

UNIT-I

Introduction to IC Technology-MOS, PMOS, NMOS, CMOS & BiCMOS:

Moore's Law:

- Gordon Moore: co-founder of Intel
- Predicted that the number of transistors per chip would grow exponentially (double every 18 months)
- Exponential improvement in technology is a natural trend:
 - e.g. Steam Engines - Dynamo - Automobile

The Cost of Fabrication:

- Current cost \$2 - 3 billion
- Typical fab line occupies 1 city block, employs a few hundred employees
- Most profitable period is first 18 months to 2 years
- For large volume IC's packaging and testing is largest cost
- For low volume IC's, design costs may swamp manufacturing costs

What is a Silicon Chip?

- A pattern of interconnected switches and gates on the surface of a crystal of semiconductor (typically Si)
- These switches and gates are made of
 - areas of n-type silicon
 - areas of p-type silicon
 - areas of insulator
 - lines of conductor (interconnects) joining areas together
- Aluminium, Copper, Titanium, Molybdenum, polysilicon, tungsten
- The geometry of these areas is known as the layout of the chip
- Connections from the chip to the outside world are made around the edge of the chip to facilitate connections to other devices

Switches:

- Digital equipment is largely composed of switches
- Switches can be built from many technologies
 - relays (from which the earliest computers were built)
 - thermionic valves
 - transistors
- The perfect digital switch would have the following:
 - switch instantly
 - use no power
 - have an infinite resistance when off and zero resistance when on
- Real switches are not like this!

Semiconductors and Doping:

- Adding trace amounts of certain materials to semiconductors alters the crystal structure and can change their electrical properties
 - in particular it can change the number of free electrons or holes
- N-Type
 - semiconductor has free electrons

- dopant is (typically) phosphorus, arsenic, antimony
- P-Type
 - semiconductor has free holes
 - dopant is (typically) boron, indium, gallium
- Dopants are usually implanted into the semiconductor using Implant Technology, followed by thermal process to diffuse the dopants

IC Technology:

- Speed / Power performance of available technologies
- The microelectronics evolution
- SIA Roadmap
- Semiconductor Manufacturers 2001 Ranking

Metal-oxide-semiconductor (MOS) and related VLSI technology:

- pMOS
- nMOS
- CMOS
- BiCMOS
- GaAs

Basic MOS Transistors:

- Minimum line width
- Transistor cross section
- Charge inversion channel
- Source connected to substrate
- Enhancement vs Depletion mode devices
- pMOS are 2.5 time slower than nMOS due to electron and hole mobilities
-

Fabrication Technology:

- Silicon of extremely high purity
 - chemically purified then grown into large crystals
- Wafers
 - crystals are sliced into wafers
 - wafer diameter is currently 150mm, 200mm, 300mm
 - wafer thickness <1mm
 - surface is polished to optical smoothness
- Wafer is then ready for processing
- Each wafer will yield many chips
 - chip die size varies from about 5mmx5mm to 15mmx15mm
 - A whole wafer is processed at a time
- Different parts of each die will be made P-type or N-type (small amount of other atoms intentionally introduced - doping -implant)
- Interconnections are made with metal

- Insulation used is typically SiO₂. SiN is also used. New materials being investigated (low-k dielectrics)
- nMOS Fabrication
- CMOS Fabrication
 - p-well process
 - n-well process
 - twin-tub process
- All the devices on the wafer are made at the same time
- After the circuitry has been placed on the chip
 - the chip is overglassed (with a passivation layer) to protect it
 - only those areas which connect to the outside world will be left uncovered (the pads)
- The wafer finally passes to a test station
 - test probes send test signal patterns to the chip and monitor the output of the chip
- The *yield* of a process is the percentage of die which pass this testing
- The wafer is then scribed and separated up into the individual chips. These are then packaged
- Chips are 'binned' according to their performance
-

CMOS Technology:

- First proposed in the 1960s. Was not seriously considered until the severe limitations in power density and dissipation occurred in NMOS circuits
- Now the dominant technology in IC manufacturing
- Employs both pMOS and nMOS transistors to form logic elements
- The advantage of CMOS is that its logic elements draw significant current only during the transition from one state to another and very little current between transitions - hence power is conserved.
- In the case of an inverter, in either logic state one of the transistors is off. Since the transistors are in series, (~ no) current flows.
- See twin-well cross sections

BiCMOS:

- A known deficiency of MOS technology is its limited load driving capabilities (due to limited current sourcing and sinking abilities of pMOS and nMOS transistors).
- Bipolar transistors have
 - higher gain
 - better noise characteristics
 - better high frequency characteristics
- BiCMOS gates can be an efficient way of speeding up VLSI circuits
- See table for comparison between CMOS and BiCMOS
- CMOS fabrication process can be extended for BiCMOS

- Example Applications
 - CMOS - Logic
 - BiCMOS - I/O and driver circuits
 - ECL - critical high speed parts of the system

TECHNOLOGIES:

Semiconductor Fabrication Processes

Starting with an uniformly doped silicon wafer, the fabrication of integrated circuits (IC's) needs hundreds of sequential process steps. The most important process steps used in the semiconductor fabrication are

a) Lithography

Lithography is used to transfer a pattern from a photomask to the surface of the wafer. For example the gate area of a MOS transistor is defined by a specific pattern. The pattern information is recorded on a layer of photoresist which is applied on the top of the wafer. The photoresist changes its physical properties when exposed to light (often ultraviolet) or another source of illumination (e.g. X-ray). The photoresist is either developed by (wet or dry) etching or by conversion to volatile compounds through the exposure itself. The pattern defined by the mask is either removed or remained after development, depending if the type of resist is positive or negative. For example the developed photoresist can act as an etching mask for the underlying layers.

b) Etching

Etching is used to remove material selectively in order to create patterns. The pattern is defined by the etching mask, because the parts of the material, which should remain, are protected by the mask. The unmasked material can be removed either by wet (chemical) or dry (physical) etching. Wet etching is strongly isotropic which limits its application and the etching time can be controlled difficultly. Because of the so-called under-etch effect, wet etching is not suited to transfer patterns with sub-micron feature size. However, wet etching has a high selectivity (the etch rate strongly depends on the material) and it does not damage the material. On the other side dry etching is highly anisotropic but less selective. But it is more capable for transferring small structures.

c) Deposition

A multitude of layers of different materials have to be deposited during the IC fabrication process. The two most important deposition methods are the physical vapor deposition (PVD) and the chemical vapor deposition (CVD). During PVD accelerated gas ions sputter particles from a sputter target in a low pressure plasma chamber. The principle of CVD is a chemical reaction of a gas mixture on the substrate surface at high temperatures. The need of high temperatures is the most restricting factor for applying CVD. This problem can be avoided with plasma enhanced chemical vapor deposition (PECVD), where the chemical reaction is enhanced with radio frequencies instead of high temperatures. An important aspect for this

technique is the uniformity of the deposited material, especially the layer thickness. CVD has a better uniformity than PVD.

d) Chemical Mechanical Planarization

Processes like etching, deposition, or oxidation, which modify the topography of the wafer surface lead to a non-planar surface. Chemical mechanical planarization (CMP) is used to plane the wafer surface with the help of a chemical slurry. First, a planar surface is necessary for lithography due to a correct pattern transfer. Furthermore, CMP enables indirect patterning, because the material removal always starts on the highest areas of the wafer surface. This means that at defined lower lying regions like a trench the material can be left. Together with the deposition of non-planar layers, CMP is an effective method to build up IC structures.

e) Oxidation

Oxidation is a process which converts silicon on the wafer into silicon dioxide. The chemical reaction of silicon and oxygen already starts at room temperature but stops after a very thin native oxide film. For an effective oxidation rate the wafer must be settled to a furnace with oxygen or water vapor at elevated temperatures. Silicon dioxide layers are used as high-quality insulators or masks for ion implantation. The ability of silicon to form high quality silicon dioxide is an important reason, why silicon is still the dominating material in IC fabrication.

OXIDATION TECHNIQUES

1. Cleaned wafers are placed in the wafer load station where dry nitrogen (N₂) is introduced into the chamber. The nitrogen prevents oxidation from occurring while the furnace reaches the required temperature.

2. Once the specified temperature in the chamber is reached, the nitrogen gas flow is shut off and oxygen (O₂) is added to the chamber. The source of the oxygen can be gas or water vapor depending upon the dry process or wet process.

- After the oxidation is complete and the oxide layer is the correct thickness, nitrogen is reintroduced into the chamber to prevent further oxidation from occurring.
- The wafers are then removed from the chamber. After inspection, they are ready for further processing.
- Thermal oxidation can be either a dry or a wet process

f) Ion Implantation

Ion implantation is the dominant technique to introduce dopant impurities into crystalline silicon. This is performed with an electric field which accelerates the ionized atoms or molecules so that these particles penetrate into the target material until they come to rest because of interactions with the silicon atoms. Ion implantation is able to control exactly the distribution and dose of the dopants in silicon, because the penetration depth depends on the kinetic energy of the ions which is proportional to the electric field. The dopant dose can be controlled by varying the ion source. Unfortunately, after ion implantation the crystal structure is damaged which implies worse electrical properties. Another problem is that the implanted dopants are electrically inactive, because they are situated on interstitial sites.

Therefore after ion implantation a thermal process step is necessary which repairs the crystal damage and activates the dopants.

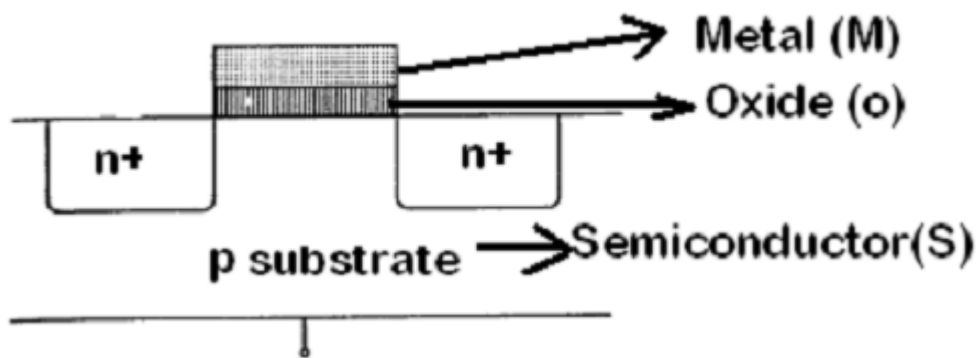
g) Diffusion

Diffusion is the movement of impurity atoms in a semiconductor material at high temperatures. The driving force of diffusion is the concentration gradient. There is a wide range of diffusivities for the various dopant species, which depend on how easy the respective dopant impurity can move through the material. Diffusion is applied to anneal the crystal defects after ion implantation or to introduce dopant atoms into silicon from a chemical vapor source. In the last case the diffusion time and temperature determine the depth of dopant penetration. Diffusion is used to form the source, drain, and channel regions in a MOS transistor. But diffusion can also be an unwanted parasitic effect, because it takes place during all high temperature process steps.

Basic MOS Transistors:

Why the name MOS?

We should first understand the fact that why the name Metal Oxide Semiconductor transistor, because the structure consists of a layer of Metal (gate), a layer of oxide (SiO_2) and a layer of semiconductor. Figure 3 below clearly tell why the name MOS



We have two types of FETs. They are Enhancement mode and depletion mode transistor. Also we have PMOS and NMOS transistors.

In Enhancement mode transistor channel is going to form after giving a proper positive gate voltage. We have NMOS and PMOS enhancement transistors.

In Depletion mode transistor channel will be present by the implant. It can be removed by giving a proper negative gate voltage. We have NMOS and PMOS depletion mode transistors.

N-MOS enhancement mode transistor:-

This transistor is normally off. This can be made ON by giving a positive gate voltage. By giving a +ve gate voltage a channel of electrons is formed between source drain.

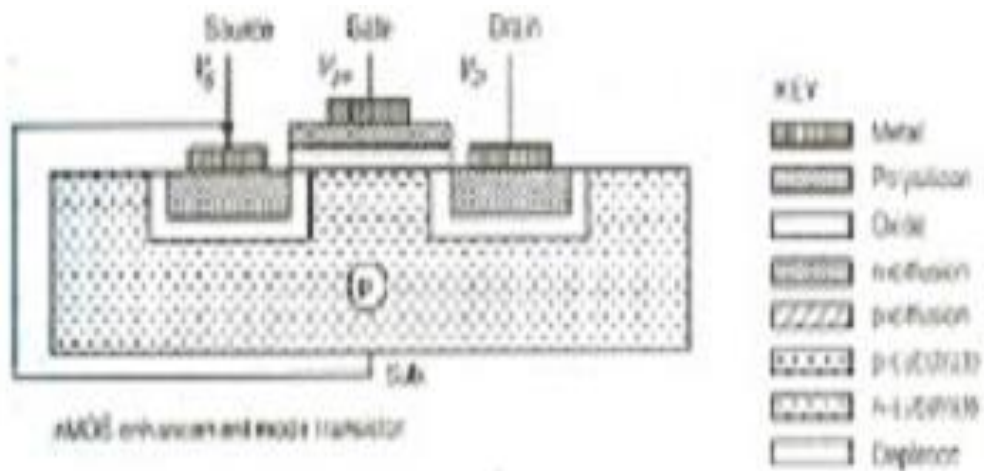


Fig 5. Nmos Enhancement transistor

P-Mos enhancement mode transistors:-This is normally on. A Channel of Holes can be performed by giving a -ve gate voltage. In P-Mos current is carried by holes and in N-Mos its by electrons. Since the mobility is of holes less than that of electrons P-Mos is slower.

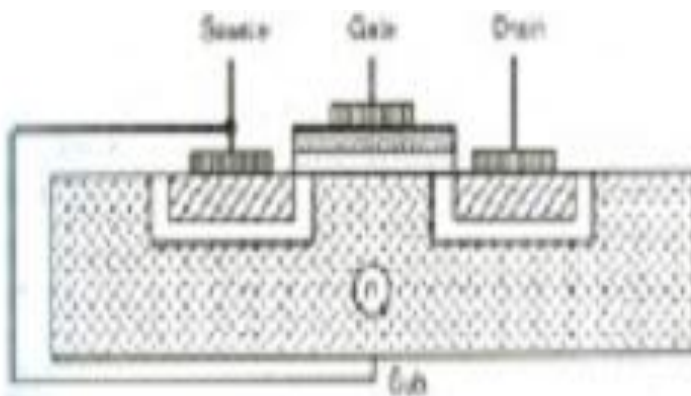


Fig 6. Pmos Enhancement transistor

N-MOS depletion mode transistor:-

This transistor is normally ON, even with $V_{gs}=0$. The channel will be implanted while fabricating, hence it is normally ON. To cause the channel to cease to exist, a $-ve$ voltage must be applied between gate and source.

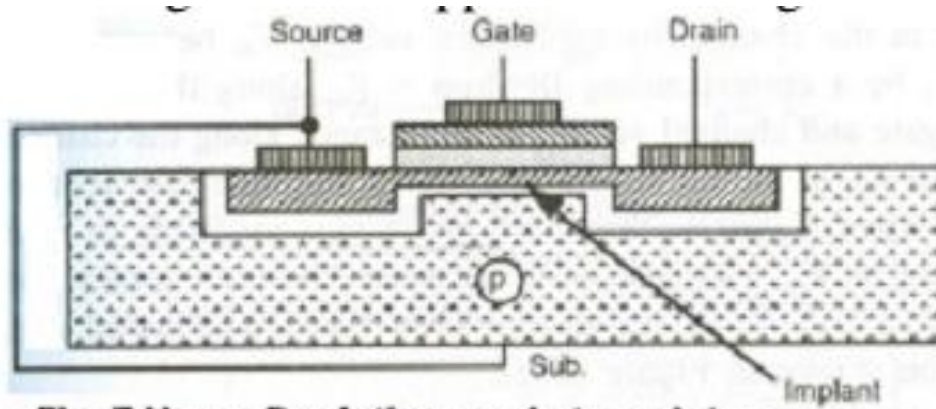
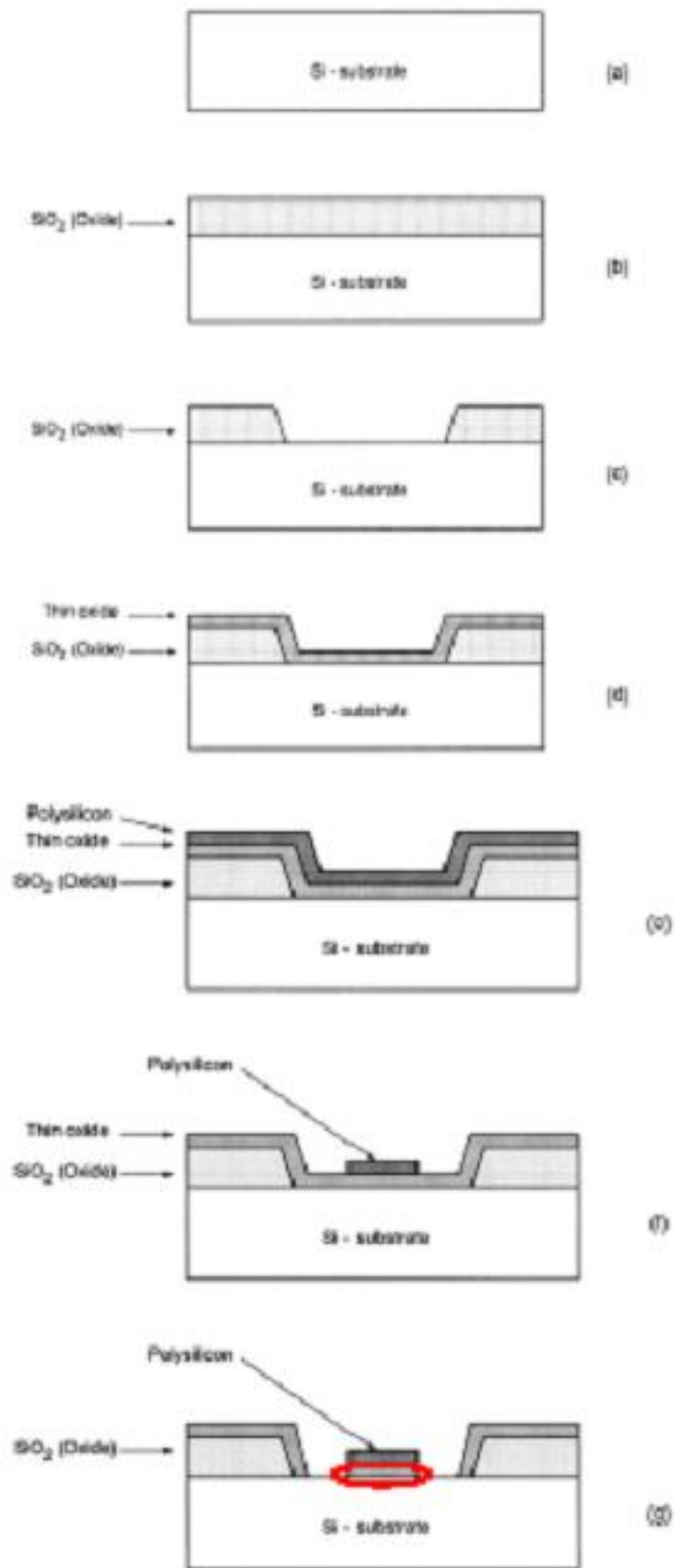


Fig. 7 Nmos Depletion mode transistor

NMOS Fabrication:



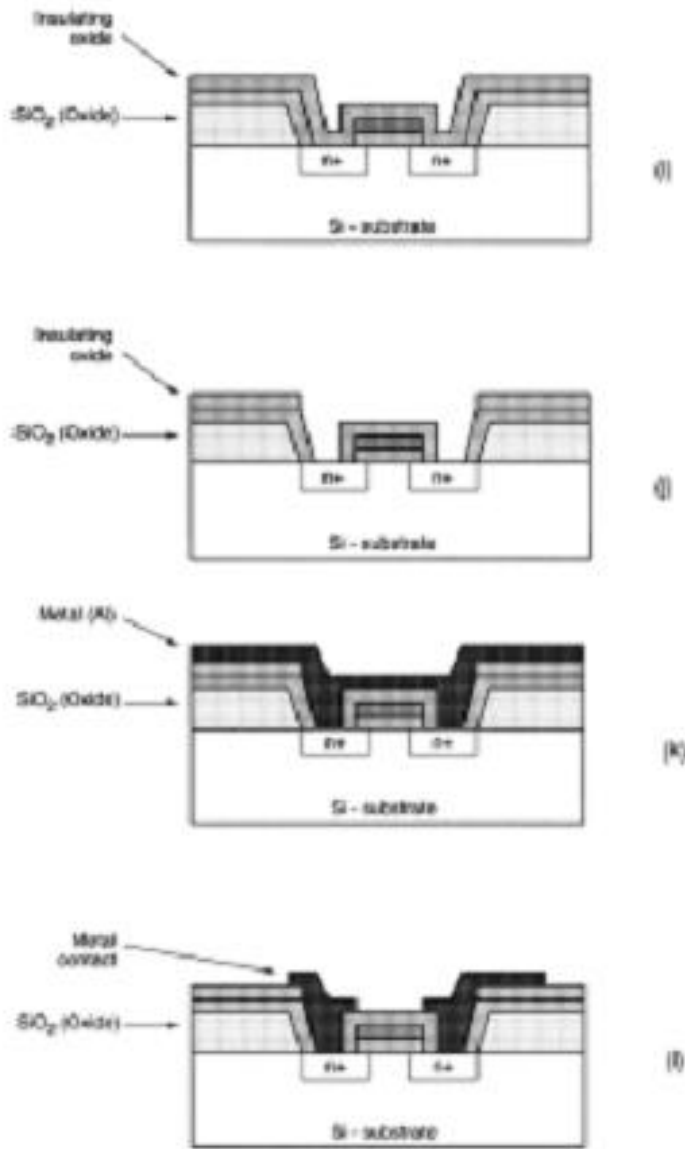


Figure 9 NMOS Fabrication process steps

The process starts with the oxidation of the silicon substrate (Fig. 9(a)), in which a relatively thick silicon dioxide layer, also called field oxide, is created on the surface (Fig. 9(b)). Then, the field oxide is selectively etched to expose the silicon surface on which the MOS transistor will be created (Fig. 9(c)).

Following this step, the surface is covered with a thin, high-quality oxide layer, which will eventually form the gate oxide of the MOS transistor (Fig. 9(d)). On top of the thin oxide, a layer of polysilicon (polycrystalline silicon) is deposited (Fig. 9(e)). Polysilicon is used both as gate electrode material for MOS transistors and also as an interconnect medium in silicon integrated circuits. Undoped polysilicon has relatively high resistivity. The resistivity of polysilicon can be reduced, however, by doping it with impurity atoms. After deposition, the polysilicon layer is patterned and etched to form the interconnects and the MOS transistor gates (Fig. 9(f)). The thin gate oxide not covered by polysilicon is

also etched away, which exposes the bare silicon surface on which the source and drain junctions are to be formed (Fig. 9(g)).

The entire silicon surface is then doped with a high concentration of impurities, either through diffusion or ion implantation (in this case with donor atoms to produce n-type doping).

Figure 9(h) shows that the doping penetrates the exposed areas on the silicon surface, ultimately creating two n-type regions (source and drain junctions) in the p-type substrate. The impurity doping also penetrates the polysilicon on the surface, reducing its resistivity.

Note that the polysilicon gate, which is patterned before doping actually defines the precise location of the channel region and, hence, the location of the source and the drain regions.

Since this procedure allows very precise positioning of the two regions relative to the gate, it is also called the self-aligned process. Once the source and drain regions are completed, the entire surface is again covered with an insulating layer of silicon dioxide (Fig. 9 (i)).

The insulating oxide layer is then patterned in order to provide contact windows for the drain and source junctions (Fig. 9 (j)).

The surface is covered with evaporated aluminum which will form the interconnects (Fig. 9 (k)).

Finally, the metal layer is patterned and etched, completing the interconnection of the MOS transistors on the surface (Fig. 9 (l)).

Usually, a second (and third) layer of metallic interconnect can also be added on top of this structure by creating another insulating oxide layer, cutting contact (via) holes, depositing, and patterning the metal.

CMOS fabrication: When we need to fabricate both nMOS and pMOS transistors on the same substrate we need to follow different processes. The three different processes are, P-well process, N-well process and Twin tub process.

P-WELL PROCESS:

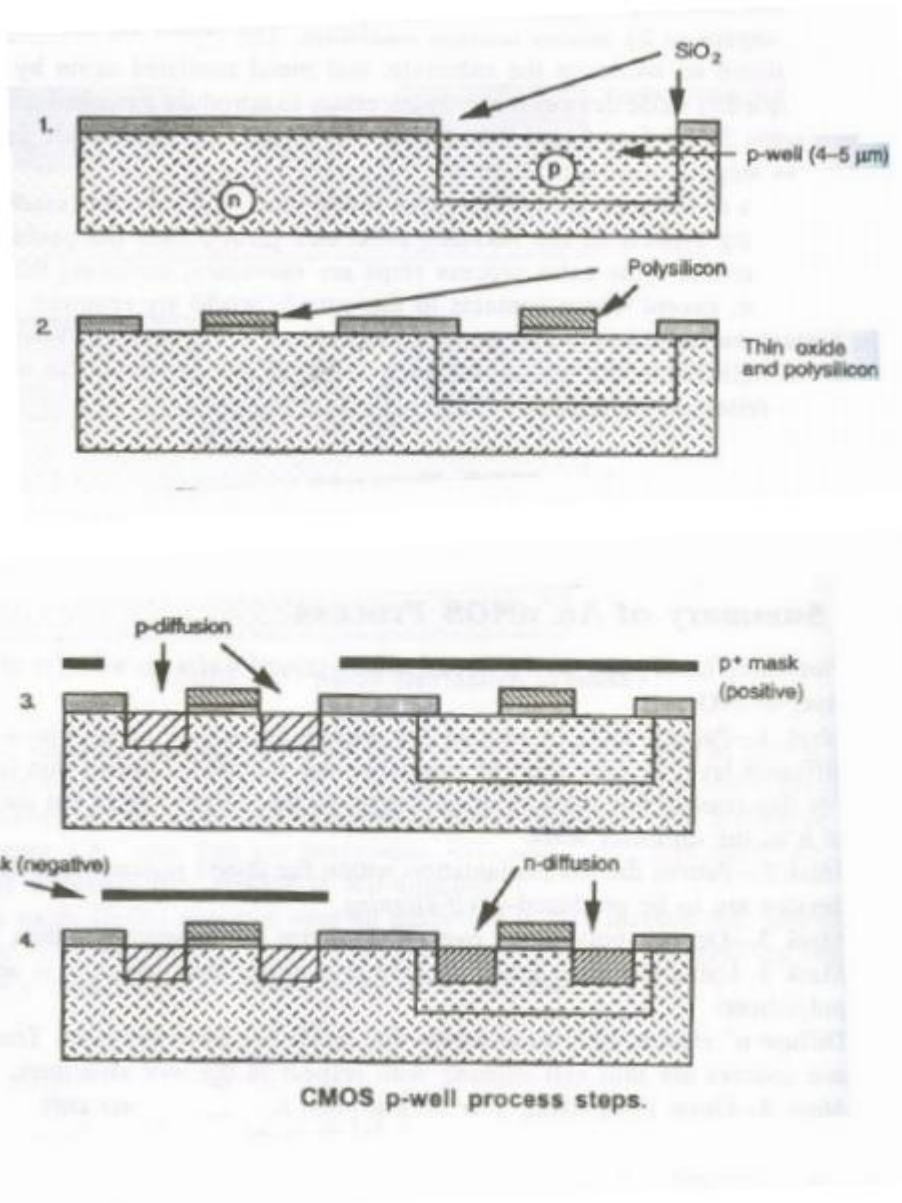


Figure 10 CMOS Fabrication (P-WELL) process steps

The p-well process starts with a n type substrate. The n type substrate can be used to implement the pMOS transistor, but to implement the nMOS transistor we need to provide a p-well, hence we have provided heplace for both n and pMOS transistor on the same n-type substrate.

Mask sequence.

Mask 1 defines the areas in which the deep p-well diffusion takes place.

Mask 2: It defines the thin oxide region (where the thick oxide is to be removed or

stripped and thin oxide grown)

Mask 3: It's used to pattern the polysilicon layer which is deposited after thin oxide.

Mask 4: A p+ mask (anded with mask 2) to define areas where p-diffusion is to take place.

Mask 5: We are using the -ve form of mask 4 (p+ mask) It defines where n-diffusion is to take place.

Mask 6: Contact cuts are defined using this mask.

Mask 7: The metal layer pattern is defined by this mask.

Mask 8: An overall passivation (overglass) is now applied and it also defines openings for accessing pads.

The cross section below shows the CMOS pwell inverter.

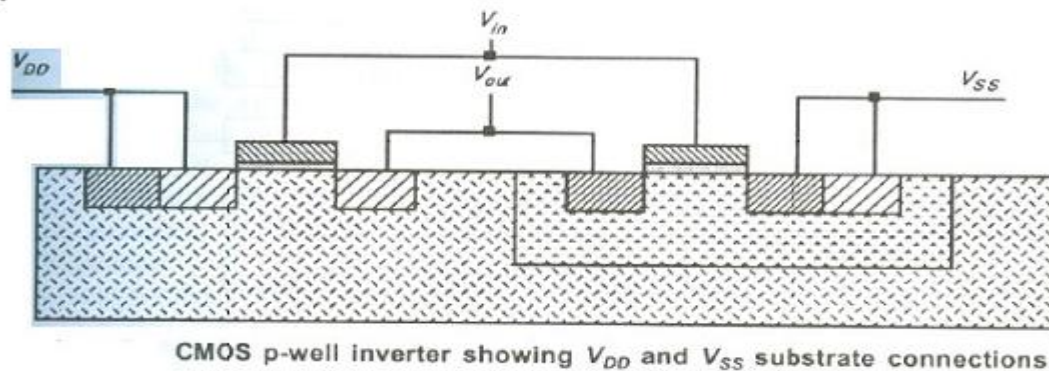


Figure 11 CMOS inverter (P-WELL)

Twin-tub process:

Here we will be using both p-well and n-well approach. The starting point is a n-type material and then we create both n-well and p-well region. To create the both well we first go for the epitaxial process and then we will create both wells on the same substrate.

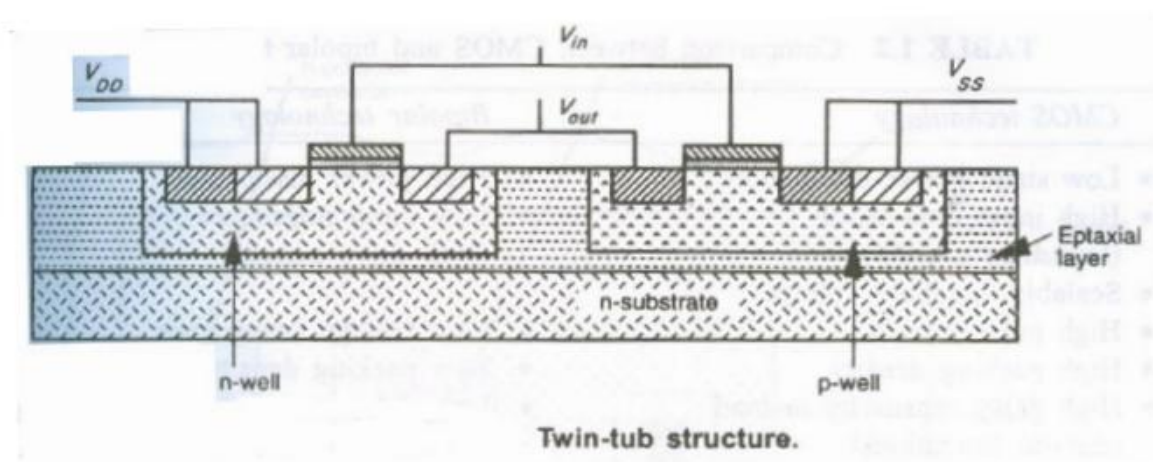


Figure 13 CMOS twin-tub inverter

Bi-CMOS technology: - (Bipolar CMOS)

The driving capability of MOS transistors is less because of limited current sourcing and sinking capabilities of the transistors. To drive large capacitive loads we can think of Bi-Cmos technology.

This technology combines Bipolar and CMOS transistors in a single integrated circuit, by retaining benefits of bipolar and CMOS, BiCMOS is able to achieve VLSI circuits with speed-power-density performance previously unattainable with either technology individually.

Characteristics of CMOS Technology

- Lower static power dissipation
- Higher noise margins
- Higher packing density – lower manufacturing cost per device
- High yield with large integrated complex functions
- High input impedance (low drive current)
- Scalable threshold voltage
- High delay sensitivity to load (fan-out limitations)
- Low output drive current (issue when driving large capacitive loads)
- Low transconductance, where transconductance, $g_m \propto V_{in}$
- Bi-directional capability (drain & source are interchangeable)
- A near ideal switching device

Characteristics of Bipolar Technology

- Higher switching speed
- Higher current drive per unit area, higher gain
- Generally better noise performance and better high frequency characteristics
- Better analogue capability
- Improved I/O speed (particularly significant with the growing importance of package limitations in high speed systems).
- high power dissipation
- lower input impedance (high drive current)
- low voltage swing logic
- low packing density
- low delay sensitivity to load
- high g_m ($g_m \propto V_{in}$)
- high unity gain band width (ft) at low currents
- essentially unidirectional

From the two previous paragraphs we can get a comparison between bipolar and CMOS technology.

The diagram given below shows the cross section of the BiCMOS process which uses an npn transistor.

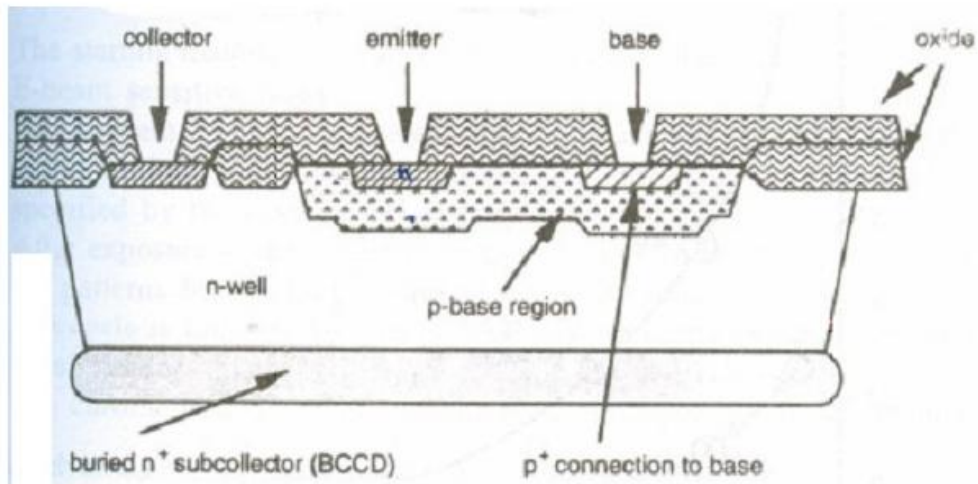


Figure 14 Cross section of BiCMOS process

The figure below shows the layout view of the BiCMOS process.

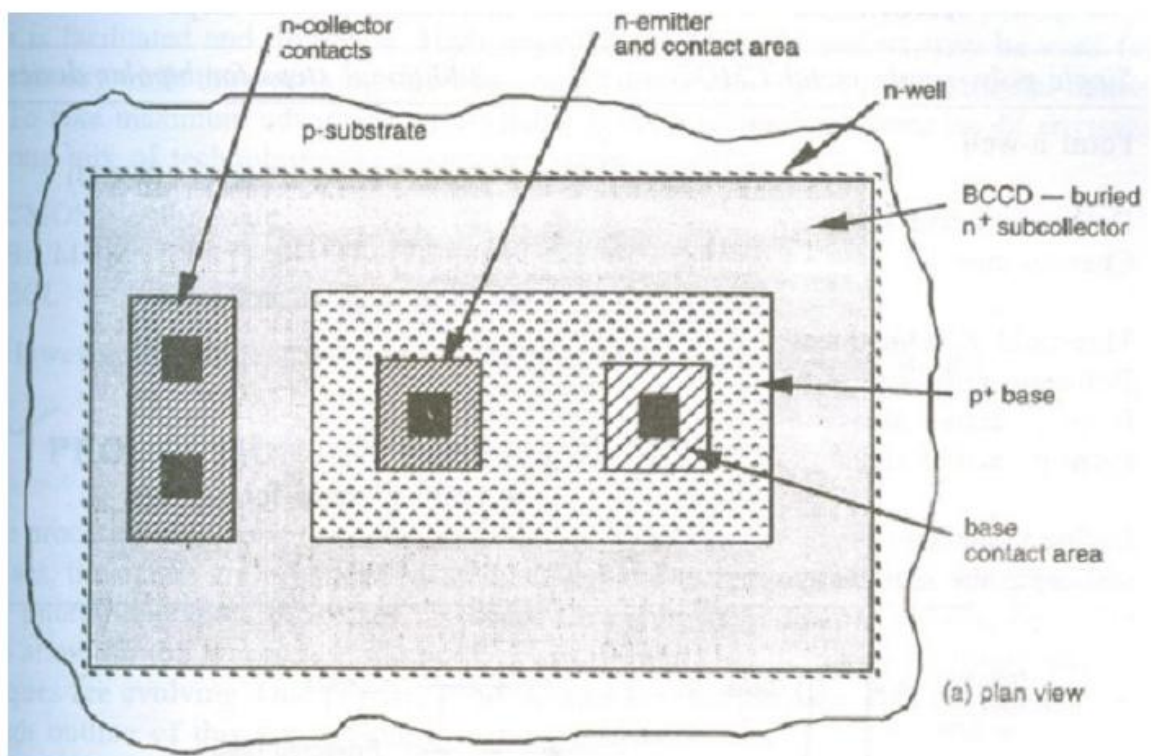


Fig.15. Layout view of BiCMOS process

BASIC ELECTRICAL PROPERTIES:

1. Linear Region:

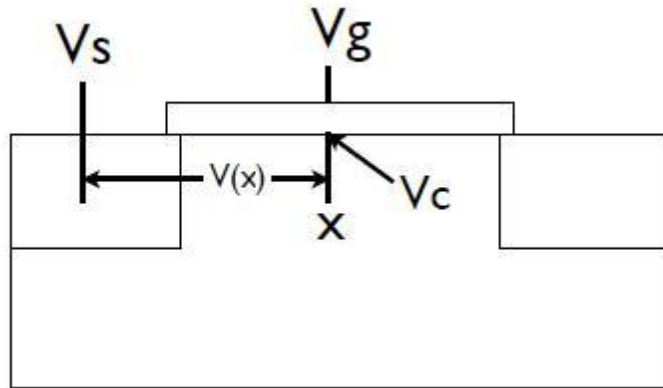


Figure 1. Concentration Contours in Linear Region. A uniform narrow channel exists.

KVL:

$$\begin{aligned}V_g - V_s &= V_g - V_c + V_c - V_s \\V_g - V_s &= V_{GS} \\V_g - V_c &= V_{GC} \\V_c - V_s &= V(x) \\V_{GS} &= V_{GC} + V(x) \text{ or } V_{GS} - V(x) = V_{GC}\end{aligned}$$

Total charge density at x on capacitor(COX) is QT (x):

Total charge density at x on capacitor(C_{OX}) is $Q_T(x)$:

$$Q_T(x) = V_{GS}C_{OX} = (V_{GS} - V(x))C_{OX}$$

$$Q_T(x) = Q(x)_{mobile} + Q(x)_{depletion}$$

$Q(x)_{mobile}$ =mobile electron charge in channel at x

$$Q(x)_{mobile} = [V_{GS} - V(x) - V_{TH}]C_{OX}$$

Use mobile charge to get current:

$$J_n = q\mu nE + qD_n \frac{dn}{dx} = q\mu nE \text{ (no diffusion current in the channel)}$$

$$qn(x) = Q(x)_{mobile} = Q_m(x)$$

$$J_n = Q_m(x)\mu E, \text{ but } E = -\frac{dV}{dx}$$

$$J_n = -Q_m(x)\mu \frac{dV}{dx}, \text{ substitute for } Q_m(x)$$

$J_n = \mu C_{ox}(V_{GS} - V(x) - V_{TH})\frac{dV}{dx}$, separate variables and neglect (-) sign. Consider only the magnitude.

$$J_n dx = \mu C_{ox}[(V_{GS} - V_{TH}) - V(x)]dV$$

Due to continuity, $J_n = \text{constant}$ (no hole current or no generation, recombination). Integrating from source to drain or from $x=0$ to $x=L$, where L =gate length:

$$J_n \int_0^L dx = \mu C_{ox} \int_{V(0)}^{V(L)} [(V_{GS} - V_{TH}) - V(x)]dV$$

$$V(L) = V_{DS}, V(0)=0$$

$$J_n \int_0^L dx = \mu C_{ox} \int_0^{V_{DS}} [(V_{GS} - V_{TH}) - V(x)]dV$$

$$J_n L = \mu C_{ox} [(V_{GS} - V_{TH})V - \frac{V^2}{2}]_0^{V_{DS}}$$

$$J_n = \frac{\mu C_{ox}}{L} [(V_{GS} - V_{TH})V_{DS} - \frac{V_{DS}^2}{2}]$$

$$I_D = J_n W \text{ (W=Device Width)}$$

J_n for channel is Amp/cm since $Q_m = \text{Charge/cm}^2$

$$I_D \text{ for Linear Region: } I_D = \mu C_{ox} \frac{W}{L} [(V_{GS} - V_{TH})V_{DS} - \frac{V_{DS}^2}{2}]$$

2. Saturation Region

When $V_{DS} = (V_{GS} - V_{TH})$ channel pinches off. This means that the channel current near the drain spreads out and the channel near drain can be approximated as the depletion region. After this occurs, at $V_{DS} = (V_{GS} - V_{TH})$, if you make V_{DS} larger, the current I_D does not change (to zero approximation). This is because any additional V_{DS} you add will get dropped across the depletion region and won't change the current I_D .

So for $V_{DS} = (V_{GS} - V_{TH})$ we find I_D by setting $V_{DS} = (V_{GS} - V_{TH})$ substituting into the linear equation.

$$I_D = \mu C_{ox} \frac{W}{L} [(V_{GS} - V_{TH})(V_{GS} - V_{TH}) - \frac{(V_{GS} - V_{TH})^2}{2}]$$

I_D for Saturation Region:

$$I_D = \mu C_{ox} \frac{W}{2L} (V_{GS} - V_{TH})^2$$

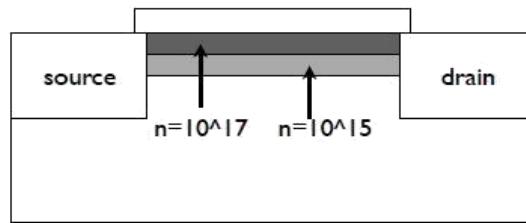


FIGURE 2. Concentration Contours in Linear Region. A uniform narrow channel exists.

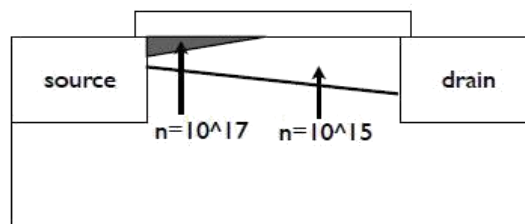


FIGURE 3. Concentration Contours in Saturation Region. Channel narrow near source and spreads out and widens near drain, said to be “pinched off”.

Transconductance:

Transconductance is a property of certain electronic components. Conductance is the reciprocal of resistance; transconductance is the ratio of the current change at the output port to the voltage change at the input port. It is written as g_m . For direct current, transconductance is defined as follows:

$$g_m = \frac{\Delta I_{\text{out}}}{\Delta V_{\text{in}}}$$

For small signal alternating current, the definition is simpler:

$$g_m = \frac{i_{\text{out}}}{v_{\text{in}}}$$

MOS transistor theory

Introduction:

A MOS transistor is a majority-carrier device, in which the current in a conducting channel between the source and the drain is modulated by a voltage applied to the gate.

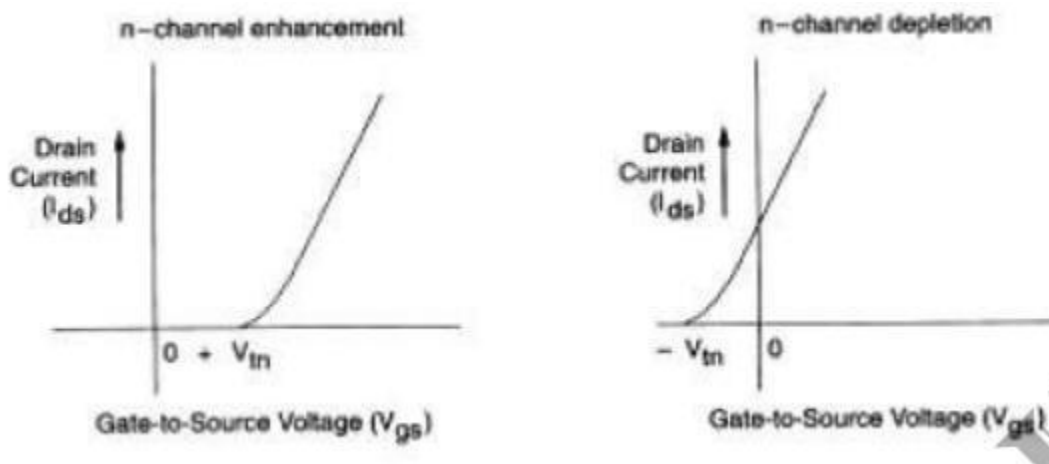
NMOS(n-type MOS transistor):

- (1) Majority carrier = electrons
- (2) A positive voltage applied on the gate with respect to the substrate enhances the number of electrons in the channel and hence increases the conductivity of the channel.
- (3) If gate voltage is less than a threshold voltage V_t , the channel is cut-off (very low current between source & drain).

PMOS(p-type MOS transistor):

- (1) Majority carrier = holes
- (2) Applied voltage is negative with respect to substrate.

Relationship between V_{gs} and I_{ds} , for a fixed V_{ds} :



MOS equations (Basic DCEquations):

Three MOS operating regions are: Cutoff or subthreshold region, linear region and saturation region.

The following equation describes all these three regions:

$$I_{ds} = \begin{cases} 0; & V_{gs} - V_t \leq 0 \text{ cut-off} \\ \beta \left[(V_{gs} - V_t)V_{ds} - \frac{V_{ds}^2}{2} \right]; & 0 < V_{ds} < V_{gs} - V_t \text{ linear} \\ \frac{\beta}{2} (V_{gs} - V_t)^2; & 0 < V_{gs} - V_t < V_{ds} \text{ saturation} \end{cases}$$

Where β is MOS transistor gain and it is given by $\beta = \mu' \epsilon / t_{ox} (W/L)$

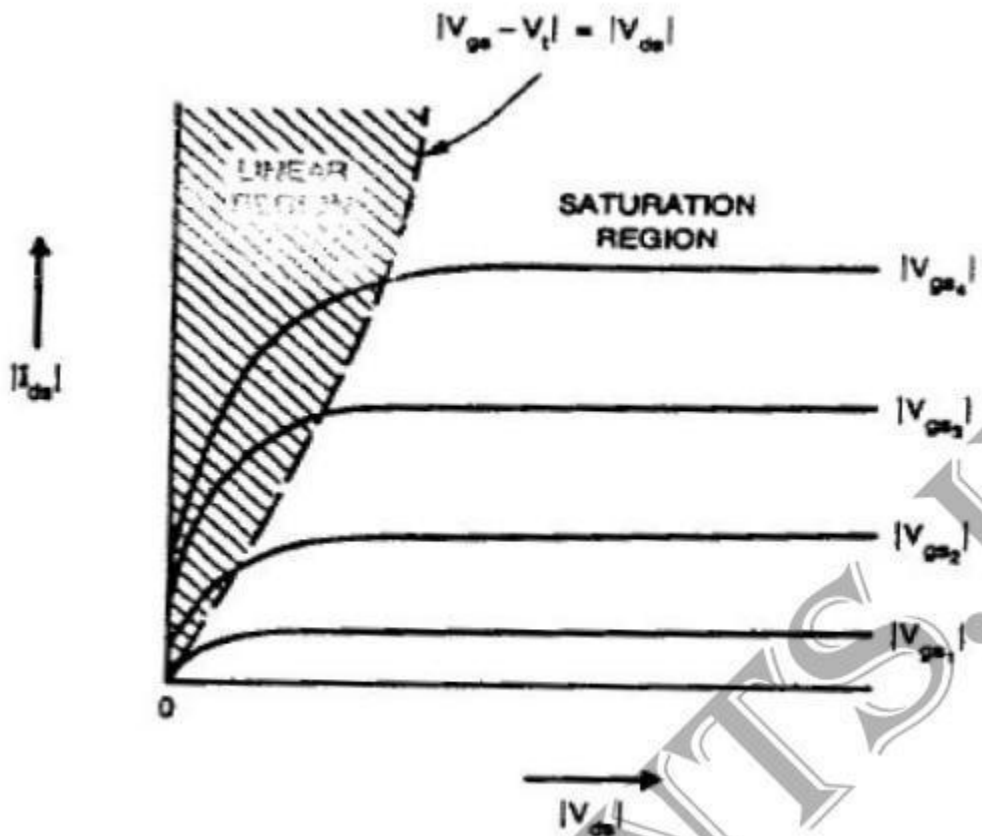
μ' is the mobility of the charge carrier

ϵ is the permittivity of the oxide layer.

t_{ox} is the thickness of the oxide layer.

W is the width of the transistor. (shown in diagram)

L is the channel length of the transistor. (shown in diagram)



VI Characteristics of MOSFET

Second Order Effects:

Following are the list of second order effects of MOSFET:

Threshold voltage –Body effect

Sub-threshold region

Channel length modulation
 Mobility variation
 Fowler_Nordheim Tunneling
 Drain Punch-through
 Impact Ionization –Hot Electrons

Threshold voltage –Bodyeffect

The change in the threshold voltage of a MOSFET, because of the voltage difference between body and source is called body effect. The expression for the threshold voltage is given by the following expression.

$$V_t = V_{t(0)} + \gamma [(V_{sb} + 2\Phi_F)^{1/2} - (2\Phi_F)^{1/2}]$$

where

- V_t is the threshold voltage,
- V_{t(0)} is the threshold voltage without body effect
- γ is the body coefficient factor
- Φ_F is the fermi potential
- V_{sb} is the potential difference between source and substrate.

If V_{sb} is zero, then V_t = V_{t(0)} that means the value of the threshold voltage will not be changed. Therefore, we short circuit the source and substrate so that, V_{sb} will be zero.

Sub-threshold region:

For V_{gs} < V_t also we will get some value of Drain current this is called as Subthreshold current And the region is called as Sub-threshold region.

Channel length modulation:

The channel length of the MOSFET is changed due to the change in the drain to source voltage. This effect is called as the channel length modulation. The effective channel length & the value of the drain current considering channel length modulation into effect is given by,

$$I_{ds} = \frac{\beta}{2} ((V_{gs} - V_t)^2 (1 + \lambda V_{ds}))$$

$$L_{eff} = L - \sqrt{\frac{2\epsilon_0 \epsilon_{Si}}{q_i N}} (V_{ds} - [V_{gs} - V_t]).$$

Where λ is the channel length modulation factor.

Mobility: Mobility is defined as the ease with which the charge carriers drift in the substrate material .

Mobility decreases with increase in doping concentration and increase in temperature. Mobility is the ratio of average carrier drift velocity and electric field. Mobility is represented by the symbol μ .

Fowler Nordhiem tunneling:

When the gate oxide is very thin there can be a current between gate and source or drain by electron tunneling through the gate oxide. This current is proportional to the area of the gate of the transistor.

Drain punch-through:

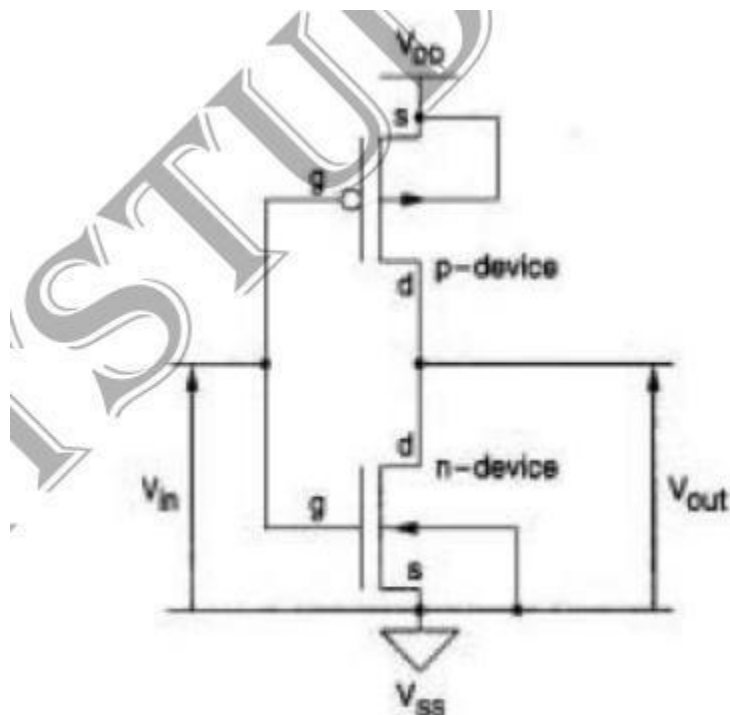
When the drain is a high voltage, the depletion region around the drain may extend to the source, causing the current to flow even if gate voltage is zero. This is known as Punch-through condition.

Impact Ionization-hot electrons:

When the length of the transistor is reduced, the electric field at the drain increases. The field can become so high that electrons are imparted with enough energy we can term them as hot. These hot electrons impact the drain, dislodging holes that are then swept toward the negatively charged substrate and appear as a substrate current. This effect is known as Impact Ionization.

1.9MOSModels MOS model includes the Ideal Equations, Second-order Effects plus the additional Curve-fitting parameters. Many semiconductor vendors expend a lot of effects to model the devices they manufacture. (Standard: Level 3 SPICE). Main SPICE DC parameters in level 1,2,3 in $1\mu\text{m}$ -well CMOS process.

CMOS INVERTER CHARACTERISTICS



CMOS inverters (Complementary MOSFET 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.

A CMOS inverter contains a PMOS and a NMOS transistor connected at the drain and gate terminals, a supply voltage V_{DD} at the PMOS source terminal, and a ground connected at the NMOS source terminal, where V_{IN} is connected to the gate terminals and V_{OUT} 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 than a regular resistor - MOSFET inverter.

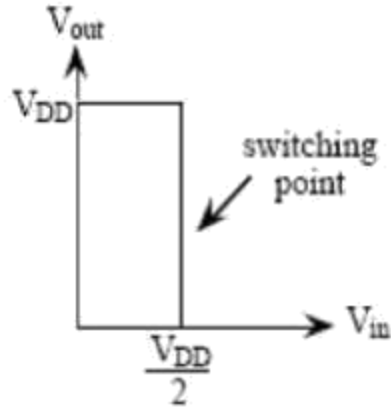
As the voltage at the input of the CMOS device varies between 0 and V_{DD} , the state of the NMOS and PMOS varies accordingly. If we model each transistor as a simple switch activated by V_{IN} , the inverter's operations can be seen very easily:

MOSFET	Condition	on	State	of
	MOSFET		MOSFET	
NMOS	$V_{gs} < V_{tn}$		OFF	
NMOS	$V_{gs} > V_{tn}$		ON	
PMOS	$V_{sg} < V_{tp}$		OFF	
PMOS	$V_{sg} > V_{tp}$		ON	

The table given, explains when each transistor is turning on and off. When V_{IN} is low, the NMOS is "off", while the PMOS stays "on": instantly charging V_{OUT} to logic high. When V_{IN} is high, the NMOS is "on" and the PMOS is "off": taking the voltage at V_{OUT} to logic low.

Inverter DC Characteristics:

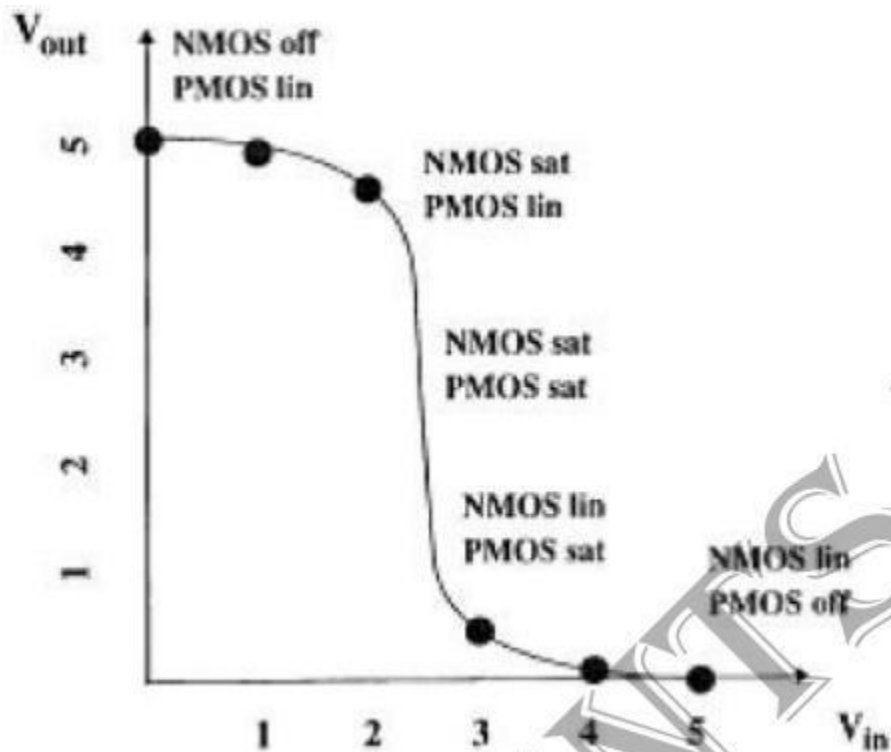
Before we study the DC characteristics of the inverter we should examine the ideal characteristics of inverter which is shown below. The characteristic shows that when input is zero output will be high and vice versa.



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. Graphical Derivation of Inverter DC Characteristics:

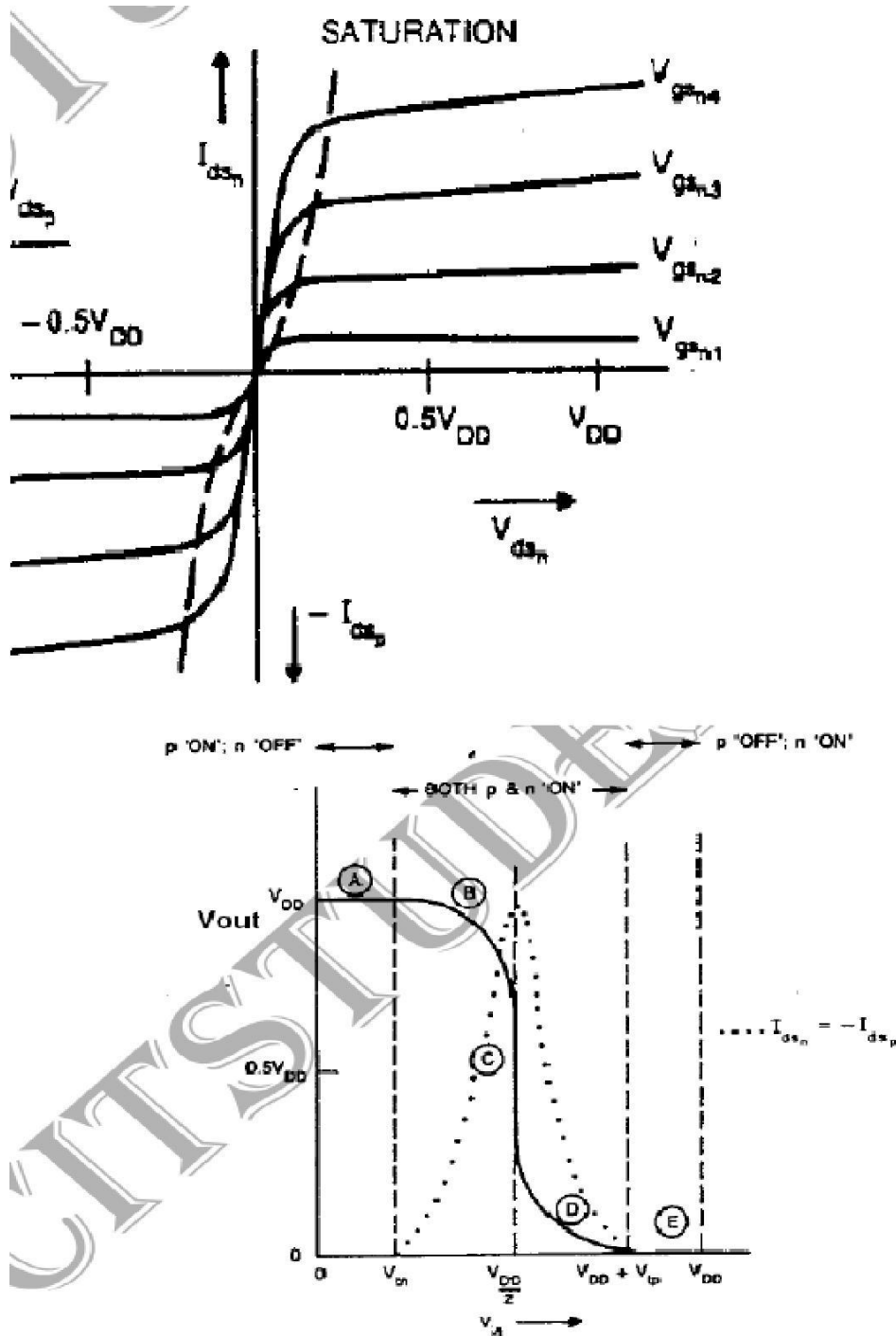
The actual characteristics are drawn by plotting the values of output voltage for different values of the input voltage. We can also draw the characteristics, starting with the VI-characteristics of PMOS and NMOS characteristics.



Actual Characteristics of an Inverter.

Graphical Derivation of Inverter DC Characteristics:

The actual characteristics are drawn by plotting the values of output voltage for different values of the input voltage. We can also draw the characteristics, starting with the VI-characteristics of PMOS and NMOS characteristics



CMOS Inverter Dc Characteristics.

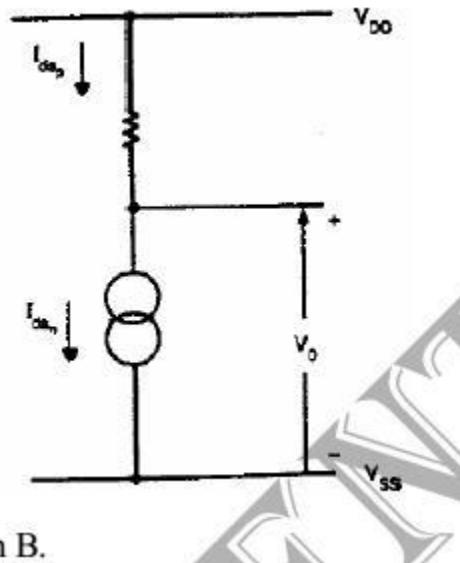
Figure shows five regions namely region A, B, C, D & E. also we have shown a dotted curve which is the current that is drawn by the inverter.

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 in region B is given below.



: 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

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

The expression for the PMOS current is

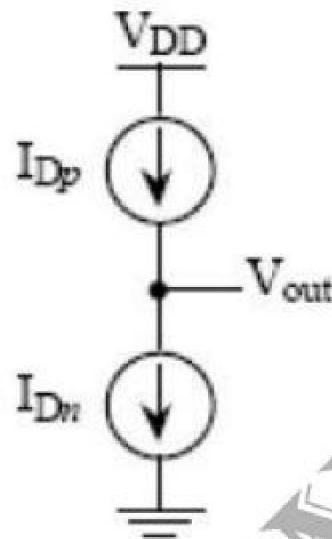
$$I_{dsp} = -\beta_p \left[(V_{in} - V_{DD} - V_{tp})(V_O - V_{DD}) - \frac{1}{2}(V_O - V_{DD})^2 \right]$$

The expression for the voltage V_O can be written as

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

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 an inverter. The corresponding equations are as follows:



: 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,

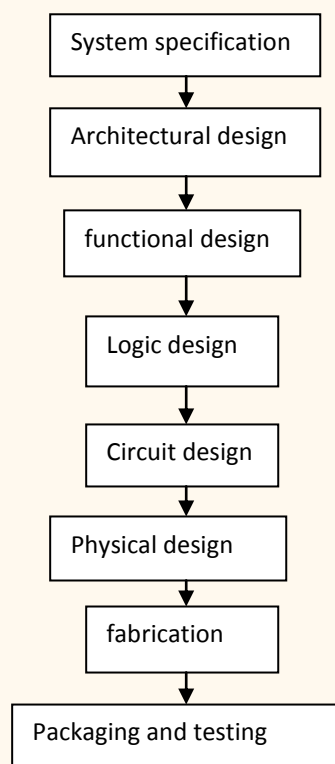
$$V_{in} = \frac{V_{DD} + V_{t_p} + V_{t_n} \sqrt{\frac{\beta_n}{\beta_p}}}{1 + \sqrt{\frac{\beta_n}{\beta_p}}}$$

UNIT II

VLSI CIRCUIT DESIGN PROCESSES

Vlsi design flow:

The VLSI design cycle starts with a formal specification of a VLSI chip, follows a series of steps, and eventually produces a packaged chip. A typical design cycle may be represented by the flow chart shown in Figure. Our emphasis is on the physical design step of the VLSI design cycle. However, to gain a global perspective, we briefly outline all the steps of the VLSI design cycle.



1.System Specification:

The first step of any design process is to lay down the specifications of the system. System specification is a high level representation of the system. The factors to be considered in this process include: performance, functionality, and physical dimensions (size of the die (chip)). The fabrication technology and design techniques are also considered.

The specification of a system is a compromise between market requirements, technology and economical viability. The end results are specifications for the size, speed, power, and functionality of the VLSI system.

2. Architectural Design:

The basic architecture of the system is designed in this step. This includes, such decisions as RISC (Reduced Instruction Set Computer) versus CISC (Complex Instruction Set Computer), number of ALUs, Floating Point units, number and structure of pipelines, and size of caches among others.

The outcome of architectural design is a Micro-Architectural Specification (MAS). While MAS is a textual (English like) description, architects can accurately predict the performance, power and die size of the design based on such a description.

3. Behavioral or Functional Design:

In this step, main functional units of the system are identified. This also identifies the interconnect requirements between the units. The area, power, and other parameters of each unit are estimated.

The behavioral aspects of the system are considered without implementation specific information. For example, it may specify that a multiplication is required, but exactly in which mode such multiplication may be executed is not specified. We may use a variety of multiplication hardware depending on the speed and word size requirements. The key idea is to specify behavior, in terms of input, output and timing of each unit, without specifying its internal structure.

The outcome of functional design is usually a timing diagram or other relationships between units. This information leads to improvement of the overall design process and reduction of the complexity of subsequent phases. Functional or behavioral design provides quick emulation of the system and allows fast debugging of the full system. Behavioral design is largely a manual step with little or no automation help available.

4. Logic Design:

In this step the control flow, word widths, register allocation, arithmetic operations, and logic operations of the design that represent the functional design are derived and tested.

This description is called Register Transfer Level (RTL) description. RTL is expressed in a Hardware Description Language (HDL), such as VHDL or Verilog. This description can be used in simulation and verification. This description consists of Boolean expressions and timing information. The Boolean expressions are minimized to achieve the smallest logic design which conforms to the functional design. This logic design of the system is simulated and tested to verify its correctness. In some special cases, logic design

can be automated using *high level synthesis* tools. These tools produce a RTL description from a behavioral description of the design.

5. Circuit Design:

The purpose of circuit design is to develop a circuit representation based on the logic design. The Boolean expressions are converted into a circuit representation by taking into consideration the speed and power requirements of the original design. *Circuit Simulation* is used to verify the correctness and timing of each component.

The circuit design is usually expressed in a detailed circuit diagram. This diagram shows the circuit elements (cells, macros, gates, transistors) and interconnection between these elements. This representation is also called a *netlist*. Tools used to manually enter such description are called *schematic capture tools*. In many cases, a netlist can be created automatically from logic (RTL) description by using *logic synthesis* tools.

6. Physical Design:

In this step the circuit representation (or netlist) is converted into a geometric representation. As stated earlier, this geometric representation of a circuit is called a *layout*. Layout is created by converting each logic component (cells, macros, gates, transistors) into a geometric representation (specific shapes in multiple layers), which perform the intended logic function of the corresponding component. Connections between different components are also expressed as geometric patterns typically lines in multiple layers.

The exact details of the layout also depend on design rules, which are guidelines based on the limitations of the fabrication process and the electrical properties of the fabrication materials. Physical design is a very complex process and therefore it is usually broken down into various sub-steps. Various verification and validation checks are performed on the layout during physical design.

In many cases, physical design can be completely or partially automated and layout can be generated directly from netlist by *Layout Synthesis* tools. Layout synthesis tools, while fast, do have an area and performance penalty, which limit their use to some designs. Manual layout, while slow and manually intensive, does have better area and performance as compared to synthesized layout. However this advantage may dissipate as larger and larger designs may undermine human capability to comprehend and obtain globally optimized solutions.

7. Fabrication:

After layout and verification, the design is ready for fabrication. Since layout data is typically sent to fabrication on a tape, the event of release of data is called *Tape Out*. Layout data is converted (or fractured) into photo-lithographic masks, one for each layer. Masks identify spaces on the wafer, where certain materials need to be deposited, diffused or even removed. Silicon crystals are grown and sliced to produce wafers. Extremely small dimensions of VLSI devices require that the wafers be polished to near perfection. The fabrication process consists of several steps involving deposition, and diffusion of various materials on the wafer. During each step one mask is used. Several dozen masks may be used to complete the fabrication process.

A large wafer is 20 cm (8 inch) in diameter and can be used to produce hundreds of chips, depending of the size of the chip. Before the chip is mass produced, a prototype is made and tested. Industry is rapidly moving towards a 30 cm (12 inch) wafer allowing even more chips per wafer leading to lower cost per chip.

8. Packaging, Testing and Debugging:

Finally, the wafer is fabricated and diced into individual chips in a fabrication facility. Each chip is then packaged and tested to ensure that it meets all the design specifications and that it functions properly. Chips used in Printed Circuit Boards (PCBs) are packaged in Dual In-line Package (DIP), Pin Grid Array (PGA), Ball Grid Array (BGA), and Quad Flat Package (QFP). Chips used in Multi-Chip Modules (MCM) are not packaged, since MCMs use bare or naked chips.

Mos 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:

Stick diagrams may be used to convey layer information through use of a color code.

For example:

- n-diffusion -- green
- poly – red
- blue -- metal yellow
- --implant black –
- contact areas

2 Encodings For Nmos Process:

COLOR	STICK ENCODING	LAYERS	MASK LAYOUT ENCODING	O/F LAYER
GREEN		n-diffusion (n+ active) Thinox*		ND
RED		Polysilicon		NP
BLUE		Metal 1		NM
BLACK		Contact out		NC
GRAY	NOT APPLICABLE	Overglass		NG
nMOS ONLY YELLOW		Implant		NI
-MOS ONLY BROWN		Buried contact		NB
FEATURE	FEATURE (STICK)	FEATURE (SYMBOL)	FEATURE (MASK)	
n-type enhancement mode transistor				
Transistor length to width ratio L:W should be shown.				
n-type depletion mode transistor nMOS only				
Source, drain and gate labeling will not normally be shown.				

Figure 1: NMOS encodings

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

Figure1 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:

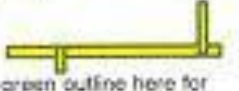
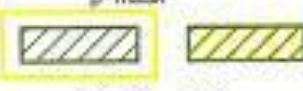


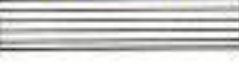


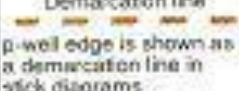


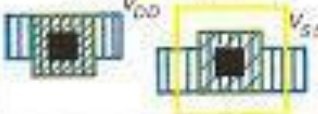

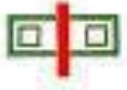

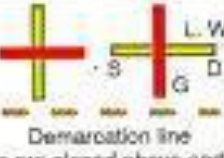

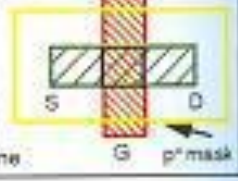
COLOR	STICK ENCODING	LAYERS	MASK LAYOUT ENCODING	CIF LAYER
GREEN	Encoding as in Color plate 1(a)	n-diffusion (n ⁺ active) Thin ^{ox} *	* Thin ^{ox} = n-diff. + p-diff. + transistor channels	CAA or CNA
RED		Polysilicon	Encoding as in Color plate 1(a)	CFF
BLUE		Metal 1		CNF
BLACK		Contact out		CC
GRAY		Overglass		COG
YELLOW (STICK)		p-diffusion (p ⁺ active)		CAA or CPA
YELLOW	Not shown on diagram	p ⁺ mask		CFF
DARK BLUE OR PURPLE		Metal 2		CMS
BLACK		VIA		CVA
BROWN		p-well		CPW
BLACK		V _{DD} or V _{SS} contact		CC
FEATURE	FEATURE (STICK)	FEATURE (SYMBOL)	FEATURE (MASK)	
n-type enhancement mode transistor (as in Color plate 1(a))				
p-type enhancement mode transistor				
Note: p-type transistors are placed above and n-type below the demarcation line				

Figure 2: CMOS encodings

Figure 2 shows when a n-transistor is formed: a transistor is formed when a green line (n⁺ diffusion) crosses a red line (poly) completely.

Figure 2 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:



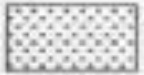


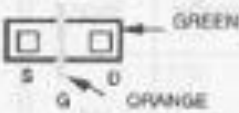



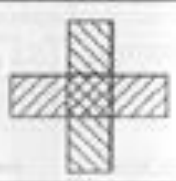


COLOR	STICK ENCODING	LAYERS	MASK LAYOUT ENCODING	CIF LAYER
ORANGE	MONOCHROME	Polysilicon 2	MONOCHROME 	CPS
SEE COLOR PLATE 1(f)		Bipolar npn transistor	see Figure 3-13(f)	Not applicable
PINK	Not separately encoded	p-base of bipolar npn transistor		CBA
PALE GREEN	Not separately encoded	Buried collector of bipolar npn transistor	n-well 	CCA
FEATURE	FEATURE (STICK) (MONOCHROME)	FEATURE (SYMBOL) (MONOCHROME)	FEATURE (MASK) (MONOCHROME)	
<i>n</i> -type enhancement poly 2 transistor Transistor length to width ratio L:W may be shown.				
<i>p</i> -type enhancement poly 2 transistor Note: <i>p</i> -type transistors are placed above and <i>n</i> -type transistors below the demarcation line.				
<i>npn</i> bipolar transistor			See Figure 3-13(f) and Color plate 6	

Figure 3: Bi CMOS 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 aluminium) a further thick layer of silicon dioxide with contact cuts through the silicon dioxide where 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 shortage of colored pen normally mean that both types of diffusion in CMOS are colored green and the polarity indicated by drawing a circle round by p-type transistor or from the context. Colorings for multiple layers of metal are even less standard.

There are three ways that nMOS inverter might be drawn:

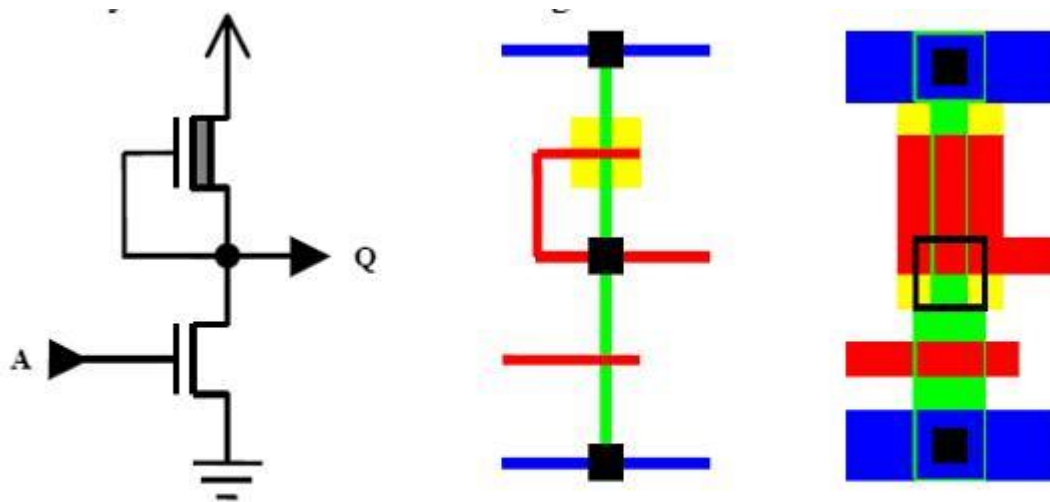


Figure 4: nMOS depletion load inverter

Figure 4 shows schematic, stick diagram and corresponding layout of nMOS depletion load inverter

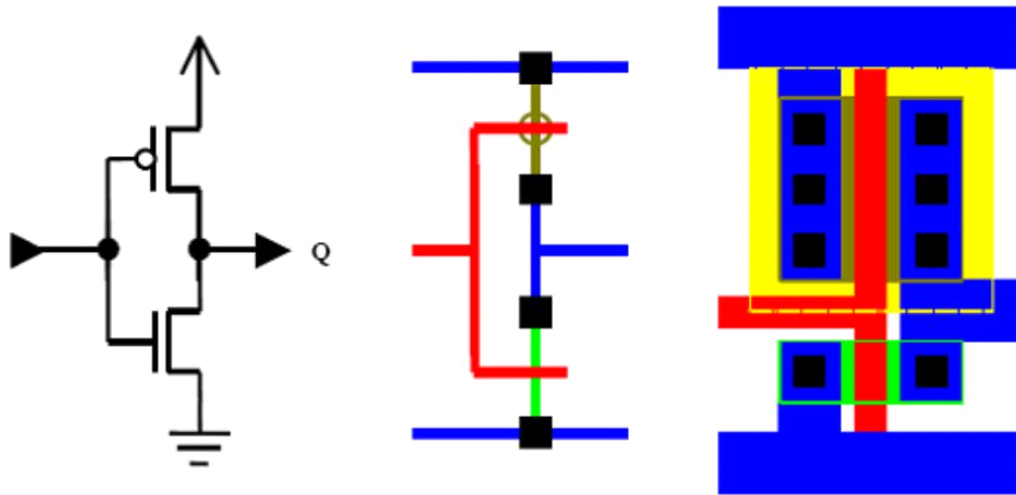


Figure 5: CMOS inverter

Figure 5 shows schematic, stick diagram and corresponding layout of CMOS inverter

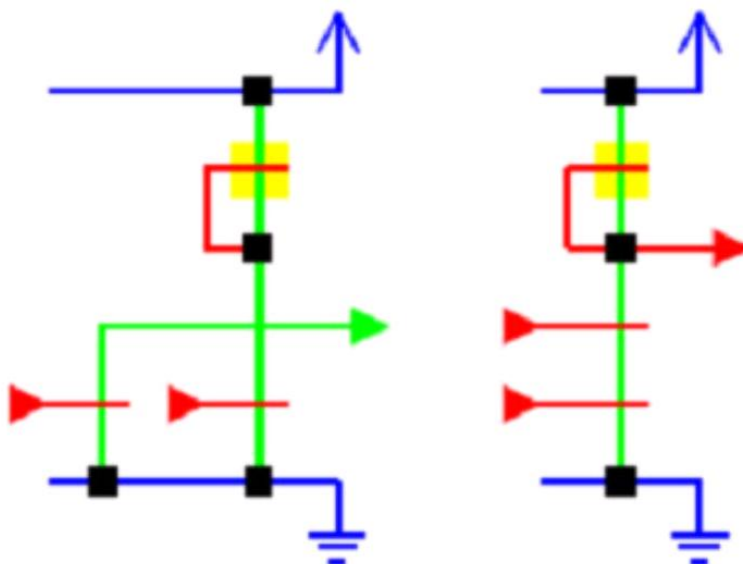


Figure 6: nMOS depletion load NAND and NOR stick diagram

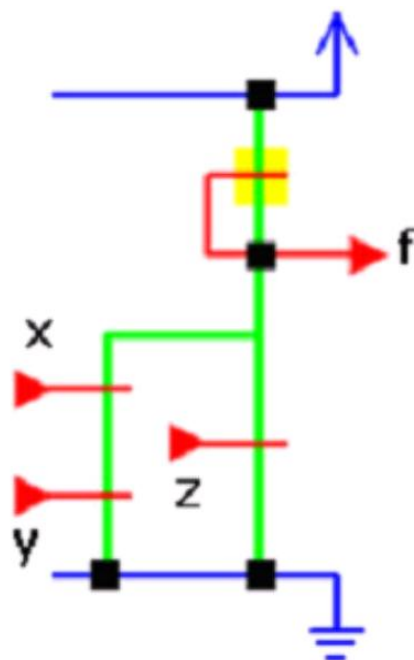


Figure 7: stick diagram of a given function f .

Figure 7 shows the stick diagram nMOS implementation of the function $f=[(xy)+z]'$

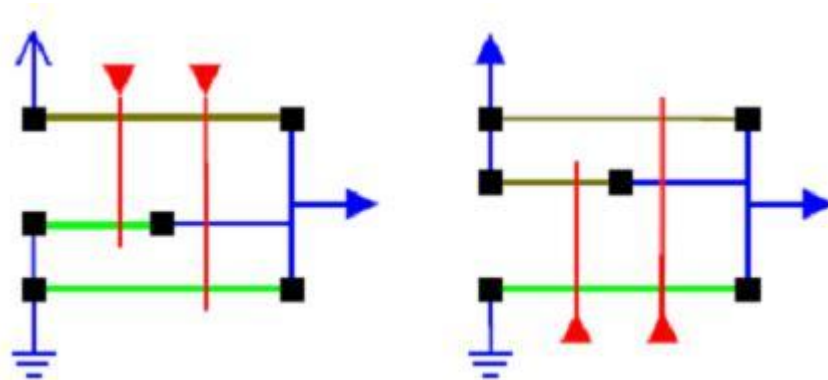


Figure 8: stick diagram of CMOS NAND and NOR

Figure 8 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.

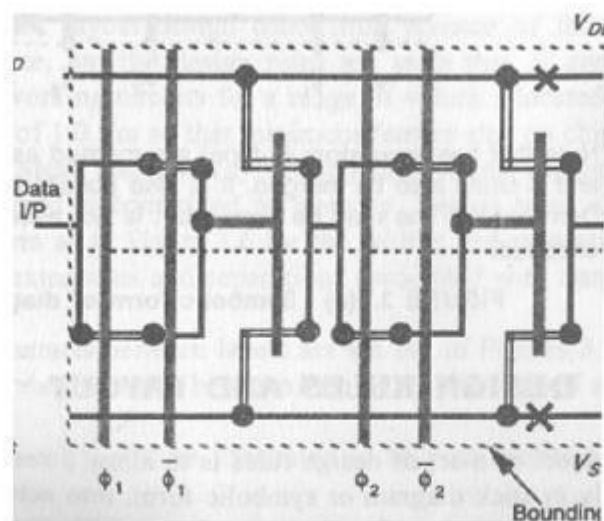


Figure 9: stick diagram of dyanmic shift register in CMOS style

Figure 9 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 λ -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 2λ . All width and spacing rules are specified in terms of the parameter λ . Suppose we have design rules that call for a minimum width of 2λ , and a minimum spacing of 3λ . If we select a $2\mu\text{m}$ technology (i.e., $\lambda = 1\mu\text{m}$), the above rules translated to a minimum width of $2\mu\text{m}$ and a minimum spacing of $3\mu\text{m}$. On the otherhand, if a $1\mu\text{m}$ technology (i.e., $\lambda = 0.5\mu\text{m}$) is selected, then the same width and spacing rules are now specified as $1\mu\text{m}$ and $1.5\mu\text{m}$, respectively.

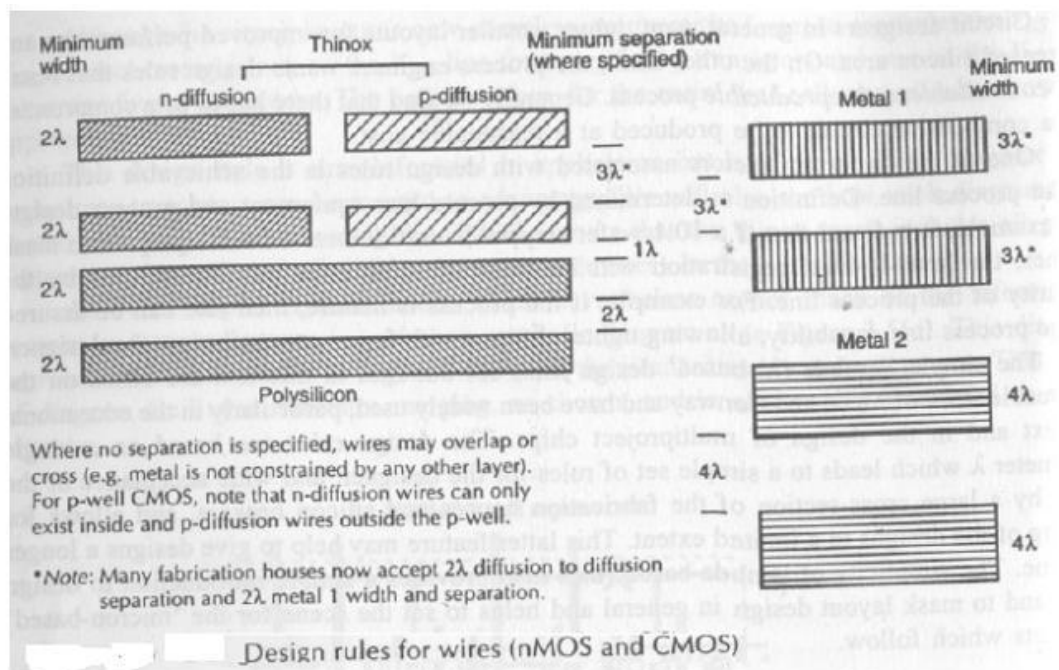


Figure 10: Design rules for the diffusion layers and metal layers

Figure 10 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.

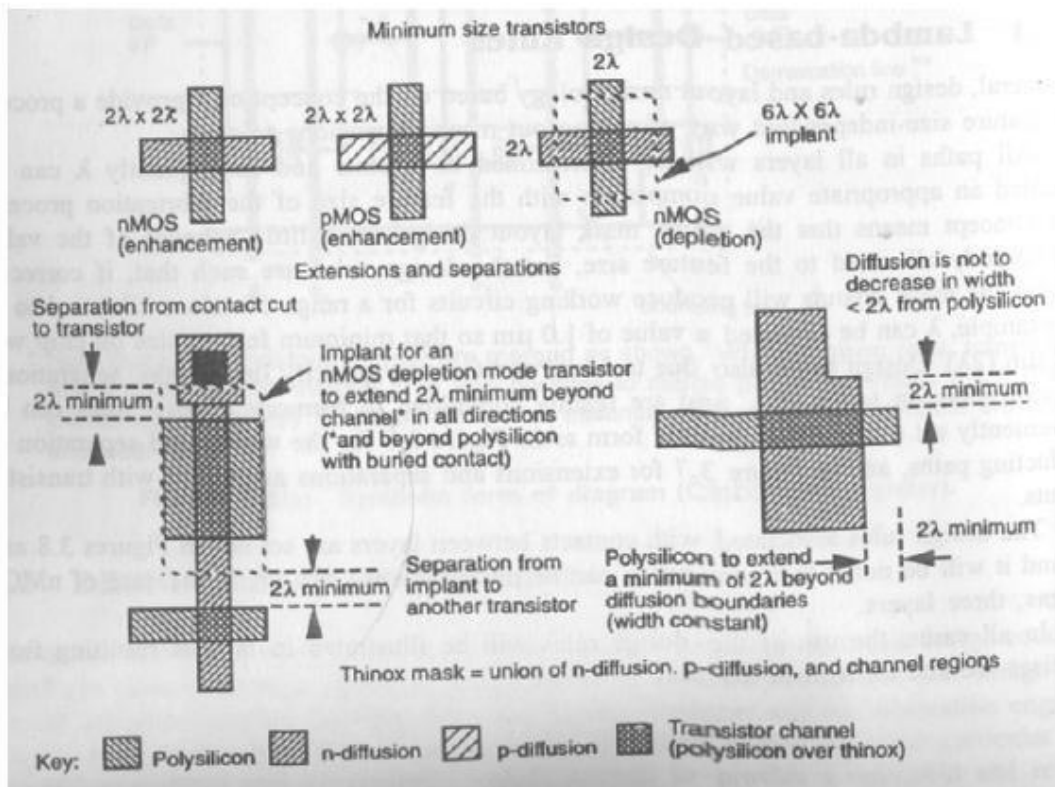


Figure 11: Design rules for transistors and gate over hang distance

Figure shows the design rule for the transistor, and it also shows that the poly should extend for a minimum of 2λ beyond the diffusion boundaries. (gate over hang distance)

Via:

It is used to connect higher level metals from metal1 connection. The cross section and layout view given figure13 explain via in a better way.

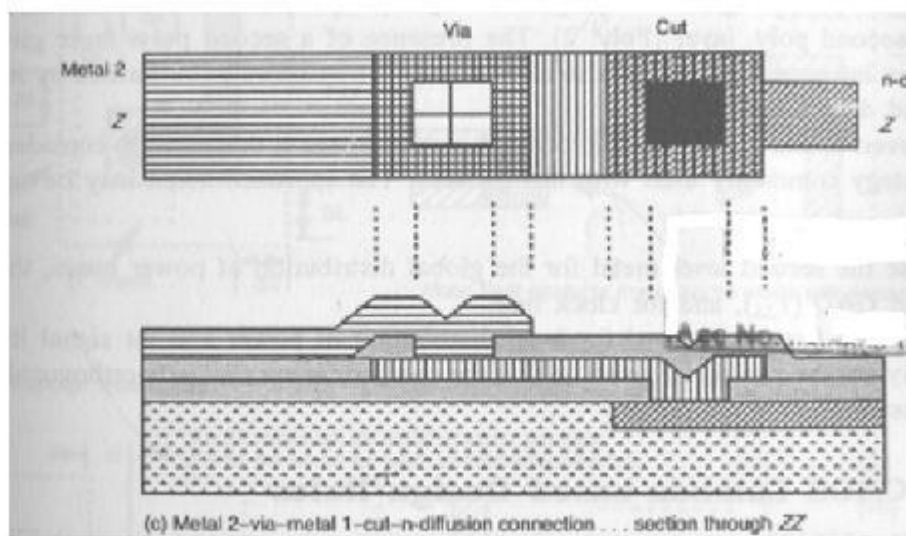


Figure 12: cross section showing the contact cut and via

The above figure shows the design rules for contact cuts and Vias. The design rule for contact is minimum $2\lambda \times 2\lambda$ and same is applicable for a Via.

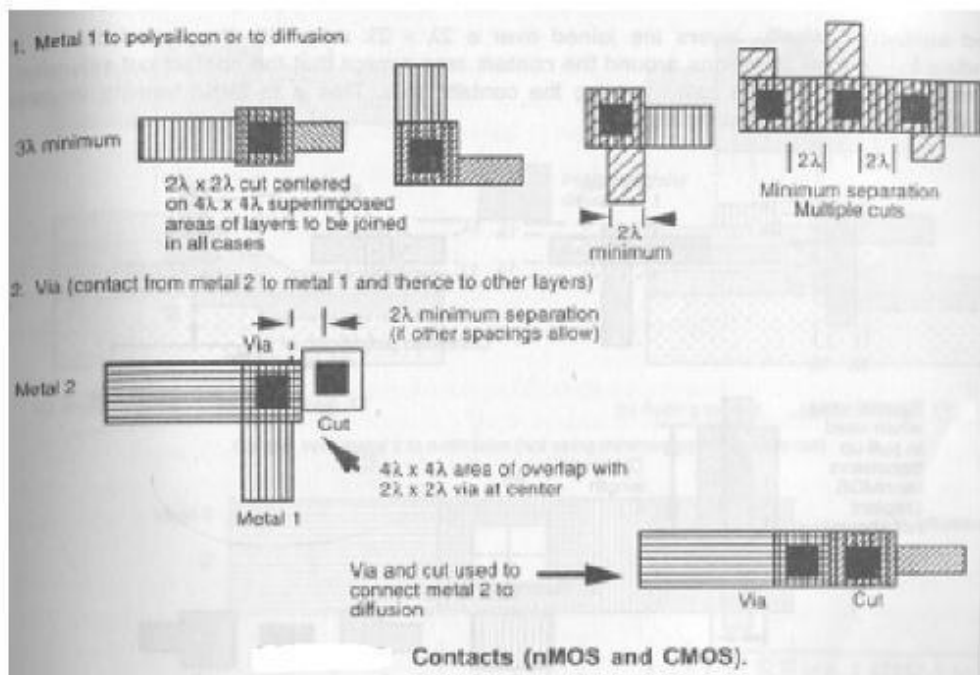


Figure 13: Design rules for contact cuts and vias

Buried contact: The contact cut is made down each layer to be joined and it is shown in figure 14

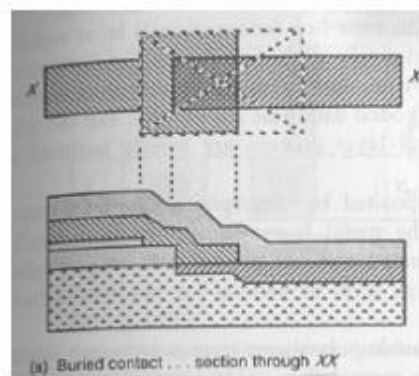


Figure 14: Buried contact

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

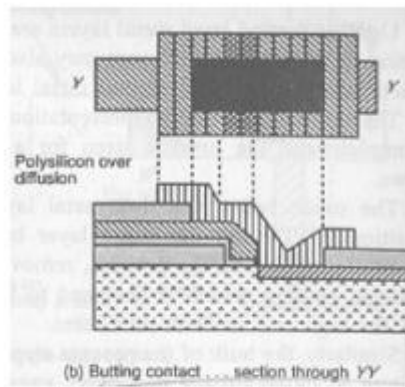


Figure 15: Butting contact

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 16 shows the rules to be followed in CMOS well processes to accommodate both n and p transistors.

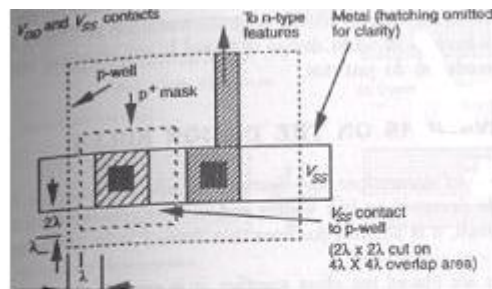


Figure 16: CMOS design rules

Orbit 2 μ m CMOS process:

In this process all the spacing between each layers and dimensions will be in terms micrometer. The 2 μ m here represents the feature size. All the design rules what ever 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 17 shows the design rule for BiCMOS process using orbit 2 μ m process.

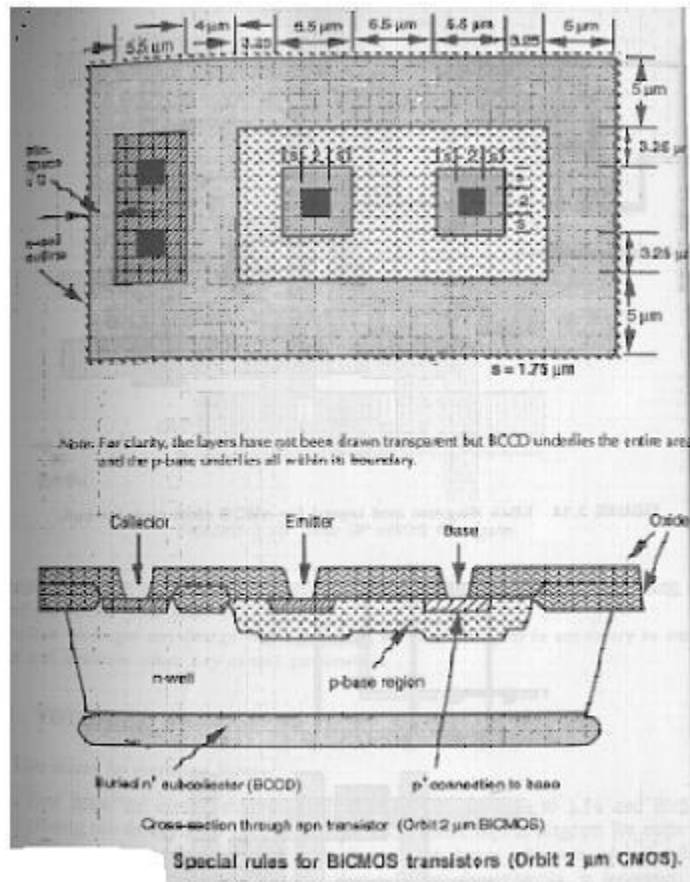


Figure 17: BiCMOS design rules

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

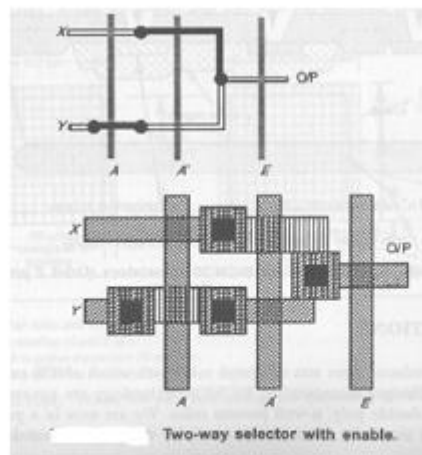


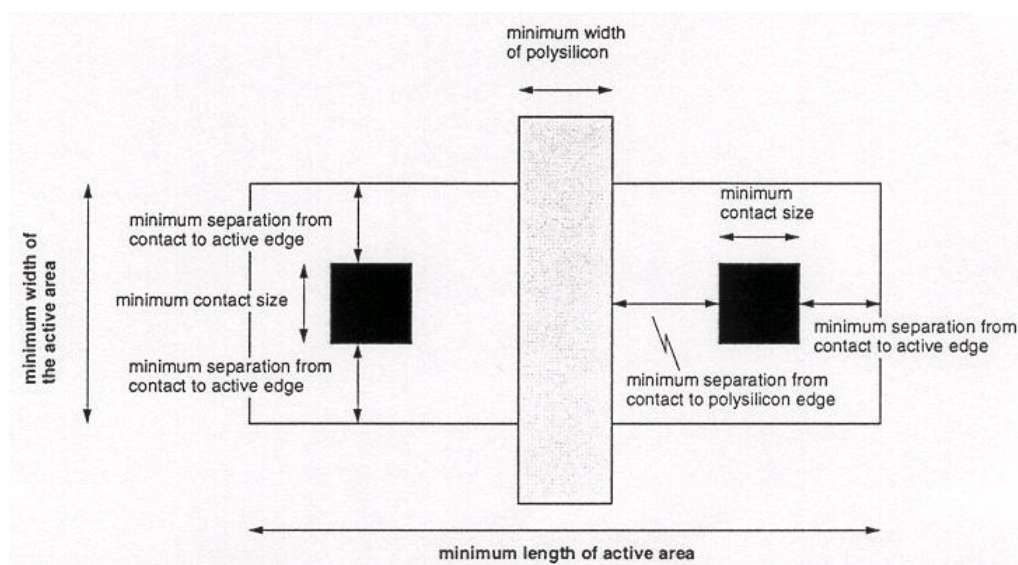
Figure 18: Two way selector stick and layout

Layout diagram of cmos inverter:

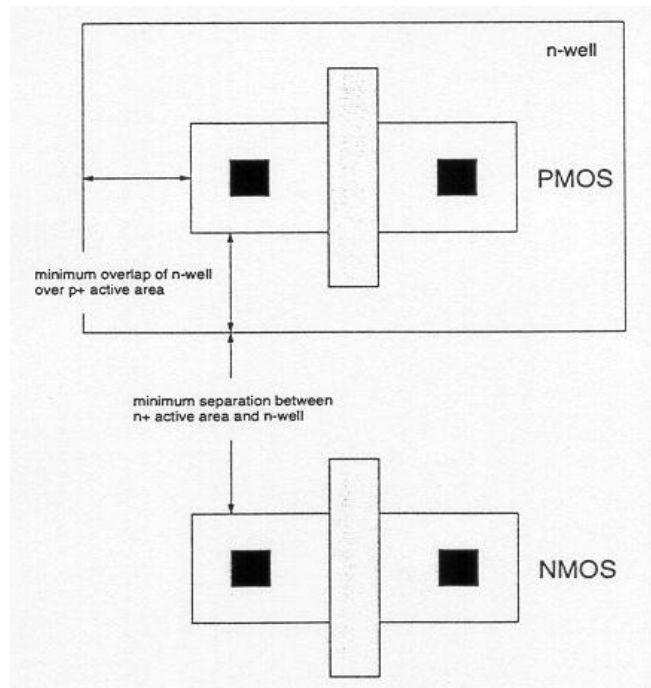
In the following, the mask layout design of a CMOS inverter will be examined step-by-step. The circuit consists of one nMOS and one pMOS transistor, therefore, one would assume that the layout topology is relatively simple. Yet, we will see that there exist quite a number of different design possibilities even for this very simple circuit.

First, we need to create the individual transistors according to the design rules. Assume that we attempt to design the inverter with minimum-size transistors. The width of the active area is then determined by the minimum diffusion contact size (which is necessary for source and drain connections) and the minimum separation from diffusion contact to both active area edges. The width of the polysilicon line over the active area (which is the gate of the transistor) is typically taken as the minimum poly width. Then, the overall length of the active area is simply determined by the following sum: (minimum poly width) + 2 x (minimum poly-to- contact spacing) + 2 x (minimum spacing from contact to active area edge). The pMOS transistor must be placed in an n-well region, and the minimum size of the n-well is dictated by the pMOS active area and the minimum n-well overlap over n+. The distance between the nMOS and the pMOS transistor is determined by the minimum separation between the n+ active area and the n-well. The polysilicon gates of the nMOS and the pMOS transistors are usually aligned. The final step in the mask layout is the local interconnections in metal, for the output node and for the VDD and GND contacts. Notice that in order to be biased properly, the n-well region must also have a VDD contact.

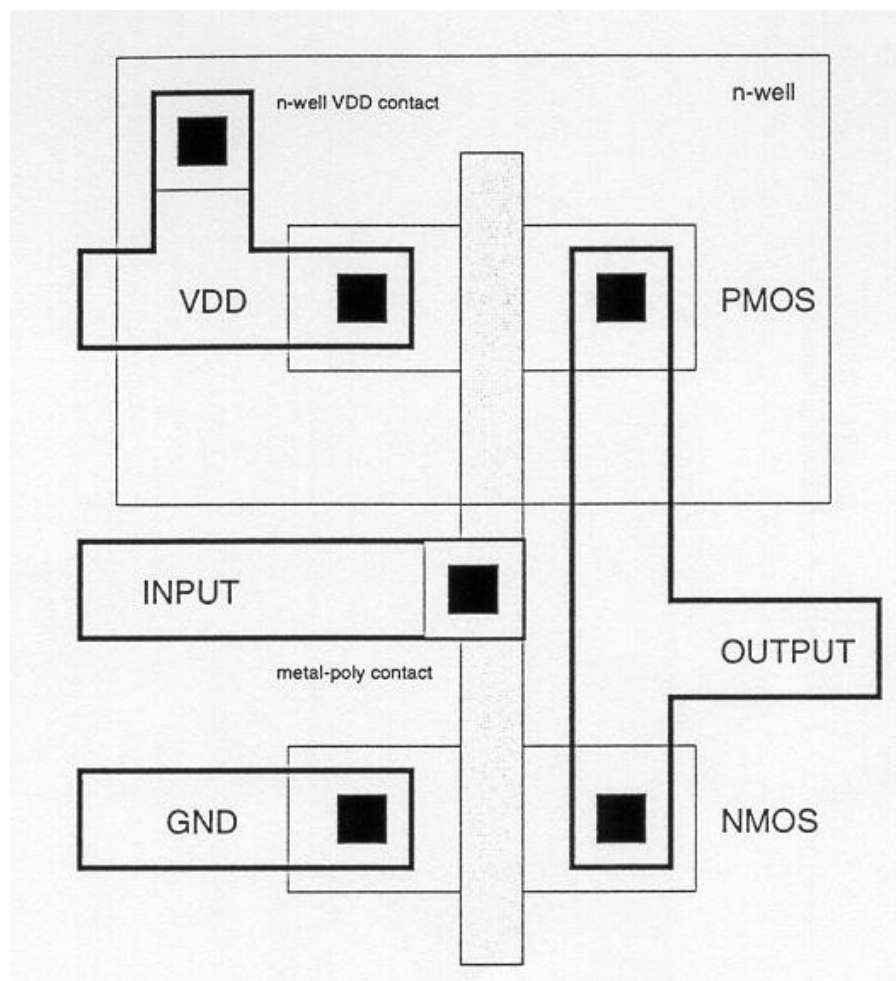
Dimensions of minimum size transistor



Placement of nmos and pmos transistors



Layout of cmos inverter



Layout of CMOS NAND and NOR Gates:

The mask layout designs of CMOS NAND and NOR gates follow the general principles examined earlier for the CMOS inverter layout. Figure 18 shows the sample layouts of a two-input NOR gate and a two-input NAND gate, using single-layer polysilicon and single-layer metal. Here, the p-type diffusion area for the pMOS transistors and the n-type diffusion area for the nMOS transistors are aligned in parallel to allow simple routing of the gate signals with two parallel polysilicon lines running vertically. Also notice that the two mask layouts show a very strong symmetry, due to the fact that the NAND and the NOR gate are have a symmetrical circuit topology. Finally, Figs 19 and 20 show the major steps of the mask layout design for both gates, starting from the stick diagram and progressively defining the mask layers.

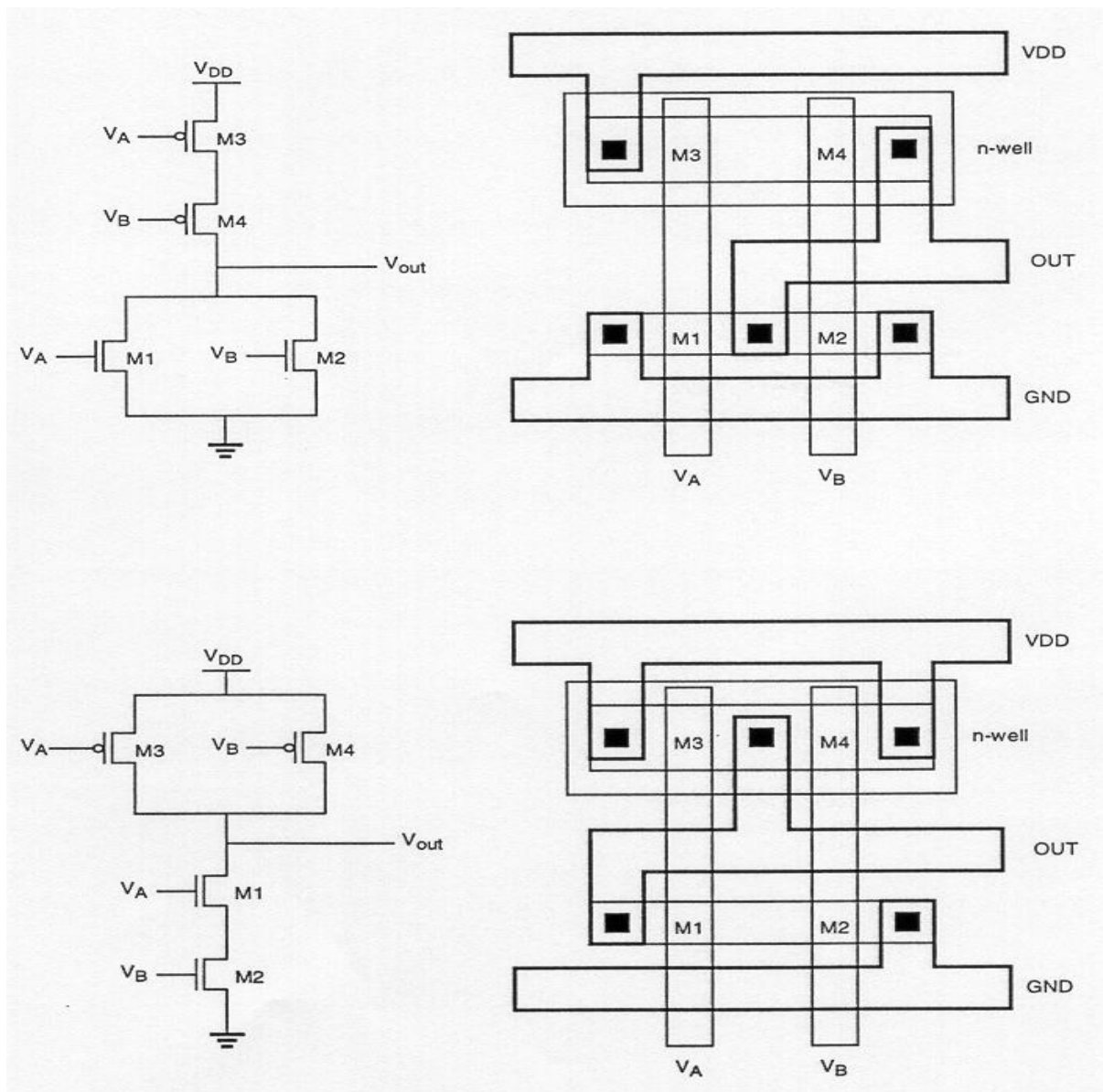
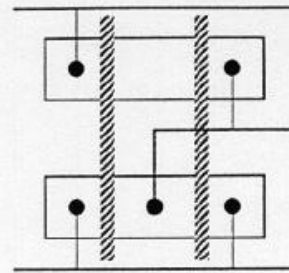
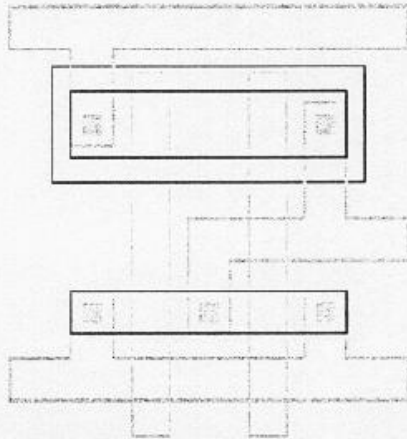


Figure 18: Sample layouts of a CMOS NOR₂ gate and a CMOS NAND₂ gate.

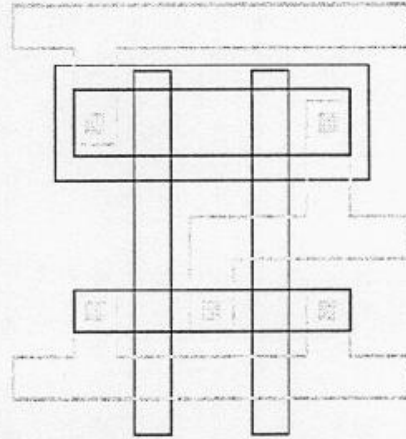
Mask layout of a CMOS NOR gate



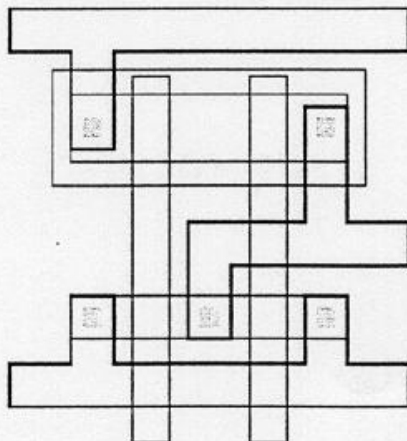
stick diagram layout



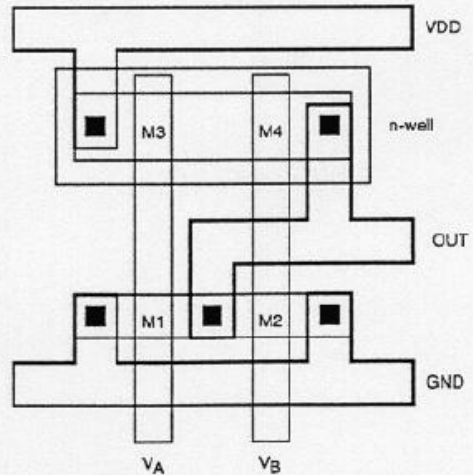
n-well and active area masks



poly mask -> define nMOS and pMOS transistors



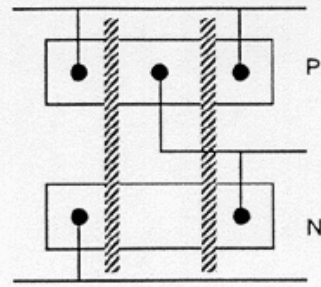
metal mask for VDD, GND and output connections



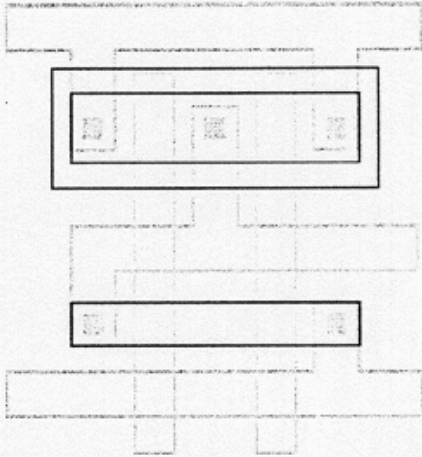
metal-diffusion contact mask

Figure 19: Major steps required for generating the mask layout of a CMOS NOR2 gate.

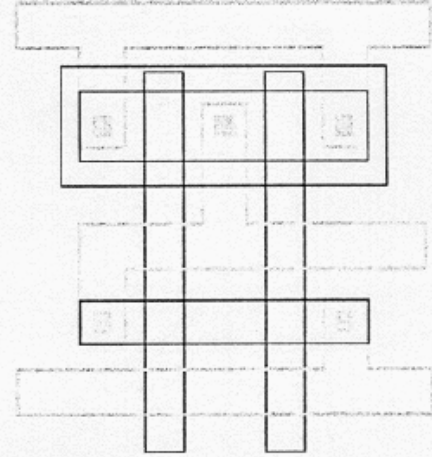
Mask layout of a CMOS NAND gate



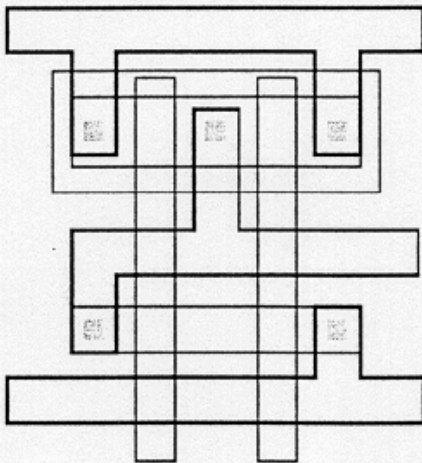
stick diagram layout



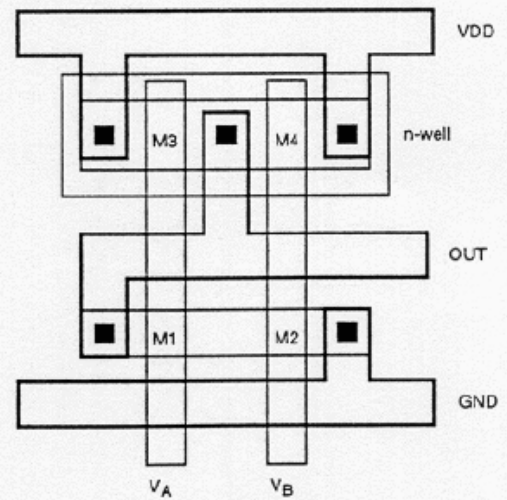
n-well and active area masks



poly mask -> define nMOS and pMOS transistors



metal mask for VDD, GND and output connections



metal-diffusion contact mask

Figure 20: Major steps required for generating the mask layout of a CMOS NAND2 gate.

Scaling of MOS Circuits:

1. What is Scaling?

Proportional Adjustment of the dimensions of an electronic device while maintaining the electrical Properties of the device, results in a device either larger or smaller than the un-scaled device. Then Which way do we scale the devices for VLSI? BIG and SLOW ... or SMALL and FAST? What do we gain?

2. Why Scaling?...

Scale the devices and wires down, Make the chips 'fatter' – functionality, intelligence, memory – and – faster, Make more chips per wafer – increased yield, Make the end user Happy by giving more for less and therefore, make MORE MONEY!!

3. FoM for Scaling

Impact of scaling is characterized in terms of several indicators:

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

Many of the FoMs can be improved by shrinking the dimensions of transistors and interconnections. Shrinking the separation between features – transistors and wires
Adjusting doping levels and supply voltages.

Technology Scaling :

Goals of scaling the dimensions by 30%:

Reduce gate delay by 30% (increase operating frequency by 43%) Double transistor density

Reduce energy per transition by 65% (50% power savings @ 43% increase in frequency)

Die size used to increase by 14% per generation

Technology generation spans 2-3 years

Figure 1 to Figure 5 illustrates the technology scaling in terms of minimum feature size, transistor count, propagation delay, power dissipation and density and technology generations.

Scaling Models:

Full Scaling (Constant Electrical Field)

Ideal model – dimensions and voltage scale together by the same scale factor
Fixed Voltage Scaling

Most common model until recently – only the dimensions scale, voltages remain constant
General Scaling

Most realistic for today's situation – voltages and dimensions scale with different factors

Scaling Factors for Device Parameters:

Device scaling modeled in terms of generic scaling factors: $1/\alpha$ and $1/\beta$

- $1/\beta$: scaling factor for supply voltage V_{DD} , and gate oxide thickness D
- $1/\alpha$: linear dimensions both horizontal and vertical dimensions

Why is the scaling factor for gate oxide thickness different from other linear horizontal and vertical dimensions? Consider the cross section of the device as in Figure 6, various parameters derived are as follows.

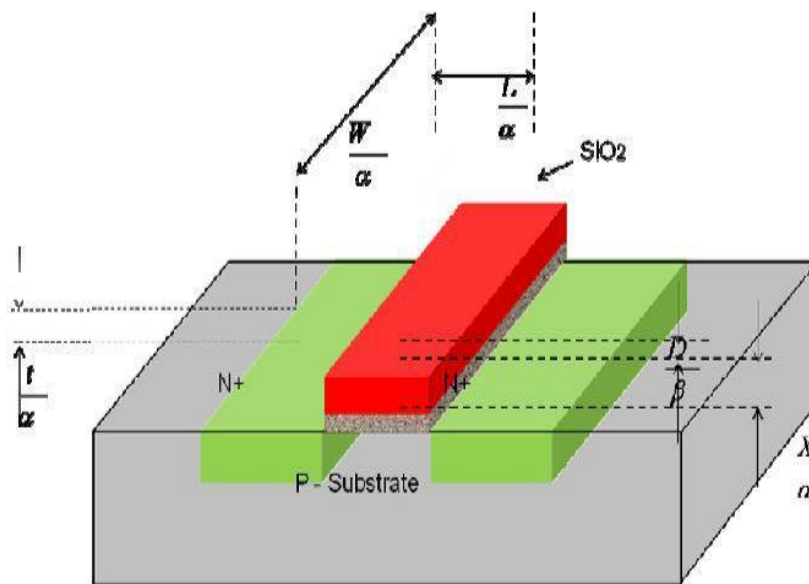


Figure-6: Technology generation

- Gate area A_g

$$A_g = L * W$$

Where L: Channel length and W: Channel width and both are scaled by $1/\alpha$

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

- Gate capacitance per unit area C_o or C_{ox}

$$C_{ox} = \epsilon_{ox}/D$$

Where ϵ_{ox} is permittivity of gate oxide(thin-ox) = $\epsilon_{ins}\epsilon_o$ and D is the gate oxide thickness scaled by $1/\beta$

Thus C_{ox} is scaled up by $\left(\frac{1}{\beta}\right) = \beta$

- Gate capacitance C_g $C_g = C_o * L * W$

Thus C_g is scaled up by $\beta * 1/\alpha^2 = \beta/\alpha^2$

- Parasitic capacitance C_x

C_x is proportional to A_x/d

where d is the depletion width around source or drain and scaled by $1/\alpha$

A_x is the area of the depletion region around source or drain, scaled by $(1/\alpha^2)$.

Thus C_x is scaled up by $\{1/(1/\alpha)\} * (1/\alpha^2) = 1/\alpha$

- Carrier density in channel Q_{on}

$$Q_{on} = C_o * V_{gs}$$

where Q_{on} is the average charge per unit area in the 'on' state.

C_o is scaled by β and V_{gs} is scaled by $1/\beta$

Thus Q_{on} is scaled by 1

- Channel Resistance R_{on}

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

Where μ = channel carrier mobility and assumed constant

Thus R_{on} is scaled by 1.

- Gate delay T_d

T_d is proportional to $R_{on} * C_g$

$$T_d \text{ is scaled by } \frac{1}{\alpha^2} * \beta = \frac{\beta}{\alpha^2}$$

- Maximum operating frequency f_o

$$f_o = \frac{W}{L} * \frac{\mu C_o V_{DD}}{C_g}$$

f_o is inversely proportional to delay T_d and is scaled by

$$\beta * \left(\frac{1}{\beta^2} \right) = \frac{1}{\beta}$$

- Saturation current I_{dss}

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

Both V_{gs} and V_t are scaled by $(1/\beta)$. Therefore, I_{dss} is scaled by $\frac{1}{\left(\frac{\beta}{\alpha^2}\right)} = \frac{\alpha^2}{\beta}$

- Current density J

Current density, $J = \frac{I_{dss}}{A}$ where A is cross sectional area of the Channel in the "on" state which is scaled by $(1/\alpha^2)$.

So, J is scaled by

$$\frac{1/\beta}{1/\alpha^2} = \frac{\alpha^2}{\beta}$$

- Switching energy per gate E_g

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

So E_g is scaled by

$$\frac{\beta}{\alpha^2} * \left(\frac{1}{\beta^2} \right) = \frac{1}{\alpha^2 \beta}$$

- Power dissipation per gate P_g

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

P_g comprises of two components: static component P_{gs} and dynamic component P_{gd} :

Where, the static power component is given by: $P_{gs} = \frac{V_{DD}^2}{R_{on}}$

And the dynamic component by: $P_{gd} = E_g f_o$

Since V_{DD} scales by $(1/\beta)$ and R_{on} scales by 1, P_{gs} scales by $(1/\beta^2)$.

Since E_g scales by $(1/\alpha^2 \beta)$ and f_o by (α_2 / β) , P_{gd} also scales by $(1/\beta^2)$. Therefore, P_g scales by $(1/\beta^2)$.

- Power dissipation per unit area P_a

$$P_a = \frac{P_g}{A_g} = \frac{\left(\frac{1}{\beta^2}\right)}{\left(\frac{1}{\alpha^2}\right)} = \frac{\alpha^2}{\beta^2}$$

- Power – speed product P_T

$$P_T = P_g * T_d = \frac{1}{\beta^2} \left(\frac{\beta}{\alpha^2}\right) = \frac{1}{\alpha^2 \beta}$$

Implications of Scaling :

Improved Performance

Improved Cost

Interconnect Woes

Power Woes

Productivity Challenges

Physical Limits

Physical Limits :

- Will Moore's Law run out of steam?
Can't build transistors smaller than an atom...
- Many reasons have been predicted for end of scaling
 - Dynamic power
 - Sub-threshold leakage, tunneling
 - Short channel effects

- Fabrication costs
- Electro-migration
- Interconnect delay
- Rumors of demise have been exaggerated

8. Limitations of Scaling

Effects, as a result of scaling down- which eventually become severe enough to prevent further miniaturization.

- Substrate doping
- Depletion width
- Limits of miniaturization
- Limits of interconnect and contact resistance
- Limits due to sub threshold currents
- Limits on logic levels and supply voltage due to noise
- Limits due to current density

UNIT-III

GATE LEVEL DESIGN

INTRODUCTION :

The module (integrated circuit) is implemented in terms of logic gates and interconnections between these gates. Designer should know the gate-level diagram of the design. In general, gate-level modeling is used for implementing lowest level modules in a design like, full-adder, multiplexers, etc.

Boolean algebra is used to represent logical (combinational logic) functions of digital circuits. A combinational logic expression is a mathematical formula which is to be interpreted using the laws of Boolean algebra. Now the goal of logic design or optimization is to find a network of logic gates that together compute the combinational logic function we want.

For example, given the expression $a+b$, we can compute its truth value for any given values of a and b , and also we can evaluate relationships such as $a+b = c$. but logic design is difficult for many reasons:

- We may not have a logic gate for every possible function, or even for every function of n inputs.
- Not all gate networks that compute a given function are alike-networks may differ greatly in their area and speed.
- Thus combinational logic expressions are the specification, Logic gate networks are the implementation, Area, delay, and power are the costs.
- A **logic gate** is an idealized or physical device implementing a Boolean function, that is, it performs a logical operation on one or more logic inputs and produces a single logic output.
- Logic gates are primarily implemented using diodes or transistors acting as electronic switches, but can also be constructed using electromagnetic relays (relay logic), fluidic logic, pneumatic logic, optics, molecules, or even mechanical elements.
- With amplification, logic gates can be cascaded in the same way that Boolean functions can be composed, allowing the construction of a physical model of all of Boolean logic.

- Simplest form of electronic logic is diode logic. This allows AND and OR gates to be built, but not inverters, and so is an incomplete form of logic. Further, without some kind of amplification it is not possible to have such basic logic operations cascaded as required for more complex logic functions.
- To build a functionally complete logic system, relays, valves (vacuum tubes), or transistors can be used.
- The simplest family of logic gates using bipolar transistors is called resistor-transistor logic (RTL). Unlike diode logic gates, RTL gates can be cascaded indefinitely to produce more complex logic functions. These gates were used in early integrated circuits. For higher speed, the resistors used in RTL were replaced by diodes, leading to diode-transistor logic (DTL).
- Transistor-transistor logic (TTL) then supplanted DTL with the observation that one transistor could do the job of two diodes even more quickly, using only half the space.
- In virtually every type of contemporary chip implementation of digital systems, the bipolar transistors have been replaced by complementary field-effect transistors (MOSFETs) to reduce size and power consumption still further, thereby resulting in complementary metal-oxide-semiconductor (CMOS) logic. that can be described with Boolean logic.

CMOS LOGIC GATES AND OTHER COMPLEX GATES:

General logic circuit:

Any Boolean logic function (F) has two possible values, either logic 0 or logic 1. For some of the input combinations, $F = 1$ and for all other input combinations, $F = 0$. So in general, any Boolean logic function can be realized using a structure as shown in figure.

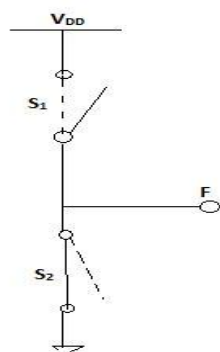


Fig: 1 Realization of Boolean function

- The switch S_1 is closed and switch S_2 is open for input combinations that produces $F = 1$.
- The switch S_1 is open and switch S_2 is closed for input combinations that produces $F = 1$.
- The switch S_1 is open and switch S_2 is open for input combinations that produces $F = 0$.

Thus the output (F) is either connected to V_{DD} or the ground, where the logic 0 is represented by the ground and the logic 1 is represented by V_{DD} . So the requirement of digital logic design is to implement the pull-up switch (S_1) and the pull-down switch(S_2).

CMOS STATIC LOGIC:

A generalized CMOS logic circuit consists of two transistor nets nMOS and pMOS. The pMOS transistor net is connected between the power supply and the logic gate output called as pull-up network , Whereas the nMOS transistor net is connected between the output and ground called as pull-down network. Depending on the applied input logic, the PUN connects the output node to V_{DD} and PDN connects the output node to the ground.

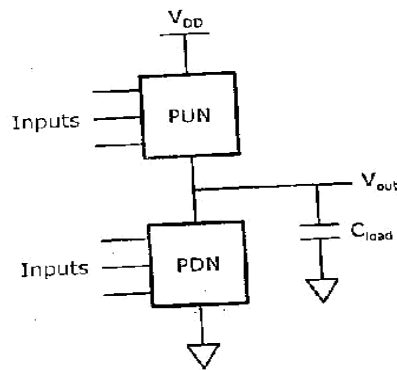


Fig: 2 CMOS Logic

The transistor network is related to the Boolean function with a straight forward design procedure:

- Design the pull down network (PDN) by realizing, AND(product) terms using series-connected nMOSFETs. OR (sum) terms using parallel-connected nMOSFETS.
- Design the pull-up network by realizing,AND(product) terms using parallel-connected nMOSFETS. OR (sum) terms using series-connected nMOSFETS.

- Add an inverter to the output to complement the function. Some functions are inherently negated, such as NAND,NOR gates do not need an inverter at the output terminal.

CMOS inverter:

A CMOS inverter is the simplest logic circuit that uses one nMOS and one pMOS transistor. The nMOS is used in PDN and the pMOS is used in the PUN as shown in figure.

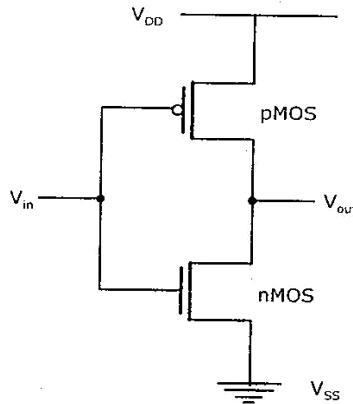


Fig: 3 CMOS inverter

Working operation:

When the input V_{in} is logic HIGH, then the nMOS transistor is ON and the pMOS transistor is OFF. Thus the output Y is pulled down to ground (logic 0) since it is connected to ground but not to source V_{DD} .

When the input V_{in} is logic LOW, then nMOS transistor is OFF and the pMOS transistor is ON, Thus the output Y is pulled up to V_{DD} (logic 1) since it is connected to source via pMOS but not to ground.

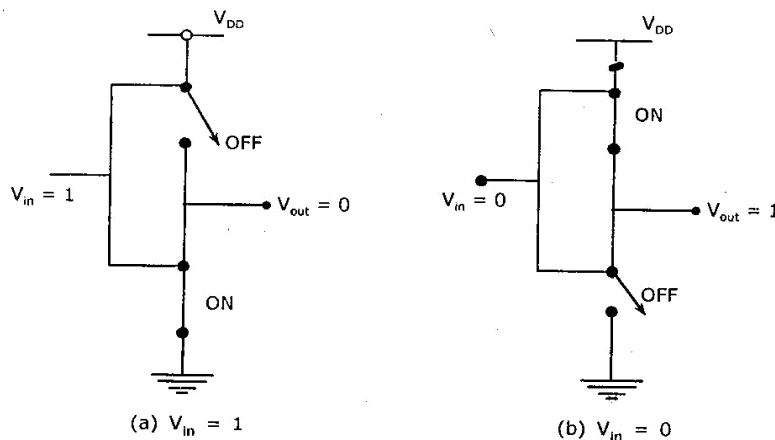


Fig: 4 CMOS inverter realization

CMOS NAND gate:

The two input NAND function is expressed by $Y = \overline{A \cdot B}$

Step 1: Take complement of Y

$$Y = \overline{A \cdot B} = \overline{A} \cdot \overline{B}$$

Step 2 Design the PDN

In this case, there is only one AND term, so there will be two nMOSFETs in series as shown in figure.

Step 3 Design the PUN. In PUN there will be two pMOSFETs in parallel, as shown in figure

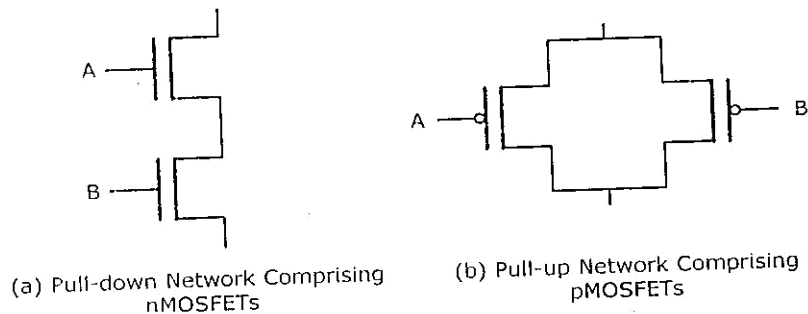


Fig: 5 CMOS NAND gate realization

Finally join the PUN and PDN as shown in figure which realizes two –input NAND gate. Note that we have realized y , rather than Y because the inversion is automatically provided by the nature of the CMOS circuit operation,

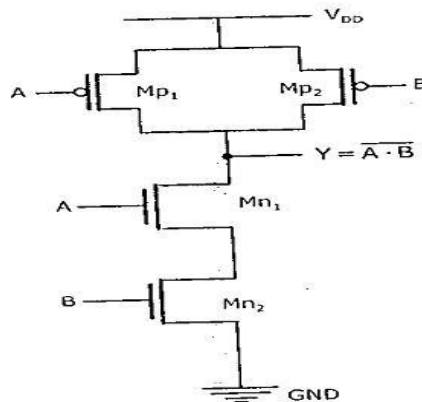


Fig: 6 CMOS NAND gate

Working operation:

Whenever at least one of the inputs is LOW, the corresponding pMOS transistor will conduct while the corresponding nMOS transistor will turn OFF. Subsequently, the output voltage will be HIGH.

Conversely, if both inputs are simultaneously HIGH, then both pMOS transistors will turn OFF, and the output voltage will be pulled LOW by the two conducting nMOS transistors.

CMOS NOR gate:

The two input NOR function is expressed by $Y = \overline{A+B}$

Step 1 Take complement of $Y = A+B$

Step 2 Design the PDN

In this case, there is only one OR term, so there will be two nMOSFETs connected in parallel, as shown in figure.

Step 3 Design the PUN

In PUN there will be two pMOSFETs in series, as shown in figure.

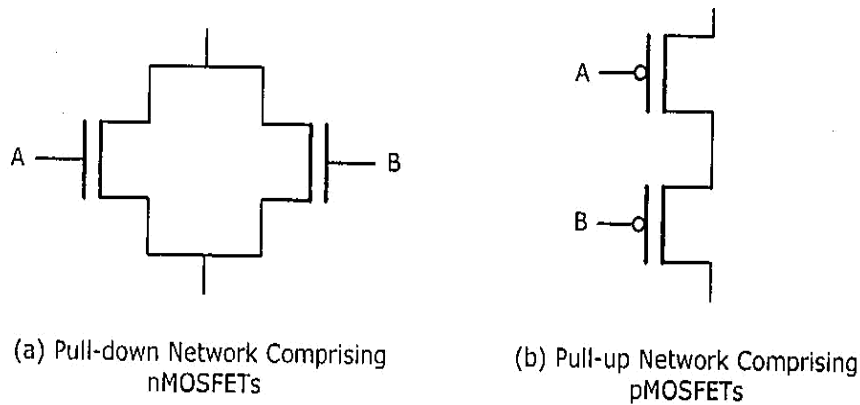


Fig: 7 CMOS NOR gate realization

Finally join the PUN and PDN as shown in figure which realizes two –input NAND gate. Note that we have realized y , rather than Y because the inversion is automatically provided by the nature of the CMOS circuit operation,

Working operation:

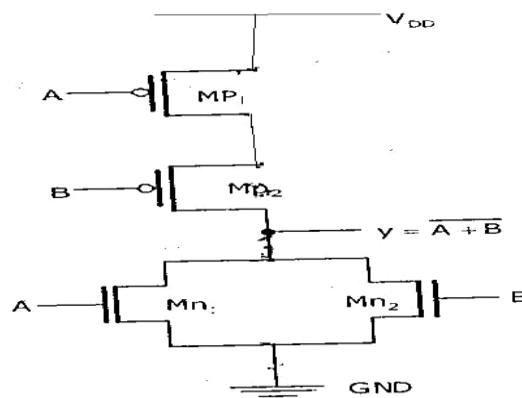


Fig: 8 CMOS NOR gate

Whenever at least one of the inputs is LOW, the corresponding pMOS transistor will conduct while the corresponding nMOS transistor will turn OFF. Subsequently, the output voltage will be HIGH.

Conversely, if both inputs are simultaneously HIGH, then both pMOS transistors will turn OFF, and the output voltage will be pulled LOW by the two conducting nMOS transistors.

Complex gates in CMOS logic:

A complex logic gate is one that implements a function that can provide the basic NOT, AND and OR operation but integrates them into a single circuit. CMOS is ideally suited for creating gates that have logic equations by exhibiting the following,

- 1) AND-OR-INVERT - AOI form
- 2) OR-AND-INVERT - OAI form

An AOI logic equation is equivalent to a complemented SOP form, while an OAI equation is equivalent to a complemented POS structure. In CMOS, output always produces NOT operation acting on input variable.

1) AOI Logic Function (OR) Design of XOR gate using CMOS logic.

AND-OR-INVERT logic function(AOI) implements operation in the order AND,OR,NOT. For example , let us consider the function $Y = \overline{AB + CD}$ i.e., $Y = \text{NOT}((A \text{ AND } B) \text{ OR } (C \text{ AND } D))$ The AOI gate implementation for Y

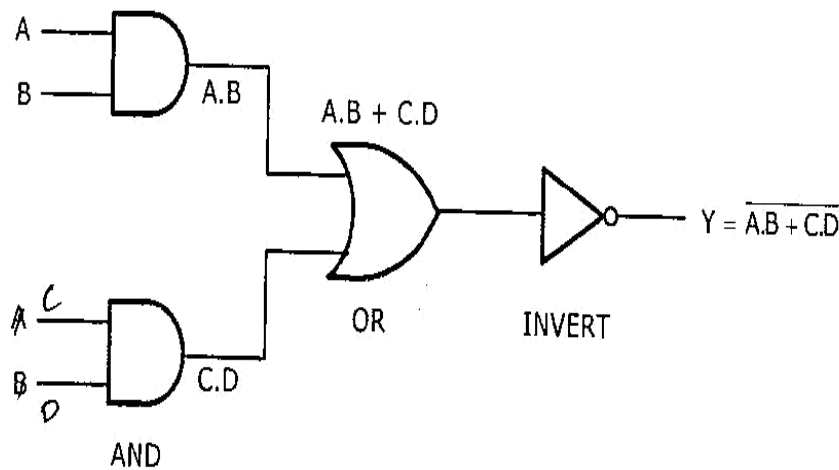
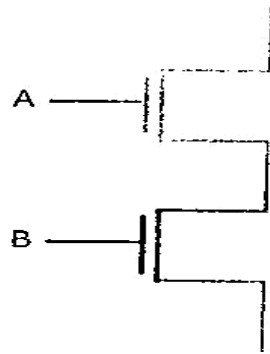


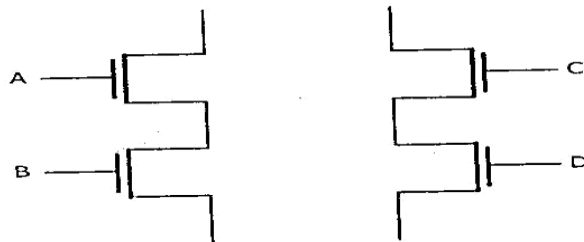
Fig: 9 XOR gate realization

CMOS implementation for Y:

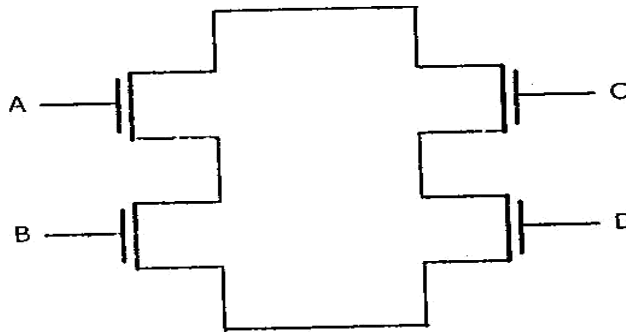
Step 1: Draw A.B (AND) function first by connecting 2 nMOS transistors in series.



Step 2: Draw C.D implementation, by using 2 nMOS transistors in series.



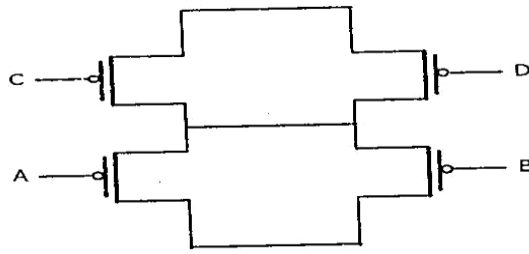
Step 3: $Y = A.B + C.D$, In this function A.B and C.D are added, for addition, we have to draw parallel connection. So, A.B series connected in parallel with C.D as shown in figure.



Step 4: Draw pMOS connection,

- I. In nMOS A,B connected in series. So, in pMOS side, A.B should be connected in parallel.
- II. In nMOS C,D connected in series. So, in pMOS side, C.D should be connected in parallel.
- III. A.B and C.D networks are connected in parallel in nMOS side. So, in pMOS side, A.B and C.D networks should be connected in series.

IV. In pMOS multiplication should be drawn in parallel, then addition should be drawn in series as shown in figure.



Step 5: Take output at the point in between nMOS and pMOS networks.

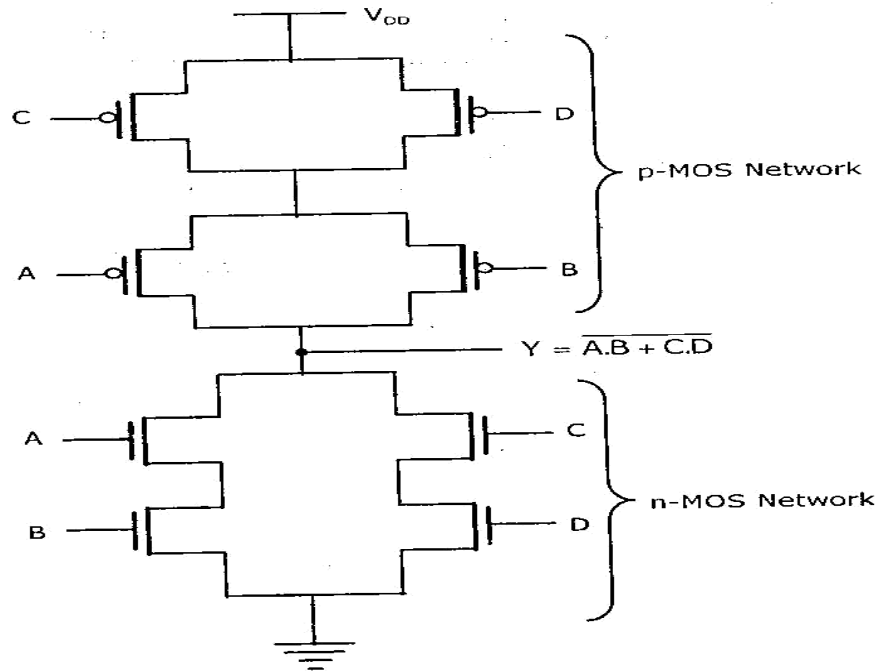


Fig: 10 CMOS XOR gate realization

1) OAI Logic Function (OR) Design of XNOR gate using CMOS logic.

OR-AND-INVERT logic function(AOI) implements operation in the order OR,AND,NOT. For example , let us consider the function $Y = (A+B).(C+D)$ i.e., $Y = \overline{\overline{(A + B).(C + D)}}$ The OAI logic gate implementation for Y

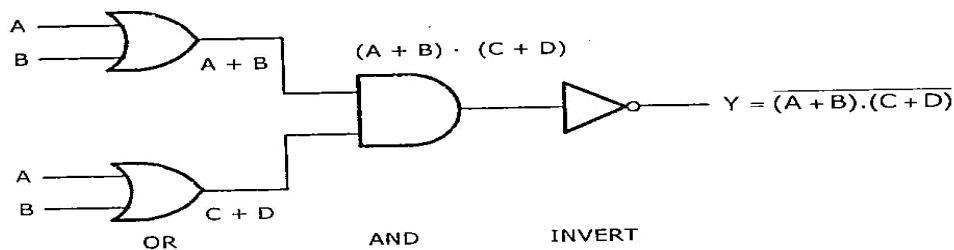


Fig: 11 XNOR gate realization

CMOS implementation for Y:

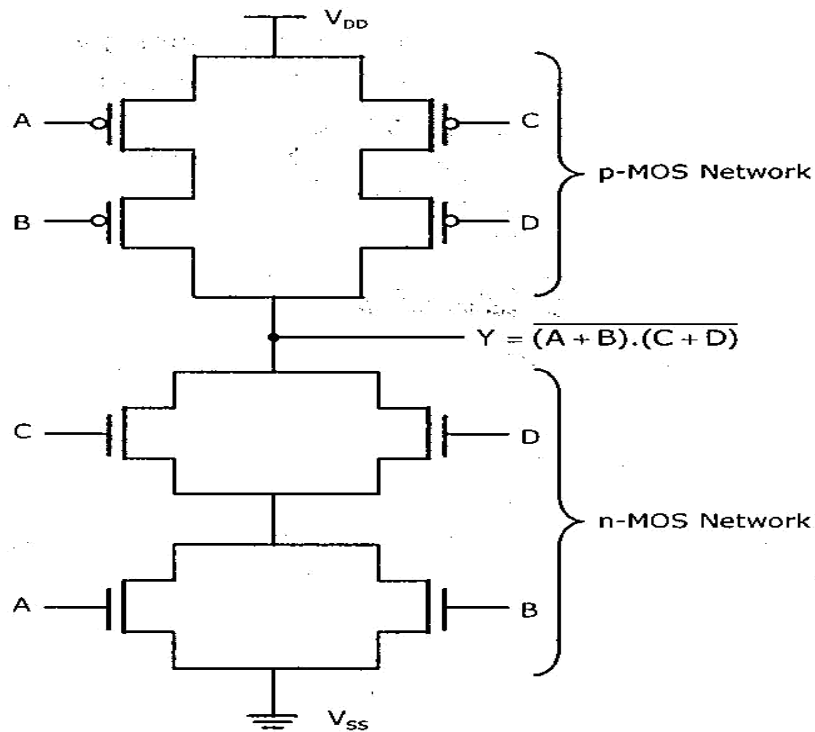


Fig: 12 CMOS XNOR gate realization

SWITCH LOGIC:

Switch logic is mainly based on pass transistor or transmission gate. It is fast for small arrays and takes no static current from the supply, V_{DD} . Hence power dissipation of such arrays is small since current only flows on switching.

Switch (pass transistor) logic is analogous to logic arrays based on relay contacts, where in path through each switch is isolated from the logic levels activating the switch.

PASS TRANSISTOR

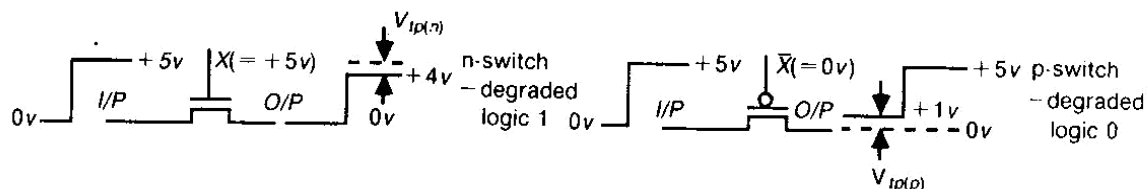


Fig: 13 (a) Pass transistor

This logic uses transistors as switches to carry logic signals from node to node instead of connecting output nodes directly to V_{DD} or ground(GND) If a single transistor is a switch between two nodes, then voltage degradation equal to v_t (threshold voltage) for high or low level

depends up on nMOS or pMOS logic. When using nMOS switch logic no pass transistor gate input may be driven through one or more pass transistors as shown in figure.

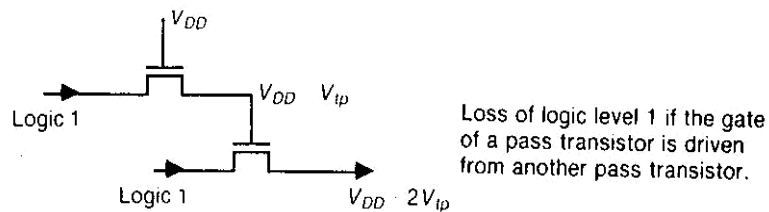


Fig: 13 (b) Pass transistor

Since the signal out of pass transistor T_1 does not reach a full logic 1 by threshold voltage effects signal is degraded by below a true logic 1, this degraded voltage would not permit the output of T_2 to reach an acceptable logic 1 level.

Advantages: They have topological simplicity.

- 1) Requires minimum geometry.
- 2) Do not dissipate standby power, since they do not have a path from supply to ground.

Disadvantages:

- 1) Degradation in the voltage levels due to undesirable threshold voltage effects.
- 2) Never drive a pass transistor with the output of another pass transistor.

TRANSMISSION GATE:

It is an electronic element, good non-mechanical relay built with CMOS technology. It is made by parallel combination of an nMOS and pMOS transistors with the input at gate of one transistor being complementary to the input at the gate of the other as shown in figure.

Thus current can flow through this element in either direction. Depending on whether or not there is a voltage on the gate, the connection between the input and output is either low resistance or high-resistance, respectively $R_{on} = 100\Omega$ and $R_{off} > 5\text{ M}\Omega$.

Operation:

- When the gate input to the nMOS transistor is '0' and the complementary '1' is gate input to the pMOS, thus both are turned off.

- When gate input to the nMOS is '1' and its complementary '0' is the gate input to the pMOS, both are turned on and passes any signal '1' and '0' equally without any degradation.
- The use of transmission gates eliminates the undesirable threshold voltage effects which give rise to loss of logic levels in pass-transistors as shown in figure.

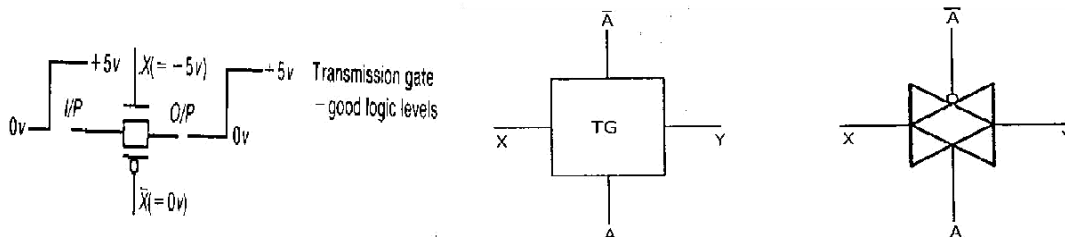


Fig: 14 Transmission gate

Advantages:

- 1) Transmission gates eliminates the signal degradation in the output logic levels.
- 2) Transmission gate consists of two transistors in parallel and except near the positive and negative rails.

Disadvantages:

- 1) Transmission gate requires more area than nMOS pass circuitry.
- 2) Transmission gate requires complemented control signals.

ALTERNATIVE GATE CIRCUITS:

CMOS suffers from increased area and correspondingly increased capacitance and delay, as the logic gates become more complex. For this reason, designers developed circuits (Alternate gate circuits) that can be used to supplement the complementary type circuits . These forms are not intended to replace CMOS but rather to be used in special applications for special purposes.

a) PSEUDO nMOS Logic:

Pseudo nMOS logic is one type of alternate gate circuit that is used as a supplement for the complementary MOS logic circuits. In the pseudo-nMOS logic, the pull up network (PUN) is

realized by a single pMOS transistor. The gate terminal of the pMOS transistor is connected to the ground. It remains permanently in the ON state. Depending on the input combinations, output goes low through the PDN. Figure shows the general building block of logic circuits that follows pseudo nMOS logic.

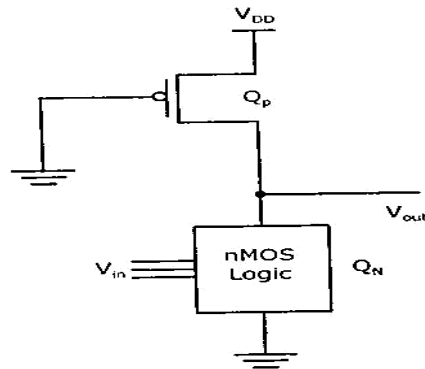


Fig: 15 Pseudo nMOS logic

Here, only the nMOS logic (Qn) is driven by the input voltage, while the gate of p-transistor(Qp) is connected to ground or substrate and Qp acts as an active load for Qn. Except for the load device, the pseudo-nMOS gate circuit is identical to the pull-down network(PDN) of the complementary CMOS gate. The realization of logic circuits using pseudo-nMOS logic is as shown in figure.

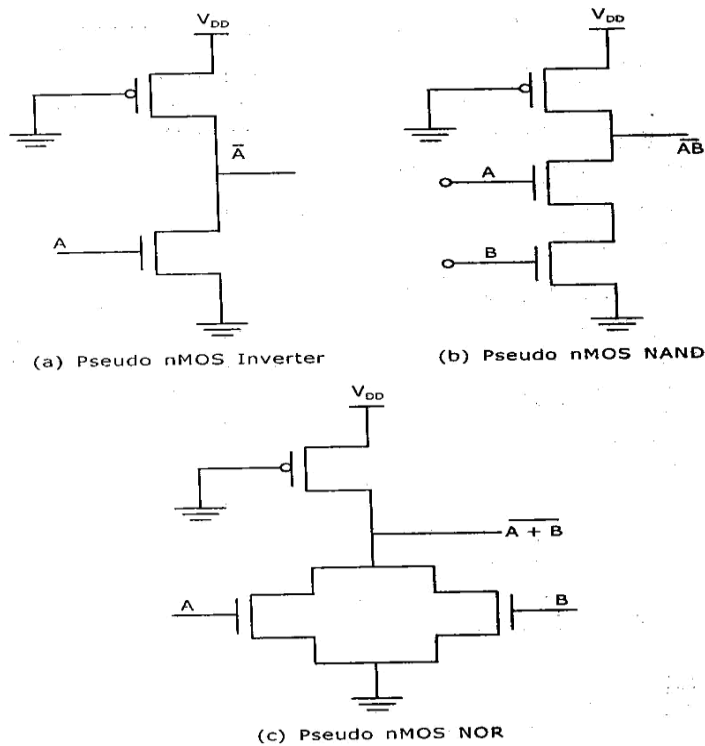


Fig: 16 Pseudo nMOS logic gates

Advantages:

- 1) Uses less number of transistors as compared to CMOS logic.
- 2) Geometrical area and delay gets reduced as it requires less transistors.
- 3) Low power dissipation.

Disadvantages:

- 1) The main drawback of using a pseudo nMOS gate instead of a CMOS gate is that the always on PMOS load conducts a steady current when the output voltage is lower than V_{DD} .
- 2) Layout problems are critical.

b) DYNAMIC CMOS LOGIC:

A dynamic CMOS logic uses charge storage and clocking properties of MOS transistors to implement logic operations. Figure shows the basic building block of dynamic CMOS logic. Here the clock ϕ drives nMOS evaluation transistor and pMOS precharge transistor. A logic is implemented using an nFET array connected between output node and ground.

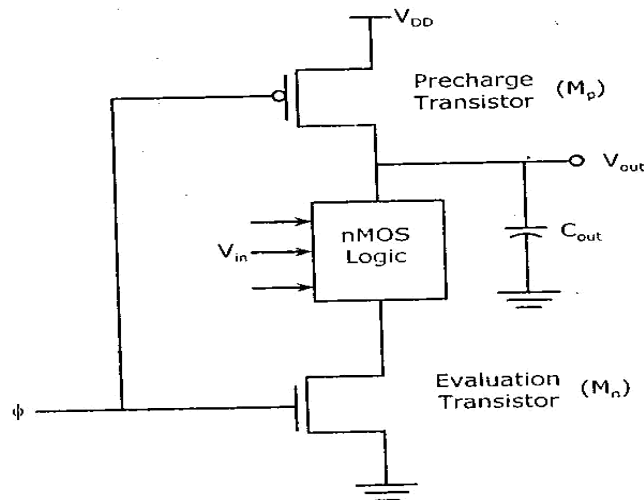


Fig: 17 Dynamic CMOS logic

The gate (clock ϕ) defines two phases, evaluation and precharge phase during each clock cycle.

Working :

- When clock $\phi = 0$ the circuit is in precharge phase with the pMOS device M_p ON and the evaluation nMOS M_n OFF. This establishes a conducting path between V_{DD} and the output allowing C_{out} to charge to a voltage $V_{out} = V_{DD}$. M_p is often called the precharge FET.

- When clock $\phi = 1$ the circuit is in evaluation phase with the pMOS device M_p OFF and the evaluation nMOS M_n ON. If the logic block acts like a closed switch the C_{out} can discharge through logic array and M_n , this gives a final result of $V_{out} = V_{DD}$, logically this is an output of $F = 1$. Charge leakage eventually drops the output to $V_{out} = 0$ V which could be an incorrect logic value.

The logic formation is formed by three series connected FETs (3-input NAND gate) is shown in figure.

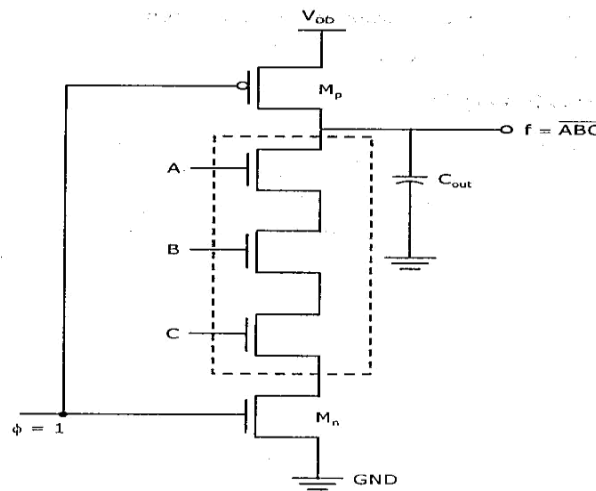


Fig: 17 Dynamic CMOS logic implementation

The dynamic CMOS logic circuit has a serious problem when they are cascaded. In the precharged phase ($\phi = 0$), output of all the stages are pre-charged to logic high. In the evaluation phase ($\phi = 1$), the output of all stages are evaluated simultaneously. Suppose in the first stage, the inputs are such that the output is logic low after the evaluation. In the second stage, the output of the first stage is one input and there are other inputs.

If the other inputs of the second stage are such that output of it discharges to logic low, then the evaluated output of the first stage can never make the output of the second stage logic high. This is because, by the time the first stage is being evaluated, output of the second stage is discharged, since evaluation happens simultaneously. Remember that the output cannot be charged to logic high in the evaluation phase ($\phi = 1$, pMOSFET in PUN is OFF), it can only be retained in the logic high depending on the inputs.

Advantages:

Large noise margin and Low power dissipation, Small area due to less number of transistors.

c) CMOS DOMINO LOGIC

Standard CMOS logic gates need a PMOS and an NMOS transistor for each logic input. The pMOS transistors require a greater area than the nMOS transistors carrying the same current. So, a large chip area is necessary to perform complex logic operations. The package density in CMOS is improved if a dynamic logic circuit, called the domino CMOS logic circuit, is used.

Domino CMOS logic is a slightly modified version of the dynamic CMOS logic circuit. In this case, a static inverter is connected at the output of each dynamic CMOS logic block. The addition of the inverter solves the problem of cascading of dynamic CMOS logic circuits. The circuit diagram of domino CMOS logic structures is shown in figure as follows

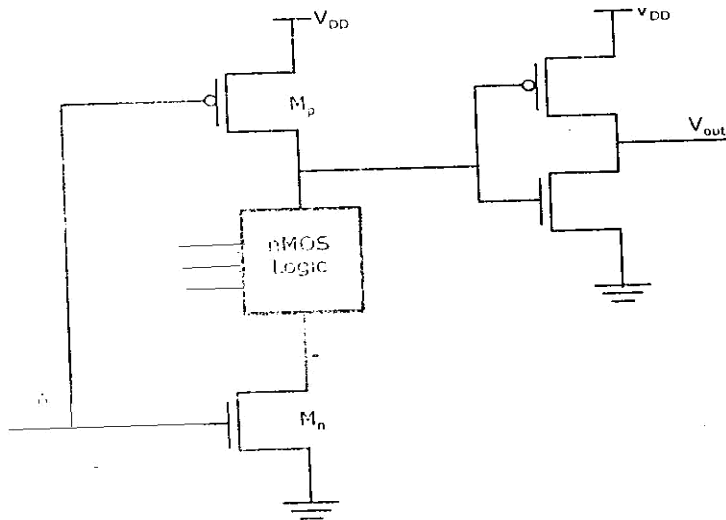


Fig: 18 CMOS Domino logic

A domino CMOS AND-OR gate that realizes the function $y = AB + CD$ is depicted in figure . The left hand part of the circuit containing $M_n, M_p, T_1, T_2, T_3,$ and T_4 forms an AND-OR-INVERTER (AOI) gate. It drives the static CMOS inverter formed by N_2 and P_2 in the right-hand part of the circuit. The domino gate is activated by the single phase clock ϕ applied to the NMOS (M_n) and the PMOS (M_p) transistors. The load on the AOI part of the circuit is the parasitic load capacitance.

Working:

When $\phi = 0$, M_p is ON and M_n is OFF, so that no current flows in the AND-OR paths of the AOI. The capacitor C_L is charged to V_{DD} through M_p since the latter is ON. The input to the inverter is high, and drives the output voltage V_0 to logic-0.

When $\phi = 1$, Mp is turned OFF and Mn is turned ON. If either (or both) A and B or C and D is at logic-1, C_L discharges through either T2,T1 and Mn or T3,T4 and Mp. So , the inverter input is driven to logic-0 and hence the output voltage V_0 to logic-1. The Boolean expression for the output voltage is $Y = AB + CD$.

Note :

Logic input can change only when $\phi = 0$. No changes of the inputs are permitted when $\phi = 1$ since a discharge path may occur.

Advantages:

- Smaller areas compared to conventional CMOS logic.
- Parasitic capacitances are smaller so that higher operating speeds are possible.
- Operation is free of glitches since each gate can make one transition.

Disadvantages:

- Non inverting structures are possible because of the presence of inverting buffer.
- Charge distribution may be a problem.

d) CLOCKED CMOS LOGIC:

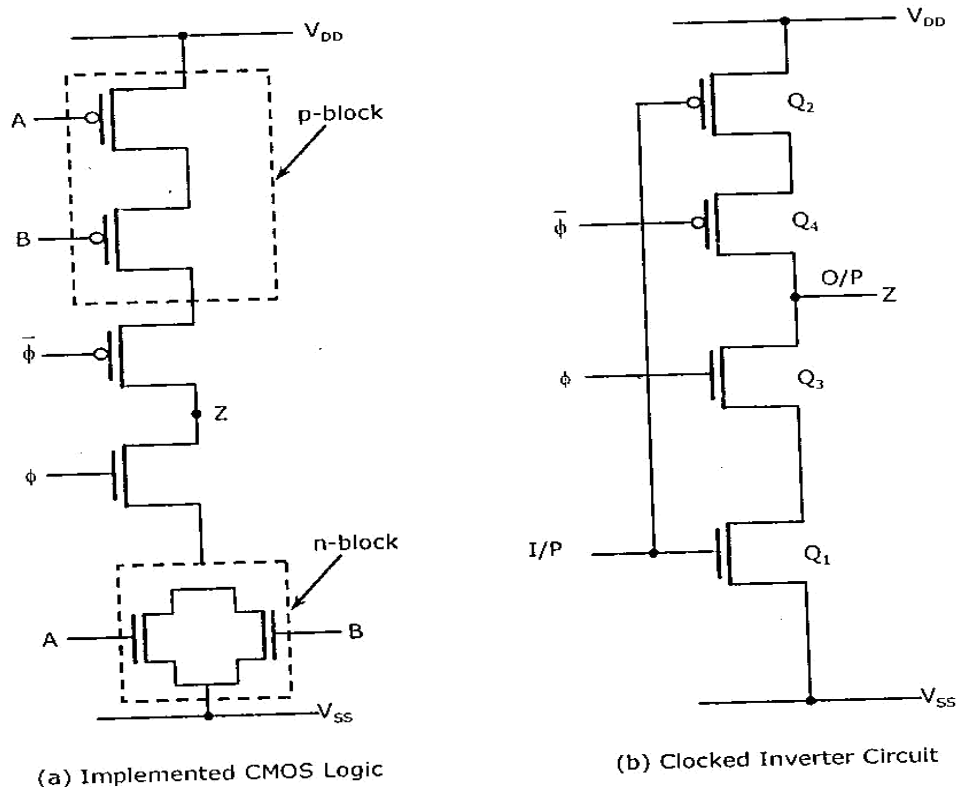


Fig: 19 Clocked CMOS logic

Working:

- During the pre charge phase $\phi = 0$, the output of the n-tree gate, OUT1, OUT3, are charged to V_{DD} , while the output of the p-tree gate OUT2 is pre discharged to 0V. Since the n-tree gate connects pMOS pull-up devices, the PUN of the p-tree is turned off at that time.
- During the evaluation phase $\phi = 1$, the outputs (OUT1,OUT3) of the n-tree gate can only make a 1 \rightarrow 0 transition, conditionally turning on some transistors in the p-tree. This ensures that no accidental discharge of OUT 2 can occur.
- Similarly n-tree blocks can follow p-tree gates without any problems, because the inputs to the n-gate are pre charged to 0.

Disadvantages:

Here, the p-tree blocks are slower than the n-tree modules, due to the lower current drive of the pMOS transistors in the logic network.

TIME DELAYS:

Consider the basic nMOS inverter has the channel length 8λ and width 2λ for pull-up transistor and channel length of 2λ and width 2λ for pull down transistor. Hence the sheet resistance for pull-up transistor is $R_{p,u} = 4R_S = 40k\Omega$ and sheet resistance for pull-down transistor is $R_{p,d} = 1R_S = 10k\Omega$.

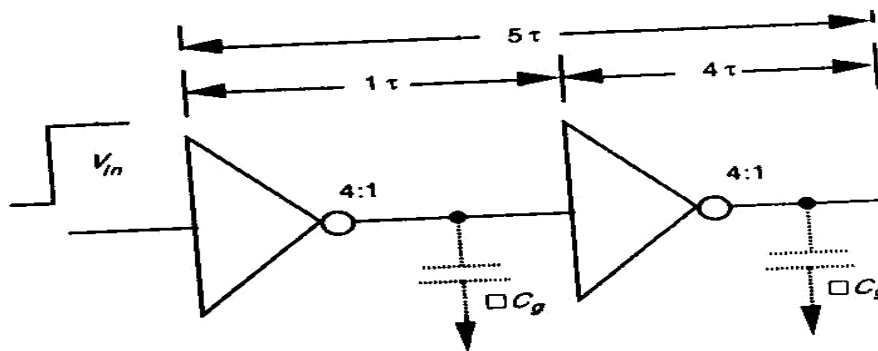


Fig: 21(a) Cascaded CMOS inverter

Since $(\tau = RC)$ depends upon the values of R & C , the delay associates with the inverter depend up on whether it is being turned on (or) off. Now, consider a pair of cascaded inverters as shown in figure, then the delay over the pair will be constant irrespective of the sense of the logic level transition of the input to the first.

In general, the delay through a pair of similar nMOS inverters is $T_d = (1 + Z_{p,u}/Z_{p,d}) \tau$

Assume that $\tau = 0.3 \text{ n sec.}$

$$\begin{aligned} \text{Then, } T_d &= (1 + 4) 0.3 \\ &= 5 \tau \end{aligned}$$

Thus, the inverter pair delay for inverters having 4:1 ratio is 5τ .

Hence, a single 4:1 inverter exhibits undesirable asymmetric delays, Since the delay in turning ON is τ and delay in turning OFF is 4τ .

CMOS inverter pair delay:

When we consider CMOS inverters, the rules for nMOS inverters are not applicable. But we need to consider the natural (R_S) uneven values for equal size pull up p-transistor and the n-type pull down transistors.

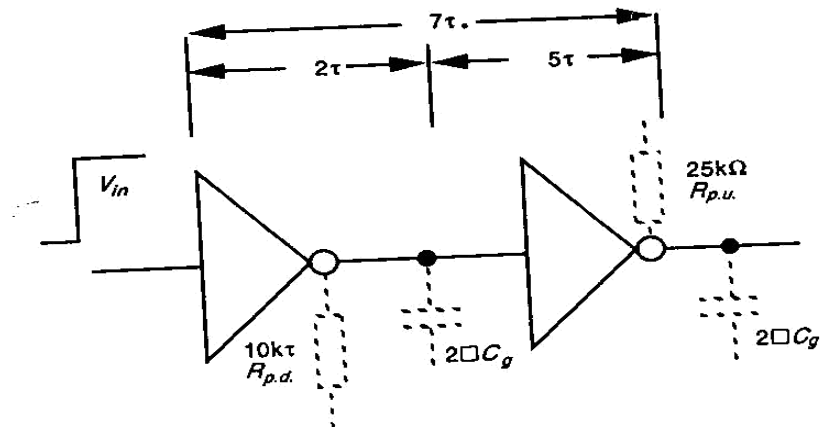


Fig: 21(b) Cascaded CMOS inverter

Figure shows the theoretical delay associated with a pair of both n and p transistors lambda based inverters. Here the gate capacitance is double comparable to nMOS inverter since the input to a CMOS inverter is connected to both transistor gate.

NOTE: Here the asymmetry (uneven) of resistance values can be eliminated by increasing the width of the p-device channel by a factor of two or three at the same time the gate capacitance of p-transistor also increased by the same factor.

Formal estimation of CMOS inverter delay:

In CMOS inverter by the charging and discharging of a capacitive load C_L , we can estimate the Rise time and fall time from the following simple analysis.

Rise time estimation:

In this analysis we assume that the p-device stays in saturation for the entire charging period of the load capacitor C_L . Consider the circuit as follows

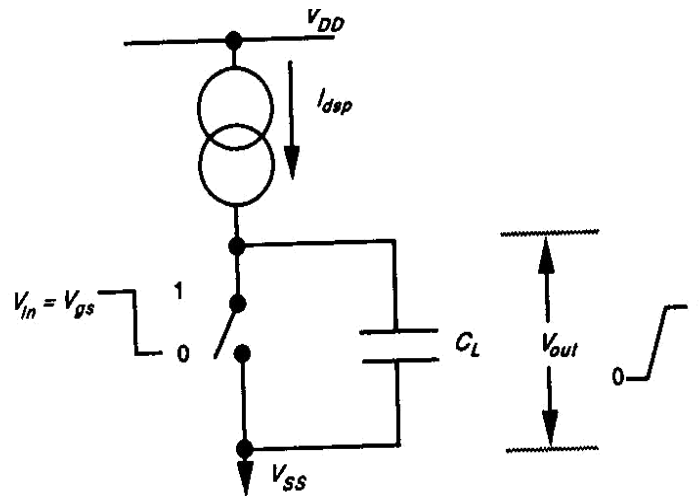


Fig: 21(c) Rise time estimation

Saturation current for the p-transistor is given by

$$I_{dsp} = \frac{\beta_p (V_{gs} - |V_{tp}|)^2}{2} \dots\dots\dots (1)$$

This current charges C_L and since its magnitude is approximately constant, we have

$$V_{out} = \frac{I_{dsp} t}{C_L} \dots\dots\dots (2)$$

Substitute the value of I_{dsp} in above equation and then the rise time is

$$t = \frac{2C_L V_{out}}{\beta_p (V_{gs} - |V_{tp}|)^2} \dots\dots\dots (3)$$

Assume that $t = \tau_r$ when $V_{out} = V_{DD}$ then

$$\tau_r = \frac{2V_{DD} C_L}{\beta_p (V_{DD} - |V_{tp}|)^2} \dots\dots\dots (4)$$

If $V_{tp} = 0.2V_{DD}$, then

$$\tau_r \doteq \frac{3C_L}{\beta_p V_{DD}} \dots\dots\dots(5)$$

Fall time estimation:

Consider the circuit for discharge of C_L through n-transistor as follows

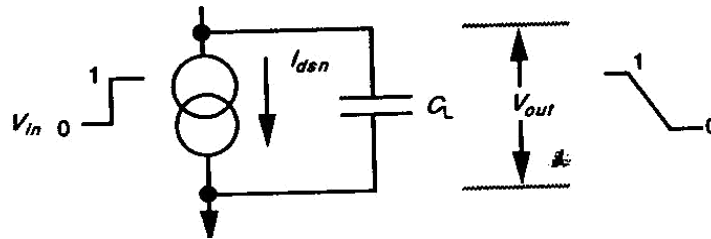


Fig: 21(d) Fall time estimation

By making similar assumptions we can write for fall-time estimation,

$$\tau_f \doteq \frac{3C_L}{\beta_n V_{DD}} \dots\dots\dots(6)$$

From the above two estimations we can deduce that

$$\frac{\tau_r}{\tau_f} = \frac{\beta_n}{\beta_p} \quad \text{Where } \mu_n = 2.5 \mu_p \dots\dots (7)$$

We know that

$$\beta_n \doteq 2.5\beta_p$$

So, the rise time is slower by a factor of 2.5 when using minimum size devices for both n & p.

- In order to achieve symmetrical operation using minimum channel length we need to make $W_p = 2.5 W_n$.
- For minimum size lambda based geometries this would result in the inverter having an input capacitance of

$$1 \Delta c_g (\text{n-device}) + 2.5 \Delta c_g (\text{p-device}) = 3.5 \Delta c_g$$

From the above equations we can conclude that

1. τ_r and τ_f are proportional to $1/V_{DD}$
2. τ_r and τ_f are proportional to C_L
3. $\tau_r = 2.5\tau_f$ for equal n and p- transistor geometries.

DRIVING LARGE CAPACITIVE LOADS:

When signals are propagated from the chip to off chip destinations we can face problems to drive large capacitive loads. Generally off chip capacitances may be several orders higher than on chip C_g values.

$$C_L \geq 10^4 \Delta c_g$$

Where C_L denotes off chip load. The capacitances which of this order must be driven through low resistances, otherwise excessively long delays will occur. Large capacitance is presented at the input, which in turn slows down the rate of change of voltage at input.

Cascaded Inverters as drivers:

Inverters to drive large capacitive loads must be present low pull-up and pull down resistance.

For MOS circuits low resistance values imply low L:W ratio (since $R_s = \frac{\rho L}{tw}$). Since length L cannot be reduced below the minimum feature size, the channels must be made very wide to reduce resistance value. Consider N cascaded inverters as on increasing the width factor of 'f' than the previous stage as shown in figure.

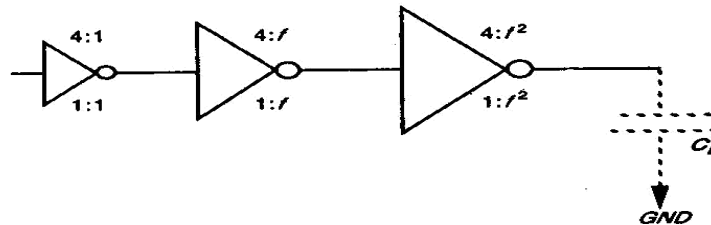


Fig: 22 Cascaded inverters as drivers

As the width factor increases, the capacitive load presented at the inverter input increases and the area occupied increases also. It is observed that as the width increases, the number N of stages are decreased to drive a particular value of C_L . Thus with large f(width), N decreases but delay per stage increases for 4:1 nMOS inverters.

Delay per stage = $f\tau$ for ΔV_{in}

= $4f\tau$ for $\Delta^+ V_{in}$

Where ΔV_{in} indicates logic 0 to 1 transition and $\Delta^- V_{in}$ indicates logic 1 to 0 transition of V_{in} .

Total delay per nMOS pair = $4f\tau$

Similarly delay per CMOS pair = $7f\tau$.

Calculation for time delay:

Let us assume $y = C_L / \square c_g = f^N$.

Determine the value of f which will minimize the overall delay for a given value of y . Apply logarithms on both sides in the above equation

$$\ln(y) = \ln(f^N) \quad \ln(y) = N \ln(f)$$

$$N = \ln(y) / \ln(f)$$

For N even : Total delay = $N/2 \cdot 5f\tau = 2.5 Nf\tau$ (nMOS).....(8)

(Or)

total delay = $N/2 \cdot 7f\tau = 3.5 Nf\tau$ (CMOS)(9)

From above relations, we can write

$$\text{Delay} \propto Nf\tau = \ln(y) / \ln(f) \cdot f\tau \quad \dots\dots(10)$$

It can be shown that total delay is minimized if f assumes the value of e for both CMOS and nMOS inverters.

Assume $f = e \rightarrow N = \ln(y) / \ln(e) \quad \dots\dots(11)$

$$N = \ln(y)$$

Overall delay $t_d \rightarrow$

N even: $t_d = 2.5 eN\tau$ (nMOS)
 (or)
 $t_d = 3.5 eN\tau$ (CMOS)(12)

N odd: $t_d = [2.5(N-1) + 1] e\tau$ (nMOS)
 $t_d = [3.5(N-1) + 2] e\tau$ (CMOS) (for logical transition 0 to 1)(13)
 (or)

$t_d = [2.5(N-1) + 4] e\tau$ (nMOS)
 $t_d = [3.5(N-1) + 5] e\tau$ (CMOS) (for logical transition 1 to 0)(14)

Super buffers:

Generally the pull-up and the pull down transistors are not equally capable to drive capacitive loads. This asymmetry is avoided in super buffers. Basically, a super buffer is a symmetric inverting or non inverting driver that can supply (or) remove large currents and is nearly symmetrical in its ability to drive capacitive load. It can switch large capacitive loads than an inverter. An inverting type nMOS super buffer as shown in figure.

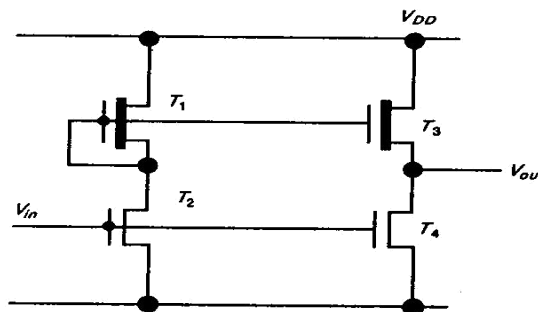


Fig: 24 (a) Super buffers

- Consider a positive going (0 to 1) transition at input V_{in} turns ON the inverter formed by T_1 and T_2 .
- With a small delay, the gate of T_3 is pulled down to 0 volts. Thus, device T_3 is cut off. Since gate of T_4 is connected to V_{in} , it is turned ON and the output is pulled down very fast.

- For the opposite transition of V_{in} (1 to 0), V_{in} drops to 0 volts. The gate of transistor T_3 is allowed to rise to V_{DD} quickly.
- Simultaneously the low V_{in} turns off T_4 very fast. This makes T_3 to conduct with its gate voltage approximately equal to V_{DD} .
- This gate voltage is twice the average voltage that would appear if the gate was connected to the source as in the conventional nMOS inverter.

Now as $I_{ds} \propto V_{gs}$, doubling the effective V_{gs} increases the current and there by reduces the delay in charging at the load capacitor of the output. The result is more symmetrical transition.

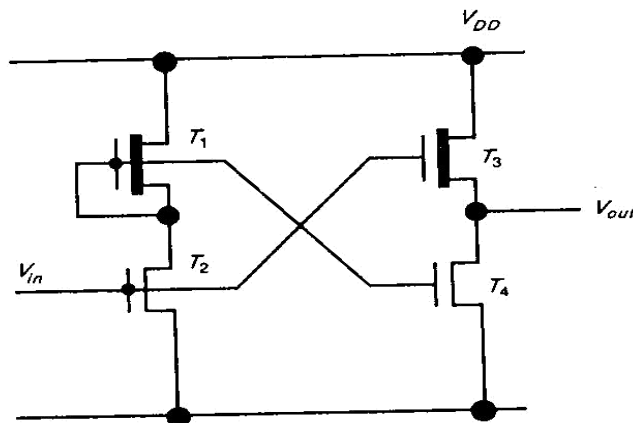


Fig: 25 (b) Operation of super buffers

Figure shows the non-inverting nMOS super buffer where the structures fabricated in $5\mu\text{m}$ technology are capable of driving capacitance of 2pF with a rise time of 5nsec .

BiCMOS drivers:

1. In BiCMOS technology we use bipolar transistor drivers as the output stage of inverter and logic gate circuits.
2. In bipolar transistors, there is an exponential dependence of the collector (output) current on the base to emitter (input) voltage V_{be} .
3. Hence, the bipolar transistors can be operated with much smaller input voltage swings than MOS transistors and still switch large current.
4. Another consideration in bipolar devices is that the temperature effect on input voltage V_{be} .
5. In bipolar transistor, V_{be} is logarithmically dependent on collector current I_C and also other parameters such as base width, doping level, electron mobility.
6. Now, the temperature differences across an IC are not very high. Thus the V_{be} values of the

bipolar devices spread over the chip remain same and do not differ by more than a few milli volts.

The switching performance of a bipolar transistor driving a capacitive load can be analyzed to begin with the help of equivalent circuit as shown in figure.

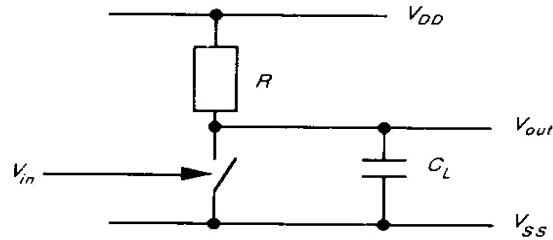


Fig: 24 (c) Equivalent circuit

The time Δt required to change the output voltage V_{out} by an amount equal to the input voltage is

$$\Delta t = C_L / g_m \dots\dots\dots(15)$$

Where, C_L is the load capacitance

g_m is the trans conductance of the bipolar transistor.

The value of Δt is small because the trans conductance of the bipolar transistors is relatively high. There are two main components which reveals the delay due to the bipolar transistors are T_{in} and T_L .

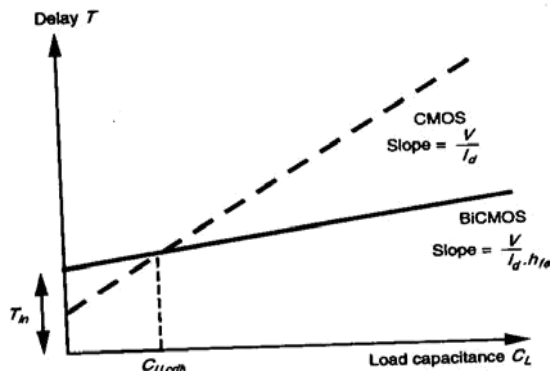


Fig: 25 T_{in} vs T_L

- T_{in} is the time required to first charge the base emitter junction of the bipolar (npn) transistor. This time is typically 2ns for the BiCMOS transistor base driver.
- For the CMOS driver the time required to charge the input gate capacitance is 1ns.
- T_L is the time required to charge the output load capacitance.
- The combined effect of T_{in} and T_L is represented as shown in figure.
- Delay of BiCMOS inverter can be described by

$$T = T_{in} + (V/I_d) (1/h_{fe}) C_L \dots\dots\dots(16)$$

- Delay for BiCMOS inverter is reduced by a factor of h_{fe} as compared with a CMOS inverter.
- In Bipolar transistors while considering delay another significant parameter is collector resistance R_c through which the charging current for C_L flows.
- For a high value of R_c , there is a long propagation delay through the transistor when charging a capacitive load.
- Figure shows the typical delay values at two values of C_L as follows.

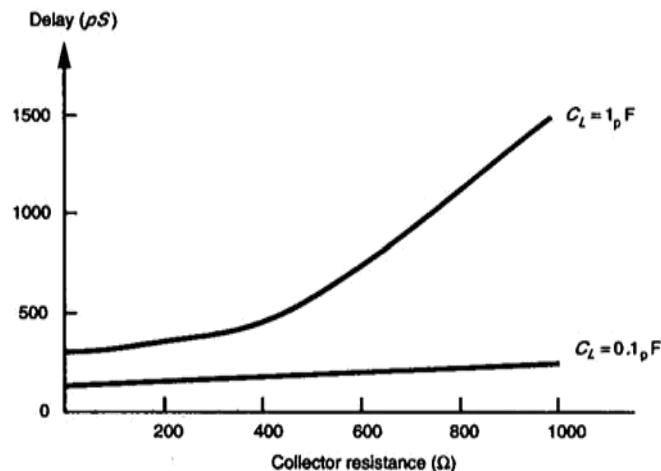


Fig: 26 Delay vs collector resistance

The devices thus have high β , high g_m , high h_{fe} and low R_c . The presence of such efficient and advantageous devices on chip offers a great deal of scope and freedom to the VLSI designer.

Propagation delays:

Propagation delay is the delay in the propagation of the signal created by the change of logical status at the input to create same change at the output.

Cascaded pass transistors:

Figure shows a chain of four pass transistors driving a capacitive load C_L . All the gates are supplied by V_{DD} so that a signal can propagate to the output. The lumped RC equivalent circuit is shown in figure, where each transistor is modeled by a series resistance and capacitance representing the gate-to-channel capacitance and stray capacitances. The minimum value of R is the turned ON resistance of each enhancement mode pass transistor.

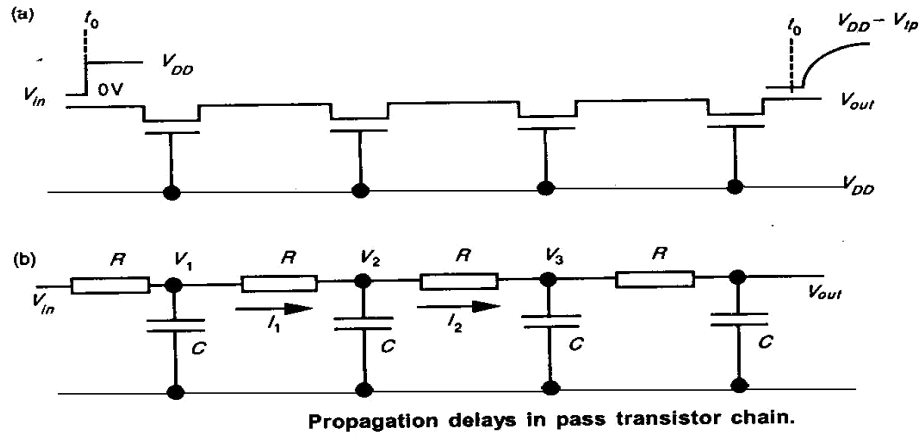


Fig: 27 Chain of pass transistors

The current through the capacitance at the node with voltage V_2 is

$$C (dV_2 / dt) \approx C \cdot \Delta V_2 / \Delta t \dots\dots\dots(17)$$

The current entering at this node is $I_1 = (V_1 - V_2)/R$ and the current leaving from this node is

$$I_2 = (V_2 - V_3)/R \dots\dots\dots(18)$$

By applying KCL at this node

$$I_C = I_1 - I_2$$

$$C \cdot \Delta V_2 / \Delta t = I_1 - I_2 = ((V_1 - V_2) - (V_2 - V_3)) / R \dots\dots(19)$$

As the number of sections in the network increases, the circuit parameters become distributed. Assume that R and C as the resistance per unit length and the capacitance per unit length respectively.

$$C \Delta \cdot \Delta V_2 / \Delta t = \Delta(\Delta V_2) / R \cdot \Delta X \dots\dots\dots(20)$$

Where x is the distance along the network from the input.

$$RC \, dv/dt = d/dx. \, (dv/dx) = d^2V/dx^2 \dots\dots\dots(21)$$

The propagation time τ_p from a signal to propagate a distance x is

$$\tau_p \propto X^2 \dots\dots\dots(22)$$

By simplifying the analysis if all sheet resistance, gate-to-channel capacitance R_S and $\square c_g$ are lumped together

$$R_{\text{total}} = nr R_S \text{ and } C_{\text{total}} = nc \square c_g \dots\dots\dots(23)$$

Where r gives relative resistance per section interms of R_S and c gives relative capacitance per section interms of $\square c_g$. Then the overall delay for n sections is given by

$$\tau_p = n^2 rc(\tau) \dots\dots\dots(24)$$

It can be shown that the signal delay in a section containing N identical pass transistors driving a matched load ($C_L = C_g$) is

$$\tau_p = 0.7 * N(N+1)/2 * RC_L \dots\dots\dots(25)$$

For large value of N , the quantity $(N + 1)$ can be replaced by N . Since the delay increases with N , the number of pass transistors is restricted to 4. A cascade of more pass transistors will produce a very slow circuit and the signal needs to be restored by an inverter after every three (or) four pass transistor.

WIRING CAPACITANCES:

The significant sources of capacitance which contribute to the overall wiring capacitance are as follows

(i)Fringing fields

Capacitance due to fringing field effects can be a major component of the overall

capacitance of interconnect wires. For fine line metallization, the value of fringing field capacitance (C_{ff}) can be of the same order as that of the area capacitance. Thus, C_{ff} should be taken into account if accurate prediction of performance is needed.

$$C_{ff} = \epsilon_{SiO_2} \epsilon_0 l \left[\frac{\pi}{1n \left\{ 1 + \frac{2d}{t} \left(1 + \sqrt{1 + \frac{t}{d}} \right) \right\}} - \frac{t}{4d} \right] \dots\dots\dots(26)$$

Where l = wire length

t = thickness of wire

d = wire to substrate separation.

Then, total wire capacitance $C_w = C_{area} + C_{ff}$ (27)

(ii) Interlayer capacitances:

From the definition of capacitance itself, it can be said that there exists a capacitance between the layers due to parallel plate effects. This capacitance will depend upon the layout i.e., where the layers cross or whether one layer underlies another etc., by the knowledge of these capacitances, the accuracy of circuit modeling and delay calculations will be improved. It can be readily calculated for regular structures.

(iii) peripheral capacitances:

- The source and drain p-diffusion regions forms junctions with the n-substrate (or n-well) at well defined and uniform depths.
- Similarly, the source and drain n-diffusion regions forms junctions with p-substrate (or p-well) at well defined and uniform depths.
- Hence, for diffusion regions, each diode thus formed has associated a peripheral (side wall) capacitance with it.
- As a whole the peripheral capacitance, C_p will be the order of pF/unit length. So its value will be greater than C_{area} of the diffusion region to substrate.

- C_p increases with reduction in source or drain area.
- Total diffusion capacitance is $C_{diff} = C_{area} + C_p$(28)

However, as the n and p-active regions are formed by impure implants at the surface of the silicon incase of orbit processes, they have negligible depth. Hence C_p is quite negligible in them.

Typical values are given in tabular form

Diffusion capacitance	Typical values		
	5 μ m	2 μ m	1.2 μ m
Area C (C_{area})	1.0×10^{-4} pF/ μ m ²	1.75×10^{-4} pF/ μ m ²	3.75×10^{-4} pF/ μ m ²
Periphery (C_{periph})	8.0×10^{-4} pF/ μ m ²	Negligible (assuming implanted regions of negligible depth)	negligible

FAN – IN AND FAN-OUT:

Fan-in: The number of inputs to a gate is called as fan - in.

Fan-out: The maximum number of similar gates that a gate can drive while remaining within the guaranteed specifications is called as fan-out.

Effects of Fan-in and Fan-out on propagation delay:

- An additional input to a CMOS logic gate requires an additional nMOS and pMOS i.e., two additional transistors, while incase of other MOS logic gates, it requires one additional transistor.
- In CMOS logic gates, due to these additional transistors, not only the chip area but also the total effective capacitance per gate also increased and hence propagation delay increases. Some of the increase in propagation delay time can be compensated by the

size-scaling method. By increasing the size of the device, its current driving capability can be preserved.

- Due to increase in both of inputs and devices size, the capacitance increases, Hence propagation delay will still increase with fan-in.
- An increase in the number of outputs of a logic gate directly adds to its load capacitances. Hence, the propagation delay increases with fan-out.

CHOICE OF LAYERS:

The following are the constraints which must be considered for the proper choice of layers.

1. Since the polysilicon layer has relatively high specific resistance (R_S), it should not be used for routing V_{DD} and V_{SS} (GND) except for small distances.
2. V_{DD} and GND (V_{SS}) must be distributed only on metal layers, due to the consideration of R_S value.
3. The capacitive effects will also impose certain restrictions in the choice of layers as follows
 - (i) Where a fast signal line are required, and in relation to signals on wiring which has relatively higher values of R_S .
 - (ii) The diffusion areas have higher values of capacitance to substrate and are harder to drive.
4. Over small equipotential regions, the signal on a wire can be treated as being identical at all points.
5. Within each region the propagation delay of the signal will comparably smaller than the gate delays and signal delays caused in a system connected by wires.

Thus the wires in a MOS system can be modeled as simple capacitors. This concept leads to the establishment of electrical rules (guidelines) for communication paths (wires) as given in tabular form.

Layer	Maximum length of communication wire		
	Lambda based 5 μ m	μ m based (2 μ m)	μ m – based(1.2 μ m)
Metal	Chip wide	Chip wide	Chip wide
Silicide	2,000 λ	NA	NA
Polysilicon	200 λ	400 μ m	250 μ m
Diffusion (active)	20 λ	100 μ m	60 μ m

CHAPTER IV

DATA PATH SUBSYSTEMS

Introduction

Most digital functions can be divided into the following categories:

1. Datapath operators
2. Memory elements
3. Control structures
4. Special-purpose cells
 - I/O
 - Power distribution
 - Clock generation and distribution
 - Analog and RF

CMOS system design consists of partitioning the system into subsystems of the types listed above. Many options exist that make trade-offs between speed, density, programmability, ease of design, and other variables. This chapter addresses design options for common datapath operators. The next chapter addresses arrays, especially those used for memory. Control structures are most commonly coded in a hardware description language and synthesized.

Datapath operators benefit from the structured design principles of hierarchy, regularity, modularity, and locality. They may use N identical circuits to process N -bit data. Related data operators are placed physically adjacent to each other to reduce wire length and delay. Generally, data is arranged to flow in one direction, while control signals are introduced in a direction orthogonal to the data flow.

Common data path operators considered in this chapter include adders, one/zero detectors, comparators, counters, shifters, ALUs, and multipliers.

Shifters

Consider a direct MOS switch implementation of a 4X4 crossbar switch as shown in Fig. 4.1. The arrangement is quite general and may be readily expanded to accommodate n -bit inputs/outputs. In fact, this arrangement is an overkill in that any input line can be connected to any or all output lines—if all switches are closed, then all inputs are connected to all outputs in one glorious short circuit. Furthermore, 16 control signals (sw_0)- sw_{15} , one for each transistor switch, must be provided.

to drive the crossbar switch, and such complexity is highly undesirable. An adaption of this arrangement) Recognizes the fact that we can couple the switch gates together in groups of four (in this case) and also form four separate groups corresponding to shifts of zero, one, two, and three bits. The arrangement is readily adapted so that the inlines also run horizontally (to conform the required strategy). The resulting arrangement is known as barrel shifter and a 4X4-bit barrel shifter circuit diagram is given in Fig. 4.2. The interbus switches have their gate inputs connected in staircase fashion in group of four and there are now four shift control inputs which must be mutually exclusive in active state. CMOS transmission gates may be used in place of the simple pass transistor switches if appropriate.

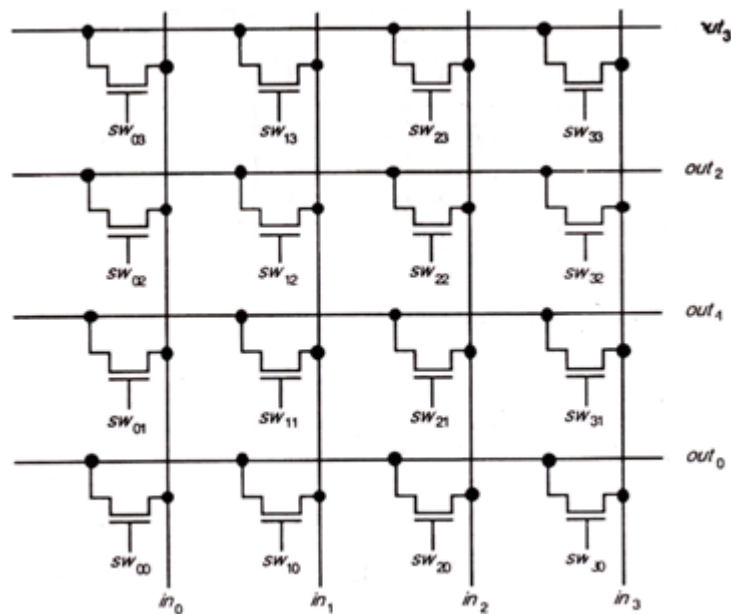


Figure 1: 4 x 4 crossbar switch.

Adders

Addition is one of the basic operations performed in various processing like counting, multiplication and filtering. Adders can be implemented in various forms to suit different speed and density requirements.

The truth table of a binary full adder is shown in Figure 4.3, along with some functions that will be of use during the discussion of adders. Adder inputs: A, B

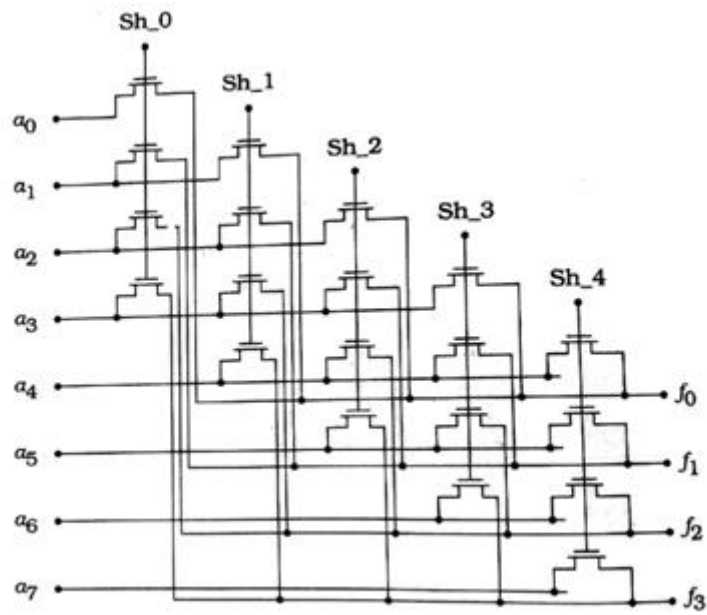


Figure 2: Barrel shifter

C	A	B	A.B (G)	A+B (P)	$A \oplus B$	SUM	CARRY
0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	0
0	1	0	0	1	1	1	0
0	1	1	1	1	0	0	1
1	0	0	0	0	0	1	0
1	0	1	0	1	1	0	1
1	1	0	0	1	1	0	1
1	1	1	1	1	0	1	1

Figure 3: Full adder truth table

Carry input: SUM

Carry output: CARRY

Generate signal: $G(A, B)$; occurs when CARRY is internally generated within adder

Propagate signal: P (A + B); when it is 1, C is passed to CARRY. In some adders A B is used as the P term because it may be reused to generate the sum term.

Single-Bit Adders

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

From the truth table these are:

$$\begin{aligned} \text{SUM} &= \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC \\ &= C(\bar{A}\bar{B} + A\bar{B}) + C(\bar{A}B + AB) \\ &= \bar{C}(A \oplus B) + C(\overline{A \oplus B}) \\ &= A \oplus B \oplus C \end{aligned}$$

$$\text{CARRY} = AB + AC + BC$$

$$= AB + C(A + B)$$

$$= \overline{A\bar{B} + C(\bar{A} + \bar{C})}$$

The direct implementation of the above equations is shown in Fig. 4 using the gate schematic and the transistors is shown in Fig. 4.

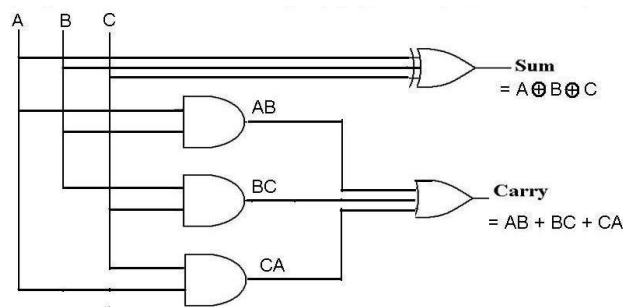


Figure 4: Logic gate implementation of 1-Bit adder

The full adder employs 32 transistors (6 for the inverters, 10 for the carry circuit, and 16 for the 3-input XOR). A more compact design is based on the observation that S can be factored to reuse the CARRY term as follows:

$$\begin{aligned} \text{SUM} &= A.B.C + (A + B + C).CARRY \\ &= A.B.C + (A + B + C).\overline{A\bar{B} + C.(A + B)} \end{aligned}$$

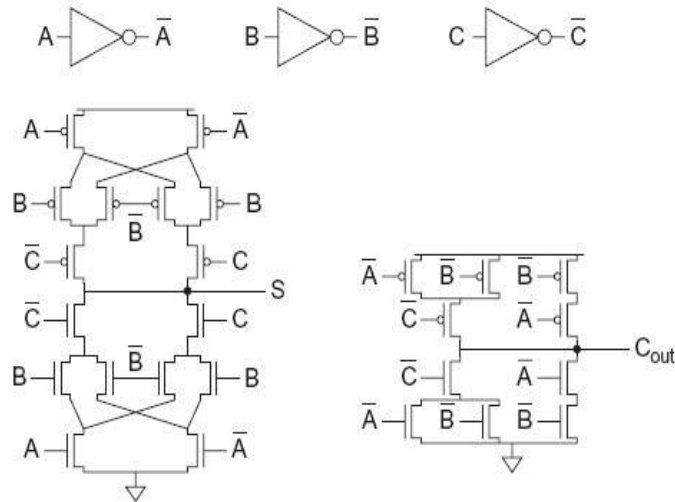


Figure 5: Transistor implementation of 1-bit adder

Such a design is shown at the transistor levels in Figure 5 and uses only 28 transistors. Note that the pMOS network is complement to the nMOS network.

Here $C_{in}=C$

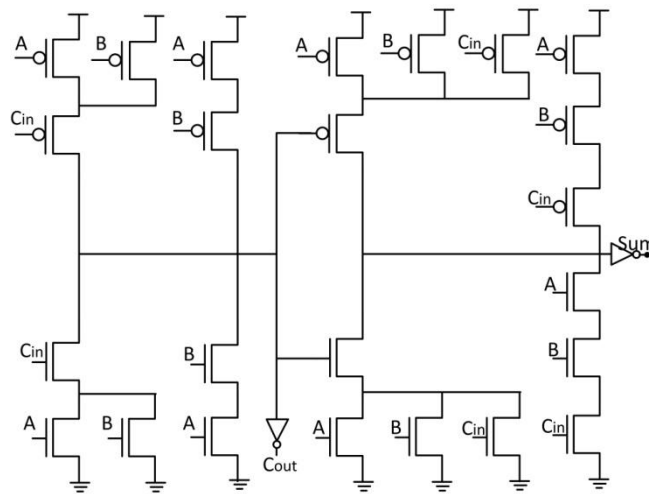


Figure 6: Transistor implementation of 1-bit adder

n-bit Parallel Adder or Ripple Carry Adder

A ripple carry adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascaded, with the carry output from each full adder connected to the carry input of the next full adder in the chain. Figure 6 shows the interconnection of four full adder (FA) circuits to provide a 4-bit ripple carry adder. Notice from Figure 6 that the input is from the right side because the first cell traditionally represents the least

significant bit (LSB). Bits a_0 and b_0 in the figure represent the least significant bits of the numbers to be added. The sum output is represented by the bits S_0 - S_3 .

Carry lookahead adder (CLA)

The carry lookahead adder (CLA) solves the carry delay problem by calculating the carry signals in advance, based on the input signals. It is based on the fact that a carry signal will be generated in two cases: (1) when both bits a_i and b_i are 1, or

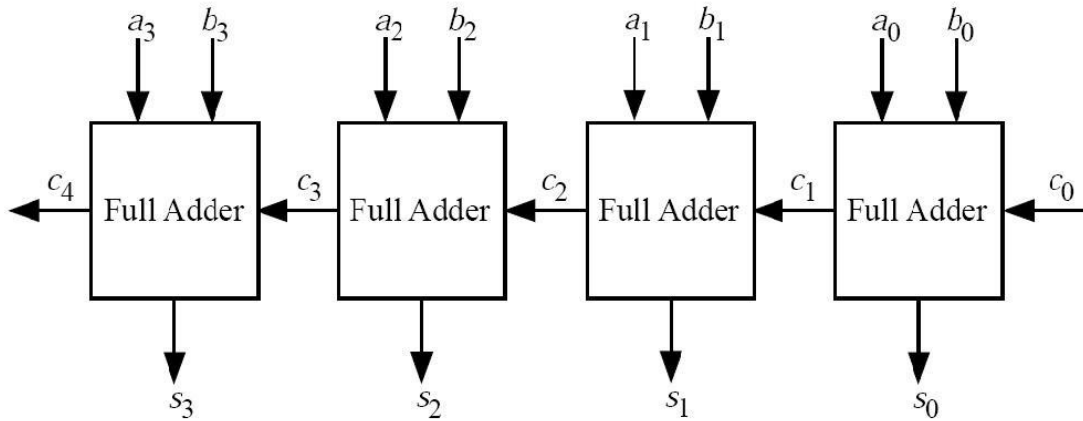


Figure 7: 4-bit ripple carry adder

(2) when one of the two bits is 1 and the carry-in is 1. Thus, one can write,

$$c_{i+1} = a_i \cdot b_i + (a_i \oplus b_i) \cdot c_i$$

$$s_i = (a_i \oplus b_i) \oplus c_i$$

The above two equations can be written in terms of two new signals P_i and G_i , which are shown in Figure 8:

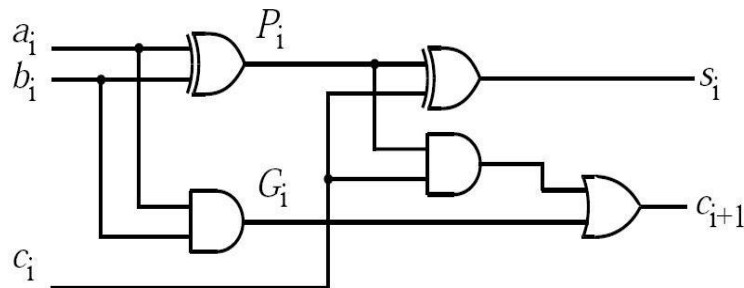


Figure 8: Full adder stage at i with P_i and G_i shown

$$c_{i+1} = G_i + P_i \cdot c_i$$

$$s_i = P_i \oplus c_i$$

Where

$$G_i = a_i \cdot b_i$$

$$P_i = (a_i \oplus b_i)$$

P_i and G_i are called carry propagate and carry generate terms, respectively. Notice that the generate and propagate terms only depend on the input bits and thus will be valid after one and two gate delay, respectively. If one uses the above expression to calculate the carry signals, one does not need to wait for the carry to ripple through all the previous stages to find its proper value. Let's apply this to a 4-bit adder to make it clear.

Putting $i = 0; 1; 2; 3$ in $c_{i+1} = G_i + P_i \cdot c_i$ we get

$$c_1 = G_0 + P_0 \cdot c_0$$

$$c_2 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot c_0$$

$$c_3 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot c_0$$

$$c_4 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot c_0$$

Notice that the carry-out bit, c_{i+1} , of the last stage will be available after four delays: two gate delays to calculate the propagate signals and two delays as a result of the gates required to implement Equation c_4 .

Figure 9 shows that a 4-bit CLA is built using gates to generate the P_i and G_i and signals and a logic block to generate the carry out signals according to Equations c_1 to c_4 .

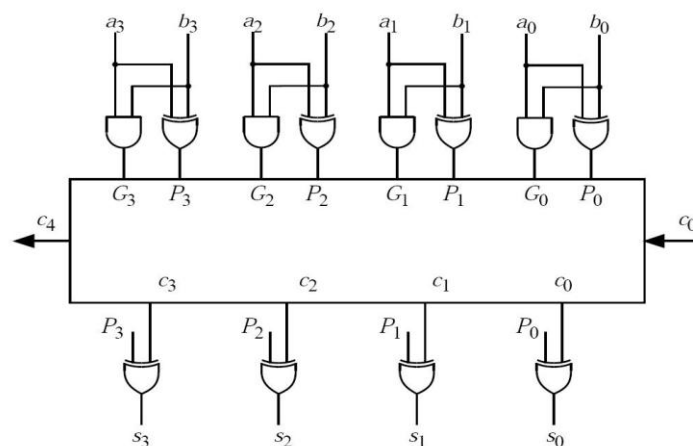
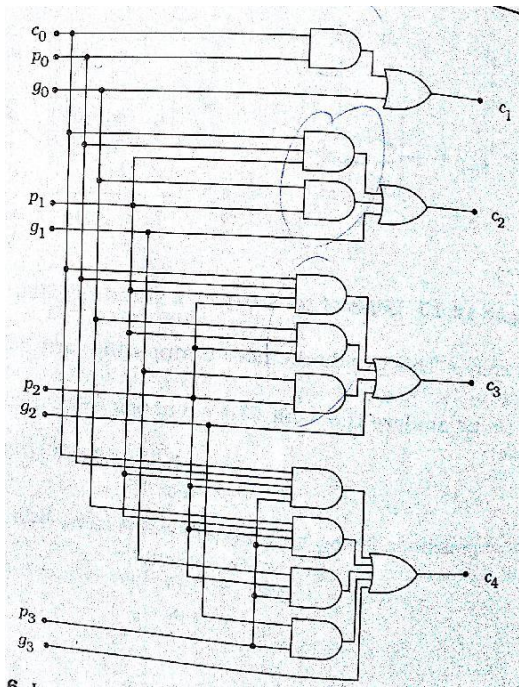
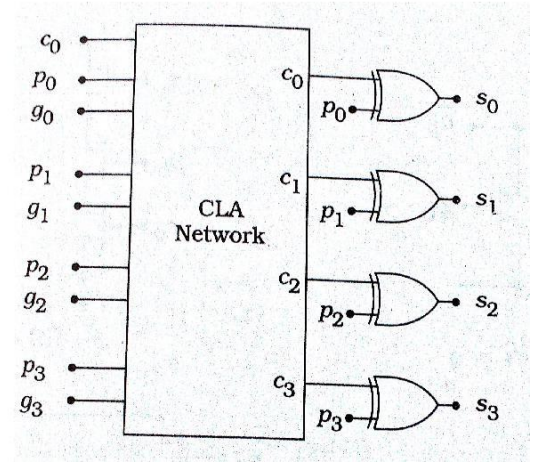


Figure 9: 4-Bit carry lookahead adder implementation in detail

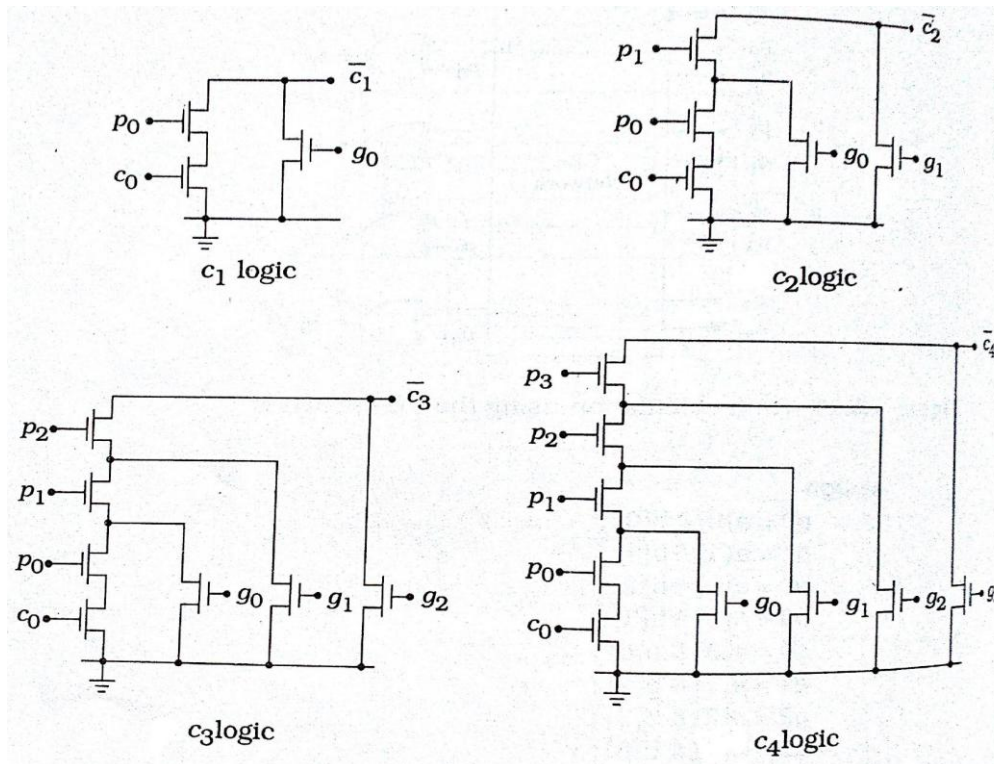
The disadvantage of CLA is that the carry logic block gets very complicated for more than 4-bits. For that reason, CLAs are usually implemented as 4-bit modules and are used in a hierarchical structure to realize adders that have multiples of 4-bits.



(a) Logic network for 4-bit CLA carry bits network



(b) Sum calculation using CLA



(c) nFET logic arrays for the CLS terms

Figure 10: Carry structures of CLA

Manchester carry chain

This implementation can be very performant (20 transistors) depending on the way the XOR function is built. The carry propagation of the carry is controlled by the output of the XOR gate. The generation of the carry is directly made by the function at the bottom. When both input signals are 1, then the inverse output carry is 0. In the schematic of Figure 11, the carry passes through a complete

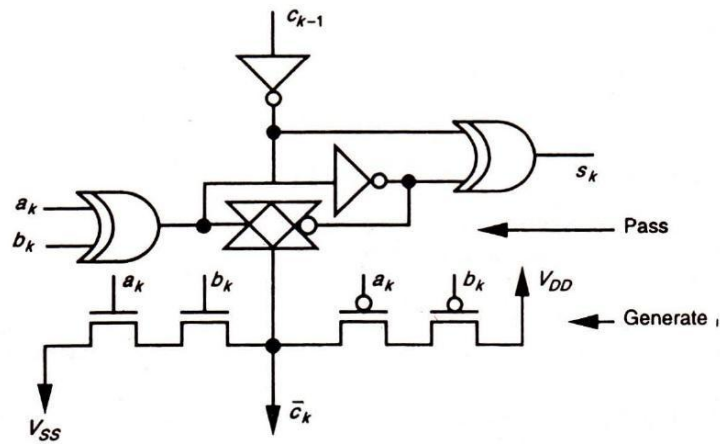


Figure 11: An adder element based on the pass/generate concept.

transmission gate. If the carry path is precharged to V_{DD} , the transmission gate is then reduced to a simple NMOS transistor. In the same way the PMOS transistors of the carry generation is removed. One gets a Manchester cell.

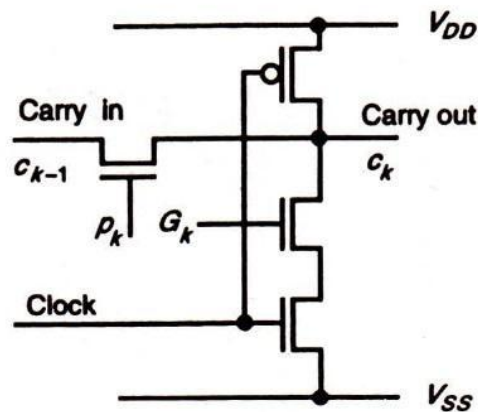


Figure 12: Manchester cell

The Manchester cell is very fast, but a large set of such cascaded cells would be slow. This is due to the distributed RC effect and the body effect making the propagation time grow with the square of the number of cells. Practically, an inverter is added every four cells, like in Figure12.

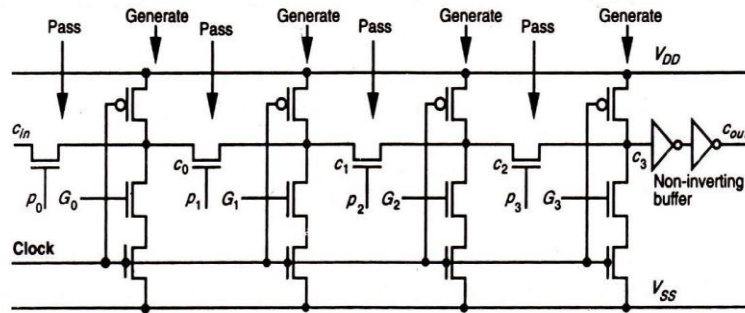


Figure 13: Cascaded Manchester carry-chain elements with buffering

Multipliers

In many digital signal processing operations - such as correlations, convolution, filtering, and frequency analysis - one needs to perform multiplication. The most basic form of multiplication consists of forming the product of two positive binary numbers. This may be accomplished through the traditional technique of successive additions and shifts, in which each addition is conditional on one of the multiplier bits. Here is an example.

multiplicand	1100 : 12 ₁₀
multiplier	0101 : 5 ₁₀

	1100
	0000
	1100
	0000

	0111100 : 60 ₁₀

Figure 14: 4-bit multiplication

The multiplication process may be viewed to consist of the following two steps:

- Evaluation of partial products.
- Accumulation of the shifted partial products.

It should be noted that binary multiplication is equivalent to a logical AND operation. Thus evaluation of partial products consists of the logical ANDing of the multiplicand and the relevant multiplier bit. Each column of partial products must then be added and, if necessary, any carry values passed to the next column.

There are a number of techniques that may be used to perform multiplication. In general, the choice is based on factors such as speed, throughput, numerical accuracy, and area. As a rule, multipliers may be classified by the format in which data words are accessed, namely:-

Serial form

Serial/parallel form

Parallel form

Array Multiplication

A parallel multiplier is based on the observation that partial products in the multiplication process may be independently computed in parallel. For example, consider the unsigned binary integers X and Y.

$$X = \sum_{i=0}^{m-1} X_i \cdot 2^i \quad \text{and} \quad Y = \sum_{j=0}^{n-1} Y_j \cdot 2^j$$

The product is found by

$$\begin{aligned} P &= X \times Y = \left(\sum_{i=0}^{m-1} X_i \cdot 2^i \right) \times \left(\sum_{j=0}^{n-1} Y_j \cdot 2^j \right) \\ &= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (X_i \cdot Y_j) \cdot 2^{i+j} \\ &= \sum_{k=0}^{m+n-1} P_k \cdot 2^k \end{aligned}$$

Thus P_k are the partial product terms called summands. There are mn summands, which are produced in parallel by a set of mn AND gates. For 4-bit numbers, the expression above may be expanded as in the table below.

				X3	X2	X1	X0	Multiplicand
				Y3	Y2	Y1	Y0	Multiplier
				X3Y0	X2Y0	X1Y0	X0Y0	
			X3Y1	X2Y1	X1Y1	X0Y1		
		X3Y2	X2Y2	X1Y2	X0Y2			
	X3Y3	X2Y3	X1Y3	X0Y3				
P7	P6	P5	P4	P3	P2	P1	P0	Product

Figure 15

An $n \times n$ multiplier requires

$(n-1)^2$ full adders,

$n-1$ half adders, and

n^2 AND gates.

The worst-case delay associated with such a multiplier is $(2n + 1)t_g$, where t_g is the worst-case adder delay.

Cell shown in Figure 16 is a cell that may be used to construct a parallel multiplier.

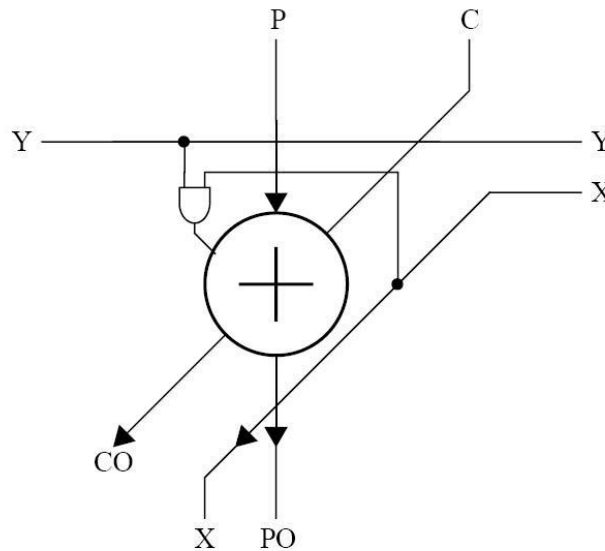


Figure 16: Basic cell to construct a parallel multiplier

The X_i term is propagated diagonally from top right to bottom left, while the y_j term is propagated horizontally. Incoming partial products enter at the top. Incoming CARRY IN values enter at the top right of the cell. The bit-wise AND is performed in the cell, and the SUM is passed to the next cell below. The CARRY OUT is passed to the bottom left of the cell.

Figure 17 depicts the multiplier array with the partial products enumerated. The Multiplier can be drawn as a square array, as shown here, Figure 18 is the most convenient for implementation. In this version the degeneration of the first two rows of the multiplier are shown. The first row of the multiplier adders has been replaced with AND gates while the second row employs half-adders rather than full adders.

This optimization might not be done if a completely regular multiplier were required (i.e. one array cell). In this case the appropriate inputs to the rst and second row would be connected to ground, as shown in the previous slide. An adder with equal carry and sum propagation times is advantageous, because the worst-case multiply time depends on both paths.

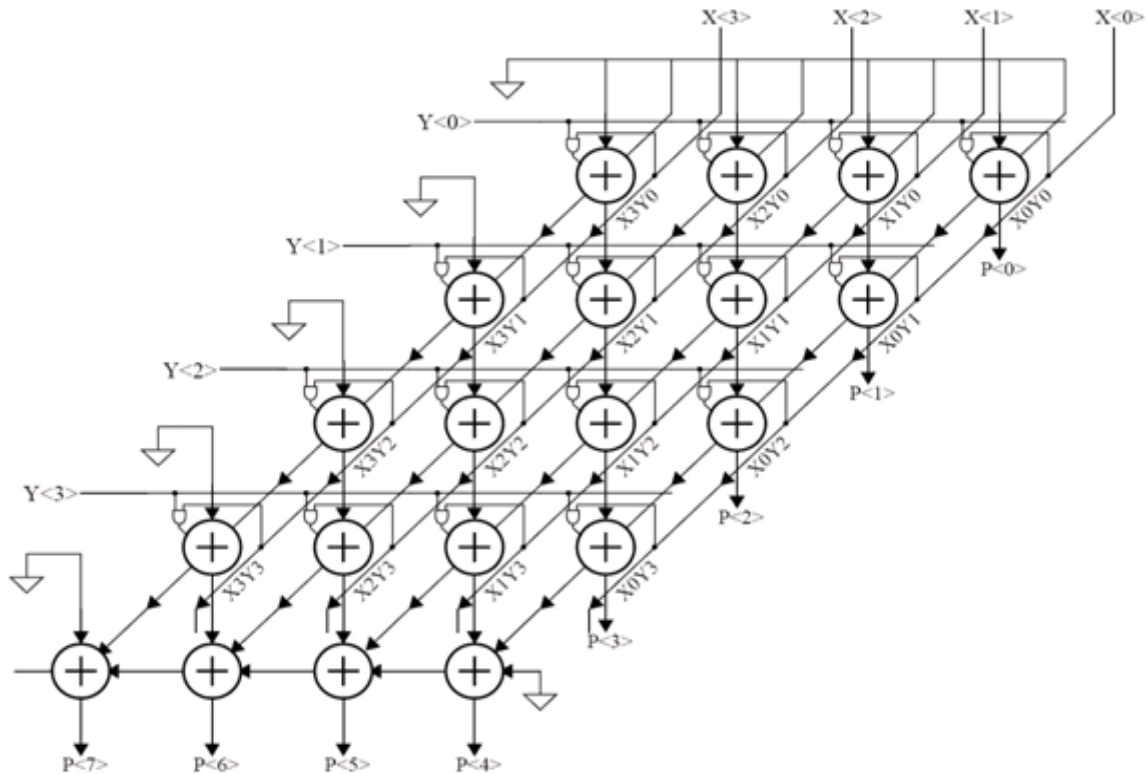


Figure 17: Array multiplier

Wallace Tree Multiplication

If the truth table for an adder, is examined, it may be seen that an adder is in effect a "one's counter" that counts the number of 1's on the A, B, and C inputs and encodes them on the SUM and CARRY outputs.

A 1-bit adder provides a 3:2 (3 inputs, 2 outputs) compression in the number of bits. The addition of partial products in a column of an array multiplier may be thought of as totaling up the number of 1's in that column, with any carry being passed to the next column to the left.

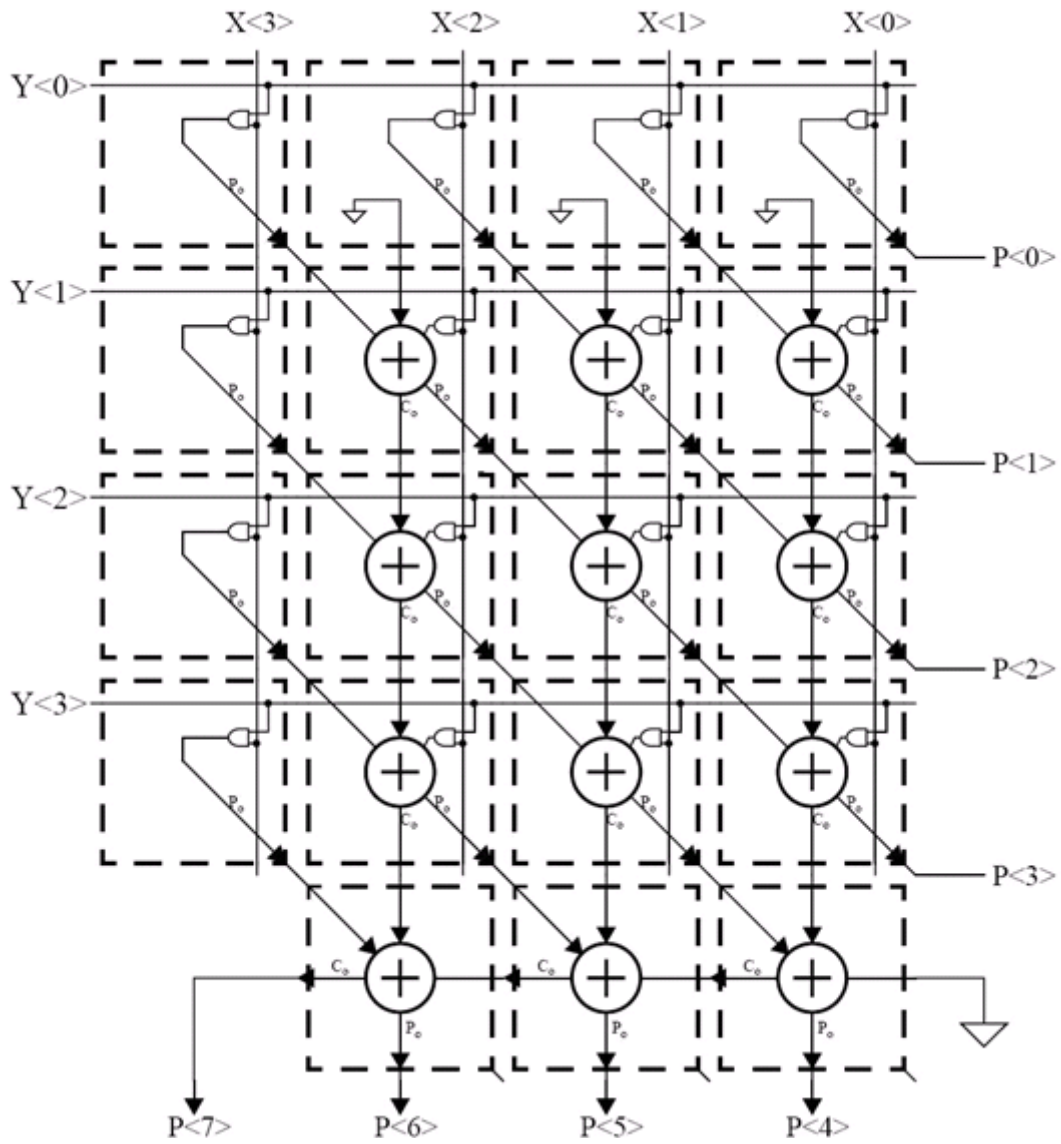


Figure 18: Most convenient way for implementation of array multiplier

ABC	Carry/Sum	Number of 1's
0 0 0	0 0	0
0 0 1	1 0	1
0 1 0	1 0	1
0 1 1	0 1	2
1 0 0	0 1	1
1 0 1	1 0	2
1 1 0	1 0	2
1 1 1	1 1	3

Figure 19

Example for implementation of 4x4 multiplier(4-bit) using Wallace Tree Multi-plication methods

				X3	X2	X1	X0	Multiplicand
				Y3	Y2	Y1	Y0	Multiplier
				X3Y0	X2Y0	X1Y0	X0Y0	
			X3Y1	X2Y1	X1Y1	X0Y1		
		X3Y2	X2Y2	X1Y2	X0Y2			
	X3Y3	X2Y3	X1Y3	X0Y3				
P7	P6	P5	P4	P3	P2	P1	P0	Product

Figure 20: Table to find product terms

Considering the product P3, it may be seen that it requires the summation of four partial products and a possible column carry from the summation of P2.

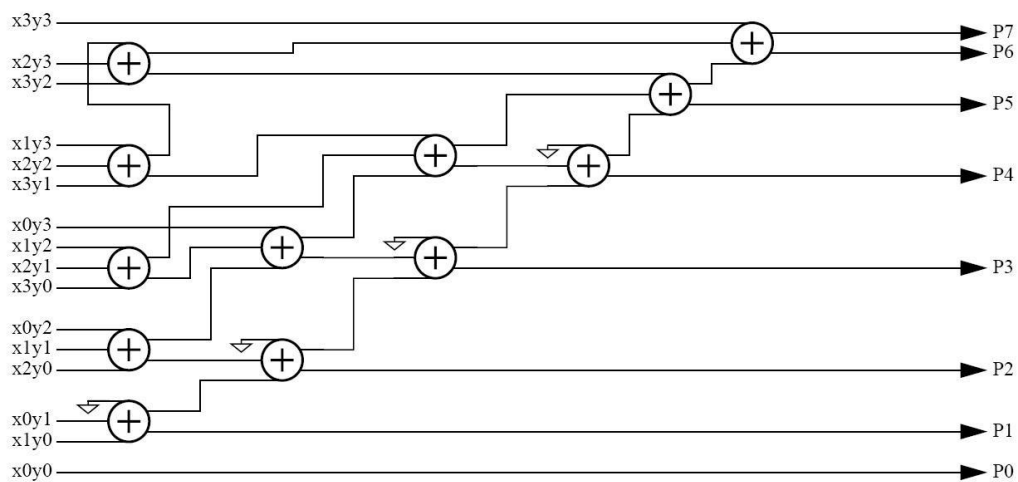


Figure 21: Wallace Tree Multiplication for 4-bits

Example for implementation of 6X6 multiplier(4-bit) using Wallace Tree Multi-plication methods.

Consider the 6 x 6 multiplication table shown below. Considering the product P5, it may be seen that it requires the summation of six partial products and a possible column carry from the summation of P4. Here we can see the adders required in a multiplier based on this style of addition.

The adders have been arranged vertically into ranks that indicate the time at which the adder output becomes available. While this small example shows the general Wallace addition technique, it does not show the real speed advantage of a Wallace tree. There is an identifiable array part, and a

CPA part, which is at the top right. While this has been shown as a ripple-carry adder, any fast CPA can be used here. The delay through the array addition (not including the CPA) is proportional to $\log_{1.5}(n)$, where n is the width of the Wallace tree.

						X5	X4	X3	X2	X1	X0	Multiplicand
						Y5	Y4	Y3	Y2	Y1	Y0	Multiplier
						X5Y0	X4Y0	X3Y0	X2Y0	X1Y0	X0Y0	
						X5Y1	X4Y1	X3Y1	X2Y1	X1Y1	X0Y1	
						X5Y2	X4Y2	X3Y2	X2Y2	X1Y2	X0Y2	
						X5Y3	X4Y3	X3Y3	X2Y3	X1Y3	X0Y3	
						X5Y4	X4Y4	X3Y4	X2Y4	X1Y4	X0Y4	
						X5Y5	X4Y5	X3Y5	X2Y5	X1Y5	X0Y5	
P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0	Product

Figure 22: 6 x 6 multiplication table

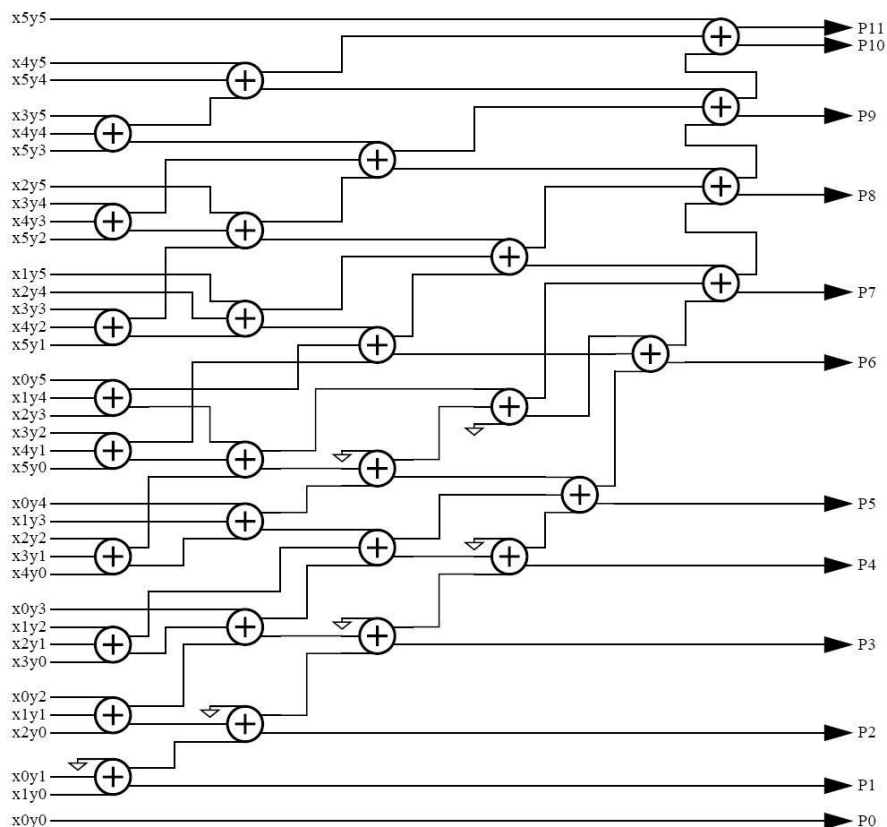


Figure 23: Wallace Tree Multiplication for 6-bits

Parity generator

1. Parity is a very useful tool in information processing in digital computers to indicate any presence of error in bit information.

2. External noise and loss of signal strength cause loss of data bit information while transporting data from one device to other device, located inside the computer or externally.
3. To indicate any occurrence of error, an extra bit is included with the message according to the total number of 1s in a set of data, which is called parity.
4. If the extra bit is considered 0 if the total number of 1s is even and 1 for odd quantities of 1s in a set of data, then it is called even parity.
5. On the other hand, if the extra bit is 1 for even quantities of 1s and 0 for an odd number of 1s, then it is called odd parity.

A parity generator is a combination logic system to generate the parity bit at the transmitting side.

Table 1.1: Truth table for generating even and odd parity bit

Four bit message $D_3D_2D_1D_0$	Even parity	Odd parity
0000	0	1
0001	1	0
0010	1	0
0011	0	1
01000	1	0
0101	0	1
0110	0	1
0111	1	0
1000	1	0
1001	0	1
1010	0	1
1011	1	0
1100	0	1
1101	1	0
1110	1	0
1111	0	1

If the message bit combination is designated as, $D_3D_2D_1D_0$ and P_e , P_o are the even and odd parity respectively, then it is obvious from the table that the Boolean expressions of even parity and odd parity are

$$P_e = D_3D_2D_1D_0$$

$$P_o = \overline{(D_3D_2D_1D_0)}$$

The above illustration is given for a message with four bits of information.

However, the logic diagrams can be expanded with more XOR gates for any number of bits.

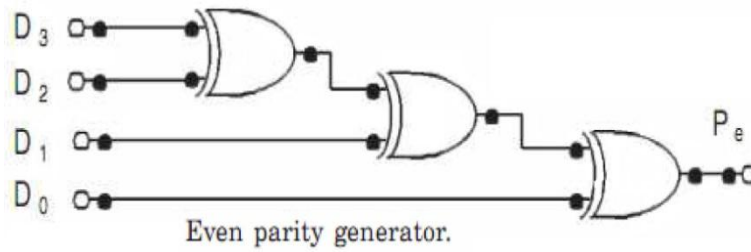


Figure 24: Even parity generator using logic gates

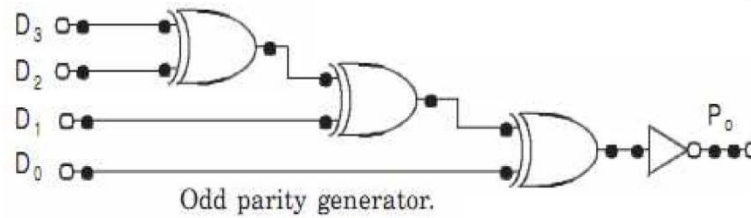


Figure 25: Odd parity generator logic gates

Zero/One detector

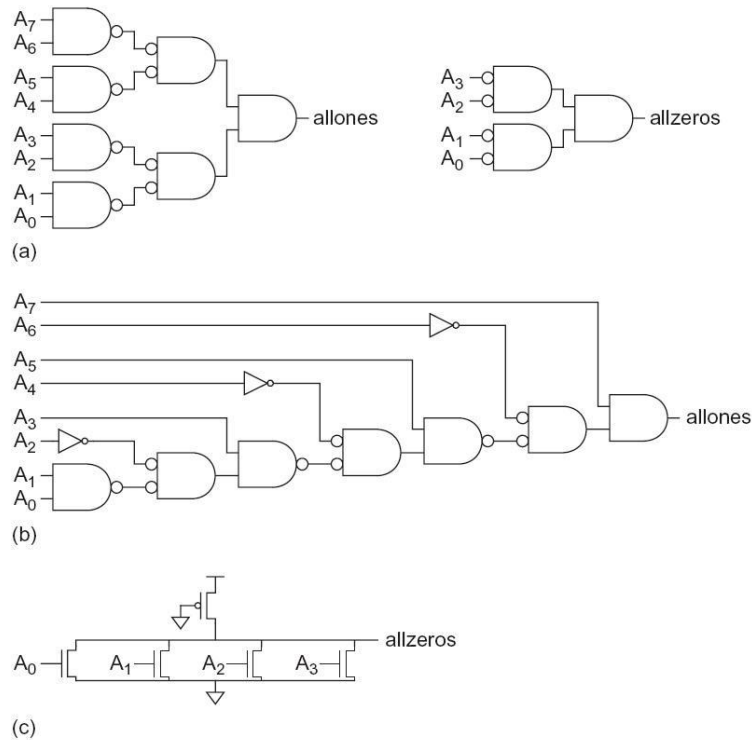


Figure 26: One/zero detectors (a) All one detector (b) All zero detector (c) All zero detector transistor level representation

Detecting all ones or zeros on wide N-bit words requires large fan-in AND or NOR gates. Recall that by DeMorgan's law, AND, OR, NAND, and NOR are fundamentally the same operation except for possible inversions of the inputs and/or outputs. You can build a tree of AND gates, as shown in Figure 26(b). Here, alternate NAND and NOR gates have been used. The path has $\log N$ stages.

Comparators

Another common and very useful combinational logic circuit is that of the Digital Comparator circuit. Digital or Binary Comparators are made up from standard AND, NOR and NOT gates that compare the digital signals present at their input terminals and produce an output depending upon the condition of those inputs.

For example, along with being able to add and subtract binary numbers we need to be able to compare them and determine whether the value of input A is greater than, smaller than or equal to the value at input B etc. The digital comparator accomplishes this using several logic gates that operate on the principles of Boolean Algebra. There are two main types of Digital Comparator available and these are.

1. Identity Comparator an Identity Comparator is a digital comparator that has only one output terminal for when $A = B$ either "HIGH" $A = B = 1$ or "LOW" $A = B = 0$
2. Magnitude Comparator a Magnitude Comparator is a type of digital comparator that has three output terminals, one each for equality, $A = B$ greater than, $A > B$ and less than $A < B$

The purpose of a Digital Comparator is to compare a set of variables or unknown numbers, for example A ($A_1, A_2, A_3, \dots, A_n$, etc) against that of a constant or unknown value such as B ($B_1, B_2, B_3, \dots, B_n$, etc) and produce an output condition or signal depending upon the result of the comparison. For example, a magnitude comparator of two 1-bits, (A and B) inputs would produce the following three output conditions when compared to each other.

$$A > B; A = B; A < B$$

Which means: A is greater than B, A is equal to B, and A is less than B

This is useful if we want to compare two variables and want to produce an output when any of the above three conditions are achieved. For example, produce an output from a counter when a certain count number is reached. Consider the simple 1-bit comparator below. Then the operation of a 1-bit digital comparator is given in the following Truth Table.

Inputs		Outputs		
B	A	A > B	A=B	A < B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	0	0

From the above table the obtained expressions for magnitude comparator using K-map are as follows

For $A < B : C = \bar{A}B$

For $A = B : D = \bar{A}\bar{B} + AB$

For $A > B : E = A\bar{B}$ The logic diagram of 1-bit comparator using basic gates is shown

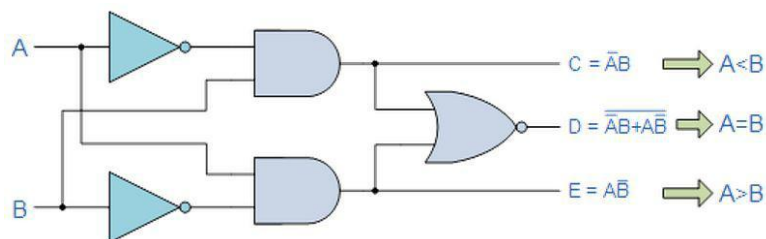


Figure 27: 1-bit Digital Comparator

Counters

Counters can be implemented using the adder/subtractor circuits and registers (or equivalently, D ip- ops)

The simplest counter circuits can be built using T ip- ops because the toggle feature is naturally suited for the implementation of the counting operation. Counters are available in two categories

1. Asynchronous(Ripple counters) Asynchronous counters, also known as ripple counters, are not clocked by a common pulse and hence every ip- op in the counter changes at different times. The ip- ops in an asynchronous counter is usually clocked by the output pulse of the preceding ip- op. The rst ip- op is clocked by an external event.

The ip- op output transition serves as a source for triggering other ip- ops i.e the C input (clock input) of some or all ip- ops are triggered NOT by the common clock pulses Eg:- Binary ripple counters, BCD ripple counters

- Synchronous counters A synchronous counter however, has an internal clock, and the external event is used to produce a pulse which is synchronized with this internal clock.

C input (clock input) of all ip- ops receive the common clock pulses

E.g.:- Binary counter, Up-down Binary counter, BCD Binary counter, Ring counter, Johnson counter,

Asynchronous Up-Counter with T Flip-Flops

Figure 28 shows a 3-bit counter capable of counting from 0 to 7. The clock inputs of the three ip- ops are connected in cascade. The T input of each ip- op is connected to a constant 1, which means that the state of the ip- op will be toggled at each active edge (here, it is positive edge) of its clock. We assume that the purpose of this circuit is to count the number of pulses that occur on the primary input called Clock. Thus the clock input of the rst ip- op is connected to the Clock line. The other two ip- ops have their clock inputs driven by the Q output of the preceding ip- op. Therefore, they toggle their states whenever the preceding ip- op changes its state from $Q = 1$ to $Q = 0$, which results in a positive edge of the Q signal.

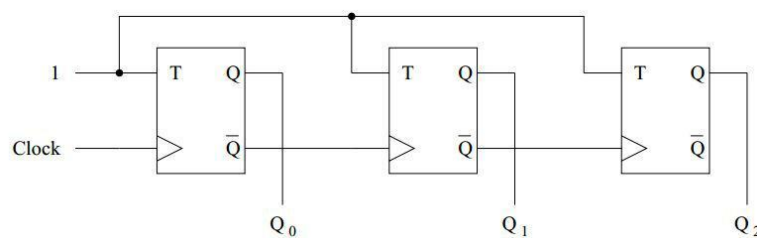


Figure 28: A 3-bit up-counter.

Note here the value of the count is the indicated by the 3-bit binary number $Q_2Q_1Q_0$. Since the second ip- op is clocked by Q_0 , the value of Q_1 changes shortly after the change of the Q_0 signal. Similarly, the value of Q_2 changes shortly after the change of the Q_1 signal. This circuit is a modulo-8 counter. Because it counts in the upward direction, we call it an up-counter. This behavior is similar to the rippling of carries in a ripple-carry adder. The circuit is therefore called an asynchronous counter, or a ripple counter.

Asynchronous Down-Counter with T Flip-Flops

Some modi cations of the circuit in Figure 4.29 lead to a down-counter which counts in the sequence 0, 7, 6, 5, 4, 3, 2, 1, 0, 7, and so on. The modi ed circuit is shown in Figure 3.

Here the clock inputs of the second and third ip- ops are driven by the Q outputs of the preceding stages, rather than by the Q outputs.

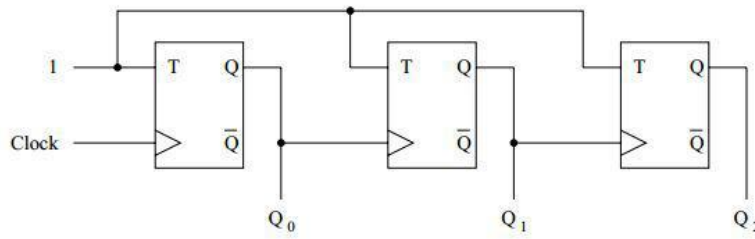


Figure 29: A 3-bit down-counter.

Although the asynchronous counter is easier to construct, it has some major disadvantages over the synchronous counter.

First of all, the asynchronous counter is slow. In a synchronous counter, all the ip- ops will change states simultaneously while for an asynchronous counter, the propagation delays of the ip- ops add together to produce the overall delay. Hence, the more bits or number of ip- ops in an asynchronous counter, the slower it will be.

Secondly, there are certain "risks" when using an asynchronous counter. In a complex system, many state changes occur on each clock edge and some ICs respond faster than others. If an external event is allowed to affect a system whenever it occurs (unsynchronized), there is a small chance that it will occur near a clock transition, after some IC's have responded, but before others have. This intermingling of transitions often causes erroneous operations. And the worse this is that these problems are difficult to foresee and test for because of the random time difference between the events.

Synchronous Counters

A synchronous counter usually consists of two parts: the memory element and the combinational element. The memory element is implemented using ip- ops while the combinational element can be implemented in a number of ways. Using logic gates is the traditional method of implementing combinational logic and has been applied for decades.

Synchronous Up-Counter with T Flip-Flops

An example of a 4-bit synchronous up-counter is shown in Figure 30. Observing the pattern of bits in each row of the table, it is apparent that bit Q0 changes on each clock cycle. Bit Q1 changes only when Q0 = 1. Bit Q2 changes only when both Q1 and Q0 are equal to 1. Bit Q3 changes only when Q2 = Q1 = Q0 = 1. In general, for an n-bit up-counter, a given ip- op changes its state only when all the preceding ip- ops are in the state Q = 1.

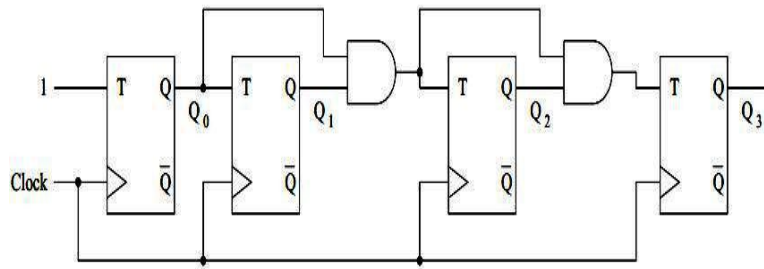


Figure 30: A 4bit synchronous upcounter

Clock Cycle	Q ₃	Q ₂	Q ₁	Q ₀
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	0	0	0	0

Figure 31: Contents of a 4bit upcounter for 16 consecutive clock cycles

Therefore, if we use T ip- ops to realize the 4-bit counter, then the T inputs should be defined as

$$T_0 = 1$$

$$T_1 = Q_0$$

$$T_2 = Q_0Q_1$$

$$T_3 = Q_0Q_1Q_2$$

In Figure 30, instead of using AND gates of increased size for each stage, we use a factored arrangement. This arrangement does not slow down the response of the counter because all inputs and outputs change their states from the propagation delay from the positive edge of the clock.

Note that a change in the value of Q0 may have to propagate through several AND gates to reach the ip- ops in the higher stages of the counter, which requires a certain amount of time. This time must not exceed the clock period. Actually, it must be 3less than the clock period minus the setup time of the ip- ops. It shows that the circuit behaves as a modulo-16 up-counter. Because all changes take place with the same delay after the active edge of the Clock signal, the circuit is called a synchronous counter.

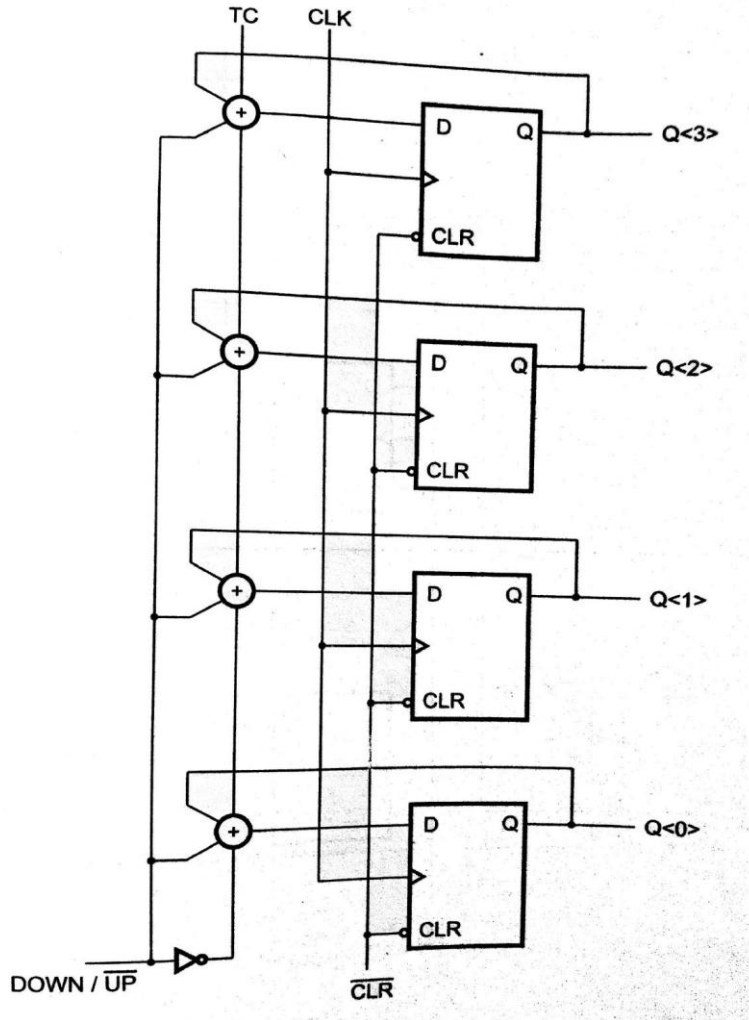


Figure 32: Design of synchronous counter using adders and registers

ARRAY SUBSYSTEMS

SRAM:

There are many reasons to use an SRAM or a DRAM in a system design. Design tradeoffs include density, speed, volatility, cost, and features. All of these factors should be considered before you select a RAM for your system design. Speed. The primary advantage of an SRAM over a DRAM is its speed. The fastest DRAMs on the market still require five to ten processor clock cycles to access the first bit of data.

Although features such as EDO and Fast Page Mode have improved the speed with which subsequent bits of data can be accessed, bus performance and other limitations mean the processor must wait for data coming from DRAM. Fast, synchronous SRAMs can operate at processor speeds of 250 MHz and beyond, with access and cycle times equal to the clock cycle used by the microprocessor. With a well designed cache using ultra-fast SRAMs, conditions in which the processor has to wait for a DRAM access become rare.

• Density:

Because of the way DRAM and SRAM memory cells are designed, readily available DRAMs have significantly higher densities than the largest SRAMs. Thus, when 64 Mb DRAMs are rolling off the production lines, the largest SRAMs are expected to be only 16 Mb.

• Volatility:

While SRAM memory cells require more space on the silicon chip, they have other advantages that translate directly into improved performance. Unlike DRAMs, SRAM cells do not need to be refreshed. This means they are available for reading and writing data 100% of the time.

• Cost:

If cost is the primary factor in a memory design, then DRAMs win hands down. If, on the other hand, performance is a critical factor, then a well-designed SRAM is an effective cost performance solution.

• Custom features:

Most DRAMs come in only one or two flavors. This keeps the cost down, but doesn't help when you need a particular kind of addressing sequence, or some other custom feature. IBM's SRAMs are tailored, via metal and substrate, for the processor or application that will be using them. Features are connected or disconnected according to the requirements of the user. Likewise, interface levels are selected to match the processor levels. IBM provides

processor specific solutions by producing a chip with a standard core design, plus metal mask options to define feature sets.

BASIC ARCHITECTURE:

The basic architecture of a static RAM includes one or more rectangular arrays of memory cells with support circuitry to decode addresses, and implement the required read and write operations. Additional support circuitry used to implement special features, such as burst operation, may also be present on the chip. Figure 1 shows a basic block diagram of a synchronous SRAM. As you read, you may wish to refer to the diagram to help you visualize how the SRAM works.

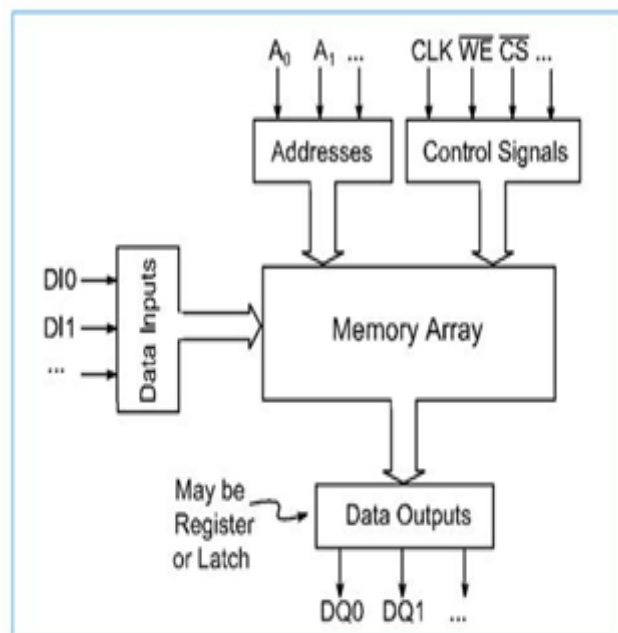


Fig:33 Block diagram of synchronous SRAM

Memory Arrays:

SRAM memory arrays are arranged in rows and columns of memory cells called wordlines and bitlines, respectively. In IBM SRAMs, the wordlines are made from polysilicon while the bitlines are metal. Each memory cell has a unique location or address defined by the intersection of a row and column. Each address is linked to a particular data input/output pin. The number of arrays on a memory chip is determined by the total size of the memory, the speed at which the memory must operate, layout and testing requirements, and the number of data I/Os on the chip.

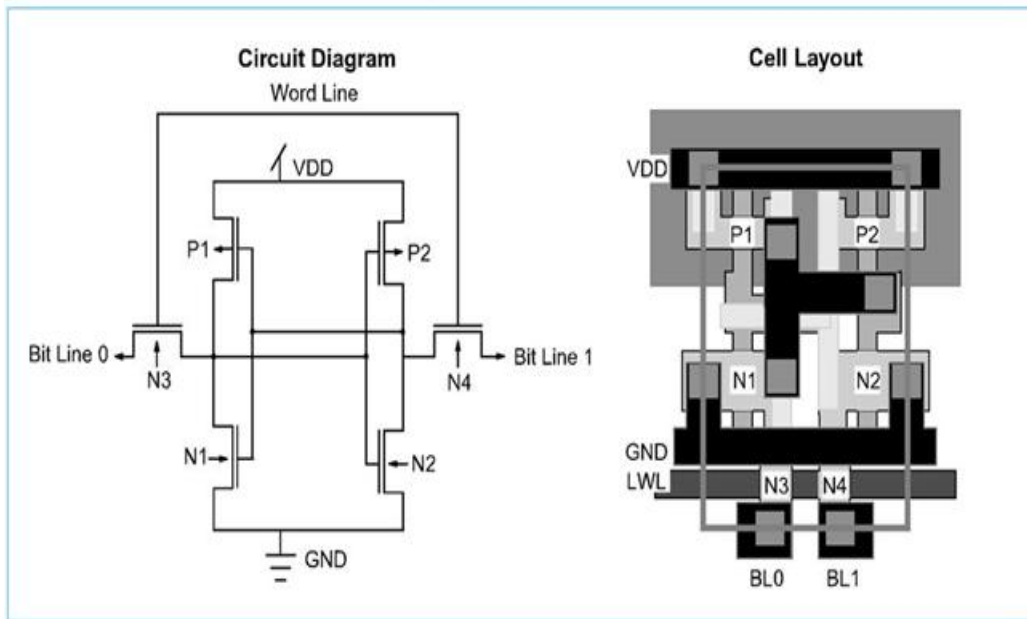


Fig:34 Six Transistor SRAM Memory Cell

Memory Cell:

An SRAM memory cell is a bi-stable flip-flop made up of four to six transistors. The flip-flop may be in either of two states that can be interpreted by the support circuitry to be a 1 or a 0. Many of the SRAMs on the market use a four transistor cell with a polysilicon load. Suitable for medium to high performance, this design has a relatively high leakage current, and consequently high standby current. Four transistor designs may also be more susceptible to various types of radiation induced soft errors. SRAMs all use a six transistor memory cell (also called a six-device cell) that is highly stable, relatively impervious to soft errors, and has low leakage and standby currents.

Reading Data from Memory:

Figure 3 shows the timing diagram for the simplified read operation for a flow thru part. It is used to illustrate the following example. To read data from a memory cell, the cell must be selected using its row and column coordinates, the state of the cell must be determined, and the information must be sent to the data output. In terms of timing, the following steps must occur:

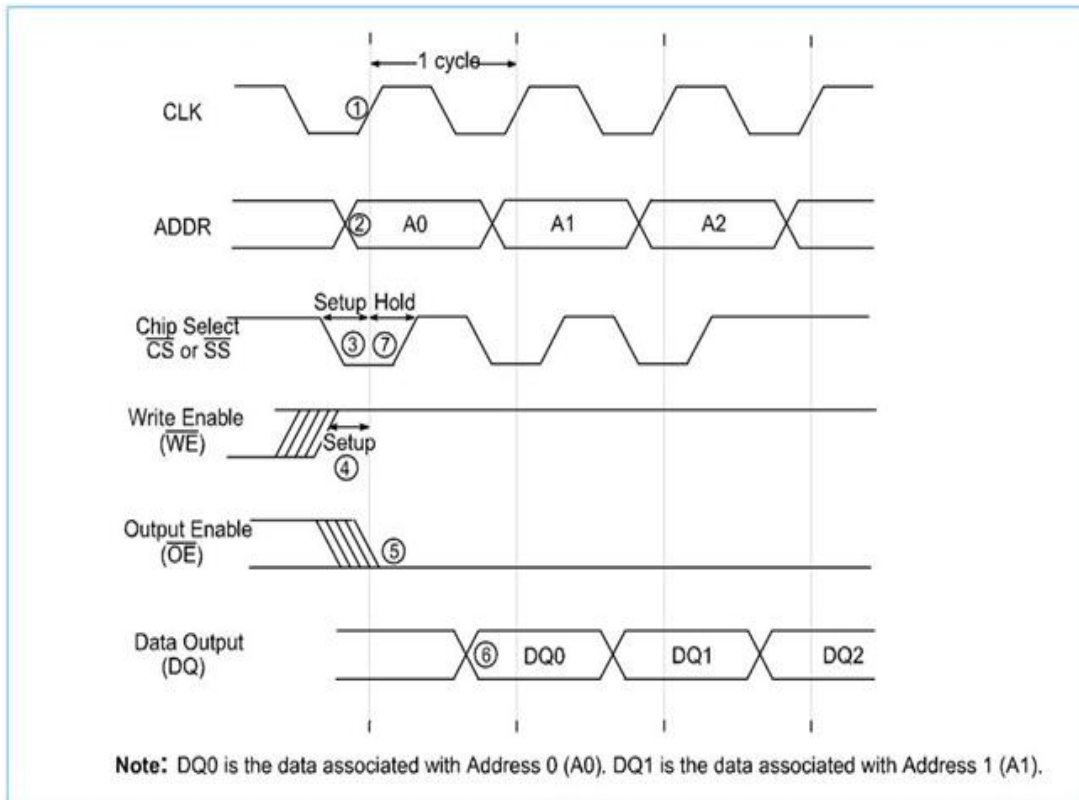


Fig:35 Reading from SRAM Memory Cell

1. Before the clock transition (low to high) that initiates the read operation (1), the row and column addresses must be applied to the address input pins (ADDR) (2), the chip must be selected (3), and the Write Enable must be high (4). Note that each of these signals must be present and valid a specified amount of time (the set up time) before the clock switches from low to high, and must remain valid for a specified amount of time (the hold time) after the clock switches (7). When the chip select (CS) is low, the chip is selected. When it is high (inactive), the chip cannot accept any input signals. The Write Enable is used to choose between reading and writing. When it is low, a write operation occurs; when it is high, a read operation occurs.
2. On the rising edge of the clock (CLK) (1), the address is registered and the read cycle begins.
3. If the Output Enable is being used to control the appearance of data at the output, OE must go low (5). OE is an asynchronous signal; it can be activated at any time. When OE is high, the DQs are tri-stated; data from the memory will not appear on the outputs.
4. Data appears at the output pins of the SRAM (6). The time at which the data appears depends on the access time of the device, the delay associated with the Output Enable and the type of SRAM you are using. The access time of the SRAM is the amount of

time required to read a bit of data from the memory when all of the timing requirements have been met.

Writing Data to Memory:

Figure 4 shows a simplified timing diagram for the write operation used in the following example. This example uses a standard write, flow thru SRAM.

To write data to a memory cell, the cell must be selected using its row and column coordinates, the data to be stored must be applied at the data input pins, and the information must be stored in the selected memory cell. In terms of timing, the following steps must occur:

1. Before the clock transition (low to high) that initiates the write operation (1), the row and column addresses must be applied to the address input pins (ADDR) (2), the chip must be selected (3), the Write Enable must be low (4) and the data to be written must be applied to the data input pins (5). If the SRAM has Byte Write Enables, they must be low as well. Note that each of these signals must be present and valid a specified amount of time (the set up time) before the clock switches from low to high, and must remain valid for a specified amount of time (the hold time) after the clock switches. When the chip select (CS) is low, the chip is selected. When it is high (inactive), the chip cannot accept any input signals. The Write Enable is used to choose between reading and writing. When it is low, a write operation occurs; when it is high, a read operation occurs. When the Byte Write Enables are high, no data may be written to the memory. When they are low, data may be written to the associated data inputs.
2. On the rising edge of the clock (CLK) (1), the address and input data are latched and the write operation begins. The data is stored in the selected memory cell.

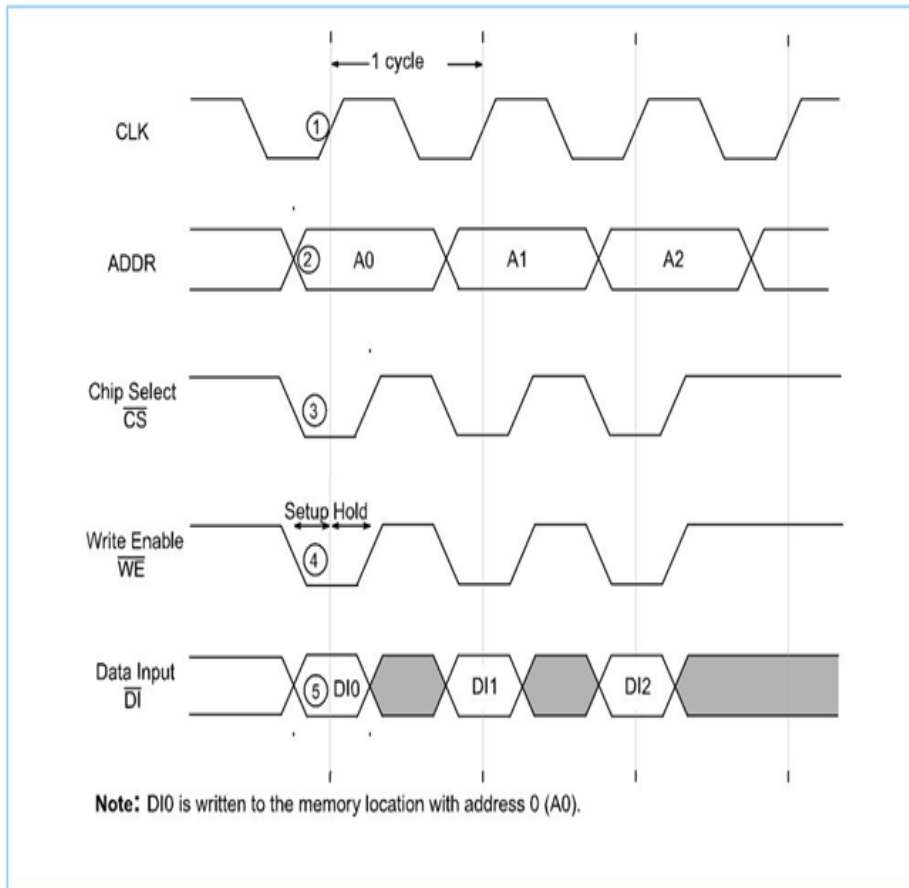


Fig:36 Writing to SRAM Memory Cell

DRAM:

Dynamic Random Access Memory (DRAM) devices are used in a wide range of electronics applications. Although they are produced in many sizes and sold in a variety of packages, their overall operation is essentially the same. DRAMs are designed for the sole purpose of storing data. The only valid operations on a memory device are reading the data stored in the device, writing (or storing) data in the device, and refreshing the data periodically. To improve efficiency and speed, a number of methods for reading and writing the memory have been developed.. While many aspects of a synchronous DRAM are similar to an asynchronous DRAM, synchronous operation differs because it uses a clocked interface and multiple bank architecture.

DRAM Architecture:

DRAM chips are large, rectangular arrays of memory cells with support logic that is used for reading and writing data in the arrays, and refresh circuitry to maintain the integrity of stored data.

Memory Arrays:

Memory arrays are arranged in rows and columns of memory cells called wordlines and bitlines, respectively. Each memory cell has a unique location or address defined by the intersection of a row and a column.

Memory Cells:

A DRAM memory cell is a capacitor that is charged to produce a 1 or a 0. Over the years, several different structures have been used to create the memory cells on a chip. In today's technologies, trenches filled with dielectric material are used to create the capacitive storage element of the memory cell.

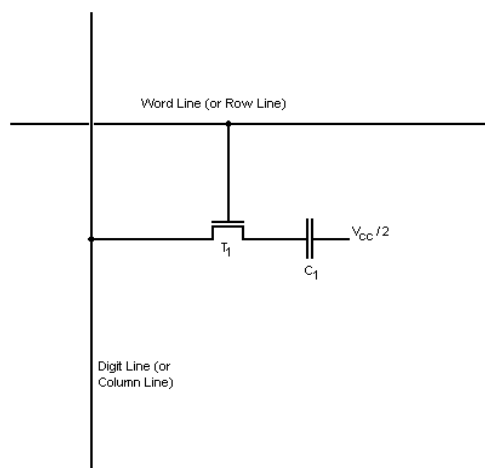


Fig:37 DRAM Memory Cell

Reading Data From Memory:

Figure 5 is the timing diagram of a simplified Read cycle that illustrates the following description. To read the data from a memory cell, the cell must be selected by its row and column coordinates, the charge on the cell must be sensed, amplified, and sent to the support circuitry, and the data must be sent to the data output. In terms of timing, the following steps must occur:

1. The row address must be applied to the address input pins on the memory device for the prescribed amount of time before RAS goes low (t_{ASR}) and held (t_{RAH}) after RAS goes low.
2. RAS must go from high to low and remain low (t_{RAS}).

3. A column address must be applied to the address input pins on the memory device for the prescribed amount of time (t_{ASC}) and held (t_{CAH}) after CAS goes low.

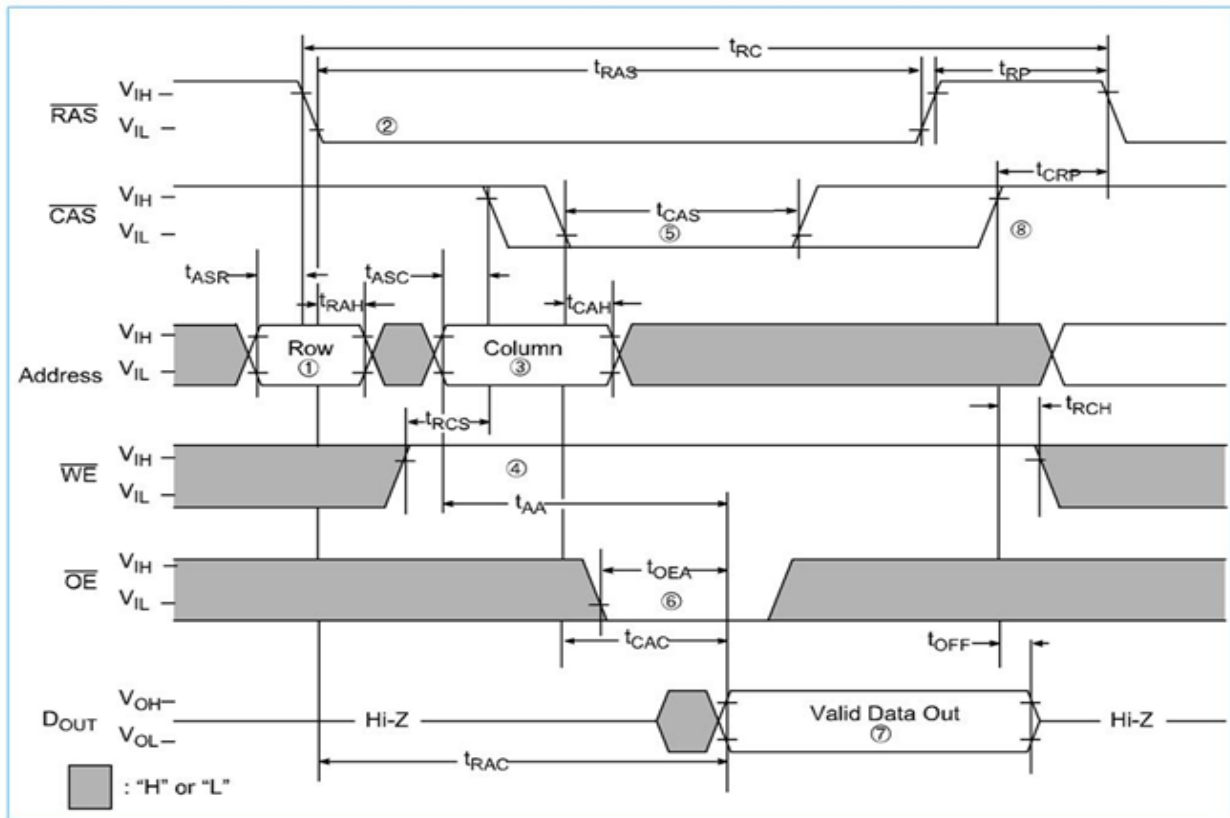


Fig: 38 Reading from DRAM Memory Cell

4. \overline{WE} must be set high for a read operation to occur prior (t_{RCS}) to the transition of \overline{CAS} , and remain high (t_{RCH}) after the transition of \overline{CAS} .

5. \overline{CAS} must switch from high to low and remain low (t_{CAS}).

6. \overline{OE} goes low within the prescribed window of time. Cycling \overline{OE} is optional; it may be tied low, if desired.

7. Data appears at the data output pins of the memory device. The time at which the data appears depends on when \overline{RAS} (t_{RAC}), \overline{CAS} (t_{CAC}), and \overline{OE} (t_{OEA}) went low, and when the address is supplied (t_{AA}).

8. Before the read cycle can be considered complete, \overline{CAS} and \overline{RAS} must return to their inactive states (t_{CRP} , t_{RP}).

Writing Data To Memory :

Figure 3 is the timing diagram of a simplified Write cycle that illustrates described below. To write to a memory cell, the row and column address for the cell must be selected and data must be presented at the data input pins. The chip's onboard logic either charges the memory cell's capacitor or discharges it, depending on whether a 1 or 0 is to be stored. In terms of timing, the following steps must occur:

1. The row address must be applied to the address input pins on the memory device for the prescribed amount of time before RAS goes low and be held for a period of time.
2. RAS must go from high to low.
3. A column address must be applied to the address input pins on the memory device for the prescribed amount of time after RAS goes low and before CAS goes low and held for the prescribed time
4. WE must be set low for a certain time for a write operation to occur (t_{WP}). The timing of the transitions are determined by CAS going low (t_{WCS} , t_{WCH}).
5. Data must be applied to the data input pins the prescribed amount of time before CAS goes low (t_{DS}) and held (t_{DH}).
6. CAS must switch from high to low.
7. Before the write cycle can be considered complete, CAS and RAS must return to their inactive states. Note: There is considerable latitude within the memory chip's timings with respect to OE when data is actually written. The memory specifications show how to set up chip timings for early and delayed write options.

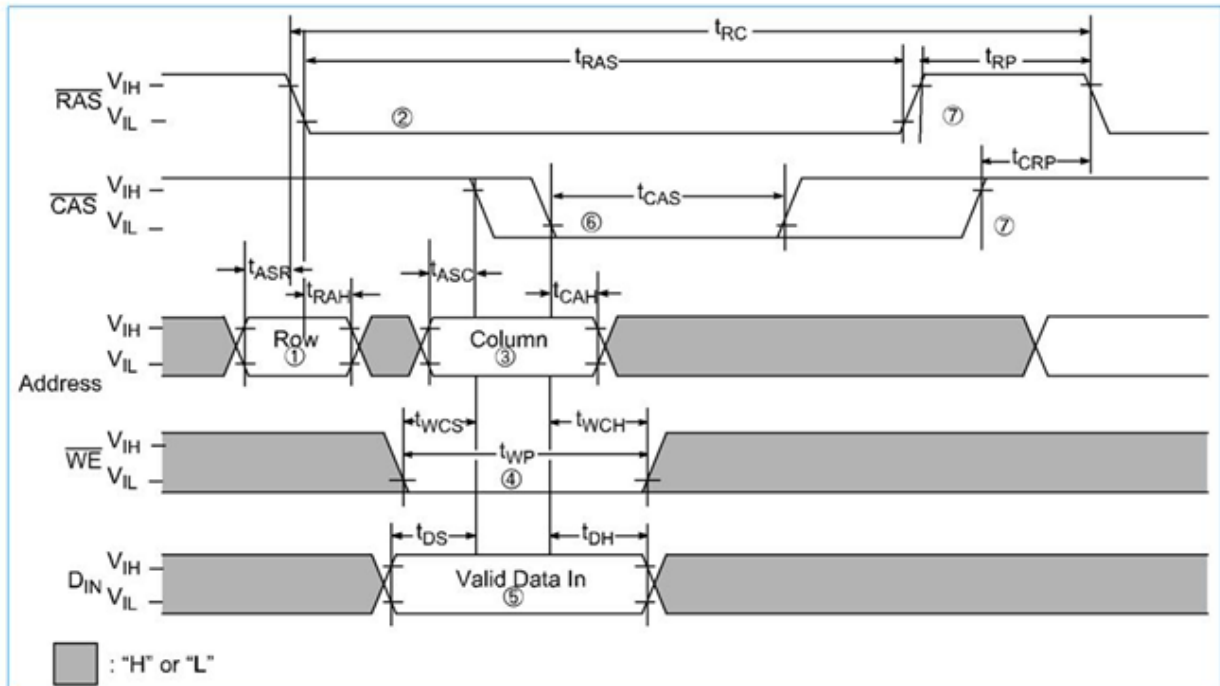


Fig: 39 Writing to DRAM Memory Cell

ROM (Read Only Memory)

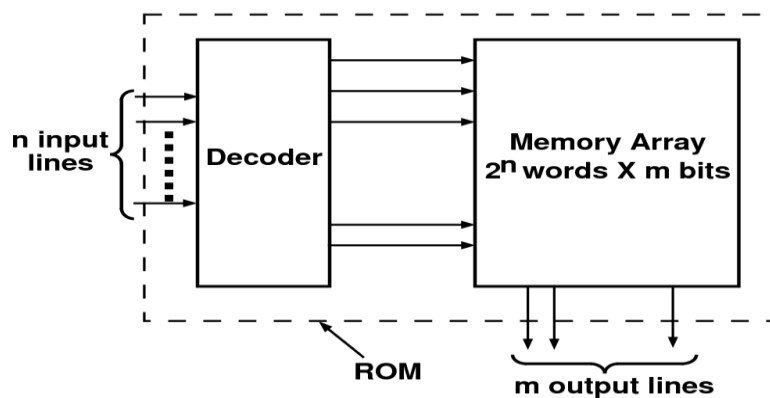


Fig: 40 ROM Architecture

- Read-only memory (usually known by its acronym, ROM) is a class of storage media used in computers and other electronic devices.
- Because data stored in ROM cannot be modified (at least not very quickly or easily), it is mainly used to distribute firmware
- Firmware is software that is very closely tied to specific hardware, and unlikely to require frequent updates.

- In its strictest sense, ROM refers only to mask ROM (the oldest type of solid state ROM), which is fabricated with the desired data permanently stored in it, and thus can never be modified.
- However, more modern types such as EPROM and flash EEPROM can be erased and re-programmed multiple times; they are still described as "read-only memory" because the reprogramming process is generally infrequent, comparatively slow, and often does not permit random access writes to individual memory locations.
- Despite the simplicity of mask ROM, economies of scale and field-programmability often make reprogrammable technologies more flexible and inexpensive, so that mask ROM is rarely used in new products as of 2007.
- Classic mask-programmed ROM chips are integrated circuits that physically encode the data to be stored, and thus it is impossible to change their contents after fabrication. Other types of non-volatile solid-state memory permit some degree of modification: Types of ROM Programmable read-only memory (PROM), or onetime programmable ROM (OTP), can be written to or programmed via a special device called a PROM programmer. Typically, this device uses high voltages to permanently destroy or create internal links (fuses or antifuses) within the chip. Consequently, a PROM can only be programmed once.
- Erasable programmable read-only memory (EPROM) can be erased by exposure to strong ultraviolet light (typically for 10 minutes or longer), then rewritten with a process that again requires application of higher than usual voltage. Repeated exposure to UV light will eventually wear out an EPROM, but the endurance of most EPROM chips exceeds 1000 cycles of erasing and reprogramming. EPROM chip packages can often be identified by the prominent quartz "window" which allows UV light to enter. After programming, the window is typically covered with a label to prevent accidental erasure.
- Electrically erasable programmable read-only memory (EEPROM) is based on a similar semiconductor structure to EPROM, but allows its entire contents (or selected banks) to be electrically erased, then rewritten electrically, so that they need not be removed from the computer (or camera, MP3 player, etc.). Writing or flashing an EEPROM is much slower (milliseconds per bit) than reading from a ROM or writing to a RAM (nanoseconds in both cases).

- Electrically alterable read-only memory (EAROM) is a type of EEPROM that can be modified one bit at a time. Writing is a very slow process and again requires higher voltage (usually around 12 V) than is used for read access. EAROMs are intended for applications that require infrequent and only partial rewriting. EAROM may be used as non-volatile storage for critical system setup information; in many applications, EAROM has been supplanted by CMOS RAM supplied by mains power and backed-up with a lithium battery.
- Flash memory (or simply flash) is a modern type of EEPROM invented in 1984. Flash memory can be erased and rewritten faster than ordinary EEPROM, and newer designs feature very high endurance (exceeding 1,000,000 cycles). Modern NAND flash makes efficient use of silicon chip area, resulting in individual ICs with a capacity as high as 16 GB as of 2007[update]; this feature, along with its endurance and physical durability, has allowed NAND flash to replace magnetic in some applications (such as USB flash drives). Flash memory is sometimes called flash ROM or flash EEPROM when used as a replacement for older ROM types, but not in applications that take advantage of its ability to be modified quickly

SERIAL ACCESS MEMORIES:

Serial access memories do not use an address

- Shift Registers
- Tapped Delay Lines
- Serial In Parallel Out (SIPO)
- Parallel In Serial Out (PISO)
- Queues (FIFO, LIFO)

SHIFT REGISTER:

- Shift registers store and delay data
- Simple design: cascade of registers

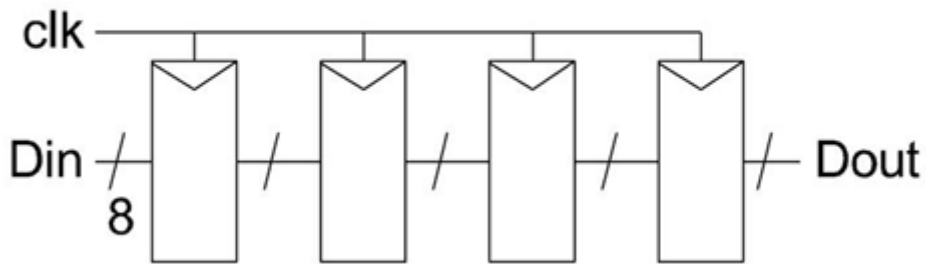


Fig: 41 Shift Register

Denser Shift Registers :

- Flip-flops aren't very area-efficient
- For large shift registers, keep data in SRAM instead
- Move read/write pointers to RAM rather than data
 - I. Initialize read address to first entry, write to last
 - II. Increment address on each cycle

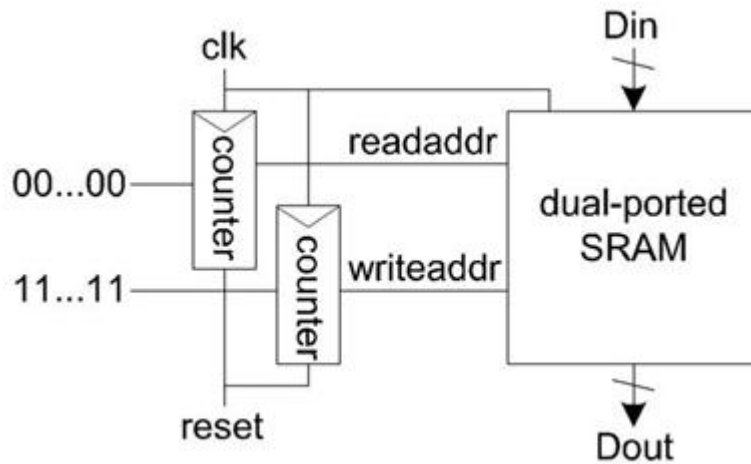


Fig: 42 Denser Shift Register

QUEUES :

- Queues allow data to be read and written at different rates.
- Read and write each use their own clock, data

- Queue indicates whether it is full or empty
- Build with SRAM and read/write counters (pointers)

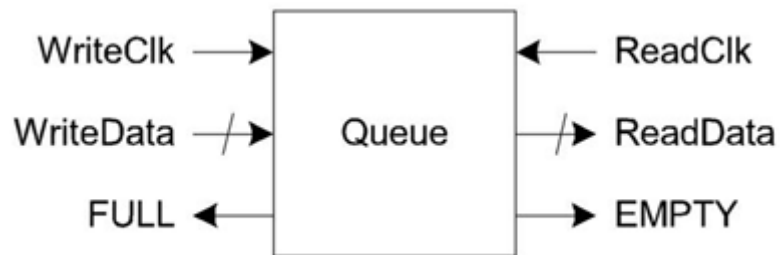


Fig: 43 Queue

FIFO, LIFO Queues:

First In First Out (FIFO) :

- Initialize read and write pointers to first element
- Queue is EMPTY " On write, increment write pointer
- If write almost catches read, Queue is FULL
- On read, increment read pointer

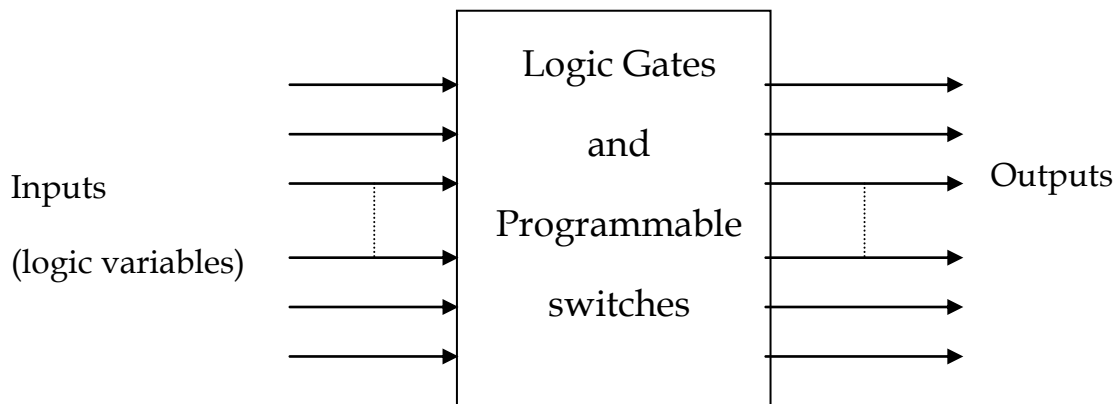
Last In First Out (LIFO):

- Also called a stack

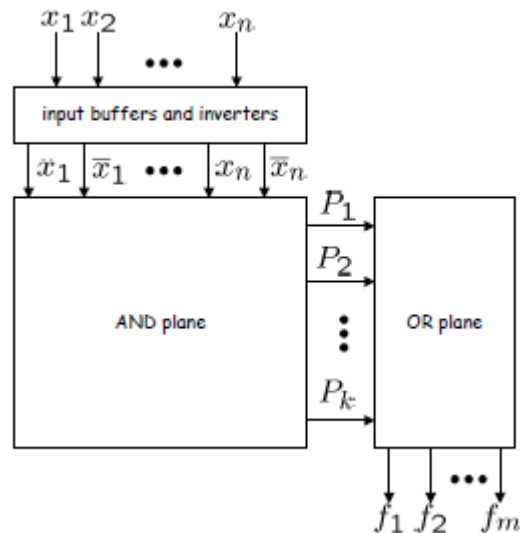
Use a single stack pointer for read and

UNIT-5 PART-I PROGRAMMABLE LOGIC DEVICES

Programmable Logic Device (PLD) — a general term that refers to any type of integrated circuit used for implementing digital hardware, where the chip can be configured by the end user to realize different designs. Programming of such a device often involves placing the chip into a special programming unit, but some chips can also be configured “in-system”. Another name for FPDs is *programmable logic devices* (PLDs); although PLDs encompass the same types of chips as FPDs, we prefer the term FPD because historically the word PLD has referred to relatively simple types of devices.



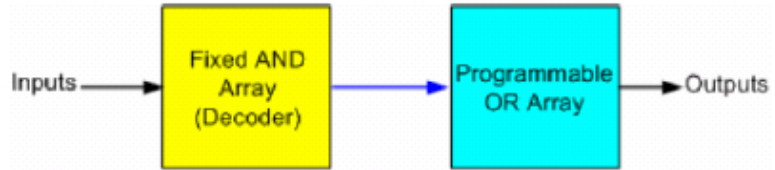
Programmable Logic Device as a black box



The fundamental 3 types of SPLDs:

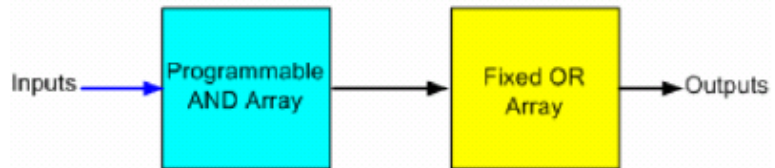


PROM - as a Memory addressed by dates



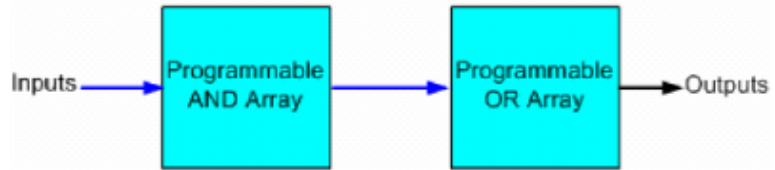
(a) Programmable Read Only Memory (PROM)

Classical PAL devices



(b) Programmable Array Logic (PAL) Device

Classical PLA devices

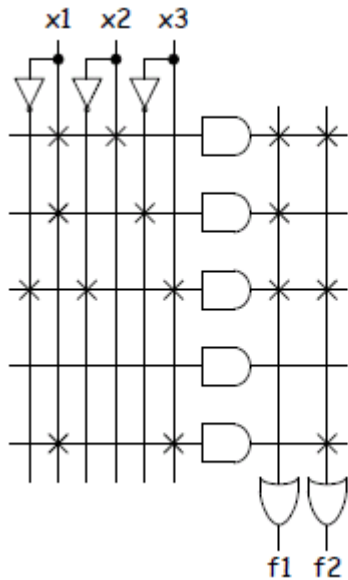


(c) Programmable Logic Array (PLA) Device

Programmable Connections
 Normal Connections



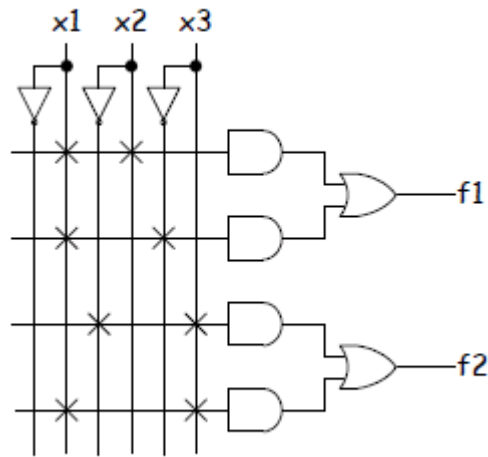
- **PLA** — a Programmable Logic Array (PLA) is a relatively small FPD that contains two levels of logic, an AND-plane and an OR-plane, where both levels are programmable (note: although PLA structures are sometimes embedded into full-custom chips, we refer here only to those PLAs that are provided as separate integrated circuits and are user-programmable).



$$f_1 = x_1x_2 + x_1\bar{x}_3 + \bar{x}_1\bar{x}_2x_3$$

$$f_2 = x_1x_2 + \bar{x}_1\bar{x}_2x_3 + x_1x_3$$

- **PAL*** — a Programmable Array Logic (PAL) is a relatively small FPD that has a programmable AND-plane followed by a fixed OR-plane.



$$f_1 = x_1x_2 + x_1\bar{x}_3$$

$$f_2 = \bar{x}_2x_3 + x_1x_3$$

Field Programmable Gate Arrays

Description

FPGA consists three major modules:

1. Configurable logic block
2. IO block
3. Routing resources.

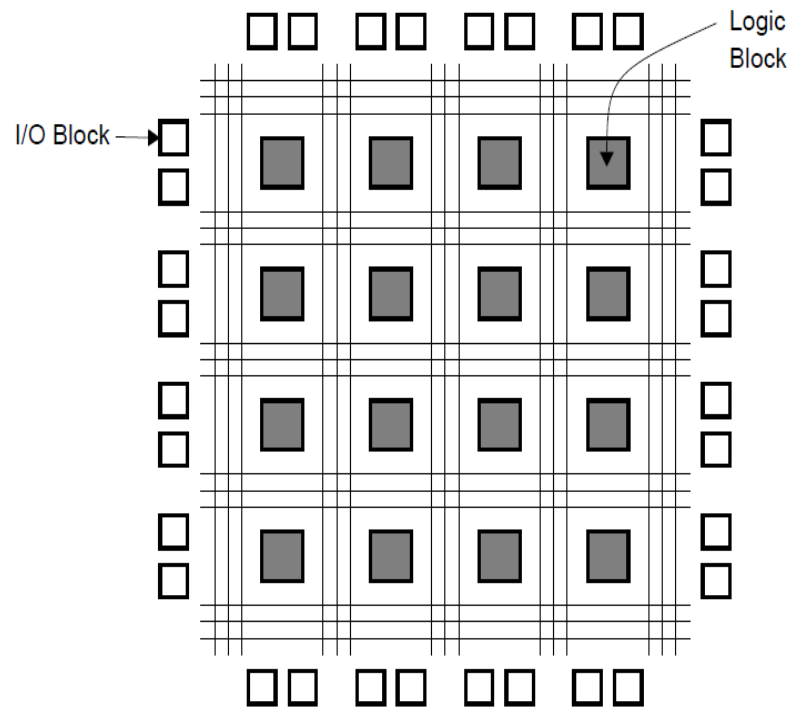


Figure 1 - Structure of an FPGA.

Detailed Functional Description

Basic Building Blocks

Xilinx user-programmable gate arrays include two major configurable elements: configurable logic blocks (CLBs) and input/output blocks (IOBs).

- CLBs provide the functional elements for constructing the user's logic.
- IOBs provide the interface between the package pins and internal signal lines.

Three other types of circuits are also available:

- 3-State buffers (TBUFs) driving horizontal long lines are associated with each CLB.
- Wide edge decoders are available around the periphery of each device.
- An on-chip oscillator is provided.

Programmable interconnect resources provide routing paths to connect the inputs and outputs of these configurable elements to the appropriate networks.

Configurable Logic Blocks (CLBs)

Configurable Logic Blocks implement most of the logic in an FPGA. Two 4-input function generators (F and G) offer unrestricted versatility. Most combinatorial logic functions need four or fewer inputs. A third function generator (H) has three inputs. Either zero, one, or two of these inputs can be the outputs of F and G; the other input(s) are from outside the CLB.

The CLB can, therefore, implement certain functions of up to nine variables, like parity check or expandable-identity comparison of two sets of four inputs.

Each CLB contains two storage elements that can be used to store the function generator outputs.

Thirteen CLB inputs and four CLB outputs provide access to the function generators and storage elements. These inputs and outputs connect to the programmable interconnect resources outside the block.

Function Generators

Four independent inputs are provided to each of two function generators (F1 - F4 and G1 - G4). These function generators, with outputs labeled F' and G', are each capable of implementing any arbitrarily defined Boolean function of four inputs. The function generators are implemented as memory look-up tables.

A third function generator, labeled H', can implement any Boolean function of its three inputs. Two of these inputs can optionally be the F' and G' functional generator outputs.

Alternatively, one or both of these inputs can come from outside the CLB (H2, H0). The third input must come from outside the block (H1).

Signals from the function generators can exit the CLB on two outputs. F' or H' can be connected to the X output. G' or H' can be connected to the Y output.

A CLB can be used to implement any of the following functions:

- Any function of up to four variables, plus any second function of up to four unrelated variables, plus any third function of up to three unrelated variables
- Any single function of five variables
- Any function of four variables together with some functions of six variables
- Some functions of up to nine variables.

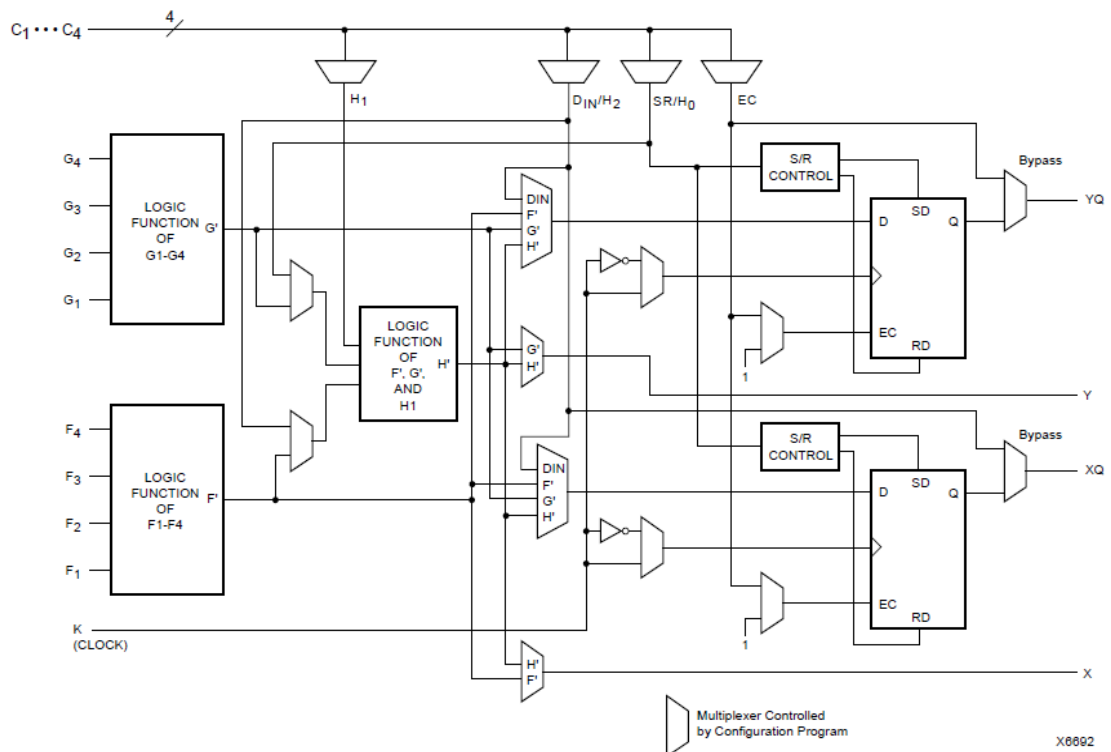


Figure 2: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown) Flip-Flops

The CLB can pass the combinational output(s) to the interconnect network, but can also store the combinational results or other incoming data in one or two flip-flops, and connect their outputs to the interconnect network as well.

Fast Carry Logic

Each CLB F and G function generator contains dedicated arithmetic logic for the fast generation of carry and borrow signals. This extra output is passed on to the function generator in the adjacent CLB. The carry chain is independent of normal routing resources. Dedicated fast

carry logic greatly increases the efficiency and performance of adders, subtractors, accumulators, comparators and counters.

The carry chain in XC4000E devices can run either up or down. At the top and bottom of the columns where there are no CLBs above or below, the carry is propagated to the right.

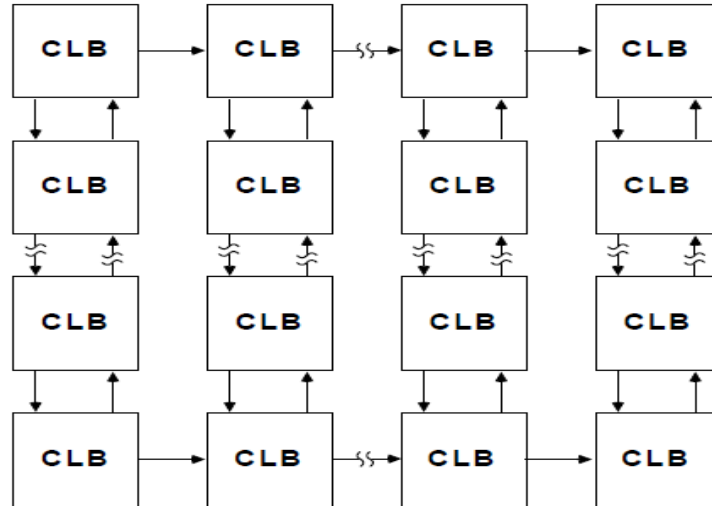


Figure 3: Available XC4000E Carry Propagation Paths

In order to improve speed in the high-capacity XC4000X devices, which can potentially have very long carry chains, the carry chain travels upward only. Additionally, standard interconnect can be used to route a carry signal in the downward direction.

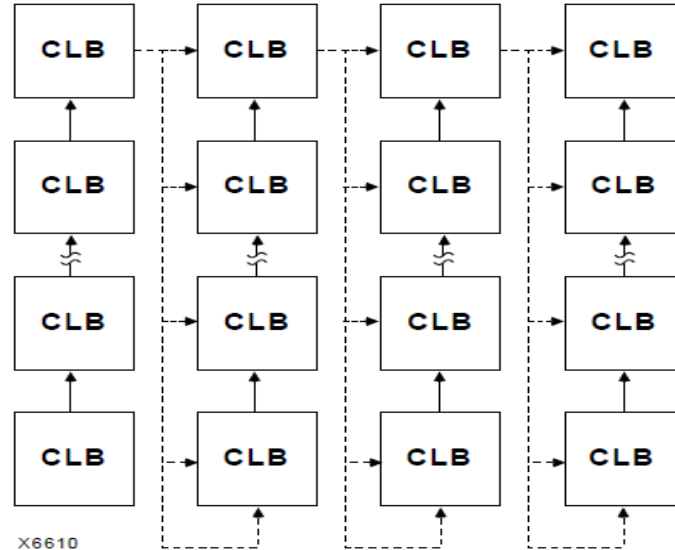


Figure 4: Available XC4000X Carry Propagation Paths (dotted lines use general interconnect)

Input/Output Blocks (IOBs)

User-configurable input/output blocks (IOBs) provide the interface between external package pins and the internal logic. Each IOB controls one package pin and can be configured for input, output, or bidirectional signals.

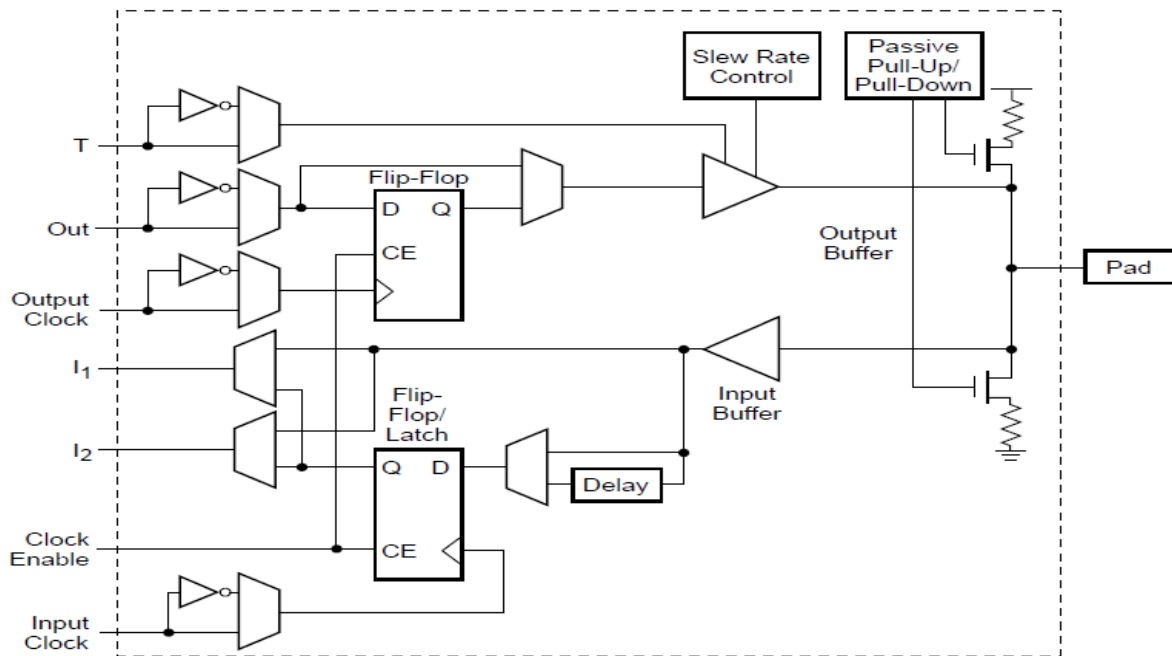


Figure 5: Simplified Block Diagram of XC4000E IOB

Programmable Interconnect

All internal connections are composed of metal segments with programmable switching points and switching matrices to implement the desired routing. A structured, hierarchical matrix of routing resources is provided to achieve efficient automated routing.

Interconnect Overview

There are several types of interconnect.

- CLB routing is associated with each row and column of the CLB array.
- IOB routing forms a ring (called a VersaRing) around the outside of the CLB array. It connects the I/O with the internal logic blocks.
- Global routing consists of dedicated networks primarily designed to distribute clocks throughout the device with minimum delay and skew. Global routing can also be used for other high-fanout signals.

Five interconnect types are distinguished by the relative length of their segments: single-length lines, double-length lines, quad and octal lines (XC4000X only), and longlines. In the XC4000X, direct connects allow fast data flow between adjacent CLBs, and between IOBs and CLBs. Extra routing is included in the IOB pad ring. The XC4000X also includes a ring of octal interconnect lines near the IOBs to improve pin-swapping and routing to locked pins.

CLB Routing Connections

A high-level diagram of the routing resources associated with one CLB is shown in Figure. The shaded arrows represent routing present only in XC4000X devices.

CLB inputs and outputs are distributed on all four sides, providing maximum routing flexibility. In general, the entire architecture is symmetrical and regular. It is well suited to established placement and routing algorithms. Inputs, outputs, and function generators can freely swap positions within a CLB to avoid routing congestion during the placement and routing operation.

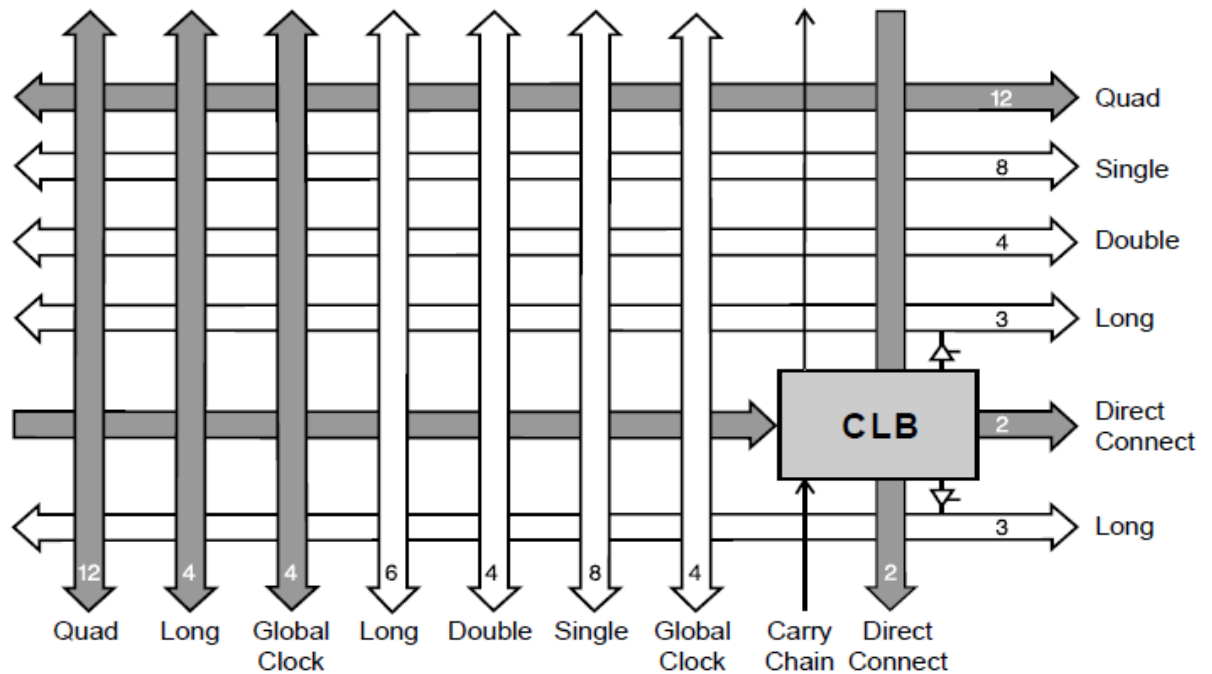


Figure 6: High-Level Routing Diagram of XC4000 Series CLB (shaded arrows indicate XC4000X only)

Programmable Switch Matrices

The horizontal and vertical single- and double-length lines intersect at a box called a programmable switch matrix (PSM). Each switch matrix consists of programmable pass transistors used to establish connections between the lines.

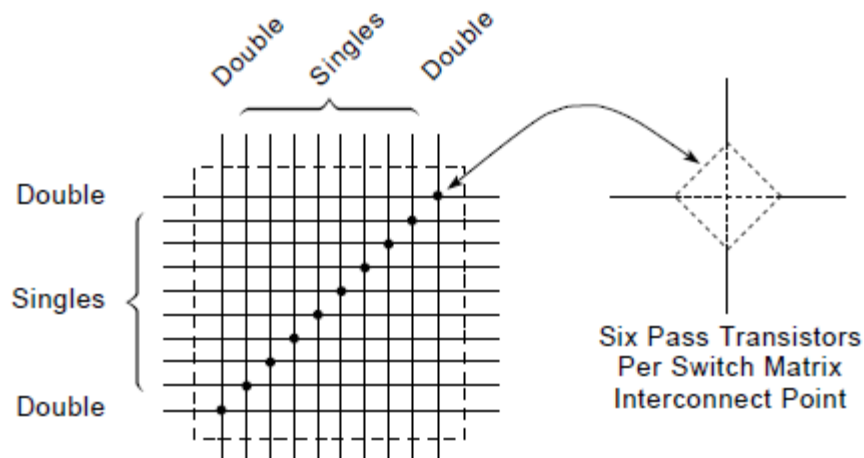


Figure 7: Programmable Switch Matrix (PSM)

Single-Length Lines

Single-length lines provide the greatest interconnect flexibility and offer fast routing between adjacent blocks. There are eight vertical and eight horizontal single-length lines associated with each CLB. These lines connect the switching matrices that are located in every row and a column of CLBs.

Single-length lines incur a delay whenever they go through a switching matrix. Therefore, they are not suitable for routing signals for long distances. They are normally used to conduct signals within a localized area and to provide the branching for nets with fanout greater than one.

Double-Length Lines

The double-length lines consist of a grid of metal segments, each twice as long as the single-length lines: they run past two CLBs before entering a switch matrix. Double-length lines are grouped in pairs with the switch matrices staggered, so that each line goes through a switch matrix at every other row or column of CLBs.

There are four vertical and four horizontal double-length lines associated with each CLB. These lines provide faster signal routing over intermediate distances, while retaining routing flexibility. Double-length lines are connected by way of the programmable switch matrices.

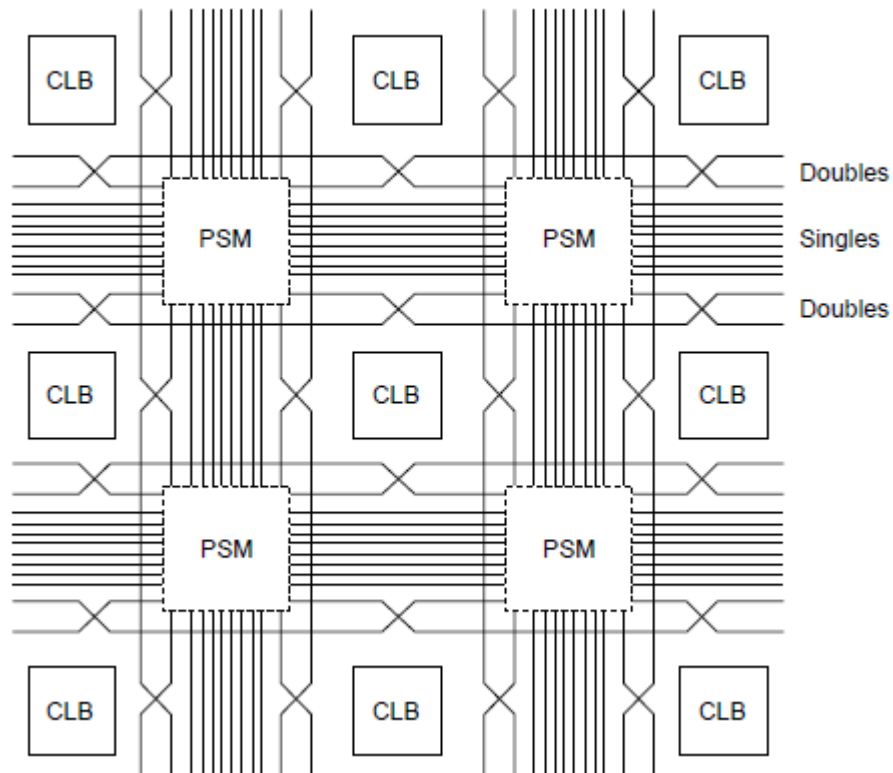


Figure 8: Single- and Double-Length Lines, with Programmable Switch Matrices (PSMs)

I/O Routing

XC4000 Series devices have additional routing around the IOB ring. This routing is called a VersaRing. The VersaRing facilitates pin-swapping and redesign without affecting board layout.

CPLD Architecture

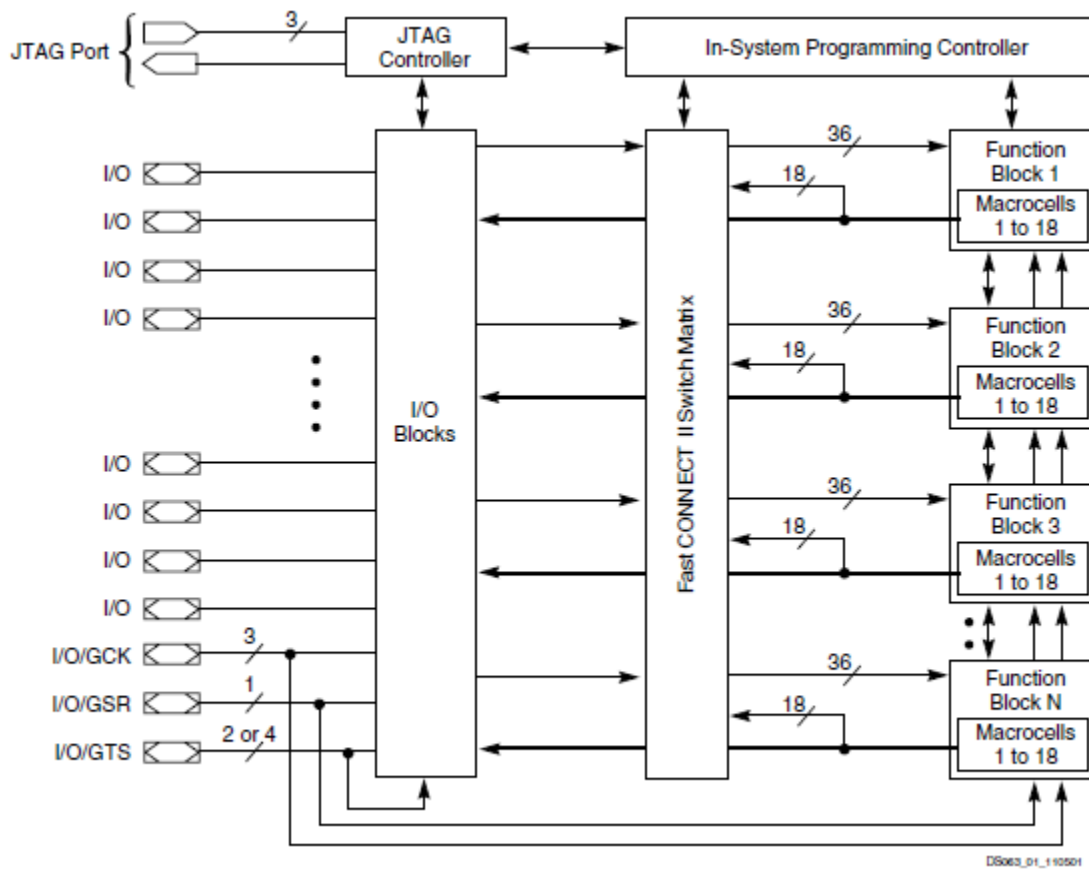
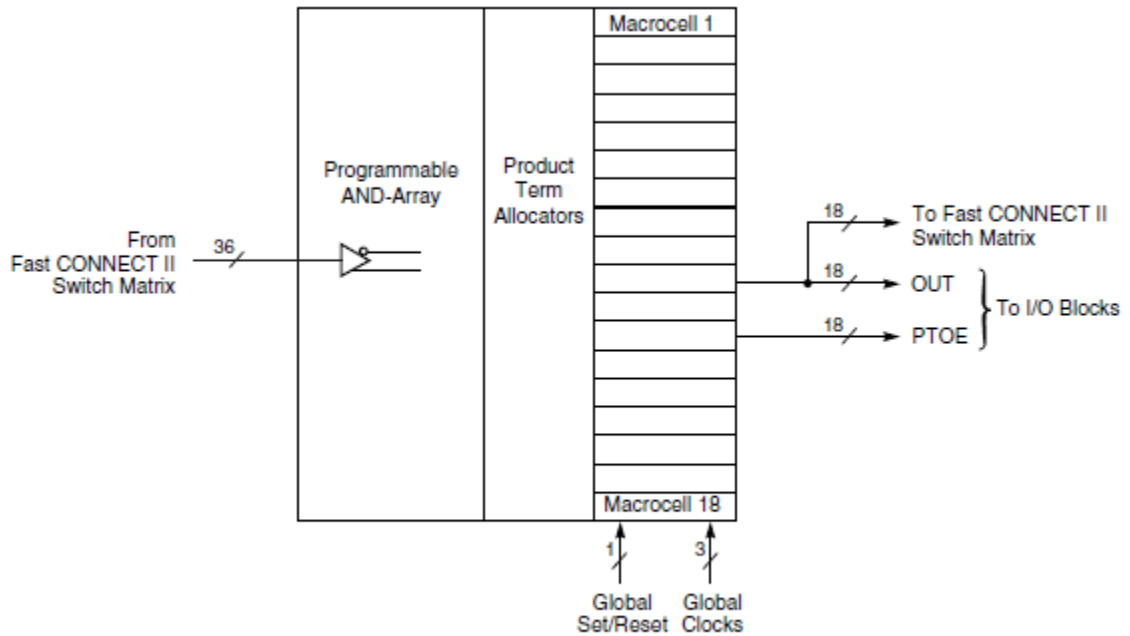


Figure 1: XC9500 Architecture

Function Block

Each Function Block is comprised of 18 independent macrocells, each capable of implementing a combinatorial or registered function. The FB also receives global clock, output enable, and set/reset signals. The FB generates 18 outputs that drive the Fast CONNECT switch matrix. These 18 outputs and their corresponding output enable signals also drive the IOB.

Logic within the FB is implemented using a sum-of-products representation. Thirty-six inputs provide 72 true and complement signals into the programmable AND-array to form 90 product terms. Any number of these product terms, up to the 90 available, can be allocated to each macrocell by the product term allocator.



DS063_02_110501

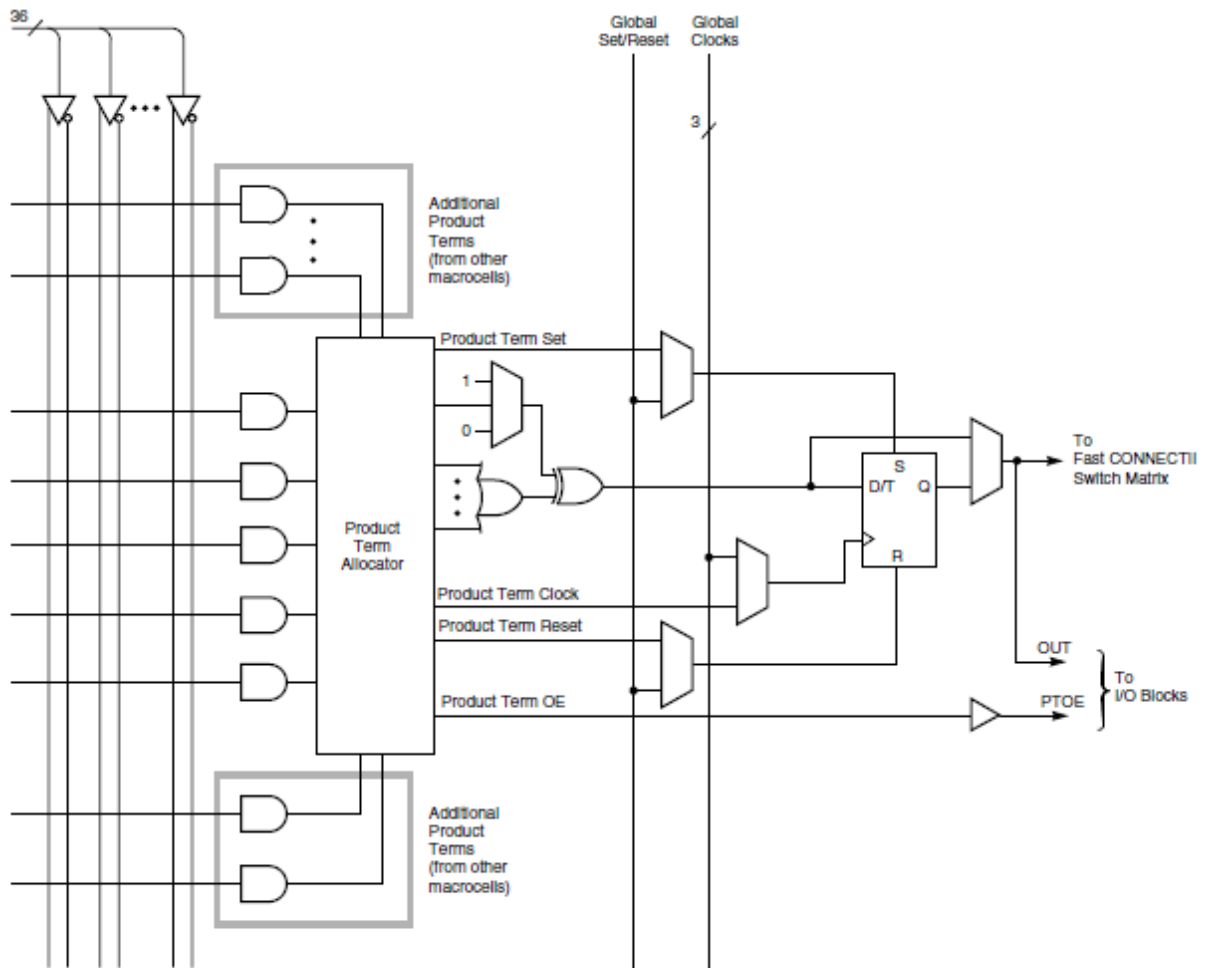
Figure 2: XC9500 Function Block

Macrocell

A macrocell array is an approach to the design and manufacture of ASICs. Essentially, it is a small step up from the otherwise similar gate array, but rather than being a prefabricated array of simple logic gates, the macrocell array is a prefabricated array of higher-level logic functions such as flip-flops, ALU functions, registers, and the like.

Each XC9500 macrocell may be individually configured for a combinatorial or registered function. Five direct product terms from the AND-array are available for each macrocell. Clock, set/reset, and output enable are control inputs. The product term allocator associated with each macrocell selects how the five direct terms are used. The macrocell register can be configured as a D-type or T-type flip-flop.

All global control signals are available to each individual macrocell, including clock, set/reset, and output enable signals. The macrocell register clock originates either from global clocks or a product term clock.



DS063_03_110501

Figure 3: XC9500 Macrocell Within Function Block

Product Term Allocator

The product term allocator controls how the five direct product terms are assigned to each macrocell.

Five direct terms drives OR function

The delay increases if only the product terms are in other macrocells. The timing of the direct product terms is not changed.

The product term allocator can re-assign other product terms within the FB to increase the logic capacity of a macrocell beyond five direct terms. Any macrocell requiring additional product terms can access uncommitted product terms in other macrocells within the FB. Up to 15 product terms can be available to a single macrocell.

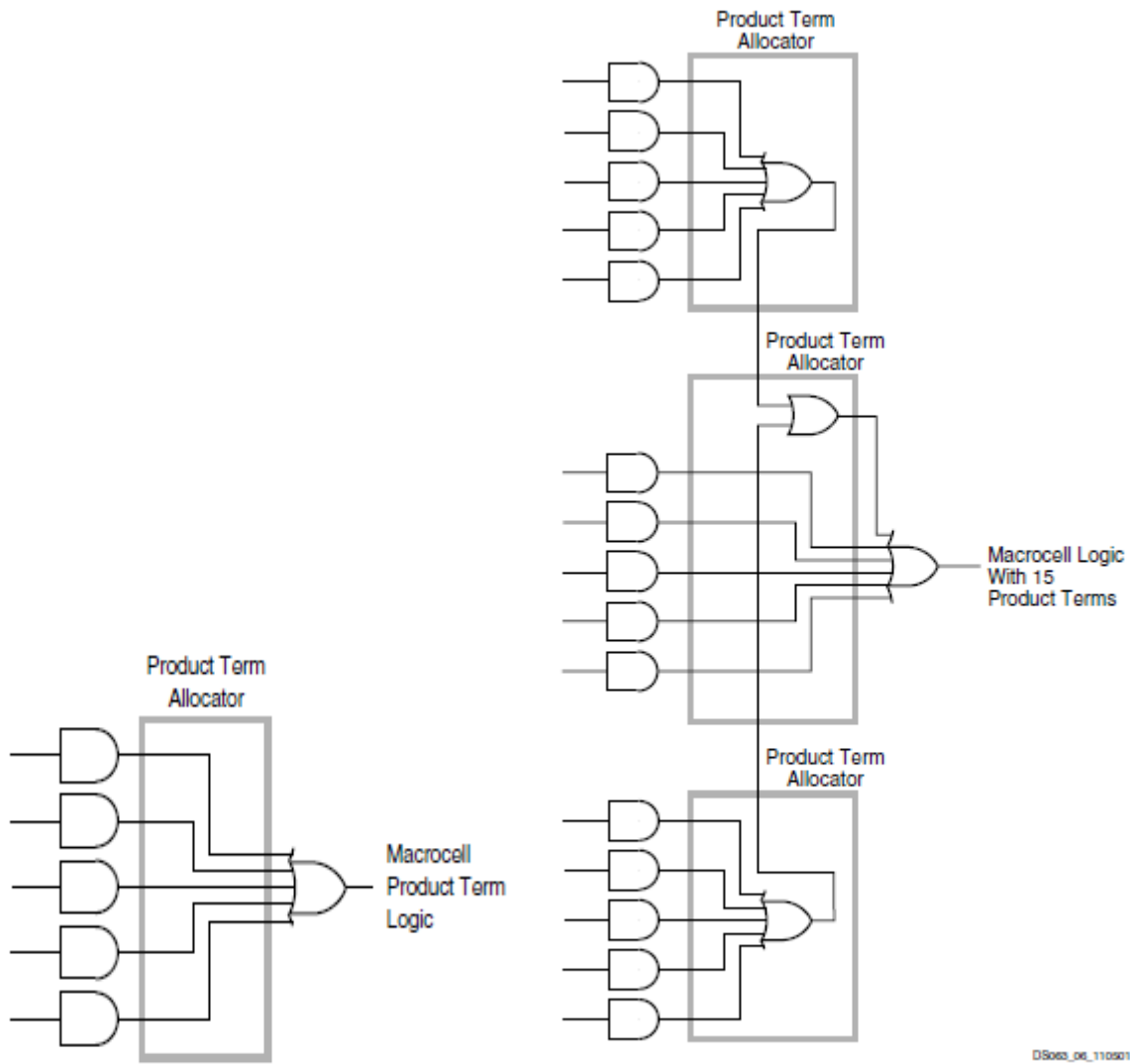


Figure 5: Macrocell Logic Using Direct Product Term

Figure 6: Product Term Allocation With 15 Product Terms

Fast CONNECT Switch Matrix

The Fast CONNECT switch matrix connects signals to the FB inputs. All IOB outputs (corresponding to user pin inputs) and all FB outputs drive the Fast CONNECT matrix.

I/O Block

The I/O Block (IOB) interfaces between the internal logic and the device user I/O pins. Each IOB includes an input buffer, output driver, output enable selection multiplexer, and user programmable ground control.

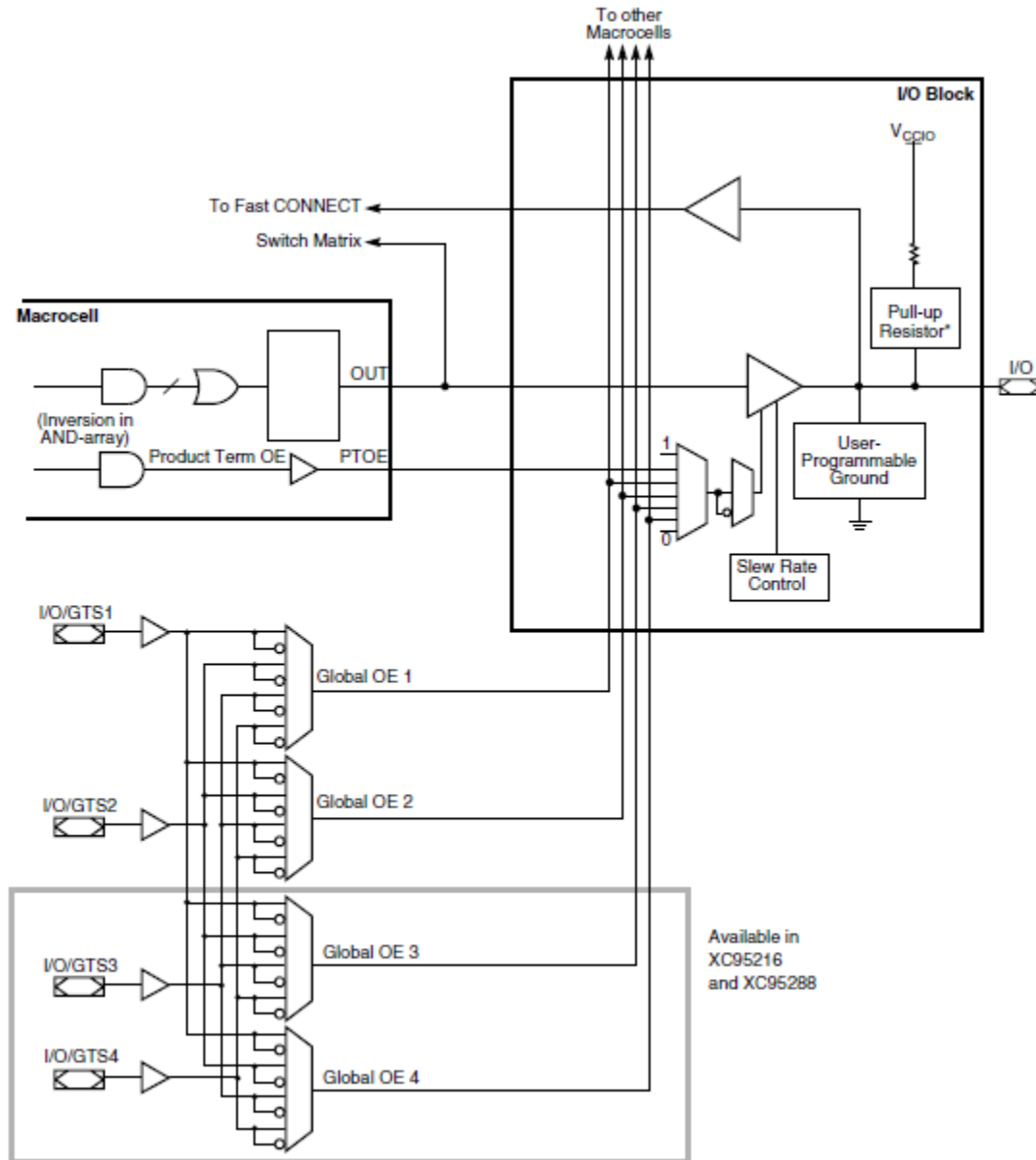


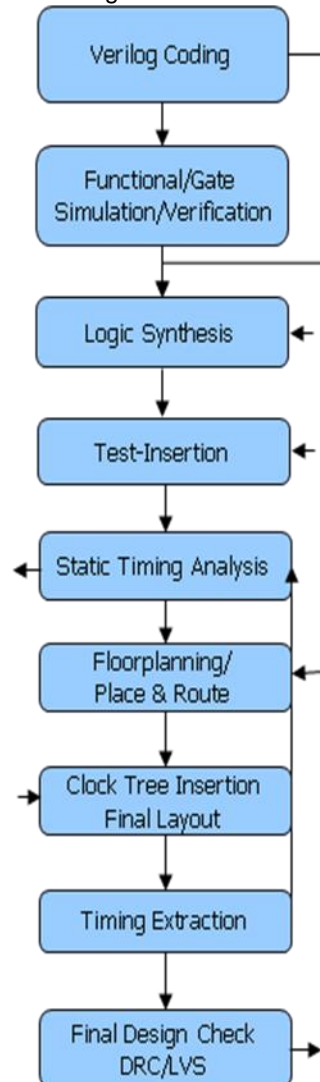
Figure 10: I/O Block and Output Enable Capability

Standard cell design

Standard cell design is another approach to ASIC using PLDs. In this a library of standard cells are available for design. The cell module consists of basic gates, flip-flops, and some commonly used functions such as decoders, multiplexers, and full adders, as well as memories. Standard cells have lower development cost. In terms of size and performance these are less efficient.

Design Approach

Design steps to flow to implement vlsi circuits using PLDs



Parameters influencing low power design

- 1.Lowest supply voltage
- 2.small frequency
- 3.Logic style
- 4.Parallelism and pipelining
- 5.Power minimization techniques
- 6.Load capacitance

CMOS TESTING:

Objective: At the end of this unit we will be able to understand

- Design for testability (DFT)
- DFT methods for digital circuits:
 - Ad-hoc methods
 - Structured methods:
 - Scan
 - Level Sensitive Scan Design
 - Boundary scan
 - Other Scan Techniques

Definition:

Design for testability (DFT) refers to those design techniques that make test generation and test application cost-effective.

Some terminologies:

Input / output (I/O) pads

- Protection of circuitry on chip from damage
- Care to be taken in handling all MOS circuits
- Provide necessary buffering between the environments On & OFF chip
- Provide for the connections of power supply
- Pads must be always placed around the peripheral

Minimum set of pads include:

- VDD connection pad
- GND(VSS) connection pad
- Input pad
- Output pad
- Bidirectional I/O pad

Designer must be aware of:

- nature of circuitry
- ratio/size of inverters/buffers on which output lines are connected
- how input lines pass through the pad circuit (pass transistor/transmission gate)

System delays

Buses:

- convenient concept in distributing data & control through a system
- bidirectional buses are convenient
- in design of datapath
- problems: capacitive load present

- largest capacitance
- sufficient time must be allowed to charge the total bus
- clock ϕ_1 & ϕ_2

Control paths, selectors & decoders

1. select registers and open pass transistors to connect cells to bus
2. Data propagation delay bus
3. Carry chain delay

Faults and Fault Modeling

A fault model is a model of how a physical or parametric fault manifests itself in the circuit Operation. Fault tests are derived based on these models Physical Faults are caused due to the following reasons:

- Defect in silicon substrate
- Photolithographic defects
- Mask contamination and scratches
- Process variations and abnormalities
- Oxide defects

Physical faults cause Electrical and Logical faults

Logical Faults are:

- Single/multiple stuck-at (*most used*)
- CMOS stuck-open
- CMOS stuck-on
- AND / OR Bridging faults

Electrical faults are due to short, opens, transistor stuck on, stuck open, excessive steady state currents, resistive shorts and open.

Design for Testability

Two key concepts

- Observability
- Controllability

DFT often is associated with design modifications that provide improved access to internal circuit elements such that the local internal state can be controlled (controllability) and/or observed (observability) more easily. The design modifications can be strictly physical in nature (e.g., adding a physical probe point to a net) and/or add active circuit elements to facilitate controllability/observability (e.g., inserting a multiplexer into a net). While controllability and observability improvements for internal circuit elements definitely are important for test, they are not the only type of DFT

What can we do to increase testability?

♦ **increase observability**

- ⇒ add more pins (?!)
- ⇒ add small "probe" bus, selectively enable different values onto bus
- ⇒ use a hash function to "compress" a sequence of values (e.g., the values of a bus over many clock cycles) into a small number of bits for later read-out
- ⇒ cheap read-out of all state information

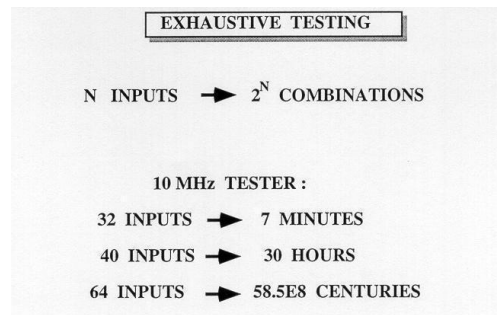
♦ **increase controllability**

- ⇒ use muxes to isolate submodules and select sources of test data as inputs
- ⇒ provide easy setup of internal state

Testing combinational logic

The solution to the problem of testing a purely combinational logic block is a good set of patterns detecting "all" the possible faults.

The first idea to test an N input circuit would be to apply an N-bit counter to the inputs (controllability), then generate all the 2^N combinations, and observe the outputs for checking (observability). This is called "exhaustive testing", and it is very efficient... but only for few- input circuits. When the input number increase, this technique becomes very time consuming.

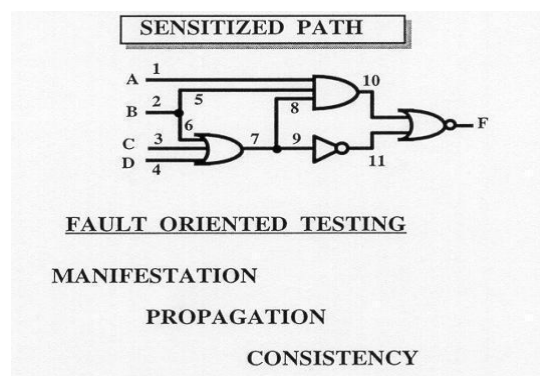


Sensitized Path Testing

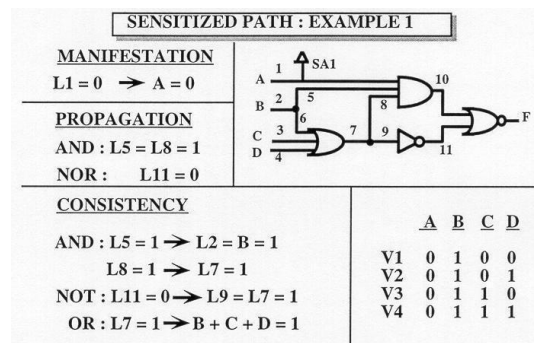
Most of the time, in exhaustive testing, many patterns do not occur during the application of the circuit. So instead of spending a huge amount of time searching for faults everywhere, the possible faults are first enumerated and a set of appropriate vectors are then generated. This is called "single-path sensitization" and it is based on "fault oriented testing".

The basic idea is to select a path from the site of a fault, through a sequence of gates leading to an output of the combinational logic under test. The process is composed of three steps :

- **Manifestation** : gate inputs, at the site of the fault, are specified as to generate the opposite value of the faulty value (0 for SA1, 1 for SA0).
- **Propagation** : inputs of the other gates are determined so as to propagate the fault signal along the specified path to the primary output of the circuit. This is done by setting these inputs to "1" for AND/NAND gates and "0" for OR/NOR gates.
- **Consistency** : or justification. This final step helps finding the primary input pattern that will realize all the necessary input values. This is done by tracing backward from the gate inputs to the primary inputs of the logic in order to receive the test patterns.



Example1 - SA1 of line1 (L1) : the aim is to find the vector(s) able to detect this fault.

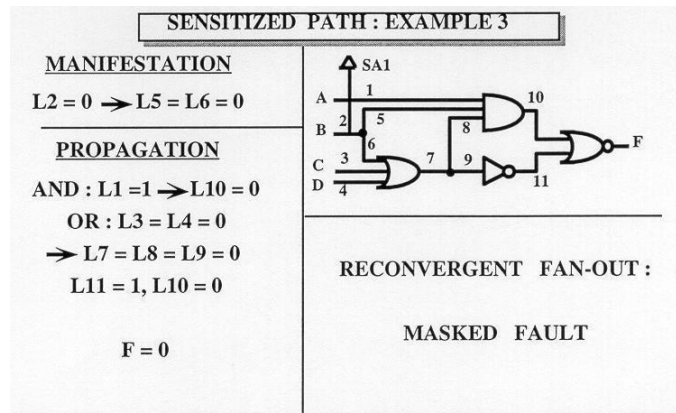


- **Manifestation:** L1 = 0 , then input A = 0. In a fault-free situation, the output F changes with A if B,C and D are fixed : for B,C and D fixed, L1 is SA1 gives F = 0, for instance, even if A = 0 (F = 1 for fault-free).
- **Propagation:** Through the AND-gate : L5 = L8 = 1, this condition is necessary for the propagation of the " L1 = 0 ". This leads to L10 = 0. Through the NOR-gate, and since L10 = 0, then L11 = 0, so the propagated manifestation can reach the primary output F. F is then read and compared with the fault-free value: F = 1.

- **Consistency:** From the AND-gate : $L5=1$, and then $L2=B=1$. Also $L8=1$, and then $L7=1$. Until now we found the values of A and B. When C and D are found, then the test vectors are generated, in the same manner, and ready to be applied to detect $L1= SA1$. From the NOT-gate, $L11=0$, so $L9=L7=1$ (coherency with $L8=L7$). From the OR-gate $L7=1$, and since $L6=L2=B=1$, so $B+C+D=L7=1$, then C and D can have either 1 or 0.

These three steps have led to four possible vectors detecting $L1=SA1$.

Example 2 - SA1 of line8 (L8) : The same combinational logic having one internal line SA1



- **Manifestation :** $L8 = 0$
- **Propagation:** Through the AND-gate: $L5 = L1 = 1$, then $L10 = 0$ Through the NOR-gate: we want to have $L11 = 0$, not to mask $L10 = 0$.
- **Consistency:** From the AND-gate $L8 = 0$ leads to $L7 = 0$. From the NOT-gate $L11 = 0$ means $L9 = L7 = 1$, $L7$ could not be set to 1 and 0 at the same time. This incompatibility could not be resolved in this case, and the fault "L8 SA1" remains undetectable.

D – Algorithm:

Given a circuit comprising combinational logic, the algorithm aims to find an assignment of input values that will allow detection of a particular internal fault by examining the output conditions. Using this algorithm the system can either be said as good or faulty. The existence of a fault in the faulty machine will cause a discrepancy between its behavior and that of the good machine for some particular values of inputs. The D-algorithm provides a systematic means of assigning input values for that particular design so that the discrepancy is driven to an output where it may be observed and thus detected. The algorithm is time-intensive and computing intensive for large circuits.

Practical design for test guidelines

Practical guidelines for testability should aim to facilitate test processes in three main ways:

- facilitate test generation
- facilitate test application
- avoid timing problems

These matters are discussed as below:

Improve Controllability and Observability

All "design for test" methods ensure that a design has enough observability and controllability to provide for a complete and efficient testing. When a node has difficult access from primary inputs or outputs (pads of the circuit), a very efficient method is to add internal pads according to this kind of node in order, for instance, to control block B2 and observe block B1 with a probe.

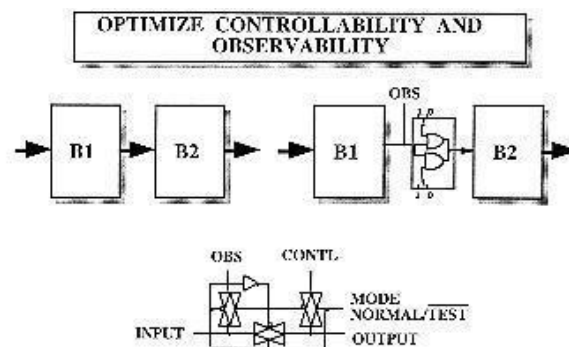


Figure 8.1 Improve Controllability and Observability

It is easy to observe block B1 by adding a pad just on its output, without breaking the link between the two blocks. The control of the block B2 means to set a 0 or a 1 to its input, and also to be transparent to the link B1-B2. The logic functions of this purpose are a NOR-gate, transparent to a zero, and a NAND-gate, transparent to a one. By this way the control of B2 is possible across these two gates.

Another implementation of this cell is based on pass-gates multiplexers performing the same function, but with less transistors than with the NAND and NOR gates (8 instead of 12).

The simple optimization of observation and control is not enough to guarantee a full testability of the blocks B1 and B2. This technique has to be completed with some other techniques of testing depending on the internal structures of blocks B1 and B2.

This technique is an extension of the precedent, while multiplexers are used in case of limitation of primary inputs and outputs.

In this case the major penalties are extra devices and propagation delays due to multiplexers. Demultiplexers are also used to improve observability. Using multiplexers and demultiplexers allows internal access of blocks separately from each other, which is the basis of techniques based on partitioning or bypassing blocks to observe or control separately other blocks.

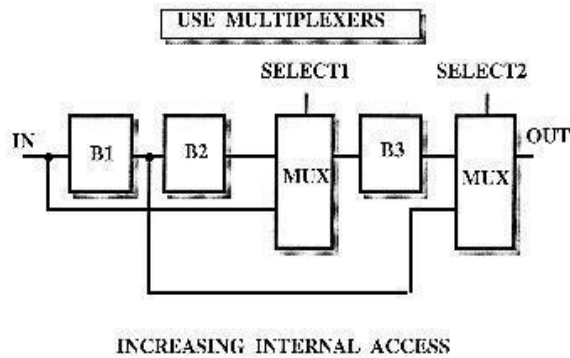


Figure 8.2: Use multiplexers

Partition Large Circuits

Partitioning large circuits into smaller sub-circuits reduces the test-generation effort. The test-generation effort for a general purpose circuit of n gates is assumed to be proportional to somewhere between n^2 and n^3 . If the circuit is partitioned into two sub-circuits, then the amount of test generation effort is reduced correspondingly.

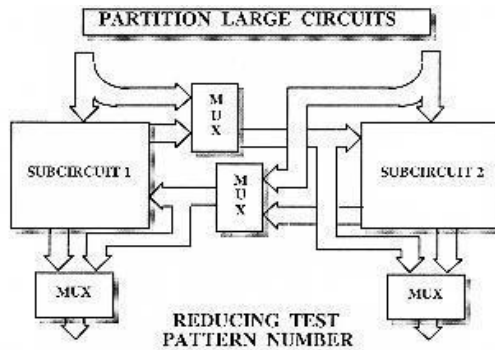


Figure 8.3: Partition Large Circuits

Logical partitioning of a circuit should be based on recognizable sub-functions and can be achieved physically by incorporating some facilities to isolate and control

clock lines, reset lines and power supply lines. The multiplexers can be massively used to separate sub-circuits without changing the function of the global circuit.

Divide Long Counter Chains

Based on the same principle of partitioning, the counters are sequential elements that need a large number of vectors to be fully tested. The partitioning of a long counter corresponds to its division into sub-counters.

The full test of a 16-bit counter requires the application of $2^{16} + 1 = 65537$ clock pulses. If this counter is divided into two 8-bit counters, then each counter can be tested separately, and the total test time is reduced 128 times (27). This is also useful if there are subsequent requirements to set the counter to a particular count for tests associated with other parts of the circuit: pre-loading facilities.

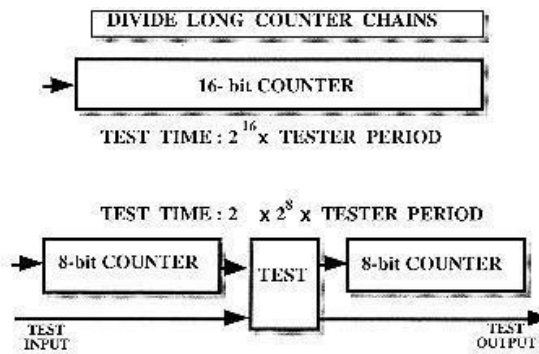


Figure 8.4: Divide Long Counter Chains

Initialize Sequential Logic

One of the most important problems in sequential logic testing occurs at the time of power-on, where the first state is random if there were no initialization. In this case it is impossible to start a test sequence correctly, because of memory effects of the sequential elements.

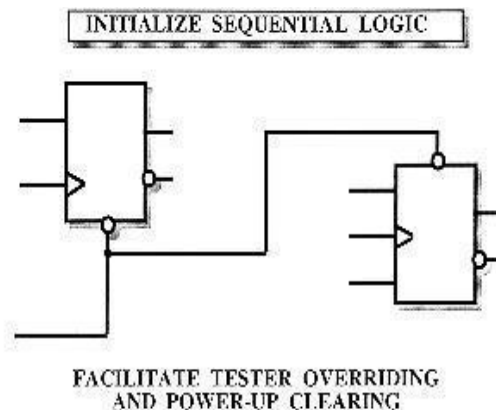


Figure 8.5: Initialize Sequential Logic

The solution is to provide flip-flops or latches with a set or reset input, and then to use them so that the test sequence would start with a known state.

Ideally, all memory elements should be able to be set to a known state, but practically this could be very surface consuming, also it is not always necessary to initialize all the sequential logic. For example, a serial-in serial-out counter could have its first flip-flop provided with an initialization, then after a few clock pulses the counter is in a known state.

Overriding of the tester is necessary some times, and requires the addition of gates before a Set or a Reset so the tester can override the initialization state of the logic.

Avoid Asynchronous Logic

Asynchronous logic uses memory elements in which state-transitions are controlled by the sequence of changes on the primary inputs. There is thus no way to determine easily when the next state will be established. This is again a problem of timing and memory effects.

Asynchronous logic is faster than synchronous logic, since the speed in asynchronous logic is only limited by gate propagation delays and interconnects. The design of asynchronous logic is then more difficult than synchronous (clocked) logic and must be carried out with due regards to the possibility of critical races (circuit behavior depending on two inputs changing simultaneously) and hazards (occurrence of a momentary value opposite to the expected value).

Non-deterministic behavior in asynchronous logic can cause problems during fault simulation. Time dependency of operation can make testing very difficult, since it is sensitive to tester signal skew.

Avoid Logical Redundancy

Logical redundancy exists either to mask a static-hazard condition, or unintentionally (design bug). In both cases, with a logically redundant node it is not possible to make a primary output value dependent on the value of the redundant node. This means that certain fault conditions on the node cannot be detected, such as a node SA1 of the function F.

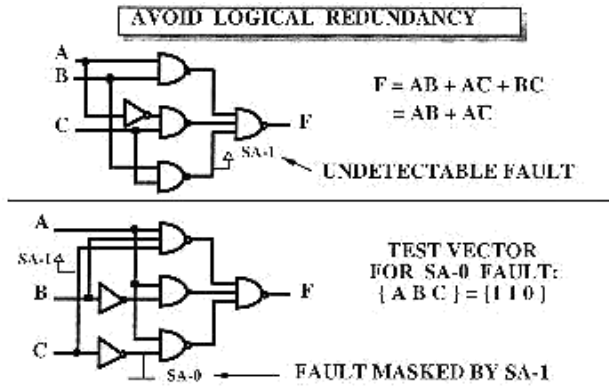


Figure 8.6: Avoid Logical Redundancy

Another inconvenience of logical redundancy is the possibility for a non-detectable fault on a redundant node to mask the detection of a fault normally-detectable, such a SA0 of input C in the second example, masked by a SA1 of a redundant node.

Avoid Delay Dependent Logic

Automatic test pattern generators work in logic domains, they view delay dependent logic as redundant combinational logic. In this case the ATPG will see an AND of a signal with its complement, and will therefore always compute a 0 on the output of the AND-gate (instead of a pulse). Adding an OR-gate after the AND-gate output permits to the ATPG to substitute a clock signal directly.

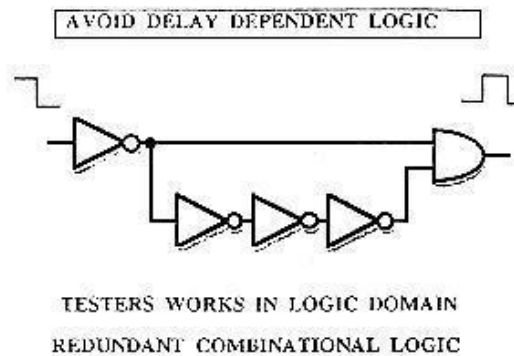


Figure 8.7: Avoid Delay Dependent Logic

Avoid Clock Gating

When a clock signal is gated with any data signal, for example a load signal coming from a tester, a skew or any other hazard on that signal can cause an error on the output of logic.

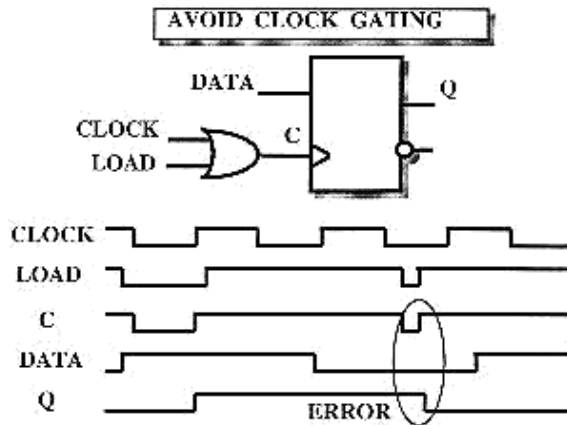


Figure 8.8: Avoid Clock Gating

This is also due to asynchronous type of logic. Clock signals should be distributed in the circuit with respect to synchronous logic structure.

Distinguish Between Signal and Clock

This is another timing situation to avoid, in which the tester could not be synchronized if one clock or more are dependent on asynchronous delays (across D-input of flip-flops, for example).

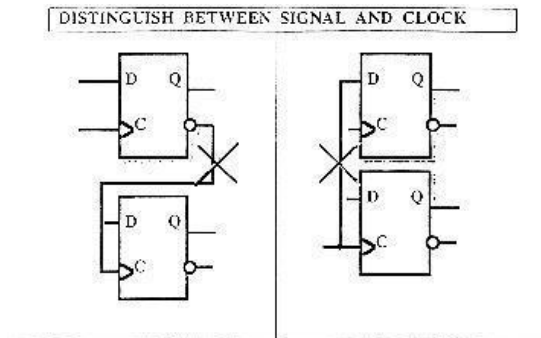


Figure 8.9: Distinguish Between Signal and Clock

Avoid Self Resetting Logic

The self resetting logic is more related to asynchronous logic, since a reset input is independent of clock signal.

Before the delayed reset, the tester reads the set value and continues the normal operation. If a reset has occurred before tester observation, then the read value is erroneous. The solution to this problem is to allow the tester to override by adding an OR-gate, for example, with an inhibition input coming from the tester. By this way the right response is given to the tester at the right time.

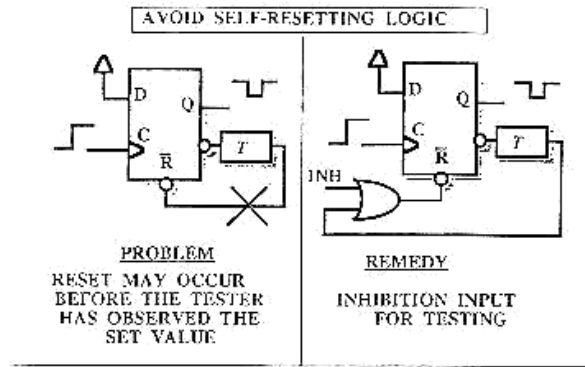


Figure 8.10: Avoid Self Resetting Logic

Use Bused Structure

This approach is related, by structure, to partitioning technique. It is very useful for microprocessor-like circuits. Using this structure allows the external tester the access of three buses, which go to many different modules.

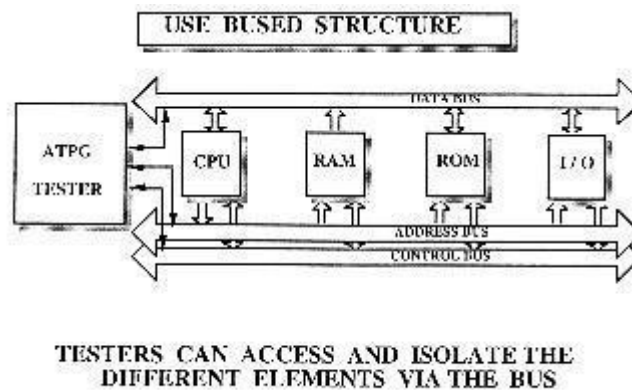


Figure 8.11: Use Bused Structure

The tester can then disconnect any module from the buses by putting its output into a high-impedance state. Test patterns can then be applied to each module separately.

Separate Analog and Digital Circuits

Testing analog circuit requires a completely different strategy than for digital circuit. Also the sharp edges of digital signals can cause cross-talk problem to the analog lines, if they are close to each other.

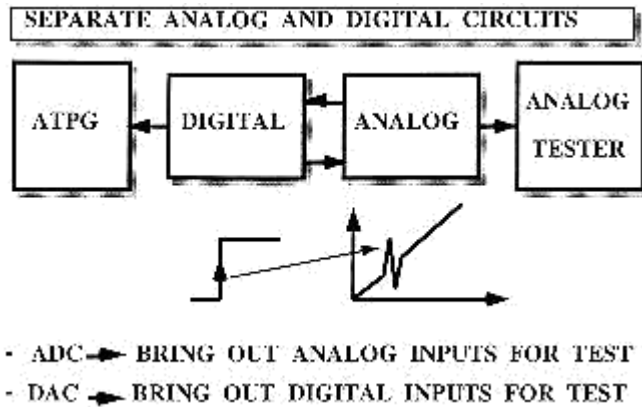


Figure 8.12: Separate Analog and Digital Circuits

If it is necessary to route digital signals near analog lines, then the digital lines should be properly balanced and shielded. Also, in the cases of circuits like Analog-Digital converters, it is better to bring out analog signals for observation before conversion. For Digital-Analog converters, digital signals are to be brought out also for observation before conversion.

Ad-Hoc DFT Method

📚 Good design practices learnt through experience are used as guidelines:

- Avoid asynchronous (unlocked) feedback.
- Make flip-flops initializable.
- Avoid redundant gates. Avoid large fan-in gates.
- Provide test control for difficult-to-control signals.
- Avoid gated clocks.
- Avoid delay dependant logic.
- Avoid parallel drivers.
- Avoid monostable and self-resetting logic.

📚 Design Reviews

Manual analysis

- Conducted by

experts Programmed analysis

- Using design auditing tools

Programmed enforcement

- Must use certain design practices and cell types.

Objective: Adherence to design guidelines and testability improvement techniques with little impact on performance and area.

- ✚ Disadvantages of ad-hoc DFT methods:
 - Experts and tools not always available.
 - Test generation is often manual with no guarantee of high fault coverage.
 - Design iterations may be necessary.

Scan Design Techniques

The set of design for testability guidelines presented above is a set of ad hoc methods to design random logic in respect with testability requirements. The scan design techniques are a set of structured approaches to design (for testability) the sequential circuits.

The major difficulty in testing sequential circuits is determining the internal state of the circuit. Scan design techniques are directed at improving the controllability and observability of the internal states of a sequential circuit. By this the problem of testing a sequential circuit is reduced to that of testing a combinational circuit, since the internal states of the circuit are under control.

Scan Path

The goal of the scan path technique is to reconfigure a sequential circuit, for the purpose of testing, into a combinational circuit. Since a sequential circuit is based on a combinational circuit and some storage elements, the technique of scan path consists in connecting together all the storage elements to form a long serial shift register. Thus the internal state of the circuit can be observed and controlled by shifting (scanning) out the contents of the storage elements. The shift register is then called a scan path.

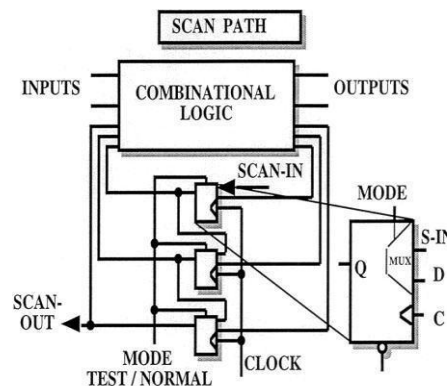


Figure 8.13: Scan Path

The storage elements can either be D, J-K, or R-S types of flip-flops, but simple latches cannot be used in scan path. However, the structure of storage elements is slightly different than classical ones. Generally the selection of the input source is achieved using a multiplexer on the data input controlled by an external mode signal. This multiplexer is integrated into the D-flip-flop, in our case; the D-flip-flop is then called MD-flip-flop (multiplexed-flip-flop).

The sequential circuit containing a scan path has two modes of operation: a normal mode and a test mode which configure the storage elements in the scan path.

As analyzed from figure 8.13, in the normal mode, the storage elements are connected to the combinational circuit, in the loops of the global sequential circuit, which is considered then as a finite state machine.

In the test mode, the loops are broken and the storage elements are connected together as a serial shift register (scan path), receiving the same clock signal. The input of the scan path is called scan-in and the output scan-out. Several scan paths can be implemented in one same complex circuit if it is necessary, though having several scan-in inputs and scan-out outputs.

A large sequential circuit can be partitioned into sub-circuits, containing combinational sub-circuits, associated with one scan path each. Efficiency of the test pattern generation for a combinational sub-circuit is greatly improved by partitioning, since its depth is reduced.

Before applying test patterns, the shift register itself has to be verified by shifting in all ones i.e. 111...11, or zeros i.e. 000...00, and comparing.

The method of testing a circuit with the scan path is as follows:

1. Set test mode signal, flip-flops accept data from input scan-in
2. Verify the scan path by shifting in and out test data
3. Set the shift register to an initial state
4. Apply a test pattern to the primary inputs of the circuit
5. Set normal mode, the circuit settles and can monitor the primary outputs of the circuit
6. Activate the circuit clock for one cycle
7. Return to test mode
8. Scan out the contents of the registers, simultaneously scan in the next pattern

Level sensitivity scan design (LSSD)

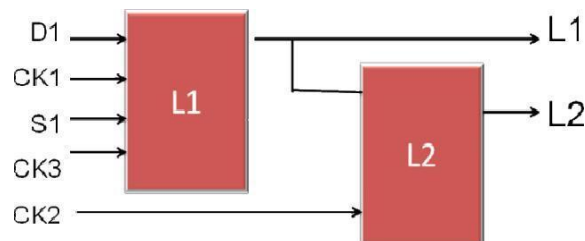


Figure 8.14: Level sensitivity scan design

The level-sensitive aspect means that the sequential network is designed so that when an input change occurs, the response is independent of the component and wiring delays within the network (Figure 8.14).

The scan path aspect is due to the use of shift register latches (SRL) employed as storage elements. In the test mode they are connected as a long serial shift register. Each SRL has a specific design similar to a master-slave FF. It is driven by two non-overlapping clocks which can be controlled readily from the primary inputs to the circuit. Input D1 is the normal data input to the SRL; clocks CK1 and CK2 control the normal operation of the SRL while clocks CK3 and CK2 control scan path movements through the SRL. The SRL output is derived at L2 in both modes of operation, the mode depending on which clocks are activated.

Advantages:

- Circuit operation is independent of dynamic characteristics of the logic elements
- ATP generation is simplified
- Eliminate hazards and races
- Simplifies test generation and fault simulation

Boundary Scan Test (BST)

Boundary Scan Test (BST) is a technique involving scan path and self-testing techniques to resolve the problem of testing boards carrying VLSI integrated circuits and/or surface mounted devices (SMD).

Printed circuit boards (PCB) are becoming very dense and complex, especially with SMD circuits, that most test equipment cannot guarantee good fault coverage.

BST (figure 8.15) consists in placing a scan path (shift register) adjacent to each component pin and to interconnect the cells in order to form a chain around the border of the circuit. The BST circuits contained on one board are then connected together to form a single path through the board.

The boundary scan path is provided with serial input and output pads and appropriate clock pads which make it possible to:

- Test the interconnections between the various chip
- Deliver test data to the chips on board for self-testing
- Test the chips themselves with internal self-test

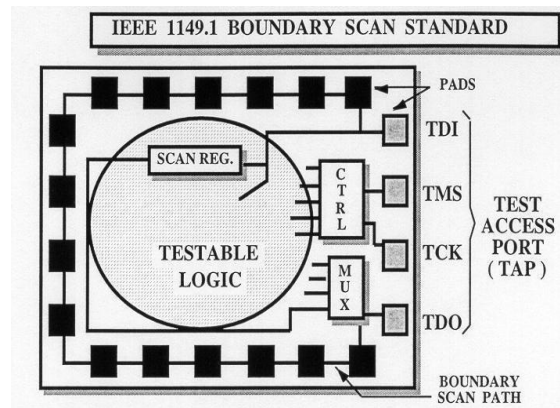


Figure 8.15: Boundary Scan Test (BST)

The advantages of Boundary scan techniques are as follows :

- No need for complex testers in PCB testing
- Test engineers work is simplified and more efficient
- Time to spend on test pattern generation and application is reduced
- Fault coverage is greatly increased.

Other scan techniques:

Partial Scan Method

- A subset of flip-flops is scanned.
- Objectives:
 - Minimize area overhead and scan sequence length, yet achieve required fault coverage
 - Exclude selected flip-flops from scan:
 - Improve performance
 - Allow limited scan design rule violations
 - Allow automation:
 - In scan flip-flop selection
 - In test generation
 - Shorter scan sequences – reduce application time

Random Access Scan Method

- The scan function is implemented like a random-access memory (RAM)
- All flip-flops form a RAM in scan mode
- A subset of flip-flops can be included in the RAM if partial scan is desired
- In scan mode, any flip-flop can be read or written

Procedure:

Set test inputs to all test points

Apply the master reset signal to initialize all memory elements

Set scan-in address & data, then apply the scan clock

Repeat the above step until all internal test inputs are scanned

Clock once for normal operation

Check states of the output points

Read the scan-out states of all memory elements by applying the address

Built-in-self test

Objectives:

1. To reduce test pattern generation cost
2. To reduce volume of test data
3. To reduce test time

Built-in Self Test, or BIST, is the technique of designing additional hardware and software features into integrated circuits to allow them to perform self-testing, i.e., testing of their own operation (functionally, parametrically, or both) using their own circuits, thereby reducing dependence on an external automated test equipment (ATE).

BIST is a Design-for-Testability (DFT) technique, because it makes the electrical testing of a chip easier, faster, more efficient, and less costly. The concept of BIST is applicable to just about any kind of circuit, so its implementation can vary as widely as the product diversity that it caters to. As an example, a common BIST approach for

DRAM's includes the incorporation onto the chip of additional circuits for pattern generation, timing, mode selection, and go-/no-go diagnostic tests.

Advantages of implementing BIST include:

- 1) Lower cost of test, since the need for external electrical testing using an ATE will be reduced, if not eliminated
- 2) Better fault coverage, since special test structures can be incorporated onto the chips
- 3) Shorter test times if the BIST can be designed to test more structures in parallel
- 4) Easier customer support and
- 5) Capability to perform tests outside the production electrical testing environment. The last advantage mentioned can actually allow the consumers themselves to test the chips prior to mounting or even after these are in the application boards.

Disadvantages of implementing BIST include:

- 1) Additional silicon area and fab processing requirements for the BIST circuits
- 2) Reduced access times
- 3) Additional pin (and possibly bigger package size) requirements, since the BIST circuitry need a way to interface with the outside world to be effective and
- 4) Possible issues with the correctness of BIST results, since the on-chip testing hardware itself can fail.

Techniques are:

- compact test: signature analysis
- linear feedback shift register
- BILBO
- self checking technique

Compact Test: Signature analysis

Signature analysis performs polynomial division that is, division of the data out of the device under test (DUT). This data is represented as a polynomial $P(x)$ which is divided by a characteristic polynomial $C(x)$ to give the signature $R(x)$, so that

$$R(x) = P(x)/C(x)$$

This is summarized as in figure 8.16.

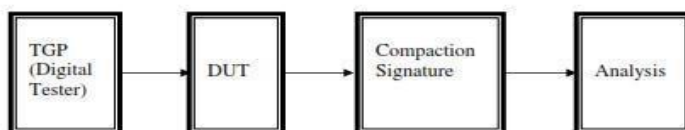


Figure 8.16: BIST – signature analysis

Linear feedback shift register (LFSR):

An LFSR is a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit. Some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. A linear feedback shift register can be formed by performing exclusive-OR (Figure 8.16) on the outputs of two or more of the flip-flops together and feeding those outputs back into the input of one of the flip-flops.

LFSR technique can be applied in a number of ways, including random number generation, polynomial division for signature analysis, and n-bit counting. LFSR can be series or parallel, the differences being in the operating speed and in the area of silicon occupied; Parallel LFSR being faster but larger than serial LFSR.

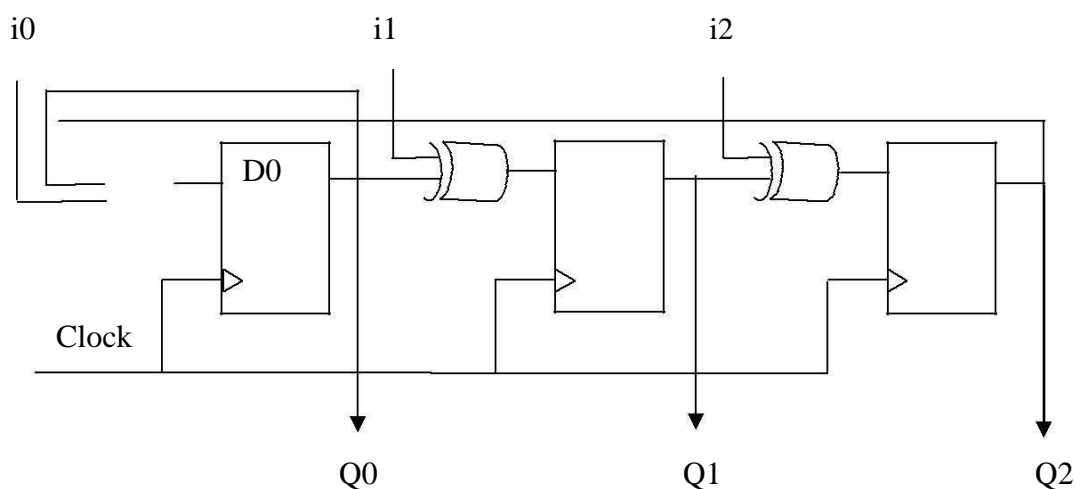


Figure 8.16: Linear feedback shift register

Built-in logic block observer (BILBO):

BILBO is a built-in test generation scheme which uses signature analysis in conjunction with a scan path. The major component of a BILBO is an LFSR with a few gates (Figure 8.17).

A **BILBO** register (**built-in logic block observer**) combines normal flipflops with a few additional gates to provide four different functions. The example circuit shown in the applet realizes a four-bit register. However, the generalization to larger bit-widths should be obvious, with the XOR gates in the LFSR feedback path chosen to implement a good polynomial for the given bit-width.

When the **A** and **B** control inputs are both 1, the circuit functions as a normal parallel D-type register.

When both **A** and **B** inputs are 0, the D-inputs are ignored (due to the AND gate connected to A), but the flipflops are connected as a shift-register via the NOR and XOR gates. The input to the first flipflop is then selected via the multiplexer controlled by the **S** input. If the S input is 1, the multiplexer transmits the value of the external **SIN** shift-in input to the first flipflop, so that the BILBO register works as a normal shift-register. This allows to initialize the register contents using a single signal wire, e.g. from an external test controller.

If all of the **A**, **B**, and **S** inputs are 0, the flipflops are configured as a shift-register, again, but the input bit to the first flipflop is computed by the XOR gates in the LFSR feedback path. This means that the register works as a standard LFSR pseudorandom pattern generator, useful to drive the logic connected to the Q outputs. Note that the start value of the LFSR sequence can be set by shifting it in via the SIN input.

Finally, if **B** and **S** are 0 but **A** is 1, the flipflops are configured as a shift-register, but the input value of each flipflop is the XOR of the D-input and the Q-output of the previous flipflop. This is exactly the configuration of a standard LFSR signature analysis register.

Because a BILBO register can be used as a pattern generator for the block it drives, as well provide signature-analysis for the block it is driven by, a whole circuit can be made self-testable with very low overhead and with only minimal performance degradation (two extra gates before the D inputs of the flipflops).

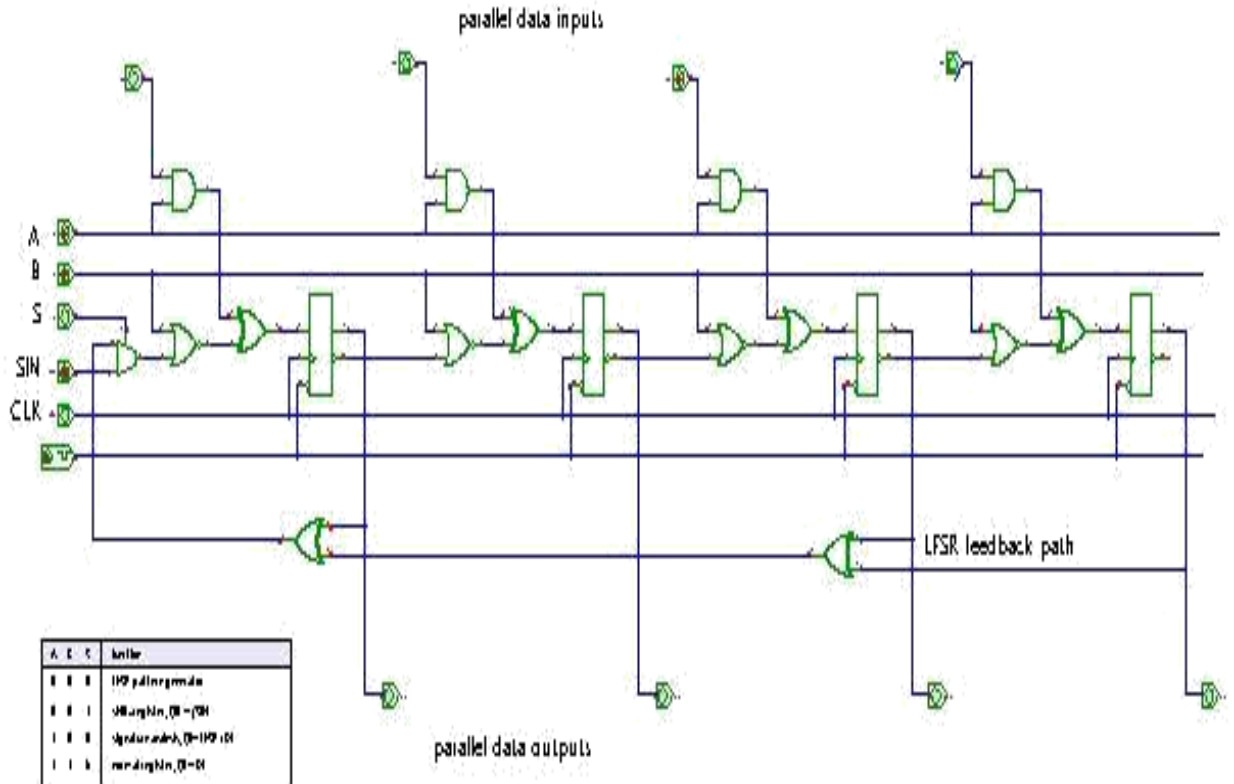


Figure 8.17: BIST – BILBO

Self-checking techniques:

It consists of logic block and checkers should then obey a set of rules in which the logic block is 'strongly fault secure' and the checker 'strongly code disjoint'. The code use in data encoding depends on the type of errors that may occur at the logic block output. In general three types are possible:

- Simple error: one bit only affected at a time.
- Unidirectional error: multiple bits at 1 instead of 0 (or 0 instead of 1)
- Multiple errors: multiple bits affected in any order.

Self-checking techniques are applied to circuits in which security is important so that fault tolerance is of major interest. Such technique will occupy more area in silicon than classical techniques such as functional testing but provide very high test coverage.

