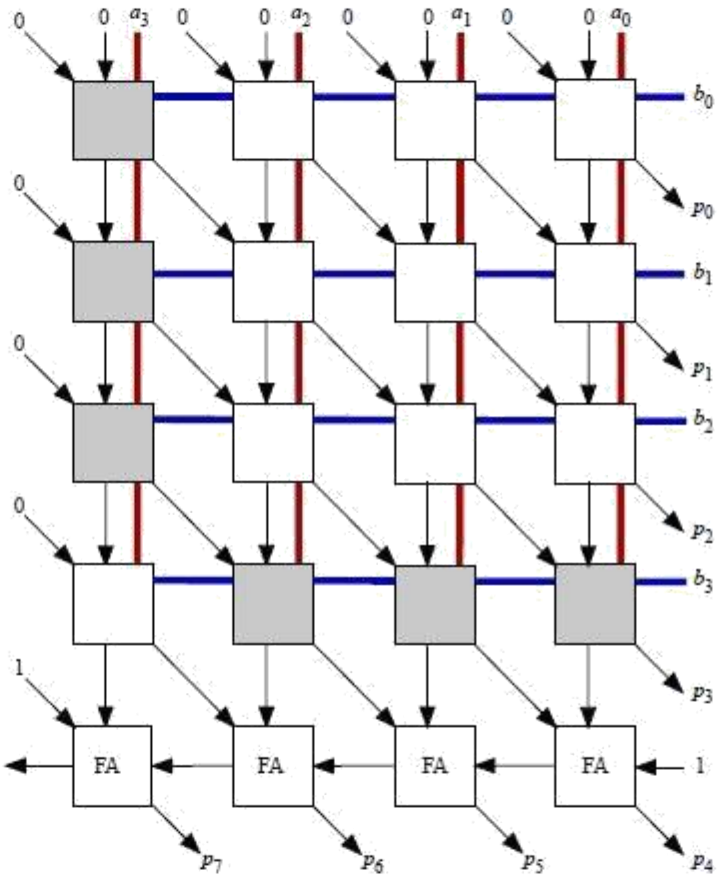
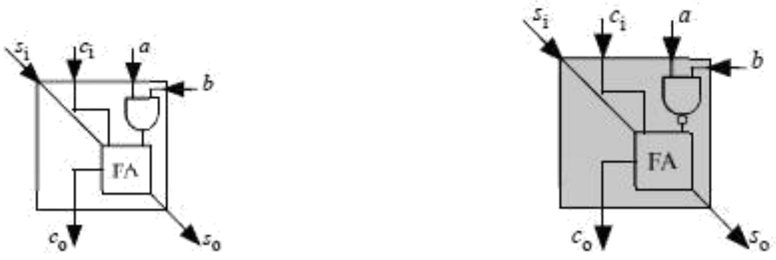


Figure 1: Block diagram of 4 × 4 Baugh-Wooley multiplier



(a) Baugh-Wooley multiplier white-cell (b) Baugh-Wooley multiplier gray-cell

Figure 2: Baugh-Wooley multiplier basic cell construction

Booth Multipliers

It is a powerful algorithm for signed-number multiplication, which treats both positive and negative numbers uniformly.For the standard add-shift operation, each multiplier bit generates one multiple of the multiplicand to be added to the partial product. If the multiplier is very large, then a large number of multiplicands have to be added. In this case the delay of multiplier is determined mainly by the number of additions to be performed. If there is a way to reduce the number of the additions, the performance will get better.

Booth algorithm is a method that will reduce the number of multiplicand multiples. For a given range of numbers to be represented, a higher representation radix leads to fewer digits. Since a k-bit binary number can be interpreted as K/2-digit radix-4 number, a K/3-digit radix-8 number, and so on, it can deal with more than one bit of the multiplier in each cycle by using high radix multiplication. This is shown for Radix-4 in

the example below.

Multiplicad A = ●●●●

Multiplier B = (●●)(●●)

Partial product bits ●●●● $(B1B0)2\ A4^0$

● ●●● $(B3B2)2\ A4^1$

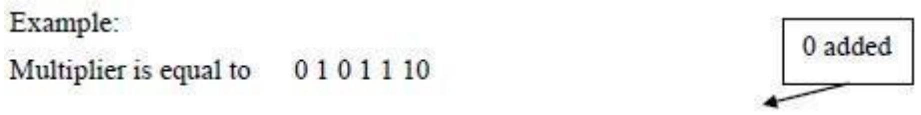
Product P = ● ●●●●●●●

Radix-4 multiplication in dot notation.

As shown in the figure above, if multiplication is done in radix 4, in each step, the partial product term (Bi+1Bi)2 A needs to be formed and added to the cumulative partial product. Whereas in radix-2 multiplication, each row of dots in the partial products matrix represents 0 or a shifted version of A must be included and added.

Table 1 Below is used to convert a binary number to radix-4 number .

Initially, a "0" is placed to the right most bit of the multiplier. Then 3 bits of the multiplicand is recoded according to table below or according to the following equation: Zi = -2xi+1 + xi + xi-1


Example:
Multiplier is equal to 0 1 0 1 1 10

then a 0 is placed to the right most bit which gives 0 1 0 1 1 10 0

the 3 digits are selected at a time with overlapping left most bit as follows:

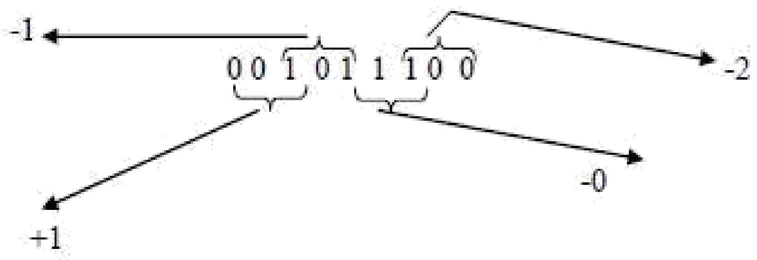


Table .1 Radix-4 Booth recoding

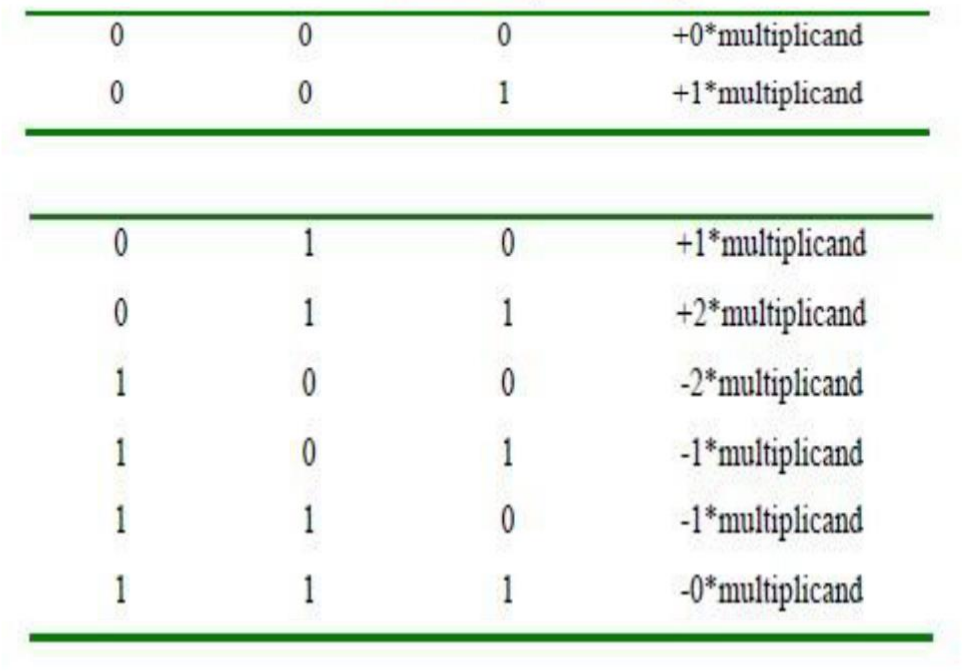
Xi+1	X	Xi-1	Zi/2
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

Example, an unsigned number can be converted into a signed-digit number radix-4:

(10 01 11 01 10 10 11 10)2 = (–2 2 –1 2 –1 –1 0 –2)4

The Multiplier bit-pair recoding is shown in Table .2

Table Multiplier recoding



0	0	0	+0*multiplicand
0	0	1	+1*multiplicand
0	1	0	+1*multiplicand
0	1	1	+2*multiplicand
1	0	0	-2*multiplicand
1	0	1	-1*multiplicand
1	1	0	-1*multiplicand
1	1	1	-0*multiplicand

Here –2*multiplicand is actually the 2s complement of the multiplicand with an equivalent left shift of one bit position. Also, +2 *multiplicand is the multiplicand shifted left one bit position which is equivalent to multiplying by 2. To enter 2*multiplicand into the adder, an (n+1)-bit adder is required. In this case, the multiplicand is offset one bit to the left to enter into the adder while for the low-order multiplicand position a0 is added. Each time the partial product is shifted two bit positions to the right and the sign is extended to the left. During each add-shift cycle, different versions of the multiplicand are added to the new partial product depends on the equation derived from the bit-pair recoding table above.

Example 1:

$$\begin{array}{rl}
000011 & (+3) \\
\times \quad \underbrace{01}\underbrace{11}\underbrace{01}\boxed{0} & (+29) \\
\underline{+2\ -1\ +1} & \\
000000000011 & \\
1111111101 & \\
\underline{00000110\qquad} &
\end{array}$$

◄1 000001010111 (+87)

Example 2:

111101 (-3)

011101 $\boxed{0}$ (+29)

+2 -1 +1

11111111110

2s complement of

1

×

0000000011 multiplicand 1111101

► 0

11111010100
1
1
(-87)

Example 3:

111101 (-3)

100011 $\boxed{0}$ (29)

-2 +1 -1

00000000001

Shifted 2s

1111111101

compleme

0000011

nt

0

00000101011

-

(+87

1 1)

Barrel Shifter

Studying the barrel shifter structure of Fig. 9.15-6 allows some interesting observations. First, note that if the data path runs horizontally, it is necessary to provide a vertical path for control to implement the shift function. This vertical connection is sometimes used to insert or extract external data to or from the data path along D4–D6 or D0–D3. Remembering that control signals S0–S3 for the data path are provided vertically, the vertical path of the barrel shifter can be used to insert data that is part of the data path control instruction. Data that is part of the data path instruction is usually called *literal* or *immediate* data. Second, recognition of the simplicity of the barrel shifter structure suggests that the vertical pitch of the data path layout will probably be limited by the vertical pitch of the register array or the ALU rather than by the barrel shifter. This observation allows minimization of the horizontal dimension of a barrel shifter while the vertical dimension is stretched to match the rest of the data path to obtain area efficiency of the layout.

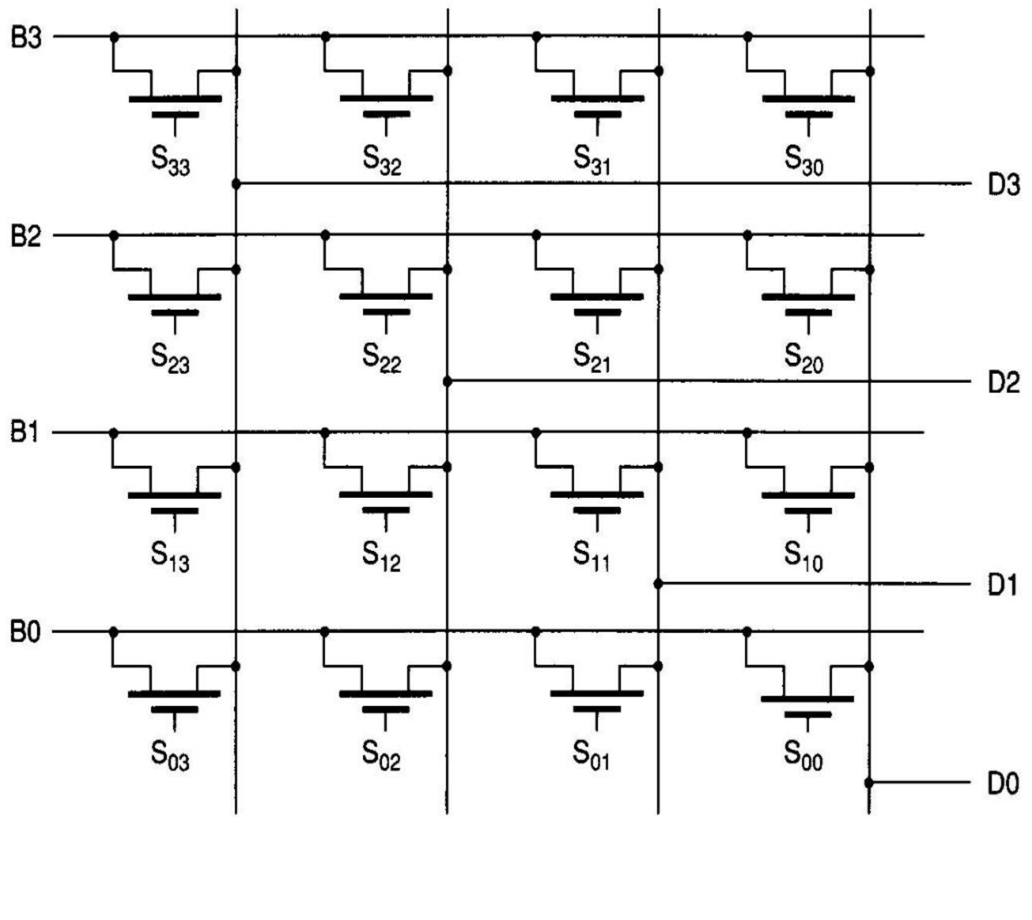


Fig. Barrel shifter (4 control lines S0-S3)

The CMOS SRAM Cell:

A random-access memory (RAM) cell is a circuit that has three main operations. Write - A data bit is stored in the circuit

Hold - The value of the data bit is maintained in the cell Read - The

value of the data bit is transferred to an external circuit

A static RAM (SRAM) cell is capable of holding a data bit so long as the power is applied to the circuit.Figure shows the basic 6 transistor (6T) CMOS SRAM cell. It consists of a central storage cell made up of two cross coupled inverters (Mn1, Mp1 and Mn2, Mp2), and two access transistors MA1 and MA2 that provide for the read and write operations. The conducting state of the access transistors is controlled by the signal *WL* on the Word line. When *WL*=1, both MA1 and MA2 conduct and provide the ability to enter or read a data bit. A value of *WL*=0 gives a hold state where both MA1 and MA2 are driven into cutoff, isolating the storage cell. The access FETs connect the storage cell input/output nodes to the data lines denoted as bit and which are complements of each other. The operation of the circuit can be summarized as follows.

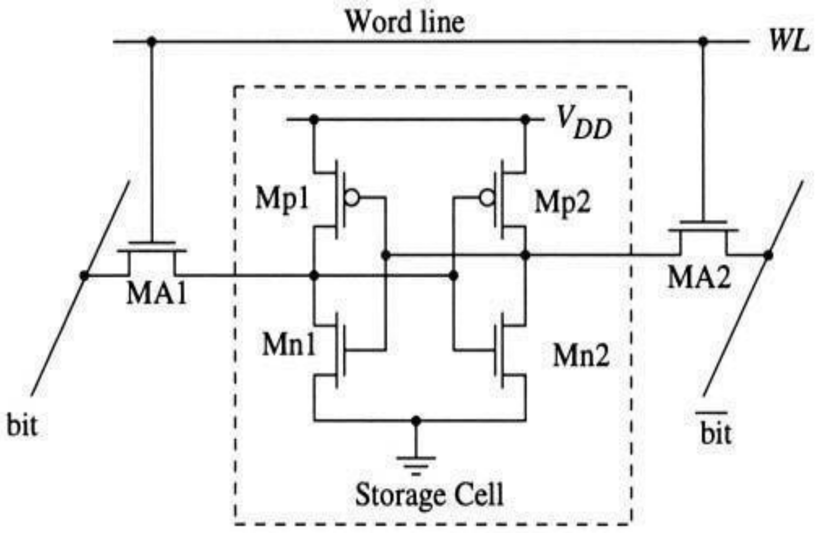


Fig: CMOS 6T SRAM circuit

Write

To write a data bit to the cell, we bring *WL* to a high voltage and placed the voltages on the bit and lines. For example, to write a 1 to the cell, and would be applied as

illustrated in Figure. This drives the internal voltages in a manner that goes high and

Because of the bistable hold characteristic discussed below, changing the voltages on the left and right sides in opposite directions helps the cell latch onto the state

Hold

The hold state of the cell is achieved by bringing the word line signal to *WL* =0. This places both access FETs MA1 and MA2 in cutoff, and isolates the storage cell from the

bit and lines. The basic feature of the storage cell is that it is able to maintain the internal

voltages and at complementary values (i.e., one high and the other low).

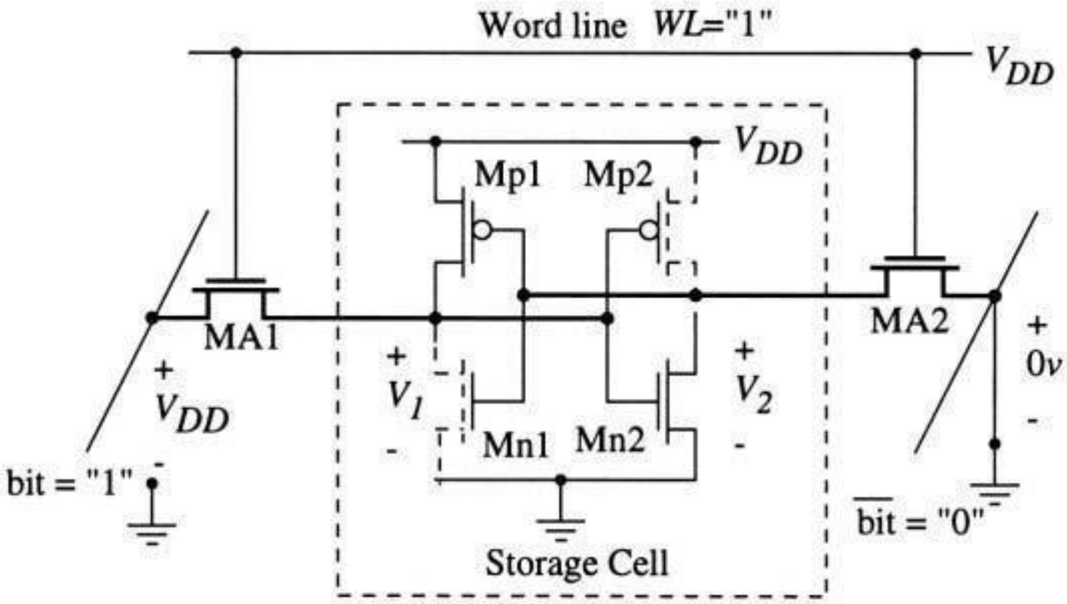
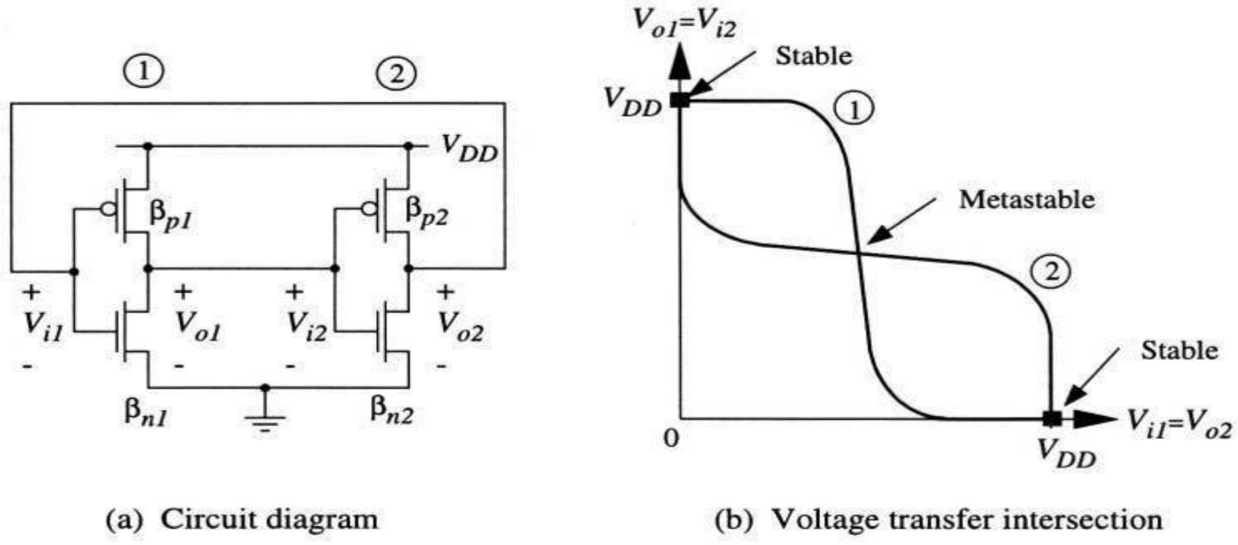


Fig. Write operation of an SRAM circuit



The latching action of the cross-coupled inverters can be studied using the circuit in Figure(a) where we have cascaded two inverters (1 and 2) and closed the loop to provide feedback.

As seen in the drawing, the input voltages and and output voltages and are related to each other by

$$V_{o1} = V_{i2}$$
$$V_{o2} = V_{i1}$$

This allows us to superpose the voltage transfer curve of the two inverters and arrive at the plot shown in Figure. we have assumed identical inverters, i.e., that is the same for both.

There are three intersection points where the VTCs satisfy the voltage relations. The two stable states occur at coordinates (0, and 0) and either can be used for data storage. The third intersection point occurs along the unity gain line, and is labelled as a “metastable” point in the drawing. In practice, the circuit cannot maintain equilibrium at this point even though it is a solution to the voltage requirements due to the dynamic gain associated with the inverter. The diagram also illustrates the voltages needed to trigger the circuit during the write operation.

To induce a fall in an inverter VTC, the input voltage should be above Thus we could arguethat the voltage received from the access device must exceed this value to insure the writing of a logic 1. Although highly simplified, this is a reasonable statement of the necessary condition; the switching is aided by the fact that the opposite side of the cell will be at 0*v*.

Read

The read operation is used to transfer the contents of the storage cell to the bit and lines. Bringing the Word line high with *WL*=1 activates the access transistors, and allows the voltage transfer to take place. Figure. shows the read operation for the case where a logic 1 is stored in the cell. This gives internal cell voltages of and 0*v* on the left and right sides, respectively. Since the access FETs act like pass transistors, the line
voltages as driven by the cell are

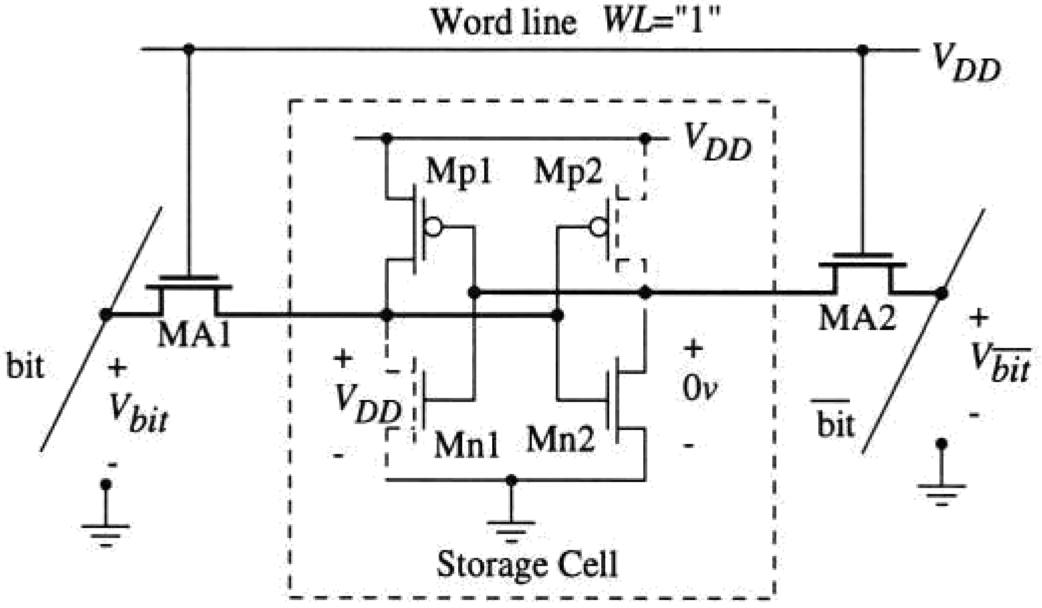


Fig. Read operation of an SRAM circuit

$$V_{bit} \rightarrow V_{max} = (V_{DD} - V_{Tn})$$
$$V_{\overline{bit}} \rightarrow 0v$$

In particular, the nFET induces a threshold voltage loss that prevents from reaching To overcome this problem (and speed up the detection process), and are used as inputs to a sensitive differential sense amplifier that can detect the state.

The Dynamic RAM Cell

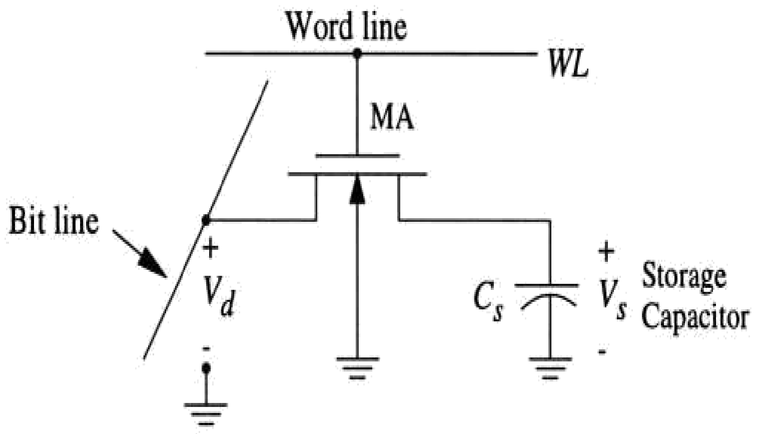
A dynamic random-access memory (DRAM) cell is a storage circuit that consists of an access transistor MA and a storage capacitor as shown in Figure. The access FET is controlled by the Word line signal and the bit line is the input/output path. The simplicity of the circuit makes it very attractive for high-density storage. As with any RAM, there are three distinct operations for a cell:

Write - A data bit is stored in the circuit;

Hold - The value of the data bit is maintained in the cell;and

Read - The value of the data bit is transferred to an external circuit.

Access to the capacitor is controlled by the word line signal that is connected to the gate of the access transistor. Aside from this change in nomenclature, the cell itself is identical to the circuit in Figure, so that the operation is easily understood by applying the results of this chapter to each of the three operational modes.



Write

The write operation is illustrated in Figure. The word line voltage is elevated to a

value of (corresponding to a logic a level to turn on the access FET; the input data voltage on the bit line then establishes the required charge on the capacitor. To store a logic 1 in the cell, is set to the value of so that the storage cell voltage increases according to

$$V_s(t) = (V_{DD} - V_{Tn})\left[\frac{t/2\tau_s}{1 + t/2\tau_s}\right]$$

where the time constant is given by

$$\tau_s = \frac{C_s}{\beta_n (V_{DD} - V_{Tn})}.$$

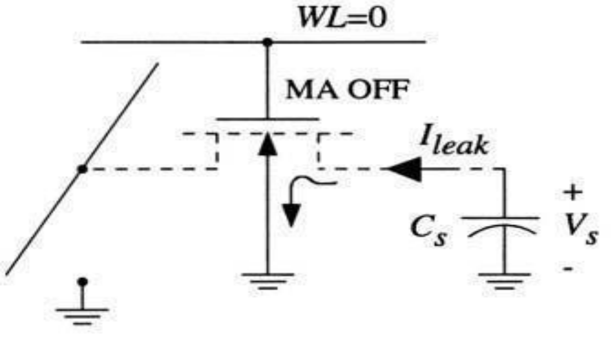
Storage of logic 0 is accomplished by using an input voltage of so that the capacitor is discharged as described by

$$V_s(t) = V_s(0)\left[\frac{2e^{-t/\tau_s}}{1 + e^{-t/\tau_s}}\right]$$

where we have designated the voltage at the beginning of the write operation as From our earlier discussions, we may conclude that writing a logic 1 value requires more time than writing a logic 0 state.

Hold

A DRAM cell holds the charge on the capacitor by turning off the access transistor using as shown in Figure. This creates an isolated node, and charge leakage occurs if a logic 1 high voltage is stored on The maximum hold time for a logic 1 bit can be estimated by

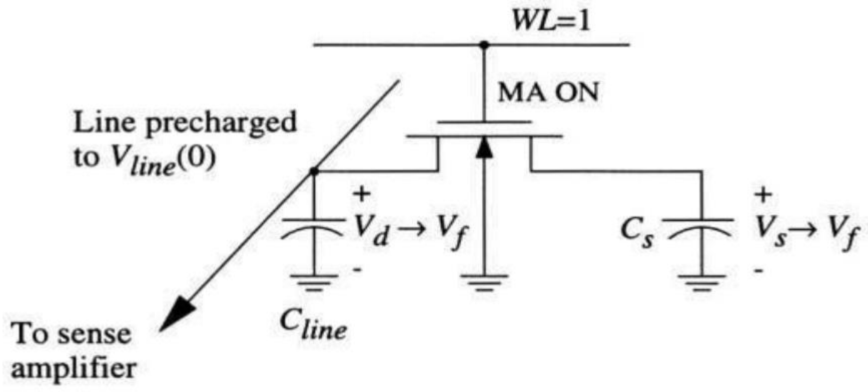


$$t_H \approx \frac{C_s}{I_{leak}}(V_{max} - V_1)$$

and is usually limited to a duration on the order of 100 milliseconds. In physical DRAM cell designs, several contributions to the leakage current are found, and much time is dedicated to the problem of increasing the charge retention time. This change in the stored charge (and hence, the logic level) in time is the origin of the name dynamic. Special refresh circuits that periodically read the bit, amplify the voltage, and rewrite the data to the cell must be added so that the circuit can be used to store data for longer periods of time.

Read:

When a read operation is performed, the data bit line is connected to the input of a high-gain sense amplifier that is designed to provide amplification of the voltage level. During this operation, the line is connected to FET gates terminals so that the line itself is capacitive. This is included in Figure by including a line capacitance During a read operation, charge sharing will take place between the storage cell and the output line, resulting in a final voltage data voltage of



$$V_d = \left(\frac{C_s}{C_s + C_{line}}\right) V_s$$

A major difficulty in designing high-density DRAM chips is that the cell storage capacitance is relatively small compared to the parasitic line capacitance For example, a ratio with a value would be considered reasonable in modern high-density design. This implies that the difference between a logic 0 and a logic 1 data voltage is very small. One way to help this situation is to precharge the line to an initial voltage before the read operation. The final voltage after charge sharing takes place is then given by

$$V_d = \left(\frac{C_s}{C_s + C_{line}}\right) V_s + \left(\frac{C_{line}}{C_s + C_{line}}\right) V_{line}(0)$$

For example, suppose that we choose Assuming that the line capacitance is large compared to the storage capacitance, the final voltage on the line will be

$$V_d \approx \Delta V_s + \left(\frac{1}{2}\right) V_{DD}$$

where ΔV_S is the change due to the stored charge. This gives an output voltage that changes around the reference value of this change can be detected using a comparator as a sense amplifier.

Programmable Logic Devices (PLDs)

Introduction:

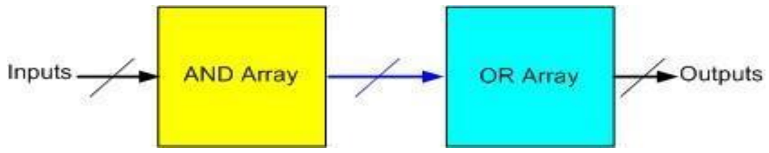
An IC that contains large numbers of gates, flip-flops, etc. that can be configured by the user to perform different functions is called a Programmable Logic Device (PLD).

The internal logic gates and/or connections of PLDs can be changed/configured by a programming process.

One of the simplest programming technologies is to use fuses. In the original state of the device, all the fuses are intact.

Programming the device involves blowing those fuses along the paths that must be removed in order to obtain the particular configuration of the desired logic function.

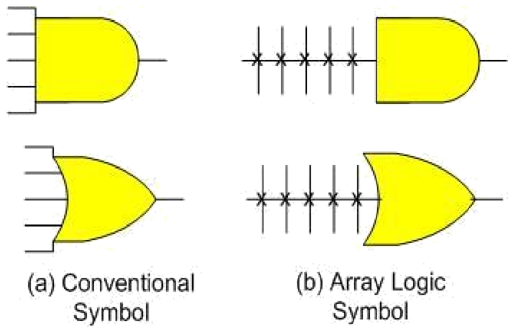
PLDs are typically built with an array of AND gates (AND-array) and an array of OR gates (OR-array).



Advantages of using PLDs:

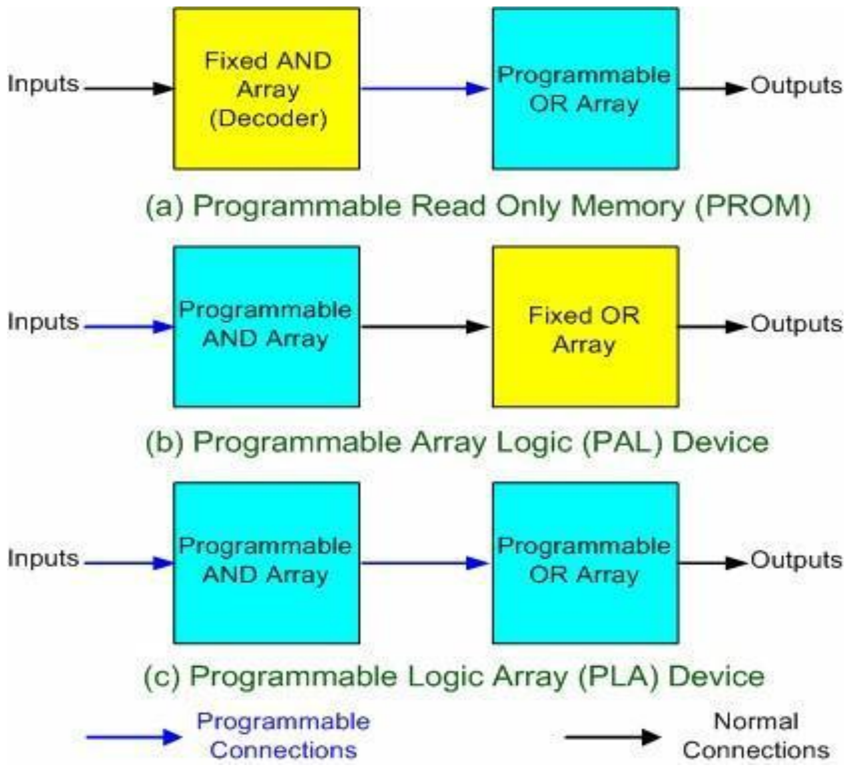
Advantages of using PLDs are less board space, faster, lower power requirements (i.e., smaller power supplies), less costly assembly processes, higher reliability (fewer ICs and circuit connections means easier troubleshooting), and availability of design software.

There are three fundamental types of standard PLDs: PROM, PAL, and PLA. A fourth type of PLD, which is discussed later, is the Complex Programmable Logic Device (CPLD), e.g., Field Programmable Gate Array (FPGA). A typical PLD may have hundreds to millions of gates. In order to show the internal logic diagram for such technologies in a concise form, it is necessary to have special symbols for array logic. Figure shows the conventional and array logic symbols for a multiple input AND and a multiple input OR gate.



Three Fundamental Types of PLDs:

The three fundamental types of PLDs differ in the placement of programmable connections in the AND-OR arrays. Figure shows the locations of the programmable connections for the three types.



d) The PROM (Programmable Read Only Memory) has a fixed AND array (constructed as a decoder) and programmable connections for the output OR gates array. The PROM implements Boolean functions in sum-of-minterms form.

e) The PAL (Programmable Array Logic) device has a programmable AND array and fixed connections for the OR array.

f) The PLA (Programmable Logic Array) has programmable connections for both AND and OR arrays. So it is the most flexible type of PLD.

The PLA (Programmable Logic Array):

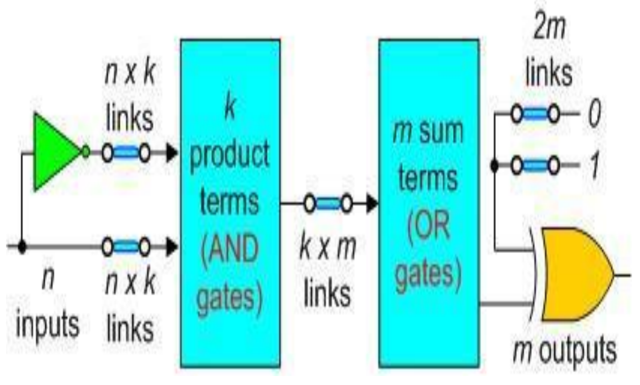
The input lines to the AND array are hard-wired and the output lines to the OR array are programmable. Each AND gate generates one of the possible AND products (i.e., minterms). In PLAs, instead of using a decoder as in PROMs, a number (k) of AND

n

gates is used where k < 2 , (n is the number of inputs). Each of the AND gates can be

programmed to generate a product term of the input variables and does not generate all the minterms as in the ROM. The AND and OR gates inside the PLA are initially fabricated with the links (fuses) among them. The specific Boolean functions are implemented in sum of products form by opening appropriate links and leaving the desired connections.

A block diagram of the PLA is shown in the figure. It consists of n inputs, m outputs, and k product terms.

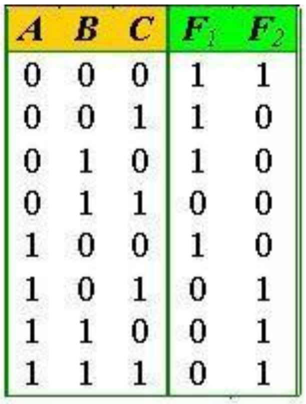


The product terms constitute a group of k AND gates each of 2n inputs. Links are inserted between all n inputs and their complement values to each of the AND gates. Links are also provided between the outputs of the AND gates and the inputs of the OR gates. Since PLA has m-outputs, the number of OR gates is m. The output of each OR gate goes to an XOR gate, where the other input has two sets of links, one connected to logic 0 and other to logic 1. It allows the output function to be generated either in the true form or in the complement form.

The size of the PLA is specified by the number of inputs (n), the number of product terms (k), and the number of outputs (m), (the number of sum terms is equal to the number of outputs).

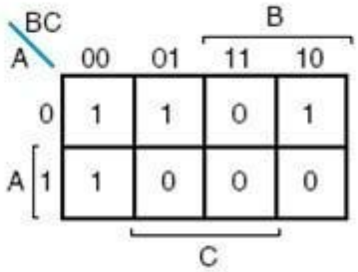
Example:

Implement the combinational circuit having the shown truth table, using PLA.

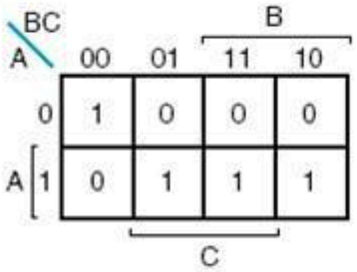


A	B	C	F_1	F_2
0	0	0	1	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

Each product term in the expression requires an AND gate. To minimize the cost, it is necessary to simplify the function to a minimum number of product terms.



$$F_1 = \overline{A}\,\overline{B} + \overline{A}\,\overline{C} + \overline{B}\,\overline{C}$$
$$\overline{F_1} = AB + AC + BC$$



$$F_2 = AB + AC + \overline{A}\,\overline{B}\,\overline{C}$$
$$\overline{F_2} = \overline{A}C + \overline{A}B + A\overline{B}\,\overline{C}$$

Designing using a PLA, a careful investigation must be taken in order to reduce the distinct product terms. Both the true and complement forms of each function should be simplified to see which one can be expressed with fewer product terms and which one provides product terms that are common to other functions.

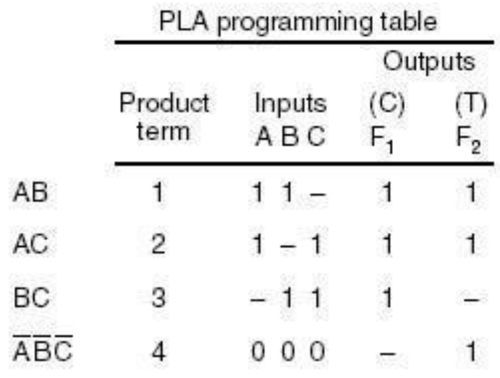
,

The combination that gives a minimum number of product terms is: F_1 = AB + AC + BC

or F_1 = (AB + AC + BC)' F_2 = AB + AC + A'B'C'

This gives only 4 distinct product terms: AB, AC, BC, and A'B'C'.

So the PLA table will be as follows:



PLA programming table

	Product term	Inputs	Outputs (C)	Outputs (T)
		A B C	F_1	F_2
AB	1	1 1 –	1	1
AC	2	1 – 1	1	1
BC	3	– 1 1	1	–
$\overline{A}\overline{B}\overline{C}$	4	0 0 0	–	1

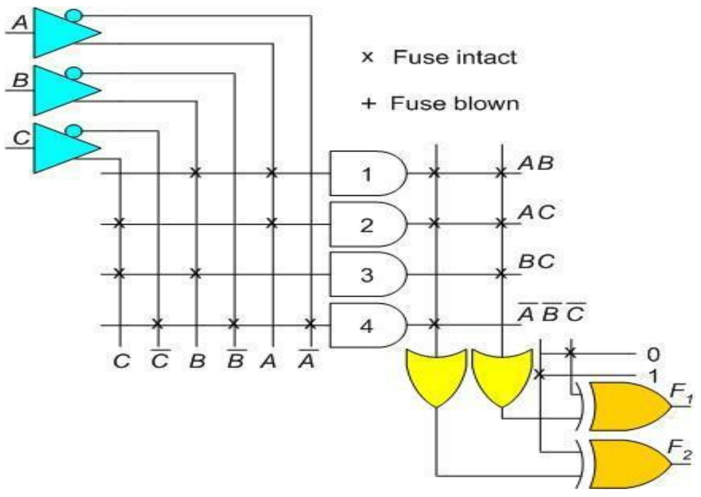
For each product term, the inputs are marked with 1, 0, or – (dash). If a variable in the product term appears in its normal form (unprimed), the corresponding input variable is

marked with a 1.

A 1 in the Inputs column specifies a path from the corresponding input to the input of the

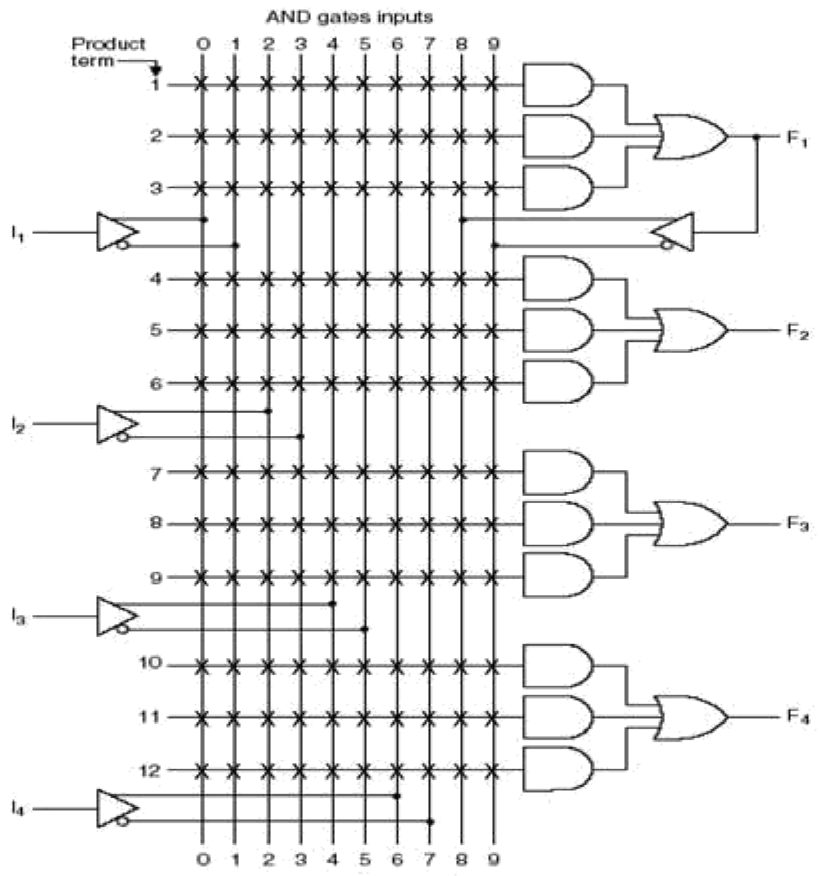
AND gate that forms the product term.

A 0 in the Inputs column specifies a path from the corresponding complemented input to the input of the AND gate. A dash specifies no connection. The appropriate fuses are blown and the ones left intact form the desired paths. It is assumed that the open terminals in the AND gate behave like a 1 input. In the Outputs column, a T (true) specifies that the other input of the corresponding XOR gate can be connected to 0, and a C (complement) specifies a connection to 1. Note that output F_1 is the normal (or true) output even though a C (for complement) is marked over it. This is because F_1' is generated with AND-OR circuit prior to the output XOR. The output XOR complements the function F_1' to produce the true F_1 output as its second input is connected to logic 1.


A
B
C
× Fuse intact
+ Fuse blown
1
2
3
4
AB
AC
BC
$\overline{A}\,\overline{B}\,\overline{C}$
C $\overline{C}$ B $\overline{B}$ A $\overline{A}$
0
1
F_1
F_2

The PAL (Programmable Array Logic):

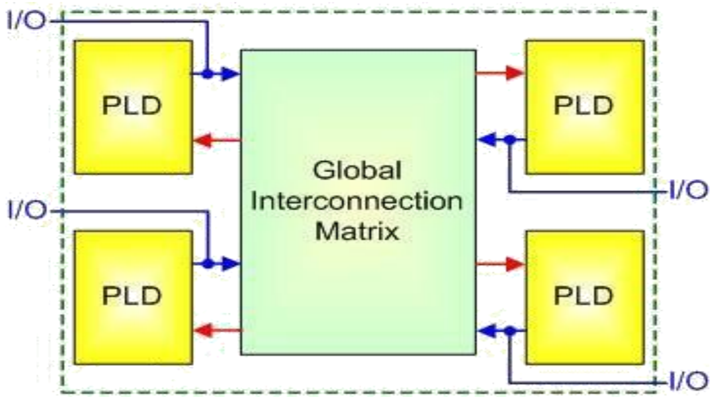
The PAL device is a PLD with a fixed OR array and a programmable AND array. As only AND gates are programmable, the PAL device is easier to program but it is not as flexible-as-the-PLA.



The device shown in the figure has 4 inputs and 4 outputs. Each input has a buffer-inverter gate, and each output is generated by a fixed OR gate. The device has 4 sections, each composed of a 3-wide AND-OR array, meaning that there are 3 programmable AND gates in each section. Each AND gate has 10 programmable input connections indicating by 10 vertical lines intersecting each horizontal line. The horizontal line symbolizes the multiple input configuration of an AND gate. One of the outputs F_1 is connected to a buffer-inverter gate and is fed back into the inputs of the
AND gates through programmed connections. Designing using a PAL device, the Boolean functions must be simplified to fit into each section. The number of product terms in each section is fixed and if the number of terms in the function is too large, it may be necessary to use two or more sections to implement one Boolean function.

Complex Programmable Logic Devices (CPLDs):

A CPLD contains a bunch of PLD blocks whose inputs and outputs are connected together by a global interconnection matrix. Thus a CPLD has two levels of programmability: each PLD block can be programmed, and then the interconnections
between the PLDs can be programmed.



Field Programmable Gate Arrays (FPGAs)

The FPGA consists of 3 main structures:

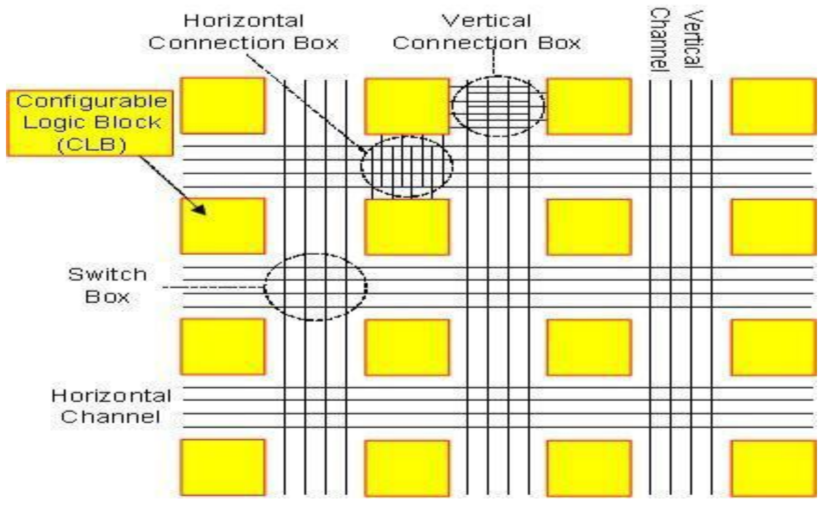
Programmable logic structure,

Programmable routing structure, and

Programmable Input / Output (I/O).

d) Programmable logic structure

The programmable logic structure FPGA consists of a 2-dimensional array of configurable logic blocks (CLBs).



Each CLB can be configured (programmed) to implement any Boolean function of its input variables. Typically CLBs have between 4-6 input variables. Functions of larger number of variables are implemented using more than one CLB. In addition, each CLB typically contains 1 or 2 FFs to allow implementation of sequential logic.

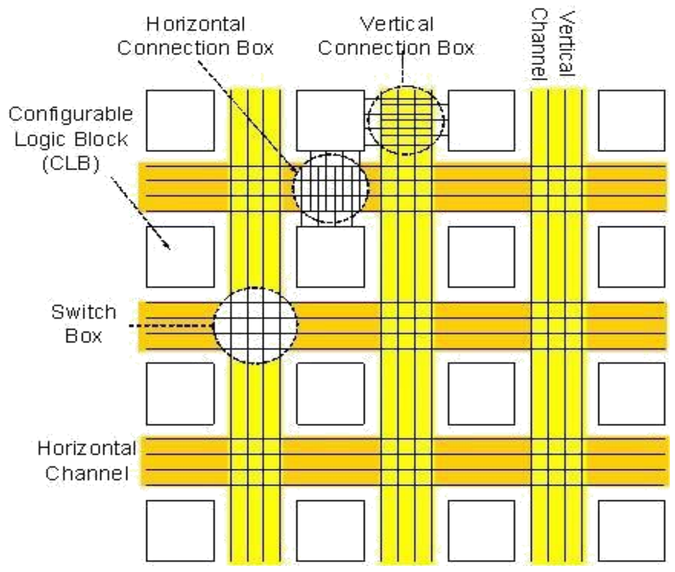
Large designs are partitioned and mapped to a number of CLBs with each CLB configured (programmed) to perform a particular function. These CLBs are then connected together to fully implement the target design. Connecting the CLBs is done using the FPGA programmable routing structure.

3 Programmable routing structure

To allow for flexible interconnection of CLBs, FPGAs have 3 programmable routing resources:

- Vertical and horizontal routing channels which consist of different length wires that can be connected together if needed. These channels run vertically and horizontally between columns and rows of CLBs as shown in the Figure.
- Connection boxes, which are a set of programmable links that can connect input and output pins of the CLBs to wires of the vertical or the horizontal routing channels.
- Switch boxes, located at the intersection of the vertical and horizontal channels.

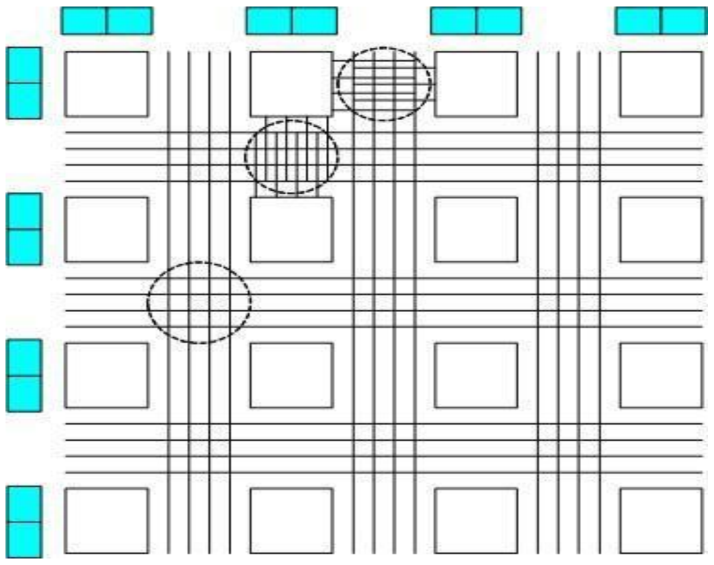
  These are a set of programmable links that can connect wire segments in the horizontal and vertical channels.



3. Programmable I/O

These are mainly buffers that can be configured either as input buffers, output buffers or input/output buffers.

- They allow the pins of the FPGA chip to function either as input pins, output pins or input/output pins



Questions for practice

PART-A

1. What are the types of multipliers available?
2. What is booth algorithm?
3. What is Array Multiplier?
4. Draw the diagram of a basic cell of GFA.
5. What is PLA?
6. What is modified booth algorithm?
7. What is Braun multiplier?
8. Draw the circuit diagram for three transistor memory.
9. Differentiate between SRAM and DRAM.
10. Define FPGA.

PART-B

1. Explain the Static and Dynamic Adder circuits
2. Explain in detail about Baugh-Wooley multiplier?
3. Discuss briefly about Braun multiplier?
4. Explain about systolic array multiplier?
5. Design and Explain the Barrel shifter.
6. Discuss in detail about SRAM memory cell.
7. Explain about PLA and PAL?
8. Discuss in detail about FPGA?

UNIT-V ASIC CONSTRUCTION.

Physical design - Goals and Objectives - Partitioning methods - Kernighan Lin algorithm - Hierarchical Floor planning - Floor planning tools -input, output and power planning -Min-cut placement, Force directed placement algorithm -Placement using simulated annealing - Greedy channel routing

----

Objective

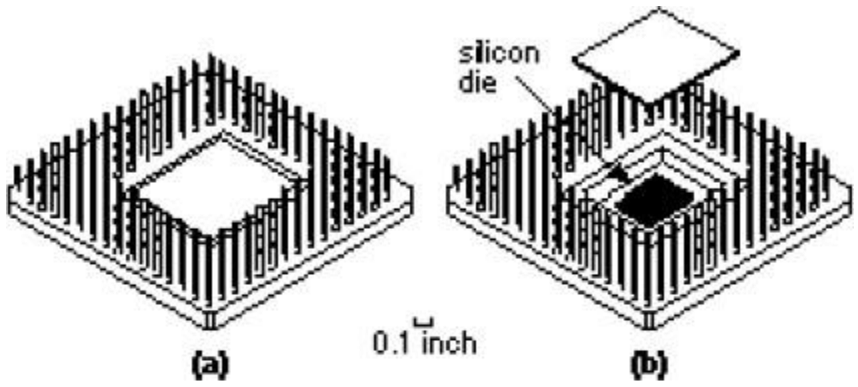
This unit provides the students, the knowledge about

– Physical design flow of IC
    - Floor-planning, Placement and Routing

Integrated Circuit

- **Wafer :** A circular piece of pure silicon (10-15 cm in dia, wafers of 30 cm dia are expected soon)
- **Wafer Lot:** 5 ~ 30 wafers, each containing hundreds of chips(dies) depending upon size of the die
- **Die:** A rectangular piece of silicon that contains one IC design
- **Mask Layers:** Each IC is manufactured with successive mask layers(10 – 15 layers)
    - First half-dozen or so layers define transistors
    - Other half-dozen define Interconnect

Integrated Circuit (IC) in a package



(a) A pin-grid array (PGA) package.

(b) The silicon die or chip is under the package lid.

Evolution of IC

- SSI (S**mall-Scale Integration)-(1962)**
  - **Tens of Transistors**
  - **NAND, NOR**
- MSI (**Medium-Scale Integration)-(late 1960)**
  - **Hundreds of Transistors**
  - **Counters**
- LSI (**Large-Scale Integration)-(mid 1970)**
  - **Tens of Thousands of Transistors**
  - **First Microprocessor**
- VLSI (**Very Large-Scale Integration)-(1980)**
  - **started Hundreds of Thousands of Transistors-several billion transistors in 2009**
  - **64 bit Microprocessor with cache memory and floating-point arithmetic units**
- ULSI (**Ultra Large-Scale Integration)-(late 1980)**
  - **More than about one million circuit elements on a single chip.**
  - **The Intel 486 and Pentium microprocessors, use ULSI technology**

More accuracy

- MOS
  - Gate-Aluminium
  - Low power consumption
  - Low cost
- CMOS
  - Gate-Poly-Silicon
  - Low power consumption
  - Low cost
- BiCMOS

ASIC Design Flow

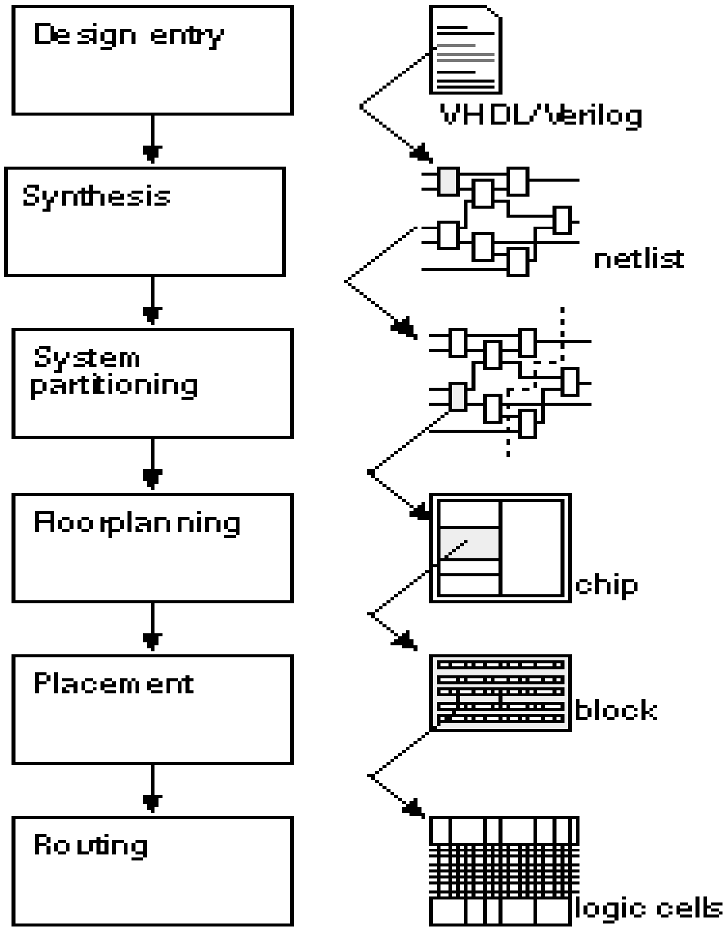
1. Design entry.	*Enter the design into an ASIC design system, either using a* ***h*****ardware description language (HDL) or** ***schematic entry*** **.**
2. Logic synthesis.	*Use an HDL (VHDL or Verilog) and a logic synthesis tool to p*roduce a **netlist —a description of the logic cells and their connections.**
3. System partitioning.	*Divide a large system into ASIC-sized pieces.*
4. Prelayout simulation.	*Check to see if the design functions correctly.*
5. Floorplanning.	*Arrange the blocks of the netlist on the chip.*
6. Placement.	*Decide the locations of cells in a block.*
7. Routing.	*Make the connections between cells and blocks.*
8. Extraction.	*Determine the resistance and capacitance of the interconnect.*
9. Postlayout simulation.	*Check to see the design still works with the added loads* of the interconnect.

Steps 1–4 are part of **logical design ,** and steps 5–9 are part of physical design **.**

- There is some overlap. For example, s**ystem partitioning might be considered as either logicalor physical design.** To put it another way, when we are performing system partitioning we have to consider both logical and physical factors.

ASIC Construction & System Partitioning

Physical Design Steps



- **Part of an ASIC design flow showing the system partitioning, floorplanning, placement, and routing steps.**
- **Performed in a slightly different order, iterated or omitted depending on the type and size of the system and its ASICs.**
- **Floorplanning assumes an increasingly important role.**
- **Sequential-Each of the steps shown in the figure must be performed and each depends on the previous step.**
- **Parallel- However, the trend is toward completing these steps in a parallel fashion and iterating, rather than in a sequential manner.**

CAD Tools

System partitioning:

- **Goal. Partition a system into a number of ASICs.**
- **Objectives.**
  - **Minimize the number of external connections between the ASICs.**
  - **Keep each ASIC smaller than a maximum size.**

Floor planning:

- **Goal. Calculate the sizes of all the blocks and assign them locations.**
- **Objective. Keep the highly connected blocks physically close to each other.**

Placement:

- **Goal. Assign the interconnect areas and the location of all the logic cells within the flexible blocks.**
- **Objectives. Minimize the ASIC area and the interconnect density.**

Global routing:

- **Goal. Determine the location of all the interconnect.**
- **Objective. Minimize the total interconnect area used.**

Detailed routing:

- **Goal. Completely route all the interconnect on the chip.**
- **Objective. Minimize the total interconnect length used.**

Methods and Algorithms

- **Each of the ASIC physical design steps, in general, belongs to a class of mathematical problems known as NP-complete problems.**
- **Definition** : This means that it is unlikely we can find an algorithm to solve the problem exactly in polynomial time.
- **Polynomial:** If the time it takes to solve a problem increases with the size of the problem at a rate that is polynomial but faster than quadratic (or worse in an exponential fashion).
- A CAD tool needs **methods or algorithms to generate a solution to each problem using a reasonable amount of computer time.**

Measurement or objective function

Measurement or objective function:

We need to make **a quantitative measurement** of the quality of the solution that we are able to find.

Often we combine **several parameters or metrics that measure our goals and objectives into a measurement function or Objective function**.

Cost Function :

If we are minimizing the measurement function, it is a cost function.

Gain function :

If we are maximizing the measurement function, we call the function a gain function (sometimes just gain).

ASIC Physical steps

- Each step of ASIC physical design steps are solved by:
  - A set of goals and objectives
  - A way to measure the goals and objectives
  - Algorithm or method to find a solution that meets the goals and objectives.

VLSI Physical Design :- Partitioning

- **System partitioning requires**
  - Goals and Objectives
  - Methods and algorithms to find solutions
  - Ways to evaluate these solutions.

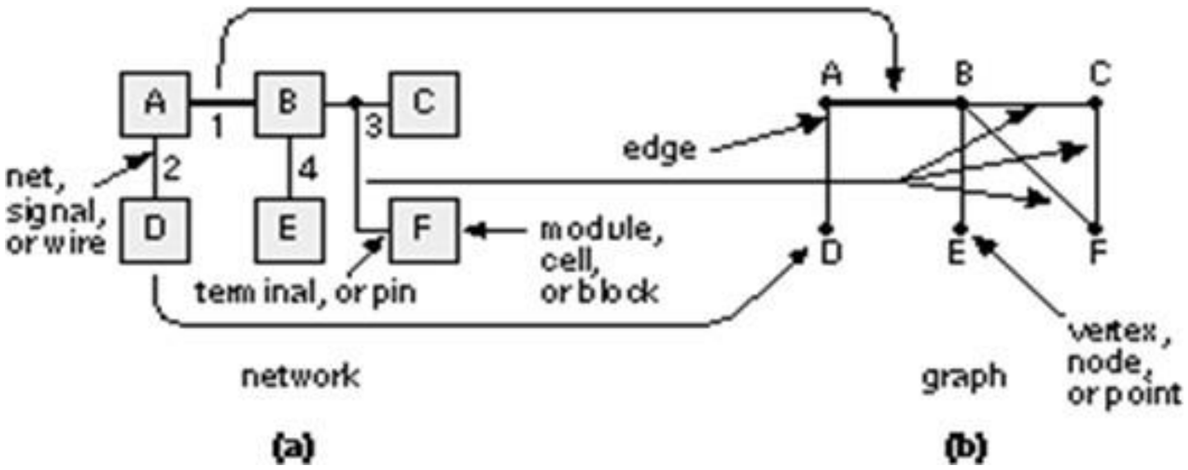
- **Goal of partitioning**
  - Divide the system into number of small systems.

- **Objectives of Partitioning**

  we may need to take into account any or all of the following objectives:

  - A maximum size for each ASIC
  - A maximum number of ASICs
  - A maximum number of connections for each ASIC
  - A maximum number of total connections between all ASICs

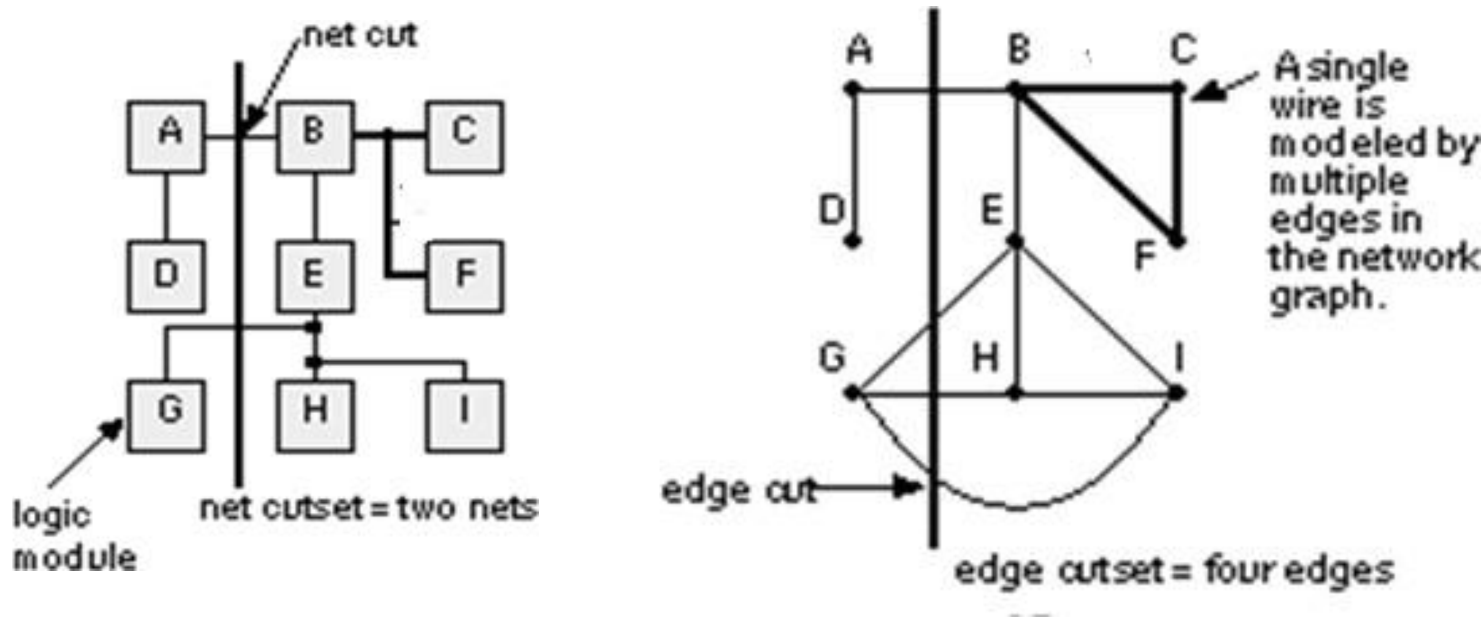
Measuring Connectivity



Measuring Connectivity

- Figure (a) shows a **circuit schematic, netlist, or network.**
- The **network** consists of **circuit modules A–F**. Equivalent terms for a **circuit module - cell, logic cell, macro, or a block**.
- A **cell or logic cell -a small logic gate (NAND etc.)**,**collection of other cells**;
- **Macro - gate-array cells**;
- **Block - a collection of gates or cells**.
- Each logic cell has **Electrical connections between the terminals- connectors or pins**.
- The **network** can be represented as the **mathematical graph** shown in Figure (b).
- A **graph** is like a **spider's web**:
  - it contains **vertexes (or vertices)** A–F -g**raph nodes or points**) that are connected by edges.
  - A **graph vertex** corresponds to a **logic cell**.
  - An **electrical connection (a net or a signal) between two logic cells** corresponds to a **graph edge**.

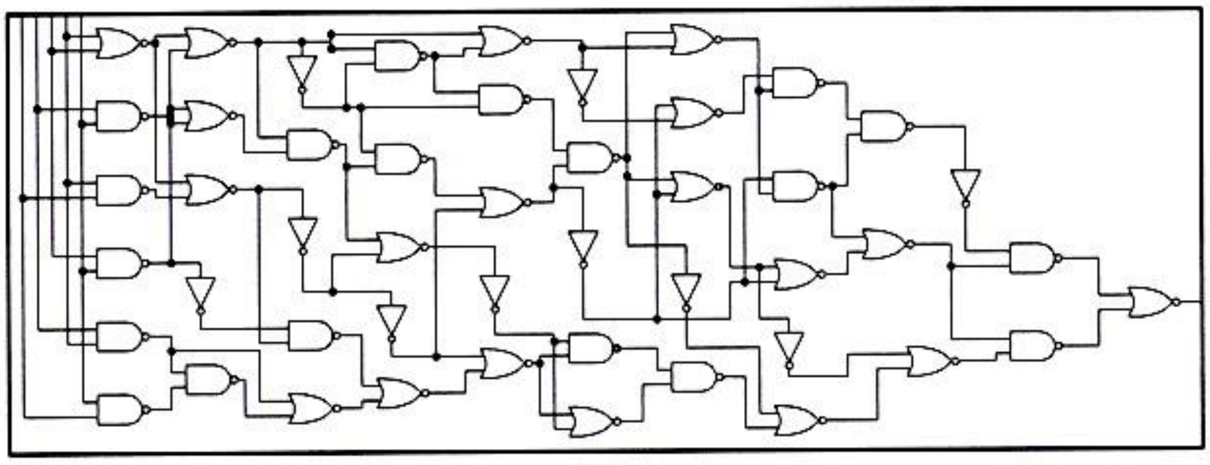
Measuring Connectivity

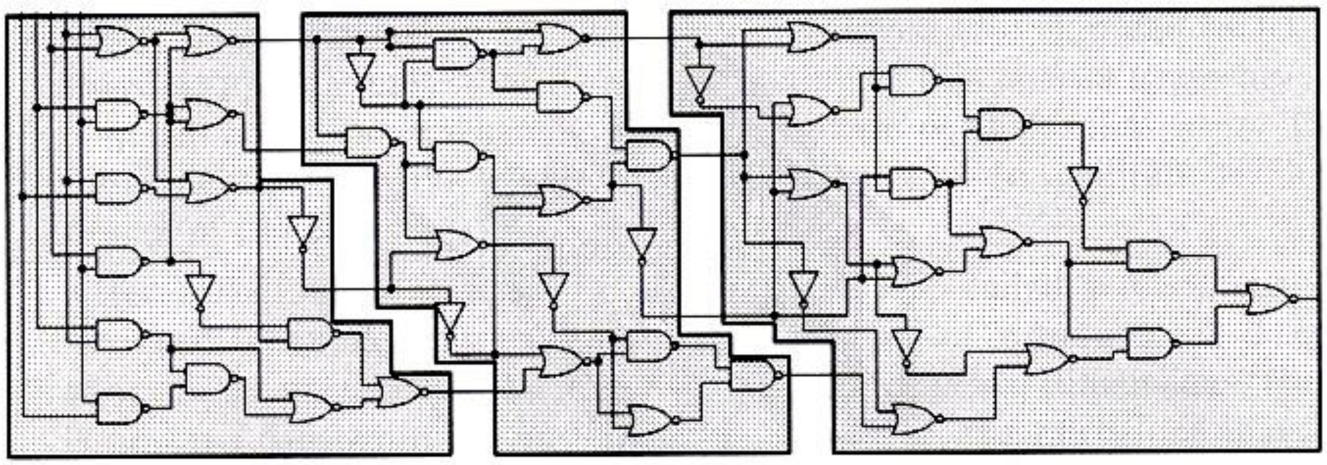


Measuring Connectivity

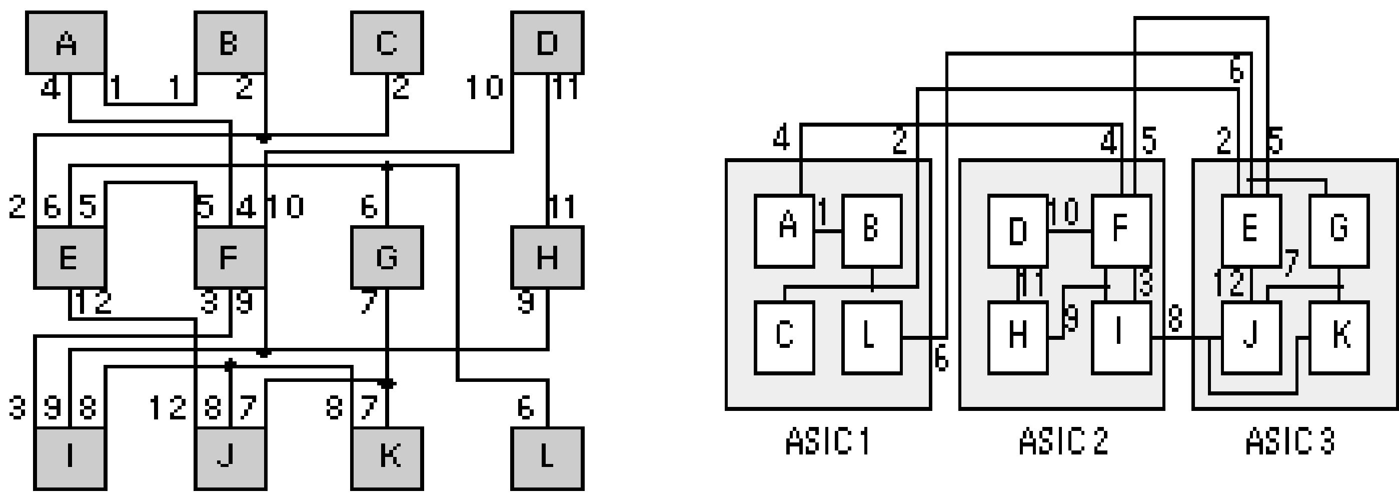
- Net Cutset
  - Divide the network into two by drawing a line across connections,make net cuts. The resulting set of net cuts is the net cutset.
  - Number of net cuts - the number of external connections between the two partitions in a network.
- Edge cutset.
  - When we divide the network graph into the same partitions we make edge cuts and we create the edge cutset.
  - Number of edge cuts – the number of external connections between the two partitions in a graph
  - **Number of edge cuts in a graph is not necessarily equal to the number of net cuts in the network.**

Partitioning of a Circuit


(a)


(b)

A Simple Partitioning Example



• FIGURE 15.7 Partitioning example.

• Goal: to partition our simple network into ASICs.

• (a) We wish to partition this network into three ASICs with no more than four logic cells per ASIC.

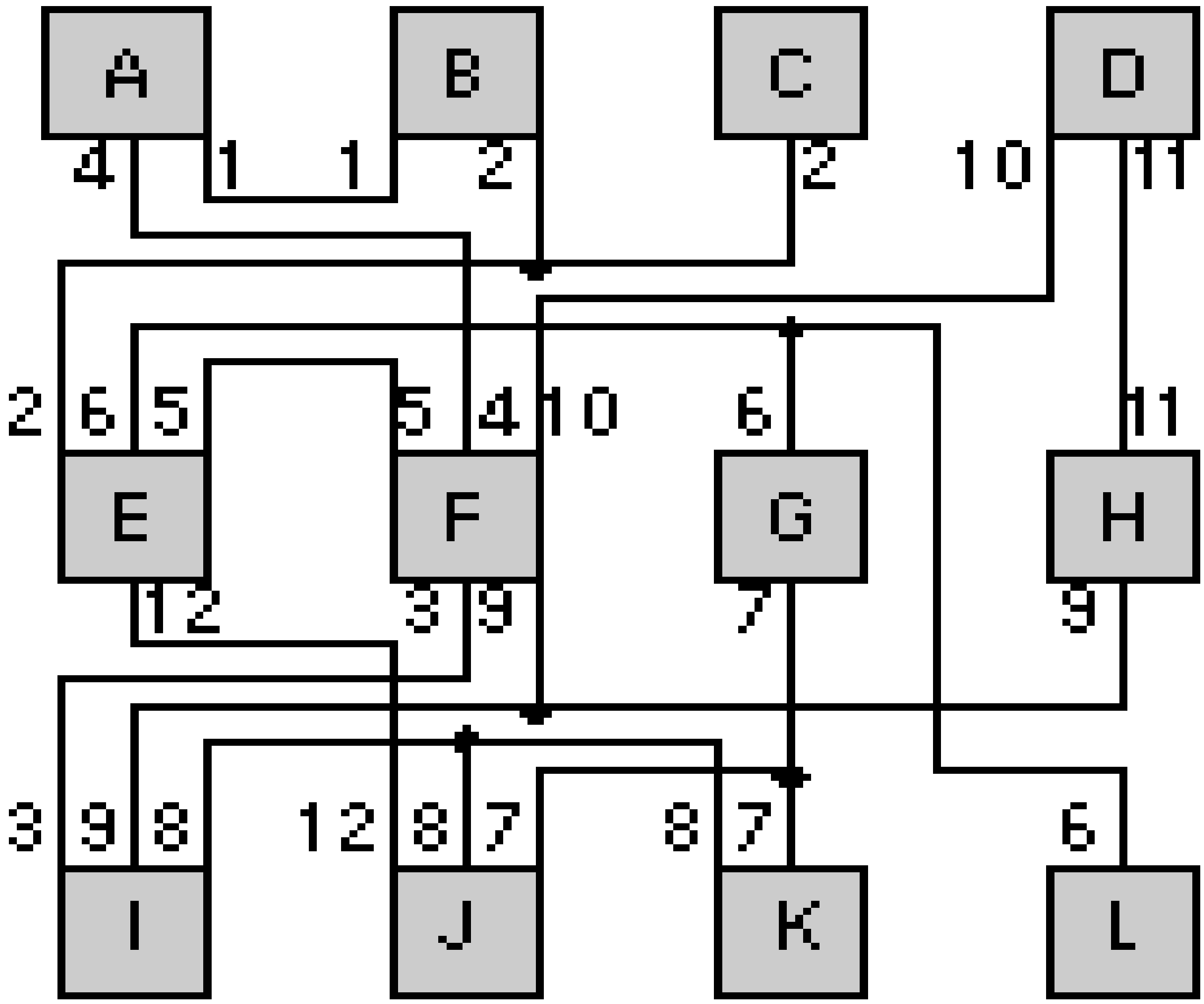
• (b) A partitioning with five external connections (nets 2, 4, 5, 6, and 8)—the minimum number.

• Objectives are the following:

• -Use no more than three ASICs.

• -Each ASIC is to contain no more than four logic cells.

• -Use the minimum number of external connections for each ASIC.


A
B
C
D
4
1
1
2
2
10
11
2
6
5
5
4
10
6
11
E
F
G
H
12
3
9
7
9
3
9
8
12
8
7
8
7
6
I
J
K
L

Types of Partitioning

Splitting a network into several pieces - network partitio

Two types of algorithms used in system partitioning are

- Constructive partitioning - uses a set of rules to find a solution.
- Iterative partitioning improvement (or iterative partitioning refinement - takes an existing solution and tries to improve it.

Constructive Partitioning

- The most common constructive partitioning algorithms - **seed growth or cluster growth.**
- The **steps** of a simple seed-growth algorithm for constructive partitioning:

1. ***Start a new partition with a seed logic cell.***
2. ***Consider all the logic cells that are not yet in a partition. Select each of these logic cells in turn.***
3. ***Calculate a gain function, g(m) , that measures the benefit of adding logic cell m to the current partition. One measure of gain is the number of connections between logic cell m and the current partition.***
4. ***Add the logic cell with the highest gain g(m) to the current partition.***
5. ***Repeat the process from step 2. If you reach the limit of logic cells in a partition, start again at step 1.***

Constructive Partitioning

- **Seed Logic cell:**

  The **logic cell with the most nets** is a good choice as the seed logic cell.

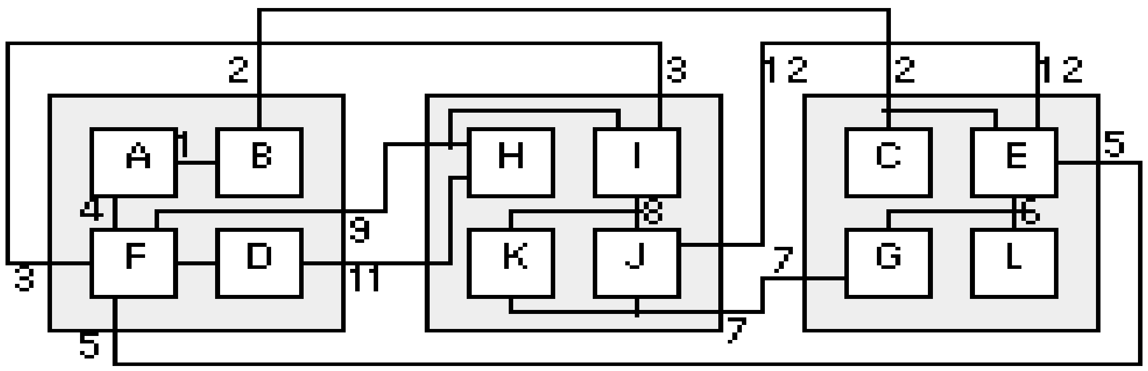
- **Cluster:**

  A set of seed logic cells known as a cluster. Called as *clique —borrowed from graph* theory.

- Clique:

  A clique of a graph is a subset of nodes where each pair of nodes is connected by an edge

Constructive Partitioning



- A constructed partition using logic cell C as a seed. It is difficult to get from this local minimum, with seven external connections (2, 3, 5, 7, 9,11,12), to the optimum solution ofb.

Improvement in Partitioning

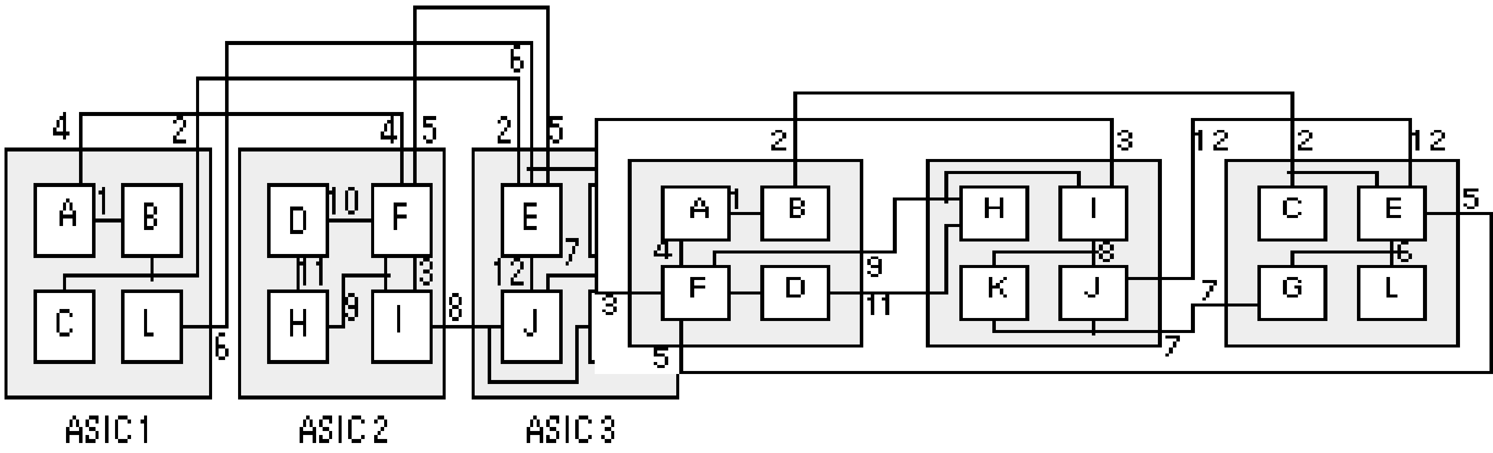


Fig 1 with 5 external connections **Fig 2** with 7 external connections

- To get from the solution shown in Fig 2 to the solution of Fig 1, which has a minimum number of external connections, requires a complicated swap.
- The three pairs: D and F, J and K, C and L need to be swapped

Iterative Partitioning Improvement

Algorithm based on Interchange method and group migration method

Interchange method (swapping a single logic cell):

If the swap improves the partition, accept the trail interchange otherwiseselect a new set of logic cells to swap.

Example: Greedy Algorithm –

It considers only one change

-Rejects it immediately if it is not an improvement.

-Accept the move only if it provides immediate benefit.

It is known as local minimum.

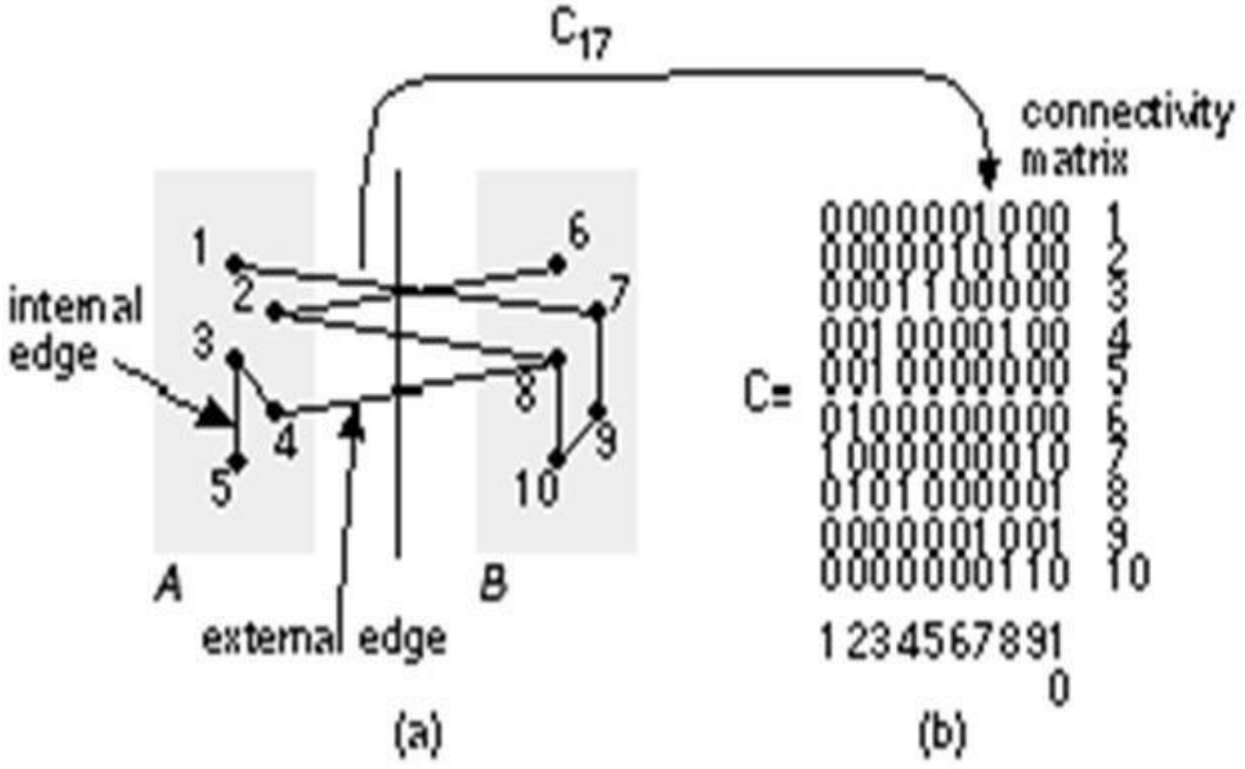
Group Migration (swapping a group of logic cell):

- Group migration consists of swapping groups of logic cells between partitions.
- The group migration algorithms –
  - **Adv:** better than simple interchange methods at improving a solution
  - **Disadv:** but are more complex.

Example: Kernighan – Lin Algorithm (K-L)

- Min cut Problem : Dividing a graph into two pieces, minimizing the nets(edges) that are cut

The Kernighan–Lin Algorithm



$$C=\begin{array}{cccccccccc|c}
0&0&0&0&0&0&1&0&0&0&1\\
0&0&0&0&0&1&0&1&0&0&2\\
0&0&0&1&1&0&0&0&0&0&3\\
0&0&1&0&0&0&0&1&0&0&4\\
0&0&1&0&0&0&0&0&0&0&5\\
0&1&0&0&0&0&0&0&0&0&6\\
1&0&0&0&0&0&0&0&1&0&7\\
0&1&0&1&0&0&0&0&0&1&8\\
0&0&0&0&0&0&1&0&0&1&9\\
0&0&0&0&0&0&0&1&1&0&10\\
\hline
1&2&3&4&5&6&7&8&9&10&
\end{array}$$

The Kernighan–Lin Algorithm (contd.,)

- Total external cost, cut cost, cut weight

$$W = \sum_{a \in A, b \in B} c_{ab}$$

- External edge cost
- Internal edge cost

$$E_a = \sum_{y \in B} c_{ay}$$

$$I_a = \sum_{z \in A} c_{az}$$

- Gain $g = D_a + D_b - 2C_{ab}$

where

$$D_a = E_a - I_a$$

The Kernighan–Lin Algorithm (contd.,)

- The K–L algorithm **finds a group of node pairs to swap that increases the gain** even though swapping individual node pairs from that group might decrease the gain.

- The steps of K-L algorithm are:

1. Find two nodes, a_i from A, and b_i from B, so that the gain from swapping them is a maximum. The gain g_i is $g_i = D_{a_i} + D_{b_i} - 2C_{a_i b_i}$

2. Next pretend swap ai and bi even if the gain g_i is zero or negative, and do not consider ai and bi eligible for being swapped again.

3. Repeat steps 1 and 2 a total of m times until all the nodes of A and B have been pretend swapped. We are back where we started, but we have ordered pairs of nodes in A and B according to the gain from interchanging those pairs.

The Kernighan–Lin Algorithm (contd.,)

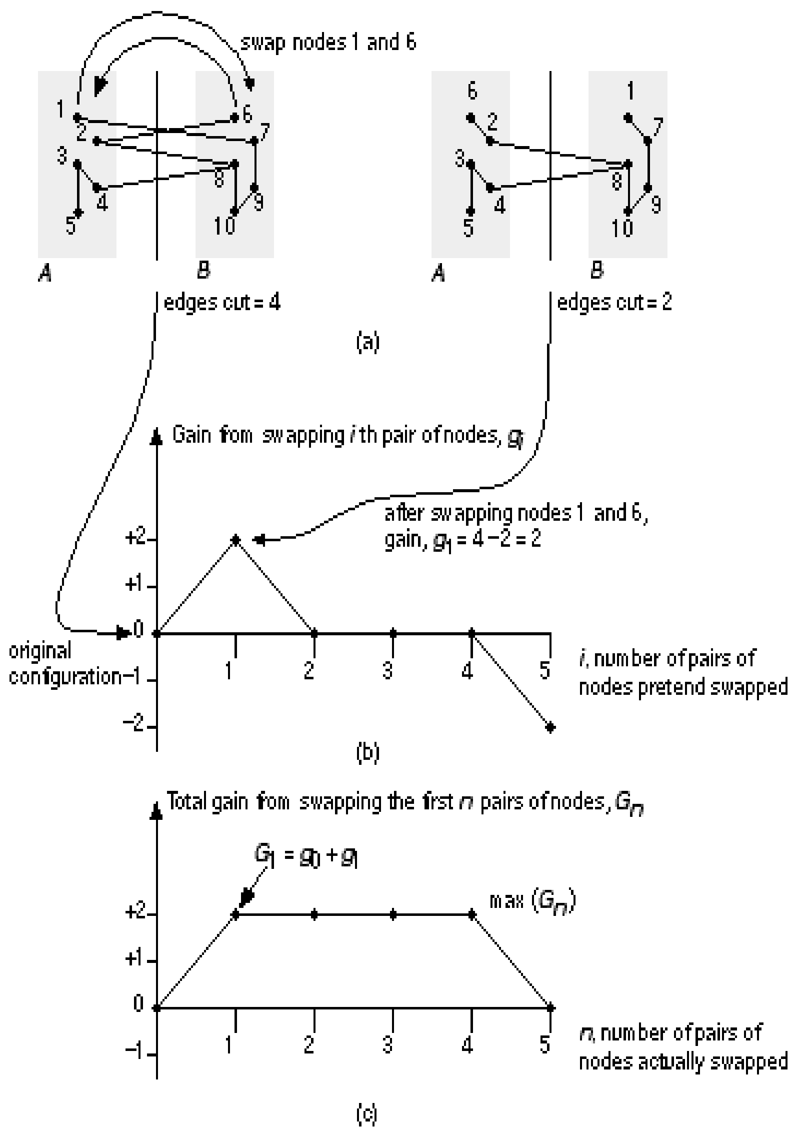
4. Now we can choose which nodes we shall actually swap. Suppose we only swap the first n pairs of nodes that we found in the preceding process. In other words we swap nodes X = a1, a2, &…., an from A with nodes Y = b1, b2,&…..,bn from B.

   The total gain would be,

$$G_n = \sum_{i=1}^{n} g_i$$

5. We now choose n corresponding to the maximum value of G_n

- If the maximum value of $G_n > 0$*, then swap the sets of nodes X and Y and thus* reduce the cut weight by G_n .
- *Use this new partitioning to start the process again* at the first step.
- If the maximum value of $G_n = 0$*, then we cannot improve the current* partitioning and we stop.
- We have found a locally optimum solution.


swap nodes 1 and 6
1
2
3
4
5
6
7
8
9
10
A
B
edges cut = 4
edges cut = 2
(a)
Gain from swapping ith pair of nodes, g_i
after swapping nodes 1 and 6, gain, $g_1 = 4 - 2 = 2$
+2
+1
0
original configuration
−1
−2
i, number of pairs of nodes pretend swapped
(b)
Total gain from swapping the first n pairs of nodes, G_n
$G_1 = g_0 + g_1$
max (G_n)
n, number of pairs of nodes actually swapped
(c)

- FIGURE 15.9
  Partitioning a graph using the Kernighan–Lin algorithm.

- (a) Shows how swapping node 1 of partition A with node 6 of partition B results in a gain of g = 2.

- (b) A graph of the gain resulting from swapping pairs of nodes.

- (c) The total gain is equal to the sum of the gains obtained at each step.

Simulated Annealing

- **Takes an existing solution** and then makes successive changes in a **series of random moves**.
- Each move is accepted or rejected based on an **energy function**.
- In the **Interchange method**,
    - **Accept the new trial configuration** only if the **energy function decreases**, which means the new configuration is an improvement
- But in the **simulated Annealing,**
    - **Accept the new configuration** even if the **energy function increases** for the new configuration—which means things are getting worse.
    - The probability of **accepting a worse configuration** is controlled by the **exponential expression exp(–ΔE / T)**,

where, Δ **E** - the resulting increase in the energy function.

T - a variable that can be controlled and corresponds to the temperature in the annealing of a metal cooling (this is why the process is called simulated annealing).

Simulated Annealing

- A parameter that relates the temperatures, T_i and T_{i+1}, at the i th and i + 1 th iteration:

$$T_{i+1} = \alpha T_i .$$

- As the temperature is slowly decreased, the probability of making moves that increase the energy function gets decreased.
- **Cooling schedule –** The critical parameter of the simulated-annealing algorithm is the rate at which the temperature T is reduced.
- Finally, as the **temperature approaches zero**, refuse to make any moves that increase the energy of the system and the system falls and comes to rest at the **nearest local minimum**.
- The **minimums of the energy function** correspond to possible solutions.
- The best solution is the **global minimum**.

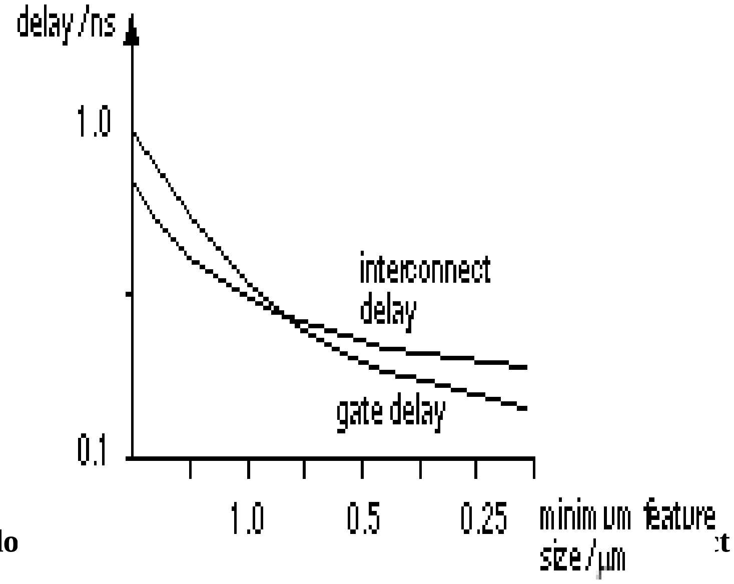
Simulated Annealing

- **Requirement of Simulated Annealing:**
- To find a good solution, a local minimum close to the global minimum, requires a **high initial temperature and a slow cooling schedule**.
- **Disadvantage:**
- This results in many trial moves and very long computer run time.(it gives optimum)
- **Advantage:**
- To solve **large graph problems**
- **Hill climbing-** Accept moves that seemingly take us away from a desirable solution to allow the system to **escape from a local minimum** and find other, better, solutions.

FLOORPLANNING

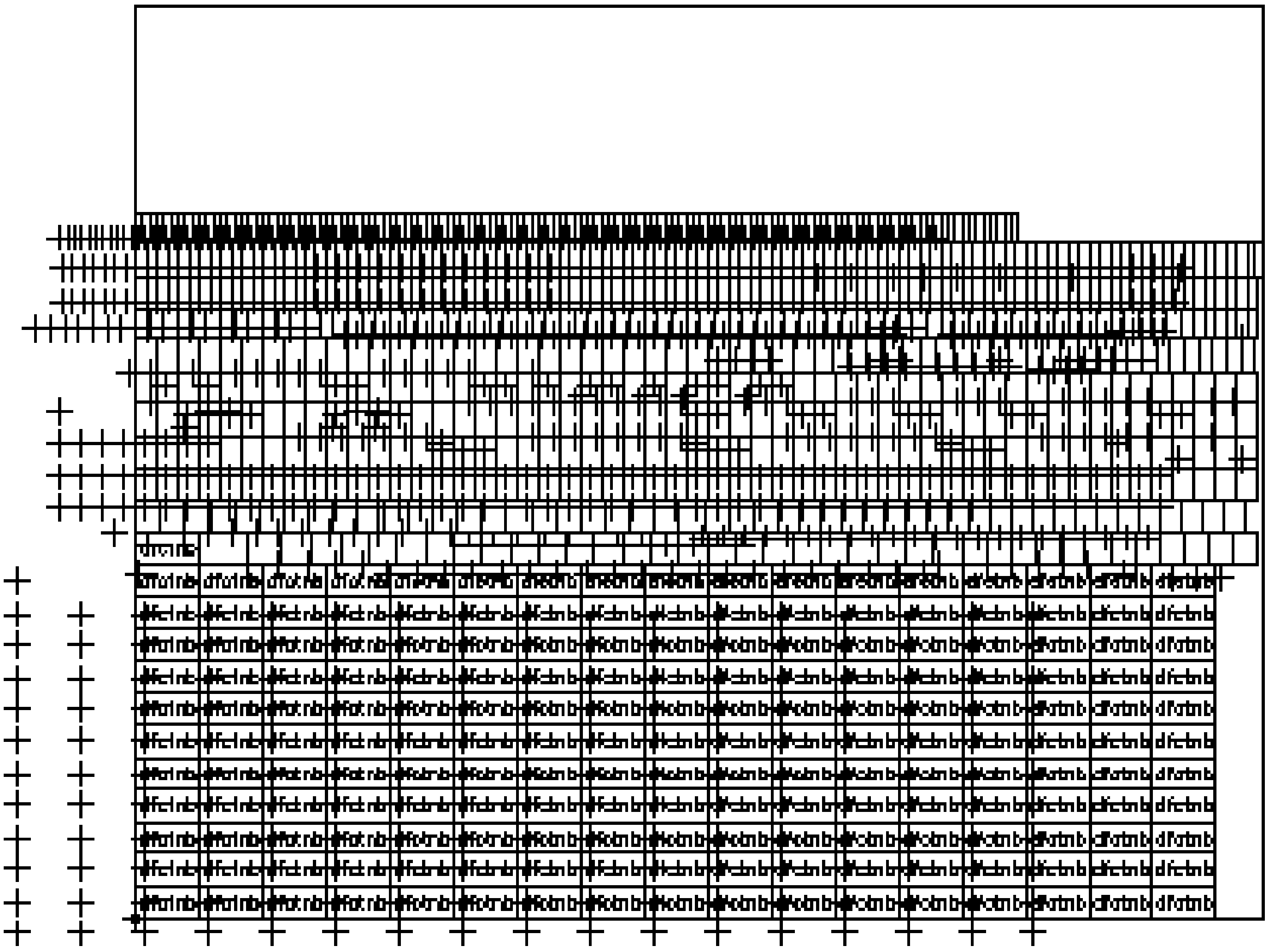
Introduction

- The input to the floorplanning step - **output of system partitioning and design entry—a netlist.**
- Netlist - describing **circuit blocks, the logic cells within the blocks, and their connections.**



● FIGURE 16.3 Interconnect and gate delays. As feature sizes decrease, both average interconnect delay and average gate delay decrease—but at different rates. This is because interconnect capacitance tends to a limit that is independent of scaling. **Interconnect delay now dominates gate delay.**

● **Flo** **:t interconnect delay**



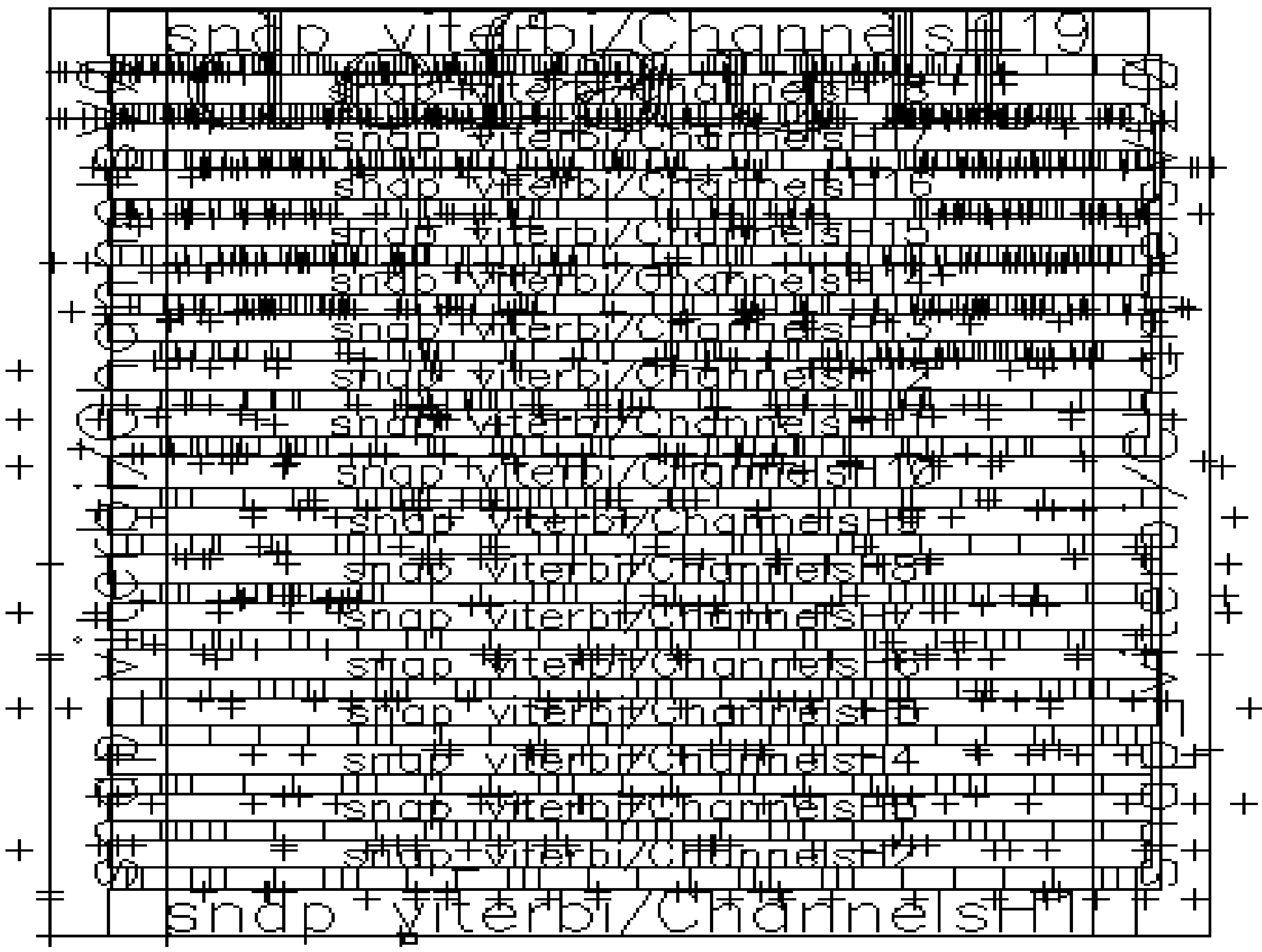
•The starting point of floorplaning and placement steps for the viterbi decoder

-collection of standard cells with no room set aside yet for

The starting point of floorplaning and placement steps for the viterbi decoder

- **Small boxes that look like bricks** - outlines of the standard cells.
- **Largest standard cells, at the bottom of the display** (labeled dfctnb)

  - 188 D flipflops.
- **'+' symbols** -drawing origins of the standard cells—for the D flip-flops they are shifted to the left and below the logic cell bottom left-hand corner.
- **Large box surrounding all the logic cells** - estimated chip size.
- (This is a screen shot from Cadence Cell Ensemble.)



The viterbi decoder after floorplanning and placement

The viterbi decoder after floorplanning and placement

- **8 rows of standard cells separated by 17 horizontal channels** (labeled 2–18).

- **Channels are routed as numbered**.

- In this example, the I/O pads are omitted to show the cell placement more clearly.

Floorplanning Goals and Objectives

- The input to a floorplanning tool is a hierarchical **netlist** that describes
  - the interconnection of the blocks (RAM, ROM, ALU, cache controller, and so on)
  - the logic cells (NAND, NOR, D flip-flop, and so on) within the blocks
  - the logic cell connectors (terminals , pins , or ports)

- The netlist is a **logical description** of the ASIC;
- The floorplan is a physical **description** of an ASIC.
- Floorplanning is a **mapping between the logical description (the netlist) and the physical description (the floorplan).**

The **Goals of Floorplanning** are to:

- Arrange the blocks on a chip,
- Decide the location of the I/O pads,
- Decide the location and number of the power pads,
- Decide the type of power distribution, and
- Decide the location and type of clock distribution.

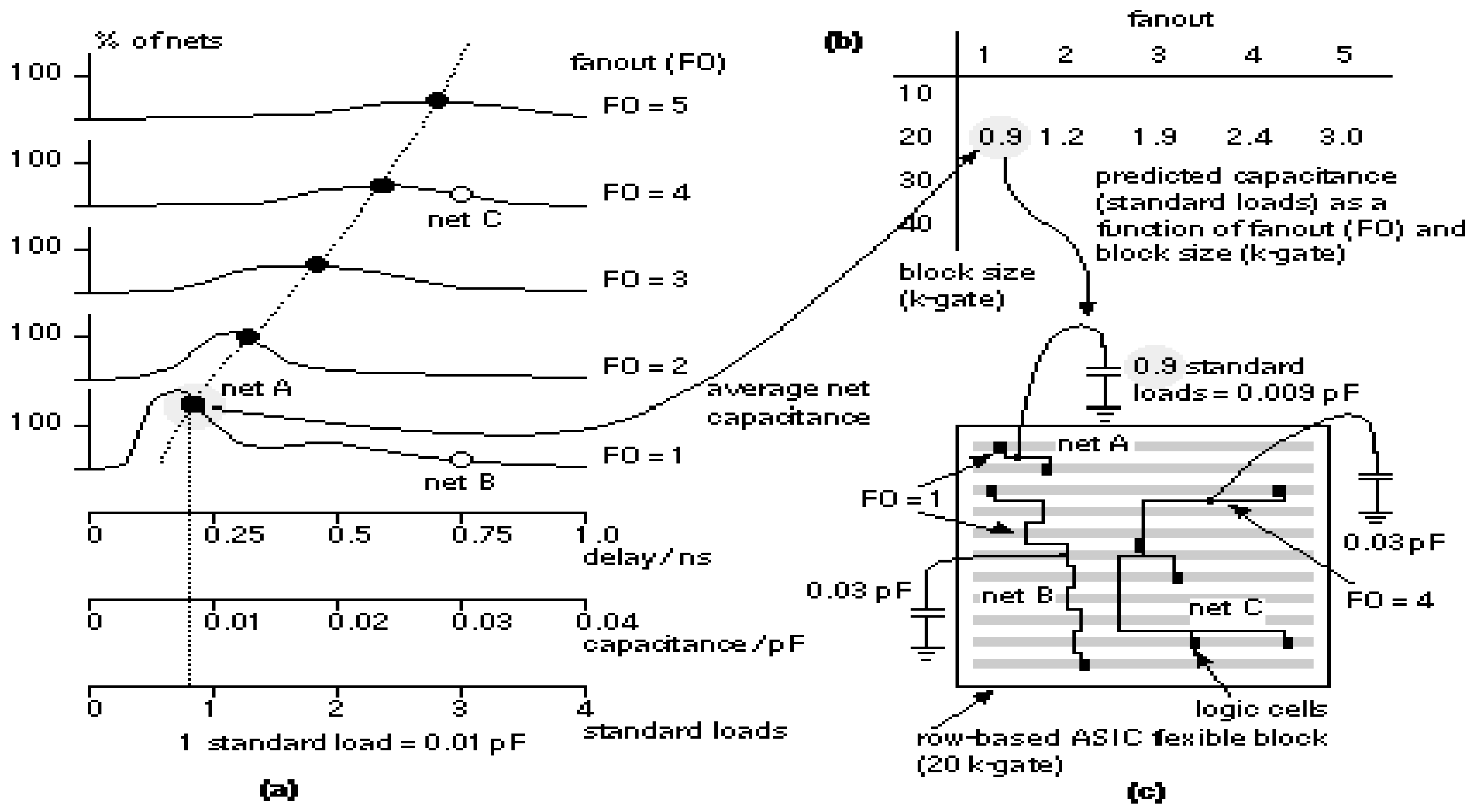
Objectives of Floorplanning –

To minimize the chip area

To minimize delay.

Measuring area is straightforward, but measuring delay is more difficult

Measurement of Delay in Floor planning



•FIGURE 16.4 Predicted capacitance. (a) Interconnect lengths as a function of fanout (FO) and circuit-block size. (b) Wire-load table. There is only one capacitance value for each fanout (typically the average value).

(c) The wire-load table predicts the capacitance and delay of a net (with a considerable error). Net A and net B both have a fanout of 1, both have the same predicted net delay, but net B in fact has a much greater delay than net A in the actual layout (of course we shall not know what the actual layout is until much later in the design process).

Measurement of Delay in Floor planning (contd.,)

- A floorplanning tool can use **predicted-capacitance tables (also known as interconnect-load tables or wire-load tables)**.
- Typically between **60 and 70 percent of nets have a FO = 1**.
- **The distribution for a FO = 1 has a very long tail**, stretching to interconnects that run from corner to corner of the chip.
- **The distribution for a FO = 1 often has two peaks**, corresponding to a distribution for close neighbors in subgroups within a block, superimposed on a distribution corresponding to routing between subgroups.

Measurement of Delay in Floor planning (contd.,)

- We often see a **twin-peaked distribution** at the chip level also, corresponding to separate distributions for **interblock routing (inside blocks) and intrablock routing (between blocks)**.
- The distributions for **FO > 1 are more symmetrical and flatter** than for FO = 1.
- **The wire-load tables can only contain one number**, for example the

  average net capacitance, for any one distribution.
- Many tools take a worst-case approach and use the 80- or 90-percentile point instead of the average. Thus a tool may **use a predicted capacitance** for which we know **90 percent of the nets will have less than the estimated**

  capacitance.

Measurement of Delay in Floor planning (contd.,)

- **Repeat the statistical analysis for blocks with different sizes**.

  For example, a net with a FO = 1 in a 25 k-gate block will have a different (larger) average length than if the net were in a 5 k-gate block.

- **The statistics depend on the shape (aspect ratio) of the block**

  (usually the statistics are only calculated for **square blocks**).

- The statistics will also **depend on the type of netlist**.

  For example, the distributions will be different for a netlist generated by setting a constraint for **minimum logic delay during synthesis**—which tends to generate large numbers of two-input NAND gates—than for netlists generated using **minimum-area constraints**.

Floorplanning Tools

- **Floorplanning in CBIC:**

- **Flexible blocks (or variable blocks)** :
  - Their total area is fixed,
  - Their shape (aspect ratio) and connector locations may be adjusted during the placement.

- **Fixed blocks:**
  - The dimensions and connector locations of the other fixed blocks (perhaps RAM, ROM, compiled cells, or megacells) can only be modified when they are created.

- **Seeding:**
  - Force logic cells to be in selected flexible blocks by **seeding** . We choose seed cells by name.
  - Seeding may be **hard or soft**.
- **Hard seed** - fixed and not allowed to move during the remaining floor planning and placement steps.
- **Soft seed** - an initial suggestion only and can be altered if necessary by the floor planner.

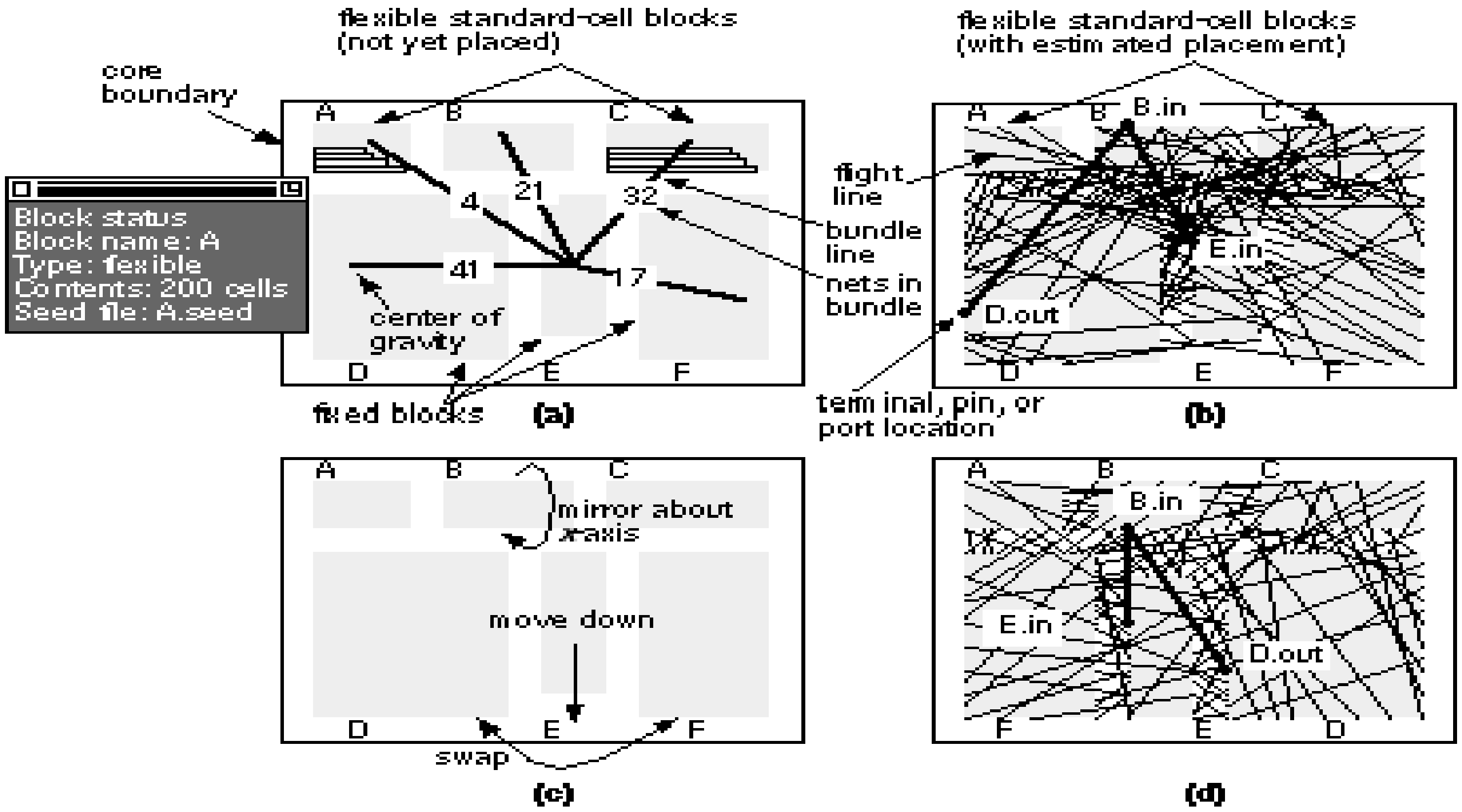
- **Seed connectors** within flexible blocks—forcing certain nets to appear in a specified order, or location at the boundary of a flexible block.

- **Rat's nest**:-display the connection between the blocks

- Connections are shown as **bundles** between the centers of blocks or as **flight lines** between connectors.

Floorplanning Tools

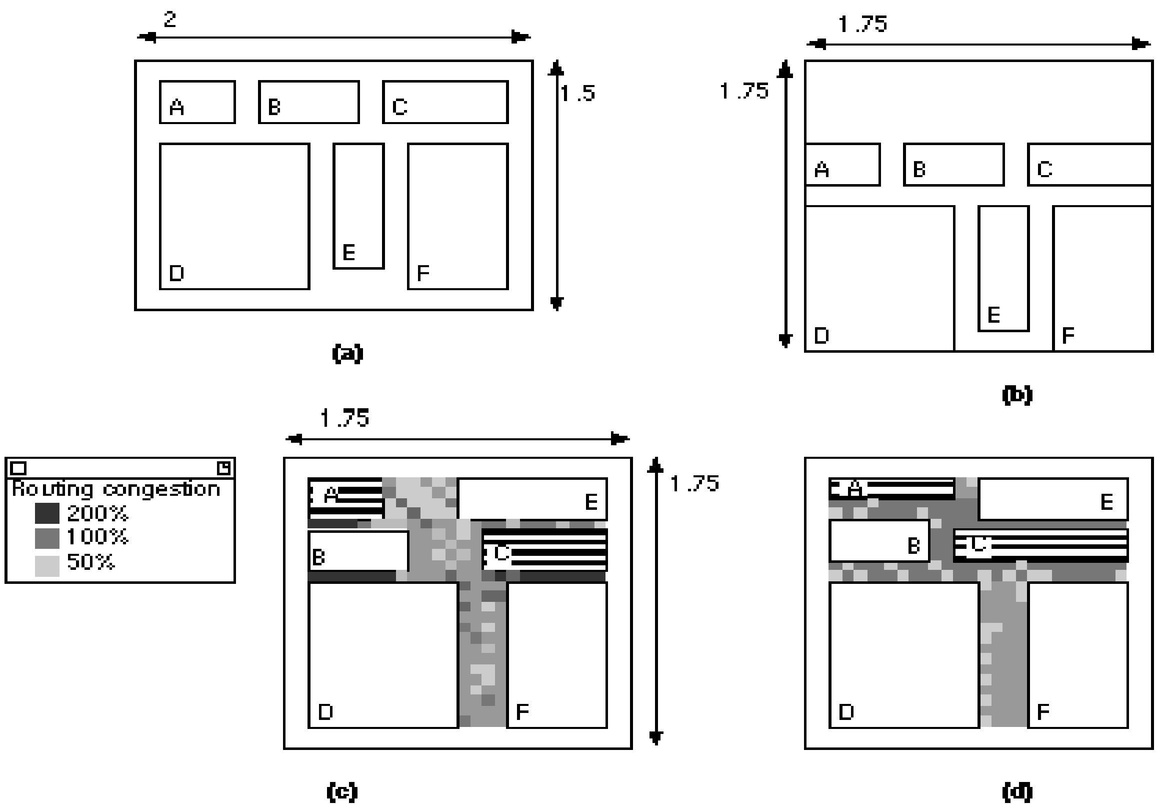
Floorplanning a cell-based ASIC.



(a) **Initial floorplan generated by the floorplanning tool.** Two of the blocks are flexible (A and C) and contain rows of standard cells (unplaced). A pop-up window shows the status of block A.
(b) **An estimated placement for flexible blocks A and C.** The connector positions are known and a **rat's nest display** shows the heavy congestion below block B.
(c) **Moving blocks to improve the floorplan**.
(d) **The updated display shows the reduced congestion** after the changes.

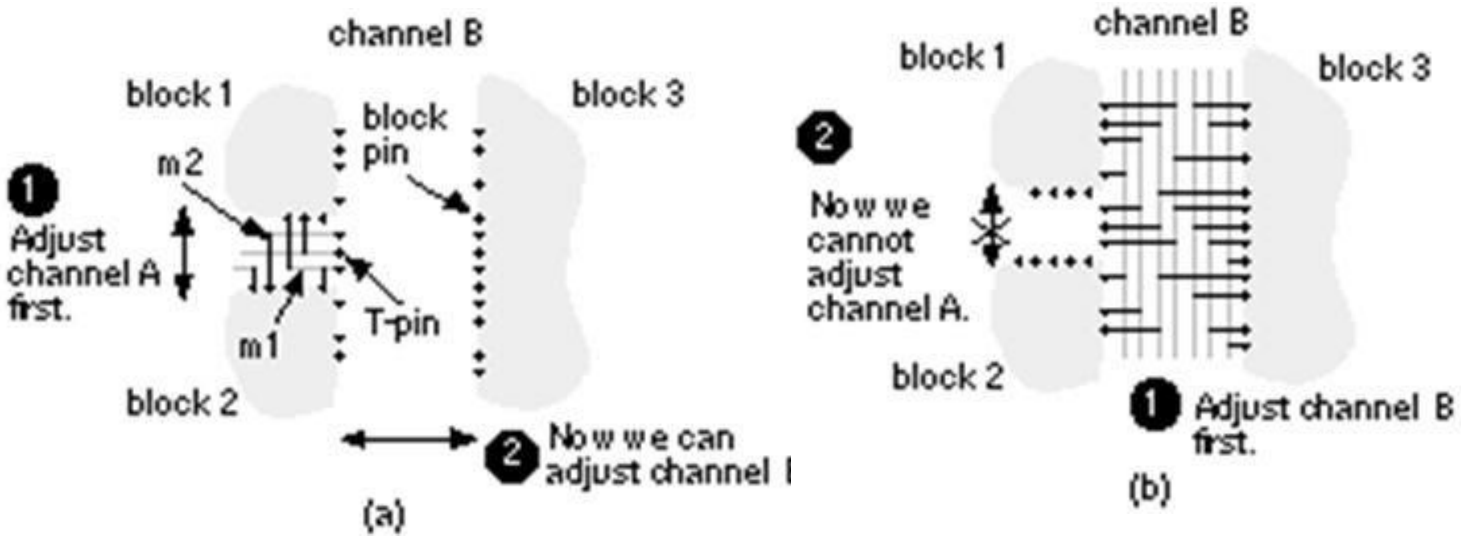
•

Aspect ratio and Congestion Analysis



(a) The initial floorplan with a 2:1.5 die aspect ratio.
(b) Altering the floorplan to give a 1:1 chip aspect ratio.
Congestion analysis-One measure of congestion is the difference between the number of interconnects that we actually need, called the channel density , and the channel capacity
(c) **A trial floorplan with a congestion map.** Blocks A and C have been placed so that we know the terminal positions in the channels. **Shading indicates the ratio of channel density to the channel capacity.** Dark areas
show regions that cannot be routed because the channel congestion exceeds the estimated capacity.
(d) Resizing flexible blocks Aand C alleviates congestion.

Channel Definition



• **Channel definition or channel allocation**

• **During the floorplanning step, assign the areas between blocks that are to be used for interconnect.**

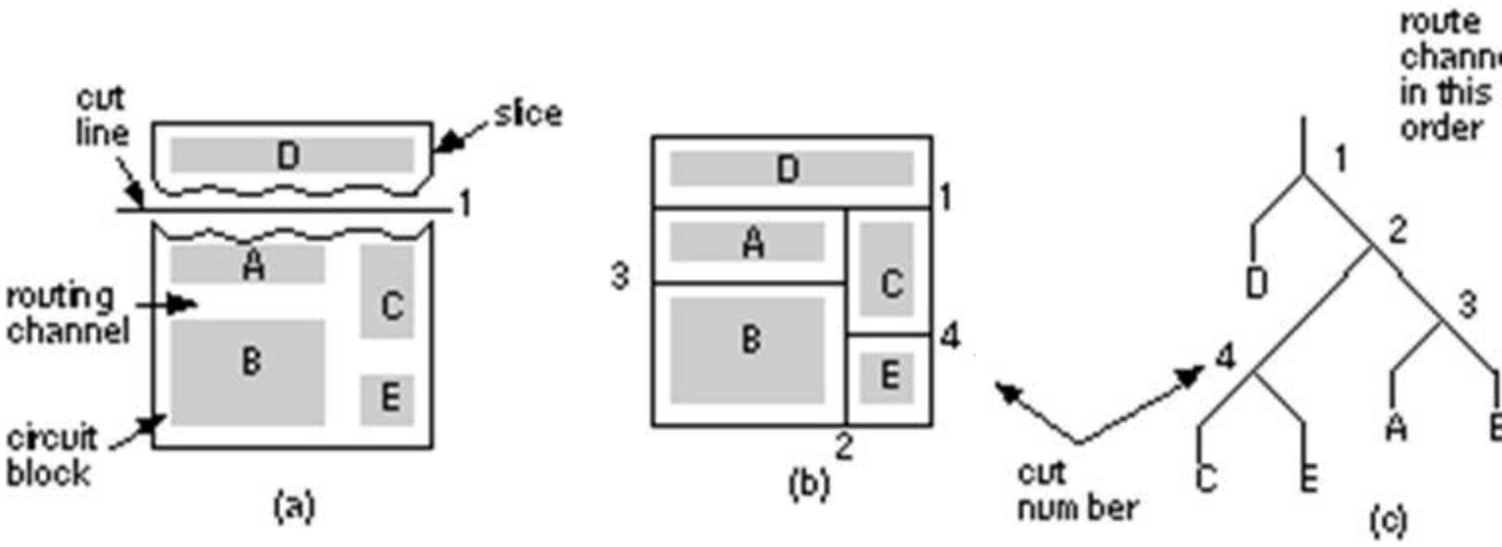
• **Routing a T-junction between two channels in two-level metal.**

• The dots represent logic cell pins.

• (a) Routing channel A (the stem of the T) first allows us to adjust the width of channel B. (b) If we route channel B first (the top of the T), this fixes the width of channel A.

• **Route the stem of a T-junction before route the top**.

Channel Routing



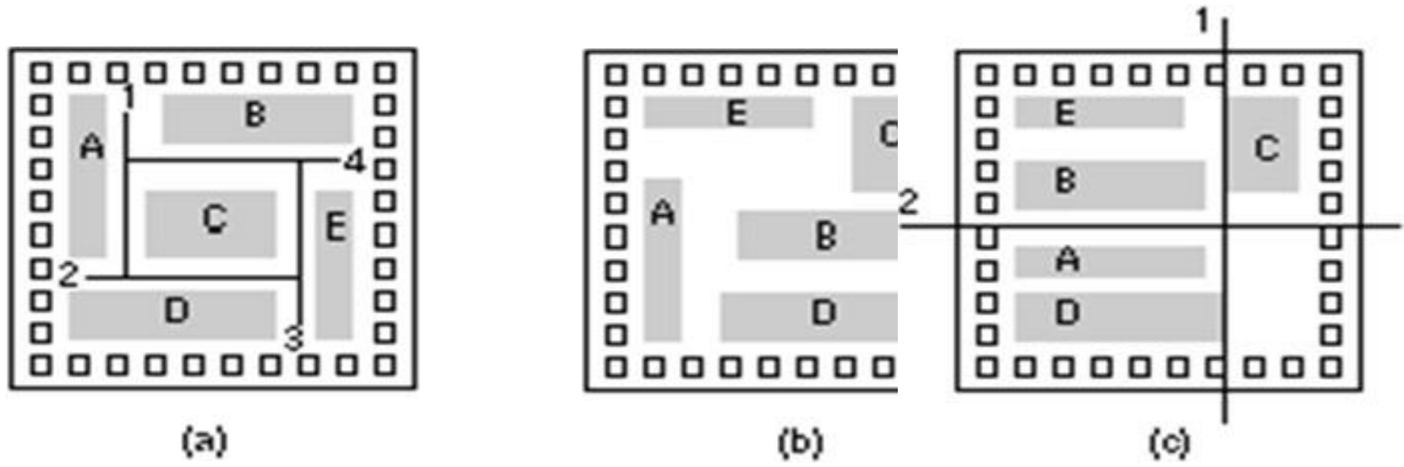
• **Defining the channel routing order for a slicing floorplan using a slicing tree.**

• (a) Make a cut all the way across the chip between circuit blocks. Continue slicing until each piece contains just one circuit block. Each cut divides a piece into two without cutting through a circuit block.

• (b) A sequence of cuts: 1, 2, 3, and 4 that successively slices the chip until only circuit blocks are left.

• (c) The slicing tree corresponding to the sequence of cuts gives the order in which to routethe channels: 4, 3, 2, and finally 1.

Cyclic Constraints

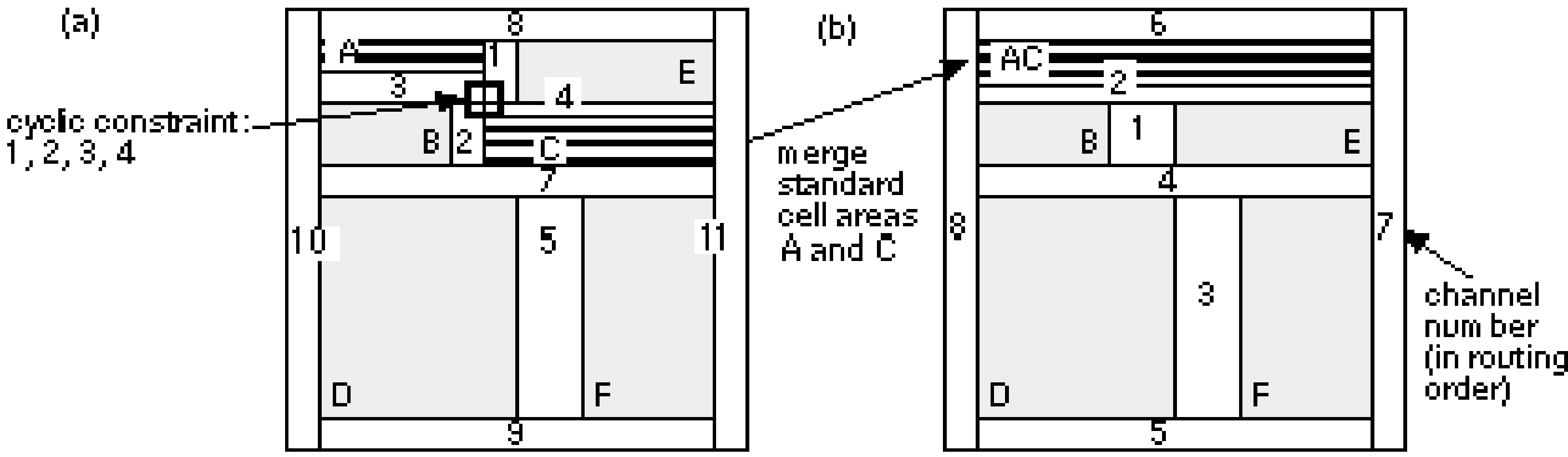


• **Cyclic constraints.**

• (a) A nonslicing floorplan with a cyclic constraint that prevents channel routing.
(b) In this case it is difficult to find a slicing floorplan without increasing the chip area.

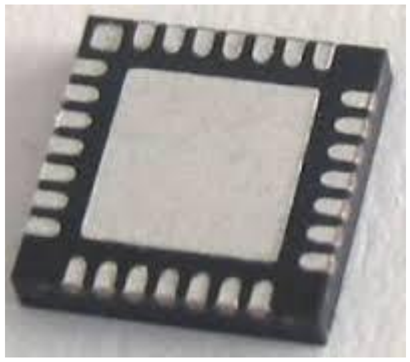
• (c) This floorplan may be sliced (with initial cuts 1 or 2) and has no cyclic constraints, but it is inefficient in area use and will be very difficult to route.

Cyclic Constraints

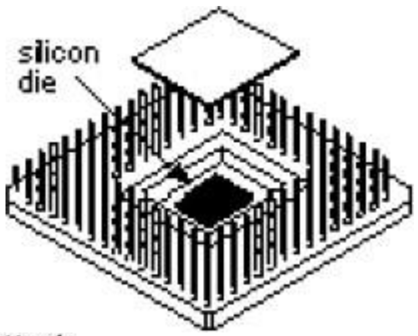

(a)
8
A
1
E
3
4
cyclic constraint : 1 , 2, 3, 4
B
2
C
7
10
5
11
D
F
9
(b)
6
AC
2
merge standard cell areas A and C
B
1
E
4
8
7
3
channel num ber (in routing order)
D
F
5

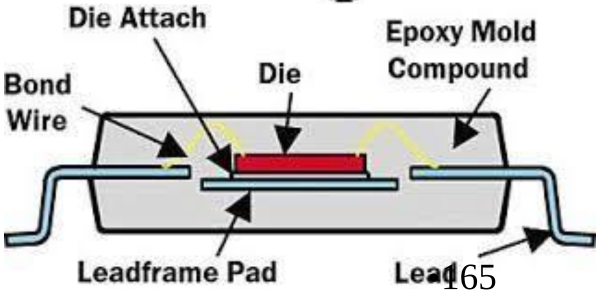
- (a) We can eliminate the cyclic constraint by merging the blocks A and C.
- (b) A slicing structure.

I/O and Power Planning (contd.,)

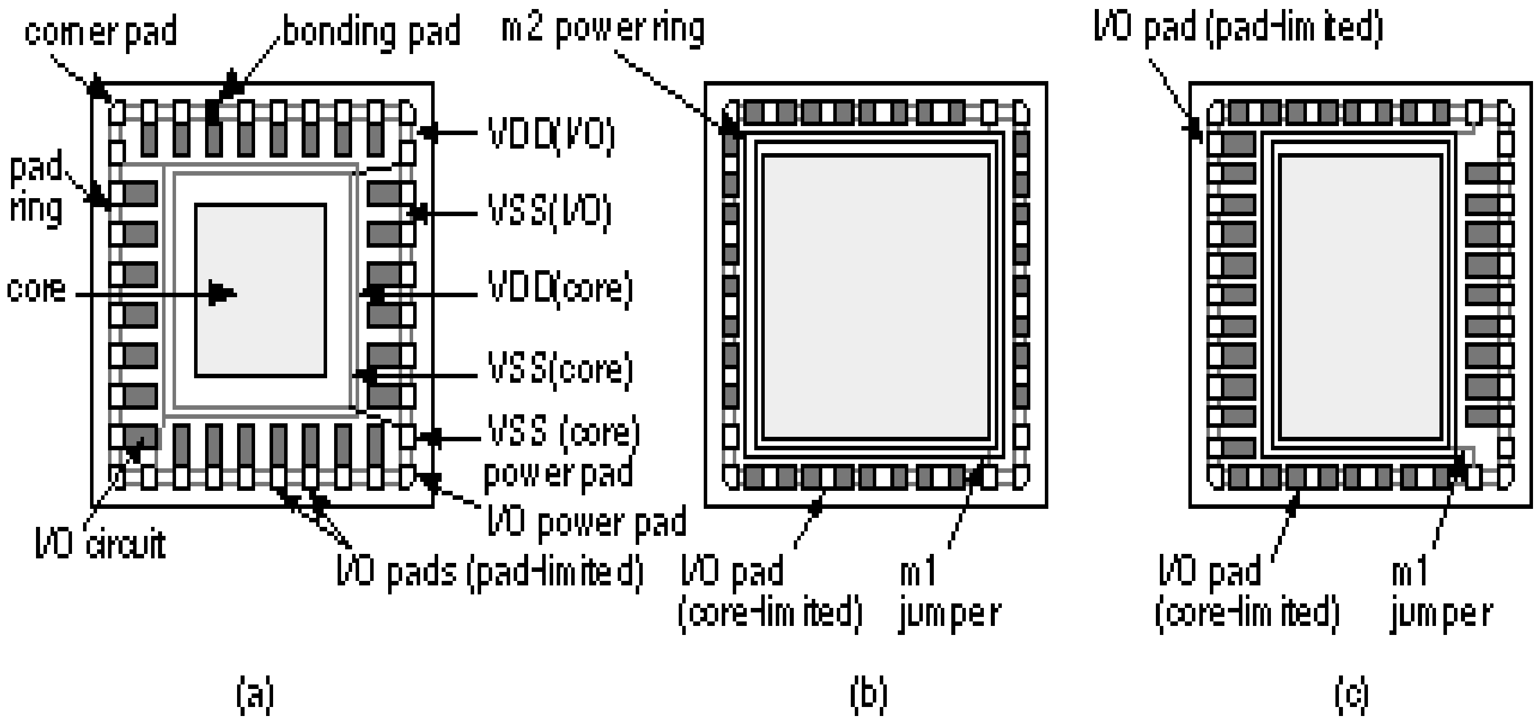


- Every chip communicates with the outside world. Signals flow onto and off the chip and we need to supply power.
- The I/O and power constraints has to considered early in the floorplanning process.
- A silicon chip or die (plural die, dies, or dice) is mounted on a chip carrier inside a chip package .
- Connections are made by bonding the chip pads to fingers on a metal lead frame that is part of the package.
- The metal lead-frame fingers connect to the package pins . A die consists of a logic core inside a pad ring .





I/O and Power Planning



• FIGURE 16.12 Pad-limited and core-limited die.

• (a) **A pad-limited die.** Tall, thin pad-limited pads, which maximize the number of pads we can fit around the outside of the chip. The number of pads determines the die size.

• (b) **A core-limited die**: short, wide core-limited pads.The core logic determines the die size.

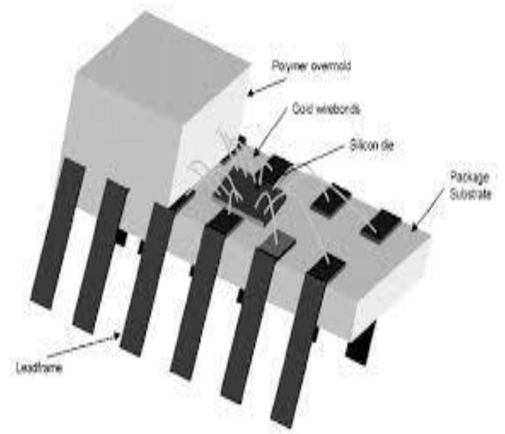
• (c) Using **both pad-limited pads and core-limited pads**

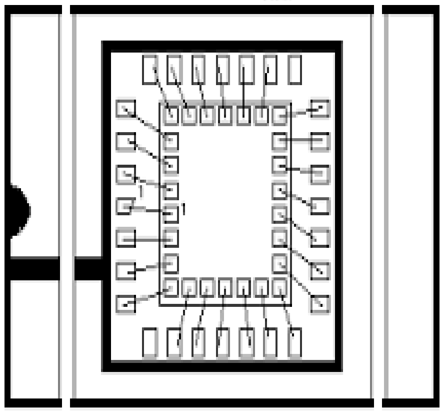
I/O and Power Planning (contd.,)

- **Power Pads and I/O Pads:**
- Special **power pads** are used for:1. positive supply, or VDD, power buses
  (or power rails) and
  2. ground or negative supply, VSS or GND.
- One set of VDD/VSS pads supplies power to the **I/O pads** only.
- Another set of VDD/VSS pads connects to a second power ring that supplies the **logic core.**
- The I/O power is a **dirty power** since it has to supply large transient currents to the output transistors. We keep dirty power separate to avoid **injecting noise into the internal-logic power** (the **clean power**).
- I/O pads also contain **special circuits to protect against electrostatic discharge (ESD)**. These circuits can withstand very short high-voltage (several kilovolt) pulses that can be generated during human or machine handling.

I/O and Power Planning (contd.,)

- **Pad Seed:** Fix the position of the chip pad for down bonding
- **Making an electrical connection between the substrate and a chip pad, or to a package pin**, it must be to VDD (n -type substrate) or VSS (p -type substrate). This **substrate connection** (for the whole chip) employs a **down bond** (or drop bond) to the carrier. We have several options:
    - ➢ *Dedicate one (or more) chip pad(s) to down bond to the chip carrier.*
    - ➢ *Make a connection from a chip pad to the lead frame and down bond from the chip pad to the chip carrier.*
    - ➢ *Make a connection from a chip pad to the lead frame and down bond from the lead frame.*
    - ➢ *Down bond from the lead frame without using a chip pad.*
    - ➢ *Leave the substrate and/or chip carrier unconnected.*
- Depending on the package design, the type and positioning of down bonds may be fixed.
- This means we need **to fix the position of the chip pad for down bonding using a pad seed**




Example of good bondingdiagram of 26 pins chip in 28-ld DIL

I/O and Power Planning (contd.,)

- A **double bond** connects two pads to one chip-carrier finger and one package pin. We can do this **to save package pins or reduce the series inductance of bond wires** (typically a few nanohenries) by parallel connection of the pads.

- **Multiple VDD and VSS pads:**
- To reduce the series resistive and inductive impedance of power supply networks, it is normal to **use multiple VDD and VSS pads.**

- This is particularly important with the **simultaneously switching outputs (SSOs)** that occur when driving buses .

The **output pads can easily I/O and Power Planning (contd.,)**

- **consume most of the power** on a CMOS ASIC, because the load on a pad (usually tens of picofarads) is much larger than typical on-chip capacitive loads.

- Depending on the technology it may be necessary to **provide dedicated VDD and VSS pads for every few SSOs.** Design rules set how many SSOs can be used per VDD/VSS pad pair. These dedicated VDD/VSS pads must "follow" groups of output pads as they are seeded or planned on the floorplan.

- Using a **pad mapping** we translate the **logical pad** in a netlist to a **physical pad** from a pad library . We might control pad seeding and mapping in the floorplanner.

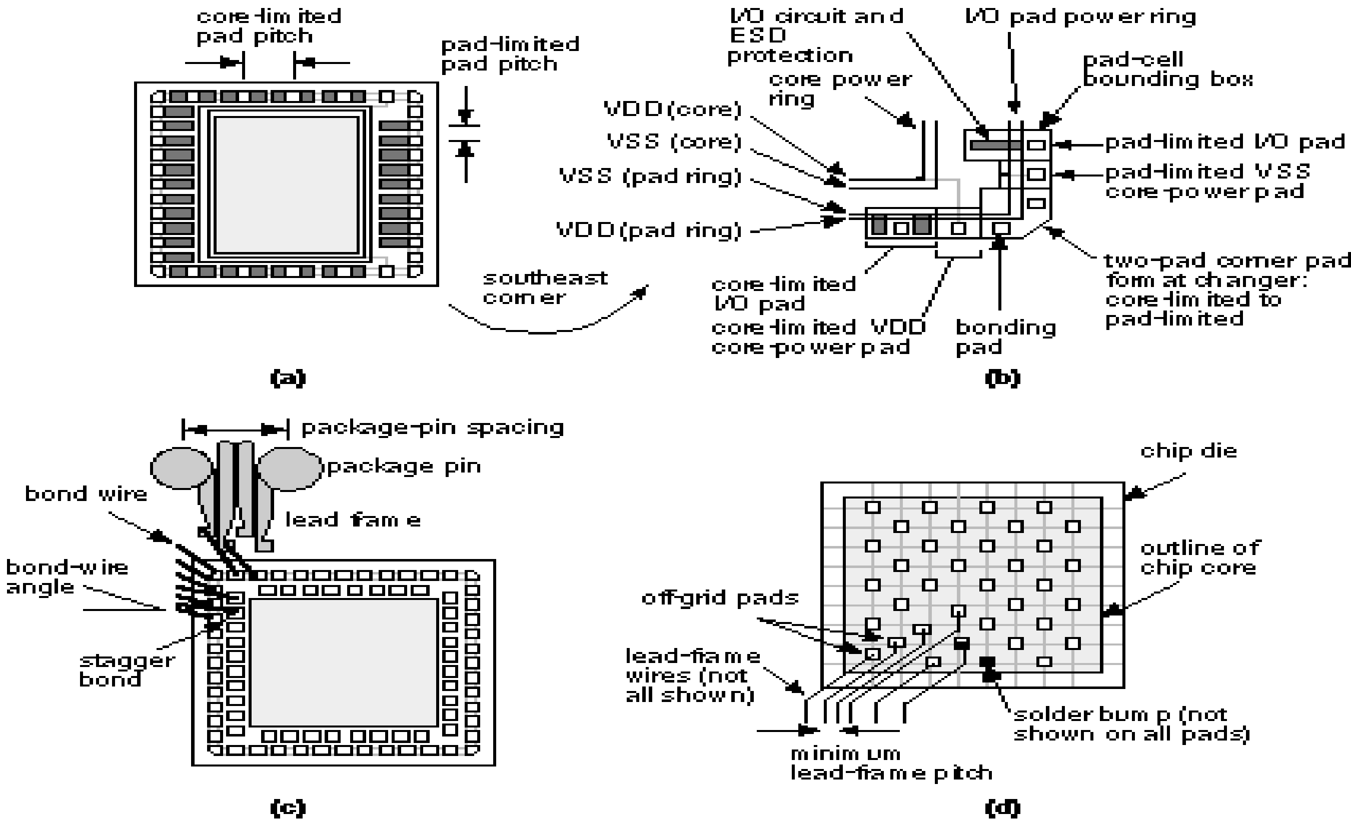
- **Handling of I/O pads can become quite complex**; there are several nonobvious factors that must be considered when generating a pad ring:

  - **Design library pad cells for one orientation**.
    - For example, an **edge pad** for the south side of the chip, and a **corner pad** for the southeast corner.
  - We could then generate other orientations by rotation and flipping (mirroring). Some ASIC vendors will not allow rotation or mirroring of logic cells in the mask file.
  - To avoid these problems we may need to have separate horizontal, vertical, left-handed, and right-handed pad cells in the library with appropriate logical to physical pad mappings.

  - If we **mix pad-limited and core-limited edge pads** in the same pad ring, this complicates the design of corner pads. Usually the two types of edge pad cannot abut. In this case a **corner pad also becomes a pad-format changer , or hybrid corner pad .**

  - In single-supply chips we have one VDD net and one VSS net, both global power nets . It is also possible to **use mixed power supplies (for example, 3.3 V and 5 V) or multiple power supplies (digital VDD, analog VDD)**.

I/O and Power Planning (contd.,)



•FIGURE 16.13 Bonding pads. (a) This chip uses both pad-limited and core-limited pads. (b) A hybrid corner pad. (c) A chip with stagger-bonded pads. (d) An area- bump bonded chip (or flip-chip). The chip is turned upside down and solder bumps

connect the pads to the lead frame

I/O and Power Planning (contd.,)

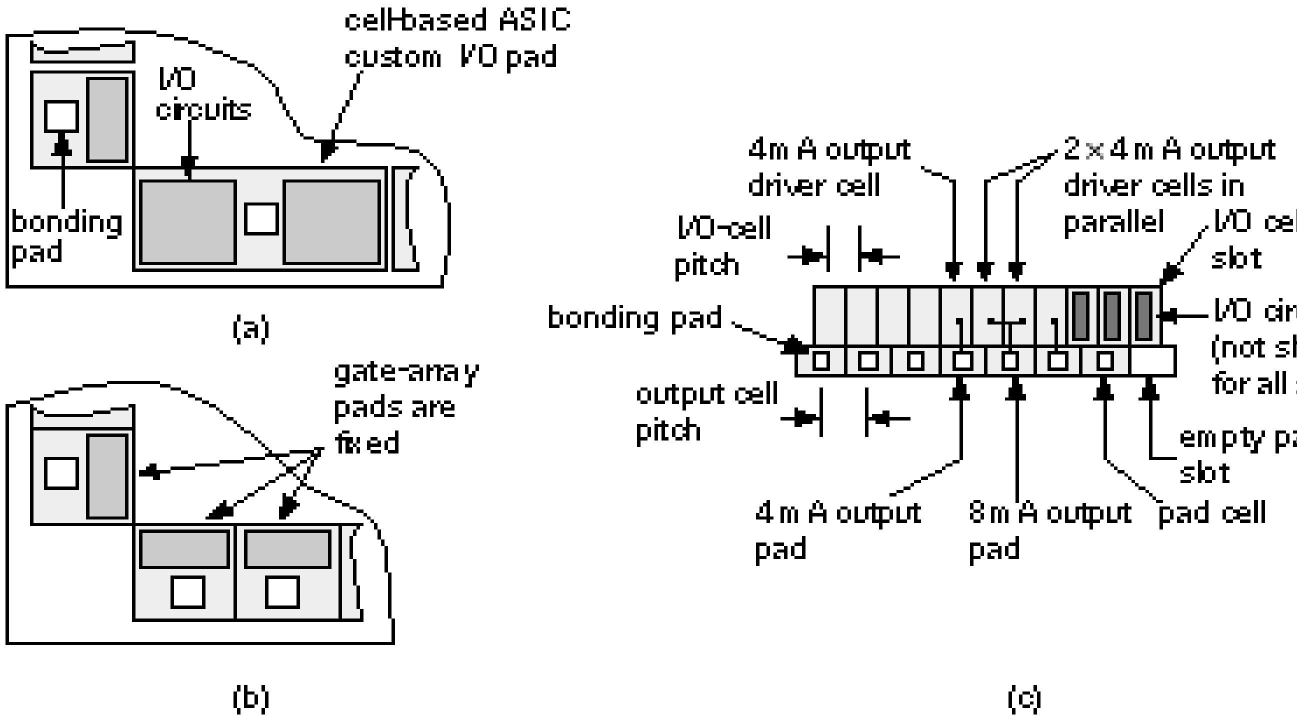
- **Other I/O bond:**
- **stagger-bond** arrangement using two rows of I/O pads. In this case the design rules for bond wires (the spacing and the angle at which the bond wires leave the pads) become very important.

- **Area-bump** bonding arrangement (also known as flip-chip, solder-bump) used, for example, with **ball-grid array (BGA)** packages.

- Even though the bonding pads are located in the center of the chip, the I/O circuits are still often located at the edges of the chip because of difficulties in power supply distribution and integrating I/O circuits together with logic in the center of the die.

- **I/O pads in MGA and CBIC:**
- In an **MGA** the pad spacing and I/O-cell spacing is fixed—each pad occupies a fixed pad slot (or pad site). This means that the properties of the pad I/O are also fixed but, if we need to, we can parallel adjacent output cells to increase the drive. To increase flexibility further the I/O cells can use a separation, the I/O-cell pitch , that is smaller than the pad pitch .

I/O and Power Planning (contd.,)



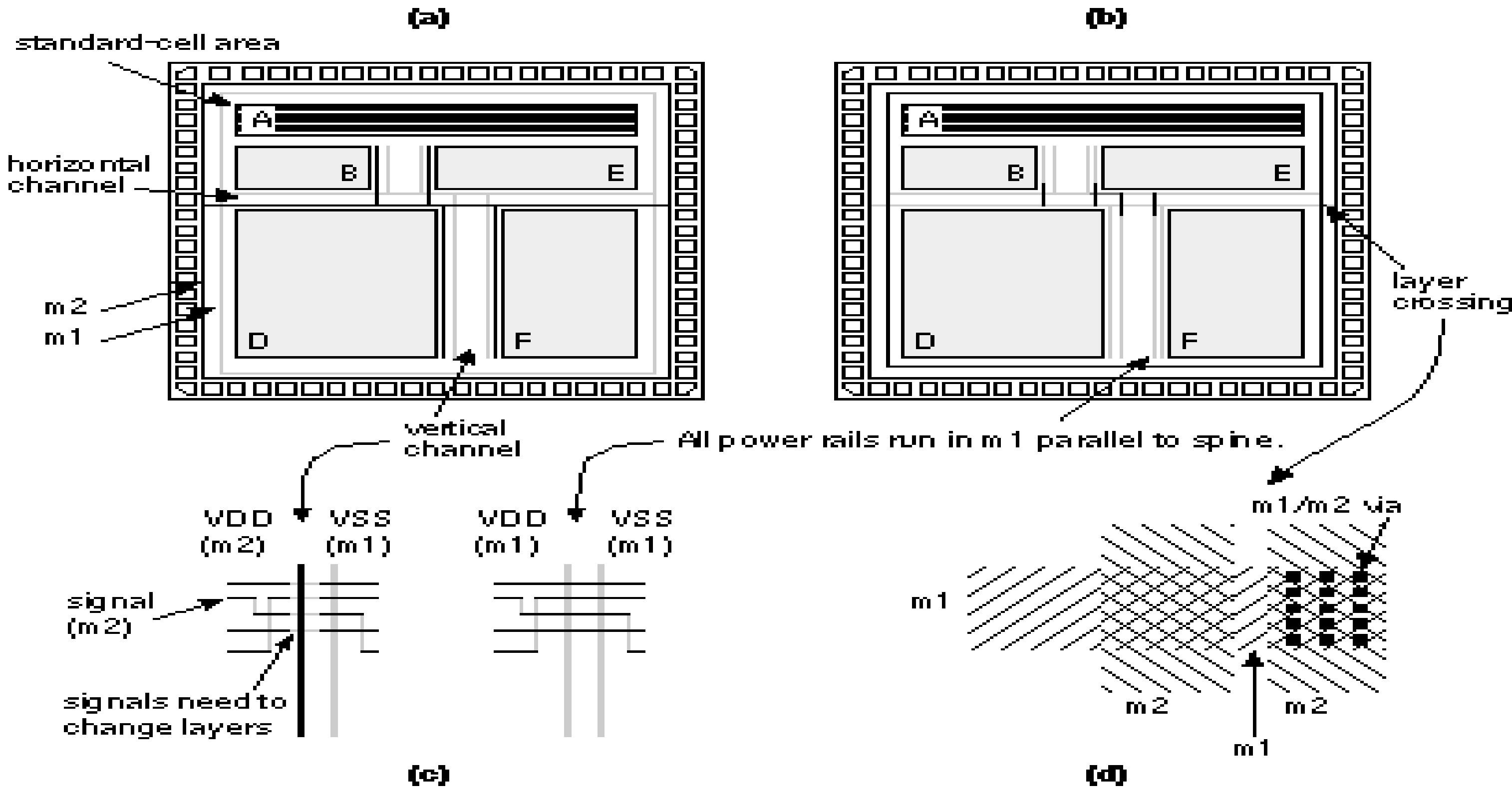
● FIGURE 16.14 Gate-array I/O pads. (a) Cell-based ASICs may contain pad cells of different sizes and widths. (b) A corner of a gate-array base. (c) A gate-array base with different I/O cell pad

pitches

I/O and Power Planning (contd.,)

- The long direction of a rectangular channel is the **channel spine .**
- Some automatic routers may require that metal lines parallel to a channel spine use a **preferred layer** (either m1, m2, or m3). Alternatively we say that a particular metal layer runs in a preferred direction .

I/O and Power Planning (contd.,)



• FIGURE 16.15 Power distribution. (a) Power distributed using m1 for VSS and m2 for VDD. This helps minimize the number of vias and layer crossings needed but causes problems in the routing channels.
(b) In this floorplan m1 is run parallel to the longest side of all channels, the channel spine. This can make automatic routing easier but may increase the number of vias and layer crossings. (c) An expanded view of part of a channel (interconnect is shown as lines). If power runs on different layers along the spine of a channel, this forces signals to change layers. (d) A closeup of VDD and VSS buses as they cross. Changing layers requires a large number of via contacts to reduce resistance.

Power distribution.

- (a) Power distributed using m1 for VSS and m2 for VDD.
  - This helps minimize the number of vias and layer crossings needed
  - but causes problems in the routing channels.

- (b) In this floorplan m1 is run parallel to the longest side of all channels, the channel spine.
  - This can make automatic routing easier
  - but may increase the number of vias and layer crossings.

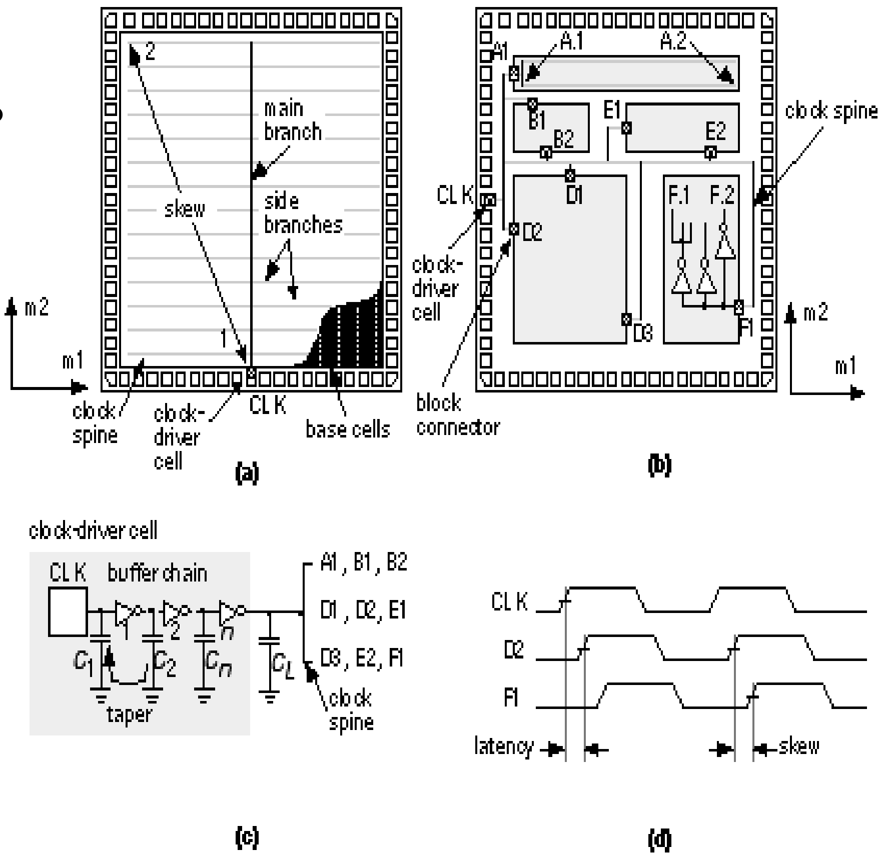
- (c) An expanded view of part of a channel (interconnect is shown as lines). If power runs on different layers along the spine of a channel, this forces signals to change layers.

- (d) A closeup of VDD and VSS buses as they cross. Changing layers requires a large number of via contacts to reduce resistance.

Clock Planning

- **clock spine** routing scheme with all clock pins driven directly from the clock driver. **MGAs and FPGAs** often use this **fish bone type of clock distribution** scheme

•



•FIGURE 16.16 Clock distribution.

•(a) A clock spine for a gate array.

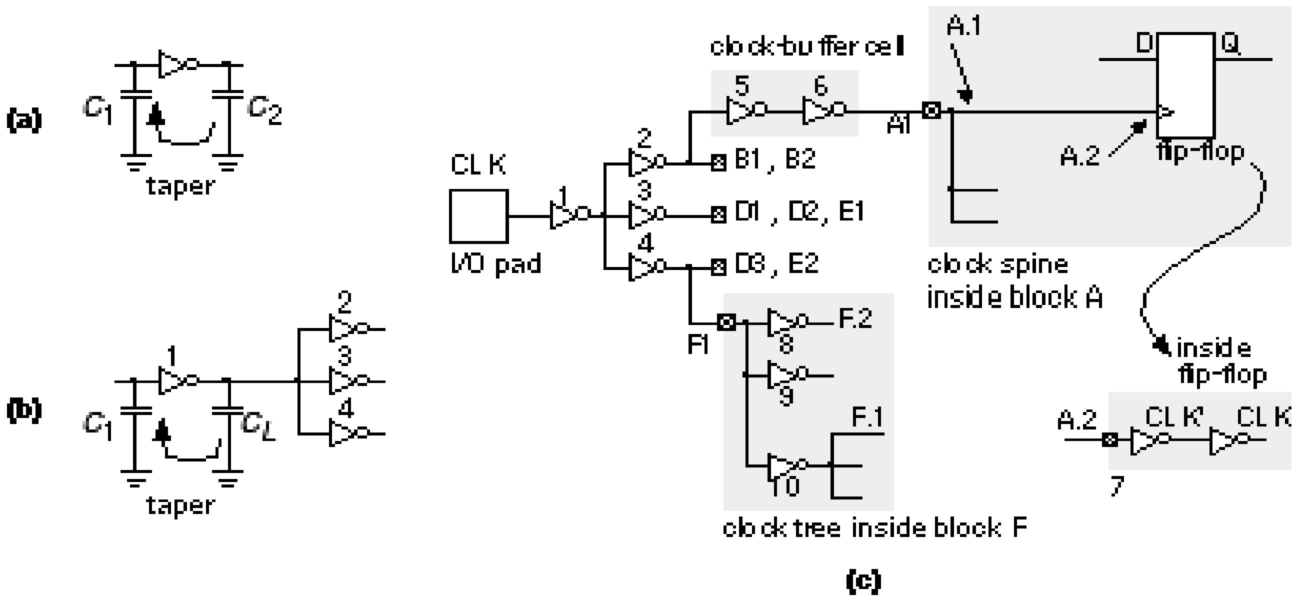
•(b) A clock spine for a cell-based ASIC (typical chips have thousands of clock nets).

•(c) A clock spine is usually driven from one or more clock-driver cells. Delay in the driver cell is a function of the number of stages and the ratio of output to input capacitance for each stage (taper).

•(d) Clock latency and clock skew. We would like to minimize both latency and skew.

Clock Planning (cont.,)

- FIGURE 16.17 A clock tree. (a) Minimum delay is achieved when the taper of successive stages is about 3. (b) Using a fanout of three at successive nodes.
  (c) A clock tree for the cell-based ASIC of Figure 16.16 b. We have to balance the clock arrival times at all of the leaf nodes to minimize clock skew.



Placement Methods

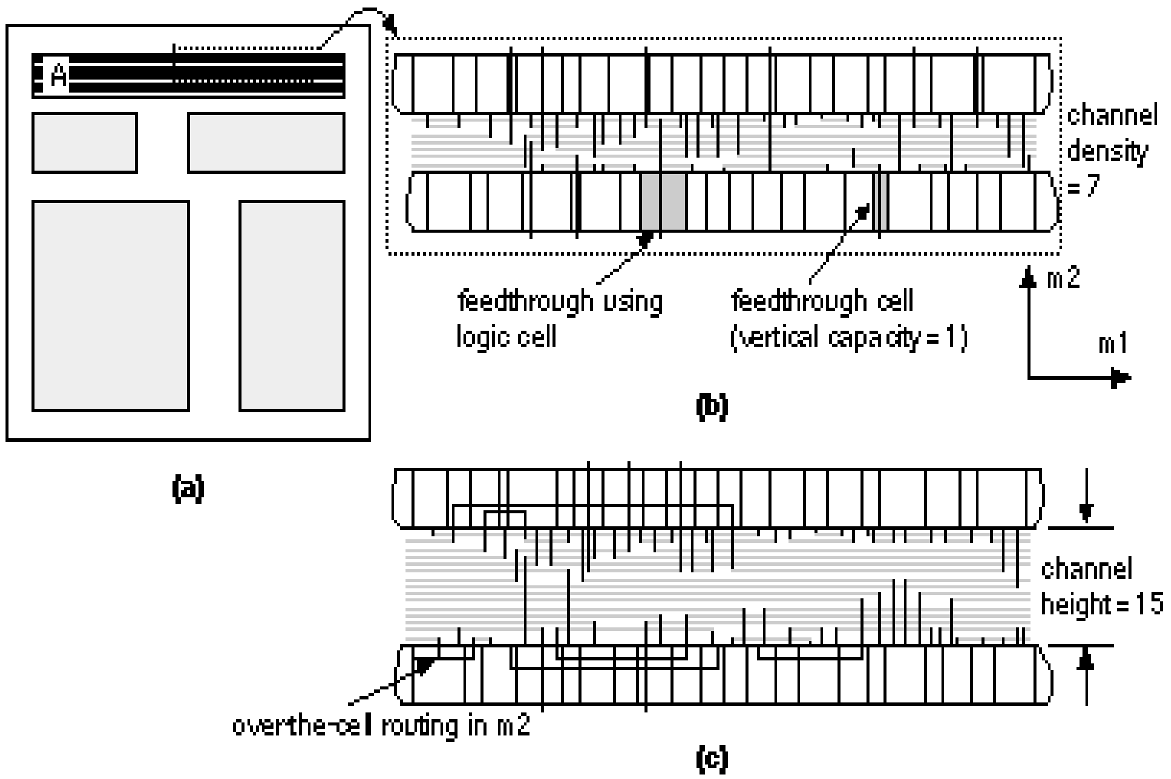
Introduction

- After completion of floorplan, **placement of the logic cells withinthe flexible blocks** takes place**.**
- **Placement is much more suited to automation** than floorplanning.
- After completion of floorplanning and placement, we can **predictboth**

  intrablock and interblock capacitances

Placement Terms and Definitions

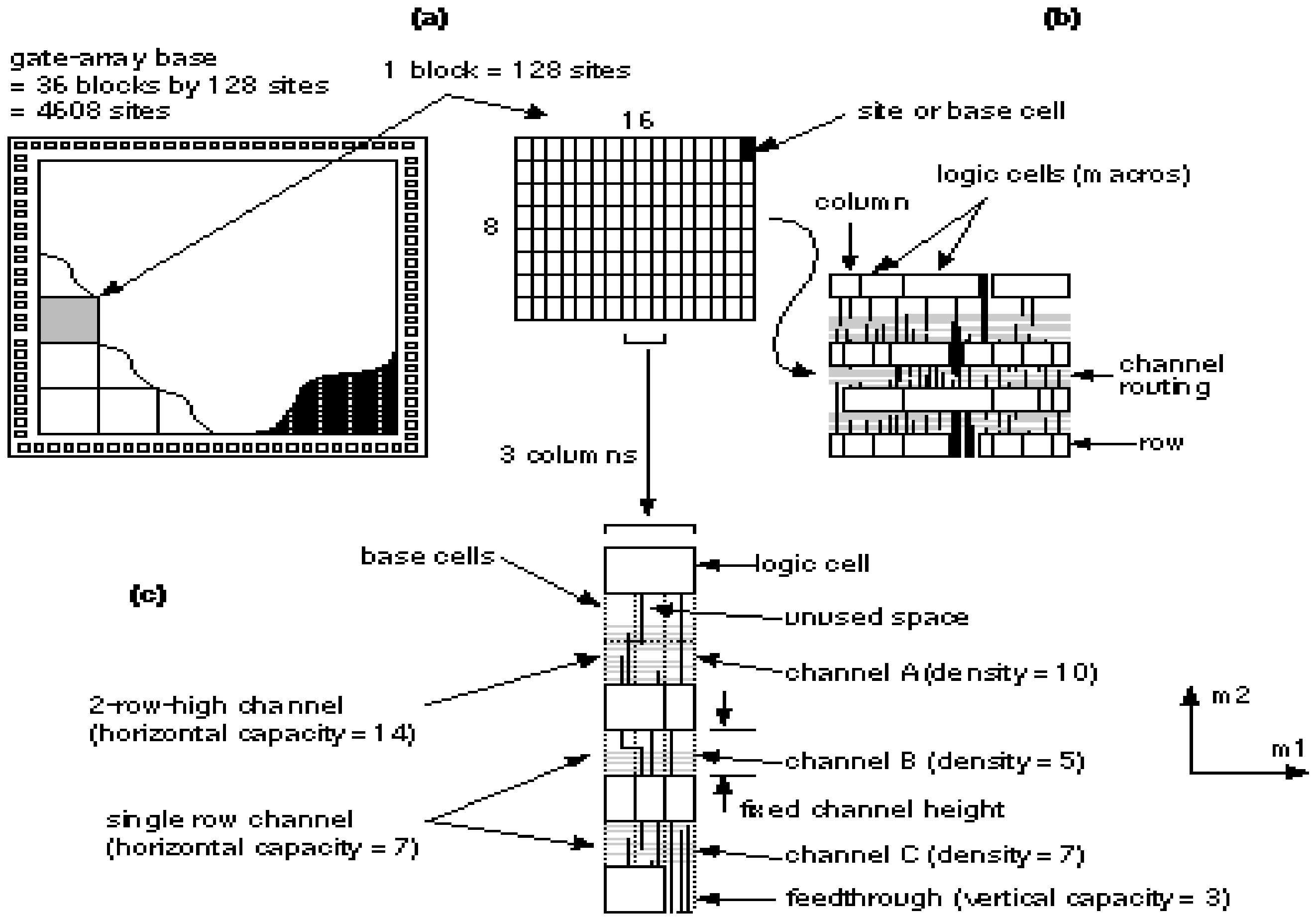
- **CBIC, MGA, and FPGA** architectures all have **rows of logic cells separated by the interconnect**—these are **row-based ASICs**



• **INTERCONNECT STRUCTURE**. (a) The two-level metal CBIC floorplan

•(b) A channel from the flexible block A. This channel has a channel height equal to the maximum channel density of 7

•(c) A channel that uses OTC (over-the-cell) routing in m2.



• **GATE-ARRAY INTERCONNECT**.

•(a) A small two-level metal gate array (about 4.6 k-gate).

•(b) Routing in a block.

•(c) Channel routing showing channel density and channel capacity. The channel height on a gate array may only be increased in increments of a row. If the interconnect does not use up all of the channel, rest of the space is wasted. The interconnect in the channel runs in m1 in the horizontal direction with m2 in the vertical direction.

Some commonly used terms:

- **Vertical interconnect** uses **feedthroughs** to cross the logic cells.
- **FEEDTHROUGH** or **JUMPER -** A vertical strip of metal that runs from the top to bottom of a cell (for double-entry cells), but has no connections inside the cell.
- **FEEDTHROUGH CELL** (or crosser cell) A dedicated empty cell (with no logic) that can hold one or more vertical interconnects
- **UNCOMMITTED FEEDTHROUGH** (also **BUILT-IN FEEDTHROUGH** , **IMPLICIT FEEDTHROUGH** , or **JUMPER**) - An unused vertical track (or just track) in a logic cell.
- **FEEDTHROUGH PIN** or **FEEDTHROUGH TERMINAL -** is an input or output that has connections at both the top and bottom of the standard cell.
- **SPACER CELL** (usually the same as a feedthrough cell) is used to fill space in rows so that the ends of all rows in a flexible block may be aligned to connect to power buses.
- **ELECTRICALLY EQUIVALENT CONNECTORS (OR EQUIPOTENTIAL CONNECTORS) -** Two connectors for the same physical net.
- **LOGICALLY EQUIVALENT CONNECTORS (**or **FUNCTIONALLY EQUIVALENT CONNECTORS, EQUIVALENT CONNECTORS**) **Example:** The **two inputs of a two-input NAND gate** may be logically equivalent connectors

Interconnect Area for CBIC,MGA and FPGA

HORIZONTAL INTERCONNECT

- **Channeled gate arrays and FPGAs,** the horizontal interconnect areas—the channels, usually on m1—have a **fixed capacity**.
- The channel capacity of **CBICs and channelless MGAs can be expanded** to hold as many interconnects as are needed.

VERTICAL INTERCONNECT

- In the **vertical interconnect** direction, usually m2, **FPGAs still have fixed resources**.

- In contrast the placement tool can always **add vertical feedthroughs** to a **channeled MGA, channelless MGA, or CBIC.**

Placement Goals and Objectives

Goal of a placement –

To arrange all the logic cells within the flexible blocks on a chip.

Ideally, the **objectives** of the placement step are to

- Guarantee the router can complete the routing step
- Minimize all the critical net delays
- Make the chip as dense as possible

Additional objectives:

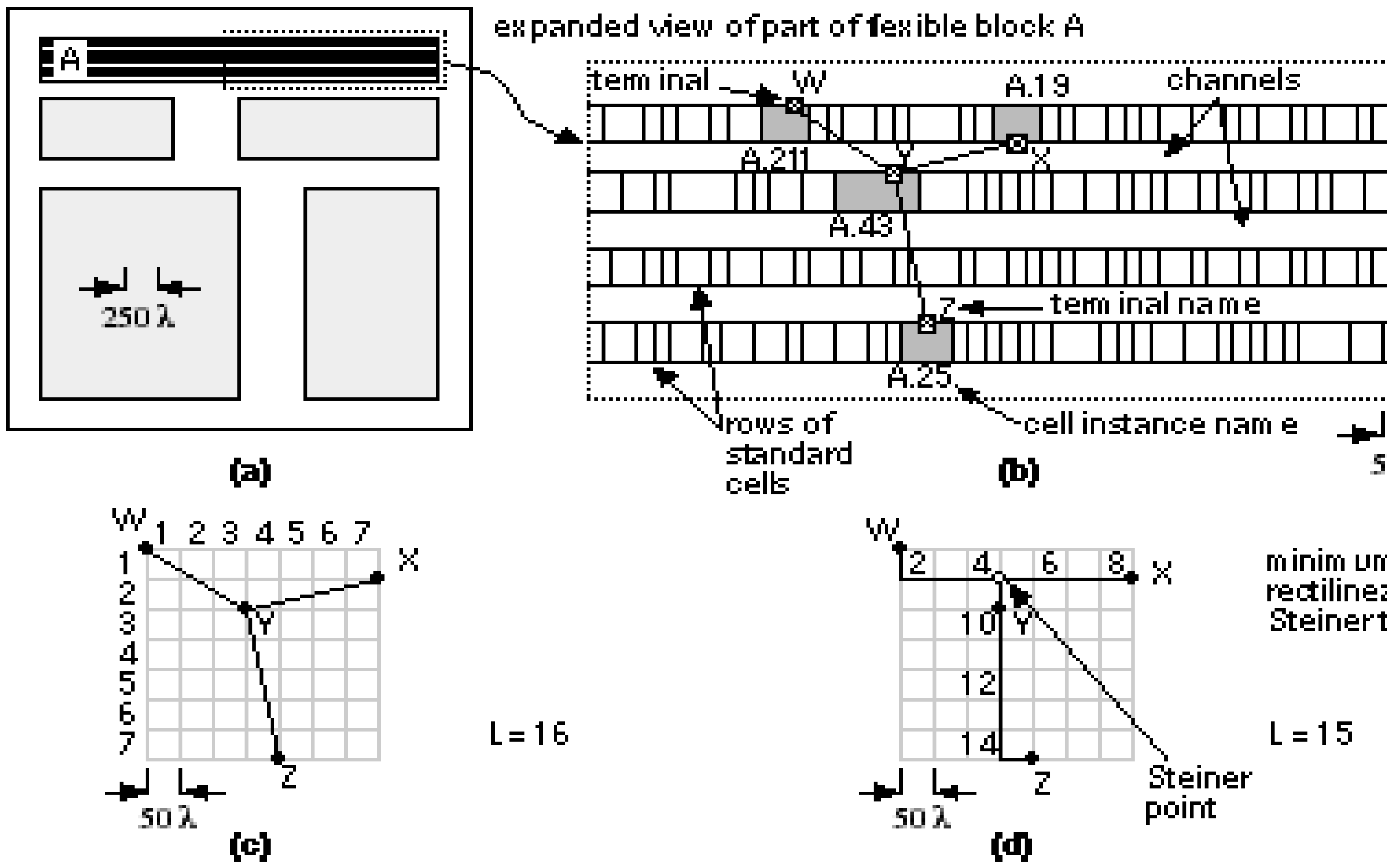
- Minimize power dissipation
- Minimize cross talk between signals

The most commonly used placement objectives (by current Placement tools):

- Minimize the total estimated interconnect length
- Minimize the interconnect congestion
- Meet the timing requirements for critical nets

Measurement of Placement Goals and Objectives - Interconnect length

- **Trees on graphs -** The graph structures that correspond to making **all the connections for a net.**
- **Steiner trees** —Special classes of trees-**minimize the total length of interconnect.**
- **Steiner tree (Rectilinear routing or Manhattan routing) -** This type of tree **uses diagonal connections**—solving problem using **interconnects on a rectangular grid**.
  - **Use Manhattan distance** than Euclidean distance.
- **Euclidean distance** between two points is the **straight-line distance**.
- **Manhattan distance - rectangular distance**.
- **Minimum rectilinear Steiner tree (MRST) - shortest interconnect using a rectangular grid.** The determination of the MRST is in general an NP- complete problem—which means it is hard to solve.



•**Placement using trees on graphs.** (a) The floorplan

•(b) An expanded view of the flexible block A showing four rows of standard cells for placement (typical blocks may contain thousands or tens of thousands of logic cells). We want to find the length of the net shown with four terminals, W through Z, given the placement of four logic cells (labeled: A.211, A.19, A.43, A.25). (c) The problem for net (W, X, Y, Z) drawn as a graph. The shortest connection is the minimum Steiner tree. (d) The minimum rectilinear Steiner tree using Manhattan routing. The rectangular (Manhattan) interconnect-length measures are shown for each tree

Measurement of Placement (contd.,) Interconnect Length

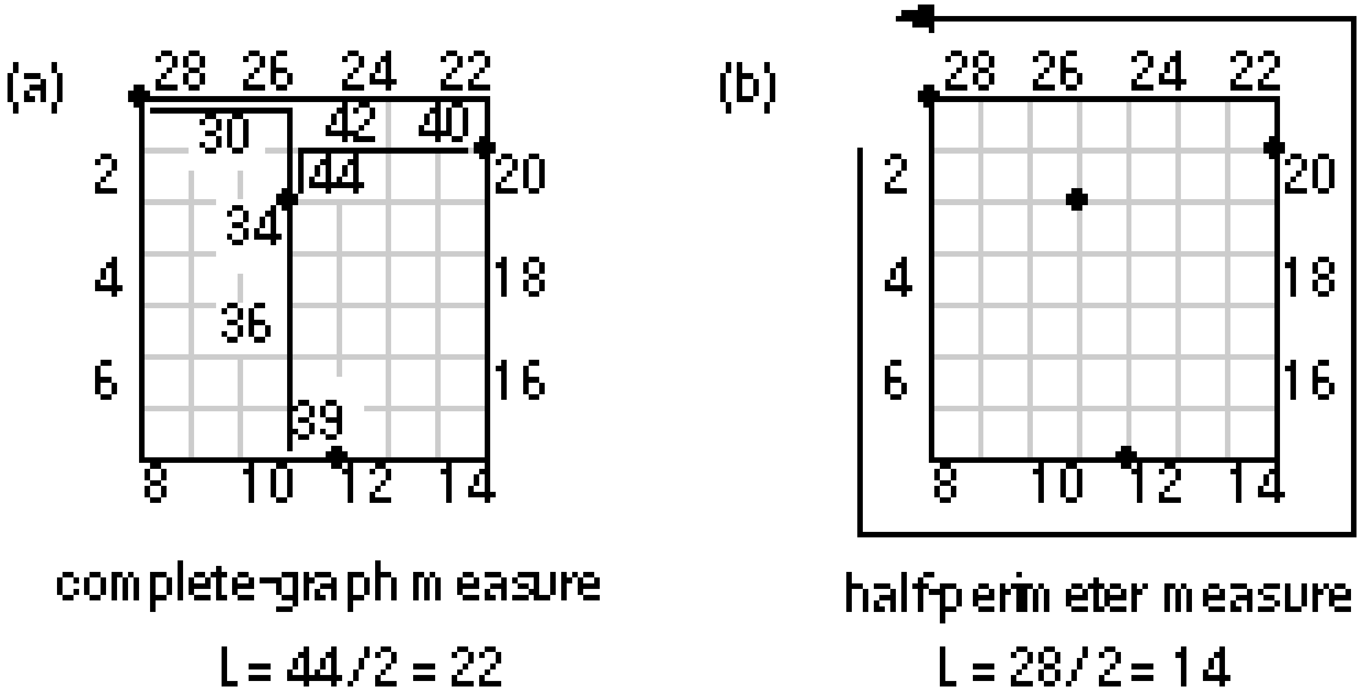
- **Complete graph** has connections from each terminal to every other terminal.
- **Complete-graph measure** adds all the interconnect lengths of the complete- graph connection together and then divides by n /2, where n is the number of terminals.

  Complete graph = (n (n -1)) / 2)
- **Bounding box** - the smallest rectangle that encloses all the terminals.
- **Half-perimeter measure** (or **bounding-box measure**) –

  one-half the perimeter of the bounding box.

  Half perimeter $f = \frac{1}{2} \Sigma^{m}_{i=1} h_i$

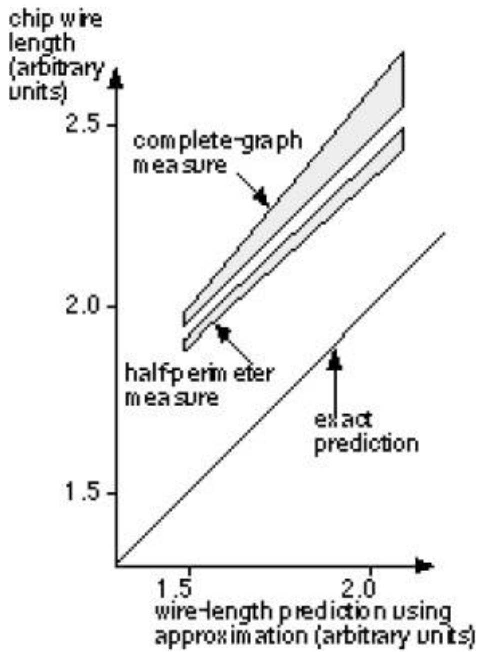
  where m is the nets, h_i is the half perimeter measure for net i.



Interconnect-length measures.

(a) Complete-graph measure. (b) Half-perimeter measure.

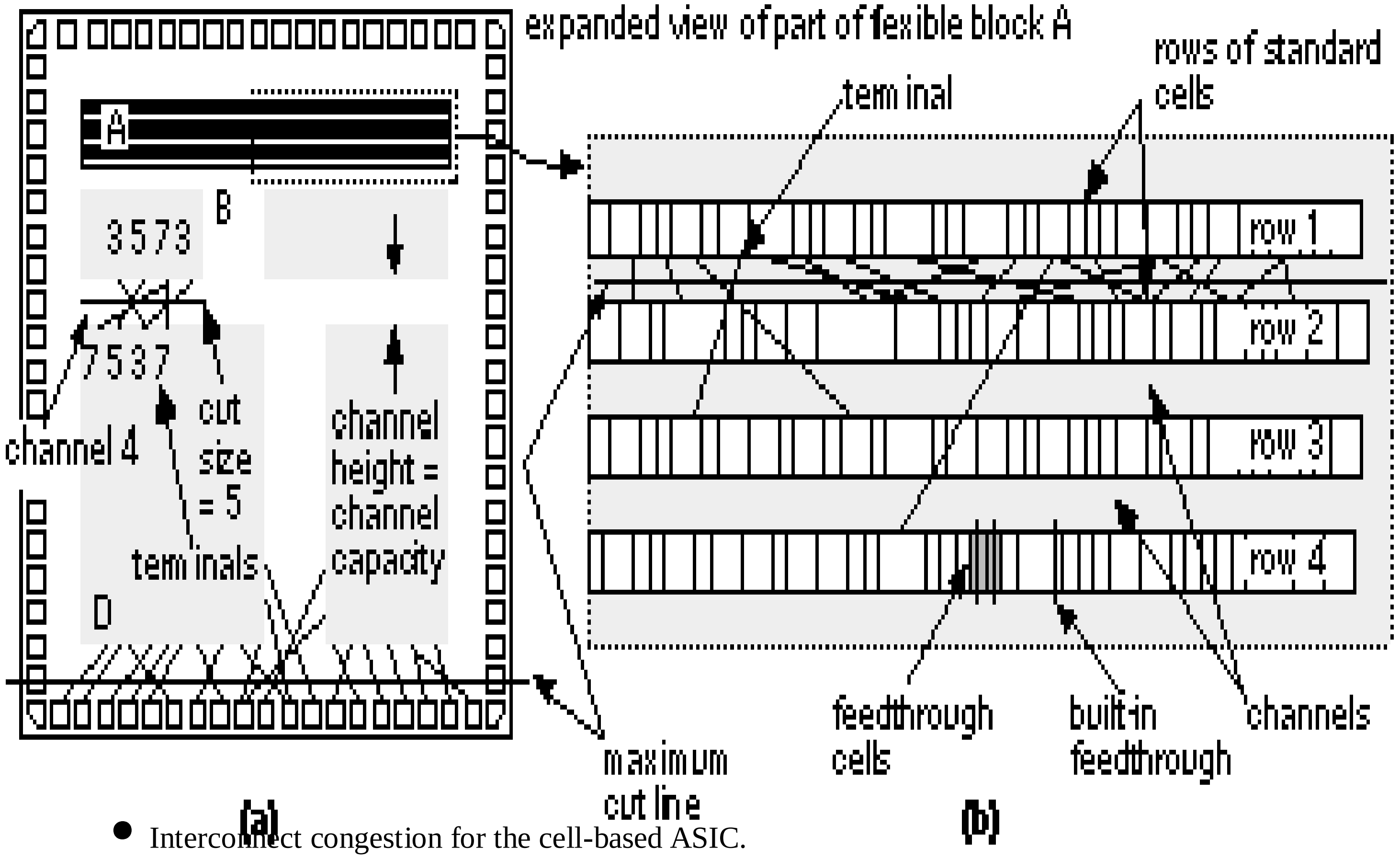
Correlation between total length of chip interconnect and the half-perimeter and complete-graph measures.



- **Meander factor** that specifies, on average, the ratio of the interconnect created by the routing tool to the interconnect-length estimate used by the placement tool.
- Another problem is **MRST that minimizes total net length may not minimize net delay.**

Interconnect Congestion

- There is no point in minimizing the interconnect length if we create a **placement that is too congested to route**.
- If we use **minimum interconnect congestion as an additional placement objective**, we need some way of measuring it.
- What we are trying to measure is **interconnect density**
- One **measure of interconnect congestion** uses the **maximum cut line** .
- **Maximum cut line:** Imagine a horizontal or vertical line drawn anywhere across a chip or block, **The number of interconnects that must cross this line** is the **cut size** (the number of interconnects we cut).
- The **maximum cut line has the highest cut size.**



- Interconnect congestion for the cell-based ASIC.
- (a) Measurement of congestion.
- (b) An expanded view of flexible block A shows a maximum cut line.

Interconnect Delay

- Many **placement tools minimize estimated interconnect length or interconnect congestion as objectives**.
- The **problem with this approach is that a logic cell may be placed a long way from another logic cell to which it has just one connection.** However, **the one long connection may be critical as far as timing delay is concerned.**
- As **technology is scaled, interconnection delays become relative to circuit delays** and this problem gets worse.
- The **minimum-length Steiner tree does not necessarily correspond to the interconnect path that minimizes delay.**

Interconnect Delay

- **Half-perimeter measure**
- In the placement phase typically a **simple interconnect length approximation is taken to this minimum-delay path .**

- **Timing-driven placement**
- In **timing-driven placement, estimate delay for every net for every trial** placement, possibly for hundreds of thousands of gates.

- **Parameters to estimate Interconnect Delay**
- No **information on which layers and how many vias the interconnect will use or how wide it will be.** Some tools allow us to include estimates for these parameters.
- Specification of **metal usage, the percentage of routing on the different layers to** expect from the router, **allows the placement tool to estimate RC values and delays—and thus minimize delay.**

Placement Algorithms

There are two classes of placement algorithms used in CAD tools:

- **Constructive placement** - uses a set of rules to arrive at a constructed placement.
  Example :**min-cut algorithm. Eigenvalue method**.
- **Iterative placement improvement.**

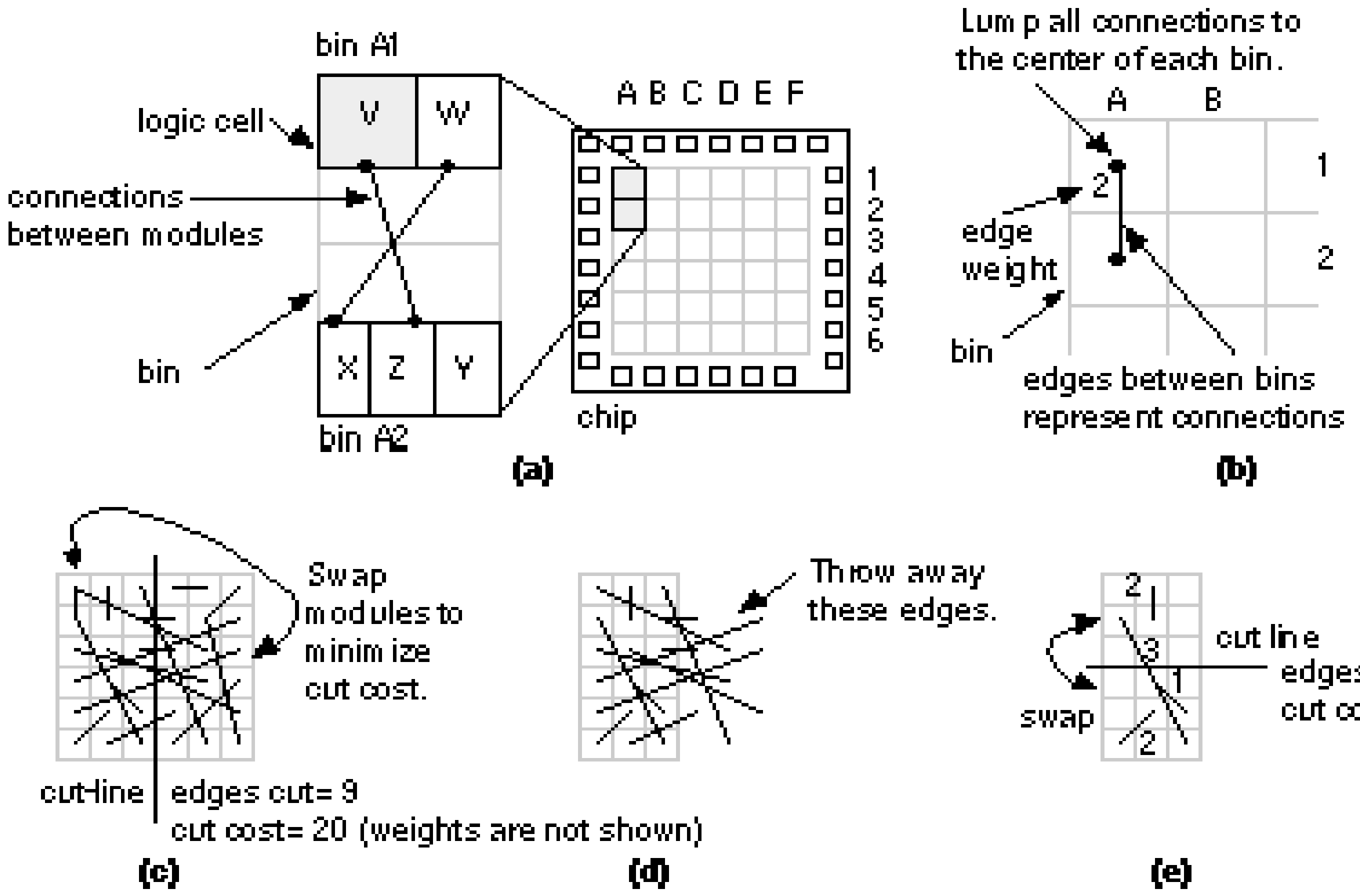
As in system partitioning, placement usually starts with a constructed solution and then improves it using an iterative algorithm.

Min-cut placement method

- uses successive application of partitioning. The steps are,

- **Cut the placement area into two pieces.**
- **Swap the logic cells to minimize the cut cost.**
- **Repeat the process from step 1, cutting smaller pieces until all the logic cells are placed**

Usually we divide the **placement area into bins** .
The size of a bin can vary, from a bin size equal to the base cell (for a gate array) to a **bin size that would hold several logic cells**.
We can **start with a large bin size, to get a rough placement, and then reduce the in size to get a final placement**.


bin A1
logic cell
V
W
connections between modules
bin
X
Z
Y
bin A2
A B C D E F
1
2
3
4
5
6
chip
(a)
Lump all connections to the center of each bin.
A
B
1
2
2
edge weight
bin
edges between bins represent connections
(b)
Swap modules to minimize cut cost.
cut-line
edges cut= 9
cut cost= 20 (weights are not shown)
(c)
Throw away these edges.
(d)
2
3
1
2
swap
cut line
(e)

•**Min-cut placement.** (a) Divide the chip into bins using a grid.

•(b) Merge all connections to the center of each bin.

•(c) Make a cut and swap logic cells between bins to minimize the cost of the cut.

•(d) Take the cut pieces and throw out all the edges that are not inside the piece.

•(e) Repeat the process with a new cut and continue until we reach the individual bins.

Placement Algorithms (contd.,)

The eigenvalue placement algorithm **uses the cost matrix or weighted connectivity matrix** (**eigen value methods** are also known as **spectral methods**).

The measure we use is **a cost function f** that we shall minimize, given by ,

$$f = \frac{1}{2} \sum_{i=1}^{n} c_{ij} d_{ij}^{2} \qquad (1)$$

where **C = [c $_{ij}$] is the (possibly weighted) connectivity matrix**, and **d $_{ij}$ is the Euclidean distance between the centers of logic cell i and logic cell j** .

Since we are going to minimize a cost function that is the square of the distance between logic cells, these methods are also known as **quadratic placement methods**.

This type of cost function leads to a simple mathematical solution. We can rewrite the cost function f in matrix form:

$$f = \frac{1}{2} \sum_{i,j=1}^{n} c_{ij} \left((x_i - x_j)^2 + (y_i - y_j)^2 \right)$$

$$f = x^T Bx + y^T By$$

B is a symmetric matrix, the disconnection matrix (also called the Laplacian).

B= D- C

where,
$$d_{ii} = \sum_{j=1}^{n} C_{ij}$$
$$d_{ij} = 0, \; i \neq j$$

We can simplify the problem by noticing that it is symmetric in the **x - and y -coordinates**.

Let us **solve the simpler problem of minimizing the cost function for the placement of logic cells along just the x – axis first.**

We can then apply this solution to the more general **two-dimensional placement problem**.

Before we solve this simpler problem, we introduce a constraint that the **coordinates of the logic cells must correspond to valid positions** (the cells do not overlap and they are placed on-grid).

We make another simplifying assumption that **all logic cells are the same size and we must place them in fixed positions.**

We can define a vector p consisting of the valid positions:

$$p = [p_1, p_2 \ldots\ldots p_n] \qquad (4)$$

For a valid placement the x -coordinates of the logic cells,

$$x = [x_1, x_2, .. \; x_n] \qquad (5)$$

must be a permutation of the fixed positions, p . We can show that requiring the logic cells to be in fixed positions in this way leads to a series of n equations restricting the values of the logic cell coordinates .If we impose all of these **constraint equations** the problem becomes very complex. Instead we choose just one of the equations:

$$\sum_{i=1}^{n} x_i^2 = \sum^{n} p_i^2$$

Simplifying the problem in this way will lead to an approximatesolution

to the placement problem.

We can write this single constraint on the x -coordinates in matrixform:

$$x^T x = P$$

$$P = \sum_{i=1}^{n} p_i^2$$

where P is a constant.

We can now summarize the formulation of the problem, with the simplifications that we have made, for a one-dimensional solution. We must **minimize a cost function, g,** where

$$g = x^T Bx \qquad (8)$$

subject to the constraint:

$$x^T x = p \qquad (9)$$

This is a standard problem that we can solve using a Lagrangian multiplier:

$$\Lambda = x^T Bx - \lambda\left[x^T x - p\right] \qquad (10)$$

To find the value of x that minimizes g we differentiate partially with respect to x and set the result equal to zero. We get the following equation:

$$\left[B - \lambda I\right]x = 0 \qquad (11)$$

This last equation is called the characteristic equation for the disconnection matrix B and occurs frequently in matrix algebra (this l has nothing to do with scaling).

The solutions to this equation are the eigenvectors and eigenvalues of B .

Multiplying Eq.(11) by x ^{T} we get:

$$\lambda x^T x = x^T Bx \qquad \lambda = \frac{g}{p}$$

However, since we imposed the constraint x ^{T} x = P and x ^{T} Bx = g , then

The eigenvectors of the disconnection matrix B are the solutions to our placement problem.

$$(B - \lambda I)x = 0$$

This last equation is called the **characteristic equation** for the disconnection matrix B and occurs frequently in matrix algebra (this l has nothing to do with scaling).

The solutions to this equation are the **eigenvectors and eigenvalues** of B .

Eigenvalue placement problem

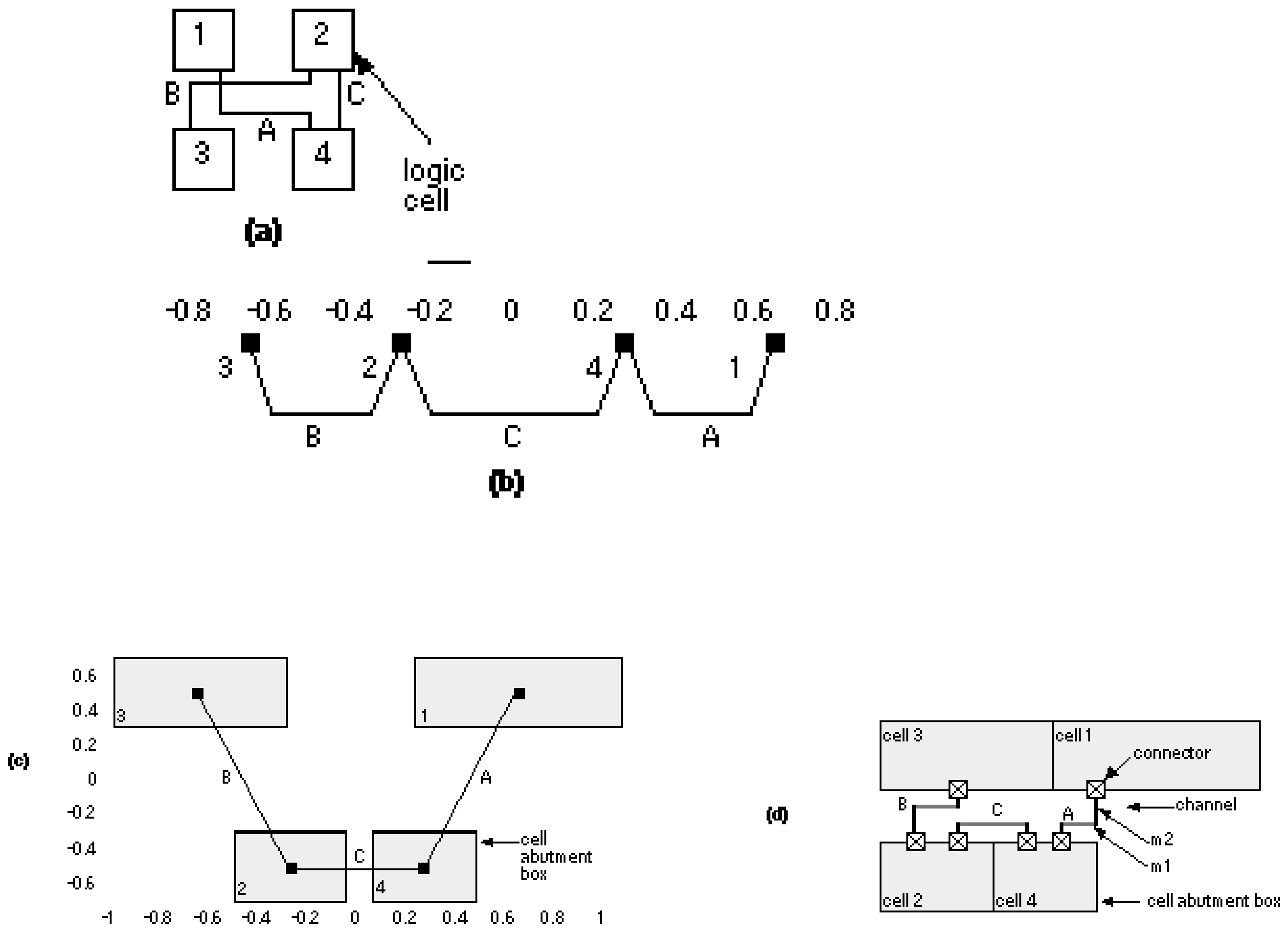
$$\mathbf{C} = \begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{matrix}$$

- B= D- C

$$\mathbf{B} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{matrix} - \begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{matrix} = \begin{matrix} 1 & 0 & 0 & -1 \\ 0 & 2 & -1 & -1 \\ 0 & -1 & 1 & 0 \\ -1 & -1 & 0 & 2 \end{matrix}$$

$$(B - \lambda I)x = 0$$

- $\lambda^4 - 6\lambda^3 + 10\lambda^2 - 4\lambda = 0$
- $\lambda = 0, 0.585, 3.414, 2$


1
2
3
4
B
C
A
logic cell
(a)
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
3
2
4
1
B
C
A
(b)
(c)
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
3
1
2
4
B
A
C
cell abutment box
(d)
cell 3
cell 1
connector
channel
B
C
A
m2
m1
cell 2
cell 4
cell abutment box

Iterative Placement Improvement

An iterative placement improvement algorithm takes an existing placement and tries to improve it by moving the logic cells. There are two parts to the algorithm:

- The **selection criteria** that decides which logic cells to try moving.
- The **measurement criteria** that decides whether to move the selected cells.

There are several **interchange or iterative exchange methods** that differ in their selection and measurement criteria:

- Pair wise interchange,
- force-directed interchange,
- force-directed relaxation, and
- force-directed pair wise relaxation.

All of these methods usually consider only pairs of logic cells to be exchanged. A source logic cell is picked for trial exchange with a destination logic cell.

Iterative Placement Improvement (contd.,)

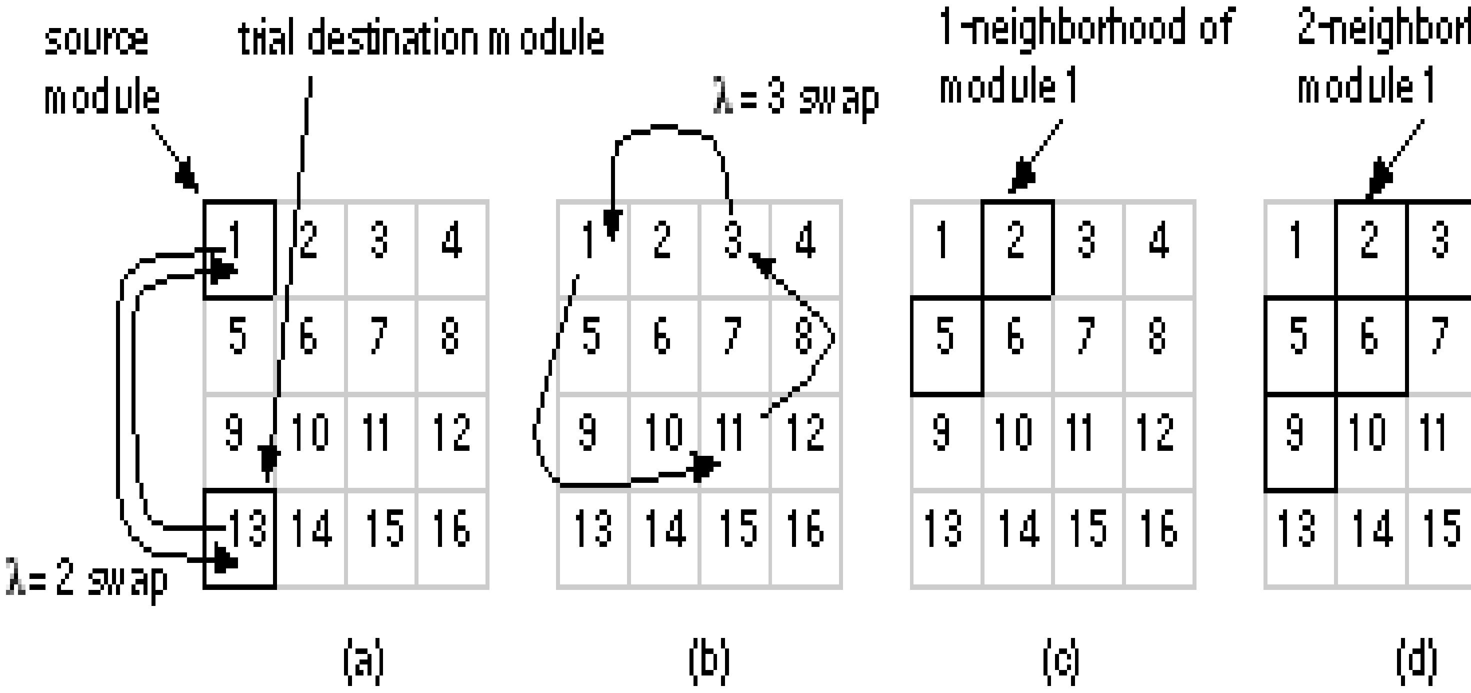
The pair wise-interchange algorithm

- Select the source logic cell at random.
- Try all the other logic cells in turn as the destination logic cell.
- Use any of the measurement methods we have discussed to decide on whether to accept the interchange.
- The process repeats from step 1, selecting each logic cell in turn as a

  source logic cell.

Neighborhood exchange algorithm

- Modification to pairwise interchange that considers onlydestination logic cells in a neighborhood —cells within a certain distance, e, of the source logic cell.
- Limiting the search area for the destination logic cell to the e - neighborhood **reduces the search time**.



• **Pair-wise Interchange.**

• (a) Swapping the source logic cell with a destination logic cell in pairwise interchange.

• (b) Sometimes we have to swap more than two logic cells at a time to reach an optimum placement, but this is expensive in computation time. Limiting the search to neighborhoods reduces the search time. Logic cells within a distance e of a logic cell form an e-neighborhood.

• (c) A one-neighborhood.

• (d) A two-neighborhood.

Iterative Placement Improvement

-

Force-directed placement method

- Imagine **connecting all the logic cells which are going to place are connected through** identical **springs** .

• The **number of springs is equal to the number of connections** between logic cells.

• The effect of the springs is to pull connected logic cells together.

• The more highly connected the logic cells, the stronger the pull of the springs.

• The **force on a logic cell i due to logic cell j** is given by **Hooke's law** , which says the force of a spring is proportional to its extension:

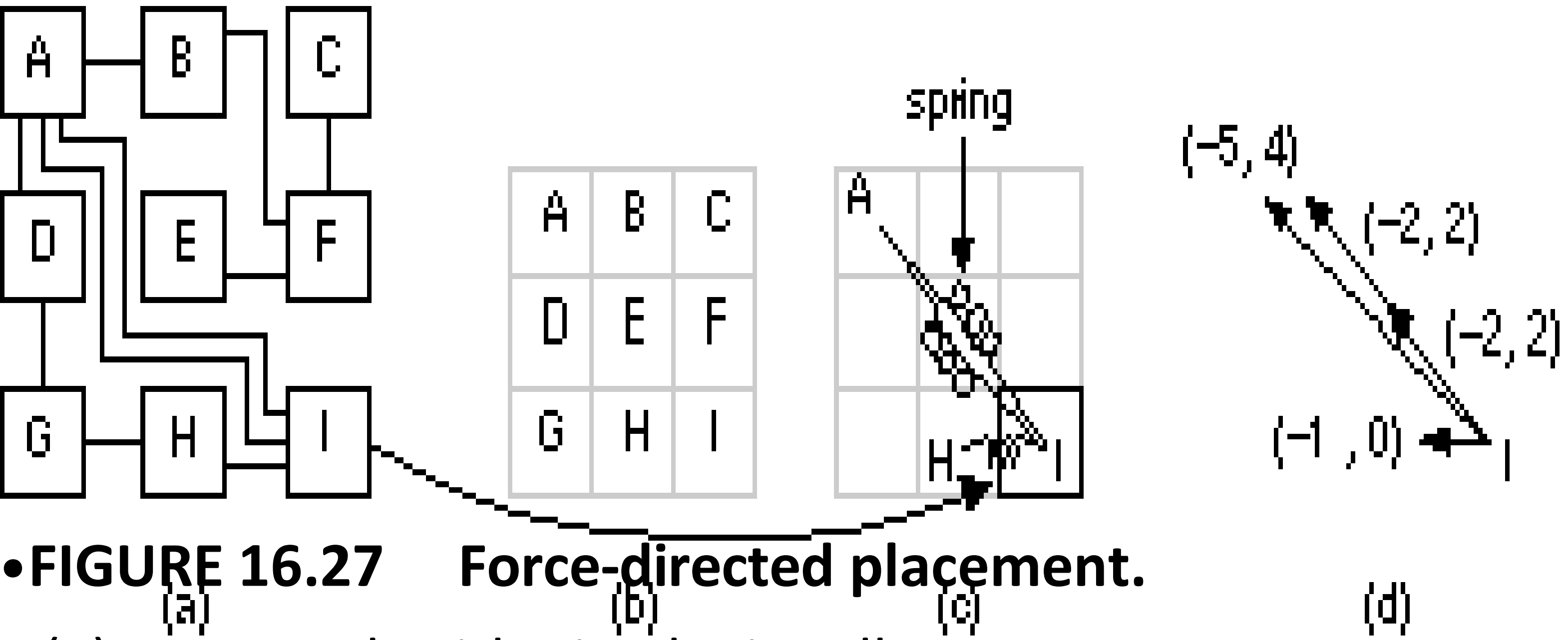
$$\mathbf{F_{ij} = -c_{ij} x_{ij}} .$$

– The vector component x_{ij} is directed from the center of logic cell i to the center of logic cell j .

– The vector magnitude is calculated as either the **Euclidean or Manhattan distance** between the logic cell centers.

– The c_{ij} form the connectivity or cost matrix (the matrix element c_{ij} is the number of connections between logic cell i and logic cell j).

•



•**FIGURE 16.27 Force-directed placement.**

• (a) A network with nine logic cells.

- (b) We make a grid (one logic cell per bin).
- (c) Forces are calculated as if springs were attached to the centers of each logic cell for each connection.The two nets connecting logic cells A and I correspond to two springs.
- (d) The forces are proportional to the spring extensions.

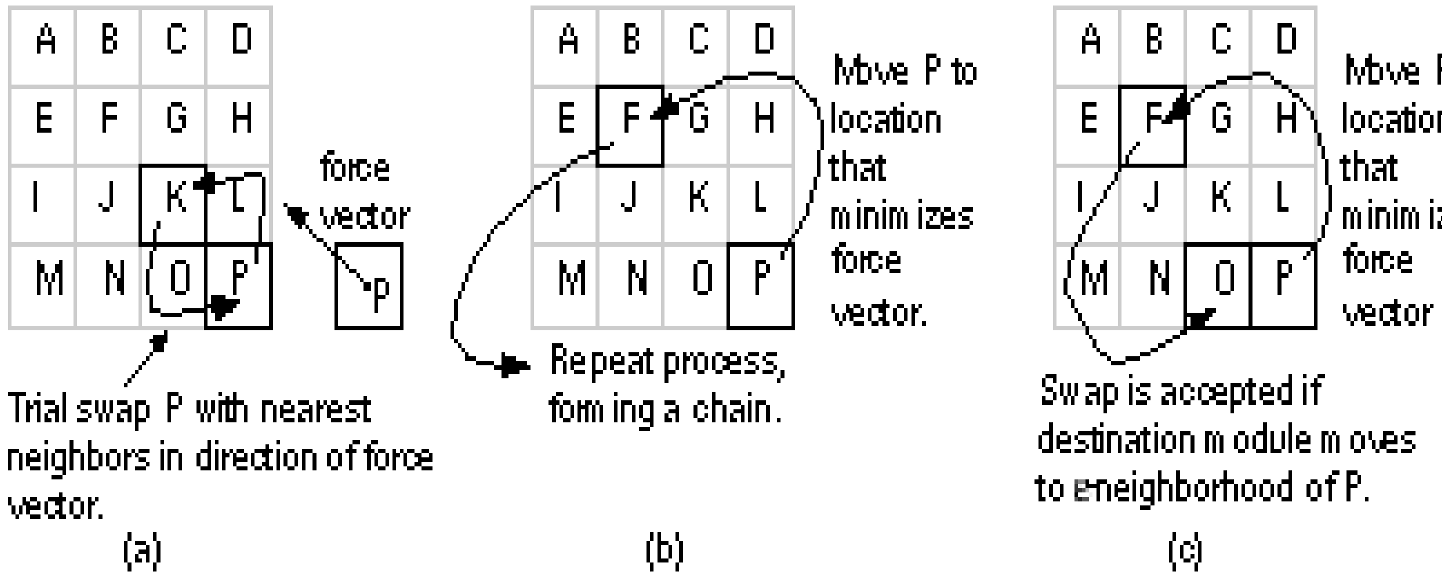
Iterative Placement Improvement (contd.,)

Force-directed placement algorithms:

- The **force-directed interchange algorithm uses the force vector** to select a pair of logic cells to swap.
- The **force-directed relaxation** a chain of logic cells is moved.
- The **force-directed pairwise relaxation algorithm** swaps one pair of logic cells at a time.

` Force-directed solution **minimize the energy of the system, corresponding to minimizing the sum of the squares of the distances separating logic cells.**

Force-directed placement algorithms thus also use **a quadratic cost function**.


A B C D
E F G H
I J K L
M N O P
force vector
p
Trial swap P with nearest neighbors in direction of force vector.
(a)
A B C D
E F G H
I J K L
M N O P
Move P to location that minimizes force vector.
Repeat process, forming a chain.
(b)
A B C D
E F G H
I J K L
M N O P
Swap is accepted if destination module moves to neighborhood of P.
(c)

- **FIGURE 16.28 Force-directed iterative placement improvement.**
- **(a) Force-directed interchange.**
- **(b) Force-directed relaxation.**
- **(c) Force-directed pairwise relaxation.**

Placement Using Simulated Annealing

Applying simulated annealing to placement, the algorithm is as follows:

- Select logic cells for a trial interchange, usually at random.
- Evaluate the objective function E for the new placement.
- **If □E is negative or zero, then exchange the logic cells.**
- **If □E is positive, then exchange the logic cells with a probability of exp(–□E/T).**
- Go back to step 1 for a fixed number of times, and **then lower the temperature T according to a cooling schedule**: $T_{n+1} = 0.9\ T_n$, for example.

Comparison of Placement Algorithms

- **Min-cut based constructive placement is faster** than simulated annealing .
- **Simulated annealing is capable of giving better results at the expense of long computer run times**.
- The **iterative improvement methods** that described earlier are capable of giving results as **good as simulated annealing**, but they **use more complex algorithms.**

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF ECE
COURSE MATERIAL

COURSE NAME: CMOS VLSI DESIGN COURSE CODE:SEC1316

Sample Questions:

Part-A

1. What are the ASIC design steps?

2. What are the objectives of System Partitioning?

3. What is Floorplanning?

4. What is Placement?

5. What are the types of Routing?

6. What is Min-Cut Placement algorithm?

Part-B

7. With an example, explain about K-L algorithm and its features.

8. Write short notes about Force directed placement algorithm.

9. Explain about Greedy Channel Routing algorithm.

10.Describe about Min-cut placement algorithm with an example**.**

--