

**VSM COLLEGE OF ENGINEERING
RAMACHANDRA PURAM**

**VLSI DESIGN (R20)
III B.TECH ECE II SEMESTER**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**



JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA
KAKINADA – 533 003, Andhra Pradesh, India
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

III Year – II Semester		L	T	P	C
		3	0	0	3
VLSI DESIGN					

UNIT-I:**INTRODUCTION AND BASIC ELECTRICAL PROPERTIES OF MOS CIRCUITS:**

VLSI Design Flow, Introduction to IC technology, Fabrication process: nMOS, pMOS and CMOS. I_{ds} versus V_{ds} Relationships, Aspects of MOS transistor Threshold Voltage, MOS transistor Trans, Output Conductance and Figure of Merit. nMOS Inverter, Pull-up to Pull-down Ratio for nMOS inverter driven by another nMOS inverter, and through one or more pass transistors. Alternative forms of pull-up, The CMOS Inverter, Latch-up in CMOS circuits, Bi-CMOS Inverter, Comparison between CMOS and BiCMOS technology, MOS Layers, Stick Diagrams, Design Rules and Layout, Layout Diagrams for MOS circuits

UNIT-II:

BASIC CIRCUIT CONCEPTS: Sheet Resistance, Sheet Resistance concept applied to MOS transistors and Inverters, Area Capacitance of Layers, Standard unit of capacitance, some area Capacitance Calculations, The Delay Unit, Inverter Delays, driving large capacitive loads, Propagation Delays, Wiring Capacitances, Choice of layers.

SCALING OF MOS CIRCUITS: Scaling models and scaling factors, Scaling factors for device parameters, Limitations of scaling, Limits due to sub threshold currents, Limits on logic levels and supply voltage due to noise and current density. Switch logic, Gate logic.

UNIT-III:

BASIC BUILDING BLOCKS OF ANALOG IC DESIGN: Regions of operation of MOSFET, Modelling of transistor, body bias effect, biasing styles, single stage amplifier with resistive load, single stage amplifier with diode connected load, Common Source amplifier, Common Drain amplifier, Common Gate amplifier, current sources and sinks.

UNIT-IV:**CMOS COMBINATIONAL AND SEQUENTIAL LOGIC CIRCUIT DESIGN:**

Static CMOS Design: Complementary CMOS, Rationed Logic, Pass-Transistor Logic.

Dynamic CMOS Design: Dynamic Logic-Basic Principles, Speed and Power Dissipation of Dynamic Logic, Issues in Dynamic Design, Cascading Dynamic Gates, Choosing a Logic Style, Gate Design in the Ultra Deep-Submicron Era, Latch Versus Register, Latch based design, timing decimation, positive feedback, in stability, Meta stability, multiplexer based latches, Master-Slave Based Edge Triggered Register, clock to q delay, setup time, hold time, reduced clock load master slave registers, Clocked CMOS register. Cross coupled NAND and NOR, SR Master Slave register, Storage mechanism, pipelining.

UNIT-V:

FPGA DESIGN: FPGA design flow, Basic FPGA architecture, FPGA Technologies, Introduction to FPGA Families.

INTRODUCTION TO ADVANCED TECHNOLOGIES: Giga-scale dilemma, Short channel effects, High-k, Metal Gate Technology, Fin-FET, TFET.



JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA
KAKINADA – 533 003, Andhra Pradesh, India
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

TEXTBOOKS:

1. Essentials of VLSI Circuits and Systems – Kamran Eshraghian, Douglas and A.Pucknell and Sholeh Eshraghian, Prentice-Hall of India Private Limited, 2005 Edition.
2. Design of Analog CMOS Integrated Circuits by Behzad Razavi, McGraw Hill, 2003
3. Digital Integrated Circuits, Jan M.Rabaey, Anantha Chandrakasan and Borivoje Nikolic, 2nd edition, 2016.

REFERENCES:

1. “Introduction to VLSI Circuits and Systems”, John P.Uyemura, John Wiley&Sons, reprint 2009.
2. Integrated Nano electronics: Nano scale CMOS, Post-CMOS and Allied Nano technologies Vinod Kumar Khanna, Springer India, 1stedition, 2016.
3. Fin-FETs and other multi-gate transistors, Colinge JP, Editor NewYork, Springer, 2008.

Course Outcomes:

At the end of this course the student will be able to:

1. Demonstrate a clear understanding of CMOS fabrication flow and technology scaling.
2. Apply the design Rules and draw layout of a given logic circuit.
3. Design basic building blocks in Analog IC design.
4. Analyze the behavior of amplifier circuits with various loads.
5. Design various CMOS logic circuits for design of Combinational logic circuits.
6. Design MOSFET based logic circuits using various logic styles like static and dynamic CMOS.
7. Design various applications using FPGA.

Introduction & Basic Electrical properties of MOS ckt:-

VLSI - Very large Scale integration

Definition of Ic:-

Ic is an electronic for integrated circuit and may be given as combination of active or passive elements that are integrated on single silicon chip.

As there are several advantages of using Silicon which includes, it acts as good insulating material, Oxidizing material.

Most of the Ic's available in the market are made using silicon only [i.e., 90%.]

Trends in micro electronics:-

The electronics now a days available in the market are categorized by reliability, size, weight, Volume, cost.

In addition these the VLSI technology made an advantage to have a more powerful & flexible processor for availing a good source.

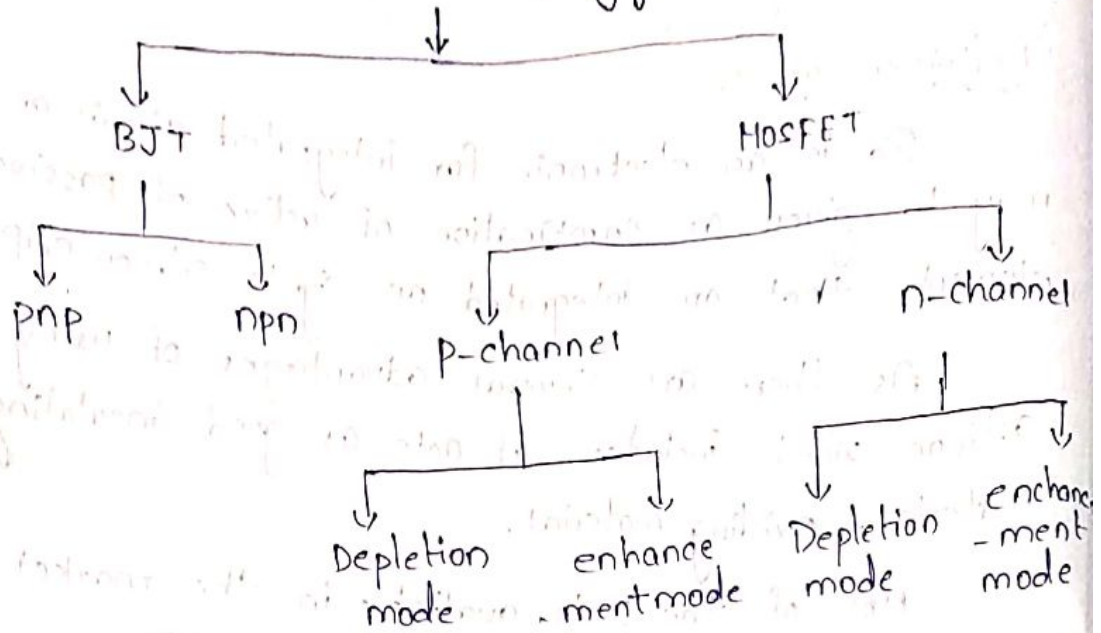
* The BJT was first invented by William Shockley & John Bardeen in 1947 at Bell Laboratories.

* up to 1950's the BJT technology was dominated by vacuum tubes.

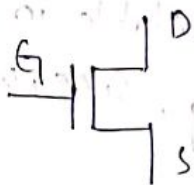
* The 1st Ic technology was developed in 1960's and there by a revolutionary come into the electronics industries.

Symbols of MOSFET:-

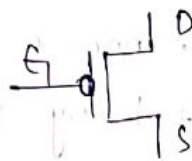
VLSI Technology



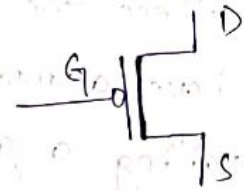
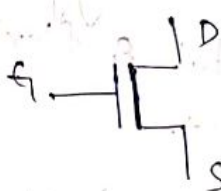
N-Mos



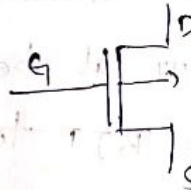
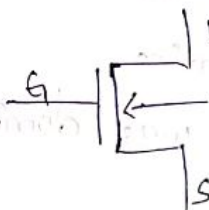
P-Mos



enhancement mode



Depletion mode



current supply

BJT is a Current Controlled device where as MOSFET is a Voltage Controlled device.

Levels of integration:-

Depending upon the complexity of integrated ckt the classification can be given as

SSI (Small scale integration) - 10 to 100

MSI (Medium scale integration) - 100 to 1000

LSI (Large scale integration) - 1000 to 10^5

VLSI (Very large scale integration) - 10^5 to 10^6

The upcoming technology i.e.,

ULSI (Ultra large scale integration) - 10 to 100 million

Difference Between BJT and MOSFET

→

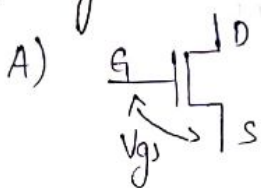
BJT

1. It is current control device.
2. Collector and emitter terminal's are not interchange.
3. Impedance is low.
4. Δp Impedance is low
5. Trans conductance is high.

MOSFET

1. It is voltage control device.
2. Drain and Source terminals are interchange.
3. Impedance is high.
4. Δp Impedance is high
5. Transconductance is low

* Why it is called FET?



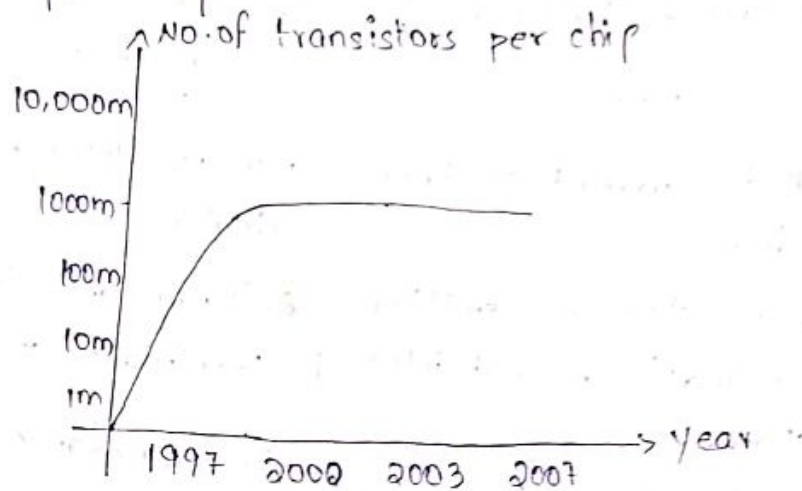
V_{gs} = Voltage b/w gate and source terminals.

By the external application of v_{gs} there is an electric field development b/w gate and source

and hence it will get effected in corresponds with vgs. Hence it is called field effect Transistor (FET).

The IC Era:-

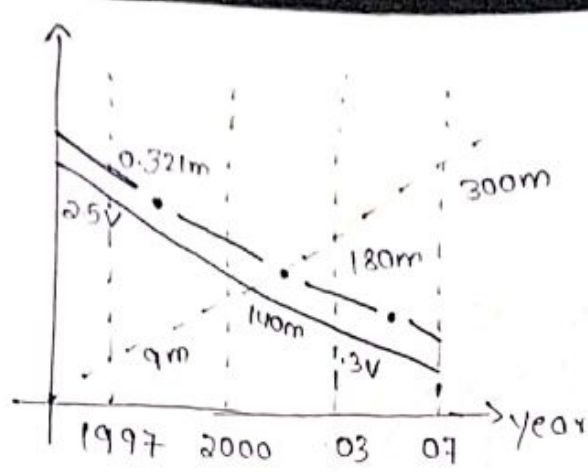
- * The first IC emerged in the early 1960's.
- * Depending upon the potential of that IC, we can find no. of transistors that are being integrated in the single silicon chip.
- * In less than 3 decades the no of transistors count has risen from 100 to 1000 millions of transistors per chip.



Moore's law:-

The graphical representation that gives the relation - ship b/w the year, $\sqrt[3]{}$ no. of transistor per chip is called moore's law.

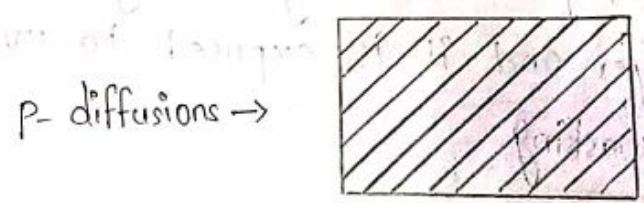
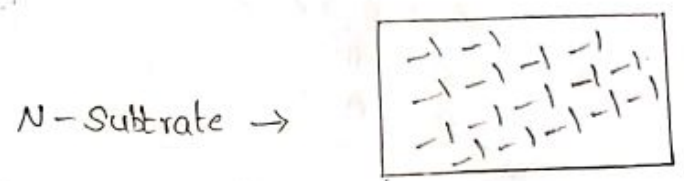
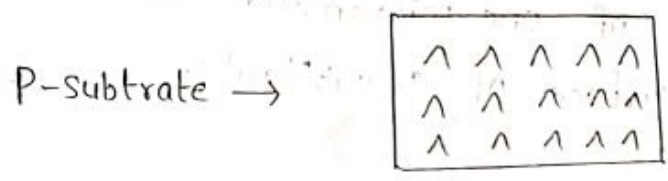
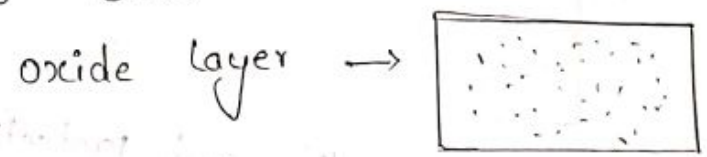
No of transistors per chip	supply voltage	channel length (μm)
10000m	3V	0.3
1000m	2.5V	0.25
100m	2V	0.2
10m	1.5V	0.15
1m	1V	1



- Channel length (μm) — — — — —
- Supply voltage — — — — —
- No. of transistors per chip - - - - -

To have powerful and flexible processors, some more modifications are made to Moore's law which includes reducing supply voltage and channel length.

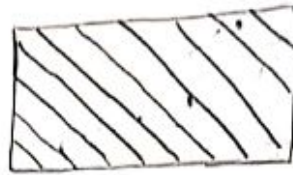
Symbol's for IC fabrication process:-



n-diffusions



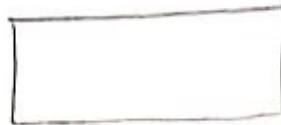
polysilicon



metal

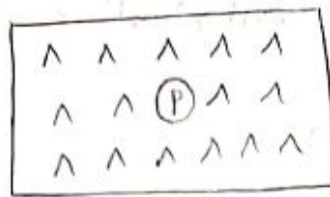


depletion

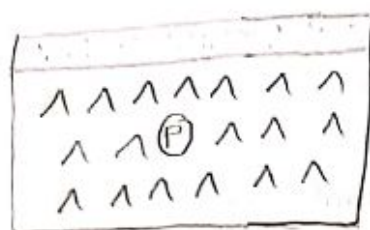


N-MOS fabrication process:-

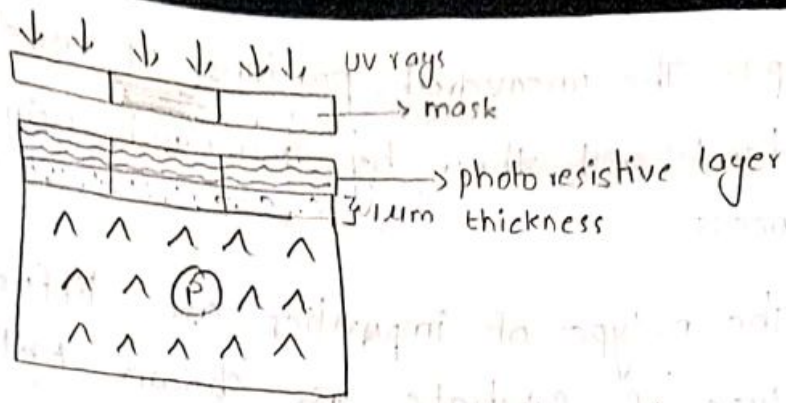
Step 1:- for the designing of n-mos transistor we have to consider p-type of substrate material.



Step 2:- To improve the quality and protection an oxide layer of one micrometer thickness is grown over all surface of p-substrate.

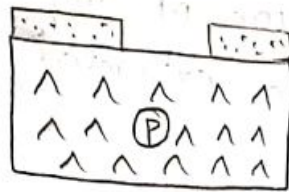


Step 3: A photoresistive layer is grown at the top of oxide layer and it is exposed to uv light through suitable masking.

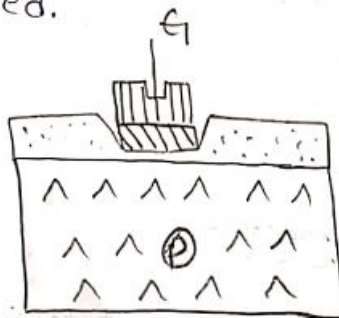


Step 4:- The uncovered portion of mask allows UV light to flow through it, the oxide layer will be get softened and remain covered position will be remain harden.

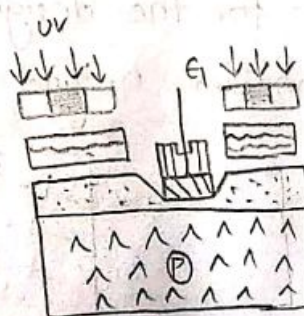
The portion which is softened will be remove and the process is called "Etching".



Step 5:- An oxide layer of $0.1 \mu\text{m}$ thickness is grown and using polysilicon and metal gate terminal can be extracted.

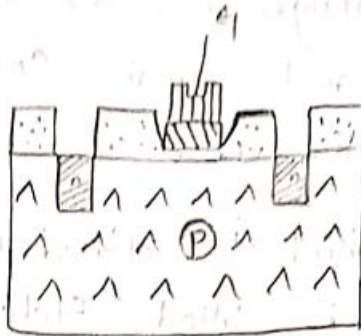


Step 6: To diffuse n-type of impurities into p-type of substrate the masking and Etching processes are again carried out

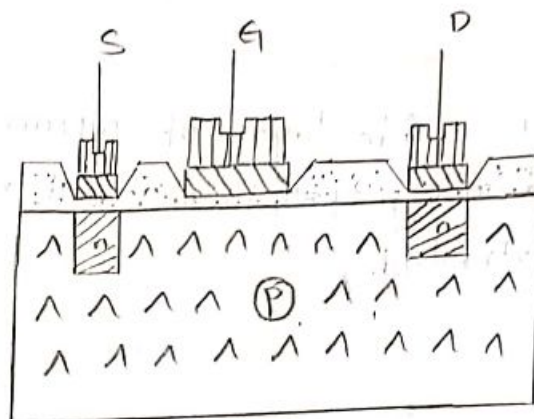


Step 7:- The uncovered portions of mask will be get softened and then by removed using etching process.

The n-type of impurities are diffused into the p-type of substrate as shown below.



Step 8:- From the n-type of impurities drain and source terminals can be extracted using polysilicon and metals.



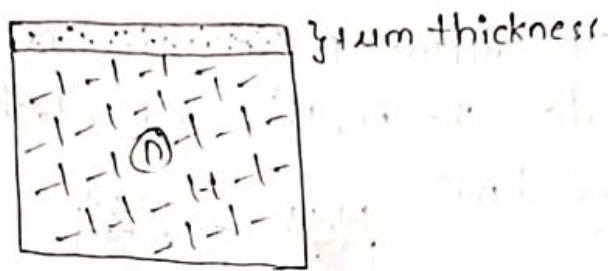
2) P-MOS fabrication Process:-

P- N -P P-diffusion
 |
 Substrate

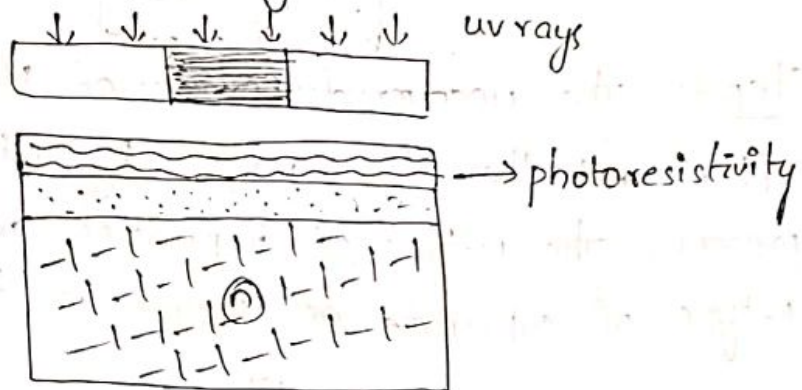
Step 1:- for the designing of P-MOS transistor we have to consider n-type of substrate material.



step 2:- To improve the quality and protection on oxide layer of $1\mu\text{m}$ thickness is grown overall surface of n-substrate.

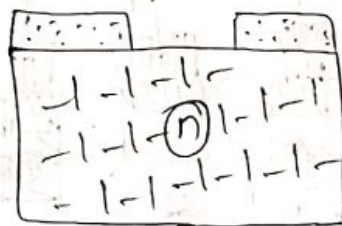


step 3:- A photoresistive layer is grown at the top of oxide layer and it is exposed to uv light through suitable Masking.

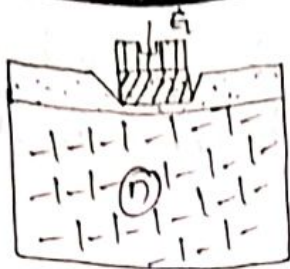


step 4:- The uncovered position of mask allows uv light to flow through it, the oxide layer will get soften and remain covered portion will remain harden.

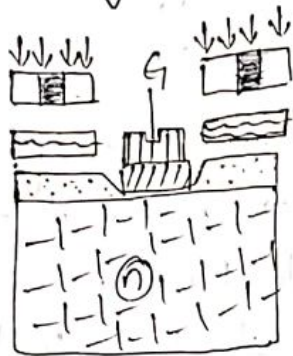
The portion which is soften will be removed and this process is called Etching.



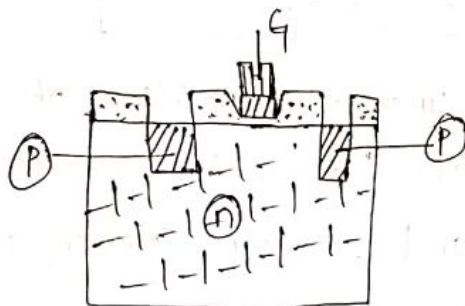
step 5:- An oxide layer of $0.1\mu\text{m}$ thickness is grown and using polysilicon and metal gate terminal can be exerted.



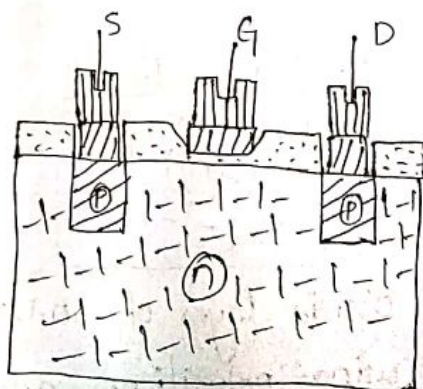
Step 6:- To diffuse p-type of impurities into n-type of substrate - the Masking and etching process are again carried out.



Step 7:- The uncovered portion of Mask will be get soften and there by removed by using "Etching" process. The n-type of impurities are diffused into p-type of substrate as shown in below.



Step 8:- from the p-type of impurities drain and source terminals can be extracted by using polysilicon and metals.



CMOS Inverter:-

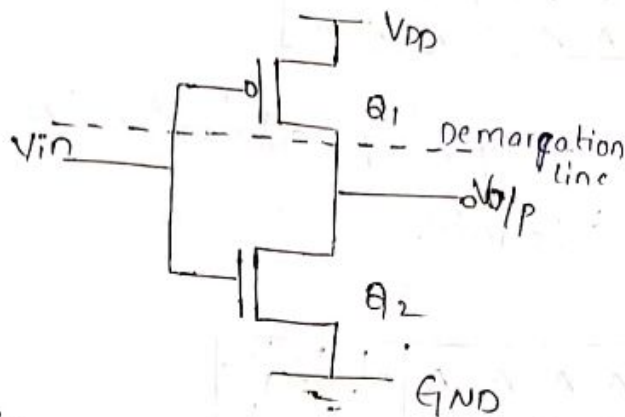
* CMOS is known as complementary metaloxide semiconductor and it will produce output has complementation of input.

* The CMOS can be designed with the help of PMOS and NMOS.

* NMOS transistors are faster than PMOS devices because the Mobility of electrons are greater. Compare to Mobility of holes.

$$\text{i.e., } \mu_n = 2.5 \mu_p$$

CMOS Inverter Circuit:-



V_{in}	Q_1	Q_2	$V_{o/p}$
0	ON	OFF	1
1	OFF	ON	0

operation:-

Case (1):- When V_{in} is zero

When the input is logic 0 then the transistor Q_1 will get ON, Q_2 will get off there by producing V_o as logic '1'.

Case (2):- when V_{in} is logic '1'.

When the Input is logic 1 then Q_1 will get turn off, Q_2 will get turn on, hence the V_o as logic '0'.

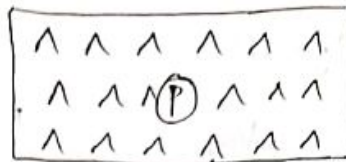
fabrication of CMOS:-

for the fabrication of CMOS we have different types

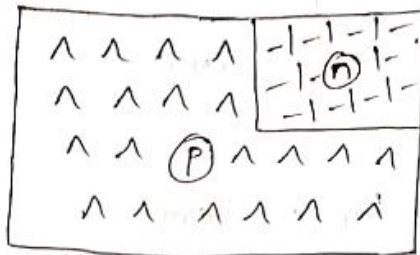
1. CMOS using p-well process
2. CMOS using n-well process
3. Twin-Tub process

CMOS fabrication using n-well process:-

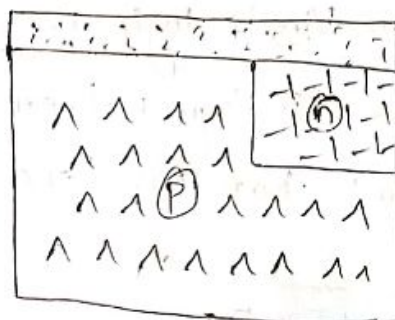
Step 1:- for the designing of CMOS using n-well process we have to consider a p-type of substrate.



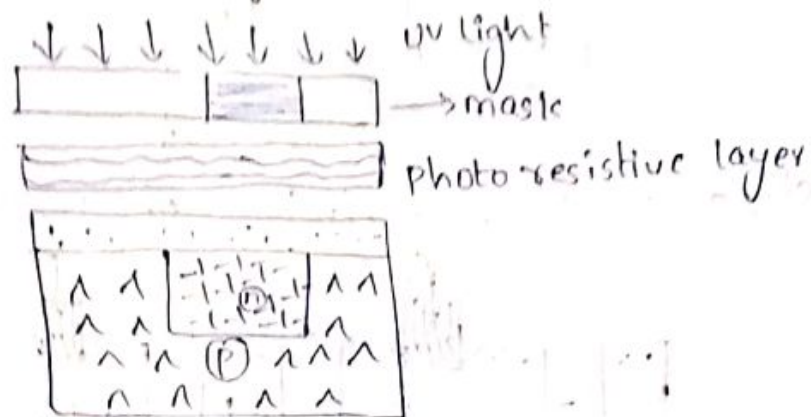
Step 2:- Diffuse N-type of substrate (n-well) into p-type of substrate.



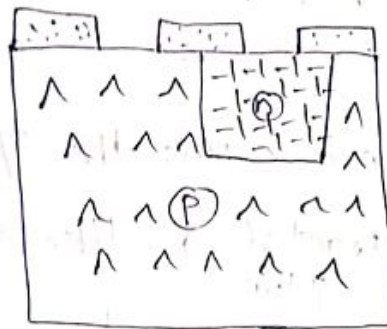
Step 3:- grow oxide layer on the surface of p-type substrate of 1 μ m thickness



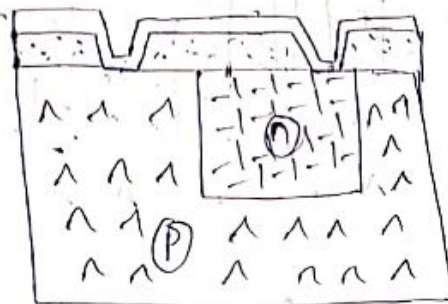
Step 4:- To improve selectiveness, a photoresistive layer is grown and it is exposed to uv light through suitable Masking.



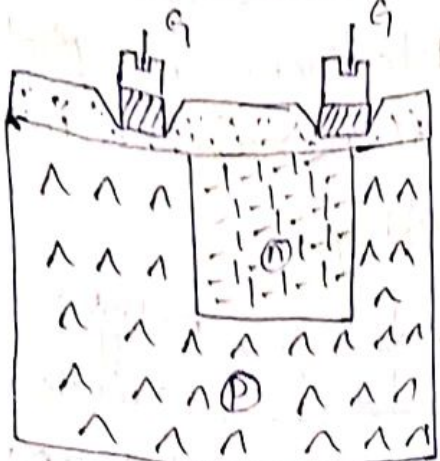
Step 5:- The uncovered portions of mass will get soften and removed by etching process.



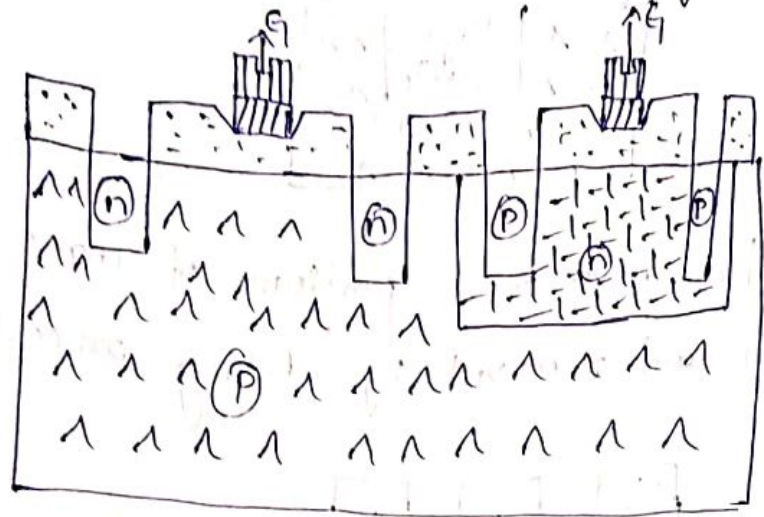
Step 6:- An oxide of 0.1 layer is grown on the top of p-type of Substrate.



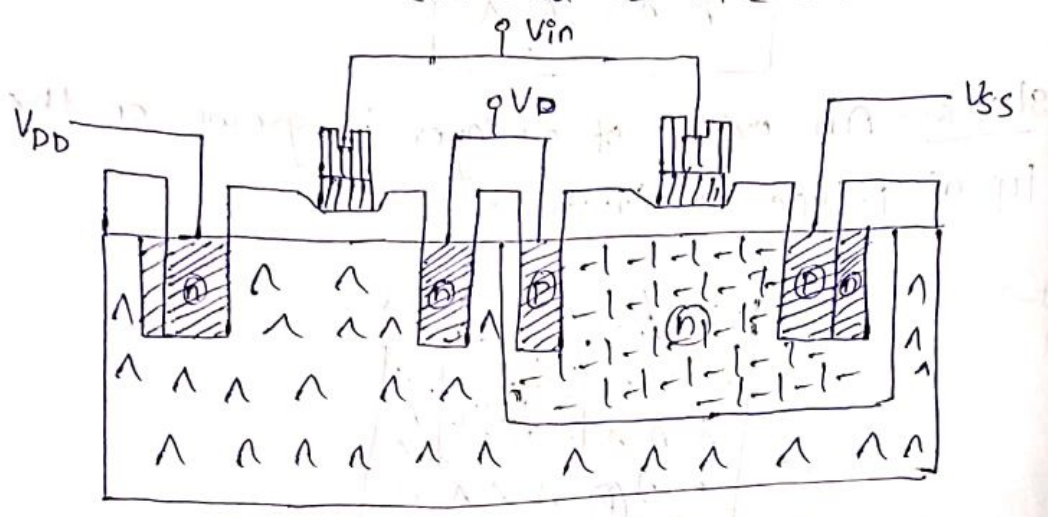
Step 7:- The gate terminals can be extracted from p and n type of substrates using polysilicon and metal.



Steps:- To have diffusions of n-type and p-type

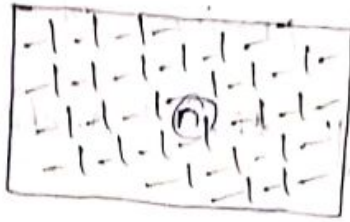


Step 9: finally Further CMOS the i/p and o/p terminals can be extracted as like shown below.



2) CMOS fabrication using p-well process:-

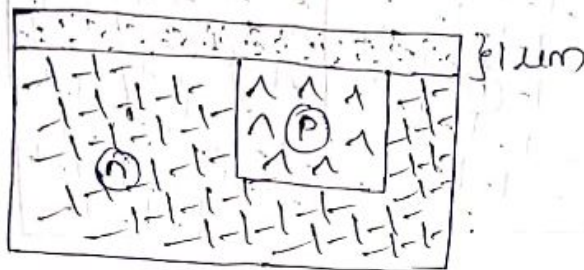
Step 1: For the designing of CMOS using P-well process we have to consider a n-type substrate.



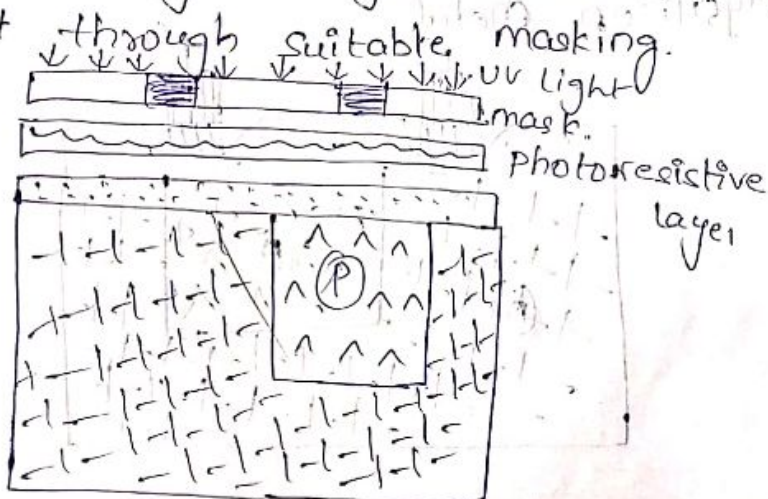
Step 2:- diffuse p-type substrate (p-well) into n-type of substrate.



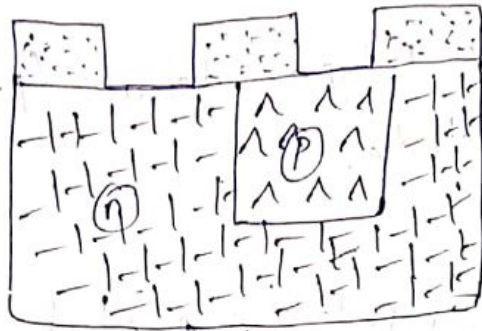
Step 3:- grow oxide layer on the surface of p-type of substrate of 1um thickness.



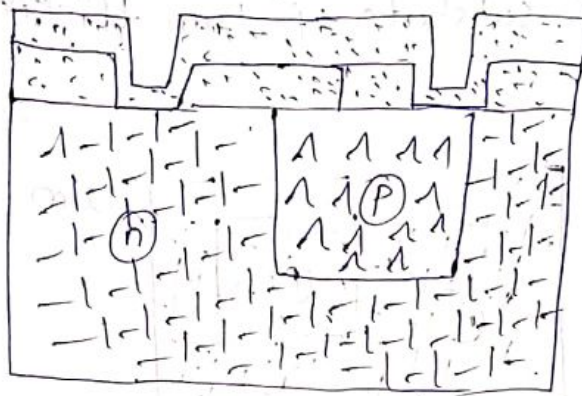
Step 4:- To improve the protective ness a photoresistive layer is grown and it is exposed to uv light through suitable masking.



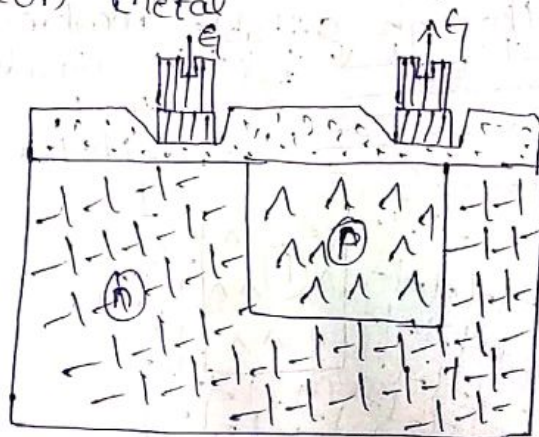
Step 5: The uncovered portions of mask will get softened and removed the "etching" process.



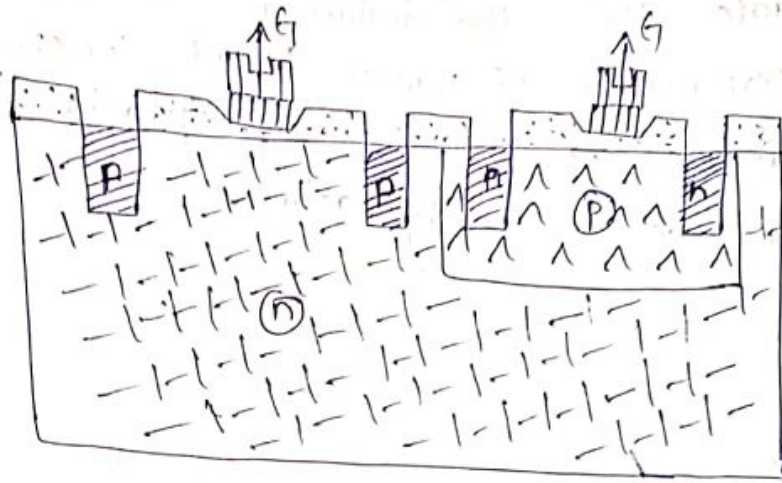
Step 6: An oxide layer of SiO_2 is grown on the top of p-type substrate.



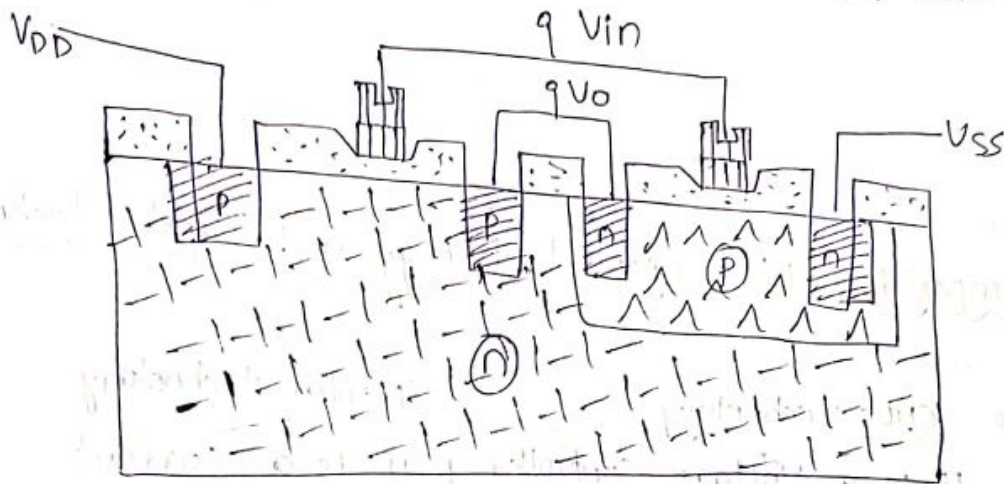
Step 7: The gate terminal can be extracted from p and n-type of the substrate using polysilicon metal.



Step 8:- To have diffusion p-type and n-type into corresponding n & p type of substrates Repeat the steps from (4) to (6).

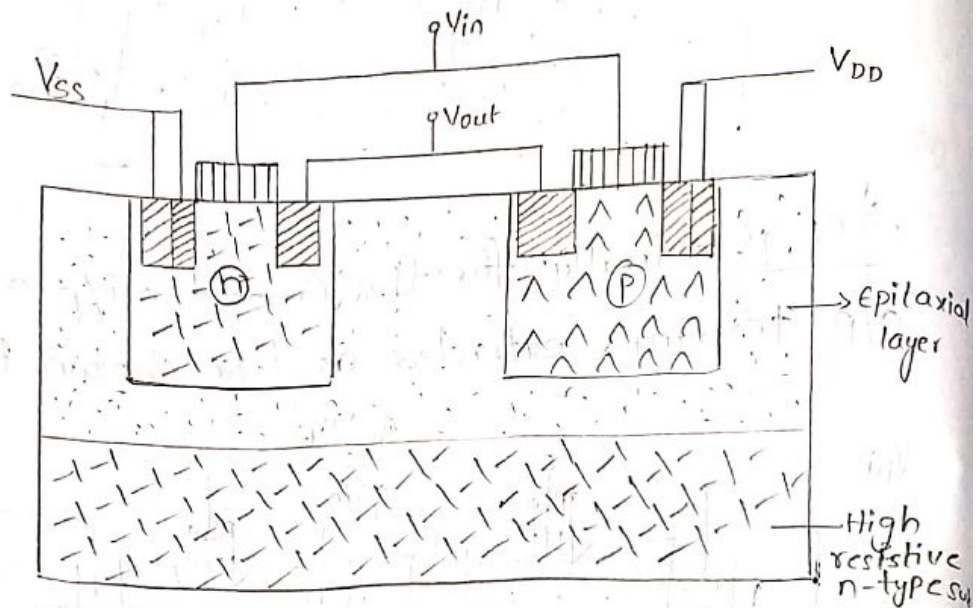


Step 9: finally for the CMOS the i/p and o/p terminals extracted as like shown below.



CMOS fabrication using Twin-Tub process:-

CMOS using twin-tub process is the logical extension of P-well and N-well process. The designing can be carried out by taking a high resistive n-type of substrate. Here the design is considered such that the performance of P-well do not compromise the performance of N-well using Epitaxial layer. Hence doping level is readily achieved.



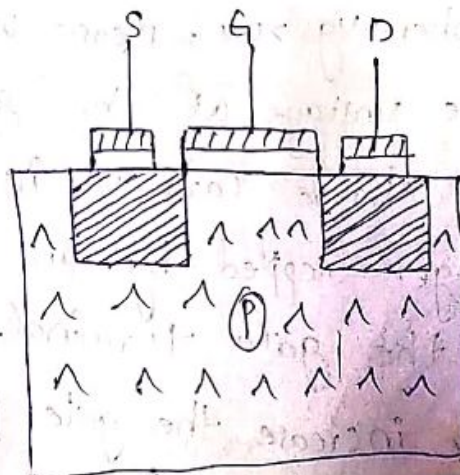
Comparison b/w CMOS technology and Bipolar technology

S.No	CMOS technology	Bipolar technology
1.	It is a Voltage controlled device.	It is a current controlled device.
2.	High input impedance and low output drain current.	Low input impedance and high output drain current.
3.	Low static power dissipation.	High static power dissipation.
4.	Scalable threshold voltage.	Threshold voltage may depend on type of semiconducting material &

5. Bidirectional Capability that is drain and source terminals can be interchange.	These are essentially uni-directional,
6. low transconductance. i.e., g_m is directly proportional to $1/\text{impedance}$.	High transconductance that is $g_m \propto \epsilon V_{in}$
7. High package density	Low package density.
8. High noise margin	Low voltage swing levels

Enhancement mode MOSFET:- [nmos]

While designing N-MOS we have to take p-type of substrate and it has two n-type of diffusions to form drain and source terminals for extracting gate, drain and source terminals polysilicon and metal contacts are use for necessary leads. Here we have to apply a suitable positive voltage at the gate terminal to create channel b/w drain and source



operation of NMOS:-

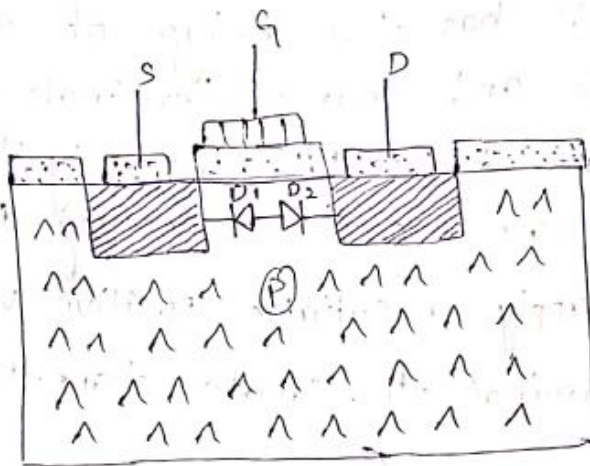
The designing of NMOS includes P-type of substrate and n^+ heavily doped n^+ impurities are diffuse into P-type of substrate to get drain and source terminals.

Case (1):-

When $V_{gs} = 0V$

When $V_{gs} = 0V$ no channel will be form and no current condition takes place.

* Here when $V_{gs} = 0$ we can find two junction diodes that are connected in series back to back manner in b/w drain and source and these two diodes are in reverse bias condition.



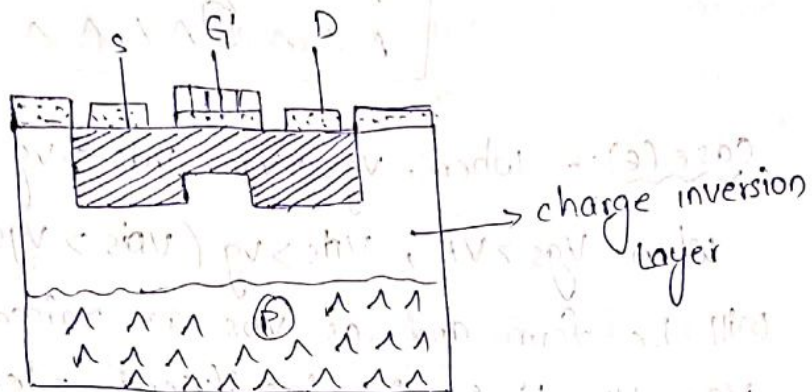
Case (2):- when $V_{gs} > 0V$, means we are applying some possible voltage at the gate terminal then the majority charge carriers in n-mos that is holes will get stepped by an amount of voltage applied at the gate terminal.

* If we increase the gate potential step by step then the holes in the substrate will get

rippled and pushed down leaves a depletion region b/w drain and source. Hence it is called charge inversion layer.

* After having a charge inversion layer the holes in the gate terminal will get attracted to n⁺ diffusions present in drain and source terminals, thereby forming N-channel b/w drain and source.

** Note:- The voltage at which charge inversion layer forms and the gate terminal can be inverted is called threshold voltage of mos device.



Case (3):- when $V_{GS} < V_T$, $V_{DS} = 0$.

No channel will be form and hence no current conduction takes place i.e., $I_{OS} = 0$

Case (4):- $V_{GS} > V_T$, $V_{DS} = 0$

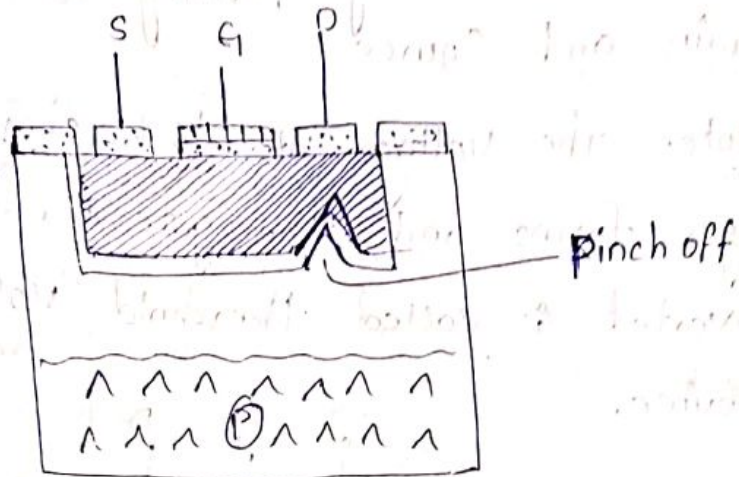
When $V_{GS} > V_T$ then channel will be form but no current conduction takes place i.e., cutoff region.

Active region → current conduction takes place

Saturation Region \rightarrow Acts as constant current source

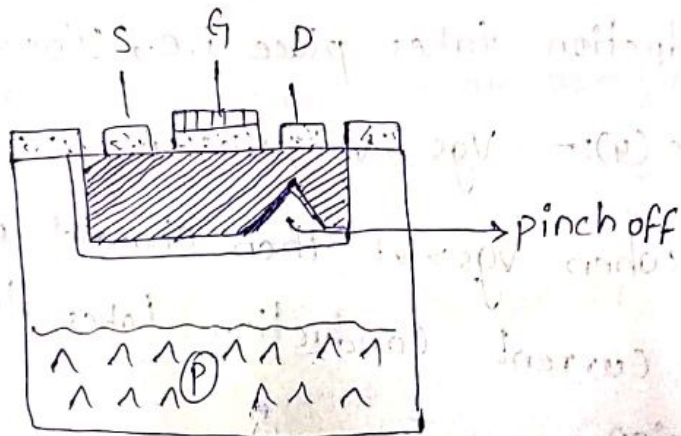
Case (5):- when $v_{gs} > v_t$, $v_{ds} \leq v_g$

When $v_{gs} > v_t$ channel will be form, when $v_{ds} = v_g$ ($v_{ds} = v_{gs} - v_t$) then near drain terminal there is an insufficient electric field and it is in non-saturation condition.



Case (6):- when $v_{gs} > v_t$, $v_{ds} > v_g$

When $v_{gs} > v_t$, $v_{ds} > v_g$ ($v_{ds} > v_{gs} - v_t$) then channel will be form and as v_{ds} is raised greater than $v_{gs} - v_t$ there is insufficient electric field near the drain terminal which causes the channel to pinch off.



Here when $V_{gs} > V_t$, $V_{ds} > V_{gs} - V_t$ it is in saturation mode and hence acts as constant current source.

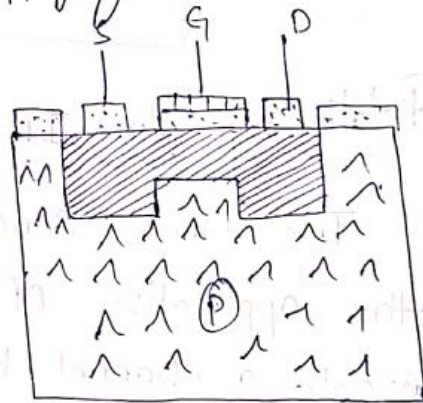
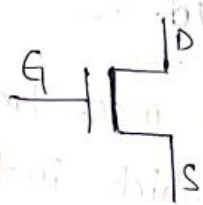
Note:- for an enhancement mode n-mos, the channel will be created by applying ^{suitable} positive voltage at the gate terminal.

* If it is p-mos, to create channel we need to apply negative voltage at the gate terminal.

Depletion Mode N-MOS:-

In Depletion mode MOSFET it is having an inbuilt channel that is no need to apply external voltage.

In this depletion mode n-mos the channel will be created b/w drain and source prior to Manufacturing stage before applying insulating and metals.

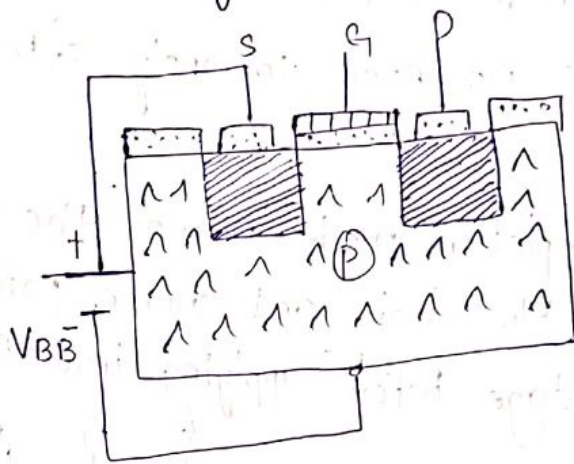


Body Mass effect:-

Note:- Depletion mode n-mos is always on. If you want to remove channel, we need to apply the negative voltage at the gate terminal.

Body Mass Effect:-

for N-mos, basically the source voltage and body potential should be equal that is $V_S = V_B = 0$. If $V_B \neq 0$ i.e., it is having some inbuilt potential then it causes an effect on threshold voltage. hence threshold voltage level is increased to avoid this the body is connected to suitable negative voltage w.r. to source hence this effect is called Body Mass effect.



Relationship b/w I_{DS} vs V_{DS} :-

The whole concept of mass transistor revolves the application of V_{GS} which in turn causes to create a channel b/w drain and source.

\therefore The current I_{DS} is a dependent on both V_{GS} and V_{DS} .

The current I_{ds} can be given by

$$-I_{ds} = -I_{sd}$$

$$\therefore I_{ds} = \frac{Q_c}{\tau_{ds}} \rightarrow (1)$$

where Q_c is charge and

τ_{ds} is electron transit time

we know that $\tau_{ds} = \frac{L}{v} \rightarrow (2)$

where $L =$ length of the channel and

$v =$ velocity

The velocity ' v ' can be given as

$$v = \mu E_{ds} \rightarrow (3)$$

where μ is called mobility constant

$E_{ds} =$ Effective electric field b/w drain and source.

$$E_{ds} = \frac{V_{ds}}{L} \rightarrow (4)$$

Sub (4) in (3)

$$v = \mu E_{ds}$$

$$v = \mu \frac{V_{ds}}{L} \rightarrow (5)$$

Sub (5) in (2)

$$\tau_{ds} = \frac{L}{v}$$

$$\tau_{ds} = \frac{L}{\mu V_{ds}}$$

$$\tau_{ds} = \frac{L^2}{\mu V_{ds}} \rightarrow (6)$$

Sub (6) in (1)

$$I_{ds} = \frac{Q_c}{\tau_{ds}}$$

$$I_{ds} = \frac{Q_c}{\frac{L^2}{\mu V_{ds}}}$$

$$I_{ds} = \frac{Q_c \mu V_{ds}}{L^2}$$

* $\mu_n - 650 \text{ cm}^2/\text{Vsec}$
 $\mu_p - 240 \text{ cm}^2/\text{Vsec}$ at
room temperature

Case (1): - Non-saturation
when it is in non-saturation then the effective
voltage is $\frac{V_{ds}}{2}$.

The charge 'Qc' can be given as

$$Q_c = E_g \epsilon_0 \epsilon_{ins} \omega L \rightarrow (1)$$

where E_g is effective gate voltage

ϵ_0 is permittivity of free space

$$\epsilon_0 = 8.854 \times 10^{-12} \text{ F/m}$$

ϵ_{ins} = relative permittivity

$$\epsilon_{ins} = 4 \text{ (for silicon)}$$

ω = width

L = length

$$\text{we know that } E_g = \frac{[V_g - \frac{V_{ds}}{2}]}{D}$$

where D = oxide thickness

$$\text{wkt } V_g = V_{gs} - V_t$$

$$E_g = \frac{[(V_{gs} - V_t) - \frac{V_{ds}}{2}]}{D} \rightarrow (2)$$

$$Q_c = \frac{\left[(V_{gs} - V_t) - \frac{V_{ds}}{2} \right] \epsilon_0 \epsilon_{ins} \omega L}{D}$$

$$Q_c = \left[(V_{gs} - V_t) - \frac{V_{ds}}{2} \right] \frac{\epsilon_0 \epsilon_{ins} \omega L}{D} \rightarrow (3)$$

WKT

$$I_{ds} = \frac{Q_c \mu V_{ds}}{L^2}$$

Sub eqn (3) in I_{ds} , we get

$$I_{ds} = \left[(V_{gs} - V_t) - \frac{V_{ds}}{2} \right] \frac{\epsilon_0 \epsilon_{ins} \omega \mu V_{ds}}{D L^2}$$

$$I_{ds} = \left[(V_{gs} - V_t) - \frac{V_{ds}}{2} \right] \frac{\epsilon_0 \epsilon_{ins} \mu \omega V_{ds}}{D L}$$

$$I_{ds} = \left[(V_{gs} - V_t) V_{ds} - \frac{V_{ds}^2}{2} \right] \frac{\epsilon_0 \epsilon_{ins} \mu \omega}{D L}$$

$$\text{Let } k = \frac{\epsilon_0 \epsilon_{ins} \mu \omega}{D}$$

$$I_{ds} = \frac{k \omega}{L} \left[(V_{gs} - V_t) V_{ds} - \frac{V_{ds}^2}{2} \right]$$

$$\text{Let } \frac{k \omega}{L} = \beta$$

$$I_{ds} = \beta \left[(V_{gs} - V_t) V_{ds} - \frac{V_{ds}^2}{2} \right]$$

$$C_g = \frac{k \omega L}{\mu} \Rightarrow k = \frac{C_g \mu}{\omega L}$$

C_g = gate to channel capacitance

$$I_{ds} = \frac{C_g \mu}{\cancel{\omega L} L} \left[(V_{gs} - V_t) V_{ds} - \frac{V_{ds}^2}{2} \right]$$

$$I_{ds} = \frac{C_g \mu}{L^2} \left[(V_{gs} - V_t) V_{ds} - \frac{V_{ds}^2}{2} \right]$$

wkt $C_g = C_o \omega L$

$$I_{ds} = \frac{C_o \omega \mu}{L^2} \left[(V_{gs} - V_t) V_{ds} - \frac{V_{ds}^2}{2} \right]$$

Case (2):- Saturation $[V_{ds} = V_{gs} - V_t]$

$$I_{ds} = \frac{k\omega}{L} \left[(V_{gs} - V_t) V_{ds} - \frac{V_{ds}^2}{2} \right] \rightarrow (1)$$

→ Saturation starts at $V_{ds} = V_{gs} - V_t$

$$\rightarrow I_{ds} = \frac{k\omega}{L} \left[V_{ds}^2 - \frac{V_{ds}^2}{2} \right]$$

$$I_{ds} = \frac{k\omega}{L} \left[\frac{V_{ds}^2}{2} \right]$$

$$I_{ds} = \frac{k\omega}{L} \left[\frac{(V_{gs} - V_t)^2}{2} \right]$$

let $\frac{k\omega}{L} = \beta$

$$I_{ds} = \beta \left[\frac{V_{gs} - V_t}{2} \right]^2$$

$$I_{ds} = \frac{k\omega}{L} \left[\frac{(V_{gs} - V_t)^2}{2} \right]$$

$$\text{WKT } C_g = \frac{kWL}{\mu} \Rightarrow k = \frac{C_g \cdot \mu}{WL}$$

$$\Rightarrow I_{ds} = \frac{C_g \mu}{WL} \frac{W}{L} \cdot \left[\frac{(V_{gs} - V_t)^2}{2} \right]$$

$$I_{ds} = \frac{C_g \mu}{L^2} \left[\frac{(V_{gs} - V_t)^2}{2} \right]$$

$$\text{WKT } C_g = C_o WL$$

$$I_{ds} = \frac{C_o W \mu}{L^2} \left[\frac{(V_{gs} - V_t)^2}{2} \right]$$

$$I_{ds} = \frac{C_o W \mu}{L} \left[\frac{(V_{gs} - V_t)^2}{2} \right]$$

Trans Conductance (g_m)

Transconductance is defined as the relationship between output current ' I_{ds} ' and input voltage ' V_{gs} '.

$$\therefore g_m = \frac{\delta I_{ds}}{\delta V_{gs}} \Big|_{V_{ds} = \text{constant}}$$

$$\text{WKT } I_{ds} = \frac{Q_c}{\tau_{ds}}$$

$$\tau_{ds} = \frac{L^2}{\mu V_{ds}}$$

$$I_{ds} = \frac{Q_c \mu V_{ds}}{L^2}$$

$$\delta I_{ds} = \frac{\delta Q_c \mu V_{ds}}{L^2}$$

$$\text{WKT } C_g = \frac{kWL}{\mu} \Rightarrow k = \frac{C_g \cdot \mu}{WL}$$

$$\Rightarrow I_{ds} = \frac{C_g \mu}{WL} \frac{W}{L} \cdot \left[\frac{(V_{gs} - V_t)^2}{2} \right]$$

$$I_{ds} = \frac{C_g \mu}{L^2} \left[\frac{(V_{gs} - V_t)^2}{2} \right]$$

$$\text{WKT } C_g = C_o WL$$

$$I_{ds} = \frac{C_o W \mu}{L^2} \left[\frac{(V_{gs} - V_t)^2}{2} \right]$$

$$I_{ds} = \frac{C_o W \mu}{L} \left[\frac{(V_{gs} - V_t)^2}{2} \right]$$

Trans Conductance (g_m)

Transconductance is defined as the relationship between output current ' I_{ds} ' and input voltage ' V_{gs} '.

$$\therefore g_m = \frac{\delta I_{ds}}{\delta V_{gs}} \Big|_{V_{ds} = \text{constant}}$$

$$\text{WKT } I_{ds} = \frac{Q_c}{\tilde{I}_{ds}}$$

$$\tilde{I}_{ds} = \frac{L^2}{\mu V_{ds}}$$

$$I_{ds} = \frac{Q_c \mu V_{ds}}{L^2}$$

$$\delta I_{ds} = \frac{\delta Q_c \mu V_{ds}}{L^2}$$

WKT

$$C_g = \frac{Q_c}{V_{gs}}$$

$$V_{gs} = \frac{Q_c}{C_g}$$

$$\Delta V_{gs} = \frac{\Delta Q_c}{C_g}$$

$$\begin{aligned} \therefore g_m &= \frac{\Delta I_{ds}}{\Delta V_{gs}} \\ &= \frac{\Delta Q_c \mu V_{ds}}{L^2} \\ &= \frac{\Delta Q_c}{C_g} \end{aligned}$$

$$g_m = \frac{C_g \mu V_{ds}}{L^2}$$

WKT $C_g = C_o \omega L$

$$\therefore g_m = \frac{C_o \omega \mu V_{ds}}{L^2}$$

$$g_m = \frac{C_o \omega \mu V_{ds}}{L}$$

Output Conductance (g_{ds}): -

The output conductance g_{ds} is defined as the relationship between output current I_{ds} and input voltage V_{gs} .

$$g_{ds} = \left. \frac{I_{ds}}{V_{gs}} \right|_{V_{ds} = \text{constant}}$$

WKT $I_{dc} = \frac{Q_c \mu V_{ds}}{L^2}$ and $V_{gs} = \frac{Q_c}{C_g}$

$$g_{ds} = \frac{\frac{Q_c \mu V_{ds}}{L^2}}{\frac{Q_c}{C_g}}$$

$$g_{ds} = C_g \frac{\mu V_{ds}}{L^2}$$

$$g_{ds} \propto \frac{1}{L^2}$$

An increase in the channel length of a device may cause reduce output conductance g_{ds} .

Figure of Merit (ω_0):-

The figure of merit ω_0 is defined as the ratio of transconductance to the gate to channel capacitance (C_g).

$$\omega_0 = \frac{g_m}{C_g}$$

WKT $g_m = \frac{C_g \mu V_{ds}}{L^2}$

$$\omega_0 = \frac{\frac{C_g \mu V_{ds}}{L^2}}{C_g}$$

$$\therefore I_{D0} = \frac{\mu C_{ox} W}{L^2} V_{ds}$$

Since $V_{ds} = V_{gs} - V_T$

$$I_{D0} = \frac{\mu C_{ox} W}{L^2} (V_{gs} - V_T)$$

N-MOS Inverter:-

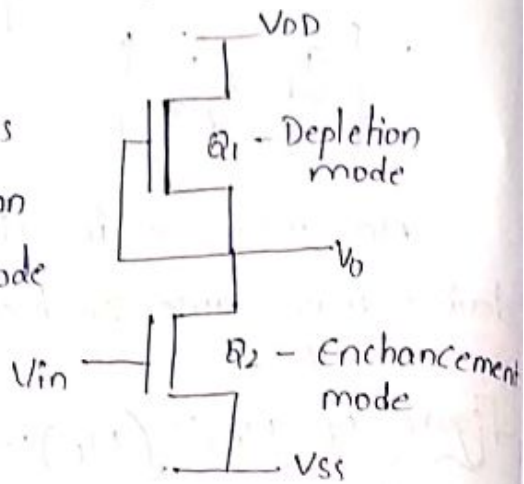
The N-mos inverter is more oftenly use and it can produce full amount of logic levels.

Description:-

* The arrangement of nmos inverter consists of depletion mode and enhancement mode transistors.

* Here the gate terminal of depletion mode nmos is connected to the drain terminal of enhancement mode nmos.

* The depletion mode transistor is always on because of the inbuilt channel.



operation:-

Case (1):-

when V_{in} is logic '1'

When Input is logic 1 the transistor Q_2 turns ON and

transistor Q_1 ^{is always} turns ON because it is depletion mode

Truth Table:-

V_{in}	Q_1	Q_2	V_o
0	ON	OFF	1
1	ON	ON	0

then V_{out} is logic 0.

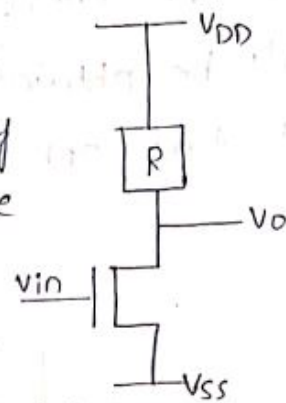
Case (2):-

When V_1 is logic '0'

When input is logic '0' then the transistor Q_2 remains off and Q_1 is on hence the o/p voltage is logic '1'.

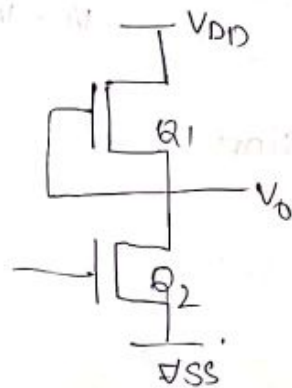
Alternative forms of pull-up:-

1. Resistive pull-up:- The Resistive pull-up configuration is not oftenly used while designing silicon substrate because high resistive values are not incooperated in that silicon substrate.



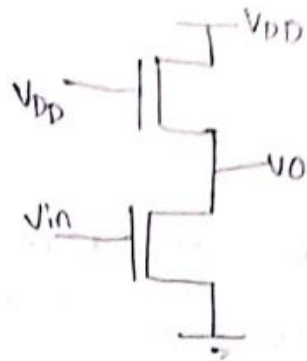
2. Depletion mode pull-up:-

It can produce High power dissipation and there may be a current flow in b/w supply rails when V_{in} is logic '1'.



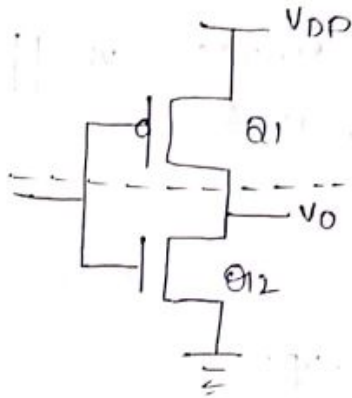
3. Enhancement mode pull-up:-

It produces high power dissipation b/w supply rails and conduction starts when $V_{DD} = V_{gg}$



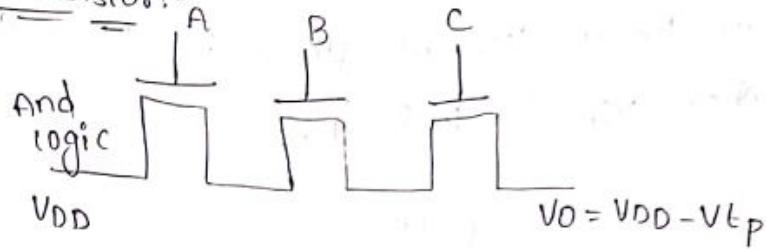
4. Complementary pass pull-up:-

In this configuration full-amount of logic level can't be obtained and only one transistor will get turn on either for logic '0' or '1'.



Pass transistor:-

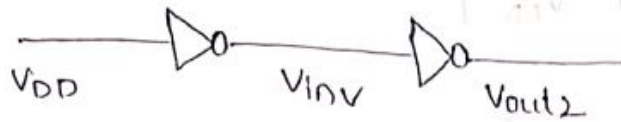
Eg:-



* unlike Bipolar transistors

pullup to pulldown ratio of nmos-inverter driven by another nmos inverter:-

The arrangement of pullup to pulldown ratio for one nmos inverter driven by another nmos is shown as below.



WKT

I_{ds} for saturation mode

$$I_{ds} = \frac{k_w}{L} \left[\frac{(V_{gs} - V_t)}{2} \right]^2$$

for depletion mode, $v_{gs} = 0$

$$I_{ds} = k \frac{w_{pu}}{L_{pu}} \left[\frac{(-V_{td})}{2} \right]^2 \rightarrow (1)$$

I_{ds} for enhancement mode, $V_{gs} = V_{inv}$

$$I_{ds} = \frac{k w_{pd}}{L_{pd}} \left[\frac{(V_{inv} - V_t)}{2} \right]^2 \rightarrow (2)$$

equating (1) and (2), (1) = (2) we get

$$\frac{k w_{pu}}{L_{pu}} \frac{(-V_{td})^2}{2} = \frac{k w_{pd}}{L_{pd}} \frac{(V_{inv} - V_t)^2}{2}$$

$$\frac{1}{z_{pu}} (-V_{td})^2 = \frac{1}{z_{pd}} (V_{inv} - V_t)^2$$

$$(-V_{td})^2 = \frac{z_{pu}}{z_{pd}} (V_{inv} - V_t)^2$$

$$\frac{z_{pu}}{z_{pd}} = \frac{(-V_{td})^2}{(V_{inv} - V_t)^2}$$

$$\frac{Z_{pu}}{Z_{pd}} = \frac{(0.6V_{DD})^2}{(0.5V_{DD} - 0.2V_{DD})^2}$$

$$= \left(\frac{0.6V_{DD}}{0.3V_{DD}} \right)^2$$

$$\frac{Z_{pu}}{Z_{pd}} = \frac{4}{1}$$

$$\frac{Z_{pu}}{Z_{pd}} = 4:1$$

$$V_{td} = 0.6V_{DD}$$

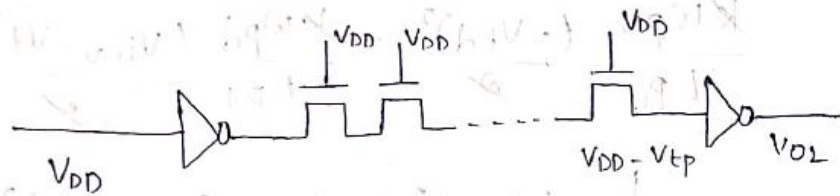
$$V_t = 0.2V_{DD}$$

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

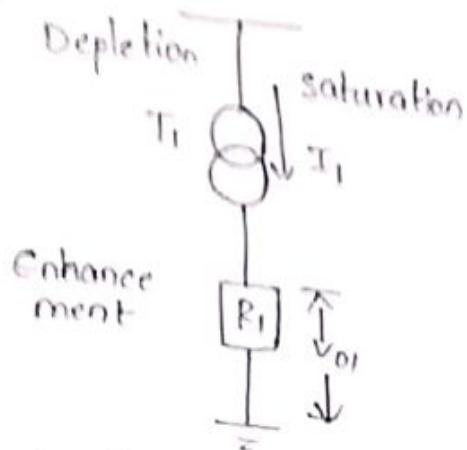
∴ The pullup to pulldown ratio of nmos inverter driven by another nmos is 4:1

pullup to pulldown ratio of nmos driven by another nmos with one or more pass transistors:-

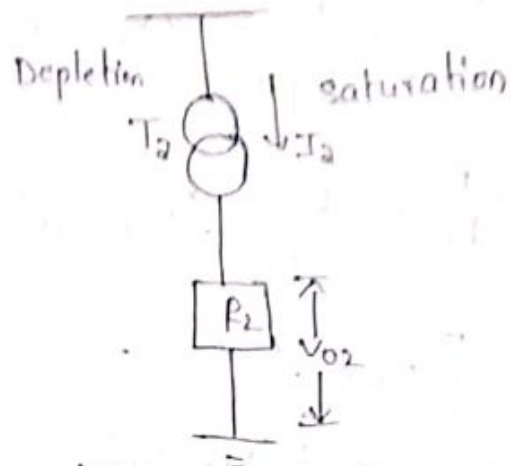
The arrangement of pullup to pulldown ratio for nmos driven by another nmos with one or more pass transistors is depicted as below.



* when the output of inverter 1 is passed through series of pass transistors then full logic levels are not obtain due to threshold voltage of pass transistors i.e., $V_{inv2} = V_{DD} - V_{tp}$



(a) inv_1 with i/p as V_{DD}



(b) inv_2 with i/p as $V_{DD} - V_{tp}$

for inverters

I_{ds} for saturation mode:

$$I_{ds} = \frac{kw}{L} \frac{(v_{gs} - V_t)^2}{2}$$

for depletion mode, $v_{gs} = 0$

$$I_{ds1} = \frac{kw_{p1}}{L_{p1}} \frac{(-V_{td})^2}{2}$$

$$I_1 = \frac{K}{Z_{p1}} \frac{(-V_{td})^2}{2}$$

for non-saturation, I_{ds} can be similar

$$I_{ds} = \frac{kw}{L} \left[(v_{gs} - V_t) V_{ds} - \frac{V_{ds}^2}{2} \right]$$

for enhancement mode, $v_{gs} = V_{DD}$

$$I_{ds1} = \frac{kw_{p1}}{L_{p1}} \left[(V_{DD} - V_t) V_{ds1} - \frac{V_{ds1}^2}{2} \right]$$

$$\frac{I_{ds1}}{V_{ds1}} = \frac{kw_{p1}}{L_{p1}} \left[(V_{DD} - V_t) - \frac{V_{ds1}}{2} \right]$$

neglecting

$$\frac{I_{ds1}}{V_{ds1}} = \frac{k}{Z_{pd1}} (V_{DD} - V_t)$$

$$\frac{1}{R_1} = \frac{k}{Z_{pd1}} (V_{DD} - V_t)$$

$$R_1 = \frac{Z_{pd1}}{k(V_{DD} - V_t)}$$

$$V_{o1} = I_1 R_1$$

$$V_{o1} = \frac{k}{Z_{pu1}} \frac{(-V_{td})^2}{2} \times \frac{Z_{pd1}}{k(V_{DD} - V_t)}$$

$$V_{o1} = \frac{Z_{pd1}}{Z_{pu1}} \frac{(-V_{td})^2}{2(V_{DD} - V_t)}$$

For inverter 2

For saturation mode, I_{ds} can be seen by

$$I_{ds} = \frac{k_{n0}}{L} \frac{(V_{gs} - V_t)^2}{2}$$

For depletion mode, $V_{gs} = 0$

$$I_{ds2} = \frac{k_{np02}}{L_{pu2}} \frac{(-V_{td})^2}{2}$$

$$I_2 = \frac{k}{Z_{pu2}} \frac{(-V_{td})^2}{2}$$

I_{ds} in non-saturation can be given by

$$I_{ds} = \frac{k_{n0}}{L} \left[(V_{gs} - V_t)V_{ds} - \frac{V_{ds}^2}{2} \right]$$

for enhancement mode, $v_{gs} = V_{DD} - V_{tp}$

$$I_{ds2} = \frac{k_{n0} \mu_{n2}}{L_{p2}} \left[(v_{gs} - v_t) v_{ds2} - \frac{v_{ds2}^2}{2} \right]$$

$$\frac{I_{ds2}}{v_{ds2}} = \frac{k}{2 \mu_{n2}} \left[(v_{gs} - v_t) - \frac{v_{ds2}}{2} \right]$$

neglecting

$$\frac{1}{R_2} = \frac{k}{2 \mu_{n2}} [v_{gs} - v_t]$$

$$\therefore \boxed{v_{gs} = V_{DD} - V_{tp}}$$

$$\frac{1}{R_2} = \frac{k}{2 \mu_{n2}} (V_{DD} - V_{tp} - v_t)$$

$$\boxed{R_2 = \frac{2 \mu_{n2}}{k (V_{DD} - V_{tp} - v_t)}}$$

$$V_{O2} = I_2 R_2$$

$$= \frac{k'}{2 \mu_{p2}} \frac{(-v_{td})^2}{2} \cdot \frac{2 \mu_{n2}}{k (V_{DD} - V_{tp} - v_t)}$$

$$\boxed{V_{O2} = \frac{\mu_{n2}}{\mu_{p2}} \frac{(-v_{td})^2}{2 (V_{DD} - V_{tp} - v_t)}}$$

$$V_{O1} = \frac{\mu_{p1}}{2 \mu_{p1}} \frac{(-v_{td})^2}{2 (V_{DD} - v_t)}$$

$$\boxed{V_{O1} = V_{O2}}$$

$$\frac{\mu_{p1}}{2 \mu_{p1}} \frac{(-v_{td})^2}{2 (V_{DD} - v_t)} = \frac{\mu_{n2}}{2 \mu_{p2}} \frac{(-v_{td})^2}{2 (V_{DD} - V_{tp} - v_t)}$$

$$\frac{Z_{pd1}}{Z_{pu1}(V_{DD}-V_t)} = \frac{Z_{pd2}}{Z_{pu2}(V_{DD}-V_{tp}-V_t)}$$

$$\frac{Z_{pd2}}{Z_{pu2}} = \frac{Z_{pd1}(V_{DD}-V_{tp}-V_t)}{Z_{pu1}(V_{DD}-V_t)}$$

$$\frac{Z_{pu2}}{Z_{pd2}} = \frac{Z_{pu1}(V_{DD}-V_t)}{Z_{pd1}(V_{DD}-V_{tp}-V_t)}$$

$$\frac{Z_{pu2}}{Z_{pd2}} = \frac{4}{1} \times \frac{V_{DD}-0.2V_{DD}}{(V_{DD}-0.3V_{DD}-0.2V_{DD})}$$

$V_t = 0.2V_{DD}$
 $V_{tp} \approx 0.3V_{DD}$

$$= \frac{4}{1} \times \frac{0.8V_{DD}}{0.5V_{DD}}$$

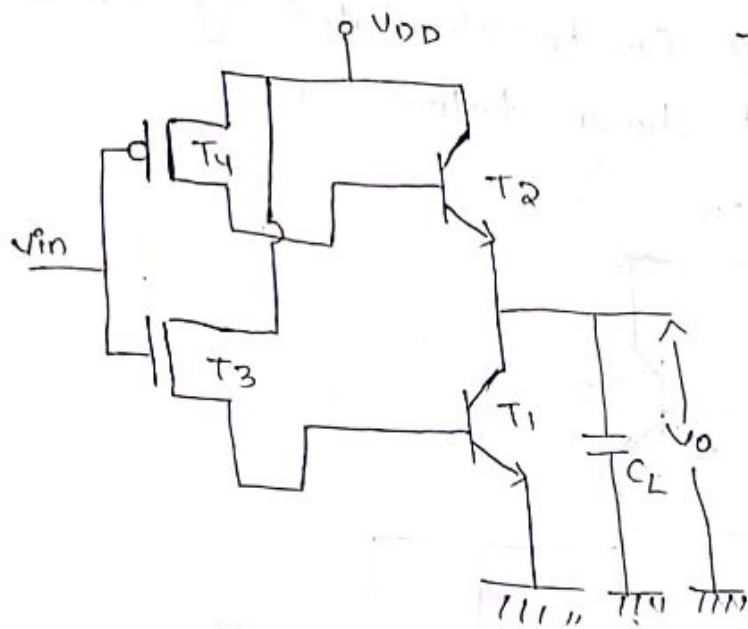
$$\approx \frac{4}{1} \times \frac{8}{5}$$

$$\frac{Z_{pu2}}{Z_{pd2}} \approx \frac{8}{1}$$

∴ The pullup to pulldown ratio of nmos driven by another nmos with one or more pass transistors is 8:1.

Bicmos Inverter:-

To get logical switching of mos transistor, bipolar transistors are attach at their ends.



Truth Table

V_{in}	T_1	T_2	T_3	T_4	V_o
0	OFF	ON	OFF	ON	1
1	ON	OFF	ON	OFF	0

BiCMOS Inverter

operation:-

Case (1):- when $V_{in} = \text{logic '0'}$.

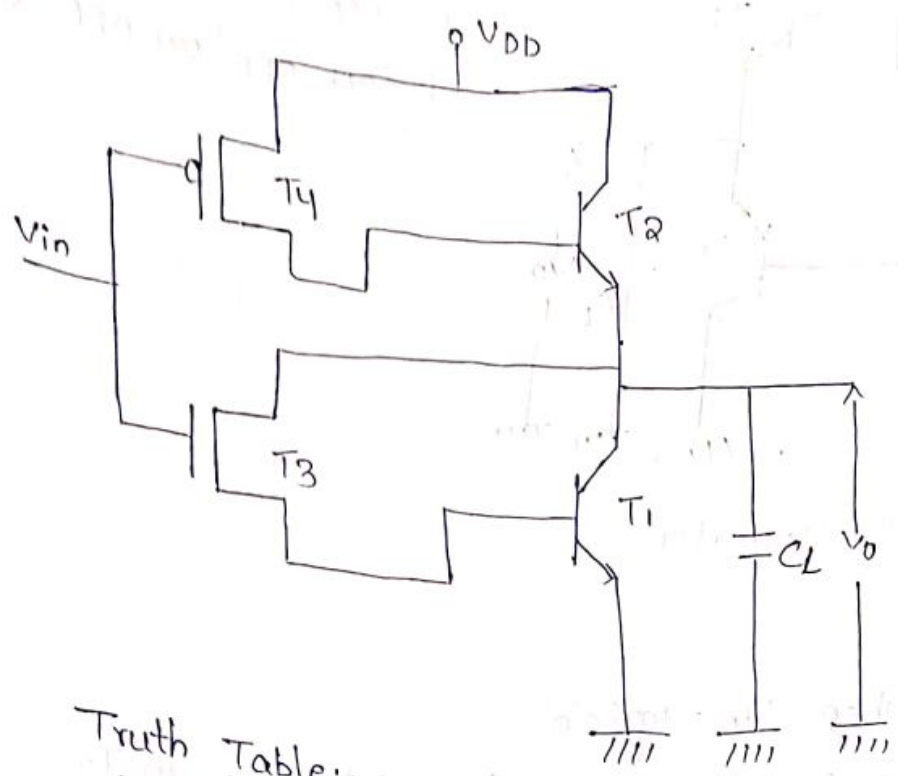
when V_{in} is logic '0' but transistor T_4 or T_2 will get turn ON and transistor T_3 and T_1 will get turn OFF. then the capacitor C_L gets charge hence output is logic 1.

Case (2):- when V_{in} is logic '1'.

when V_{in} is logic '1' then T_3 and T_1 will get ON and T_2 and T_4 will get OFF. Here that capacitor C_L gets discharge hence output is logic '0'.

* for logic '1' T_1 and T_3 will get ON and their by it forms a short circuit path between V_{DD} and V_{SS} which in turn cause static power dissipation and this will slowdown transistor action.

* The above drawback can be eliminated by the modified arrangement shown below.



Truth Table:- Alternative BiCMOS with no static power dissipation

V_{in}	T_1	T_2	T_3	T_4	V_o
0	OFF	ON	OFF	ON	1
1	ON	OFF	ON	OFF	0

* In this circuit for logic '1' there is no short ckt path between V_{DD} and V_{SS} , means there is no static power dissipation.

* Hence output swing is reduce.

Since output voltage can't fall below the base emitter of T_2 that is V_{BE} of T_2 and this can be overcome by another improved circuit shown below.

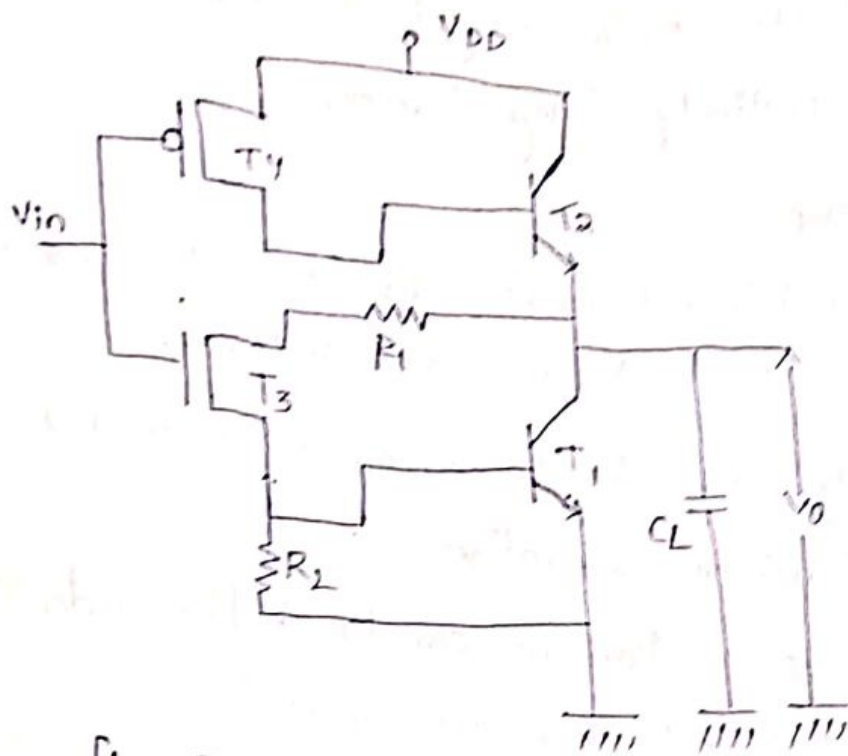


fig: Improved BJT inverter with Better logic levels

* In this arrangement the resistors provide the improved swing of output voltages when BJT's are off and also provide discharge path for Base Currents during turn off.

* However the provision of onchip Resistors of suitable value is not always convenient and maybe space consuming so the above circuit may be further modified as shown below.

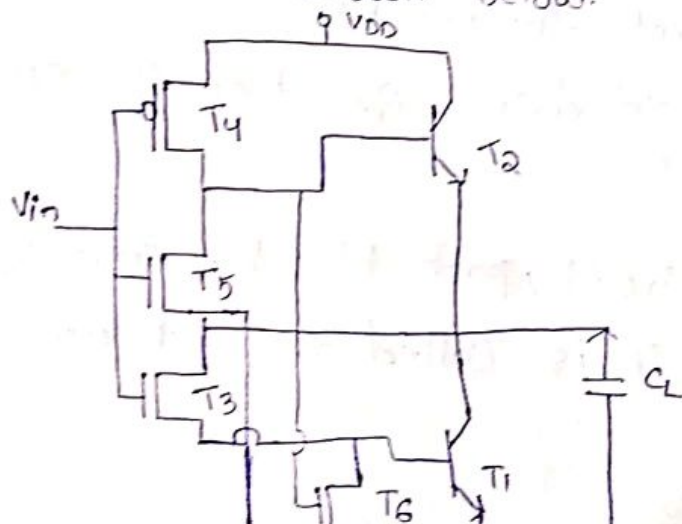


fig: Improve CMOS Inverter using mos transistors for base current drive

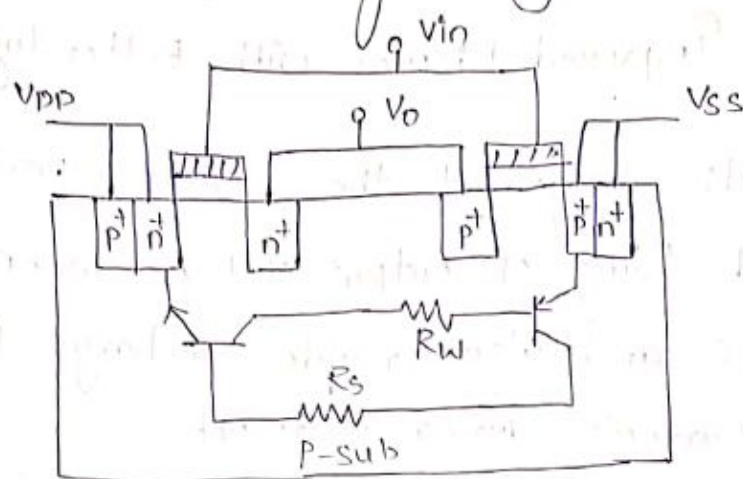
The Transistors T_5, T_6 will get -turn ON when T_1 and T_2 respectively being turn off.

Latch up in CMOS-

* Latch up is a inherent problem in CMOS, that provides a low impedance path between V_{DD} and V_{SS} .

* Latch up may arise due to noise, switch ON and OFF or by Incident radiation.

* The latch up mechanism can be better understood with the following arrangement.

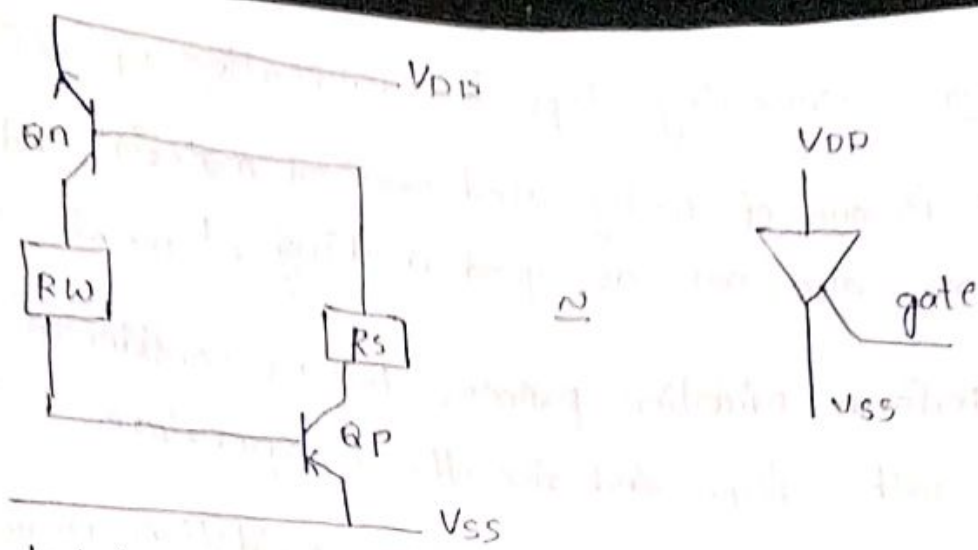


Latch up effecting n-well substrate

* In the above figure, if sufficient substrate current flows for Q_p , then Q_p will turn ON & it draws some current through R_s .

* If it is enough to drive Q_n then it will also turn ON.

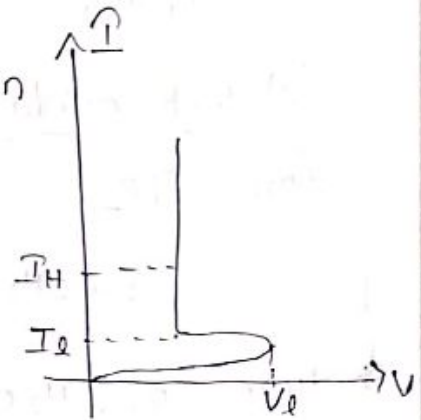
Hence a short circuit path b/w V_{DD} and V_{SS} is obtained and it is called as latch up problem.



Latchup circuit model

* The swinging characteristics of the arrangement are shown as below.

* Once it is latch, this condition will be maintain untill latch up current drops below I_L



Remidies for latch up:-

* Introduction of guard rings can eliminate latch up problem.

* Increase in the substrate doping levels, with a constitute drop in the value of R_s .

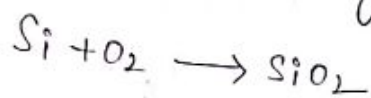
* Reduce R_w by control of fabrication parameters, and by ensuring low contact resistance to V_{SS} .

latch up $V-I$ characteristics of SCR.

* Oxidation:- Processing steps for fabrication of ICs.
Silicon is one of mostly used oxidised material and which may also acts as good masking element.

* To perform oxidation process let us consider a furnace with silicon and rise the temperature.

* Oxidation is two types 1) Dry oxidation: Here the silicon is reacting with O_2 to form SiO_2 .



2) Wet oxidation: Silicon reacts with H_2O to form SiO_2 .



* Here O_2 , H_2O are called oxidants that are used to oxidise silicon.

* Ion implantation:-

ion implantation process is used to diffuse the dopants into a specified material (or) substrate.

* Here the dopant is to be diffused into a substrate material with a sufficient energy.

* By the strength of the dopant it penetrates through the substrate and may cause some effect on lattice atom.

Nuclear stopping:- when the dopant is injected into the substrate, depending upon the strength the dopant

may change the position of lattice atom and may damage the lattice atom. If the strength is further more increase. Hence it is called Nuclear stopping.

electronic
* During ion implantation process, if the dopants change the position of the lattice atom and it shows no damage of lattice atom. Hence it is called electronic stopping.

* photolithography (or) lithography: -
photolithography is ^{a process that is} used to diffuse dopants into substrate in a selected position through masking element.

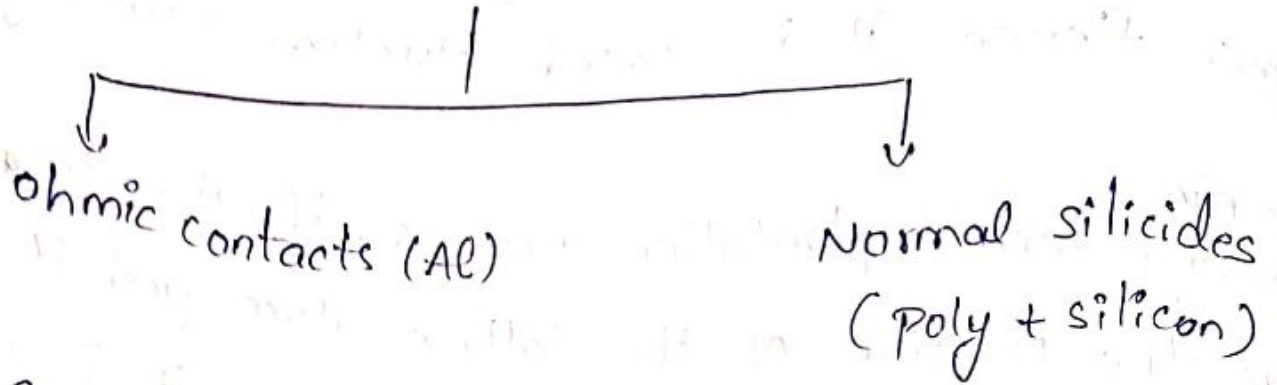
i.e., Lithography (or) photolithography is the process of transferring geometrical patterns from masking element to silicon.

* In olden days, we don't have photolithography technology then a mask of suitable pattern is transfer into silicon using a litho (stone).

* Metallization: -

Metallization is the process that is use to extract terminals from the device. these terminals are used to have contact to the outside world that is means to measure output (throughput).

Metalization



* Encapsulation:-

Encapsulation is the process that is used after manufacturing of the device.

Encapsulation provides protection to the whole body (or) package.

UNIT-II Mass and BiCMOS circuit designing processes

Introduction:-

To design any circuit first of all we need to take the design considerations that is the specifications of the component of the system.

All those individual components are to be interconnected and metalizations are used to have a package circuit.

MOS layers:-

The basic mos layers that are used to design a circuit includes n^+ diffusions, p^+ diffusions, poly-silicon and etc.

Stick diagrams:-

Stick diagrams are outlined representations of layouts.

The stick diagrams convey the designing of the circuit through colour codes.

* For the designing of stick diagrams and layouts the MOS layers using colour codes are drawn as below.

For nmos

<u>S.No</u>	<u>layer</u>	<u>color</u>	<u>stick diagram representation</u>	<u>layout representation</u>
1.	n ⁺ diffusion	Green		
2.	p ⁺ diffusion	Yellow		
3.	Polysilicon	Red		
4.	Metal	Blue		
5.	Ion implantation	Yellow		
6.	Buried contact	Brown	•	
7.	Contact cut	Black	•	
8.	Supply lines	Black	X	
9.	Demarcation line	Brown	---	

For pmos:-

For pmos designing the only change is ion implantation.

<u>S.No</u>	<u>layer</u>	<u>color</u>	<u>stick diagram representation</u>	<u>layout representation</u>
1.	Ion Implantation	green		

Mos transistor representations:-

NMOS Enhancement

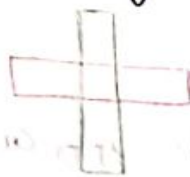
PMOS Enhancement

stick diagram

layout

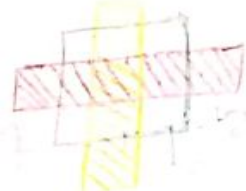
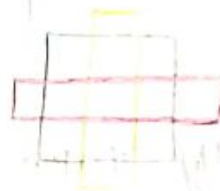
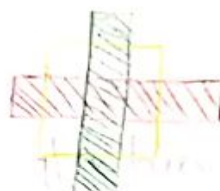
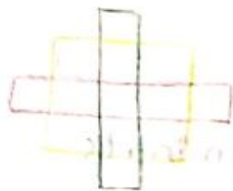
stick diagram

layout



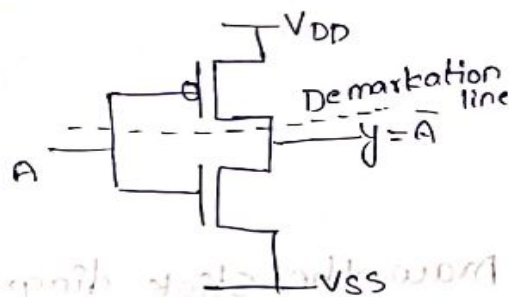
NMOS depletion mode
stick diagram · layout

PMOS depletion mode
stick diagram layout

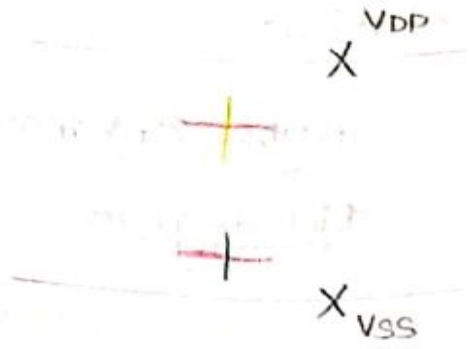


1. Design a stick diagram for CMOS inverter:-

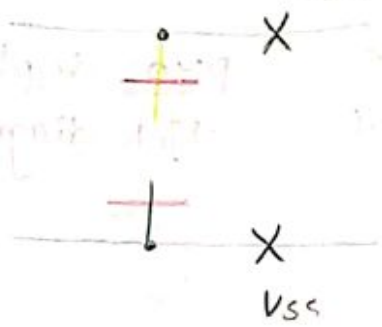
Step 1:- To design a CMOS inverter, first we have to draw the supply lines V_{DD} and V_{SS}



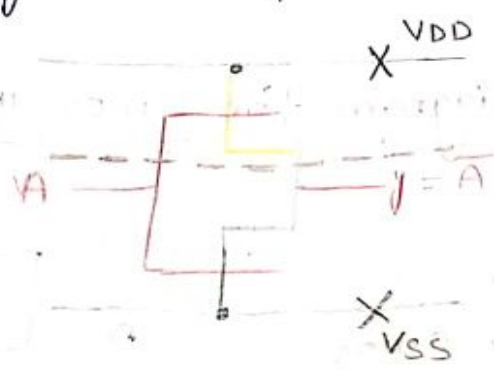
Step 2:- After V_{DD} and V_{SS} place the required transistors at a given place.



Step 3:- extend the transistors towards V_{DD} and V_{SS} and make contact cuts V_{DD}

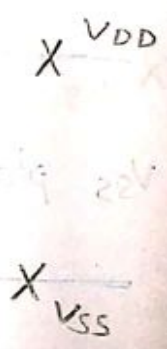
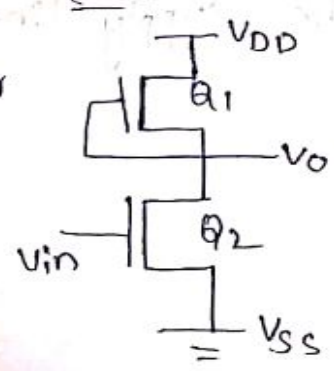


Step 4:- finally take i/p and o/p terminals

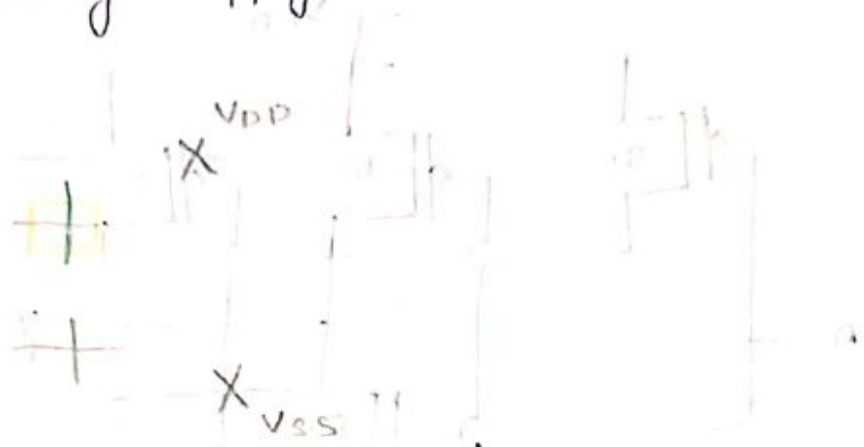


Q. Draw the stick diagram for nmos inverter:-

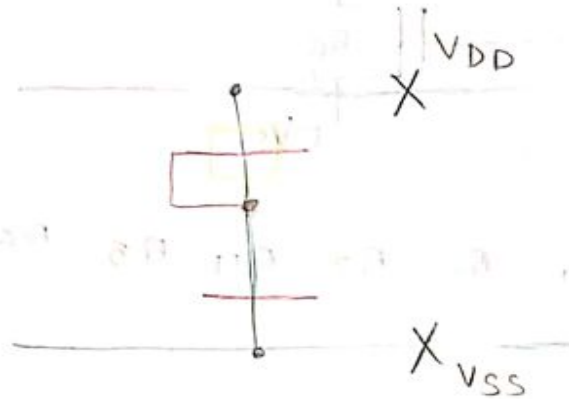
Step 1: To draw stick diagram for nmos inverter first we have to draw supply lines V_{DD} & V_{SS} .



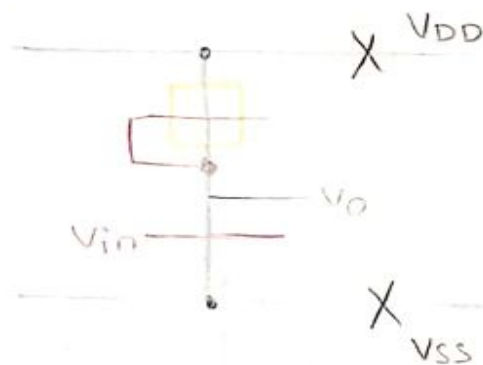
Step 2:- After drawing supply lines place the required transistors.



Step 3:- extend the transistor lines toward VDD and VSS and place contacts and Buried Contact.



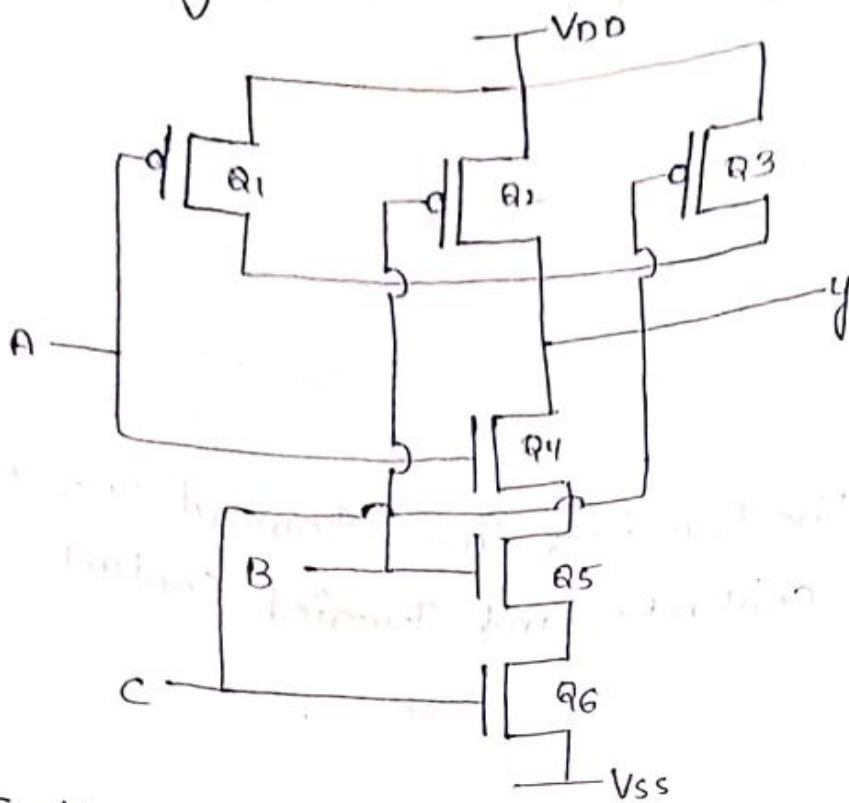
Step 4:- finally take i/p and o/p's terminals.



3. Draw stick diagram for three input NAND gate using CMOS technology.

	Nmos	pmos
NAND	Series	parallel
NOR	parallel	Series

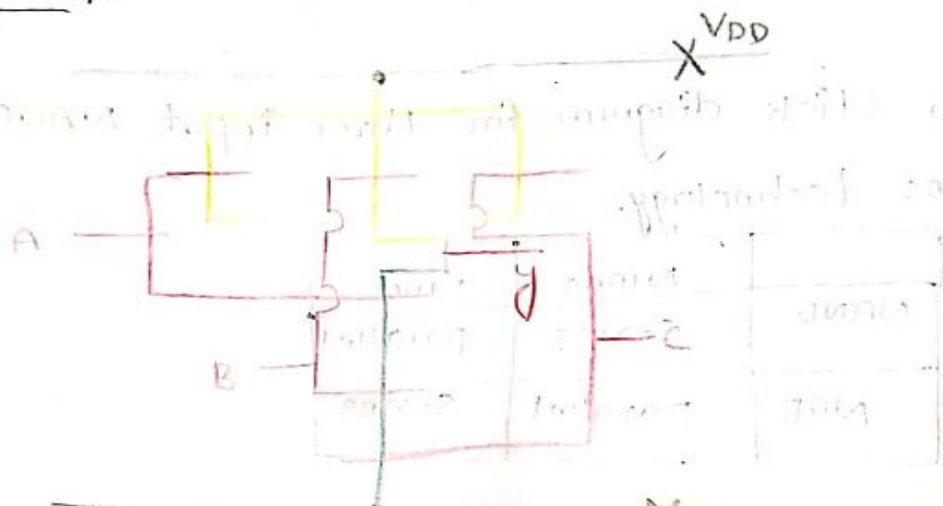
Circuit diagram



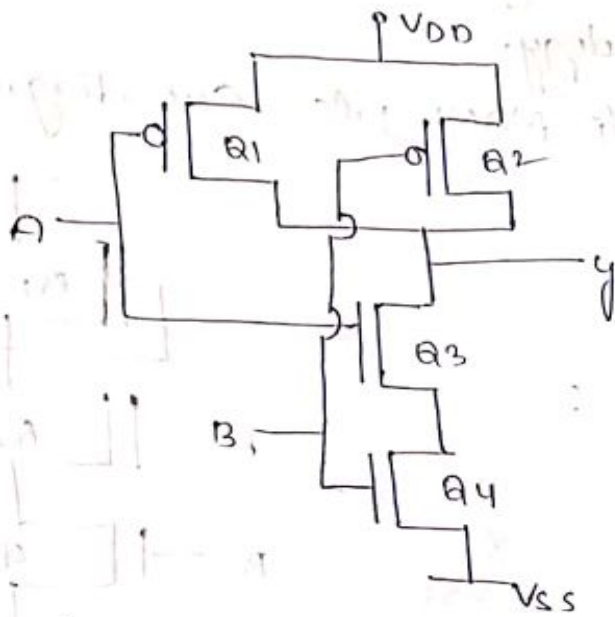
Truth Table

A	B	C	Q1	Q2	Q3	Q4	Q5	Q6	y
0	0	0							0
0	0	1							0
0	1	0							0
0	1	1							0
1	0	0							1
1	0	1							1
1	1	0							1
1	1	1							1

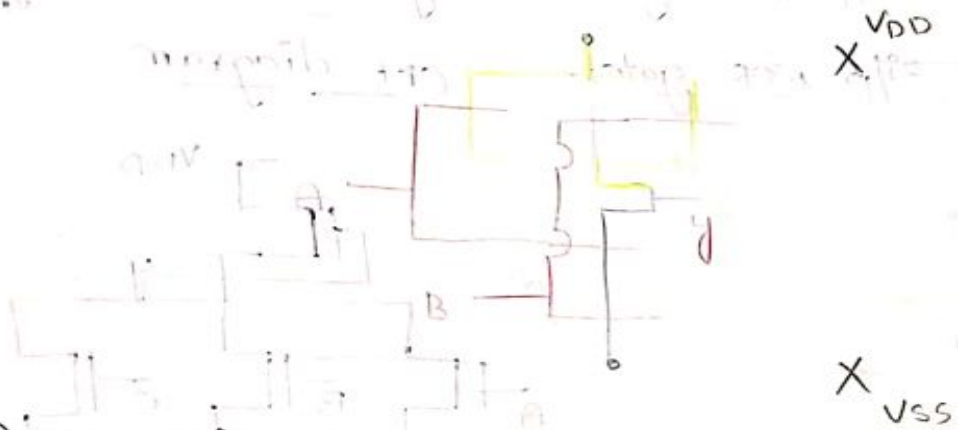
stick diagram:-



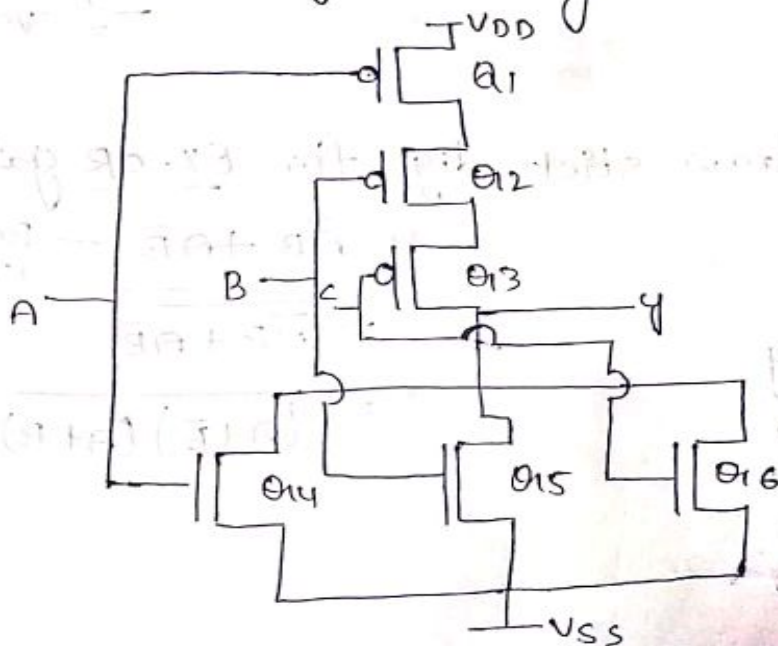
4. Draw the 2 i/p NAND gate using CMOS technology



stick diagram:-

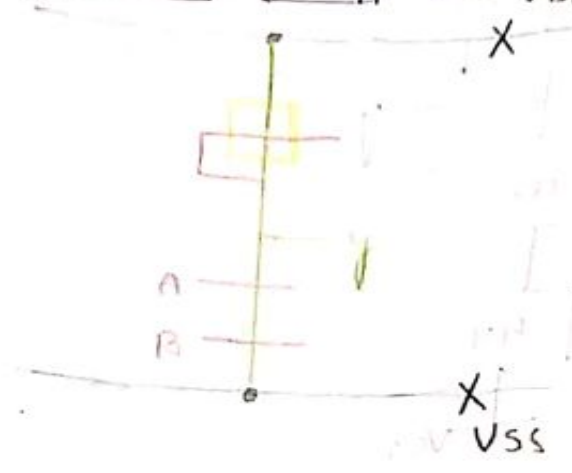


5) Draw 3 i/p NOR gate using CMOS technology

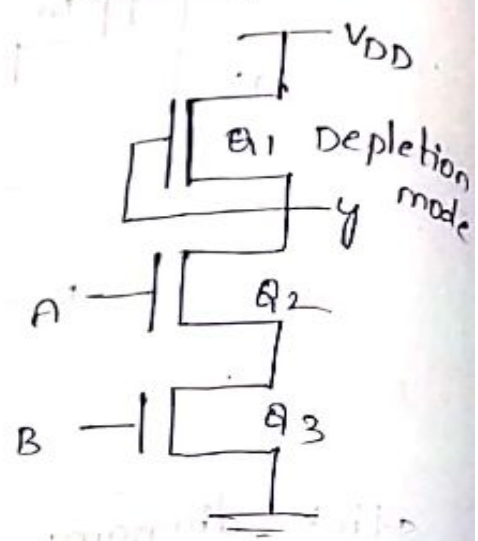


6) Draw the stick diagram for 2 i/p NAND gate using nmos technology.

stick diagram for 2 i/p NAND gate



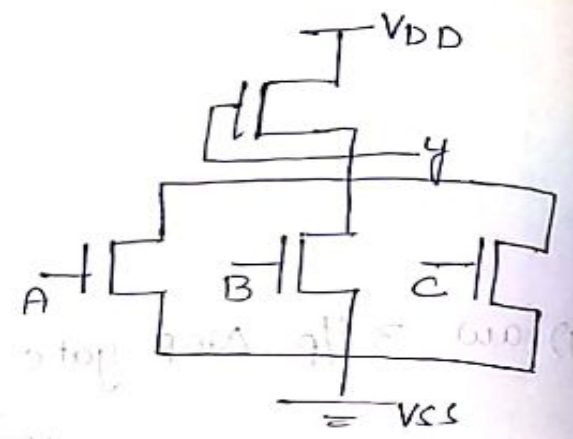
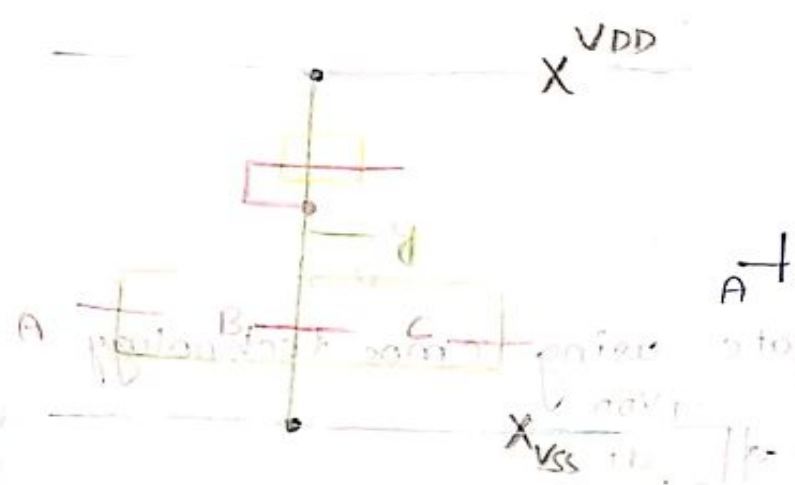
Ckt diagram



7) Draw the 3 i/p NOR gate using nmos technology

stick dig for 3 i/p NOR gate:-

ckt diagram



8) Design and draw stick dig for EX-OR gate

Truth Table

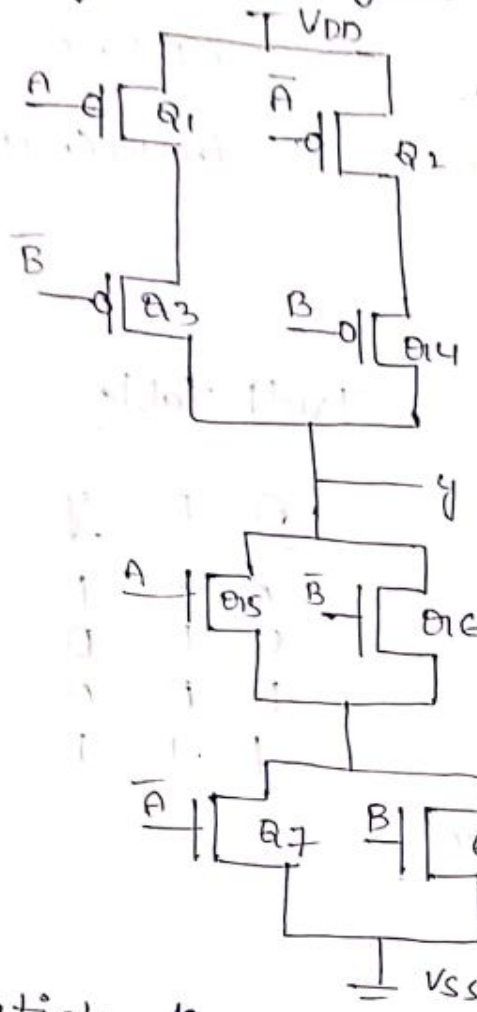
A	B	y
0	0	0
0	1	1
1	0	1
1	1	0

$y = \overline{A}B + A\overline{B}$ — Boolean expression

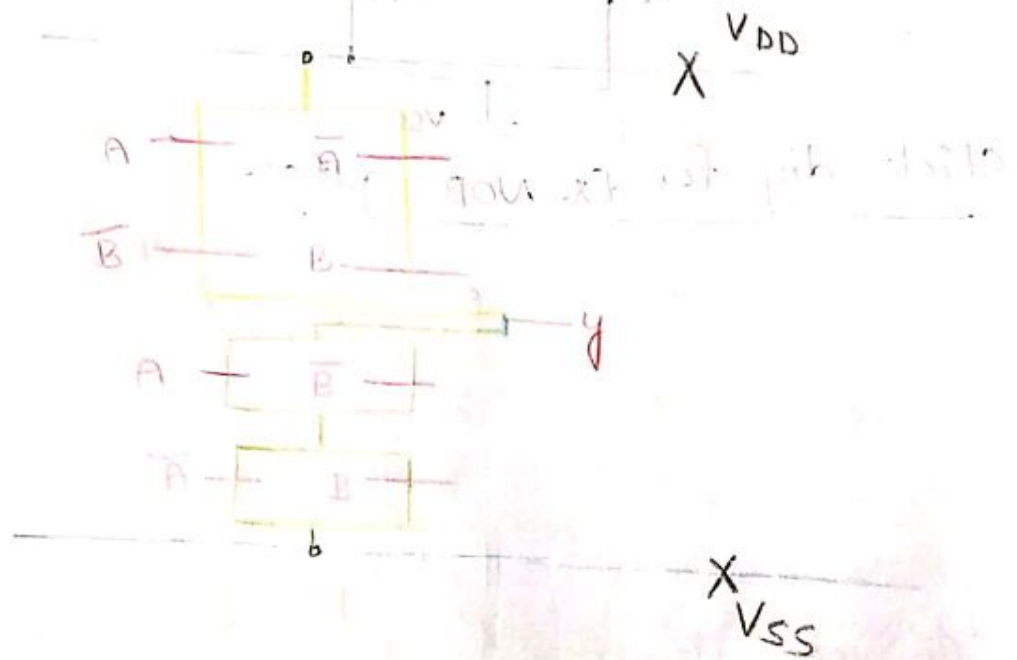
$= \overline{\overline{\overline{A}B + A\overline{B}}}$

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

Ckt dig. for Ex-OR gate:-



stick diagram for Ex-or. gate:-

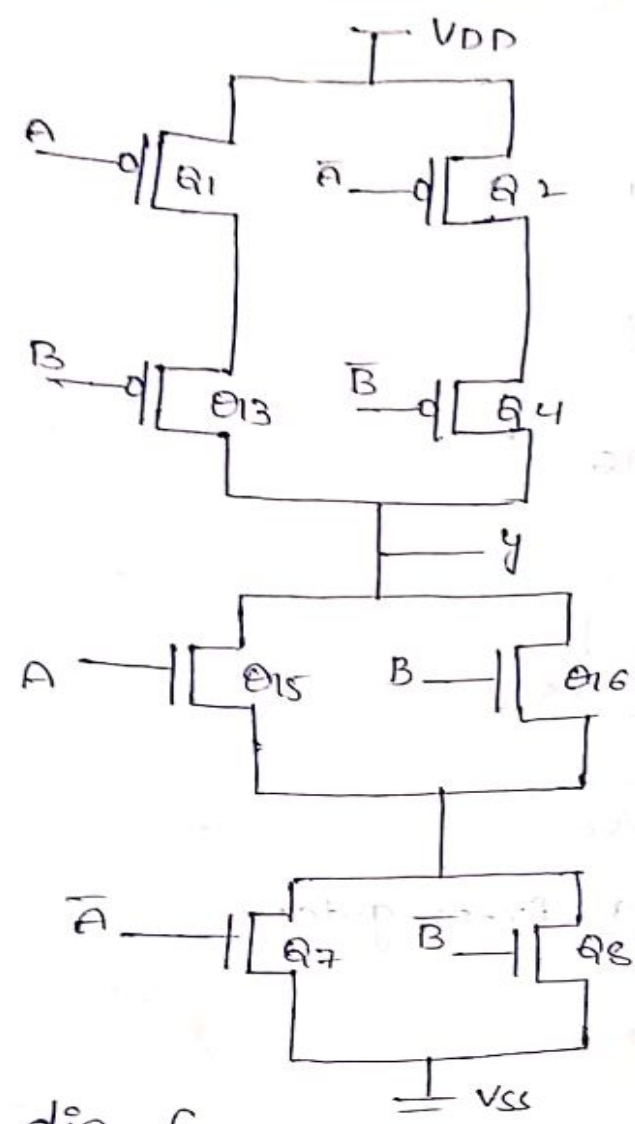


9) Design ckt for Ex-NOR gate $y = \overline{\overline{A}B + A\overline{B}}$

$$= \overline{\overline{A}B + A\overline{B}}$$

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

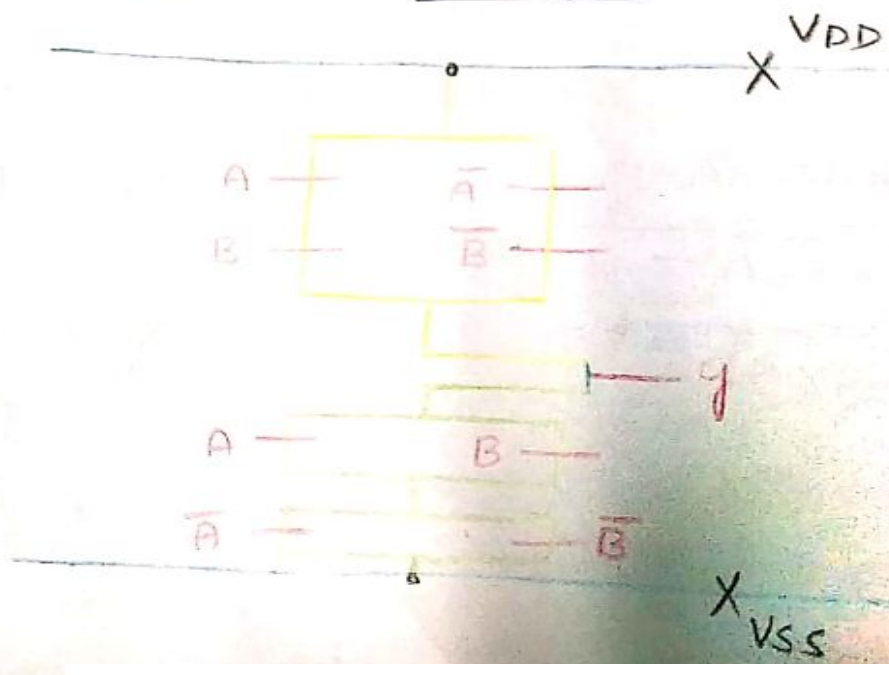
Ckt dig for Ex-NOR gate:-



Truth table

A	B	y
0	0	1
0	1	0
1	0	0
1	1	1

Stick dig for Ex-NOR gate:-



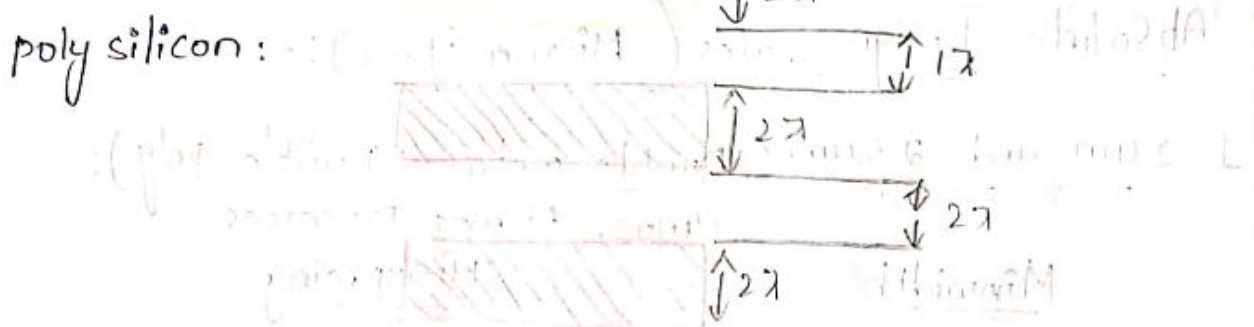
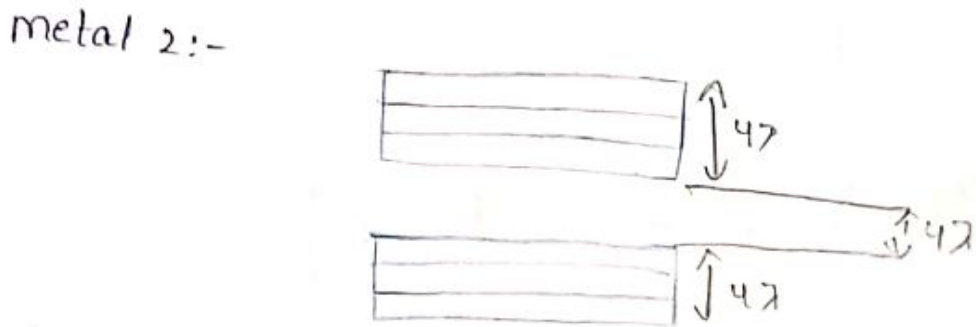
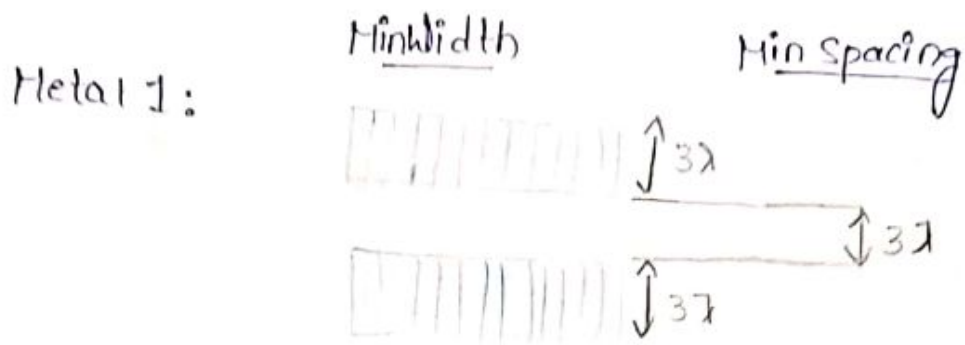
Design rules and layouts:-

- * As the complexity of VLSI technology is more and more to have better understanding we are using a set of rules called design rules.
- * The design rules are the effective interface b/w Design Engineering and fabrication Engineering.
- * CKT designers in general want lighter, smaller layouts for improved performance and reduced silicon area.
- * The process engineer want's Design rules that result's in a controllable and reproducible process.
- * So, we need to have compromise b/w CKT design-er and process engineer requirements.
- * Basically we are having two types of design rules
 1. Scalable Design rules (λ base)
 2. Absolute Design rules (micron base)

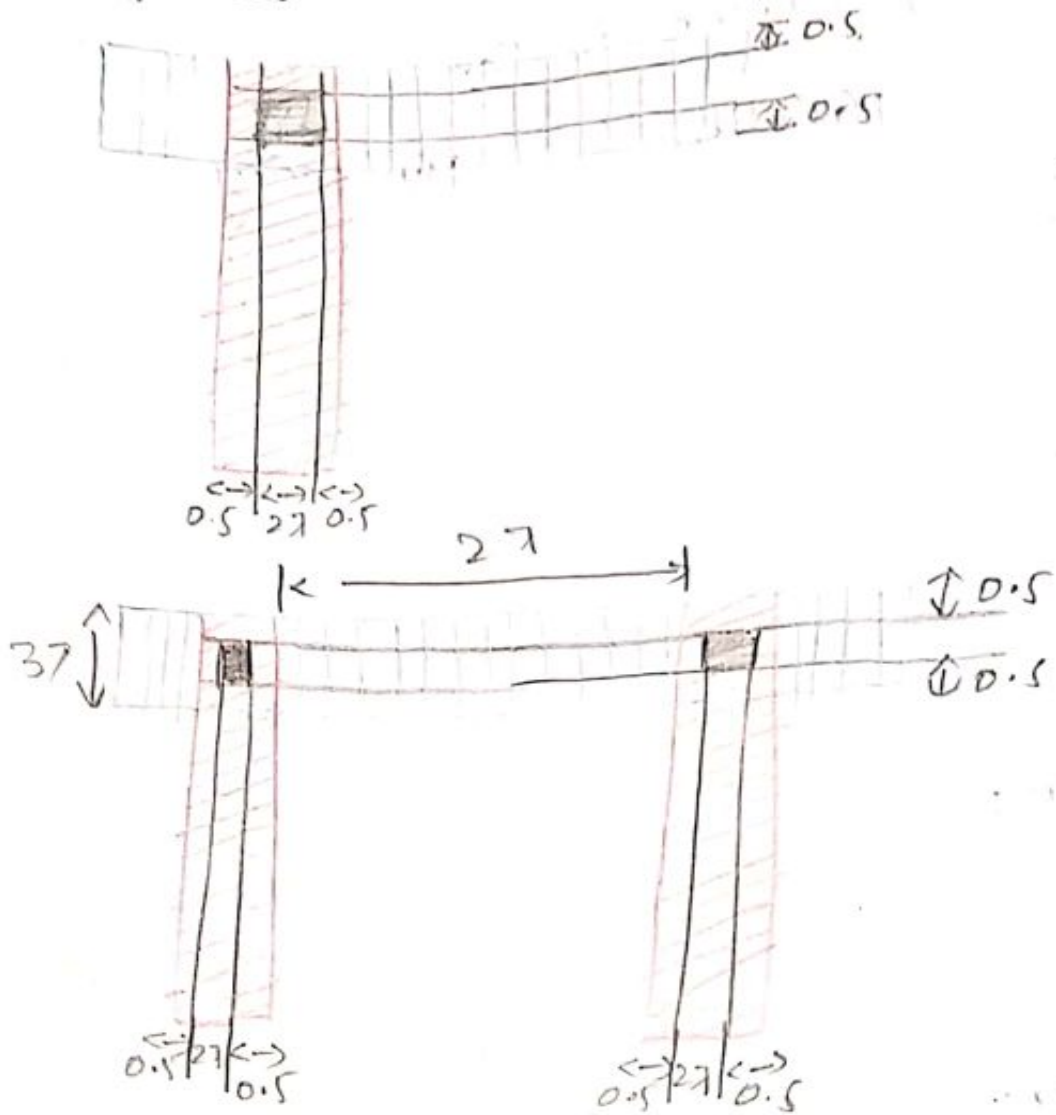
Scalable Design rules:-

1. The minwidth of metal 1 is 3λ and spacing b/w metal 1 and metal 1 is 3λ .
2. The minwidth of metal 2 is 4λ and spacing b/w metal 2 & metal 2 is 4λ .
3. The minwidth of n^+ diffusion (or) p^+ diffusion is 2λ and spacing b/w n^+ diffusion to n^+ diff & p^+ diffusion to p^+ diffusion is 3λ .

4. The minwidth of polysilicon is 2λ and spacing b/w polysilicon to polysilicon is 1λ .



Contact Cuts

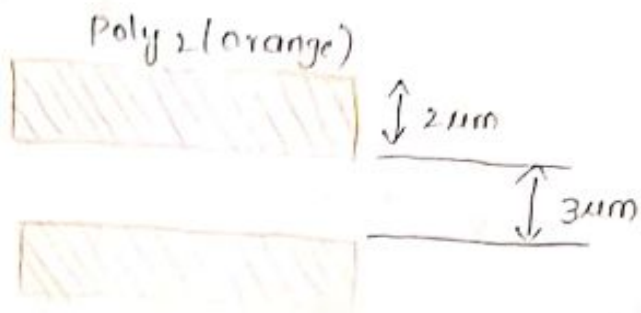


The spacing b/w the polysilicon 1 & polysilicon 2 is 2.7 .

Absolute Design Rules (Micron Base):-

- $2\mu\text{m}$ and $2.5\mu\text{m}$ (Double metal ; Double poly):-
Cmos, bicmos processes
Minwidth Minspacing

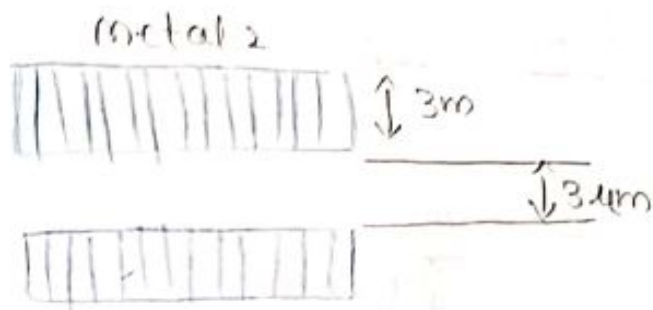




$P^+/n^+ \rightarrow P_1 \rightarrow 1 \mu m$

$P^+/n^+ \rightarrow P_2 \rightarrow 1.5 \mu m$

$P_1 \rightarrow P_2 \rightarrow 2 \mu m$



P_1, m_1

$P_1 \rightarrow 2 \mu m \text{ \& } 2.5 \mu m$

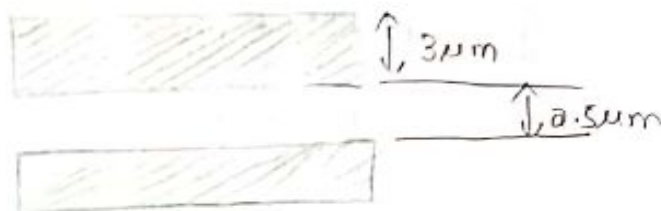
$m_1 \rightarrow 2 \mu m \text{ \& } 2.5 \mu m$

$P_2 \rightarrow 2 \mu m \text{ \& } 3 \mu m$

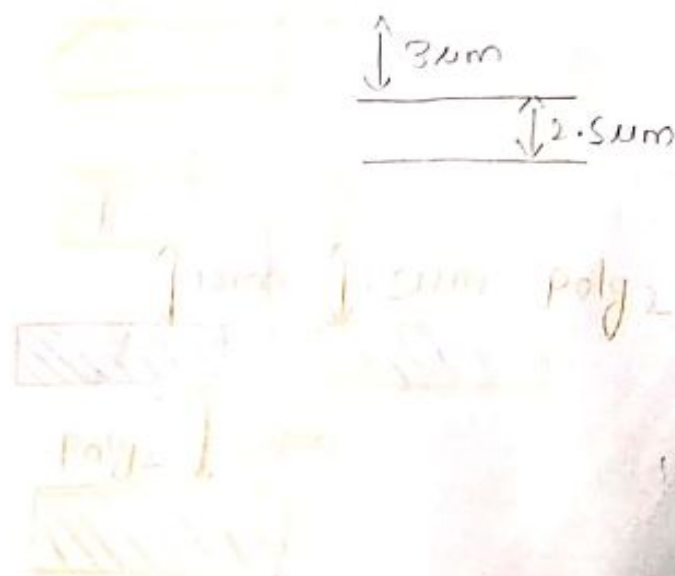
$m_2 \rightarrow 3 \mu m, 3 \mu m$

$P^+/n^+ \rightarrow 3 \mu m \text{ \& } 2.5 \mu m$

n^+ Diff



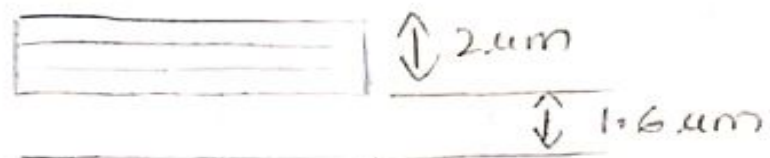
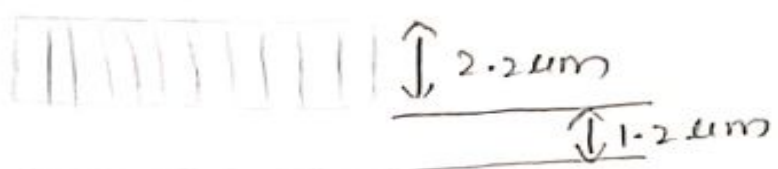
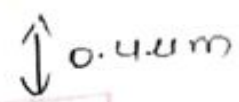
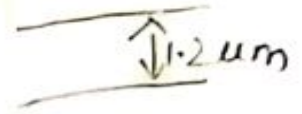
P^+ Diffusion



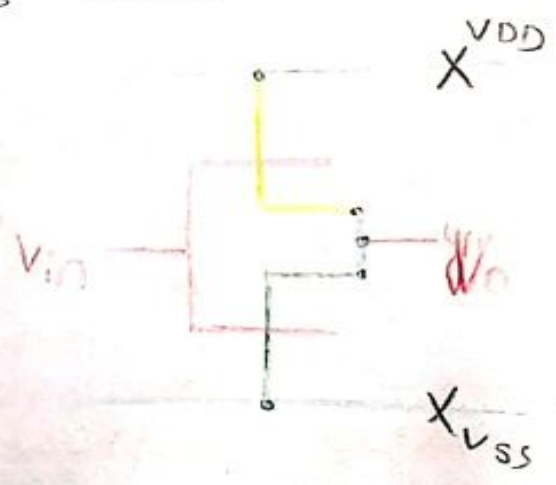
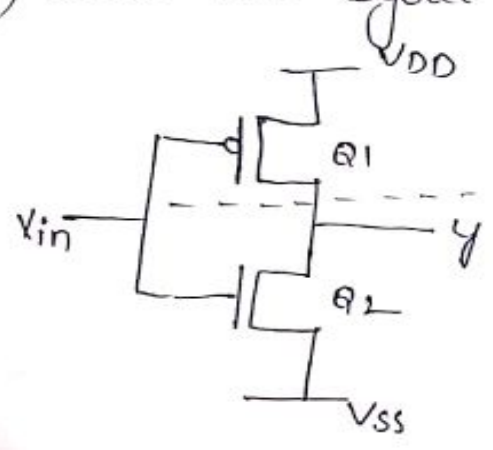
1.2um (Double metal and single poly):-

Minwidth

Minspacing



i) Draw the layout for CMOS inverter.

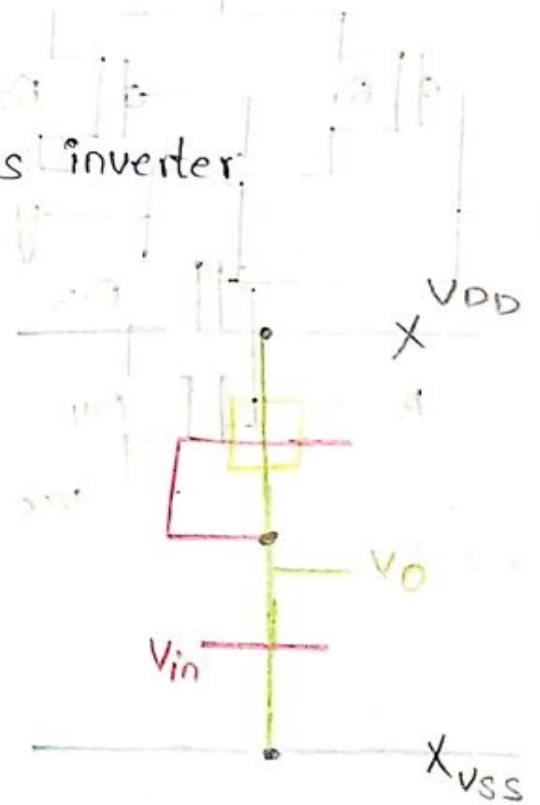
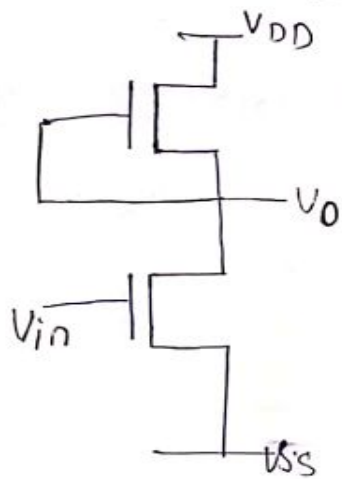


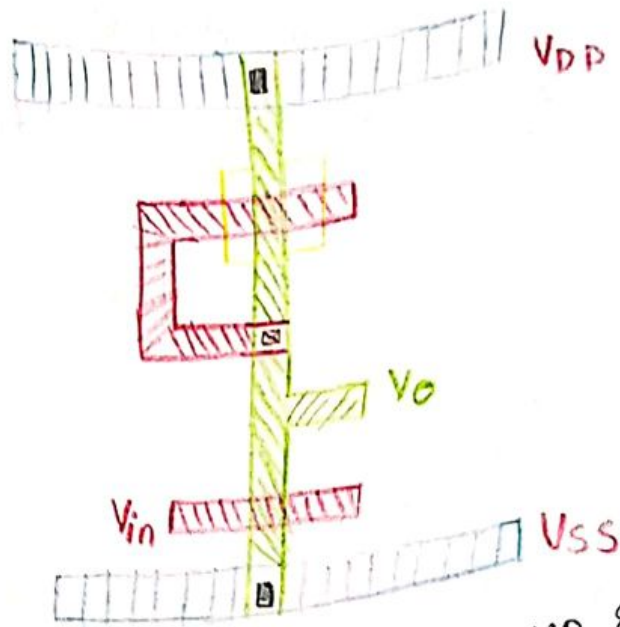
CMOS inverter ckt

Handwritten notes in Hindi, partially legible, appearing to be a student's attempt at a question. The text includes phrases like "गति" (motion) and "वोल्ट" (voltage).

2) Draw the layout for nmos inverter.

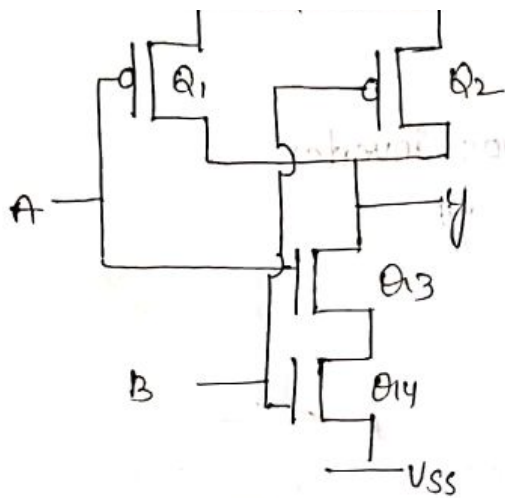
Nmos inverter ckt





* Draw the layout for 2 i/p NAND gate using Cmos logic.

A. Circuit Diagram

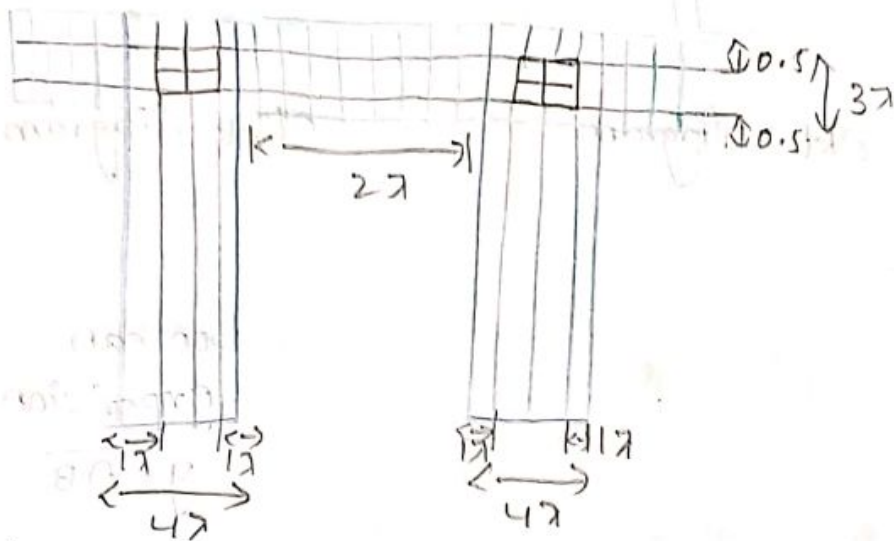


Draw the layout for 2 i/p NAND gate using Cmos logic.



Via

Metal 1

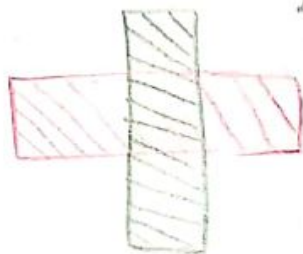


The spacing b/w via 1 & via 2 length is 2λ

Transistor representation:-

nmos enhancement

$2\lambda:2\lambda$



nmos Depletion

$6\lambda:6\lambda$



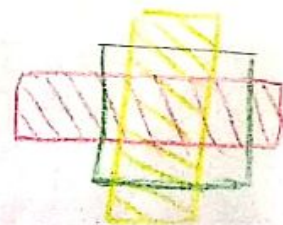
pmos enhancement

$2\lambda:2\lambda$

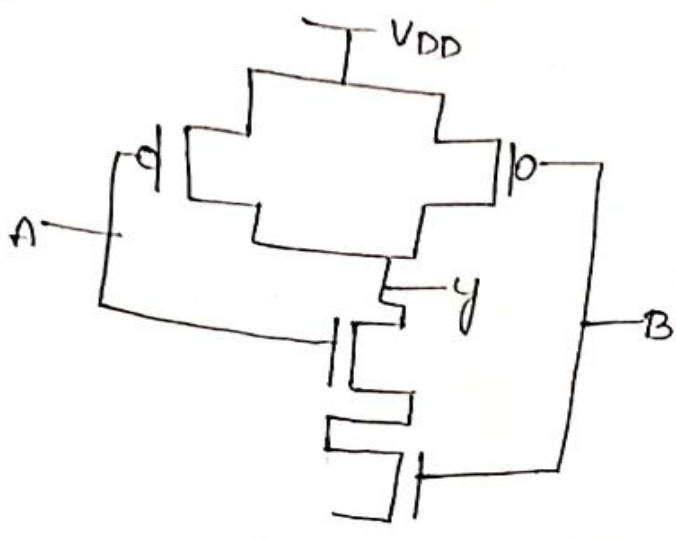


pmos depletion

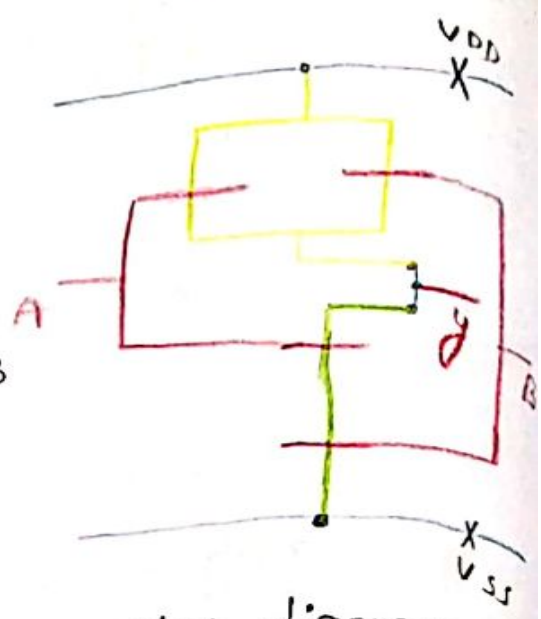
$6\lambda:6\lambda$



* Draw the layout for 2 i/p NAND gate using CMOS logic.

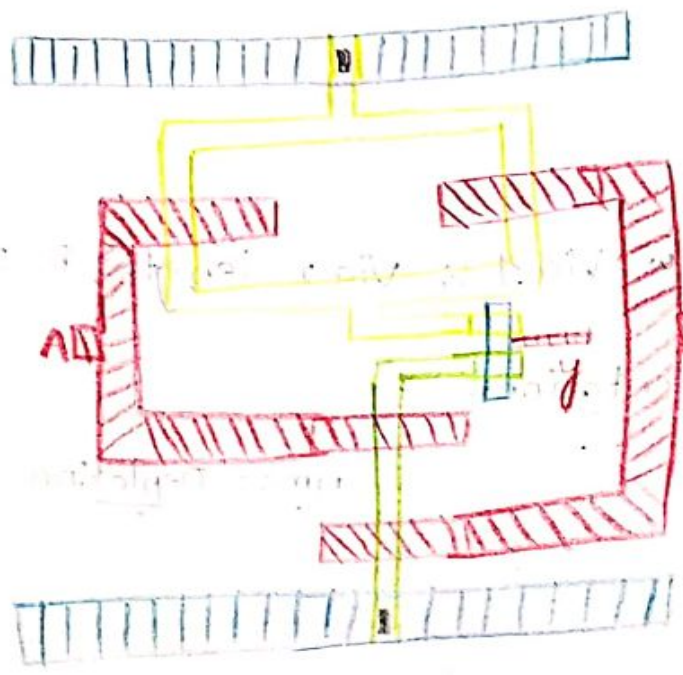


Ckt diagram

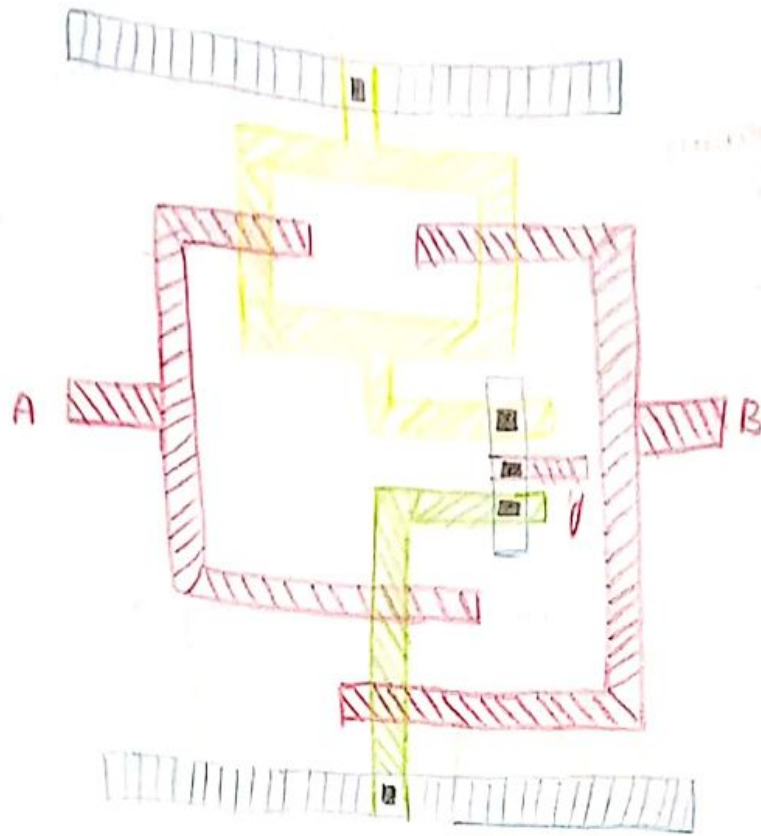


Stick diagram

layout

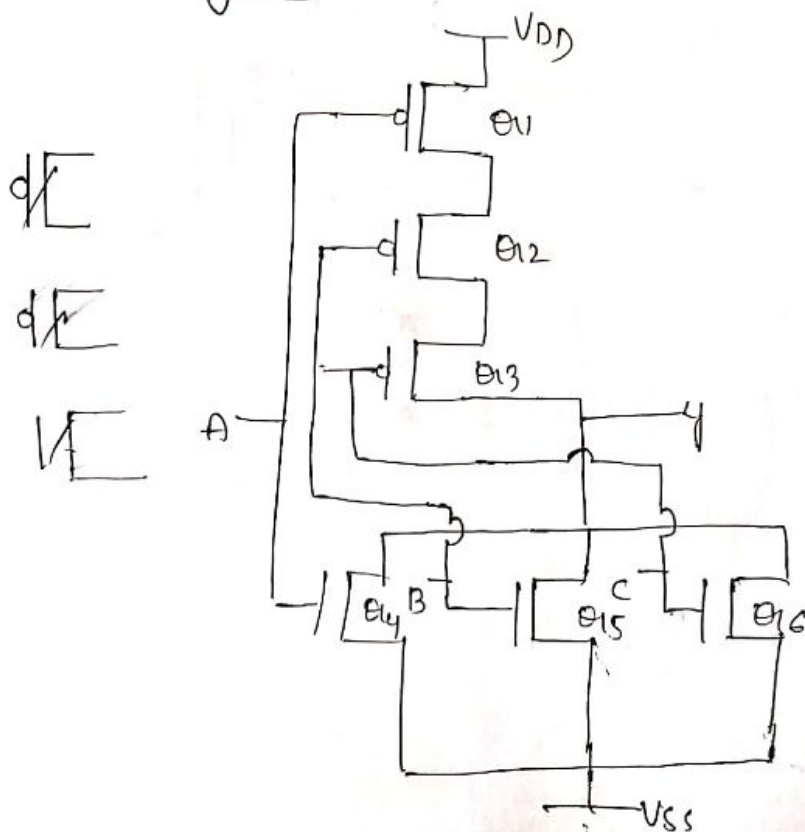


boolean
expansion
 $y = \overline{AB}$

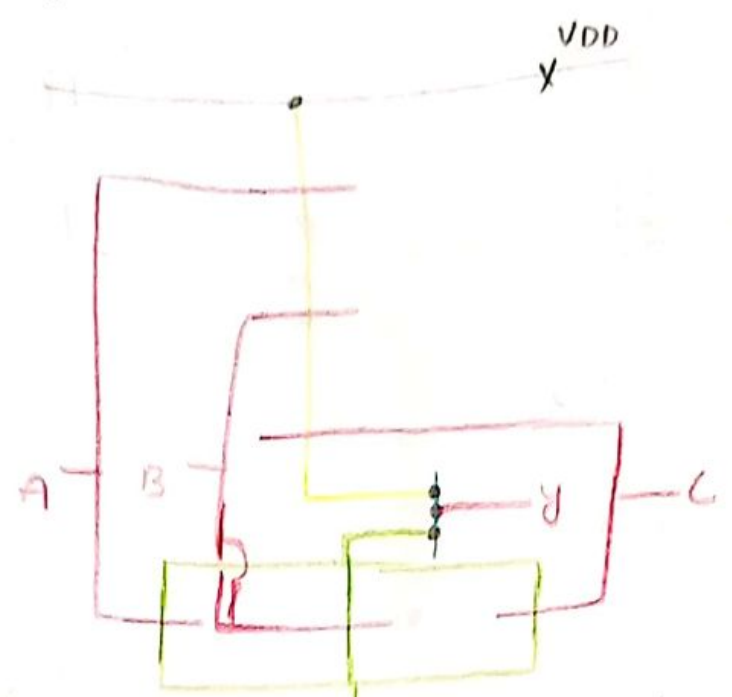


* Draw the layout for boolean equation $y = \overline{(A+B+C)}$ using CMOS logic.

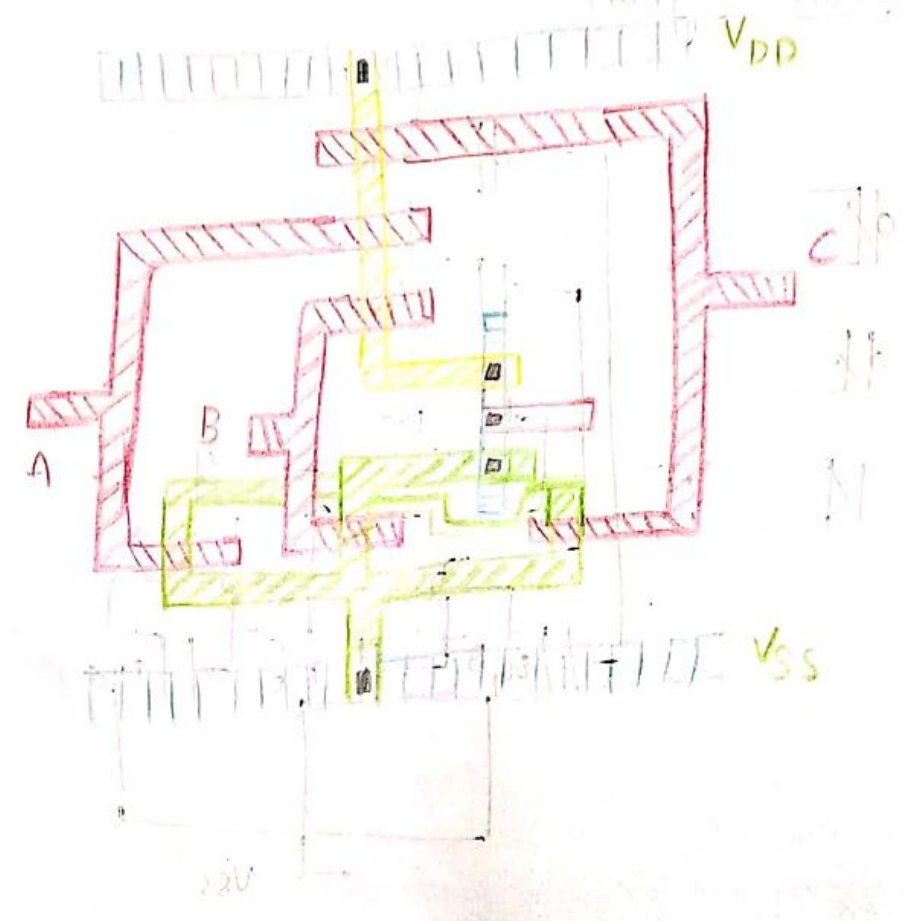
A. Circuit Diagram



General
Stick diagram



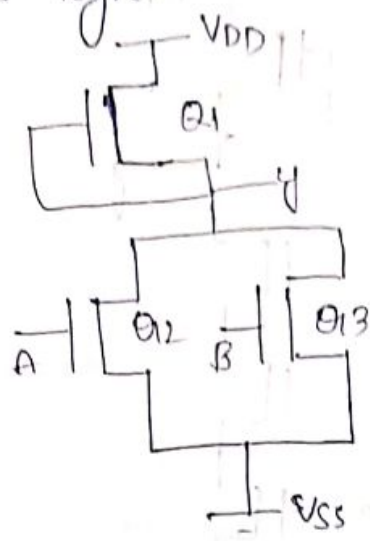
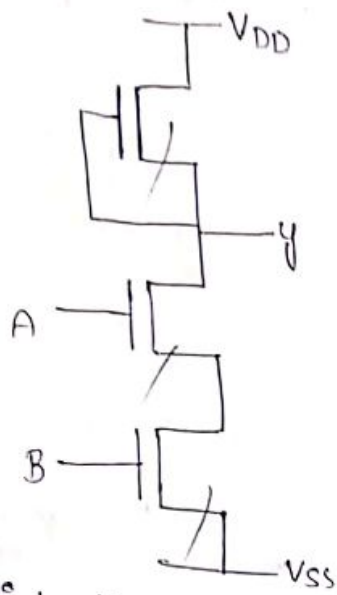
From the layout the layout editor will generate the circuit schematic.



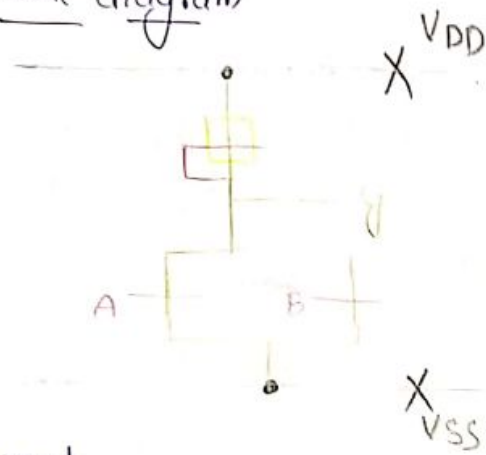
→ 2 i/p NOR gate using nmos logic → $y = \overline{(A+B)}$

→ 3 i/p NAND gate using nmos logic → $y = \overline{ABC}$

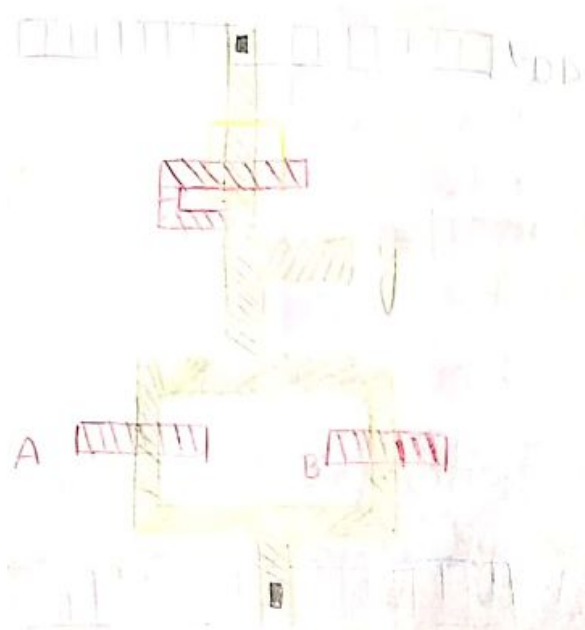
1. 2 i/p NOR gate using nmos logic.



Stick diagram

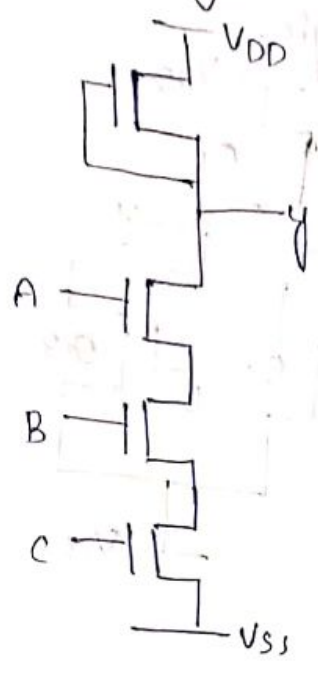


Layout

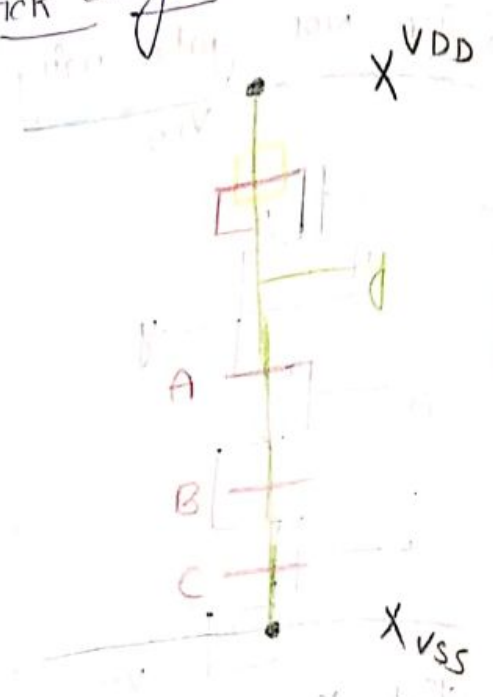


* 3i/p NAND gate using nmos logic. $y = \overline{ABC}$

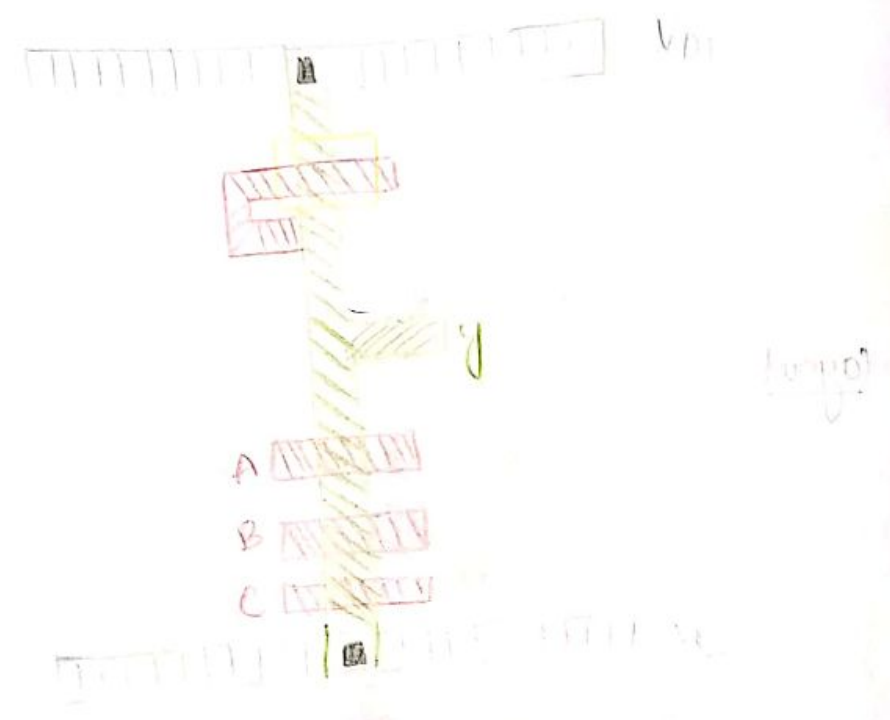
Circuit diagram



Stick diagram



Layout



General observations for design rules:-

A VLSI designer is dealing with silicon circuits whose dimensions and features are extremely microscopic in nature.

* If the line widths are too small, the lines so defined may tend to be discontinuous at places (or) may accordingly merge at places.

* The main purpose of design rules are

1. provides communication links between circuit designer and process engineer responsible for fabrication.

2. The goal of any set of design rules should be to optimise yield while keeping the geometry as small as possible without compromising the reliability of the finished circuit.

3. Try to exclude poly from the areas of p⁺mask/n⁺mask that may affect the resistance of poly at certain paths.

4. Metal is deposited at the top of all oxide layers, since it is light reflective, these factors combine to give poor edge definition.

5. Metal 2 is having even more uneven terrain on which to be deposited and patterned. Hence metal 2 is wider than metal 1.

6. metal to metal separation is also large and is brought about mainly by difficulties in defining metal edges.

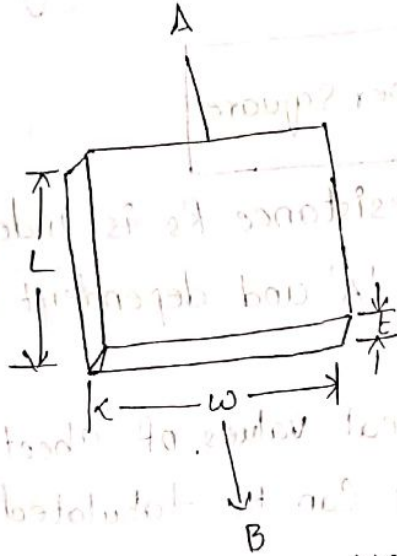
31/1/18

UNIT-II

Basic Circuit Concepts

Sheet Resistance (Rs):-

The concept of sheet Resistance can be understood by considering a uniform slab material as shown in the figure below.



Sheet Resistance Model

* So from the given diagram we can say that it is having a length of 'L' and width of 'w' and with a thickness of 't'.

* The Resistance b/w the terminals A and B, can be given as R_{AB} .

$$R_{AB} = \frac{\rho L}{A}$$

where ρ - resistivity

L - length of the material

A - cross-sectional Area

* for a uniform slab $L=w$

$$WKT \quad R_{AB} = \frac{\rho L}{A}$$

$$R_{AB} = \frac{\rho l}{A}$$

$$R_{AB} = \frac{\rho l}{wt}$$

$$= \frac{\rho l}{wt} \quad (\text{uniform slab } l=w)$$


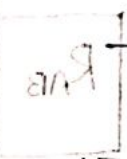
$$R_{AB} = \frac{\rho}{t}$$

∴ The sheet Resistance R_s can be given as

$$R_s = \frac{\rho}{t} \quad \Omega \text{ per square}$$

Note:- The sheet Resistance R_s is independent of Cross-sectional Area 'A' and dependent on thickness 't'.

* Some of the typical values of Sheet Resistance of different technologies can be tabulated as like below

Layer	sheet Resistance		
	5μm	2μm (orbit)	1.2μm
Metal	0.03	0.04	0.04
Diffusion (Active)	10 → 50	20 → 45	20 → 45
silicid side	2 → 4		
polysilicon	15 → 100	15 → 30	15 → 30
n-transistor channel	10	2×10^4	2×10^4
p-transistor channel	2.5×10^4	4.5×10^4	4.5×10^4

Sheet Resistance Concept Applied to MOS transistors
 and Inverters:-
 The sheet Resistance Concept can be extended to
 inverters.

Here let us consider a CMOS inverter as shown
 below.

* The channel Resistance of
 pmos can be given as

$$R_{pus} = z R_s$$

$$\text{WKT } z = \frac{L}{w}$$

$$\Rightarrow \frac{27}{27} = 1$$

$$R_{pus} = 1 \cdot R_s$$

$$= (2.5 \times 10^4)$$

$$R_{pus} = 25 \text{ k}\Omega$$

* The channel resistance of nmos can be given
 as

$$R_{pds} = z \cdot R_s$$

$$z = \frac{L}{w}$$

$$= \frac{27}{27} = 1$$

$$R_{pds} = 1 \cdot R_s$$

$$= 10^4$$

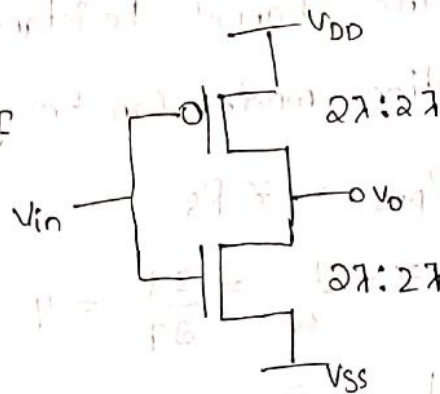
$$R_{pds} = 10 \text{ k}\Omega$$

\(\therefore\) The ON channel Resistance of CMOS inverter is

$$R_{ON} = R_{pus} + R_{pds}$$

$$= 25 \text{ k}\Omega + 10 \text{ k}\Omega$$

$$= 35 \text{ k}\Omega$$

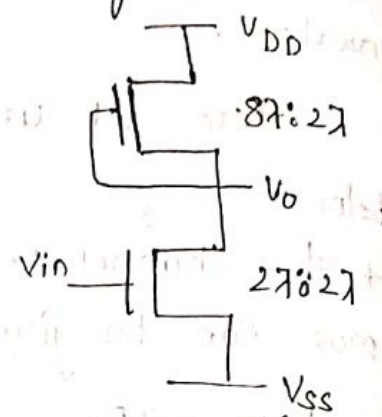


R_s values can be
 taken for 5um technology
 for convenience.

Sheet Resistance Concept - Applied for nmos Inverter.

The nmos Inverter is diagrammatically as shown below.

The channel Resistance of depletion mode can be given as:



$$R_{pus} = z \cdot R_s$$

$$z = \frac{L}{w} = \frac{8\lambda}{2\lambda} = 4$$

$$\begin{aligned} \text{The } R_{pus} &= 4 \cdot R_s \\ &= 4 \cdot (2.5 \times 10^4) \\ &= 40k\Omega \end{aligned}$$

The channel Resistance of enhancement mode can be given as:

$$R_{pds} = z \cdot R_s$$

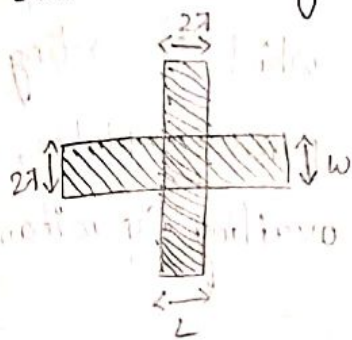
$$z = \frac{L}{w} = \frac{2\lambda}{2\lambda} = 1$$

$$\begin{aligned} R_{pds} &= 1 \cdot R_s \\ &= 1(10^4) \\ &= 10k\Omega \end{aligned}$$

∴ The ON channel Resistance of nmos inverter can be is $R_{ON} = R_{pus} + R_{pds}$

$$\begin{aligned} &= 40k\Omega + 10k\Omega \\ &= 50k\Omega \end{aligned}$$

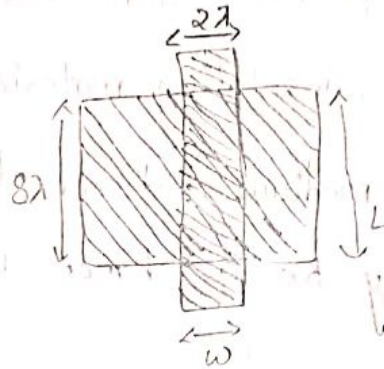
The sheet Resistance Concept can be extended for transistors, for example. Consider two transistors as shown in figure 'a' and 'b' shown below.



$$L:W$$

$$2\lambda:2\lambda$$

figure 'a'



$$L:W$$

$$8\lambda:2\lambda$$

figure 'b'

In the above diagrams figure 'a' is having a length of '2λ' and a width of '2λ' and figure 'b' carries a length of '8λ' with a width of '2λ'.

for a fig 'a' the channel Resistance can be calculated as $R = Z R_s$

where $Z = \frac{L}{W} = \frac{2\lambda}{2\lambda} = 1$

$$R = 1 \cdot R_s$$

$$R = 1 \cdot (10^4)$$

$$= 10 \text{ k}\Omega$$

the channel Resistance for fig 'b' can be calculated as $R = Z \cdot R_s$

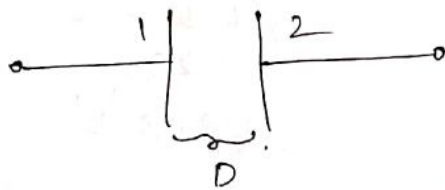
$$Z = \frac{L}{W} = \frac{8\lambda}{2\lambda} = 4$$

$$R = 4 \cdot R_s$$

$$= 4(10^4)$$

$$R = 40 \text{ k}\Omega$$

Area Capacitance of Layers :-
 In the IC fabrication process, the layers can be separated from one another by an oxide layer (i.e., insulating material) which is acting as dielectric medium between two parallel plates, so there may be a chance of available capacitance.



The capacitance 'C' can be given as

$$C = \frac{\epsilon A}{D}$$

where ϵ = permittivity and can be given as

$$\epsilon = \epsilon_0 \epsilon_{ins}$$

ϵ_0 → Absolute permittivity (or) permittivity of free space. (8.854×10^{-12} faraday/meter)

ϵ_{ins} → Relative permittivity = 4 for silicon

A = Area of plates

D = thickness of SiO_2

* Some of the typical values of capacitance for 5 μ m, 2 μ m and 1.2 μ m technologies can be tabulated below.

Capacitance	Value in $\text{PF} \times 10^4 / \mu\text{m}^2$ (relative values in brackets)		
	$5\mu\text{m}$	$2\mu\text{m}$	$1.2\mu\text{m}$
gate to channel capacitance	4 (1.0)	8 (1.0)	16 (1.0)
diffusion (active)	1 (0.25)	1.75 (0.22)	3.75 (0.23)
polysilicon to substrate	0.4 (0.1)	0.6 (0.075)	0.6 (0.038)
Metal 1 to substrate	0.3 (0.075)	0.33 (0.04)	0.33 (0.02)
Metal 2 to substrate	0.2 (0.5)	0.17 (0.02)	0.17 (0.01)
Metal 2 to Metal 1	0.4 (0.1)	0.5 (0.06)	0.5 (0.03)
Metal 2 to polysilicon	0.3 (0.075)	0.3 (0.038)	0.3 (0.018)

Standard unit of Capacitance ($\square C_g$) :-

The standard unit of Capacitance can be define as gate to channel Capacitance of a MOS transistor having the feature size of $L=W$.

Hence, for example standard unit of Capacitance can be calculated for different technologies.

For $5\mu\text{m}$ technology :-

$$\begin{aligned} \text{The area per standard square} &= 5\mu\text{m} \times 5\mu\text{m} \\ &= 25\mu\text{m}^2 \end{aligned}$$

The standard unit of Capacitance ($\square C_g$) can be given as

$$\square C_g = \text{Area/standard square} \times \text{Capacitance}$$

$$\Rightarrow 25 \mu\text{m}^2 \times 4 \times 10^{-4} \text{ PF}/\mu\text{m}^2$$

$$\Rightarrow 100 \times 10^{-4} \text{ PF}$$

$$\square C_g = 0.01 \text{ PF}$$

for 2 μm technology

$$\text{Area/standard square} = 2 \mu\text{m} \times 2 \mu\text{m} \\ = 4 \mu\text{m}^2$$

The standard unit Capacitance ($\square C_g$) can be given as

$$\square C_g = \text{Area/standard square} \times \text{Capacitance}$$

$$= 4 \mu\text{m}^2 \times 8 \times 10^{-4} \text{ PF}/\mu\text{m}^2$$

$$= 32 \times 10^{-4} \text{ PF}$$

$$= 0.0032 \text{ PF}$$

for 1.2 μm technology

$$\text{Area/standard square} = 1.2 \mu\text{m} \times 1.2 \mu\text{m}$$

$$\therefore \text{Area/standard square} = 1.44 \mu\text{m}^2$$

The standard unit Capacitance ($\square C_g$) can be given as

$$\square C_g = \text{Area/standard square} \times \text{Capacitance}$$

$$= 1.44 \mu\text{m}^2 \times 16 \times 10^{-4} \text{ PF}/\mu\text{m}^2$$

$$= 23.04 \times 10^{-4} \text{ PF}$$

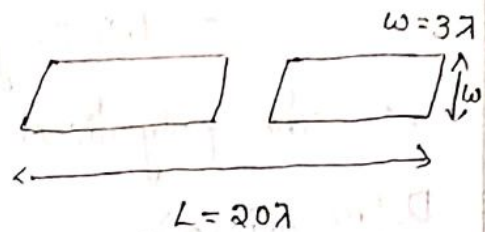
$$= 0.023 \text{ PF}$$

Some Area Capacitance Calculations:

In this the relative values of capacitance can be used for the representation of capacitance and all values carried out in 'x' based values.

The relative Area can be represented as the ratio of Area of interest to the standard Area.

$$\therefore \text{Relative Area} = \frac{\text{Area of interest}}{\text{standard Area}}$$



$$\begin{aligned} \text{Relative Area} &= \frac{20\lambda \times 3\lambda}{2\lambda \times 2\lambda} \\ &= \frac{60\lambda^2}{4\lambda^2} = 15 \end{aligned}$$

(*) for 5μm technology

(i) The Capacitance to substrate can be given as

⇒ Relative area × Capacitance

$$= 15 \times 0.075 \square Cg$$

$$= 1.125 \square Cg$$

∴ The Capacitance to substrate is 1.125 $\square Cg$ times of Square Cg ($\square Cg$).

(2) The diffusion Capacitance can be calculated

as = Relative area × Capacitance

$$= 15 \times 0.25 \square Cg$$

$$= 3.75 \square Cg$$

∴ The diffusion Capacitance is 3.75 times of $\square Cg$.

3) For polysilicon

The capacitance for polysilicon can be calculated as = Relative Area \times Capacitance.

$$= 15 \times 0.1 \square \text{cg}$$

$$= 1.5 \square \text{cg}$$

\therefore The polysilicon capacitance is 1.5 times of $\square \text{cg}$

Delay unit (τ):-

The delay unit (τ) can be given as the product of Sheet Resistance (R_s) and standard unit of gate Capacitance ($\square \text{cg}$).

$$\tau = R_s \square \text{cg}$$

for 5 μm technology

$$\text{WKT } \tau = R_s \square \text{cg}$$

$$= 10^4 \times 0.01 \text{PF}$$

$$= 10^4 \times 0.01 \times 10^{-12}$$

$$= 100 \times 10^{-12}$$

$$= 1 \times 10^{-10}$$

$$= 0.1 \times 10^{-9}$$

$$\tau = 0.1 \text{ nsec}$$

for 2 μm technology

$$\text{WKT } \tau = R_s \square \text{cg}$$

$$= 2 \times 10^4 \times 32 \times 10^{-4} \text{PF}$$

$$= 64 \text{PF}$$

$$\tau = 64 \times 10^{-13}$$

$$= 0.064 \times 10^{-9}$$

$$\tau = 0.064 \text{ nsec}$$

for 1.2 μm technology

$$\tau = k_s C_g$$

$$= 2 \times 10^4 \times 23.04 \times 10^9 \text{ pF}$$

$$= 46.08 \times 10^{-13}$$

$$\tau = 0.046 \text{ nsec}$$

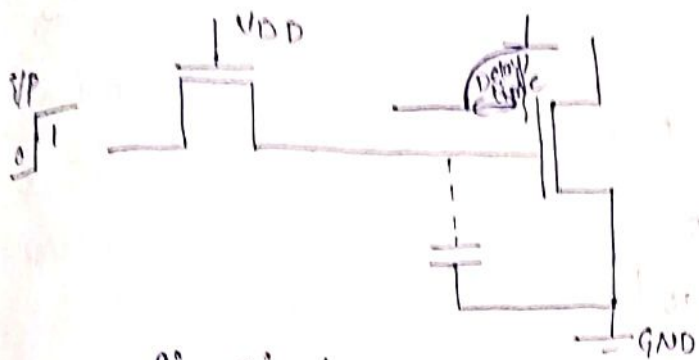


Fig: Simple model to calculate delay time.

We know that the electron transit time τ_{ds} as

$$\tau_{ds} = \frac{L^2}{\mu v_{ds}}$$

* When i/p signal is transmitted through the pass transistors the output voltage will get reduce to 63% of V_{DD} that is $0.63 \times V_{DD}$
 $= 0.63 \times 5 \text{ V}$

which come to $\approx 3 \text{ V}$

* for 5 μm technology

$$\tau_{ds} = \frac{L^2}{\mu v_{ds}}$$

$$= \frac{5 \mu\text{m} \times 5 \mu\text{m}}{650 \text{ cm}^2/\text{V-sec} \times 3 \text{ V}}$$

$$\begin{aligned}
 &= \frac{25 \times (10^{-6})^2 \text{ m}^2}{650 (0.01) \text{ m}^2 / \text{sec} \times 3 \text{ V}} \\
 &= \frac{25 \times 10^{-12}}{650 \times 0.01 \times 3} \text{ sec} \\
 &= \frac{25 \times 10^{-12}}{19.5} \\
 &= \frac{25 \times 10^{-3} \times 10^{-9}}{19.5} \\
 &= \frac{2.5 \times 10^{-9}}{19.5 \times 10^3} \\
 &= 0.128 \times 10^{-9} \\
 &\approx 0.13 \text{ nsec}
 \end{aligned}$$

Note:- The delay unit τ_d is approximately equal to the electron transit time τ_{ds} .

Inverter Delay:-

nmos Inverter Delay:-

here let us consider an nmos inverter with a ratio of 4:1.

The pull up to pulldown ratio of nmos driven by another nmos is 4:1.

$$\text{i.e.} \quad \frac{Z_{pu}}{Z_{pd}} = \frac{4}{1}$$

$$Z_{pu} = 4 Z_{pd}$$

$$Z_{pu} = 4 R_s$$

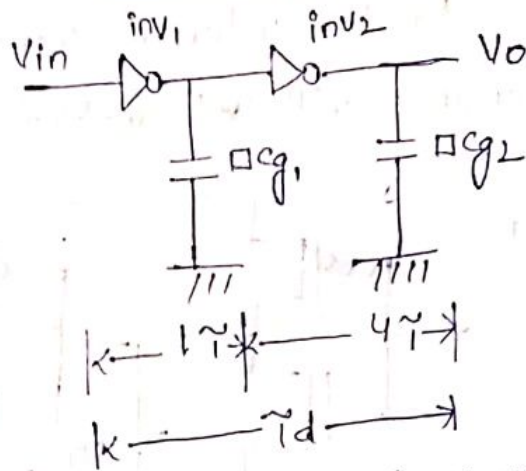
$$\frac{R_{pu}}{R_{pd}} = \frac{4}{1}$$

$$R_{pu} = 4 R_{pd}$$

$$R_{pu} = 4 R_s$$

$$R_{pu} = 4 \times 10^4$$

$$R_{pu} = 40 \text{ k}\Omega$$



* The delay is not effected by the cascade connections of inverters but it is due to the turning on/off actions

* The total delay τ_d is the combination of both inverters.

$$\text{i.e., } \tau_d = 1\tau + 4\tau$$

$$= \tau [1 + 4]$$

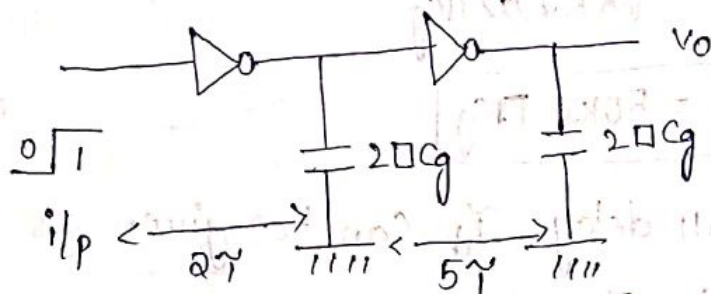
$$= \tau \left[1 + \frac{4}{1}\right]$$

$$\tau_d = \tau \left[1 + \frac{Z_{pu}}{Z_{pd}}\right]$$

\therefore The total delay for an nmos inverter is $\tau_d = 5\tau$.

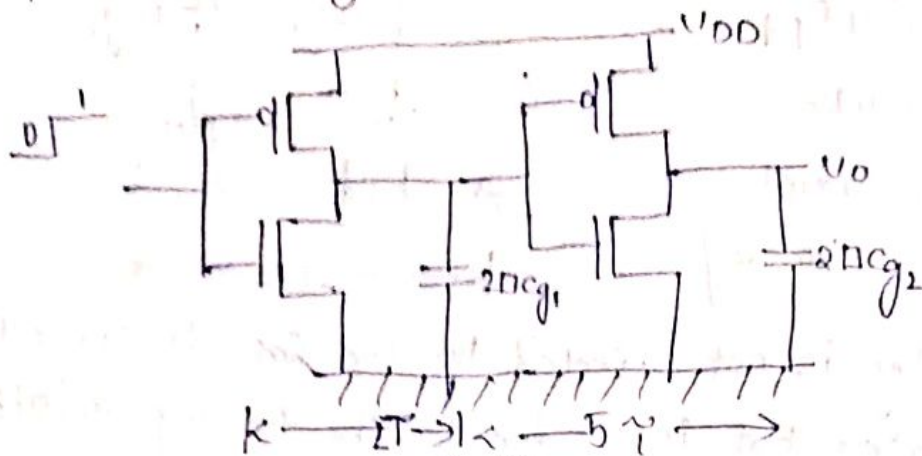
CMOS Inverter Delay:-

Here let us consider an arrangement shown below for the calculation of CMOS inverter delay.



fig(a)

Due to the internal wiring capacitance the load capacitance will get doubled that is $2 \square C_g$.



fig(b):

* when i/p is V_{DD} (logic '1') then the nmos of inverter will get on and the capacitor will get discharge path through the nmos.

i.e., $\tau_1 = R_n 2 \square C_g$

$$\tau_1 = 20 \text{ k}\Omega \square C_g$$

* when the capacitor of inverter 1 is discharge then at the input of inverter 2 we have logic '0'. i.e., pmos of inverter 2 will get on and the capacitor will get charge.

i.e., $\tau_2 = R_p 2 \square C_g$
 $= (25 \text{ k}\Omega) \times 2 \square C_g$

$$\tau_2 = 50 \text{ k}\Omega \square C_g$$

\therefore The overall delay τ_d can be given as

$$\tau_d = \tau_1 + \tau_2$$

$$= 20 \text{ k}\Omega \square C_g + 50 \text{ k}\Omega \square C_g$$

$$= 2R_s \square C_g + 5R_s \square C_g$$

$$= R_s \square C_g [2+5]$$

$$\boxed{\tilde{T}_d = 7\tilde{T}}$$

∴ The overall delay of CMOS inverter is $7\tilde{T}$.

* The overall delay can be better understood with a calculations of Rise time (T_r) and fall time (T_f).

Rise time Calculation (T_r):-

The calculation of Rise time ' T_r ' can be done when input is logic '0'.

wkt I_{ds} for saturation

$$I_{ds} = \frac{k_n}{2} [V_{gs} - V_{t_n}]^2$$

$$I_{ds} = \beta_p \frac{[V_{gs} - V_{t_p}]^2}{2} \rightarrow (1)$$

$$V_{out} = I_{ds} R \rightarrow (2)$$

$$\text{wkt } \tilde{T}_r = RC_L$$

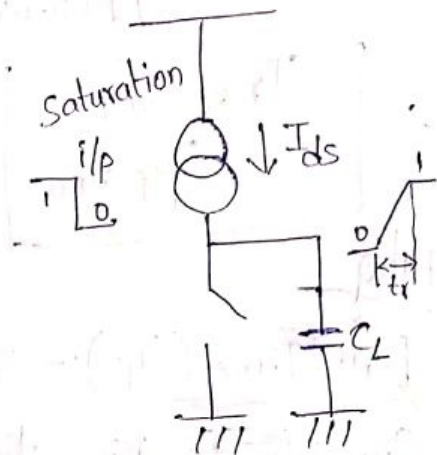
$$\boxed{R = \frac{\tilde{T}_r}{C_L}} \rightarrow (3)$$

Sub (3) in (2)

$$V_{out} = I_{ds} \cdot \frac{\tilde{T}_r}{C_L}$$

$$\tilde{T}_r = \frac{V_{out} C_L}{I_{ds}}$$

$$= \frac{V_{out} C_L}{\beta_p [V_{gs} - V_{t_p}]^2 / 2}$$



$$T_r = \frac{2 V_{out} C_L}{\beta P [V_{gs} - |V_{tp}|]^2}$$

$$V_{gs} = V_{DD}$$

$$V_{out} = V_{DD}$$

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

$$T_r = \frac{2 \cdot V_{DD} C_L}{\beta P [V_{DD} - 0.2 V_{DD}]^2}$$

$$= \frac{2 V_{DD} C_L}{\beta P (0.8 V_{DD})^2}$$

$$= \frac{2 V_{DD} C_L}{\beta P (0.64) V_{DD}^2}$$

$$= \frac{2 C_L}{\beta P (0.64) V_{DD}}$$

$$T_r \approx \frac{3 C_L}{\beta P V_{DD}}$$

Hint

Fall Time (T_f):- The fall time (T_f) can be calculated similarly to Rise time (T_r) calculation.

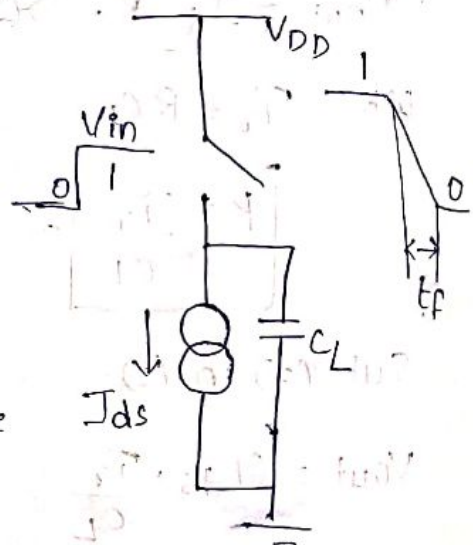
* The fall time T_f can be given as

$$T_f = \frac{3 C_L}{\beta_n V_{DD}}$$

(reference from Rise time calculation)

* The ratio of Rise time to fall time can be given as

$$\frac{T_r}{T_f} = \frac{\frac{3 C_L}{\beta P / V_{DD}}}{\frac{3 C_L}{\beta_n / V_{DD}}}$$



$$\frac{T_r}{T_f} = \frac{\beta_n}{\beta_p}$$

\therefore $T_r \propto \frac{1}{\beta_p}$ and fall time $T_f \propto \frac{1}{\beta_n}$

* The rise time ' T_r ' and fall time ' T_f ' both are $\propto CL$ and $\propto \frac{1}{V_{DD}}$

Driving large Capacitive loads:-

* The concept of driving large capacitive loads may arise when the signals are propagating from on-chip to off-chip peripherals, which are having comparatively very large values.

* Here the load capacitance ' C_L ' is equal orders $>$ than the gate capacitance ' C_g ' i.e.,

$$C_L > 10^4 C_g \text{ [Assumption].}$$

* In order to have decreased delay we need to maintain increased channel length which may further decrease the resistance.

* To drive large capacitive loads we are having 3 techniques.

1. Cascoded connections of inverters as drivers.
2. Superbuffers as drivers.
3. Bi-CMOS drivers.

1. Cascaded Connections of Inverter & as drivers:-

- * When driving large capacitive loads to have a minimum delay we should have low resistances.
- * Low resistances can be obtained by having low

Z_{pu} & Z_{pd} 's

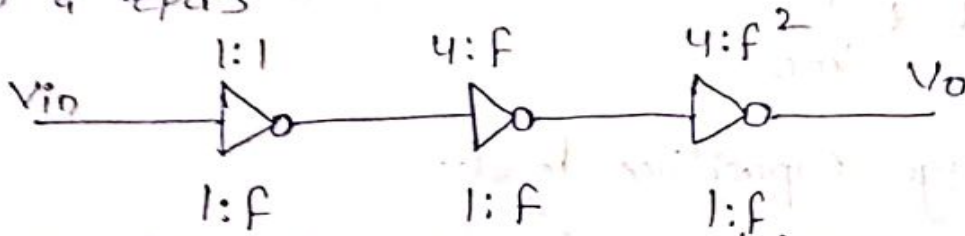


fig:- Cascaded inv as drivers

- * The arrangement of Cascaded inverters as drivers is shown below.

- * If we increase the width factor 'f' then the load on capacitance may increase that results in larger capacitive area.

- * If there is increase in width factor then the resistance will get automatically decrease thereby we can have minimum delay.

for nmos:- Delay / stage = $1f\gamma$ for ΔV_{in}
 $= 4f\gamma$ for ∇V_{in}

$$\tilde{T}_d = 1f\gamma + 4f\gamma = 5f\gamma$$

for pmos:- Delay / stage = $2f\gamma$ for ΔV_{in}
 $= 5f\gamma$ for ∇V_{in}

$$\tilde{T}_d = 2f\gamma + 5f\gamma = 7f\gamma$$

* The relation b/w no. of inverters 'N' to width factor 'f' can be given as

$$N^f = \frac{C_L}{C_g} = y$$

$$N^f = y$$

Apply 'log' on both sides

$$f \log N = y$$

* If $f = e$, $e \log N = y$.

$$N \log e = y$$

$$N = y$$

* If $N = \text{even}$, then the delay is

$$\tau = \frac{N}{2} 5f\tau = 2.5f\tau \text{ for nmos}$$

$$\tau = \frac{N}{2} 7f\tau = 3.5f\tau \text{ for cmos}$$

* If $N = \text{odd}$, then the delay is

$$\tau = [2.5(N-1) + 4]f\tau \text{ for nmos } \left. \begin{array}{l} \Delta V_{in} \\ \Delta V_{in} \end{array} \right\}$$

$$\tau = [3.5(N-1) + 5]f\tau \text{ for cmos } \left. \begin{array}{l} \Delta V_{in} \\ \Delta V_{in} \end{array} \right\}$$

$$\tau = [2.5(N-1) + 1]f\tau \text{ for nmos } \left. \begin{array}{l} \Delta V_{in} \\ \Delta V_{in} \end{array} \right\}$$

$$\tau = [3.5(N-1) + 2]f\tau \text{ for cmos } \left. \begin{array}{l} \Delta V_{in} \\ \Delta V_{in} \end{array} \right\}$$

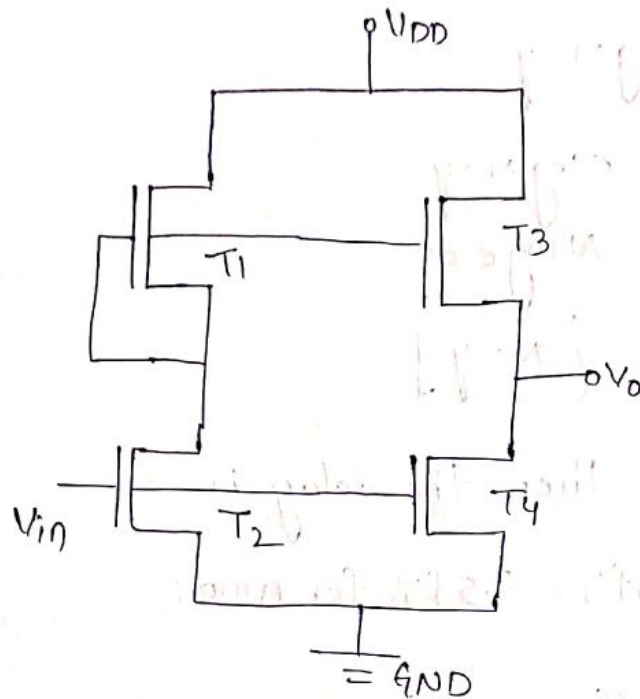
Note:- If there is increasing width factor 'f' then no. of transistors per chip will get reduced.

2. Superbuffers as drivers:- It is classified into two types.

1. Inverting type super buffers

2. Non inverting type super buffers

1. Inverting type Superbuffers:-



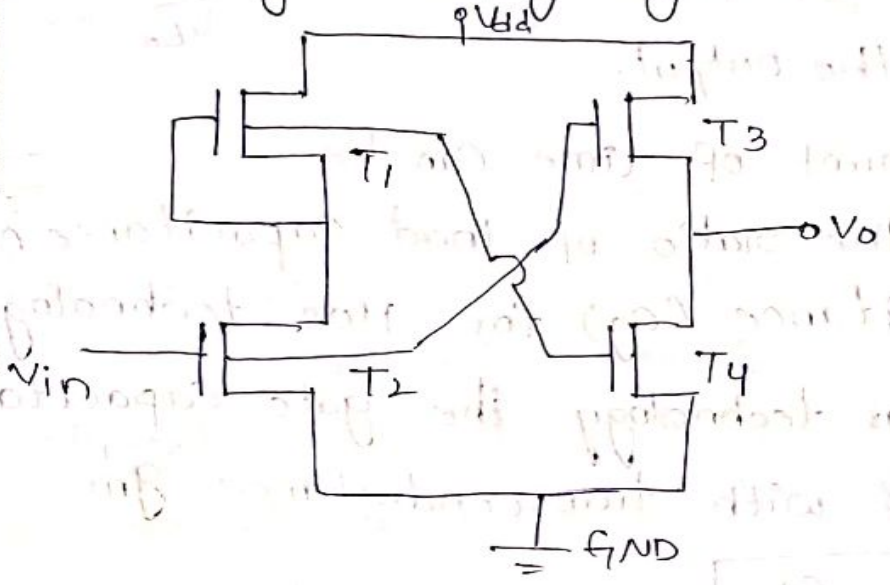
* When V_{in} is logic '0' then the transistors T_2 & T_4 will get off and the transistors T_1 & T_3 will get on, thereby producing o/p as logic '1'.

* If V_{in} is logic '1' then the transistors T_1 & T_3 will get on and the transistors T_2 & T_4 will get on thereby producing o/p as logic '0'.

2. Non-inverting type Superbuffers:-

* When V_{in} is logic '0' then the transistors T_1 & T_4 will get on and the transistors T_2 & T_3 will get off thereby producing o/p as logic '0'.

* when V_{in} is logic '1' then the transistors T_2 & T_3 will get on and the transistors T_1 & T_4 will get off and thereby producing logic '1'.



8/1/19
Bipolar Drivers:-

1. In Bipolar technology, ^{the} output drive current is more for a given minimum silicon area.

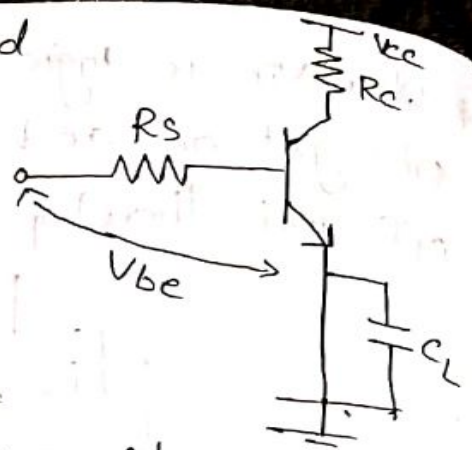
2. In this the transconductance g_m and low current driving capabilities and current/Area (or) more compared to MOS technology.

* in this the current I_c is exponentially related to the input voltage.

* in this technology it is having the capability of driving large currents for the application of smaller voltages. and the current through the device depends on the base width w_b and the amount of doping level.

A simple steps presentation that is use bipolar technology to change their states is shown below.

* The amount of time required to change the input is equal to amount of time taken to change the output.



* The amount of time can be given as the ratio of load capacitance (\$C_L\$) to gate capacitance (\$C_g\$) for MOS technology $\Delta t = \frac{C_L}{C_g}$.

* In Bipolar technology the gate capacitance '\$C_g\$' is replaced with transconductance '\$g_m\$'.

$$\Delta t = \frac{C_L}{g_m}$$

* Therefore the total time taken can be represented as follows

$$T = T_{in} + \left(\frac{V}{I}\right) C_L \frac{1}{h_{fe}}$$

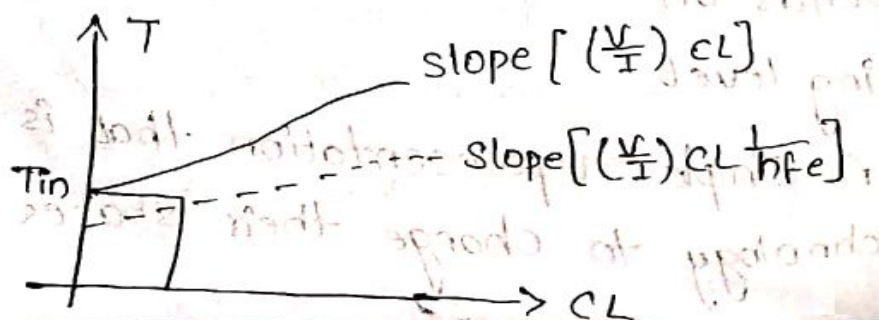
for bipolar

where \$T_{in}\$ - the inbuilt time produced by device

\$C_L\$ - load capacitance

\$h_{fe}\$ - current amplification factor.

The graphical representation of mos and bipolar technology are drawn as below



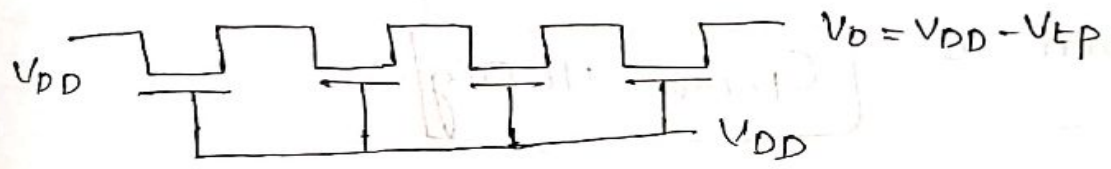
$$T = T_{in} + \left(\frac{V}{I}\right) C_L \text{ for CMOS}$$

propagation delays:-

To transfer logic levels from one place to another place we are using series of pass transistors in b/w two points :-

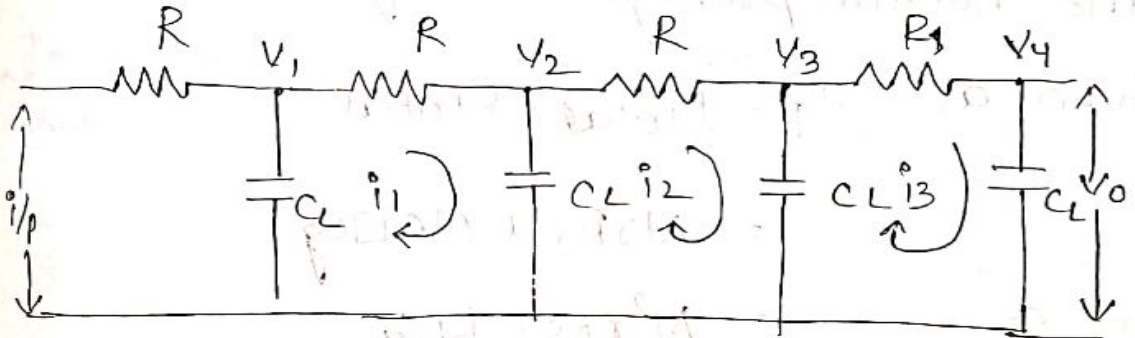
Here in this case nmos pass transistors are use in series connection, all the gate terminals are tight together and it is being given V_{DD} .

for example a series of four pass transistors are shown below.



Model for propagation delay

The equivalent circuit model can be drawn as



At node 2, we get write $C \frac{dV_2}{dt} = i_1 - i_2$

$$C \frac{dV_2}{dt} = \frac{V_1 - V_2}{R} - \frac{V_2 - V_3}{R}$$

$$C \frac{dV_2}{dt} = \frac{V_1 - 2V_2 + V_3}{R}$$

$$RC \frac{dV_2}{dt} = V_1 - 2V_2 + V_3$$

$$Rc \frac{dv}{dt} = \frac{d^2x}{dx^2}$$

$\therefore x \Rightarrow$ distance

\therefore The propagation delay $t_p \propto x^2$.

* For 'N' no of networks the total resistance

can be given as $R_{total} = N \gamma R_s$

where R_s - sheet Resistance

γ - relative Resistance

N - no. of stages

* For 'C' the total no of capacitance can be given as

$$C_{total} = N C \square C_g$$

where C -

* The overall propagation delay t_p' can be

given as $t_p' = R_{total} \times C_{total}$

$$= N \gamma R_s \times N C \square C_g$$

$$= N^2 \gamma R_s C \square C_g$$

$$= N^2 \gamma C R_s \square C_g$$

$$t_p = N^2 \gamma C \tau$$

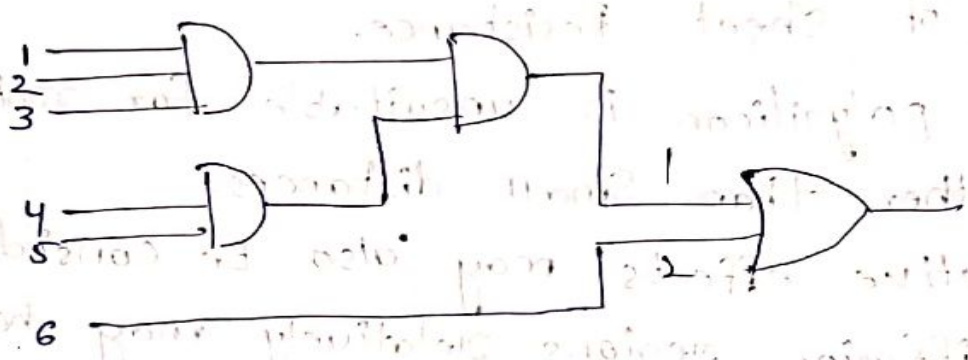
$$\tau = R_s \square C_g$$

The propagation delay is $\propto N^2$.
 If there is increase in N , it results in increased propagation delay.

Fan-in and fan-out characteristics:-

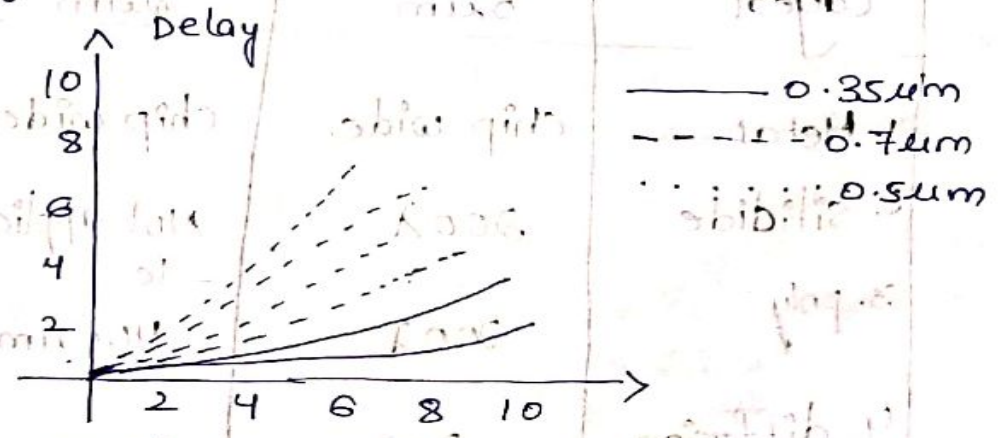
We have two major factors that influence operational speed of a gate terminal these are fan-in and fan-out.

Fan-in :- The maximum no. of inputs that are applied to driven in gate is called fan-in



Fan-out :- The max no. of i/p's that are applied to driven gate is called fan-out.

* The delay associated with Fan-in & Fan-out for three technologies is represented as



Fan-in & fan-out characteristics

Choice of layers :-

In designing circuits for our convenience of suitable specifications we have to consider several number of considerations which includes choice of layers

1. V_{DD} and V_{SS} should be distributed on metal layer whenever possible.
2. The length of polysilicon should be used after careful consideration because of relatively high value of sheet resistance.
3. The polysilicon is unsuitable for routing V_{DD} & V_{SS} other than small distances.
4. Capacitive effects may also be considered because the diffusion regions relatively may have high capacitive values to the substrate.

Table for electrical rules :-

Layers	Max length of wires		
	5 μ m	2 μ m	1.2 μ m
1. Metal	chip wide	chip wide	chip wide
2. silicide	200 λ	Not applicable	Not applicable
3. poly	200 λ	400 μ m	250 μ m
4. diffusion	20 λ	100 μ m	60 μ m

choice of layers:-

layer	capacitance	Resistance	comment
1. Metal	Low	Low	* Good current capability without large voltage drop and it is used for power distributions and global.
2. silicide	Moderate	Moderate	* It has RC product has a moderate value, long wires are applicable. this layer is useful in place of poly silicon in some cases of nmos process
3. poly silicon	Moderate	High	* It has RC product has IR moderate and high drop
4. diffusion (Active)	High	Moderate	* RC product is moderate and it has moderate IR drop hence it is hard to drive

Wiring Capacitance:-

We have Area Capacitance contributed in the Calculation of overall Calculation Capacitance.

The Area Capacitances are associated with the layers to substrate and from gate to channel.

We have three other source for the calculation of overall Capacitance.

1. Inter layer Capacitance

2. peripheral (Junction) Capacitance

3. (fringing field) fringing fields capacitance

1. Inter-layer Capacitance:-

parallel plate effects are present b/w one layer to another layer.

for example, for a given area metal to polysilicon capacitance is higher than metal to substrate capacitance.

2. peripheral Capacitance:-

The source and drains of n-diffusion regions forms junctions with p-type substrate at uniform depth.

Similarly, p-active regions may form junctions with n-well (or) n type of substrate.

* for diffusion regions each diode thus formed has associated with peripheral capacitance which is measured in PF/unit length.

The typical values for different technologies given by

diffusion Capacitance	Chemical Values		
	5 μm	2 μm	1.2 μm
1. C_{Area}	1.082×10^{-4} PF/ μm^2	1.25×10^{-4} PF/ μm^2	1.7×10^{-4} PF/ μm^2
2. $C_{\text{peripheral}}$	8×10^{-4} PF/ μm^2	negligible	negligible

3. fringing fields:-

Capacitance due to fringing field effect can be a major component of overall capacitance of inter connected wires.

fringing field capacitance can be of same order of area capacitance.

The capacitance of fringing field can be given as

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

where l = length of the wire

t = thickness of the wire

$d =$ Separation b/w wire and substrate.

* The total wire capacitance, be given as

$$C_w = C_{Area} + C_{ff}$$

where	$C_{Area} =$	Area	Capacitance
	$C_{ff} =$	fringing field	Capacitance

3.2 Scaling of Mos ckts

Micro electronic technology can be characterised with the help of several indicators (or) figure of merits which includes

1. No of transistors per chip
2. minimum feature size
3. power dissipation
4. max operational frequency
5. die size
6. production cost

* many of these fig of merits can be improved by reducing dimensions of transistors, interconnections and separation b/w features and by adjusting doping level and supply voltage.

Scaling Models and Scaling factors:-

Basically we are having two scaling models

1. constant electric field scaling model
2. constant voltage scaling model

In accordance with these two scaling models we are having a special scaling model which is the combination of both scaling models stated above and is called as combined voltage above and dimension scaling model.

The following fig indicates that substrate doping level which are associated with scaling of transistors.

* To scale any parameter we are using two scaling factors as $\frac{1}{\alpha}$ & $\frac{1}{\beta}$.

* $\frac{1}{\beta}$ is used for supply voltage levels (V_{dd}) and for gate oxide thickness (t_o) for all other linear dimensions we use $\frac{1}{\alpha}$ as a scaling factor for both horizontal and vertical dimensions.

Note:- For constant electric field scaling model we use $\beta = \alpha$ and for constant voltage scaling model $\beta = 1$.

Scaling factors for device parameters:-

1. Gate area (A_g):- $A_g = L \times W$

'L' is the length of the channel which is scaled by $\frac{1}{\alpha}$

and 'w' is the width of channel which is scaled by $1/\alpha$

$$A_g = L \times w \\ = \frac{1}{\alpha} \times \frac{1}{\alpha}$$

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

2. Gate Capacitance per unit area (C_{ox} or C_{ox}):-

$$C_x = \frac{\epsilon}{D}$$

where ϵ = permittivity

D = gate oxide thickness

$$C_x = \frac{\epsilon}{D} = \frac{\epsilon_0 \epsilon_{ins}}{D} = \frac{1}{1/\beta} = \beta$$

3. Gate Capacitance (C_g):- $C_g = C_{ox}WL$

where $C_{ox} \Rightarrow$ absolute capacitance and it is scaled by β .

$$C_g = C_{ox}WL = \beta WL = \beta \left(\frac{1}{\alpha}\right) \left(\frac{1}{\alpha}\right)$$

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

4. parasite Capacitance (C_x):-

$$C_x = \frac{A_x}{d}$$

A_x = Area, d = separation

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

5. Carrier density in the channel (Q_{on}):-

$$C = \frac{Q}{V}$$

$$C_{ox} V_{gs} = Q_{on}$$

$$R_{on} = \beta \cdot \frac{1}{\beta} = 1$$

6. channel ON Resistance :-

$$R_{on} = \frac{L}{W} = \frac{1/d}{1/d} = 1$$

7. gate delay (T_d) :-

$$\begin{aligned} T_d &= R_{on} C_g \\ &= 1 \cdot \beta/d^2 \\ &= \beta/d^2 \end{aligned}$$

8. Maximum operating frequency (f_o) :-

$$\begin{aligned} f_o &= \frac{W}{L} \frac{\mu C_o V_{dd}}{C_g} = \frac{\beta \cdot \frac{1}{\beta}}{\beta/d^2} = \frac{1}{\beta/d^2} \\ &= \frac{d^2}{\beta} \end{aligned}$$

9. Saturation Current (I_{ds}) :-

$$\begin{aligned} I_{ds} &= \frac{K W}{L} \frac{[V_{gs} - V_T]^2}{2} \\ &= 1 \left[\frac{1}{\beta} - \frac{1}{\beta} \right]^2 \\ &= 1/\beta^2 \end{aligned}$$

$$I_{ds} = \frac{C_o W \mu}{L} \frac{[V_{gs} - V_T]^2}{2}$$

$$= \beta \cdot \frac{1}{\beta^2}$$

$$= 1/\beta$$

10. Current density (J):-

$$J = \frac{I_{ds}}{A} = \frac{1/\beta}{1/d^2}$$
$$= \alpha^2/\beta$$

11. Switching energy per gate:-

$$E_g = \frac{1}{2} C_g V_{dd}^2$$
$$= \frac{1}{2} \frac{\beta}{\alpha^2} \cdot \frac{1}{\beta^2}$$
$$= \frac{1}{2\alpha^2\beta}$$

12. power dissipation per gate:-

$$P_g = P_{gs} + P_{gd} \quad P_{gs} = \frac{V_{dd}^2}{R_{on}}, \quad P_{gd} = E_g \cdot f_0$$

$$P_{gs} = \frac{1/\beta^2}{1} = 1/\beta^2 \quad ; \quad P_{gd} = 1/\beta^2$$

$$P_g = 2/\beta^2 = 1/\beta^2$$

13. power dissipation per unit area (PA):-

$$P_A = \frac{P_g}{\text{Area}} = \frac{1/\beta^2}{1/d^2}$$
$$= \alpha^2/\beta^2$$

14. Power Speed product:-

$$P_t = P_g T_d = 1/\beta^2 \cdot \beta/d^2$$
$$= 1/\alpha^2\beta$$

Scaling effects:-

S.No	parameter	Combined Voltage and dimension model	Constant electric field model $\beta = \alpha$	Constant Voltage Scaling model $\beta = 1$
1.	Supply voltage (V_{DD})	$1/\beta$	$1/\alpha$	1
2.	Channel length (L)	$1/\alpha$	$1/\alpha$	$1/\alpha$
3.	width of the channel (w)	$1/\alpha$	$1/\alpha$	$1/\alpha$
4.	gate oxide thickness (D)	$1/\beta$	$1/\alpha$	1
5.	gate Area (A_g)	$1/\alpha^2$	$\frac{1}{\alpha^2}$	$\frac{1}{\alpha^2}$
6.	gate capacitance per unit area (C_D)	β	α	1
7.	gate capacitance (C_g)	β/α^2	$\frac{\alpha}{\alpha^2} = \frac{1}{\alpha}$	$\frac{1}{\alpha^2}$
8.	parastic Capacitance	$\frac{1}{\alpha}$	$\frac{1}{\alpha}$	$\frac{1}{\alpha}$
9.	Carrier density in the channel (Q_{ON})	1	1	1
10.	channel ON Resistance	1	1	1
11.	gate delay	β/α^2	$1/\alpha$	$1/\alpha^2$
12.	maximum operating frequency	α^2/β	α	α^2

13.	Saturation Current	$1/\beta$	$1/d$	1
14.	Current density	α^2/β	α	α^2
15.	Switching energy per gate	$1/d^2\beta$	$1/d^3$	$1/d^2$
16.	power dissipation per gate	$1/\beta^2$	$1/d^2$	1
17.	power dissipation per unit area	α^2/β^2	1	α^2
18.	Power speed product	$1/d^2\beta$	$1/d^3$	$1/d^2$

Limitations of Scaling:-

Substrate Doping:-

So far we are discussed about various effects, we have neglected built in (junction) potential V_B which in turn depends on substrate doping level and this is acceptable so long as V_B is smaller compared to V_{DD} .

Substrate doping Scaling factors:-

As the length of the channel of a mos transistor is reduce, that depletion region width also to be scaled down to prevent source and drain depletions regions from meeting.

The depletion width 'd' for the junction can be given as

$$d = \sqrt{\frac{2\epsilon_0\epsilon_{ins} V_B}{qN_B}}$$

Where $q = \text{charge}$

$\epsilon_0 = \text{Permittivity of free Space}$

$\epsilon_{ins} = \text{Permittivity of material}$

$V_B = \text{V. built in potential}$

The inbuilt potential V_B can be given as

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

Where $N_D = \text{drain (or) Source doping level}$

$n_i = \text{intrinsic Carrier Concentration}$

* If we increase N_B to reduce d , at the same time V_B is also increase.

* For Combined Voltage & dimension model the total applied voltage can be given as

$$V_a = mV_B$$

Where m is a real number

then $V = V_a + V_B$

$$V = mV_B + V_B$$

$$V = V_B (m+1)$$

Now if is scale down V_a then the voltage can be given as

$$V_a = \frac{mV_B}{\beta}$$

$$V = \frac{mV_B}{\beta} + V_B$$

$$V = \frac{mV_B + \beta V_B}{\beta}$$

$$V_2 = \frac{V_B(m + \beta)}{\beta}$$

∴ The effective scale voltage can be given as

$$V_S = \frac{V_2}{V_1} = \frac{V_B(m + \beta)}{\beta V_B(m + 1)}$$

$$V_S = \frac{(m + \beta)}{\beta(m + 1)}$$

Limitations due to Sub threshold Currents (I_{sub}):
 one of the major concerns in the scaling of devices is the effect of sub threshold current I_{sub} which can be given as

$$I_{sub} \propto e^{\frac{(V_{GS} - V_T)}{kT/q}}$$

When a transistor is in off state, the value of $V_{GS} - V_T$ is negative and it should be as large as possible to minimize I_{sub} .

As the voltages are scaled down then the ratio of $V_{GS} - V_T$ to kT/q will reduce so that sub threshold increase.

- * For this reason, it may be desirable to scale both v_{gs} & v_t by a factor $1/b > 1/a$. Since 'a' is generally greater than 'b'.
- * The maximum electric field across the depletion region can be given as

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

- * The junction breakdown voltage can be given as

$$BV = \frac{\epsilon_0 \epsilon_{ins} E^2}{2qNB}$$

Note- Extra care is therefore, require in estimating the breakdown voltages for scaled devices.

- * The electric field are greater and breakdown voltage is greater at the corners of diffusion regions underlying SiO_2 .

Limitations on Logic levels and Supply voltages

due to noise :-

The major advantages in scaling of devices are smaller gate delay time i.e., higher operating frequency and lower internal power consumption.

- * The lowering of interface spacing and higher switching increase noise in VLSI chips. so noise may also be amplified and is thus a major concern.

* The mean square current fluctuations in the channel can be given as

$$i^2 = 4kT R_{ngm} \Delta f \rightarrow (1)$$

where $R_n =$ noise Resistance

$\Delta f =$ Band width

$g_m =$ BVp

$V_p =$ pinch off Voltage

The equivalent Resistance R_n can be given as

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

where $V_g' = V_{gs} - V_t + V_B$

$V_p' = V_p + V_B$

Similarly the thermal equivalent ' R_{ngm} ' can be given as

$$R_{ngm} = \frac{1}{2} \frac{(V_{gs} - V_t + V_B)}{(V_p + V_B)} + \frac{1}{6} \rightarrow (3)$$

Observing eqn(3) the value ' R_{ngm} ' is also dependent on V_g but very small extent. τ_{ox}

The modified expression for R_{ngm} when τ_{ox} is scaled as

$$R_{ngm}^{\tau_{ox}} = \frac{1}{2} \left[\frac{V_g}{V_g - \frac{1}{2} \left(\frac{a}{C_{ox}} \right)^2 \left(1 + \frac{V_g' C_{ox}}{a} \right)^{1/2} - 1} \right] + \frac{1}{6}$$

* If there is an increase in the value of C_{ox} then R_{ngm} decreases by a small amount which in turn decrease the ratio of logic levels to thermal noise by same amount.

29/1/19

Switch logic:-

Switch logic is based on pass transistors (or) on the transmission gates.

* This approach is fast for small arrays and takes no static current from supply rails.

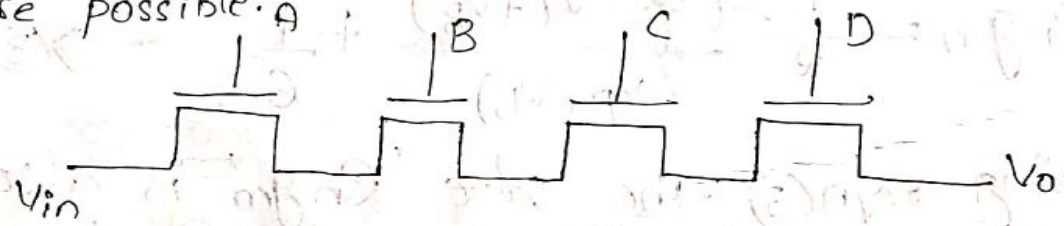
* Hence power dissipation such arrays is small.

∴ Current flow only on switching.

* pass transistor logic is similar to logic arrays based on delayed contacts.

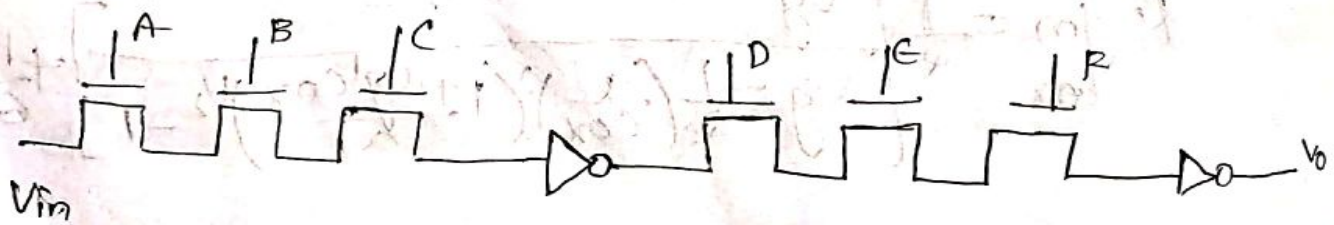
* The basic AND connections are set out as shown in fig below but many combinations of switches

are possible.



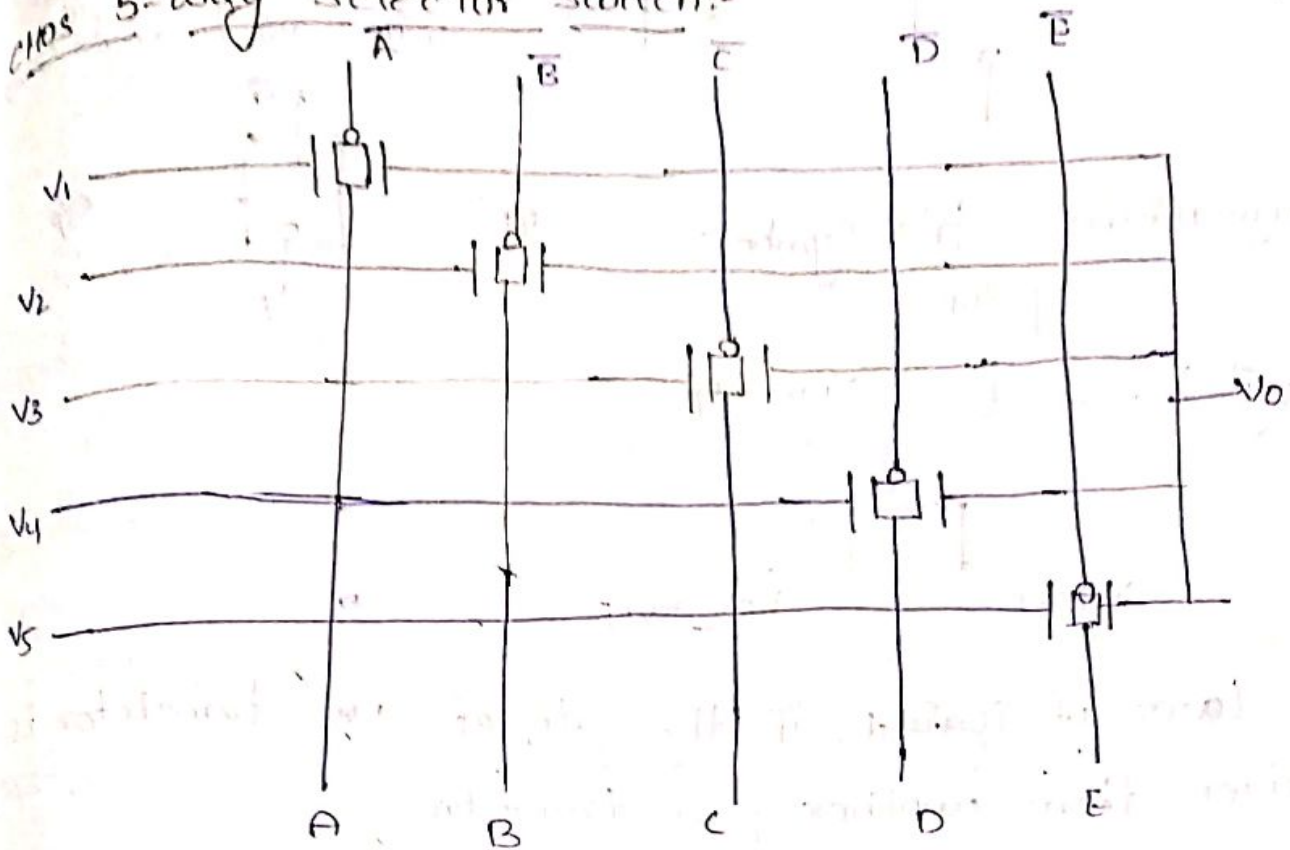
$V_o = V_{in}$ when $ABCD = 1$

∴ here V_o logic levels will be degraded by V_t effects.



$$V_o = V_{in} \text{ when } ABCDEF = 1$$

5-way Selector Switch:-

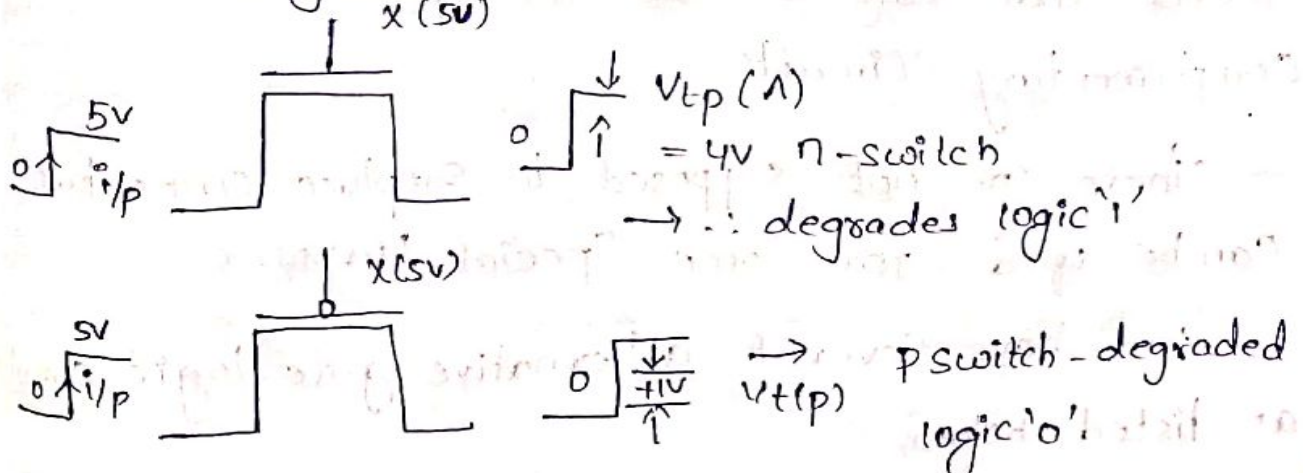


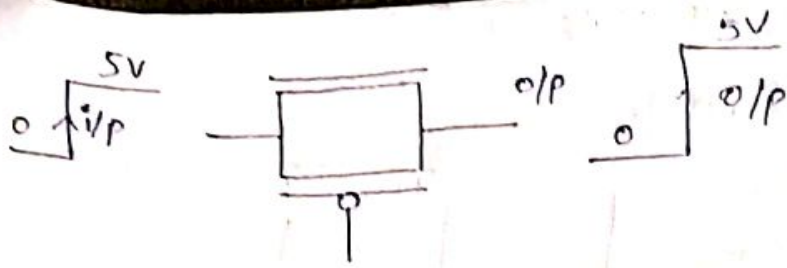
$$V_o = V_1A + V_2B + V_3C + V_4D + V_5E$$

Assuming A, B, C, D, E are mutually exclusive i.e., V_{out} logic levels are not degraded by ' V_t ' effect.

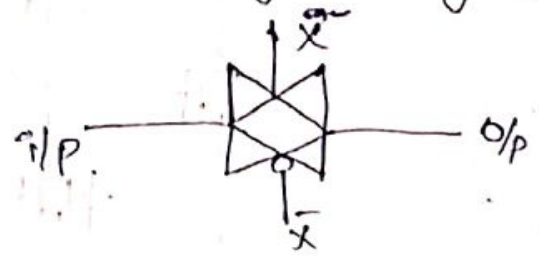
Pass transistor and Transmission gates:-

Switches and switch logics may be formed from simple 'n' (or) p-type pass transistors in parallel as shown in figure below.

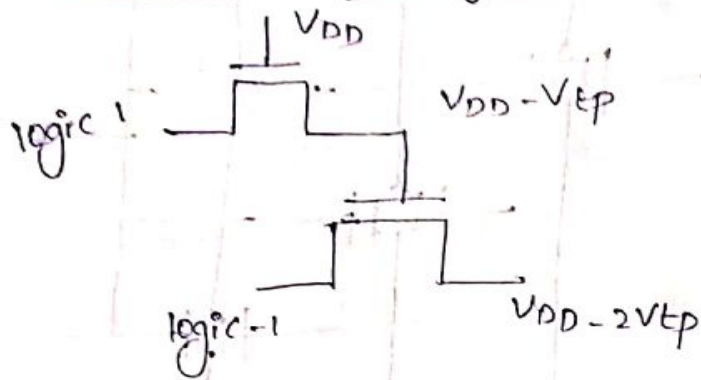




→ Transmission gate with good logic levels



Transmission Gate Symbol:-



Losses of logic-1, if the gate of pass transistor is driven from another pass transistor.

fig:- Some properties of pass transistor and some logic families.

Alternative Gate Circuits (or) Gate Logic:-

CMOS circuits suffer from increased area and corresponding increasing capacitance and delays logic gates becomes more complicate.

For this reason, the designers develop the circuits that can be used to supplement the complimentary circuits.

There are not supposed to replace CMOS but can be used for some special purposes.

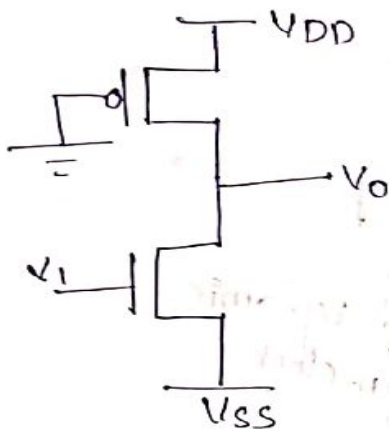
We have several alternative gate logic circuits as listed below,

- (1) pseudo nmos
- (2) dynamic
- (3) c²mos (clocked cmos)
- (4) domino logic
- (5) np cmos logic

pseudo nmos logic:-

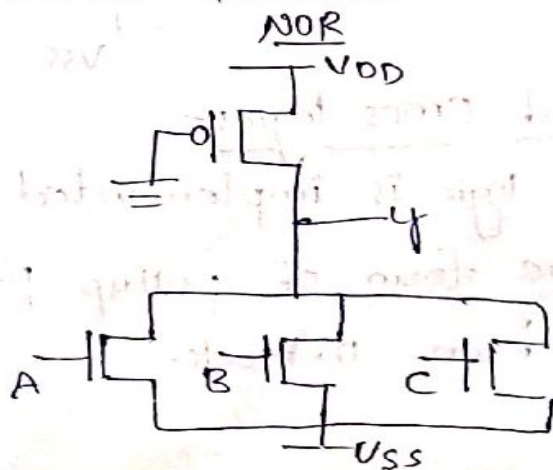
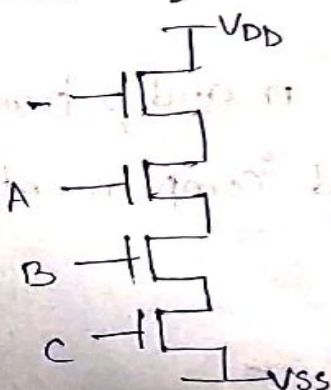
The pseudo nmos logic is one of the type of alternative gate circuits i.e., use to supplement for cmos circuits.

In this pseudo nmos circuit the depletion mode pull up mos transistor is replaced with p-mos transistor whose gate terminal is always ground.



Implementation of 3 Input NAND gate & NOR gates

Implementation of three i/p NOR gate and NAND gate using pseudo nmos logic:-



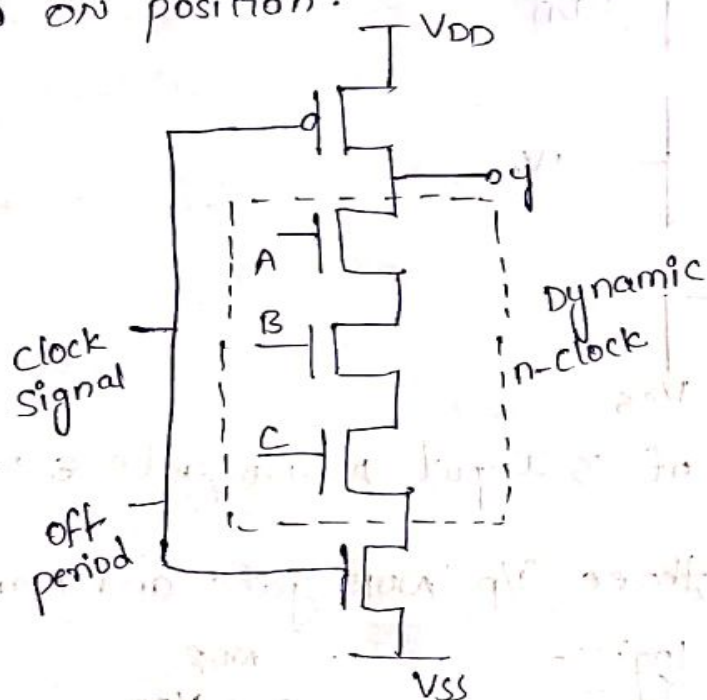
Note:- For 'n' numbers of i/p Pseudo nmos logic requires 'n+1' numbers of transistors Cmos logic require '2n' numbers of transistor.

Dynamic Cmos logic-

The actual logic is implemented in the n-block and P transistor is used for non-time critical pre-charging output.

i.e The output capacitance is charged to VDD during off period of clock signal (ϕ).

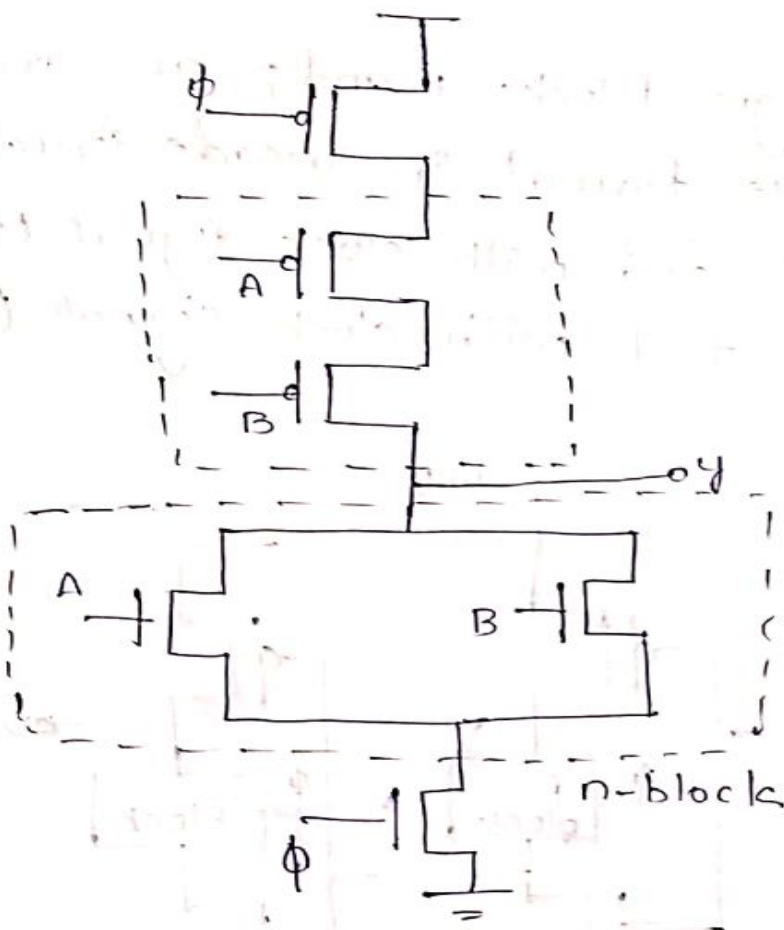
During this time, inputs are applied to n-block and state of logic is then evaluated during on period of clock... when the bottom n-transistor is in on position.



Clocked Cmos logic-

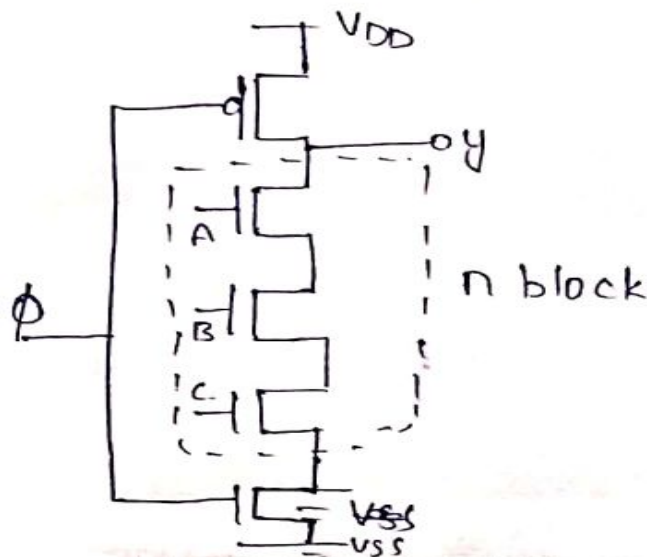
The logic is implemented in both n and p transistors in the form of pullup p-block and complimentary pull down n-block.

The logic in this can is evaluated only during the on period of clock.



Domino CMOS logic:-

An extension of dynamic CMOS logic is called domino CMOS logic. This is a modified arrangement that allows cascading of logic structures using only a single phase clock. So, at the output we use a buffer.



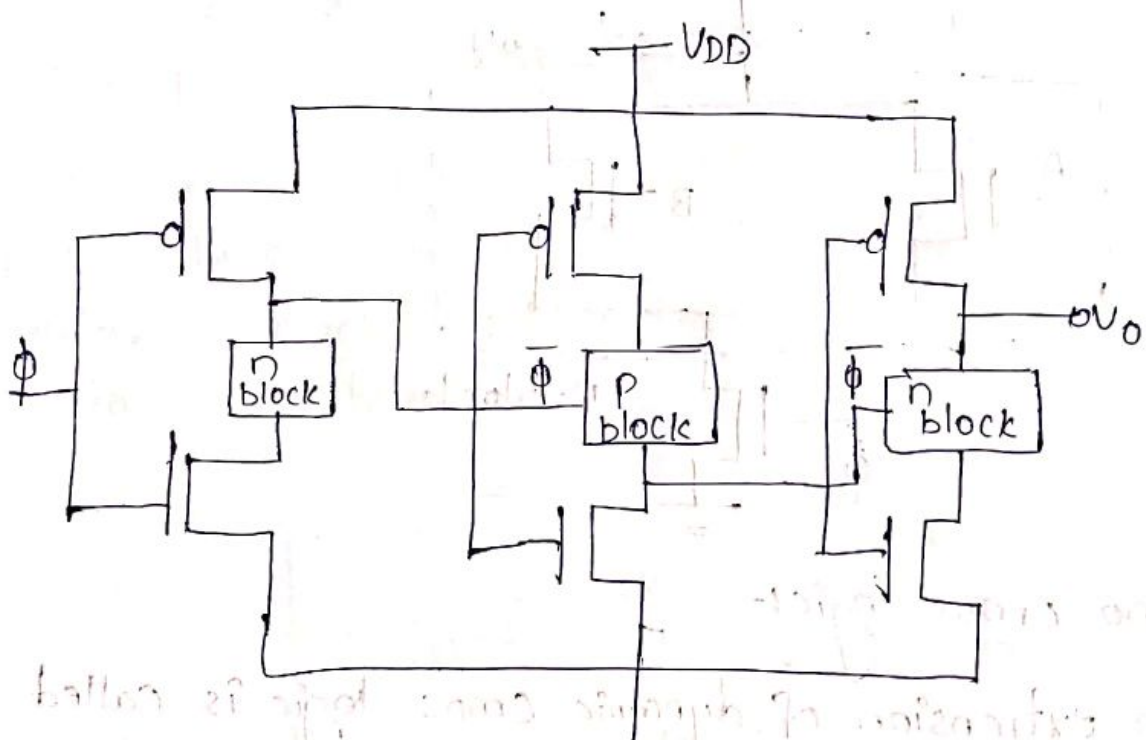
n_p cmos logic

This is another version of basic dynamic logic circuit.

Circuit:

The actual logic blocks n and p are arranged in the alternative format in cascade structure.

One block is fed with clock signal (ϕ) and another block is fed with clock signal ($\bar{\phi}$).



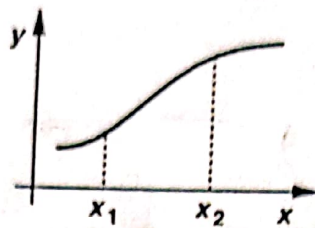


Figure 3.1 Input-output characteristic of a nonlinear system.

approximation, and higher order terms are insignificant. In other words, $\Delta y = a_1 \Delta x$, indicating a linear relationship between the *increments* at the input and output. As $x(t)$ increases in magnitude, higher order terms manifest themselves, leading to nonlinearity and necessitating large-signal analysis. From another point of view, if the slope of the characteristic (the increment $\Delta y / \Delta x$) varies with the signal level, then the system is nonlinear. These concepts are described in detail in Chapter 13.

What aspects of the performance of an amplifier are important? In addition to gain and speed, such parameters as power dissipation, supply voltage, linearity, noise, or maximum voltage swings may be important. Furthermore, the input and output impedances determine how the circuit interacts with preceding and subsequent stages. In practice, most of these parameters trade with each other, making the design a multi-dimensional optimization problem. Illustrated in the "analog design octagon" of Fig. 3.2, such trade-offs present many challenges in the design of high-performance amplifiers, requiring intuition and experience to arrive at an acceptable compromise.

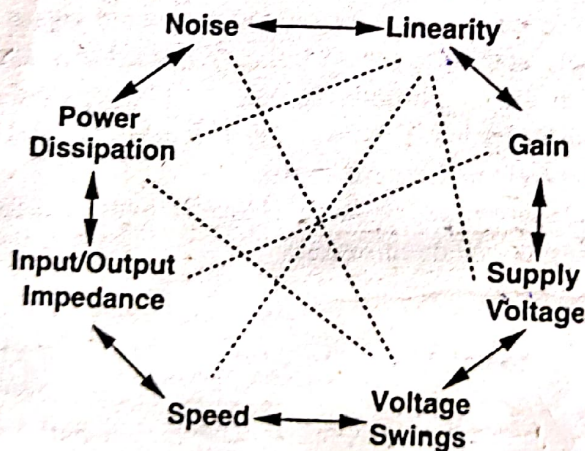


Figure 3.2 Analog design octagon.

3.2 Common-Source Stage

3.2.1 Common-Source Stage with Resistive Load

By virtue of its transconductance, a MOSFET converts variations in its gate-source voltage to a small-signal drain current, which can pass through a resistor to generate an output voltage. Shown in Fig. 3.3(a), the common-source (CS) stage performs such an operation.

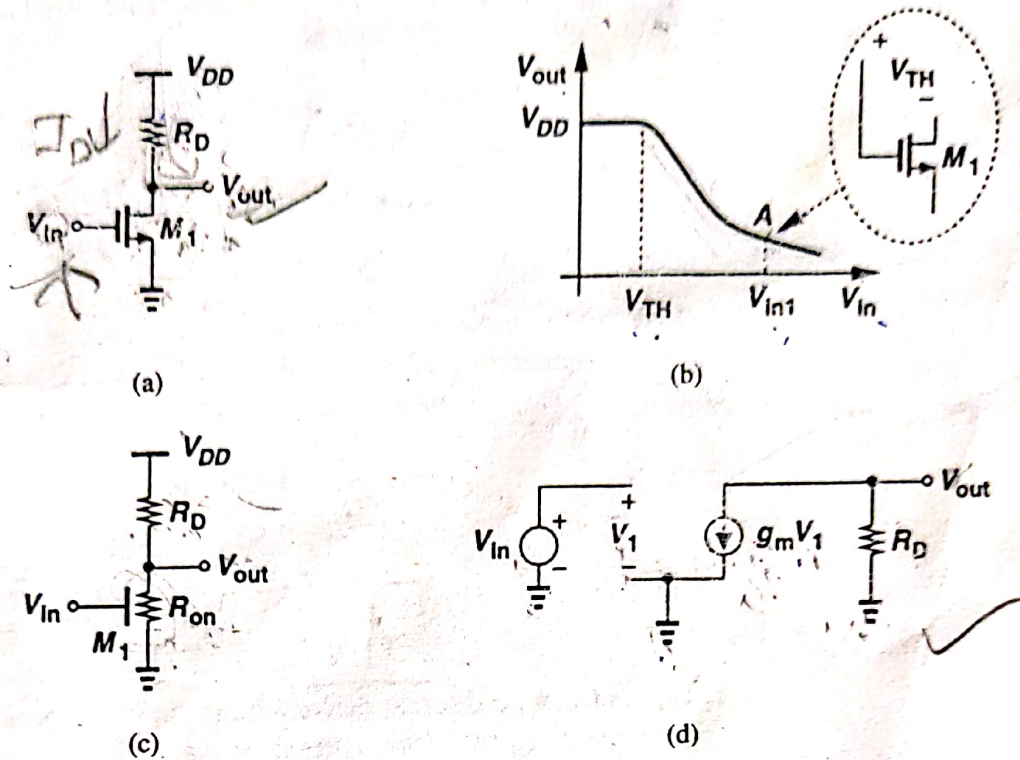


Figure 3.3 (a) Common-source stage, (b) input-output characteristic, (c) equivalent circuit in deep triode region, (d) small-signal model for the saturation region.

We study both the large-signal and the small-signal behavior of the circuit. Note that the input impedance of the circuit is very high at low frequencies.

If the input voltage increases from zero, M_1 is off and $V_{out} = V_{DD}$ [Fig. 3.3(b)]. As V_{in} approaches V_{TH} , M_1 begins to turn on, drawing current from R_D and lowering V_{out} . If V_{DD} is not excessively low, M_1 turns on in saturation, and we have

$$V_{out} = V_{DD} - R_D \frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_{in} - V_{TH})^2 \tag{3.3}$$

where channel-length modulation is neglected. With further increase in V_{in} , V_{out} drops more and the transistor continues to operate in saturation until V_{in} exceeds $\sqrt{V_{out}}$ by V_{TH} [point A in Fig. 3.3(b)]. At this point,

$$V_{in1} - V_{TH} = \sqrt{V_{DD} - R_D \frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_{in1} - V_{TH})^2} \tag{3.4}$$

from which $V_{in1} - V_{TH}$ and hence V_{out} can be calculated.

For $V_{in} > V_{in1}$, M_1 is in the triode region:

$$V_{out} = V_{DD} - R_D \frac{1}{2} \mu_n C_{ox} \frac{W}{L} [2(V_{in} - V_{TH})V_{out} - V_{out}^2] \tag{3.5}$$

Handwritten note: $g_m(V_{in} - V_{TH})V_{out} - V_{out}^2$

If V_{in} is high enough to drive M_1 into deep triode region, $V_{out} \ll 2(V_{in} - V_{TH})$, and, from the equivalent circuit of Fig. 3.3(c),

$$V_{out} = V_{DD} \frac{R_{on}}{R_{on} + R_D} \quad (3.6)$$

$$= \frac{V_{DD}}{1 + \mu_n C_{ox} \frac{W}{L} R_D (V_{in} - V_{TH})} \quad (3.7)$$

Since the transconductance drops in the triode region, we usually ensure that $V_{out} > V_{in} - V_{TH}$, operating to the left of point A in Fig. 3.3(b). Using (3.3) as the input-output characteristic and viewing its slope as the small-signal gain, we have:

$$A_v = \frac{\partial V_{out}}{\partial V_{in}} \quad (3.8)$$

$$= -R_D \mu_n C_{ox} \frac{W}{L} (V_{in} - V_{TH}) \quad (3.9)$$

$$A_v = -g_m R_D \quad (3.10)$$

This result can be directly derived from the observation that M_1 converts an input voltage change ΔV_{in} to a drain current change $g_m \Delta V_{in}$, and hence an output voltage change $-g_m R_D \Delta V_{in}$. The small-signal model of Fig. 3.3(d) yields the same result.

Even though derived for small-signal operation, the equation $A_v = -g_m R_D$ predicts certain effects if the circuit senses a *large* signal swing. Since g_m itself varies with the input signal according to $g_m = \mu_n C_{ox} (W/L) (V_{GS} - V_{TH})$, the gain of the circuit changes substantially if the signal is large. In other words, if the gain of the circuit *varies* significantly with the signal swing, then the circuit operates in the large-signal mode. The dependence of the gain upon the signal level leads to nonlinearity (Chapter 13), usually an undesirable effect.

A key result here is that to minimize the nonlinearity, the gain equation must be a weak function of signal-dependent parameters such as g_m . We present several examples of this concept in this chapter and in Chapter 13.

Example 3.1

Sketch the drain current and transconductance of M_1 in Fig. 3.3(a) as a function of the input voltage.

Solution

The drain current becomes significant for $V_{in} > V_{TH}$, eventually approaching V_{DD}/R_D if $R_{on1} \ll R_D$ [Fig. 3.4(a)]. Since in saturation, $g_m = \mu_n C_{ox} (W/L) (V_{in} - V_{TH})$, the transconductance begins to rise for $V_{in} > V_{TH}$. In the triode region, $g_m = \mu_n C_{ox} (W/L) V_{DS}$, falling as V_{in} exceeds V_{in1} [Fig. 3.4(b)].

How do we maximize the voltage gain of a common-source stage? Writing (3.10) as

$$A_v = -\sqrt{2\mu_n C_{ox} \frac{W}{L} I_D} \frac{V_{RD}}{I_D} \quad (3.11)$$

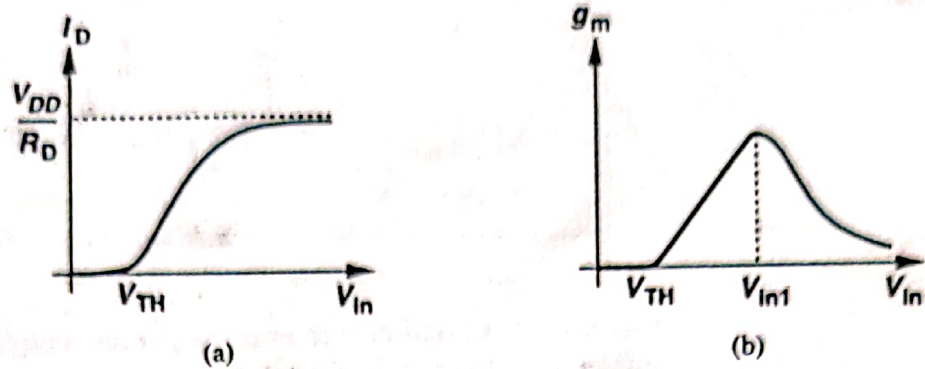


Figure 3.4

where V_{RD} denotes the voltage drop across R_D , we have

$$A_v = -\sqrt{2\mu_n C_{ox} \frac{W}{L} \frac{V_{RD}}{I_D}} \quad (3.12)$$

Thus, the magnitude of A_v can be increased by increasing W/L or V_{RD} or decreasing I_D if other parameters are constant. It is important to understand the trade-offs resulting from this equation. A larger device size leads to greater device capacitances, and a higher V_{RD} limits the maximum voltage swings. For example, if $V_{DD} - V_{RD} = V_{in} - V_{TH}$, then M_1 is at the edge of the triode region, allowing only very small swings at the output (and input). If V_{RD} remains constant and I_D is reduced, then R_D must increase, thereby leading to a greater time constant at the output node. In other words, as noted in the analog design octagon, the circuit exhibits trade-offs between gain, bandwidth, and voltage swings. Lower supply voltages further tighten these trade-offs.

For large values of R_D , the effect of channel length modulation in M_1 becomes significant. Modifying (3.4) to include this effect,

$$V_{out} = V_{DD} - R_D \frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_{in} - V_{TH})^2 (1 + \lambda V_{out}) \quad (3.13)$$

we have

$$\frac{\partial V_{out}}{\partial V_{in}} = -R_D \mu_n C_{ox} \frac{W}{L} (V_{in} - V_{TH})(1 + \lambda V_{out}) - R_D \frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_{in} - V_{TH})^2 \lambda \frac{\partial V_{out}}{\partial V_{in}} \quad (3.14)$$

Using the approximation $I_D \approx (1/2)\mu_n C_{ox}(W/L)(V_{in} - V_{TH})^2$, we obtain:

$$A_v = -R_D g_m - R_D I_D \lambda A_v \quad (3.15)$$

$$A_v + R_D I_D \lambda A_v = -R_D g_m$$

$$A_v [1 + R_D I_D \lambda] = -R_D g_m$$

$$A_v = -g_m R_D / [1 + R_D I_D \lambda]$$

and hence

$$A_v = -\frac{g_m R_D}{1 + R_D \lambda I_D} = -\frac{g_m R_D}{1 + R_D \lambda \left[\frac{1}{r_o} \right]} \quad (3.16)$$

Since $\lambda I_D = 1/r_o$,

$$A_v = -g_m \frac{r_o R_D}{r_o + R_D} \quad (3.17)$$

The small-signal model of Fig. 3.5 gives the same result with much less effort. That is, since

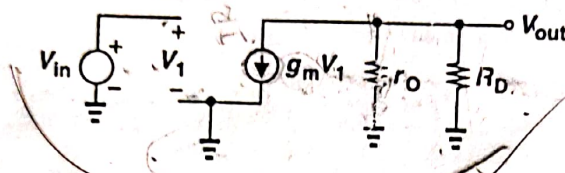


Figure 3.5 Small-signal model of CS stage including the transistor output resistance.

$g_m V_1 (r_o \parallel R_D) = -V_{out}$ and $V_1 = V_{in}$, we have $V_{out}/V_{in} = -g_m (r_o \parallel R_D)$. Note that, as mentioned in Chapter 1, V_{in} , V_1 , and V_{out} in this figure denote small-signal quantities.

Example 3.2

Assuming M_1 in Fig. 3.6 is biased in saturation, calculate the small-signal voltage gain of the circuit.

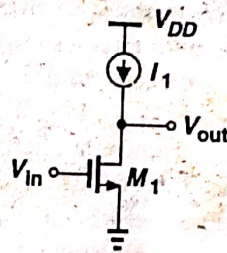


Figure 3.6

Solution

Since I_1 introduces an infinite impedance, the gain is limited by the output resistance of M_1 :

$$A_v = -g_m r_o. \quad (3.18)$$

Called the "intrinsic gain" of a transistor, this quantity represents the maximum voltage gain that can be achieved using a single device. In today's CMOS technology, $g_m r_o$ of short-channel devices is between roughly 10 and 30. Thus, we usually assume $1/g_m \ll r_o$.

In Fig. 3.6, Kirchhoff's current law (KCL) requires that $I_{D1} = I_1$. Then, how can V_{in} change the current of M_1 if I_1 is constant? Writing the total drain current of M_1 as

$$I_{D1} = \frac{1}{2} \mu_n C_{ox} (V_{in} - V_{TH})^2 (1 + \lambda V_{out}) \quad (3.19)$$

$$= I_1, \quad (3.20)$$

we note that V_{in} appears in the square term and V_{out} in the linear term. As V_{in} increases, V_{out} must decrease such that the product remains constant. We may nevertheless say " I_{D1} increases as V_{in} increases." This statement simply refers to the quadratic part of the equation.

3.2.2 CS Stage with Diode-Connected Load

In many CMOS technologies, it is difficult to fabricate resistors with tightly-controlled values or a reasonable physical size (Chapter 17). Consequently, it is desirable to replace R_D in Fig. 3.3(a) with a MOS transistor.

A MOSFET can operate as a small-signal resistor if its gate and drain are shorted [Fig. 3.7(a)]. Called a "diode-connected" device in analogy with its bipolar counterpart,

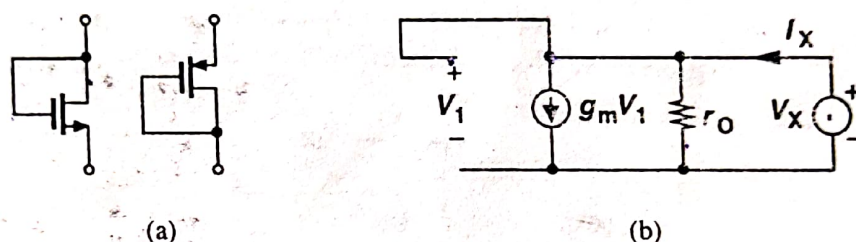


Figure 3.7 (a) Diode-connected NMOS and PMOS devices, (b) small-signal equivalent circuit.

this configuration exhibits a small-signal behavior similar to a two-terminal resistor. Note that the transistor is always in saturation because the drain and the gate have the same potential. Using the small-signal equivalent shown in Fig. 3.7(b) to obtain the impedance of the device, we write $V_1 = V_X$ and $I_X = V_X/r_o + g_m V_X$. That is, the impedance of the diode is simply equal to $(1/g_m) \parallel r_o \approx 1/g_m$. If body effect exists, we can use the circuit in Fig. 3.8 to write $V_1 = -V_X$, $V_{bs} = -V_X$ and

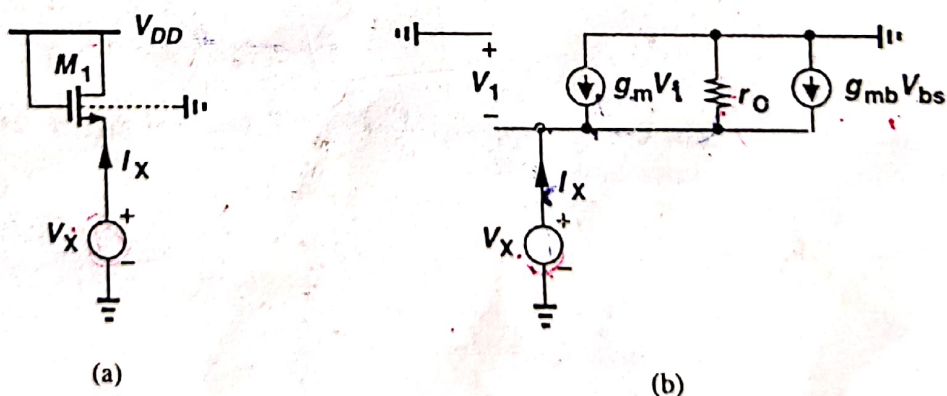


Figure 3.8 (a) Arrangement for measuring the equivalent resistance of a diode-connected MOSFET, (b) small-signal equivalent circuit.

$$(g_m + g_{mb})V_X + \frac{V_X}{r_o} = I_X. \tag{3.21}$$

It follows that

$$\frac{V_x}{I_x} = \frac{1}{g_m + g_{mb} + r_o^{-1}} \quad (3.22)$$

$$= \frac{1}{g_m + g_{mb}} \parallel r_o \quad (3.23)$$

$$\approx \frac{1}{g_m + g_{mb}} \quad (3.24)$$

Interestingly, the impedance seen at the source of M_1 is *lower* when body effect is included. Intuitive explanation of this effect is left as an exercise for the reader.

We now study a common-source stage with a diode-connected load (Fig. 3.9). For negligible channel-length modulation, (3.24) can be substituted in (3.10) for the load impedance,

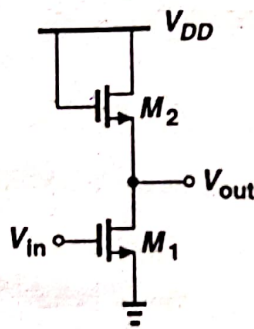


Figure 3.9 CS stage with diode-connected load.

yielding

$$A_v = -g_{m1} \frac{1}{g_{m2} + g_{mb2}} \quad (3.25)$$

$$= -\frac{g_{m1}}{g_{m2}} \frac{1}{1 + \eta} \quad (3.26)$$

where $\eta = g_{mb2}/g_{m2}$. Expressing g_{m1} and g_{m2} in terms of device dimensions and bias currents, we have

$$A_v = -\frac{\sqrt{2\mu_n C_{ox}} (W/L)_1 I_{D1}}{\sqrt{2\mu_n C_{ox}} (W/L)_2 I_{D2}} \frac{1}{1 + \eta} \quad (3.27)$$

and, since $I_{D1} = I_{D2}$,

$$A_v = -\sqrt{\frac{(W/L)_1}{(W/L)_2}} \frac{1}{1 + \eta} \quad (3.28)$$

This equation reveals an interesting property: if the variation of η with the output voltage is neglected, the gain is independent of the bias currents and voltages (so long as M_1 stays in saturation). In other words, as the input and output signal levels vary, the gain remains relatively constant, indicating that the input-output characteristic is relatively linear.

The linear behavior of the circuit can also be confirmed by large-signal analysis. Neglecting channel-length modulation for simplicity, we have in Fig. 3.9

$$\frac{1}{2}\mu_n C_{ox} \left(\frac{W}{L}\right)_1 (V_{in} - V_{TH1})^2 = \frac{1}{2}\mu_n C_{ox} \left(\frac{W}{L}\right)_2 (V_{DD} - V_{out} - V_{TH2})^2, \quad (3.29)$$

and hence

$$\sqrt{\left(\frac{W}{L}\right)_1} (V_{in} - V_{TH1}) = \sqrt{\left(\frac{W}{L}\right)_2} (V_{DD} - V_{out} - V_{TH2}). \quad (3.30)$$

Thus, if the variation of V_{TH2} with V_{out} is small, the circuit exhibits a linear input-output characteristic. The small-signal gain can also be computed by differentiating both sides with respect to V_{in} :

$$\sqrt{\left(\frac{W}{L}\right)_1} = \sqrt{\left(\frac{W}{L}\right)_2} \left(-\frac{\partial V_{out}}{\partial V_{in}} - \frac{\partial V_{TH2}}{\partial V_{in}}\right), \quad (3.31)$$

which, upon application of the chain rule $\partial V_{TH2}/\partial V_{in} = (\partial V_{TH2}/\partial V_{out})(\partial V_{out}/\partial V_{in}) = \eta(\partial V_{out}/\partial V_{in})$, reduces to

$$\frac{\partial V_{out}}{\partial V_{in}} = -\sqrt{\frac{(W/L)_1}{(W/L)_2}} \frac{1}{1 + \eta}. \quad (3.32)$$

It is instructive to study the overall large-signal characteristic of the circuit as well. But let us first consider the circuit shown in Fig. 3.10(a). What is the final value of V_{out} if I_1 drops to zero? As I_1 decreases, so does the overdrive of M_2 . Thus, for small I_1 , $V_{GS2} \approx V_{TH2}$ and $V_{out} \approx V_{DD} - V_{TH2}$. In reality, the subthreshold conduction in M_2 eventually brings V_{out} to V_{DD} if I_D approaches zero, but at very low current levels, the finite capacitance at the output node slows down the change from $V_{DD} - V_{TH2}$ to V_{DD} . This is illustrated in the time-domain waveforms of Fig. 3.10(b). For this reason, in circuits that have frequent switching activity, we assume V_{out} remains around $V_{DD} - V_{TH2}$ when I_1 falls to small values.

Now we return to the circuit of Fig. 3.9. Plotted in Fig. 3.11 versus V_{in} , the output voltage equals $V_{DD} - V_{TH2}$ if $V_{in} < V_{TH1}$. For $V_{in} > V_{TH1}$, Eq. (3.30) holds and V_{out} follows an approximately straight line. As V_{in} exceeds $V_{out} + V_{TH1}$ (beyond point A), M_1 enters the triode region, and the characteristic becomes nonlinear.

The diode-connected load of Fig. 3.9 can be implemented with a PMOS device as well. Shown in Fig. 3.12, the circuit is free from body effect, providing a small-signal voltage gain equal to

$$A_v = -\sqrt{\frac{\mu_n(W/L)_1}{\mu_p(W/L)_2}}, \quad (3.33)$$

where channel-length modulation is neglected.

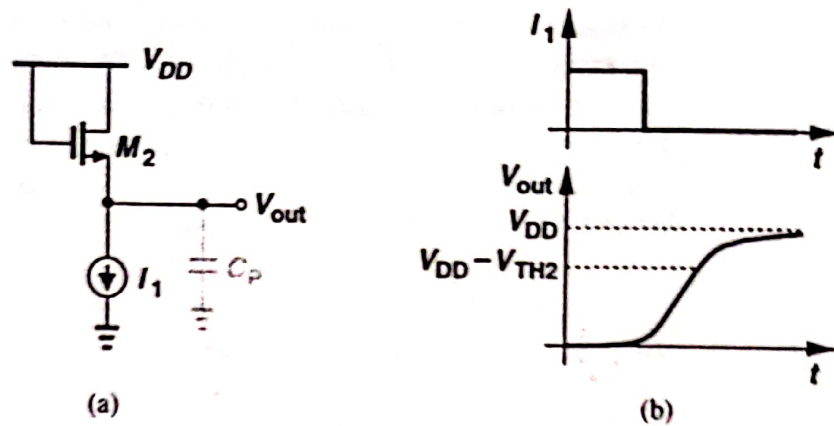


Figure 3.10 (a) Diode-connected device with stepped bias current, (b) variation of source voltage versus time.

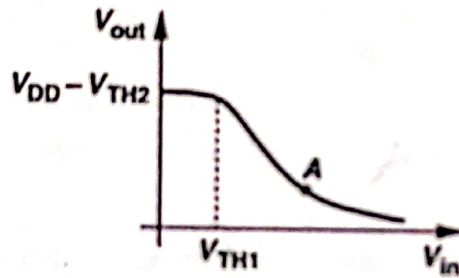


Figure 3.11 Input-output characteristic of a CS stage with diode-connected load.

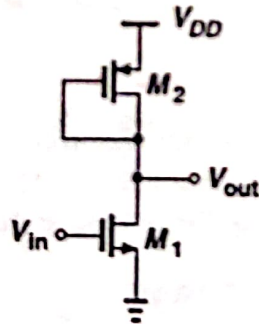


Figure 3.12 CS stage with diode-connected PMOS device.

Equations (3.28) and (3.33) indicate that the gain of a common-source stage with diode-connected load is a relatively weak function of the device dimensions. For example, to achieve a gain of 10, $\mu_n(W/L)_1/[\mu_p(W/L)_2] = 100$, implying that, with $\mu_n \approx 2\mu_p$, we must have $(W/L)_1 \approx 50(W/L)_2$. In a sense, a high gain requires a “strong” input device and a “weak” load device. In addition to disproportionately wide or long transistors (and hence a large input or load capacitance), a high gain translates to another important limitation: reduction in allowable voltage swings. Specifically, since in Fig. 3.12, $I_{D1} = |I_{D2}|$,

$$\mu_n \left(\frac{W}{L} \right)_1 (V_{GS1} - V_{TH1})^2 \approx \mu_p \left(\frac{W}{L} \right)_2 (V_{GS2} - V_{TH2})^2, \quad (3.34)$$

revealing that

$$\frac{|V_{GS2} - V_{TH2}|}{V_{GS1} - V_{TH1}} \approx A_v. \quad (3.35)$$

In the above example, the overdrive voltage of M_2 must be 10 times that of M_1 . For example, with $V_{GS1} - V_{TH1} = 200$ mV, and $|V_{TH2}| = 0.7$ V, we have $|V_{GS2}| = 2.7$ V, severely limiting the output swing. This is another example of the trade-offs suggested by the analog design octagon. Note that, with diode-connected loads, the swing is constrained by both the required overdrive voltage and the threshold voltage. That is, even with a small overdrive, the output level cannot exceed $V_{DD} - |V_{TH}|$.

An interesting paradox arises here if we write $g_m = \mu C_{ox}(W/L)|V_{GS} - V_{TH}|$. The voltage gain of the circuit is then given by

$$A_v = \frac{g_{m1}}{g_{m2}} \quad (3.36)$$

$$= \frac{\mu_n C_{ox}(W/L)_1 (V_{GS1} - V_{TH1})}{\mu_p C_{ox}(W/L)_2 |V_{GS2} - V_{TH2}|}. \quad (3.37)$$

Equation (3.37) implies that A_v is *inversely* proportional to $|V_{GS2} - V_{TH2}|$. It is left for the reader to resolve the seemingly opposite trends suggested by (3.35) and (3.37).

Example 3.3

In the circuit of Fig. 3.13, M_1 is biased in saturation with a drain current equal to I_1 . The current source $I_S = 0.75I_1$ is added to the circuit. How is (3.35) modified for this case?

Solution

Since $|I_{D2}| = I_1/4$, we have

$$A_v \approx -\frac{g_{m1}}{g_{m2}} \quad (3.38)$$

$$= -\sqrt{\frac{4\mu_n(W/L)_1}{\mu_p(W/L)_2}}. \quad (3.39)$$

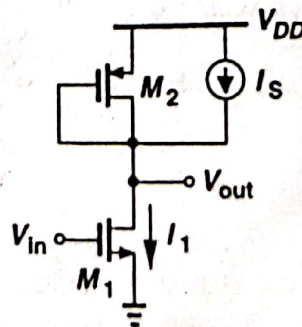


Figure 3.13

Moreover,

$$\mu_n \left(\frac{W}{L}\right)_1 (V_{GS1} - V_{TH1})^2 \approx 4\mu_p \left(\frac{W}{L}\right)_2 (V_{GS2} - V_{TH2})^2, \quad (3.40)$$

yielding

$$\frac{|V_{GS2} - V_{TH2}|}{V_{GS1} - V_{TH1}} \approx \frac{A_v}{4}. \quad (3.41)$$

Thus, for a gain of 10, the overdrive of M_2 need be only 2.5 times that of M_1 . Alternatively, for a given overdrive voltage, this circuit achieves a gain four times that of the stage in Fig. 3.12. Intuitively, this is because for a given $|V_{GS2} - V_{TH2}|$, if the current decreases by a factor of 4, then $(W/L)_2$ must decrease proportionally, and $g_{m2} = \sqrt{2\mu_p C_{ox}(W/L)_2 I_{D2}}$ is lowered by the same factor.

We should also mention that in today's CMOS technology, channel-length modulation is quite significant and, more importantly, the behavior of transistors notably departs from the square law (Chapter 16). Thus, the gain of the stage in Fig. 3.9 must be expressed as

$$A_v = -g_{m1} \left(\frac{1}{g_{m2}} \parallel r_{O1} \parallel r_{O2} \right), \quad (3.42)$$

where g_{m1} and g_{m2} must be obtained as described in Chapter 16.

3.2.3 CS Stage with Current-Source Load

In applications requiring a large voltage gain in a single stage, the relationship $A_v = -g_m R_D$ suggests that we increase the load impedance of the CS stage. With a resistor or diode-connected load, however, increasing the load resistance limits the output voltage swing.

A more practical approach is to replace the load with a current source. Described briefly in Example 3.2, the resulting circuit is shown in Fig. 3.14, where both transistors operate in saturation. Since the total impedance seen at the output node is equal to $r_{O1} \parallel r_{O2}$, the gain is

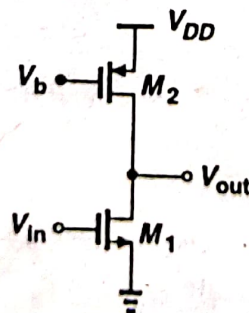


Figure 3.14 CS stage with current-source load.

$$A_v = -g_{m1}(r_{O1} \parallel r_{O2}). \quad (3.43)$$

The key point here is that the output impedance and the minimum required $|V_{DS}|$ of M_2 are less strongly coupled than the value and voltage drop of a resistor. The voltage

$|V_{DS2,min}| = |V_{GS2} - V_{TH2}|$ can be reduced to even a few hundred millivolts by simply increasing the width of M_2 . If r_{O2} is not sufficiently high, the length and width of M_2 can be increased to achieve a smaller λ while maintaining the same overdrive voltage. The penalty is the large capacitance introduced by M_2 at the output node.

We should remark that the output bias voltage of the circuit in Fig. 3.14 is not well-defined. Thus, the stage is reliably biased only if a feedback loop forces V_{out} to a known value (Chapter 8). The large-signal analysis of the circuit is left as an exercise for the reader.

As explained in Chapter 2, the output impedance of MOSFETs at a given drain current can be scaled by changing the channel length, i.e., to the first order, $\lambda \propto 1/L$ and hence $r_o \propto L/I_D$. Since the gain of the stage shown in Fig. 3.14 is proportional to $r_{O1} \parallel r_{O2}$, we may surmise that longer transistors yield a higher voltage gain.

Let us consider M_1 and M_2 separately. If L_1 is scaled by a factor $\alpha (> 1)$, then W_1 may need to be scaled proportionally as well. This is because, for a given drain current, $V_{GS1} - V_{TH1} \propto 1/\sqrt{(W/L)_1}$, i.e., if W_1 is not scaled, the overdrive voltage increases, limiting the output voltage swing. Also, since $g_{m1} \propto \sqrt{(W/L)_1}$, scaling up only L_1 lowers g_{m1} .

In applications where these issues are unimportant, W_1 can remain constant while L_1 increases. Thus, the intrinsic gain of the transistor can be written as

$$g_{m1}r_{O1} = \sqrt{2 \left(\frac{W}{L}\right)_1 \mu_n C_{ox} I_D \frac{1}{\lambda I_D}}, \quad (3.44)$$

indicating that the gain *increases* with L because λ depends more strongly on L than g_m does. Also, note that $g_m r_o$ *decreases* as I_D increases.

Increasing L_2 while keeping W_2 constant increases r_{O2} and hence the voltage gain, but at the cost of higher $|V_{DS2}|$ required to maintain M_2 in saturation.

3.2.4 CS Stage with Triode Load

A MOS device operating in deep triode region behaves as a resistor and can therefore serve as the load in a CS stage. Illustrated in Fig. 3.15, such a circuit biases the gate of M_2 at a sufficiently low level, ensuring the load is in deep triode region for all output voltage swings.

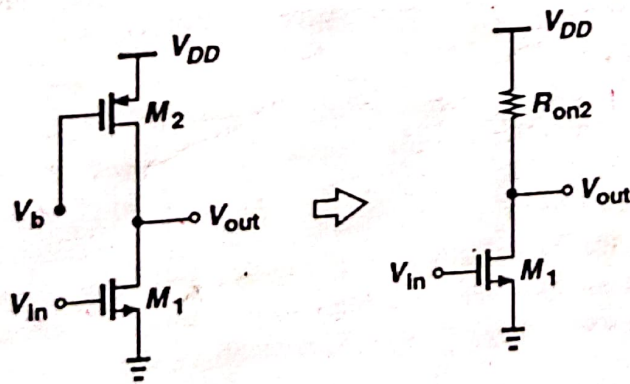


Figure 3.15 CS stage with triode load.

Since

$$R_{on2} = \frac{1}{\mu_p C_{ox} (W/L)_2 (V_{DD} - V_b - |V_{THP}|)} \quad (3.45)$$

the voltage gain can be readily calculated.

The principal drawback of this circuit stems from the dependence of R_{on2} upon $\mu_p C_{ox}$, V_b , and V_{THP} . Since $\mu_p C_{ox}$ and V_{THP} vary with process and temperature and since generating a precise value for V_b requires additional complexity, this circuit is difficult to use. Triode loads, however, consume less voltage headroom than do diode-connected devices because in Fig. 3.15 $V_{out,max} = V_{DD}$ whereas in Fig. 3.12, $V_{out,max} \approx V_{DD} - |V_{THP}|$.

3.2.5 CS Stage with Source Degeneration

In some applications, the square-law dependence of the drain current upon the overdrive voltage introduces excessive nonlinearity, making it desirable to “soften” the device characteristic. In Section 3.2.2, we noted the linear behavior of a CS stage using a diode-connected load. Alternatively, as depicted in Fig. 3.16, this can be accomplished by placing a “degeneration” resistor in series with the source terminal. Here, as V_{in} increases, so do I_D and the

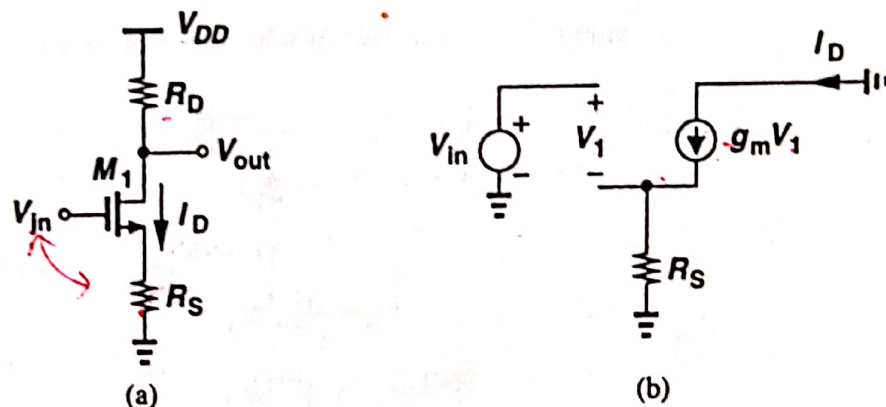


Figure 3.16 CS stage with source degeneration.

voltage drop across R_S . That is, a fraction of V_{in} appears across the resistor rather than as the gate-source overdrive, thus leading to a smoother variation of I_D . From another perspective, we intend to make the gain equation a weaker function of g_m . Since $V_{out} = -I_D R_D$, the nonlinearity of the circuit arises from the nonlinear dependence of I_D upon V_{in} . We note that $\partial V_{out} / \partial V_{in} = -(\partial I_D / \partial V_{in}) R_D$, and define the equivalent transconductance of the circuit as $G_m = \partial I_D / \partial V_{in}$. Now, assuming $I_D = f(V_{GS})$, we write

$$G_m = \frac{\partial I_D}{\partial V_{in}} \quad (3.46)$$

$$= \frac{\partial f}{\partial V_{GS}} \frac{\partial V_{GS}}{\partial V_{in}} \quad (3.47)$$

Since $V_{GS} = V_{in} - I_D R_S$, we have $\partial V_{GS} / \partial V_{in} = 1 - R_S \partial I_D / \partial V_{in}$, obtaining

$$G_m = \left(1 - R_S \frac{\partial I_D}{\partial V_{in}} \right) \frac{\partial f}{\partial V_{GS}}. \quad (3.48)$$

But, $\partial f / \partial V_{GS}$ is the transconductance of M_1 , and

$$G_m = \frac{g_m}{1 + g_m R_S}. \quad (3.49)$$

The small-signal voltage gain is thus equal to

$$A_v = -G_m R_D \quad (3.50)$$

$$= \frac{-g_m R_D}{1 + g_m R_S}. \quad (3.51)$$

The same result can be derived using the small-signal model of Fig. 3.16(b). Equation (3.49) implies that as R_S increases, G_m becomes a weaker function of g_m and hence the drain current. In fact, for $R_S \gg 1/g_m$, we have $G_m \approx 1/R_S$, i.e., $\Delta I_D \approx \Delta V_{in} / R_S$, indicating that most of the change in V_{in} appears across R_S . We say the drain current is a "linearized" function of the input voltage. The linearization is obtained at the cost of lower gain [and higher noise (Chapter 7)].

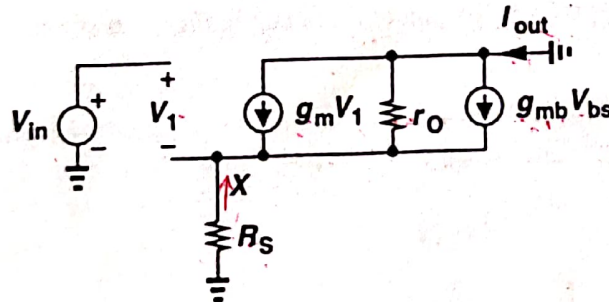


Figure 3.17 Small-signal equivalent circuit of a degenerated CS stage.

For our subsequent calculations, it is useful to determine G_m in the presence of body effect and channel-length modulation. With the aid of the equivalent circuit shown in Fig. 3.17, we recognize that the current through R_S equals I_{out} and, therefore, $V_{in} = V_1 + I_{out} R_S$. Summing the currents at node X , we have

$$I_{out} = g_m V_1 - g_{mb} V_X - \frac{I_{out} R_S}{r_o} \quad (3.52)$$

$$= g_m (V_{in} - I_{out} R_S) + g_{mb} (-I_{out} R_S) - \frac{I_{out} R_S}{r_o}. \quad (3.53)$$

It follows that

$$G_m = \frac{I_{out}}{V_{in}} \quad (3.54)$$

$$= \frac{g_m r_o}{R_S + [1 + (g_m + g_{mb}) R_S] r_o}. \quad (3.55)$$

3.3 Source Follower

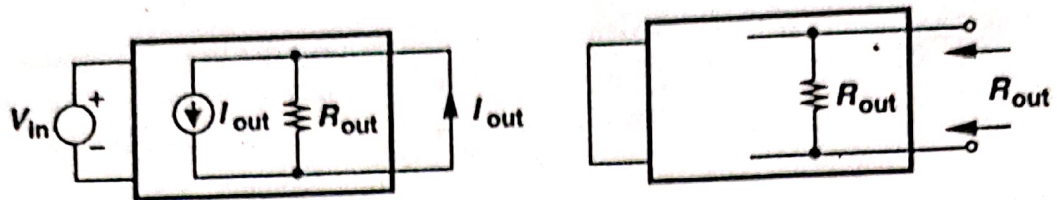


Figure 3.25 Modeling output port of an amplifier by a Norton equivalent.

Defining $G_m = I_{out}/V_{in}$, we have $V_{out} = -G_m V_{in} R_{out}$. This lemma proves useful if G_m and R_{out} can be determined by inspection.

Example 3.6

Calculate the voltage gain of the circuit shown in Fig. 3.26. Assume I_0 is ideal.

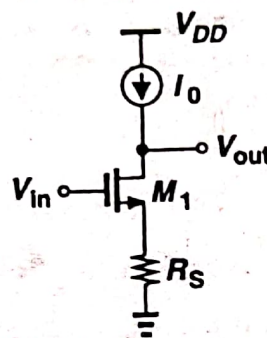


Figure 3.26

Solution

The transconductance and output resistance of the stage are given by Eqs. (3.55) and (3.60), respectively. Thus,

$$A_v = -\frac{g_m r_o}{R_S + [1 + (g_m + g_{mb})R_S]r_o} \{[1 + (g_m + g_{mb})r_o]R_S + r_o\} \quad (3.74)$$

$$= -g_m r_o. \quad (3.75)$$

Interestingly, the voltage gain is equal to the intrinsic gain of the transistor and independent of R_S . This is because, if I_0 is ideal, the current through R_S cannot change and hence the small-signal voltage drop across R_S is zero—as if R_S were zero itself.

3 Source Follower

Our analysis of the common-source stage indicates that, to achieve a high voltage gain with limited supply voltage, the load impedance must be as large as possible. If such a stage is to drive a low-impedance load, then a "buffer" must be placed after the amplifier so as to drive the load with negligible loss of the signal level. The source follower (also called the "common-drain" stage) can operate as a voltage buffer.

Illustrated in Fig. 3.27(a), the source follower senses the signal at the gate and drives

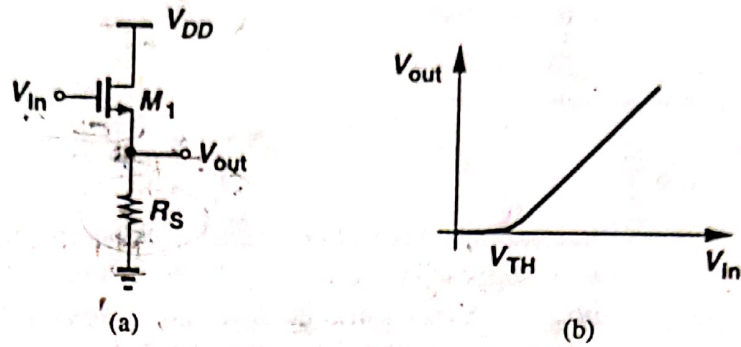


Figure 3.27 (a) Source follower, and (b) its input-output characteristic.

the load at the source, allowing the source potential to “follow” the gate voltage. Beginning with the large-signal behavior, we note that for $V_{in} < V_{TH}$, M_1 is off and $V_{out} = 0$. As V_{in} exceeds V_{TH} , M_1 turns on in saturation (for typical values of V_{DD}) and I_{D1} flows through R_S [Fig. 3.27(b)]. As V_{in} increases further, V_{out} follows the input with a difference (level shift) equal to V_{GS} . We can express the input-output characteristic as:

$$\frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_{in} - V_{TH} - V_{out})^2 R_S = V_{out} \quad (3.76)$$

Let us calculate the small-signal gain of the circuit by differentiating both sides of (3.76) with respect to V_{in} :

$$\frac{1}{2} \mu_n C_{ox} \frac{W}{L} 2(V_{in} - V_{TH} - V_{out}) \left(1 - \frac{\partial V_{TH}}{\partial V_{in}} - \frac{\partial V_{out}}{\partial V_{in}} \right) R_S = \frac{\partial V_{out}}{\partial V_{in}} \quad (3.77)$$

Since $\partial V_{TH} / \partial V_{in} = \eta \partial V_{out} / \partial V_{in}$,

$$\frac{\partial V_{out}}{\partial V_{in}} = \frac{\mu_n C_{ox} \frac{W}{L} (V_{in} - V_{TH} - V_{out}) R_S}{1 + \mu_n C_{ox} \frac{W}{L} (V_{in} - V_{TH} - V_{out}) R_S (1 + \eta)} \quad (3.78)$$

Also, note that

$$g_m = \mu_n C_{ox} \frac{W}{L} (V_{in} - V_{TH} - V_{out}) \quad (3.79)$$

Consequently,

$$A_v = \frac{g_m R_S}{1 + (g_m + g_{mb}) R_S} \quad (3.80)$$

The same result is more easily obtained with the aid of a small-signal equivalent circuit. From Fig. 3.28, we have $V_{in} - V_1 = V_{out}$, $V_{bs} = -V_{out}$, and $g_m V_1 - g_{mb} V_{out} = V_{out} / R_S$.

3.3 Source Follower

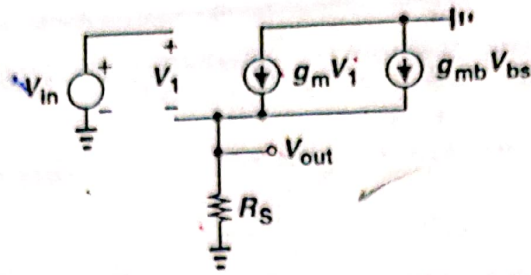


Figure 3.28 Small-signal equivalent circuit of source follower.

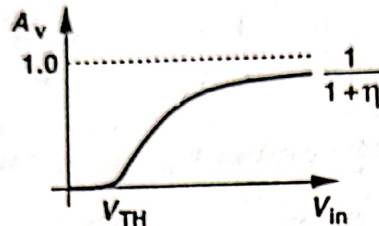


Figure 3.29 Voltage gain of source follower versus input voltage.

Thus, $V_{out}/V_{in} = g_m R_S / [1 + (g_m + g_{mb}) R_S]$.

Sketched in Fig. 3.29 vs. V_{in} , the voltage gain begins from zero for $V_{in} \approx V_{TH}$ (that is, $g_m \approx 0$) and monotonically increases. As the drain current and g_m increase, A_v approaches $g_m / (g_m + g_{mb}) = 1 / (1 + \eta)$. Since η itself slowly decreases with V_{out} , A_v would eventually become equal to unity, but for typical allowable source-bulk voltages, η remains greater than roughly 0.2.

An important result of (3.80) is that even if $R_S = \infty$, the voltage gain of a source follower is not equal to one. We return to this point later. Note that M_1 in Fig. 3.27 does not enter the triode region if V_{in} remains below V_{DD} .

In the source follower of Fig. 3.27, the drain current of M_1 heavily depends on the input dc level. For example, if V_{in} changes from 1.5 V to 2 V, I_D may increase by a factor of 2 and hence $V_{GS} - V_{TH}$ by $\sqrt{2}$, thereby introducing substantial nonlinearity in the input-output characteristic. To alleviate this issue, the resistor can be replaced by a current source as shown in Fig. 3.30(a). The current source itself is implemented as an NMOS transistor operating in the saturation region [Fig. 3.30(b)].

Handwritten notes:
 $V_{out} = V_{DD} - I_D R_S$
 $\frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_{in} - V_{TH} - V_{out})^2 R_S = V_{out}$

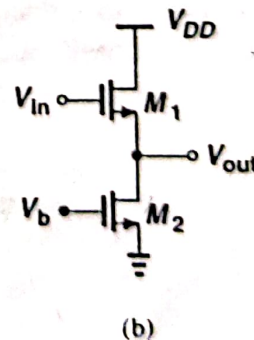
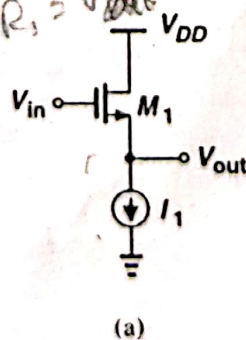


Figure 3.30 Source follower using an NMOS transistor as current source.

Example 3.7

Suppose in the source follower of Fig. 3.30(a), $(W/L)_1 = 20/0.5$, $I_1 = 200 \mu\text{A}$, $V_{TH0} = 0.6 \text{ V}$, $2\Phi_F = 0.7 \text{ V}$, $\mu_n C_{ox} = 50 \mu\text{A/V}^2$, and $\gamma = 0.4 \text{ V}^2$.

(a) Calculate V_{out} for $V_{in} = 1.2 \text{ V}$.

(b) If I_1 is implemented as M_2 in Fig. 3.30(b), find the minimum value of $(W/L)_2$ for which M_2 remains saturated.

Solution

(a) Since the threshold voltage of M_1 depends on V_{out} , we perform a simple iteration. Noting that

$$(V_{in} - V_{TH} - V_{out})^2 = \frac{2I_D}{\mu_n C_{ox} \left(\frac{W}{L}\right)_1} \quad (3.81)$$

we first assume $V_{TH} \approx 0.6 \text{ V}$, obtaining $V_{out} = 0.153 \text{ V}$. Now we calculate a new V_{TH} as

$$V_{TH} = V_{TH0} + \gamma(\sqrt{2\Phi_F + V_{SB}} - \sqrt{2\Phi_F}) \quad (3.82)$$

$$= 0.635 \text{ V}. \quad (3.83)$$

This indicates that V_{out} is approximately 35 mV less than that calculated above, i.e., $V_{out} \approx 0.119 \text{ V}$.

(b) Since the drain-source voltage of M_2 is equal to 0.119 V, the device is saturated only if $(V_{GS} - V_{TH})_2 \leq 0.119 \text{ V}$. With $I_D = 200 \mu\text{A}$, this gives $(W/L)_2 \geq 283/0.5$. Note the substantial drain junction and overlap capacitance contributed by M_2 to the output node.

To gain a better understanding of source followers, let us calculate the small-signal output resistance of the circuit in Fig. 3.31(a). Using the equivalent circuit of Fig. 3.31(b) and noting that $V_1 = -V_X$, we write

$$I_X - g_m V_X - g_{mb} V_X = 0. \quad (3.84)$$

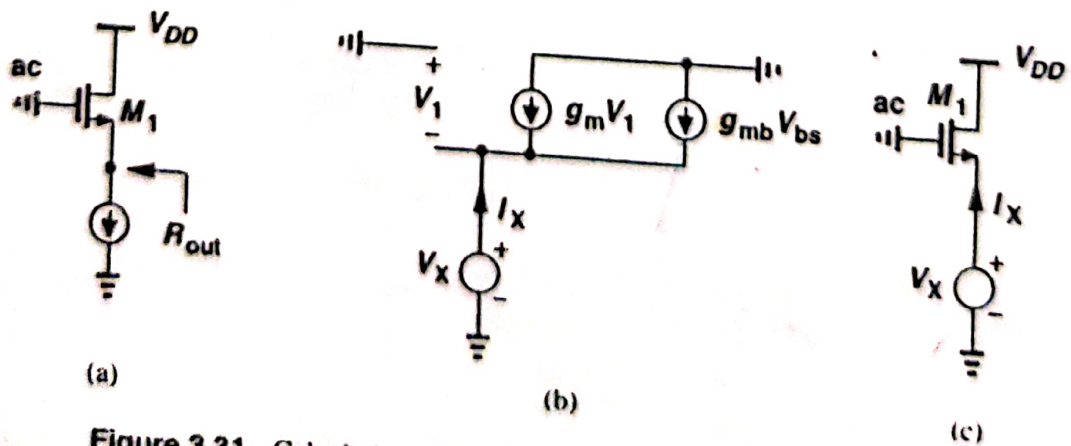


Figure 3.31 Calculation of the output impedance of a source follower.

As explained in Chapter 7, source followers also introduce substantial noise. For this reason, the circuit of Fig. 3.39(b) is ill-suited to low-noise applications.

3.4 Common-Gate Stage

In common-source amplifiers and source followers, the input signal is applied to the gate of a MOSFET. It is also possible to apply the signal to the source terminal. Shown in Fig. 3.40(a), a common-gate (CG) stage senses the input at the source and produces the output at the drain. The gate is connected to a dc voltage to establish proper operating conditions. Note that the bias current of M_1 flows through the input signal source. Alternatively, as depicted in Fig. 3.40(b), M_1 can be biased by a constant current source, with the signal capacitively coupled to the circuit.

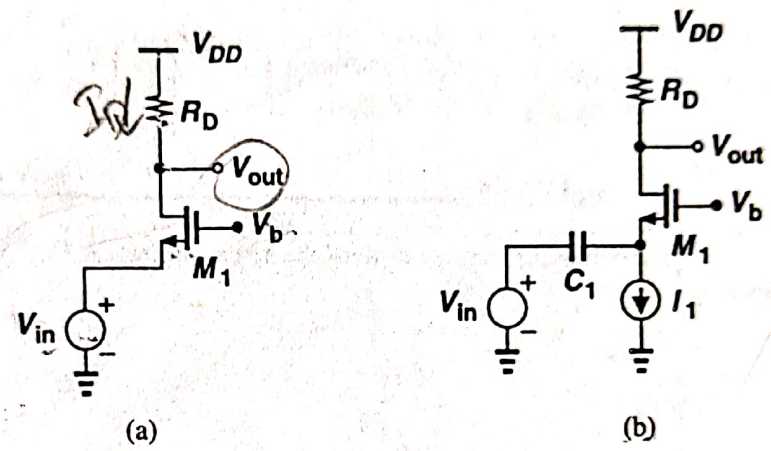


Figure 3.40 (a) Common-gate stage with direct coupling at input, (b) CG stage with capacitive coupling at input.

We first study the large-signal behavior of the circuit in Fig. 3.40(a). For simplicity, let us assume that V_{in} decreases from a large positive value. For $V_{in} \geq V_b - V_{TH}$, M_1 is off and $V_{out} = V_{DD}$. For lower values of V_{in} , we can write

$$I_D = \frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_b - V_{in} - V_{TH})^2, \tag{3.95}$$

if M_1 is in saturation. As V_{in} decreases, so does V_{out} , eventually driving M_1 into the triode region if

$$V_{DD} - I_D R_D = V_b - V_{TH}$$

$$V_{DD} - \frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_b - V_{in} - V_{TH})^2 R_D = V_b - V_{TH}. \tag{3.96}$$

The input-output characteristic is shown in Fig. 3.41. If M_1 is saturated, we can express the output voltage as

$$V_{out} = V_{DD} - I_D R_D = V_{DD} - \frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_b - V_{in} - V_{TH})^2 R_D. \tag{3.97}$$

$$\frac{dV_{out}}{dV_{in}} = 0 - \frac{1}{2} \mu_n C_{ox} \frac{W}{L} \cdot 2 (V_b - V_{in} - V_{TH}) R_D \left(-1 - \frac{\partial V_{TH}}{\partial V_{in}} \right)$$

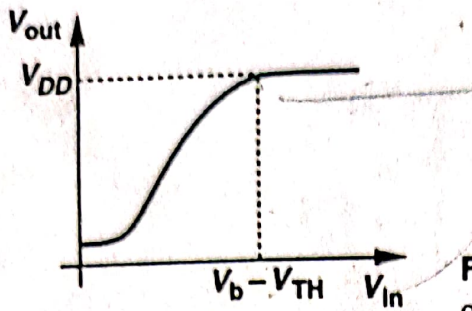


Figure 3.41 Common-gate input-output characteristic.

obtaining a small-signal gain of

$$\frac{\partial V_{out}}{\partial V_{in}} = \mu_n C_{ox} \frac{W}{L} (V_b - V_{in} - V_{TH}) \left(-1 - \frac{\partial V_{TH}}{\partial V_{in}} \right) R_D \quad (3.98)$$

Since $\partial V_{TH}/\partial V_{in} = \partial V_{TH}/\partial V_{SB} = \eta$, we have

$$\frac{\partial V_{out}}{\partial V_{in}} = \mu_n C_{ox} \frac{W}{L} R_D (V_b - V_{in} - V_{TH}) (1 + \eta) \quad (3.99)$$

$$A_v = g_m (1 + \eta) R_D \quad g_m = \mu_n C_{ox} \frac{W}{L} (V_b - V_{in} - V_{TH}) \quad (3.100)$$

Note that the gain is positive. Interestingly, body effect increases the equivalent transconductance of the stage.

The input impedance of the circuit is also important. We note that, for $\lambda = 0$, the impedance seen at the source of M_1 in Fig. 3.40(a) is the same as that at the source of M_1 in Fig. 3.31, namely, $1/(g_m + g_{mb}) = 1/[g_m(1 + \eta)]$. Thus, the body effect decreases the input impedance of the common-gate stage. The relatively low input impedance of the common-gate stage proves useful in some applications.

Example 3.10

In Fig. 3.42, transistor M_1 senses ΔV and delivers a proportional current to a 50- Ω transmission line. The other end of the line is terminated by a 50- Ω resistor in Fig. 3.42(a) and a common-gate stage in Fig. 3.42(b). Assume $\lambda = \gamma = 0$.

- (a) Calculate V_{out}/V_{in} at low frequencies for both arrangements.
- (b) What condition is necessary to minimize wave reflection at node X?

Solution

(a) For small signals applied to the gate of M_1 , the drain current experiences a change equal to $g_{m1} \Delta V_X$. This current is drawn from R_D in Fig. 3.42(a) and M_2 in Fig. 3.42(b), producing an output voltage swing equal to $-g_{m1} \Delta V_X R_D$. Thus, $A_v = -g_m R_D$ for both cases.

(b) To minimize reflection at node X, the resistance seen at the source of M_2 must equal 50 Ω and the reactance must be small. Thus, $1/(g_m + g_{mb}) = 50 \Omega$, which can be ensured by proper sizing and biasing of M_2 . To minimize the capacitances of the transistor, it is desirable to use a small device biased at a large current. (Recall that $g_m = \sqrt{2\mu_n C_{ox}(W/L)I_D}$.) In addition to higher power dissipation, this remedy also requires a large V_{GS} for M_2 .

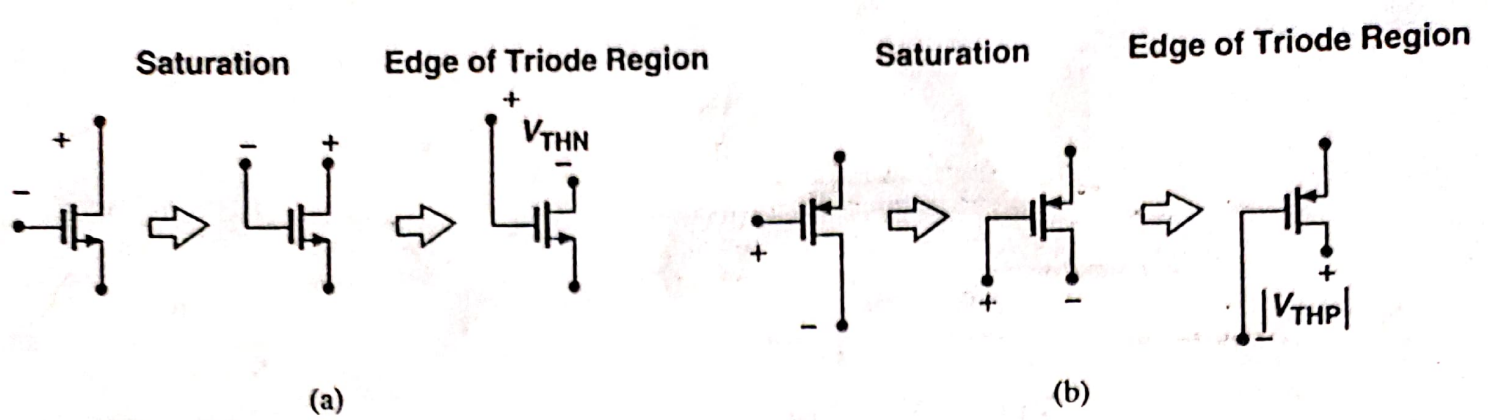


Figure 2.20 Conceptual visualization of saturation and triode regions.

if $V_D - V_G$ of a PFET is not large enough ($< |V_{THP}|$), the device is saturated. Note that this view does not require knowledge of the source voltage. This means we must know a priori which terminal operates as the drain.

Second-Order Effects

Our analysis of the MOS structure has thus far entailed various simplifying assumptions, some of which are not valid in many analog circuits. In this section, we describe three second-order effects that are essential in our subsequent circuit analyses. Other phenomena that appear in submicron devices are studied in Chapter 16.

Body Effect In the analysis of Fig. 2.10, we tacitly assumed that the bulk and the source of the transistor were tied to ground. What happens if the bulk voltage of an NFET drops below the source voltage (Fig. 2.21)? Since the S and D junctions remain reverse-biased, we surmise that the device continues to operate properly but certain characteristics may

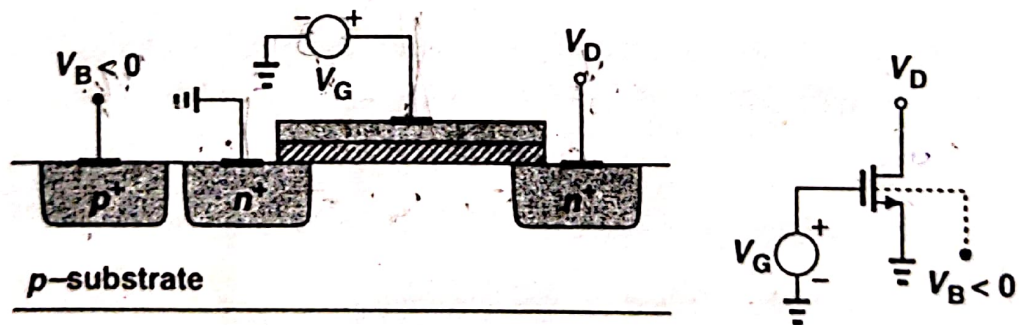


Figure 2.21 NMOS device with negative bulk voltage.

change. To understand the effect, suppose $V_S = V_D = 0$, and V_G is somewhat less than V_{TH} so that a depletion region is formed under the gate but no inversion layer exists. As V_B becomes more negative, more holes are attracted to the substrate connection, leaving a larger negative charge behind, i.e., as depicted in Fig. 2.22, the depletion region becomes wider. Now recall from Eq. (2.1) that the threshold voltage is a function of the total charge in the depletion region because the gate charge must mirror Q_d before an inversion layer is

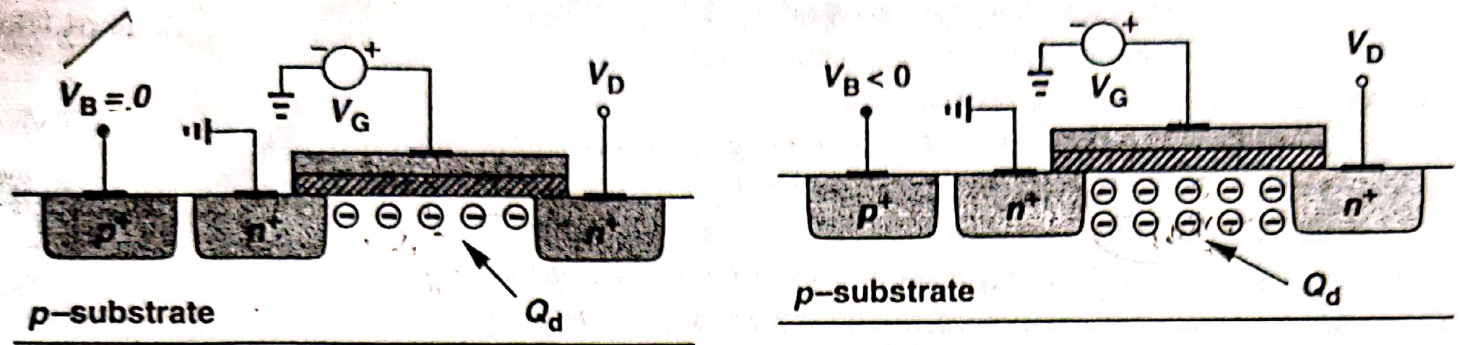


Figure 2.22 Variation of depletion region charge with bulk voltage.

formed. Thus, as V_B drops and Q_d increases, V_{TH} also increases. This is called the “body effect” or the “backgate effect.”

It can be proved that with body effect:

$$V_{TH} = V_{TH0} + \gamma \left(\sqrt{|2\Phi_F + V_{SB}|} - \sqrt{|2\Phi_F|} \right), \quad (2.22)$$

where V_{TH0} is given by (2.1), $\gamma = \sqrt{2q\epsilon_{si}N_{sub}}/C_{ox}$ denotes the body effect coefficient, and V_{SB} is the source-bulk potential difference [1]. The value of γ typically lies in the range of 0.3 to 0.4 $V^{1/2}$.

Example 2.3

In Fig. 2.23(a), plot the drain current if V_X varies from $-\infty$ to 0. Assume $V_{TH0} = 0.6$ V, $\gamma = 0.4$ $V^{1/2}$, and $2\Phi_F = 0.7$ V.

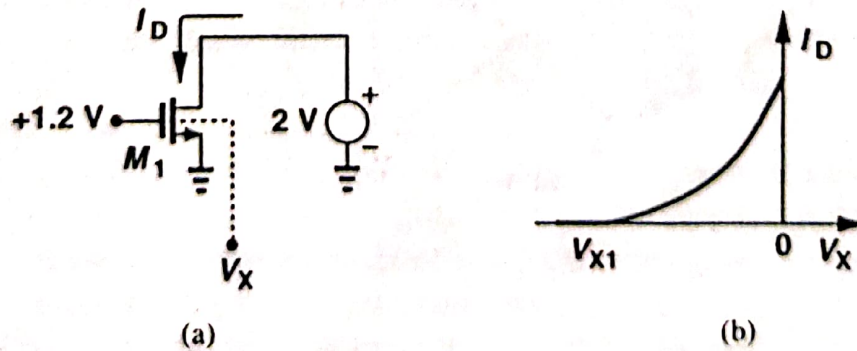


Figure 2.23

Solution

If V_X is sufficiently negative, the threshold voltage of M_1 exceeds 1.2 V and the device is off. That is,

$$1.2 \text{ V} = 0.6 + 0.4 \left(\sqrt{0.7 - V_{X1}} - \sqrt{0.7} \right), \quad (2.23)$$

6.1 Introduction

The design considerations for a simple inverter circuit were presented in the previous chapter. We now extend this discussion to address the synthesis of arbitrary digital gates, such as NOR, NAND, and XOR. The focus is on *combinational logic* or *nonregenerative* circuits—that is, circuits having the property that at any point in time, the output of the circuit is related to its current input signals by some Boolean expression (assuming that the transients through the logic gates have settled). No intentional connection from outputs back to inputs is present.

This is in contrast to another class of circuits, known as *sequential* or *regenerative*, for which the output is not only a function of the current input data, but also of previous values of the input signals (see Figure 6-1). This can be accomplished by connecting one or more outputs intentionally back to some inputs. Consequently, the circuit “remembers” past events and has a sense of *history*. A sequential circuit includes a combinational logic portion and a module that holds the state. Example circuits are registers, counters, oscillators, and memory. Sequential circuits are the topic of the next chapter.

There are numerous circuit styles to implement a given logic function. As with the inverter, the common design metrics by which a gate is evaluated are area, speed, energy, and power. Depending on the application, the emphasis will be on different metrics. For example, the switching speed of digital circuits is the primary metric in a high-performance processor, while in a battery operated circuit, it is energy dissipation. Recently, power dissipation also has become an important concern and considerable emphasis is placed on understanding the sources of power and approaches to dealing with power. In addition to these metrics, robustness to noise and reliability are also very important considerations. We will see that certain logic styles can significantly improve performance, but they usually are more sensitive to noise.

6.2 Static CMOS Design

The most widely used logic style is static complementary CMOS. The static CMOS style is really an extension of the static CMOS inverter to multiple inputs. To review, the primary advantage of the CMOS structure is robustness (i.e., low sensitivity to noise), good performance, and low power consumption with no static power dissipation. Most of those properties are carried over to large fan-in logic gates implemented using a similar circuit topology.

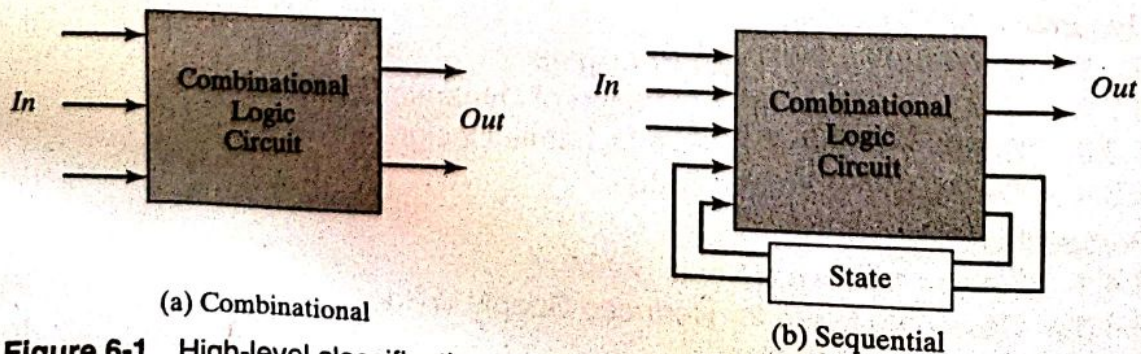


Figure 6-1 High-level classification of logic circuits.

The complementary CMOS circuit style falls under a broad class of logic circuits called *static* circuits in which at every point in time, each gate output is connected to either V_{DD} or V_{SS} via a low-resistance path. Also, the outputs of the gates assume at all times the value of the Boolean function implemented by the circuit (ignoring, the transient effects during switching periods). This is in contrast to the *dynamic* circuit class, which relies on temporary storage of signal values on the capacitance of high-impedance circuit nodes. The latter approach has the advantage that the resulting gate is simpler and faster. Its design and operation are, however, more involved and prone to failure because of increased sensitivity to noise.

In this section, we sequentially address the design of various static circuit flavors, including complementary CMOS, ratioed logic (pseudo-NMOS and DCVSL), and pass-transistor logic. We also deal with issues of scaling to lower power supply voltages and threshold voltages.

6.2.1 Complementary CMOS

Concept

A static CMOS gate is a combination of two networks—the *pull-up network* (PUN) and the *pull-down network* (PDN), as shown in Figure 6-2. The figure shows a generic N -input logic gate where all inputs are distributed to both the pull-up and pull-down networks. The function of the PUN is to provide a connection between the output and V_{DD} anytime the output of the logic gate is meant to be 1 (based on the inputs). Similarly, the function of the PDN is to connect the output to V_{SS} when the output of the logic gate is meant to be 0. The PUN and PDN networks are constructed in a mutually exclusive fashion such that *one and only one* of the networks is conducting in steady state. In this way, once the transients have settled, a path always exists between V_{DD} and the output F for a high output (“one”), or between V_{SS} and F for a low output (“zero”). This is equivalent to stating that the output node is always a *low-impedance* node in steady state.

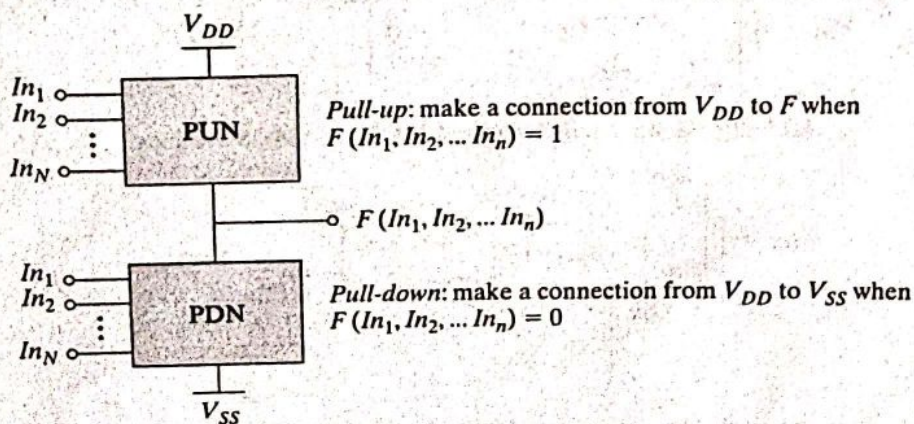


Figure 6-2 Complementary logic gate as a combination of a PUN (pull-up network) and a PDN (pull-down network).

In constructing the PDN and PUN networks, the designer should keep the following observations in mind:

- A transistor can be thought of as a switch controlled by its gate signal. An NMOS switch is *on* when the controlling signal is high and is *off* when the controlling signal is low. A PMOS transistor acts as an inverse switch that is *on* when the controlling signal is low and *off* when the controlling signal is high.
- The PDN is constructed using NMOS devices, while PMOS transistors are used in the PUN. The primary reason for this choice is that NMOS transistors produce “strong zeros,” and PMOS devices generate “strong ones.” To illustrate this, consider the examples shown in Figure 6-3. In Figure 6-3a, the output capacitance is initially charged to V_{DD} . Two possible discharge scenarios are shown. An NMOS device pulls the output all the way down to GND, while a PMOS lowers the output no further than $|V_{Tp}|$ —the PMOS turns *off* at that point and stops contributing discharge current. NMOS transistors are thus the preferred devices in the PDN. Similarly, two alternative approaches to charging up a capacitor are shown in Figure 6-3b, with the output initially at GND. A PMOS switch succeeds in charging the output all the way to V_{DD} , while the NMOS device fails to raise the output above $V_{DD} - V_{Tn}$. This explains why PMOS transistors are preferentially used in a PUN.
- A set of rules can be derived to construct logic functions (see Figure 6-4). NMOS devices connected in series correspond to an AND function. With all the inputs high, the series combination conducts and the value at one end of the chain is transferred to the other end. Similarly, NMOS transistors connected in parallel represent an OR function. A conducting path exists between the output and input terminal if at least one of the inputs is high. Using similar arguments, construction rules for PMOS networks can be formulated. A series con-

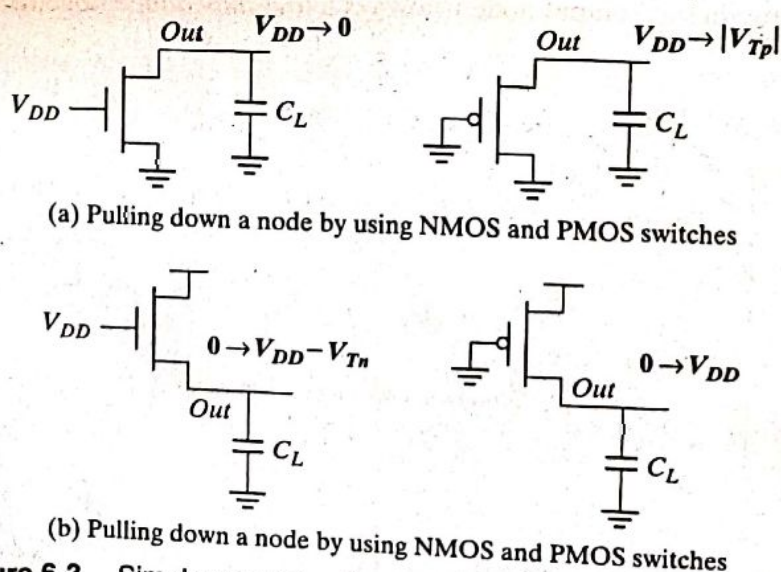


Figure 6-3 Simple examples illustrate why an NMOS should be used as a pull-down, and a PMOS should be used as a pull-up device.

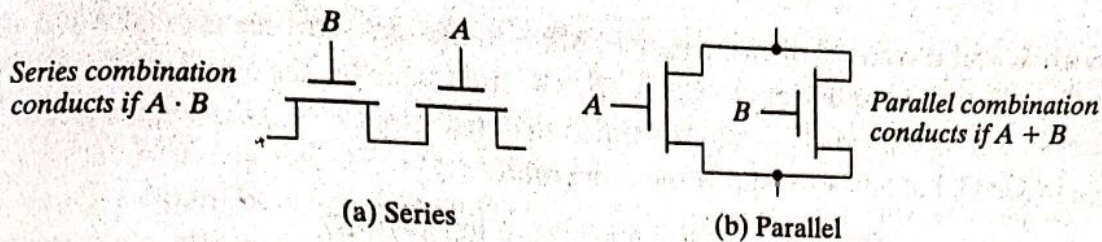


Figure 6-4 NMOS logic rules—series devices implement an AND, and parallel devices implement an OR.

nection of PMOS conducts if both inputs are low, representing a NOR function ($\overline{A} \cdot \overline{B} = \overline{A + B}$), while PMOS transistors in parallel implement a NAND ($\overline{A} + \overline{B} = \overline{A \cdot B}$).

- Using De Morgan's theorems ($\overline{A + B} = \overline{A} \cdot \overline{B}$ and $\overline{A \cdot B} = \overline{A} + \overline{B}$), it can be shown that the pull-up and pull-down networks of a complementary CMOS structure are *dual* networks. This means that a parallel connection of transistors in the pull-up network corresponds to a series connection of the corresponding devices in the pull-down network, and vice versa. Therefore, to construct a CMOS gate, one of the networks (e.g., PDN) is implemented using combinations of series and parallel devices. The other network (i.e., PUN) is obtained using the duality principle by walking the hierarchy, replacing series subnets with parallel subnets, and parallel subnets with series subnets. The complete CMOS gate is constructed by combining the PDN with the PUN.
- The complementary gate is naturally *inverting*, implementing only functions such as NAND, NOR, and XNOR. The realization of a noninverting Boolean function (such as AND OR, or XOR) in a single stage is not possible, and requires the addition of an extra inverter stage.
- The number of transistors required to implement an N -input logic gate is $2N$.

Example 6.1 Two-Input NAND Gate

Figure 6-5 shows a two-input NAND gate ($F = \overline{A \cdot B}$). The PDN network consists of two NMOS devices in series that conduct when both A and B are high. The PUN is the dual

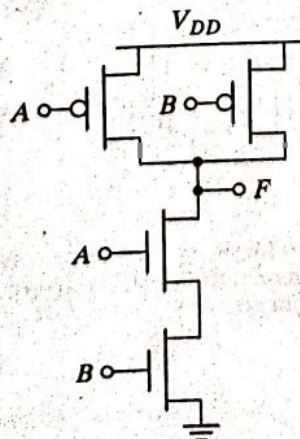


Figure 6-5 Two-input NAND gate in complementary static CMOS style.

network, and it consists of two parallel PMOS transistors. This means that F is 1 if $A = 0$ or $B = 0$, which is equivalent to $F = \overline{A \cdot B}$. The truth table for the simple two input NAND gate is given in Table 6-1. It can be verified that the output F is always connected to either V_{DD} or GND, but never to both at the same time.

Table 6-1 Truth Table for two-Input NAND.

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

Example 6.2 Synthesis of Complex CMOS Gate

Using complementary CMOS logic, consider the synthesis of a complex CMOS gate whose function is $F = \overline{D + A \cdot (B + C)}$. The first step in the synthesis of the logic gate is to derive the pull-down network as shown in Figure 6-6a by using the fact that NMOS devices in series implements the AND function and parallel device implements the OR function. The next step is to use duality to derive the PUN in a hierarchical fashion. The PDN network is broken into smaller networks (i.e., subset of the PDN) called subnets that simplify the derivation of the PUN. In Figure 6-6b, the subnets (SN) for the pull-down net-

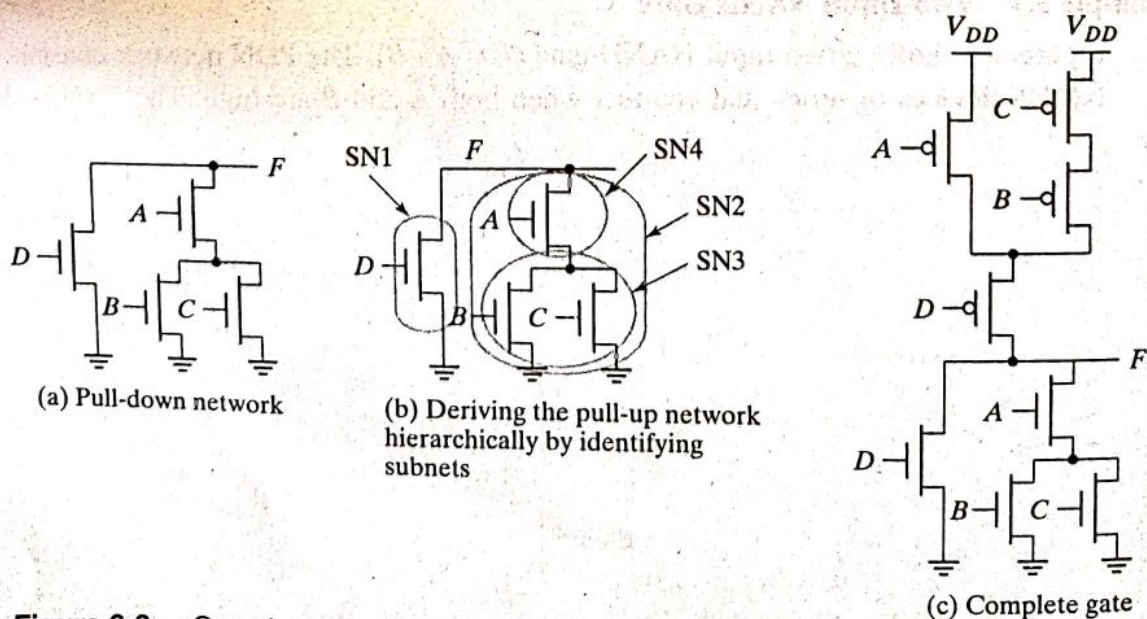


Figure 6-6 Complex complementary CMOS gate.

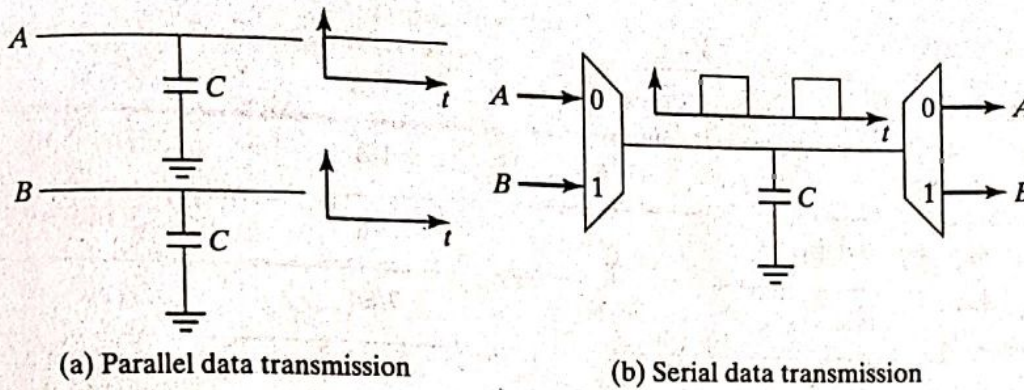


Figure 6-25 Parallel versus time-multiplexed data busses.

the bus toggles between 0 and 1. Care must be taken in digital systems to avoid time-multiplexing data streams with very distinct data characteristics.

4. **Glitch Reduction by balancing signal paths** The occurrence of glitching in a circuit is mainly due to a mismatch in the path lengths in the network. If all input signals of a gate change simultaneously, no glitching occurs. On the other hand, if input signals change at different times, a dynamic hazard might develop. Such a mismatch in signal timing is typically the result of different path lengths with respect to the primary inputs of the network. This is illustrated in Figure 6-26. Assume that the XOR gate has a unit delay. The first network (a) suffers from glitching as a result of the wide disparity between the arrival times of the input signals for a gate. For example, for gate F_3 , one input settles at time 0, while the second one only arrives at time 2. Redesigning the network so that all arrival times are identical can dramatically reduce the number of superfluous transitions (network b).

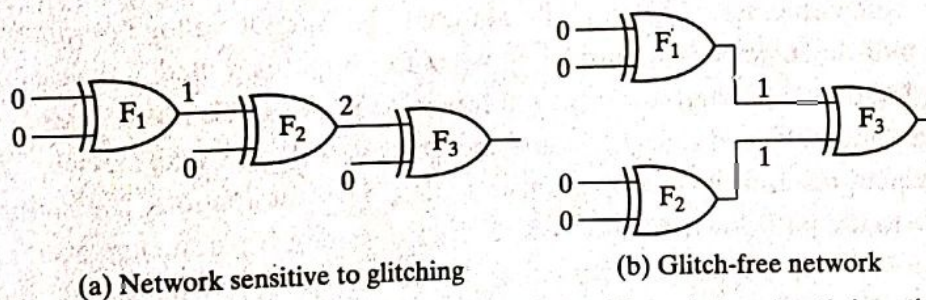


Figure 6-26 Glitching is influenced by matching of signal path lengths. The annotated numbers indicate the signal arrival times.

Summary

The CMOS logic style described in the previous section is highly robust and scalable with technology, but requires $2N$ transistors to implement an N -input logic gate. Also, the load capacitance is significant, since each gate drives two devices (a PMOS and an NMOS) per fan-out. This has opened the door for alternative logic families that either are simpler or faster.

6.2.2 Ratioed Logic

Concept

Ratioed logic is an attempt to reduce the number of transistors required to implement a given logic function, often at the cost of reduced robustness and extra power dissipation. The purpose

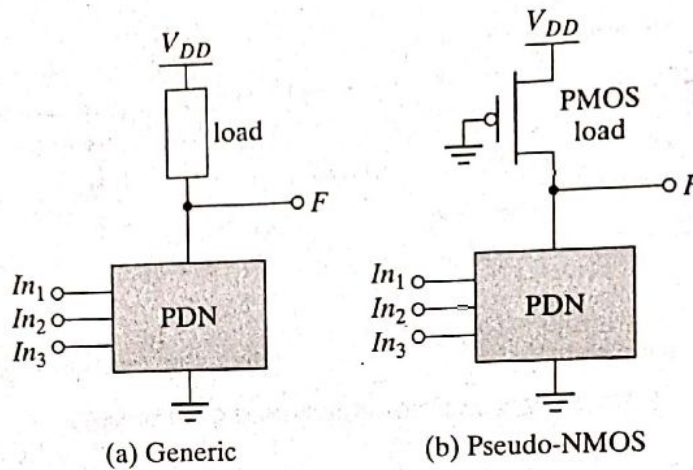


Figure 6-27 Ratioed logic gate.

of the PUN in complementary CMOS is to provide a conditional path between V_{DD} and the output when the PDN is turned *off*. In ratioed logic, the entire PUN is replaced with a single unconditional load device that pulls up the output for a high output as in Figure 6-27a. Instead of a combination of active pull-down and pull-up networks, such a gate consists of an NMOS pull-down network that realizes the *logic function*, and a simple *load device*. Figure 6-27b shows an example of ratioed logic, which uses a grounded PMOS load and is referred to as a pseudo-NMOS gate.

The clear advantage of a pseudo-NMOS gate is the reduced number of transistors ($N + 1$, versus $2N$ for complementary CMOS). The nominal high output voltage (V_{OH}) for this gate is V_{DD} since the pull-down devices are turned *off* when the output is pulled high (assuming that V_{OL} is below V_{Tn}). On the other hand, the **nominal low output voltage is not 0 V**, since there is contention between the devices in the PDN and the grounded PMOS load device. This results in reduced noise margins and, more importantly, static power dissipation. The sizing of the load device relative to the pull-down devices can be used to trade off parameters such as *noise margin*, *propagation delay*, and *power dissipation*. Since the voltage swing on the output and the overall functionality of the gate depend on the ratio of the NMOS and PMOS sizes, the circuit is called *ratioed*. This is in contrast to the *ratioless* logic styles, such as complementary CMOS, where the low and high levels do not depend on transistor sizes.

Computing the dc-transfer characteristic of the pseudo-NMOS proceeds along paths similar to those used for its complementary CMOS counterpart. The value of V_{OL} is obtained by equating the currents through the driver and load devices for $V_{in} = V_{DD}$. At this operation point, it is reasonable to assume that the NMOS device resides in linear mode (since, ideally, the output should be close to 0V), while the PMOS load is saturated:

$$k_n \left((V_{DD} - V_{Tn}) V_{OL} - \frac{V_{OL}^2}{2} \right) + k_p \left((-V_{DD} - V_{Tp}) \cdot V_{DSATp} - \frac{V_{DSATp}^2}{2} \right) = 0 \quad (6.27)$$

Assuming that V_{OL} is small relative to the gate drive ($V_{DD} - V_T$), and that V_{Tn} is equal to V_{Tp} in magnitude, V_{OL} can be approximated as

$$V_{OL} \approx \frac{k_p(V_{DD} + V_{Tp}) \cdot V_{DSATp}}{k_n(V_{DD} - V_{Tn})} \approx \frac{\mu_p \cdot W_p}{\mu_n \cdot W_n} \cdot V_{DSATp} \quad (6.28)$$

In order to make V_{OL} as small as possible, the PMOS device should be sized much smaller than the NMOS pull-down devices. Unfortunately, this has a negative impact on the *propagation delay* for charging up the output node since the current provided by the PMOS device is limited.

A major disadvantage of the pseudo-NMOS gate is the static power that is dissipated when the output is low through the direct current path that exists between V_{DD} and GND. The static power consumption in the low-output mode is easily derived:

$$P_{low} = V_{DD} I_{low} \approx V_{DD} \cdot \left| k_p \left((-V_{DD} - V_{Tp}) \cdot V_{DSATp} - \frac{V_{DSATp}^2}{2} \right) \right| \quad (6.29)$$

Example 6.7 Pseudo-NMOS Inverter

Consider a simple pseudo-NMOS inverter (where the PDN network in Figure 6-27 degenerates to a single transistor) with an NMOS size of $0.5 \mu\text{m}/0.25 \mu\text{m}$. In this example, we study the effect of sizing the PMOS device to demonstrate the impact on various parameters. The $W-L$ ratio of the grounded PMOS is varied over values from 4, 2, 1, 0.5 to 0.25. Devices with a $W-L < 1$ are constructed by making the length greater than the width. The voltage transfer curve for the different sizes is plotted in Figure 6-28.

Table 6-9 summarizes the nominal output voltage (V_{OL}), static power dissipation, and the low-to-high propagation delay. The low-to-high delay is measured as the time it takes to reach 1.25 V from V_{OL} (which is not 0V for this inverter)—by definition. The trade-off between the static and dynamic properties is apparent. A larger pull-up device not only improves performance, but also increases static power dissipation and lowers noise margins by increasing V_{OL} .

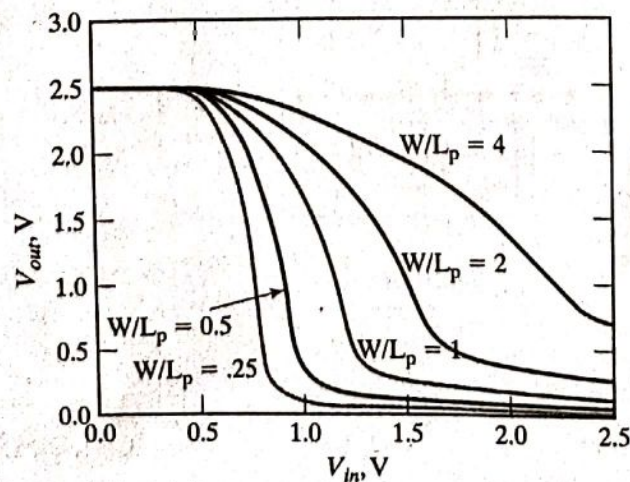


Figure 6-28 Voltage-transfer curves of the pseudo-NMOS inverter as a function of the PMOS size.

Table 6-9 Performance of a pseudo-NMOS inverter.

Size	V_{OL}	Static Power Dissipation	t_{plh}
4	0.693 V	564 μ W	14 ps
2	0.273 V	298 μ W	56 ps
1	0.133 V	160 μ W	123 ps
0.5	0.064 V	80 μ W	268 ps
0.25	0.031 V	41 μ W	569 ps

Notice that the simple first-order model to predict V_{OL} is quite effective. For a PMOS $W-L$ of 4, V_{OL} is given by $(30/115) (4) (0.63V) = 0.66V$.

The static power dissipation of pseudo-NMOS limits its use. When area is most important however, its reduced transistor count compared with complementary CMOS is quite attractive. Pseudo-NMOS thus still finds occasional use in large fan-in circuits. Figure 6-29 shows the schematics of pseudo-NMOS NOR and NAND gates.

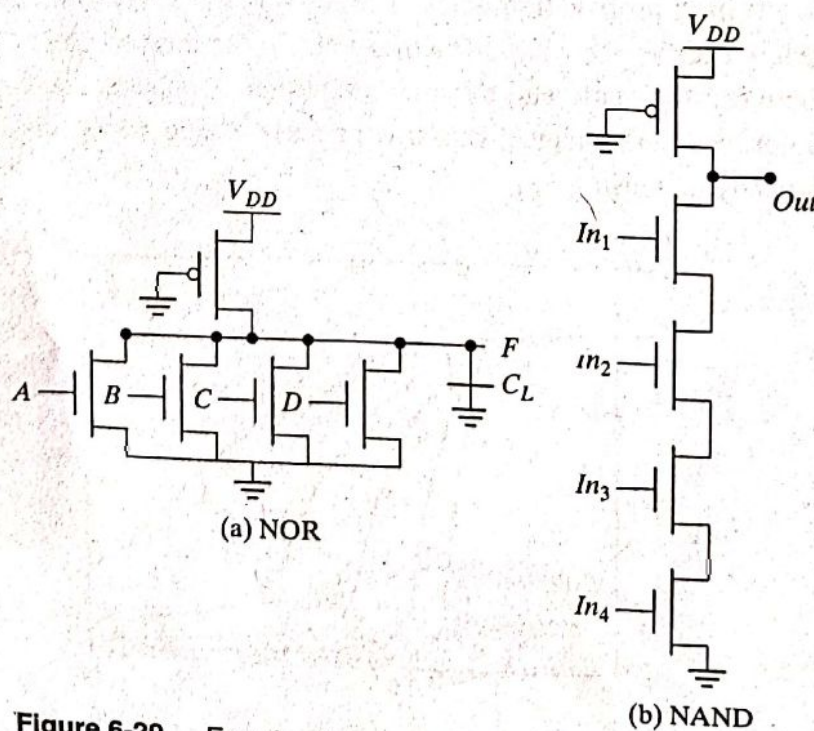


Figure 6-29 Four-input pseudo-NMOS NOR and NAND gates.

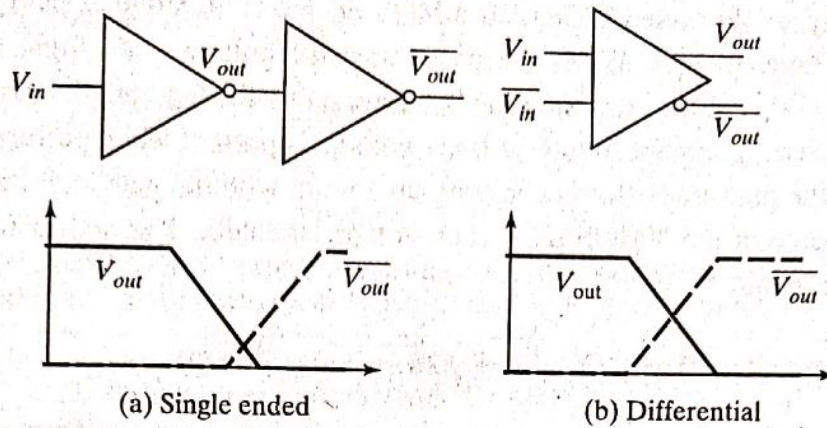


Figure 6-32 Advantage of over single-ended (a) differential (b) gate. ■

6.2.3 Pass-Transistor Logic

Pass-Transistor Basics

A popular and widely used alternative to complementary CMOS is *pass-transistor logic*, which attempts to reduce the number of transistors required to implement logic by allowing the primary inputs to drive gate terminals as well as source-drain terminals [Radhakrishnan85]. This is in contrast to logic families that we have studied so far, which only allow primary inputs to drive the gate terminals of MOSFETS.

Figure 6-33 shows an implementation of the AND function constructed that way, using only NMOS transistors. In this gate, if the B input is high, the top transistor is turned on and copies the input A to the output F . When B is low, the bottom pass-transistor is turned on and passes a 0. The switch driven by \bar{B} seems to be redundant at first glance. Its presence is essential to ensure that the gate is static—a low-impedance path must exist to the supply rails under all circumstances (in this particular case, when B is low).

The promise of this approach is that fewer transistors are required to implement a given function. For example, the implementation of the AND gate in Figure 6-33 requires 4 transistors (including the inverter required to invert B), while a complementary CMOS implementation would require 6 transistors. The reduced number of devices has the additional advantage of lower capacitance.

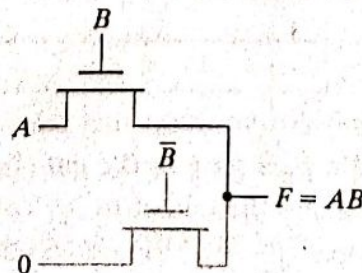


Figure 6-33 Pass-transistor implementation of an AND gate.

Obviously, the number of switches per segment grows with increasing values of t_{buf} . In current technologies, m_{opt} typically equals 3 or 4. The presented analysis ignores that tp_{buf} itself is a function of the load m . A more accurate analysis taking this factor into account is presented in Chapter 9.

Example 6.14 Transmission-Gate Chain

Consider the same 16-transmission-gate chain. The buffers shown in Figure 6-51 can be implemented as inverters (instead of two cascaded inverters). In some cases, it might be necessary to add an extra inverter to produce the correct polarity. Assuming that each inverter is sized such that the NMOS is $0.5 \mu\text{m}/0.25 \mu\text{m}$ and PMOS is $0.5 \mu\text{m}/0.25 \mu\text{m}$, Eq. (6.39) predicts that an inverter must be inserted every 3 transmission gates. The simulated delay when placing an inverter every two transmission gates is 154 ps; for every three transmission gates, the delay is 154 ps; and for four transmission gates, it is 164 ps. The insertion of buffering inverters reduces the delay by a factor of almost 2.

CAUTION: Although many of the circuit styles discussed in the previous sections sound very interesting, and might be superior to static CMOS in many respects, none has the *robustness and ease of design* of complementary CMOS. Therefore, use them sparingly and with caution. For designs that have no extreme area, complexity, or speed constraints, complementary CMOS is the recommended design style.

6.3 Dynamic CMOS Design

It was noted earlier that static CMOS logic with a fan-in of N requires $2N$ devices. A variety of approaches were presented to reduce the number of transistors required to implement a given logic function including pseudo-NMOS, pass-transistor logic, etc. The pseudo-NMOS logic style requires only $N + 1$ transistors to implement an N input logic gate, but unfortunately it has static power dissipation. In this section, an alternate logic style called *dynamic logic* is presented that obtains a similar result, while avoiding static power consumption. With the addition of a clock input, it uses a sequence of *precharge* and conditional *evaluation* phases.

6.3.1 Dynamic Logic: Basic Principles

The basic construction of an (n -type) dynamic logic gate is shown in Figure 6-52a. The PDN (pull-down network) is constructed exactly as in complementary CMOS. The operation of this circuit is divided into two major phases—*precharge* and *evaluation*—with the mode of operation determined by the *clock signal CLK*.

Precharge

When $CLK = 0$, the output node *Out* is precharged to V_{DD} by the PMOS transistor M_p . During that time, the evaluate NMOS transistor M_e is off, so that the pull-down path is disabled. The

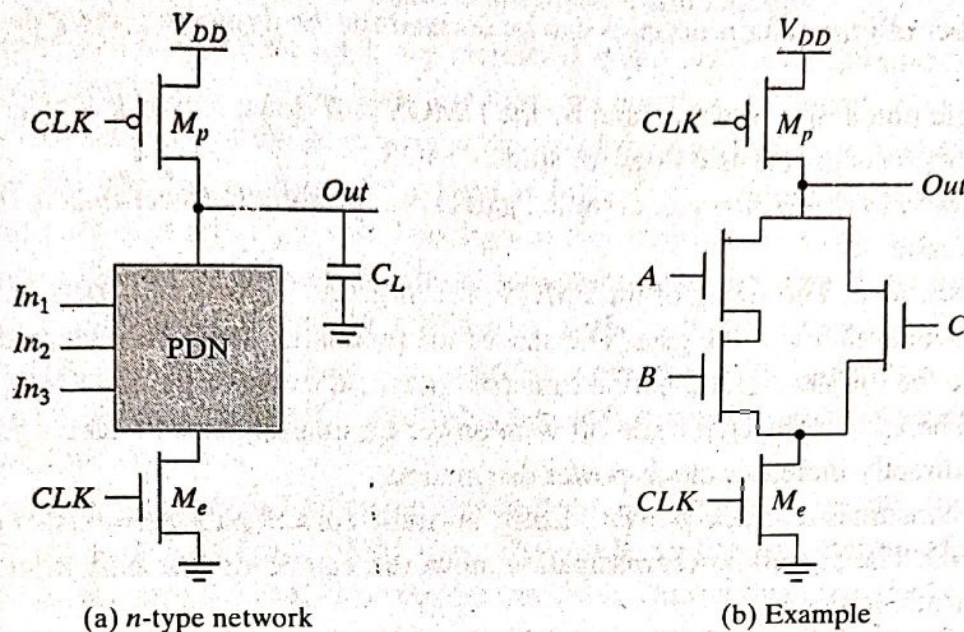


Figure 6-52 Basic concepts of a dynamic gate.

evaluation FET eliminates any static power that would be consumed during the precharge period (i.e., static current would flow between the supplies if both the pull-down and the precharge device were turned on simultaneously).

Evaluation

For $CLK = 1$, the precharge transistor M_p is off, and the evaluation transistor M_e is turned on. The output is conditionally discharged based on the input values and the pull-down topology. If the inputs are such that the PDN conducts, then a low resistance path exists between *Out* and GND, and the output is discharged to GND. If the PDN is turned off, the precharged value remains stored on the output capacitance C_L , which is a combination of junction capacitances, the wiring capacitance, and the input capacitance of the fan-out gates. During the evaluation phase, the only possible path between the output node and a supply rail is to GND. Consequently, once *Out* is discharged, it cannot be charged again until the next precharge operation. **The inputs to the gate can thus make at most one transition during evaluation.** Notice that the output can be in the *high-impedance state* during the evaluation period if the pull-down network is turned off. This behavior is fundamentally different from the static counterpart that always has a low resistance path between the output and one of the power rails.

As an example, consider the circuit shown in Figure 6-52b. During the precharge phase ($CLK = 0$), the output is precharged to V_{DD} regardless of the input values, because the evaluation device is turned off. During evaluation ($CLK = 1$), a conducting path is created between *Out* and GND if (and only if) $A \cdot B + C$ is TRUE. Otherwise, the output remains at the precharged state of V_{DD} . The following function is thus realized:

$$Out = \overline{CLK} + \overline{(A \cdot B + C)} \cdot CLK \quad (6.40)$$

6.3.2 Speed and Power Dissipation of Dynamic Logic

The main advantages of dynamic logic are increased speed and reduced implementation area. Fewer devices to implement a given logic function implies that the overall load capacitance is much smaller. The analysis of the switching behavior of the gate has some interesting peculiarities to it. After the precharge phase, the output is high. For a low input signal, no additional switching occurs. As a result, $t_{pLH} = 0$! The high-to-low transition, on the other hand, requires the discharging of the output capacitance through the pull-down network. Therefore, t_{pHL} is proportional to C_L and the current-sinking capabilities of the pull-down network. The presence of the evaluation transistor slows the gate somewhat, as it presents an extra series resistance. Omitting this transistor, while functionally not forbidden, may result in static power dissipation and potentially a performance loss.

The preceding analysis is somewhat unfair because it ignores the influence of the precharge time on the switching speed of the gate. The precharge time is determined by the time it takes to charge C_L through the PMOS precharge transistor. During this time, the logic in the gate cannot be utilized. Very often, however, the overall digital system can be designed in such a way that the precharge time coincides with other system functions. For instance, the precharge of the arithmetic unit in a microprocessor could coincide with the instruction decode. The designer has to be aware of this "dead zone" in the use of dynamic logic and thus should carefully consider the pros and cons of its usage, taking the overall system requirements into account.

Example 6.15 A Four-Input Dynamic NAND Gate

Figure 6-53 shows the design of a four-input NAND example designed using the dynamic-circuit style. Due to the dynamic nature of the gate, the derivation of the voltage-transfer

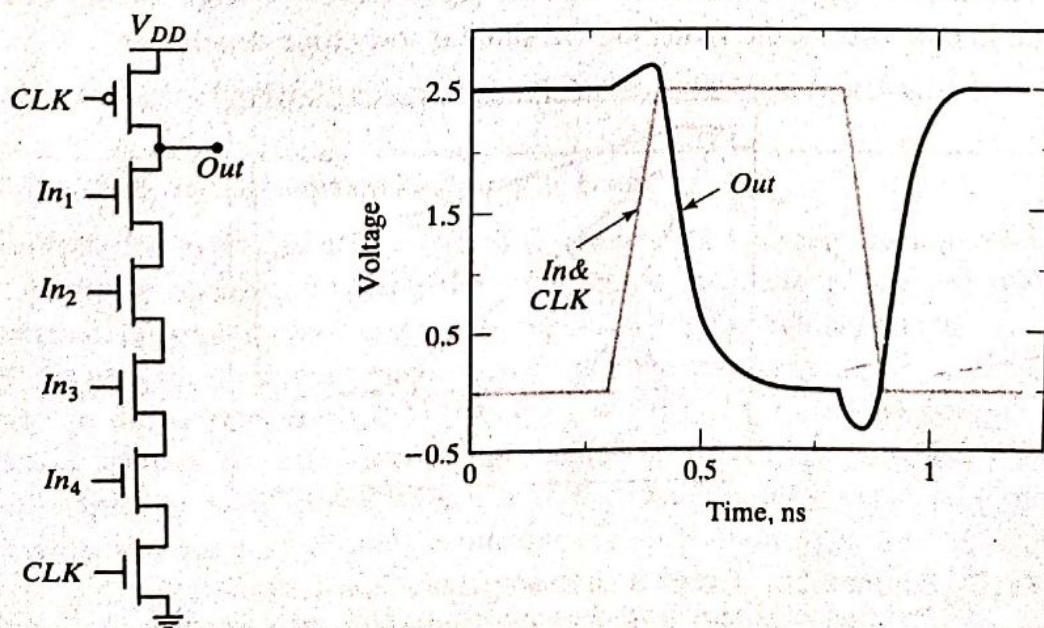


Figure 6-53 Schematic and transient response of a four-input dynamic NAND gate.

characteristic diverges from the traditional approach. As discussed earlier, we assume that the switching threshold of the gate equals the threshold of the NMOS pull-down transistor. This results in asymmetrical noise margins, as shown in Table 6-10.

Table 6-10 The dc and ac parameters of a four-input dynamic NAND.

Transistors	V_{OH}	V_{OL}	V_M	NM_H	NM_L	t_{pHL}	t_{pLH}	t_{pre}
6	2.5 V	0 V	V_{TN}	$2.5 - V_{TN}$	V_{TN}	110 ps	0 ps	83 ps

The dynamic behavior of the gate is simulated with SPICE. It is assumed that all inputs are set high when the clock goes high. On the rising edge of the clock, the output node is discharged. The resulting transient response is plotted in Figure 6-53, and the propagation delays are summarized in Table 6-10. The duration of the precharge cycle can be adjusted by changing the size of the PMOS precharge transistor. Making the PMOS too large should be avoided, however, as it both slows down the gate and increases the capacitive load on the clock line. For large designs, the latter factor might become a major design concern because the clock load can become excessive and hard to drive.

As mentioned earlier, the static gate parameters are time dependent. To illustrate this, consider a four-input NAND gate with all the partial inputs tied together, and are making a low-to-high transition. Figure 6-54 shows a transient simulation of the output voltage for three different input transitions—from 0 to 0.45 V, 0.5 V and 0.55 V, respectively. In the preceding discussion, we have defined the switching threshold of the dynamic gate as the device threshold. However, notice that the amount by which the output voltage drops is a strong function of the input voltage and the *available evaluation time*. The noise voltage needed to corrupt the signal has to be larger if the evaluation time is short. In other words, the switching threshold is truly time dependent.

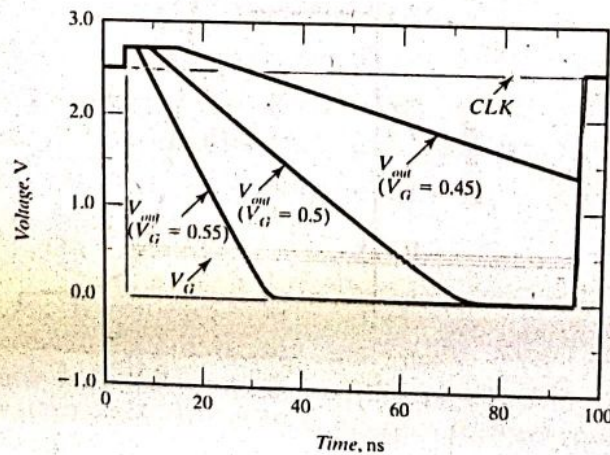


Figure 6-54 Effect of an input glitch on the output. The switching threshold depends on the time for evaluation. A larger glitch is acceptable if the evaluation phase is shorter.

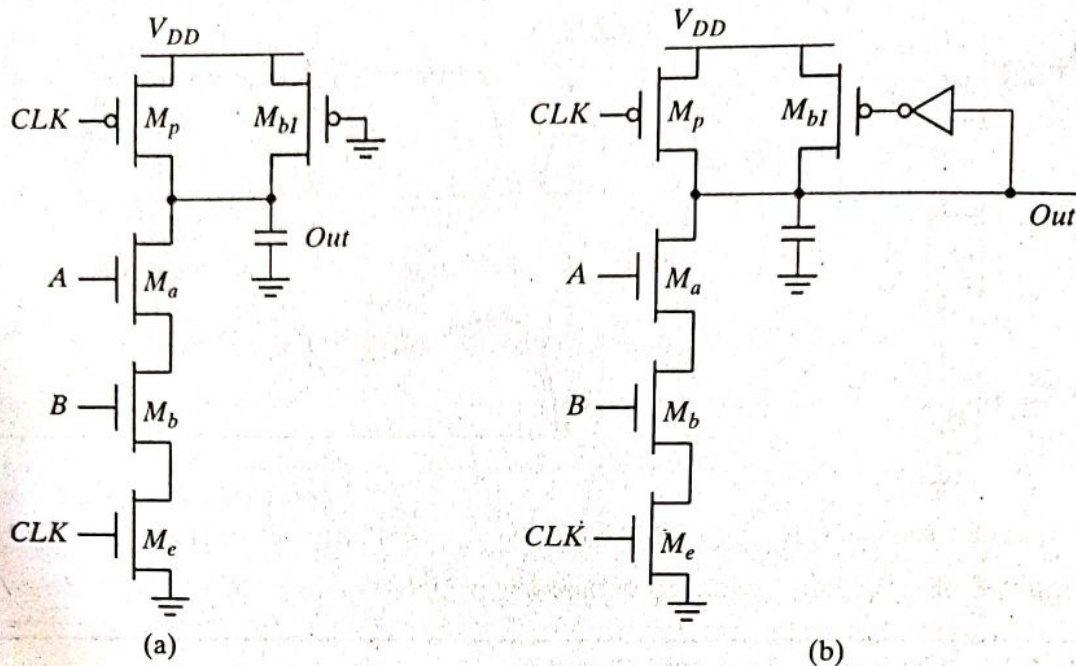


Figure 6-58 Static bleeders compensate for the charge leakage.

adding a *bleeder transistor*, as shown in Figure 6-58a. The only function of the bleeder—an NMOS style pull-up device—is to compensate for the charge lost due to the pull-down leakage paths. To avoid the ratio problems associated with this style of circuit and the associated static power consumption, the bleeder resistance is made high (in other words, the device is kept small). This allows the (strong) pull-down devices to lower the *Out* node substantially below the switching threshold of the next gate. Often, the bleeder is implemented in a feedback configuration to eliminate the static power dissipation altogether (Figure 6-58b).

Charge Sharing

Another important concern in dynamic logic is the impact of charge sharing. Consider the circuit in Figure 6-59. During the precharge phase, the output node is precharged to V_{DD} . Assume that all inputs are set to 0 during precharge, and that the capacitance C_a is discharged. Assume further that input *B* remains at 0 during evaluation, while input *A* makes a $0 \rightarrow 1$ transition, turning transistor M_a on. The charge stored originally on capacitor C_L is redistributed over C_L and C_a . This causes a drop in the output voltage, which cannot be recovered due to the dynamic nature of the circuit.

The influence on the output voltage is readily calculated. Under the assumptions given previously, the following initial conditions are valid: $V_{out}(t=0) = V_{DD}$ and $V_X(t=0) = 0$. As a result, two possible scenarios must be considered:

1. $\Delta V_{out} < V_{Tn}$. In this case, the final value of V_X equals $V_{DD} - V_{Tn}(V_X)$. Charge conservation then yields

$$\begin{aligned}
 C_L V_{DD} &= C_L V_{out}(\text{final}) + C_a [V_{DD} - V_{Tn}(V_X)] \\
 \text{or} \\
 \Delta V_{out} &= V_{out}(\text{final}) + (-V_{DD}) = -\frac{C_a}{C_L} [V_{DD} - V_{Tn}(V_X)]
 \end{aligned}
 \tag{6.43}$$

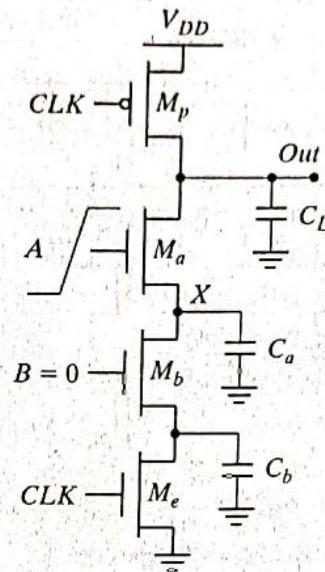


Figure 6-59 Charge sharing in dynamic networks.

2. $\Delta V_{out} > V_{Tn}$. V_{out} and V_X then reach the same value:

$$\Delta V_{out} = -V_{DD} \left(\frac{C_a}{C_a + C_L} \right) \tag{6.44}$$

We determine which of these scenarios is valid by the capacitance ratio. The boundary condition between the two cases can be determined by setting ΔV_{out} equal to V_{Tn} in Eq. (6.44), yielding

$$\frac{C_a}{C_L} = \frac{V_{Tn}}{V_{DD} - V_{Tn}} \tag{6.45}$$

Case 1 holds when the (C_a/C_L) ratio is smaller than the condition defined in Eq. (6.45). If not, Eq. (6.44) is valid. Overall, it is desirable to keep the value of ΔV_{out} below $|V_{Tp}|$. The output of the dynamic gate might be connected to a static inverter, in which case the low level of V_{out} would cause static power consumption. One major concern is a circuit malfunction if the output voltage is brought below the switching threshold of the gate it drives.

Example 6.18 Charge Sharing

Let us consider the impact of charge sharing on the dynamic logic gate shown in Figure 6-60, which implements a three-input EXOR function $y = A \oplus B \oplus C$. The first question to be resolved is what conditions cause the worst case voltage drop on node y. For simplicity, ignore the load inverter, and assume that all inputs are low during the precharge operation and that all isolated internal nodes ($V_a, V_b, V_c,$ and V_d) are initially at 0 V.

Inspection of the truth table for this particular logic function shows that the output stays high for 4 out of 8 cases. The worst case change in output is obtained by exposing the maximum amount of internal capacitance to the output node during the evaluation

to V_{DD} during precharge, charge sharing does not occur. This solution obviously comes at the cost of increased area and capacitance.

Capacitive Coupling

The relatively high impedance of the output node makes the circuit very sensitive to crosstalk effects. A wire routed over or next to a dynamic node may couple capacitively and destroy the state of the floating node. Another equally important form of capacitive coupling is *backgate* (or *output-to-input*) *coupling*. Consider the circuit shown in Figure 6-62a, in which a dynamic two-input NAND gate drives a static NAND gate. A transition in the input In of the static gate may cause the output of the gate (Out_2) to go low. This output transition couples capacitively to the other input of the gate (the dynamic node Out_1) through the gate-source and gate-drain capacitances of transistor M_4 . A simulation of this effect is shown in Figure 6-62b. It demonstrates how the coupling causes the output of the dynamic gate Out_1 to drop significantly. This further causes the output of the static NAND gate not to drop all the way down to 0 V and a small amount of static power to be dissipated. If the voltage drop is large enough, the circuit can evaluate incorrectly, and the NAND output may not go low. When designing and laying out dynamic circuits, special care is needed to minimize capacitive coupling.

Clock Feedthrough

A special case of capacitive coupling is clock feedthrough, an effect caused by the capacitive coupling between the clock input of the precharge device and the dynamic output node. The coupling capacitance consists of the gate-to-drain capacitance of the precharge device, and includes both the overlap and channel capacitances. This capacitive coupling causes the output of the dynamic node to rise above V_{DD} on the low-to-high transition of the clock, assuming that the pull-down network is turned off. Subsequently, the fast rising and falling edges of the clock couple onto the signal node, as is quite apparent in the simulation of Figure 6-62b.

The danger of clock feedthrough is that it may cause the normally reverse-biased junction diodes of the precharge transistor to become forward biased. This causes electron injection into the substrate, which can be collected by a nearby high-impedance node in the 1 state, eventually resulting in faulty operation. CMOS latchup might be another result of this injection. For all purposes, high-speed dynamic circuits should be carefully simulated to ensure that clock feedthrough effects stay within bounds.

All of the preceding considerations demonstrate that the design of dynamic circuits is rather tricky and requires extreme care. It should therefore be attempted only when high performance is required, or high quality design-automation tools are available.

6.3.4 Cascading Dynamic Gates

Besides the signal integrity issues, there is one major catch that complicates the design of dynamic circuits: Straightforward cascading of dynamic gates to create multilevel logic structures does not work. The problem is best illustrated with two cascaded n -type dynamic

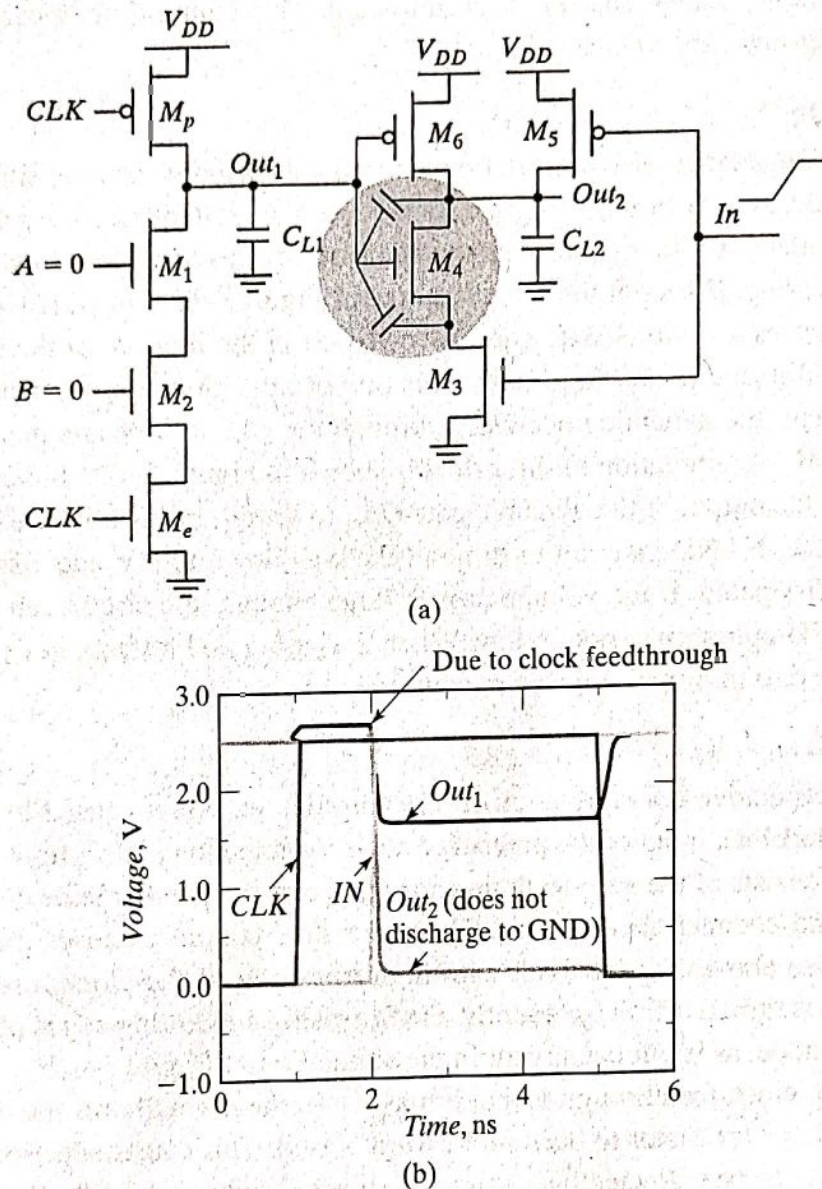


Figure 6-62 Example demonstrating the effect of backgate coupling: (a) circuit schematics; (b) simulation results.

inverters, shown in Figure 6-63a. During the precharge phase (i.e., $CLK = 0$), the outputs of both inverters are precharged to V_{DD} . Assume that the primary input In makes a $0 \rightarrow 1$ transition (Figure 6-63b). On the rising edge of the clock, output Out_1 starts to discharge. The second output should remain in the precharged state of V_{DD} as its expected value is 1 (Out_1 transitions to 0 during evaluation). However, there is a finite propagation delay for the input to discharge Out_1 to GND. Therefore, the second output also starts to discharge. As long as Out_1 exceeds the switching threshold of the second gate, which approximately equals V_{Tn} , a conducting path exists between Out_2 and GND, and precious charge is lost at Out_2 . The conducting path is only disabled once Out_1 reaches V_{Tn} , and turns off the NMOS pull-down transistor. This leaves Out_2 at an intermediate voltage level. The correct level will not be recovered, because dynamic gates rely on capacitive storage, in contrast to static gates, which have dc restoration. The charge loss leads to reduced noise margins and potential malfunctioning.

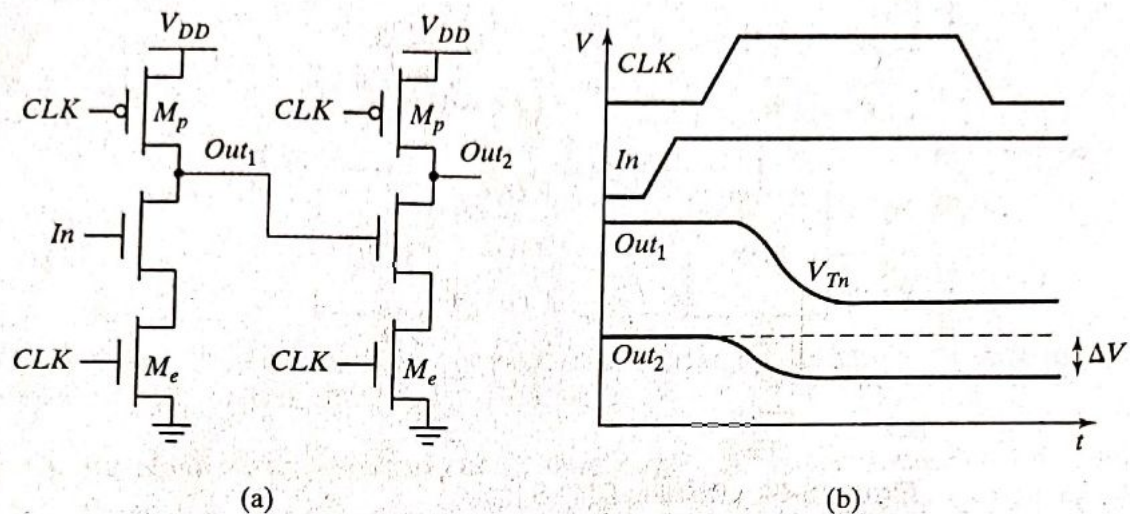


Figure 6-63 Cascade of dynamic n -type blocks.

The cascading problem arises because the outputs of each gate—and thus the inputs to the next stages—are precharged to 1. This may cause inadvertent discharge in the beginning of the evaluation cycle. Setting all the inputs to 0 during precharge addresses that concern. When doing so, all transistors in the pull-down network are turned off after precharge, and no inadvertent discharging of the storage capacitors can occur during evaluation. In other words, correct operation is guaranteed as long as **the inputs can only make a single 0 → 1 transition during the evaluation period.**⁵ Transistors are turned on only when needed—and at most, once per cycle. A number of design styles complying with this rule have been conceived, but the two most important ones are discussed next.

Domino Logic

Concept A domino logic module [Krambeck82] consists of an n -type dynamic logic block followed by a static inverter (Figure 6-64). During precharge, the output of the n -type dynamic gate is charged up to V_{DD} , and the output of the inverter is set to 0. During evaluation, the dynamic gate conditionally discharges, and the output of the inverter makes a conditional transition from 0 → 1. If one assumes that all the inputs of a domino gate are outputs of other domino gates,⁶ then it is ensured that all inputs are set to 0 at the end of the precharge phase, and that the only transitions during evaluation are 0 → 1 transitions. Hence, the formulated rule is obeyed. The introduction of the static inverter has the additional advantage that the fan-out of the gate is driven by a static inverter with a low-impedance output, which increases noise immunity. Also, the buffer reduces the capacitance of the dynamic output node by separating internal and load capacitances. Finally, the inverter can be used to drive a bleeder device to combat leakage and charge redistribution, as shown in the second stage of Figure 6-64.

⁵This ignores the impact of charge distribution and leakage effects, discussed earlier.

⁶It is required that all other inputs that do not fall under this classification (for instance, primary inputs) stay constant during evaluation.

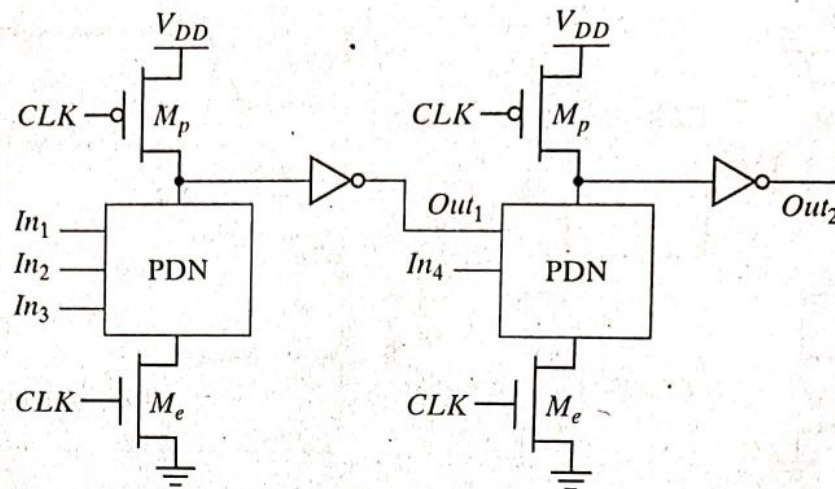


Figure 6-64 Domino CMOS logic.

Consider now the operation of a chain of domino gates. During precharge, all inputs are set to 0. During evaluation, the output of the first domino block either stays at 0 or makes a $0 \rightarrow 1$ transition, affecting the second gate. This effect might ripple through the whole chain, one after the other, similar to a line of falling dominoes—hence the name. Domino CMOS has the following properties:

- Since each dynamic gate has a static inverter, only noninverting logic can be implemented. Although there are ways to deal with this, as discussed in a subsequent section, this is a major limiting factor, and pure domino design has thus become rare.
- Very high speeds can be achieved: only a rising edge delay exists, while t_{pHL} equals zero. The inverter can be sized to match the *fan-out*, which is already much smaller than in the complimentary static CMOS case, as only a single gate capacitance has to be accounted for per fan-out gate.

Since the inputs to a domino gate are low during precharge, it is tempting to eliminate the evaluation transistor because this reduces clock load and increases pull-down drive. However, eliminating the evaluation device extends the precharge cycle—the precharge now has to ripple through the logic network as well. Consider the logic network shown in Figure 6-65, where the evaluation devices have been eliminated. If the primary input In_1 is 1 during evaluation, the output of each dynamic gate evaluates to 0, and the output of each static inverter is 1. On the falling edge of the clock, the precharge operation is started. Assume further that In_1 makes a high-to-low transition. The input to the second gate is initially high, and it takes two gate delays before In_2 is driven low. During that time, the second gate cannot precharge its output, as the pull-down network is fighting the precharge device. Similarly, the third gate has to wait until the second gate precharges before it can start precharging, etc. Therefore, the time taken to precharge the logic circuit is equal to its critical path. Another important negative is the extra power dissipation when both pull-up and pull-down devices are on. Therefore, it is good practice to always utilize evaluation devices.

- 7.6 Nonbistable Sequential Circuits
 - 7.6.1 The Schmitt Trigger
 - 7.6.2 Monostable Sequential Circuits
 - 7.6.3 Astable Circuits
- 7.7 Perspective: Choosing a Clocking Strategy
- 7.8 Summary
- 7.9 To Probe Further

7.1 Introduction

As described earlier, combinational logic circuits have the property that the output of a logic block is only a function of the *current* input values, assuming that enough time has elapsed for the logic gates to settle. Still, virtually all useful systems require storage of state information, leading to another class of circuits called *sequential logic* circuits. In these circuits, the output depends not only on the *current* values of the inputs, but also on *preceding* input values. In other words, a sequential circuit remembers some of the past history of the system—it has memory.

Figure 7-1 shows a block diagram of a generic *finite-state machine* (FSM) that consists of combinational logic and registers, which hold the system state. The system depicted here belongs to the class of *synchronous* sequential systems, in which all registers are under control of a single global clock. The outputs of the FSM are a function of the current *Inputs* and the *Current State*. The *Next State* is determined based on the *Current State* and the current *Inputs* and is fed to the inputs of registers. On the rising edge of the clock, the *Next State* bits are copied to the outputs of the registers (after some propagation delay), and a new cycle begins. The register then ignores changes in the input signals until the next rising edge. In general, registers can be *positive edge triggered* (where the input data is copied on the rising edge of the clock) or *negative edge triggered* (where the input data is copied on the falling edge, as indicated by a small circle at the clock input)

This chapter discusses the CMOS implementation of the most important sequential building blocks. A variety of choices in sequential primitives and clocking methodologies exist; making the correct selection is getting increasingly important in modern digital circuits, and can

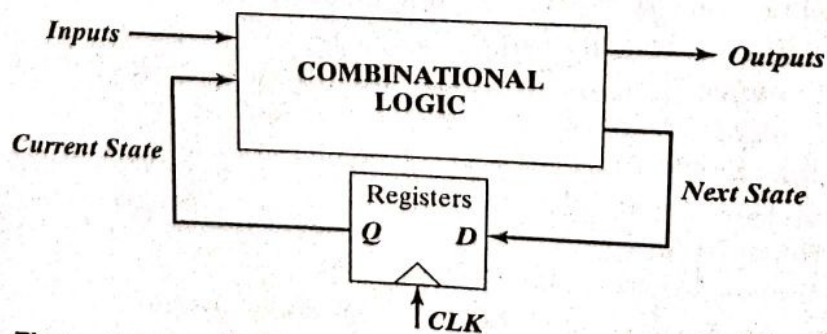


Figure 7-1 Block diagram of a finite-state machine, using positive edge-triggered registers.

have a great impact of performance, power, and/or design complexity. Before embarking on a detailed discussion of the various design options, a review of the relevant design metrics and a classification of the sequential elements is necessary.

7.1.1 Timing Metrics for Sequential Circuits

There are three important timing parameters associated with a register. They are shown in Figure 7-2. The *setup time* (t_{su}) is the time that the data inputs (D) must be valid before the clock transition (i.e., the $0 \rightarrow 1$ transition for a *positive edge-triggered* register). The *hold time* (t_{hold}) is the time the data input must remain valid after the clock edge. Assuming that the *setup* and *hold* times are met, the data at the D input is copied to the Q output after a worst case *propagation delay* (with reference to the clock edge) denoted by t_{c-q} .

Once we know the timing information for the registers and the combinational logic blocks, we can derive the system-level timing constraints (see Figure 7-1 for a simple system view). In synchronous sequential circuits, switching events take place concurrently in response to a clock stimulus. Results of operations await the next clock transitions before progressing to the next stage. In other words, the next cycle cannot begin unless all current computations have completed and the system has come to rest. The *clock period* T , at which the sequential circuit operates, must thus accommodate the longest delay of any stage in the network. Assume that the worst case propagation delay of the logic equals t_{plogic} , while its minimum delay—also called the *contamination delay*—is t_{cd} . The minimum clock period T required for proper operation of the sequential circuit is given by

$$T \geq t_{c-q} + t_{plogic} + t_{su} \quad (7.1)$$

The *hold time* of the register imposes an extra constraint for proper operation, namely

$$t_{cdregister} + t_{cdlogic} \geq t_{hold} \quad (7.2)$$

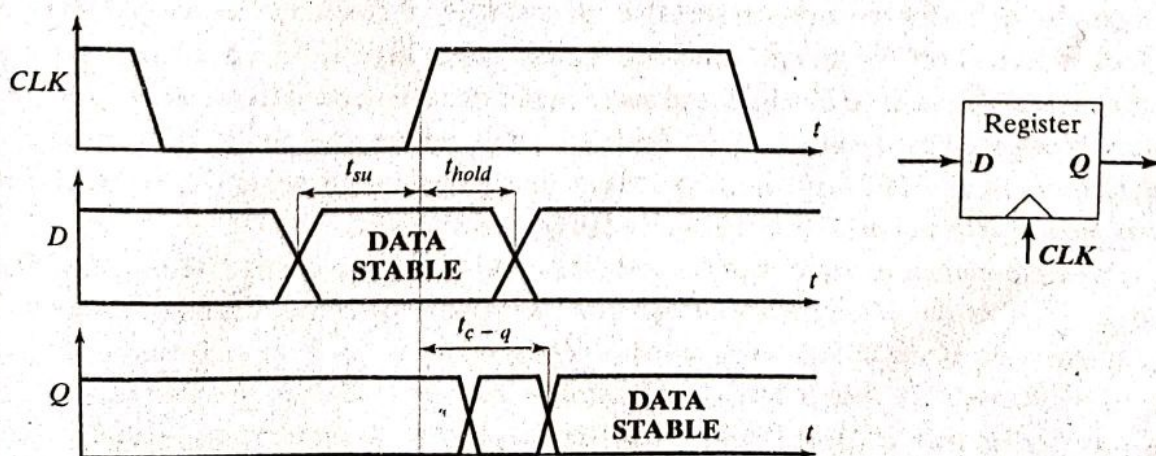


Figure 7-2 Definition of *setup time*, *hold time*, and *propagation delay* of a synchronous register.

where $t_{cdregister}$ is the minimum *propagation delay* (or *contamination delay*) of the register. This constraint ensures that the input data of the sequential elements is held long enough after the clock edge and is not modified too soon by the new wave of data coming in.

As seen from Eq. (7.1), it is important to minimize the values of the timing parameters associated with the register, as these directly affect the rate at which a sequential circuit can be clocked. In fact, modern high-performance systems are characterized by a very low logic depth, and the register *propagation delay* and *setup* times account for a significant portion of the clock period. For example, the DEC Alpha EV6 microprocessor [Gieseke97] has a maximum logic depth of 12 gates, and the register overhead stands for approximately 15% of the clock period. In general, the requirement of Eq. (7.2) is not difficult to meet, although it becomes an issue when there is little or no logic between registers.¹

7.1.2 Classification of Memory Elements

Foreground versus Background Memory

At a high level, memory is classified into background and foreground memory. Memory that is embedded into logic is *foreground memory* and is most often organized as individual registers or register banks. Large amounts of centralized memory core are referred to as *background memory*. Background memory, discussed in Chapter 12, achieves higher area densities through efficient use of array structures and by trading off performance and robustness for size. In this chapter, we focus on foreground memories.

Static versus Dynamic Memory

Memories can be either static or dynamic. Static memories preserve the state as long as the power is turned on. They are built by using *positive feedback* or regeneration, where the circuit topology consists of intentional connections between the output and the input of a combinational circuit. Static memories are most useful when the register will not be updated for extended periods of time. Configuration data, loaded at power-up time, is a good example of static data. This condition also holds for most processors that use conditional clocking (i.e., gated clocks) where the clock is turned off for unused modules. In that case, there are no guarantees on how frequently the registers will be clocked, and static memories are needed to preserve the state information. Memory based on positive feedback falls under the class of elements called *multivibrator circuits*. The *bistable* element is its most popular representative, but other elements such as *monostable* and *astable* circuits also are frequently used.

Dynamic memories store data for a short period of time, perhaps milliseconds. They are based on the principle of temporary *charge storage* on parasitic capacitors associated with MOS devices. As with dynamic logic, discussed earlier, the capacitors have to be refreshed periodically to compensate for charge leakage. Dynamic memories tend to be simpler, resulting in significantly higher performance and lower power dissipation. They are most useful in datapath

¹Or when the clocks at different registers are somewhat out of phase due to clock skew. We discuss this topic in Chapter 10.

circuits that require high performance levels and are periodically clocked. It is possible to use dynamic circuitry even when circuits are conditionally clocked, if the state can be discarded when a module goes into idle mode.

✓ Latches versus Registers

A latch is an essential component in the construction of an *edge-triggered* register. It is a *level-sensitive* circuit that passes the *D* input to the *Q* output when the clock signal is high. This latch is said to be in *transparent* mode. When the clock is low, the input data sampled on the falling edge of the clock is held stable at the output for the entire phase, and the latch is in *hold* mode. The inputs must be stable for a short period around the falling edge of the clock to meet setup and hold requirements. A latch operating under these conditions is a *positive latch*. Similarly, a *negative latch* passes the *D* input to the *Q* output when the clock signal is low. Positive and negative latches are also called *transparent high* or *transparent low*, respectively. The signal waveforms for a positive and negative latch are shown in Figure 7-3. A wide variety of static and dynamic implementations exists for the realization of latches.

Contrary to *level-sensitive* latches, *edge-triggered* registers only sample the input on a clock transition—that is, $0 \rightarrow 1$ for a *positive edge-triggered* register, and $1 \rightarrow 0$ for a *negative edge-triggered* register. They are typically built to use the latch primitives of Figure 7-3. An often-recurring configuration is the *master-slave* structure, that cascades a positive and negative latch. Registers also can be constructed by using one-shot generators of the clock signal (“glitch” registers), or by using other specialized structures. Examples of these are shown later in this chapter.

The literature on sequential circuits has been plagued by ambiguous definitions for the different types of storage elements (i.e., register, flip-flop, and latch). To avoid confusion, we adhere strictly to the following set of definitions in this book:

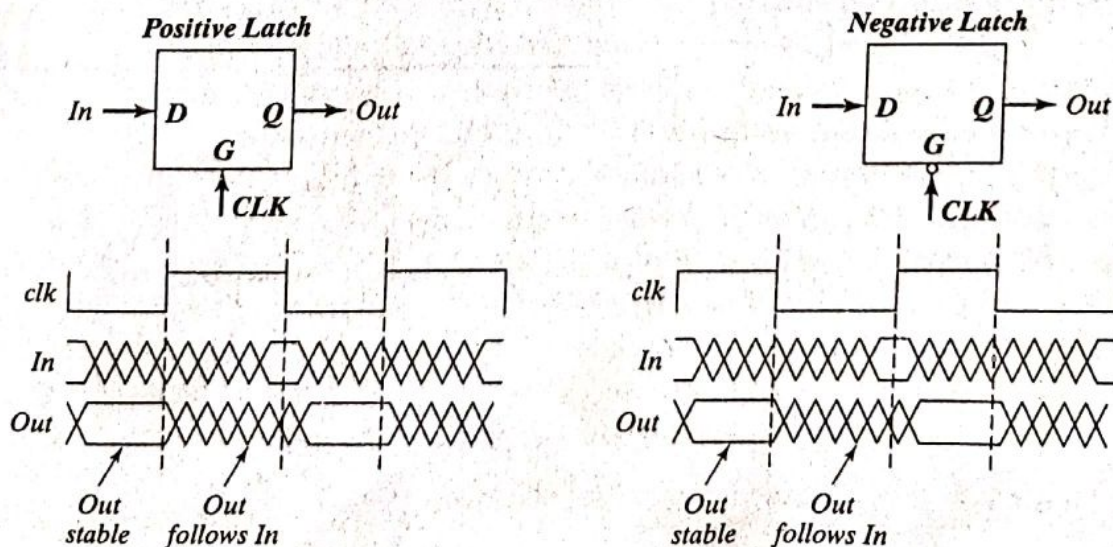


Figure 7-3 Timing of positive and negative latches.

- An *edge-triggered* storage element is called a *register*;
- A *latch* is a *level-sensitive* device;
- and any *bistable* component, formed by the cross coupling of gates, is called a *flip-flop*.²

7.2 Static Latches and Registers

7.2.1 The Bistability Principle

Static memories use positive feedback to create a *bistable circuit*—a circuit having two stable states that represent 0 and 1. The basic idea is shown in Figure 7-4a, which shows two inverters connected in cascade along with a voltage-transfer characteristic typical of such a circuit. Also plotted are the VTCs of the first inverter—that is, V_{o1} versus V_{i1} —and the second inverter (V_{o2} versus V_{i2}). The latter plot is rotated to accentuate that $V_{i2} = V_{o1}$. Assume now that the output of the second inverter V_{o2} is connected to the input of the first V_{i1} , as shown by the dotted lines in Figure 7-4a. The resulting circuit has only three possible operation points (A, B, and C), as demonstrated on the combined VTC. It is easy to prove the validity of the following important conjecture:

When the gain of the inverter in the transient region is larger than 1, A and B are the only stable operation points, and C is a metastable operation point.

Suppose that the cross-coupled inverter pair is biased at point C. A small deviation from this bias point, possibly caused by noise, is amplified and *regenerated* around the circuit loop.

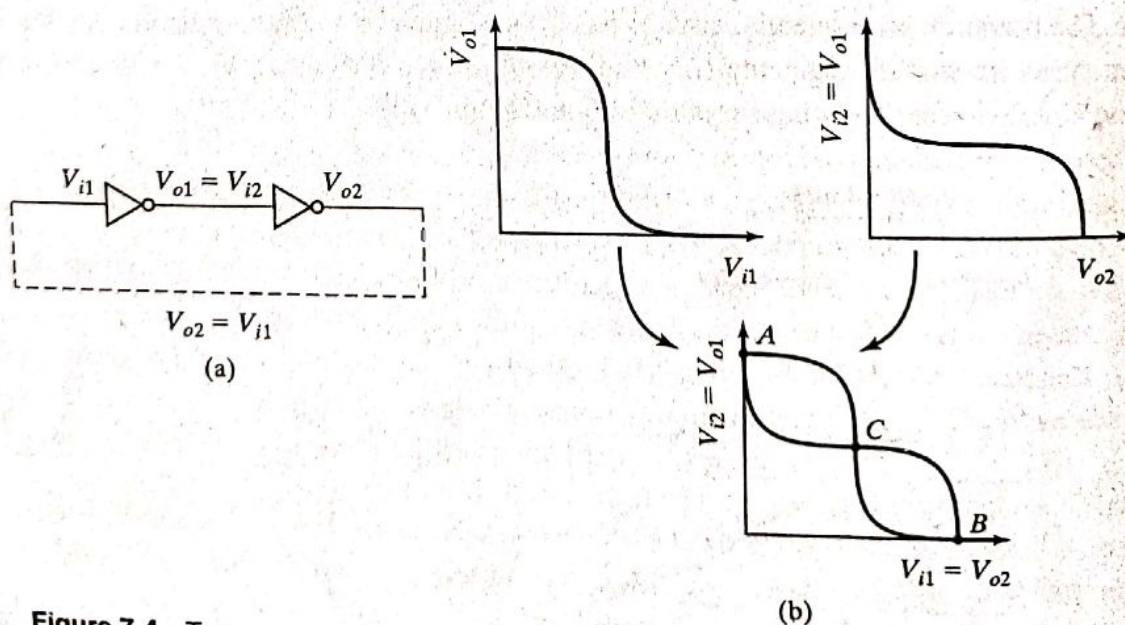


Figure 7-4 Two cascaded inverters (a) and their VTCs (b).

²An edge-triggered register is often referred to as a flip-flop as well. In this text, flip-flop is used to uniquely mean bistable element.

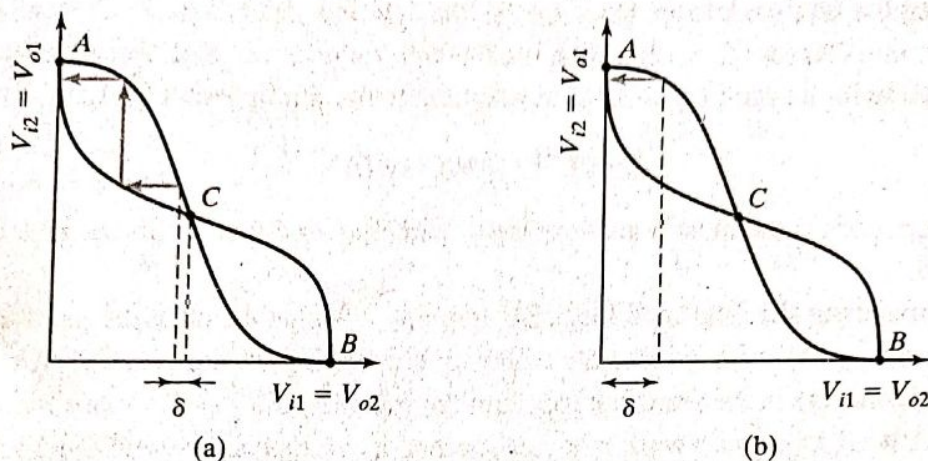


Figure 7-5 Metastable versus stable operation points.

This is a result of the gain around the loop being larger than 1. The effect is demonstrated in Figure 7-5a. A small deviation δ is applied to V_{i1} (biased in C). This deviation is amplified by the gain of the inverter. The enlarged divergence is applied to the second inverter and amplified once more. The bias point moves away from C until one of the operation points A or B is reached. In conclusion, C is an unstable operation point. Every deviation (even the smallest one) causes the operation point to run away from its original bias. The chance is indeed very small that the cross-coupled inverter pair is biased at C and stays there. Operation points with this property are termed *metastable*.

On the other hand, A and B are stable operation points, as demonstrated in Figure 7-5b. In these points, the **loop gain is much smaller than unity**. Even a rather large deviation from the operation point reduces in size and disappears.

Hence, the cross coupling of two inverters results in a *bistable* circuit—that is, a circuit with two stable states, each corresponding to a logic state. The circuit serves as a memory, storing either a 1 or a 0 (corresponding to positions A and B).

In order to change the stored value, we must be able to bring the circuit from state A to B and vice versa. Since the precondition for stability is that the loop gain G is smaller than unity, we can achieve this by making A (or B) temporarily unstable by increasing G to a value larger than 1. This is generally done by applying a trigger pulse at V_{i1} or V_{i2} . For example, assume that the system is in position A ($V_{i1} = 0$, $V_{i2} = 1$). Forcing V_{i1} to 1 causes both inverters to be on simultaneously for a short time and the loop gain G to be larger than 1. The positive feedback regenerates the effect of the trigger pulse, and the circuit moves to the other state (B, in this case). The width of the trigger pulse need be only a little larger than the total propagation delay around the circuit loop, which is twice the average propagation delay of the inverters.

In summary, a bistable circuit has two stable states. In absence of any triggering, the circuit remains in a single state (assuming that the power supply remains applied to the circuit) and thus remembers a value. Another common name for a bistable circuit is *flip-flop*. A flip-flop is useful only if there also exists a means to bring it from one state to the other one. In general, two different approaches may be used to accomplish the following:

- **Cutting the feedback loop.** Once the feedback loop is open, a new value can easily be written into *Out* (or *Q*). Such a latch is called *multiplexer based*, as it realizes that the logic expression for a synchronous latch is identical to the multiplexer equation:

$$Q = \overline{Clk} \cdot Q + Clk \cdot In \quad (7.3)$$

This approach is the most popular in today's latches, and thus forms the bulk of this section.

- **Overpowering the feedback loop.** By applying a trigger signal at the input of the flip-flop, a new value is forced into the cell by overpowering the stored value. A careful sizing of the transistors in the feedback loop and the input circuitry is necessary to make this possible. A weak trigger network may not succeed in overruling a strong feedback loop. This approach used to be in vogue in the earlier days of digital design, but has gradually fallen out of favor. It is, however, the dominant approach to the implementation of static background memories (which we discuss more fully in Chapter 12). A short introduction will be given later in the chapter.

7.2.2 ✓ Multiplexer-Based Latches

The most robust and common technique to build a latch involves the use of transmission-gate multiplexers. Figure 7-6 shows an implementation of positive and negative static latches based on multiplexers. For a negative latch, input 0 of the multiplexer is selected when the clock is low, and the *D* input is passed to the output. When the clock signal is high, input 1 of the multiplexer, which connects to the output of the latch, is selected. The feedback ensures a stable output as long as the clock is high. Similarly, in the positive latch, the *D* input is selected when the clock signal is high, and the output is held (using feedback) when the clock signal is low.

A transistor-level implementation of a positive latch based on multiplexers is shown in Figure 7-7. When *CLK* is high, the bottom transmission gate is on and the latch is transparent—that is, the *D* input is copied to the *Q* output. During this phase, the feedback loop is open, since the top transmission gate is off. Sizing of the transistors therefore is not critical for realizing correct functionality. The number of transistors that the clock drives is an important metric from a power perspective, because the clock has an *activity factor* of 1. This particular latch implemen-

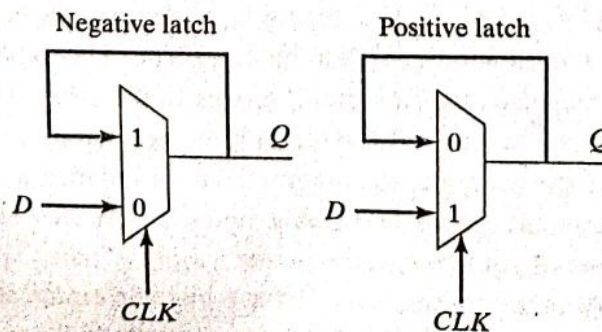


Figure 7-6 Negative and positive latches based on multiplexers.

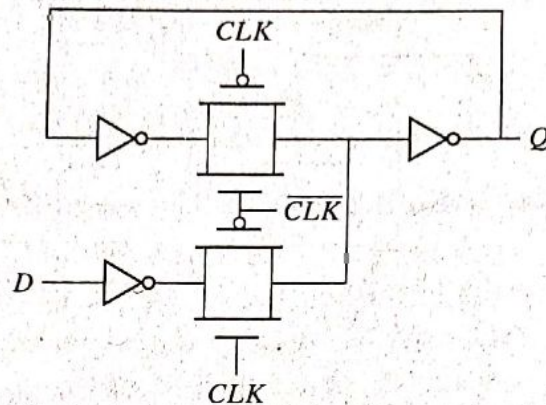


Figure 7-7 Transistor-level implementation of a positive latch built by using transmission gates.

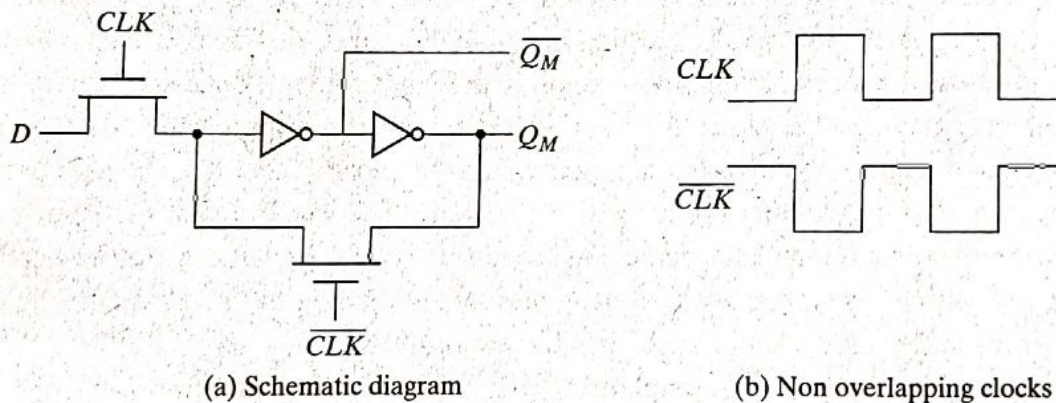


Figure 7-8 Multiplexer-based NMOS latch by using NMOS-only pass transistors for multiplexers.

tation is not very efficient from this perspective: It presents a load of four transistors to the CLK signal.

It is possible to reduce the clock load to two transistors by implementing multiplexers that use as NMOS-only pass transistors, as shown in Figure 7-8. When CLK is high, the latch samples the D input, while a low clock signal enables the feedback loop, and puts the latch in the hold mode. While attractive for its simplicity, the use of NMOS-only pass transistors results in the passing of a degraded high voltage of $V_{DD} - V_{Tn}$ to the input of the first inverter. This impacts both noise margin and the switching performance, especially in the case of low values of V_{DD} and high values of V_{Tn} . It also causes static power dissipation in the first inverter, because the maximum input voltage to the inverter equals $V_{DD} - V_{Tn}$, and the PMOS device of the inverter is never fully turned off.

7.2.3 Master-Slave Edge-Triggered Register

The most common approach for constructing an *edge-triggered register* is to use a *master-slave* configuration, as shown in Figure 7-9. The register consists of cascading a negative latch (master stage) with a positive one (slave stage). A multiplexer-based latch is used in this particular

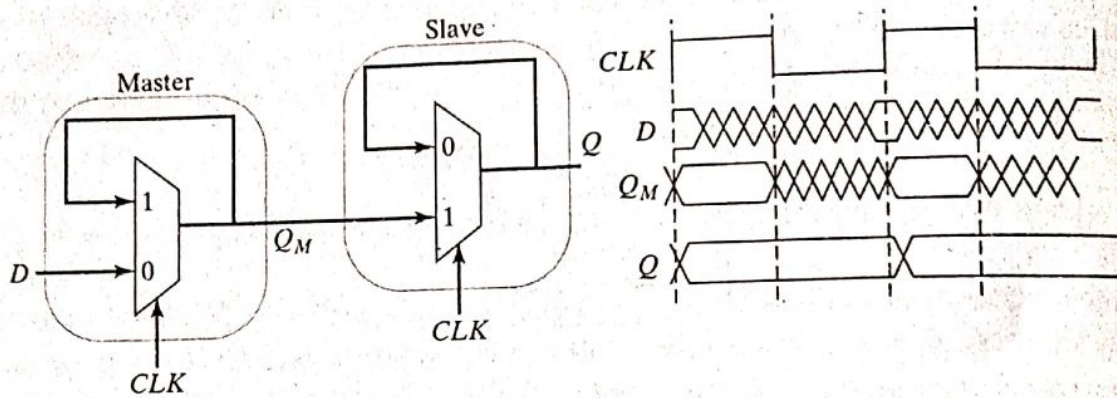


Figure 7-9 Positive edge-triggered register based on a master-slave configuration.

implementation, although any latch could be used. On the low phase of the clock, the master stage is transparent, and the D input is passed to the master stage output, Q_M . During this period, the slave stage is in the hold mode, keeping its previous value by using feedback. On the rising edge of the clock, the master stage stops sampling the input, and the slave stage starts sampling. During the high phase of the clock, the slave stage samples the output of the master stage (Q_M), while the master stage remains in a hold mode. Since Q_M is constant during the high phase of the clock, the output Q makes only one transition per cycle. The value of Q is the value of D right before the rising edge of the clock, achieving the *positive edge-triggered* effect. A negative edge-triggered register can be constructed by using the same principle by simply switching the order of the positive and negative latches (i.e., placing the positive latch first).

A complete transistor-level implementation of the master-slave positive edge-triggered register is shown in Figure 7-10. The multiplexer is implemented by using transmission gates as discussed in the previous section. When the clock is low ($\overline{CLK} = 1$), T_1 is on and T_2 is off, and the D input is sampled onto node Q_M . During this period, T_3 and T_4 are off and on, respectively. The cross-coupled inverters (I_5, I_6) hold the state of the slave latch. When the clock goes high, the master stage stops sampling the input and goes into a hold mode. T_1 is off and T_2 is on, and the cross-coupled inverters I_2 and I_3 hold the state of Q_M . Also, T_3 is on and T_4 is off, and Q_M is copied to the output Q .

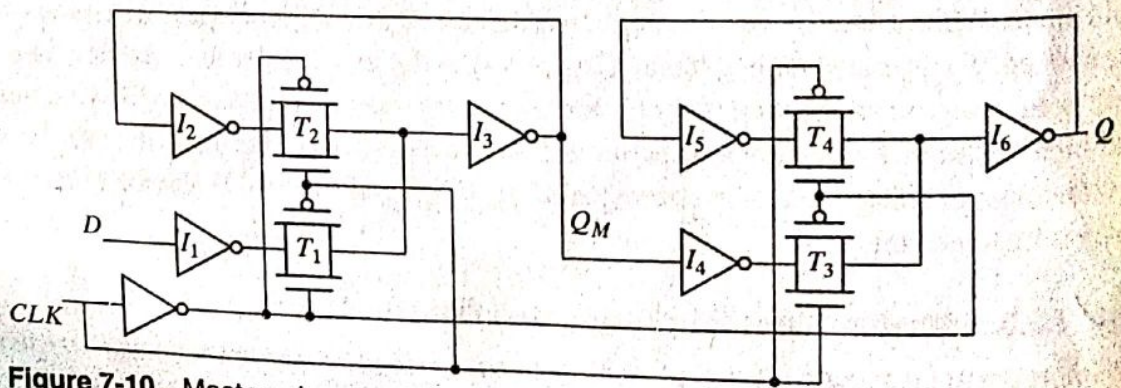


Figure 7-10 Master-slave positive edge-triggered register, using multiplexers.

Problem 7.1 Optimization of the Master–Slave Register

It is possible to remove the inverters I_1 and I_4 from Figure 7-10 without loss of functionality. Is there any advantage to including these inverters in the implementation?

Timing Properties of Multiplexer-Based Master–Slave Registers

Registers are characterized by three important timing parameters: the setup time, the hold time and the propagation delay. It is important to understand the factors that affect these timing parameters and develop the intuition to manually estimate them. Assume that the propagation delay of each inverter is t_{pd_inv} , and the propagation delay of the transmission gate is t_{pd_tx} . Also assume that the contamination delay is 0, and that inverter, deriving \overline{CLK} from CLK , has a delay of 0 as well.

The setup time is the time before the rising edge of the clock that the input data D must be valid. This is similar to asking the question, how long before the rising edge of the clock must the D input be stable such that Q_M samples the value reliably? For the transmission gate multiplexer-based register, the input D has to propagate through I_1 , T_1 , I_3 , and I_2 before the rising edge of the clock. This ensures that the node voltages on both terminals of the transmission gate T_2 are at the same value. Otherwise, it is possible for the cross-coupled pair I_2 and I_3 to settle to an incorrect value. The setup time is therefore equal to $3 \times t_{pd_inv} + t_{pd_tx}$.

The propagation delay is the time it takes for the value of Q_M to propagate to the output Q . Note that, since we included the delay of I_2 in the setup time, the output of I_4 is valid before the rising edge of the clock. Therefore, the delay t_{c-q} is simply the delay through T_3 and I_6 ($t_{c-q} = t_{pd_tx} + t_{pd_inv}$).

The *hold time* represents the time that the input must be held stable after the rising edge of the clock. In this case, the transmission gate T_1 turns off when the clock goes high. Since both the D input and the CLK pass through inverters before reaching T_1 , any changes in the input after the clock goes high do not affect the output. Therefore, the hold time is 0.

Example 7.1 Timing Analysis, Using SPICE

To obtain the setup time of the register while using SPICE, we progressively skew the input with respect to the clock edge until the circuit fails. Figure 7-11 shows the setup-time simulation assuming a skew of 210 ps and 200 ps. For the 210 ps case, the correct value of input D is sampled (in this case, the Q output remains at the value of V_{DD}). For a skew of 200 ps, an incorrect value propagates to the output, as the Q output transitions to 0. Node Q_M starts to go high, and the output of I_2 (the input to transmission gate T_2) starts to fall. However, the clock is enabled before the two nodes across the transmission gate T_2 settle to the same value. This results in an incorrect value being written into the master latch. The setup time for this register is 210 ps.

In a similar fashion, the hold time can be simulated. The D -input edge is once again skewed relative to the clock signal until the circuit stops functioning. For this design, the

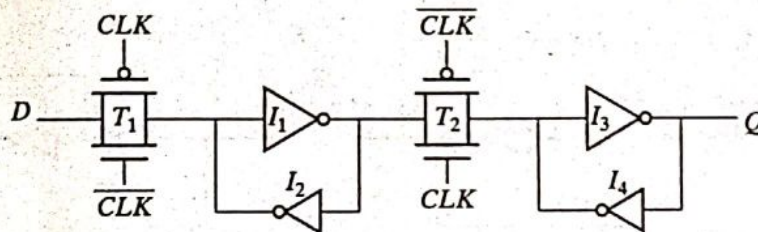


Figure 7-13 Reduced load clock load static master-slave register.

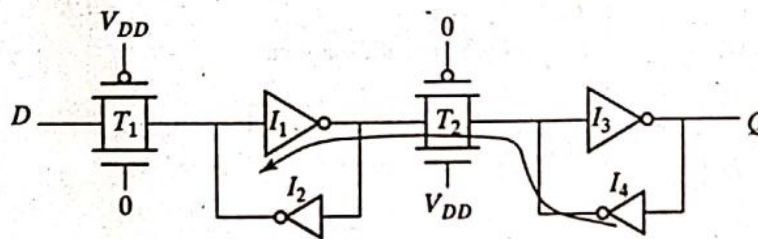


Figure 7-14 Reverse conduction possible in the transmission gate.

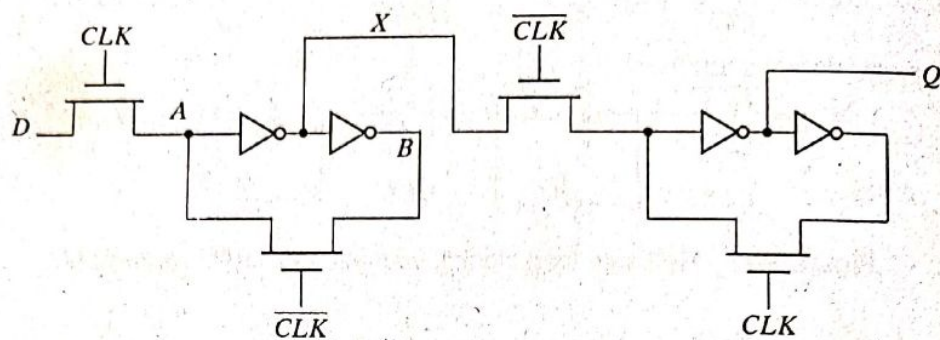
ratioed. Figure 7-13 shows that the feedback transmission gate can be eliminated by directly cross-coupling the inverters.

The penalty paid for the reduced in clock load is an increased design complexity. The transmission gate (T_1) and its source driver must overpower the feedback inverter (I_2) to switch the state of the cross-coupled inverter. The sizing requirements for the transmission gates can be derived by using an analysis similar to the one used for the sizing of the level-restoring device in Chapter 6. The input to the inverter I_1 must be brought below its switching threshold in order to make a transition. If minimum-sized devices are to be used in the transmission gates, it is essential that the transistors of inverter I_2 should be made even weaker. This can be accomplished by making their channel lengths larger than minimum. Using minimum or close-to-minimum size devices in the transmission gates is desirable to reduce the power dissipation in the latches and the clock distribution network.

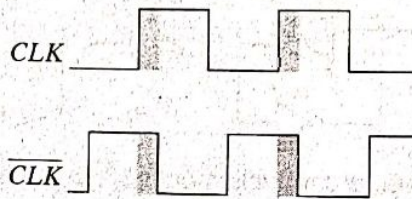
Another problem with this scheme is *reverse conduction*—the second stage can affect the state of the first latch. When the slave stage is on (Figure 7-14), it is possible for the combination of T_2 and I_4 to influence the data stored in the I_1 - I_2 latch. As long as I_4 is a weak device, this fortunately not a major problem.

Non-Ideal Clock Signals

So far, we have assumed that \overline{CLK} is a perfect inversion of CLK , or in other words, that the delay of the generating inverter is zero. Even if this were possible, this still would not be a good assumption. Variations can exist in the wires used to route the two clock signals, or the load capacitances can vary based on data stored in the connecting latches. This effect, known as *clock skew*, is a major problem, causing the two clock signals to overlap, as shown in Figure 7-15b. *Clock overlap* can cause two types of failures, which we illustrate for the NMOS-only negative master-slave register of Figure 7-15a.



(a) Schematic diagram

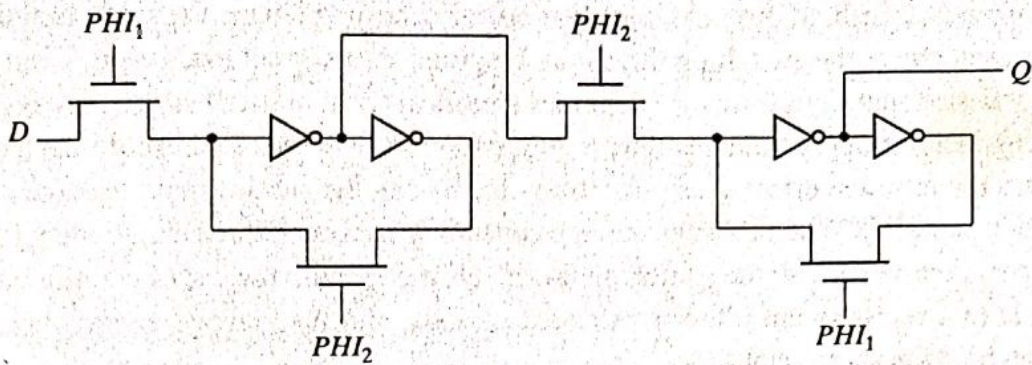


(b) Overlapping clock pairs

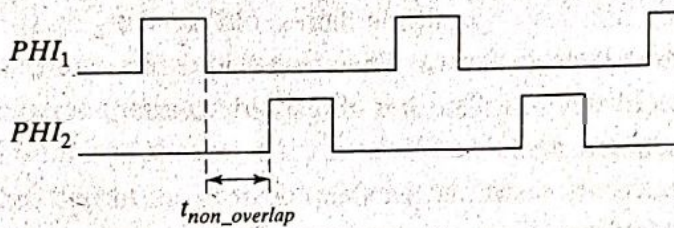
Figure 7-15 Master-slave register based on NMOS-only pass transistors.

1. When the clock goes high, the slave stage should stop sampling the master stage output and go into a hold mode. However, since CLK and \overline{CLK} are both high for a short period of time (the *overlap period*), both sampling pass transistors conduct, and there is a direct path from the D input to the Q output. As a result, data at the output can change on the rising edge of the clock, which is undesired for a negative edge-triggered register. This is known as a *race condition* in which the value of the output Q is a function of whether the input D arrives at node X before or after the falling edge of \overline{CLK} . If node X is sampled in the metastable state, the output will switch to a value determined by noise in the system.
2. The primary advantage of the multiplexer-based register is that the feedback loop is open during the sampling period, and therefore the sizing of the devices is not critical to functionality. However, if there is clock overlap between CLK and \overline{CLK} , node A can be driven by both D and B , resulting in an undefined state.

These problems can be avoided by using two *nonoverlapping clocks* instead, PHI_1 and PHI_2 (Figure 7-16), and by keeping the nonoverlap time $t_{non_overlap}$ between the clocks large enough so that no overlap occurs even in the presence of clock-routing delays. During the nonoverlap time, the FF is in the high-impedance state—the feedback loop is open, the loop gain is zero, and the input is disconnected. Leakage will destroy the state if this condition holds for too long—hence the name *pseudostatic*: The register employs a combination of static and dynamic storage approaches, depending upon the state of the clock.



(a) Schematic diagram



(b) Two-phase nonoverlapping clocks

Figure 7-16 Pseudostatic two-phase D register.

Problem 7.2 Generating Nonoverlapping Clocks

Figure 7-17 shows one possible implementation of the clock generation circuitry for generating a two-phase nonoverlapping clock. Assuming that each gate has a unit gate delay, derive the timing relationship between the input clock and the two output clocks. What is the nonoverlap period? How can this period be increased if needed?

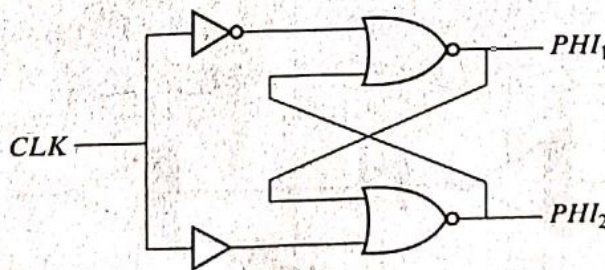


Figure 7-17 Circuitry for generating a two-phase nonoverlapping clock.

✓ **7.2.4 Low-Voltage Static Latches**

The scaling of supply voltages is critical for low-power operation. Unfortunately, certain latch structures do not function at reduced supply voltages. For example, without the scaling of device thresholds, NMOS-only pass transistors (e.g., Figure 7-16) don't scale well with supply voltage

due to its inherent threshold drop. At very low power supply voltages, the input to the inverter cannot be raised above the switching threshold, resulting in incorrect evaluation. Even with the use of transmission gates, performance degrades significantly at reduced supply voltages.

Scaling to low supply voltages thus requires the use of reduced threshold devices. However, this has the negative effect of exponentially increasing the subthreshold leakage power (as discussed in Chapter 6). When the registers are constantly accessed, the leakage energy typically is insignificant compared with the switching power. However, with the use of conditional clocks, it is possible that registers are idle for extended periods, and the leakage energy expended by registers can be quite significant.

Many solutions are being explored to address the problem of high leakage during idle periods. One approach involves the use of Multiple Threshold devices, as shown in Figure 7-18 [Mutoh95]. Only the negative latch is shown. The shaded inverters and transmission gates are implemented in low-threshold devices. The low-threshold inverters are gated by using high-threshold devices to eliminate leakage.

During the normal mode of operation, the sleep devices are turned on. When the clock is low, the D input is sampled and propagates to the output. The latch is in the hold mode when the clock is high. The feedback transmission gate conducts and the cross-coupled feedback is enabled. An extra inverter, in parallel with the low-threshold one, is added to store the state when the latch is in *idle* (or *sleep*) mode. Then, the high-threshold devices in series with the low-threshold inverter are turned off (the $SLEEP$ signal is high), eliminating leakage. It is assumed that clock

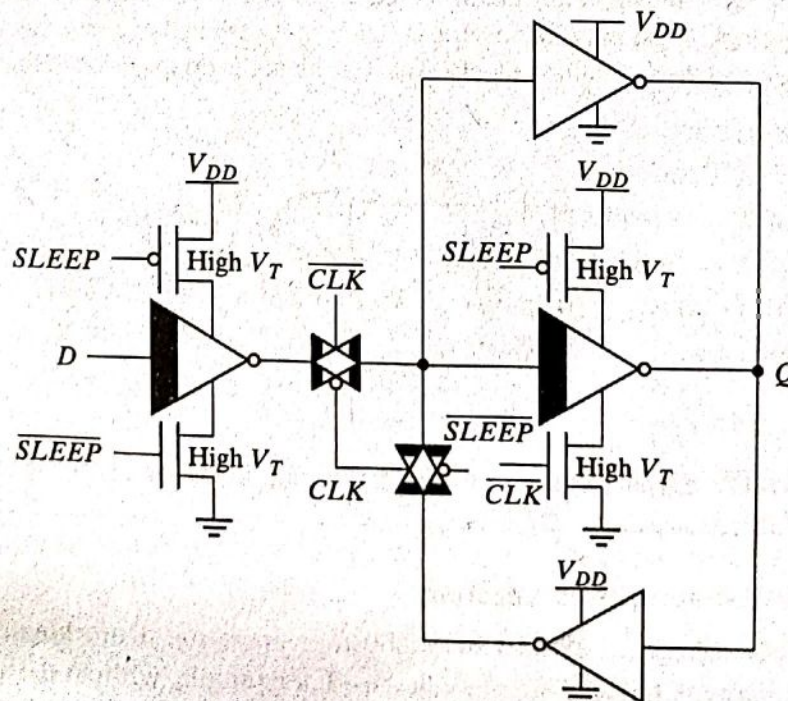


Figure 7-18 Solving the leakage problem, using multiple-threshold CMOS.

such as channel length modulation and DIBL. Figure 7-22b plots the transient response for different device sizes and confirms that an individual W/L ratio of greater than 3 is required to overpower the feedback and switch the state of the latch.

✓ 7.3 Dynamic Latches and Registers

Storage in a static sequential circuit relies on the concept that a cross-coupled inverter pair produces a bistable element and can thus be used to memorize binary values. This approach has the useful property that a stored value remains valid as long as the supply voltage is applied to the circuit—hence the name *static*. The major disadvantage of the static gate, however, is its complexity. When registers are used in computational structures that are constantly clocked (such as a pipelined datapath), the requirement that the memory should hold state for extended periods of time can be significantly relaxed.

This results in a class of circuits based on temporary storage of charge on parasitic capacitors. The principle is exactly identical to the one used in dynamic logic—charge stored on a capacitor can be used to represent a logic signal. The absence of charge denotes a 0, while its presence stands for a stored 1. No capacitor is ideal, unfortunately, and some charge leakage is always present. A stored value can thus only be kept for a limited amount of time, typically in the range of milliseconds. If one wants to preserve signal integrity, a periodic *refresh* of the value is necessary; hence, the name *dynamic* storage. Reading the value of the stored signal from a capacitor without disrupting the charge requires the availability of a device with a high-input impedance.

7.3.1 Dynamic Transmission-Gate Edge-Triggered Registers

A fully dynamic positive edge-triggered register based on the master–slave concept is shown in Figure 7-23. When $CLK = 0$, the input data is sampled on storage node 1, which has an equivalent capacitance of C_1 , consisting of the gate capacitance of I_1 , the junction capacitance of T_1 , and the overlap gate capacitance of T_1 . During this period, the slave stage is in a hold mode, with node 2 in a high-impedance (floating) state. On the rising edge of clock, the transmission gate T_2 turns on, and the value sampled on node 1 right before the rising edge propagates to the output Q (note that node 1 is stable during the high phase of the clock, since the first transmission gate is

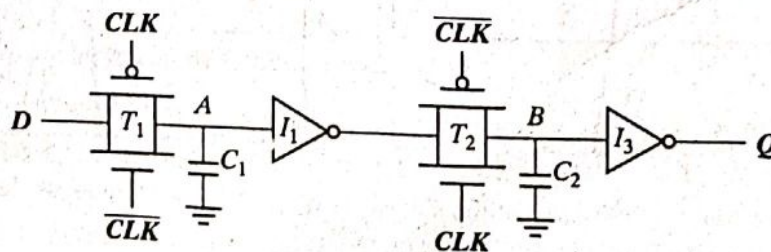


Figure 7-23 Dynamic edge-triggered register.

WARNING: The dynamic circuits shown in this section are very appealing from the perspective of complexity, performance, and power. Unfortunately, robustness considerations limit their use. In a fully dynamic circuit like that shown in Figure 7-23, a signal net that is capacitively coupled to the internal storage node can inject significant noise and destroy the state. This is especially important in ASIC flows, where there is little control over coupling between signal nets and internal dynamic nodes. Leakage currents cause another problem: Most modern processors require that the clock can be slowed down or completely halted, to conserve power in low-activity periods. Finally, the internal dynamic nodes do not track variations in power supply voltage. For example, when CLK is high for the circuit in Figure 7-23, node A holds its state, but it does not track variations in the power supply seen by I_1 . This results in reduced noise margins.

Most of these problems can be adequately addressed by adding a weak feedback inverter and making the circuit *pseudostatic* (Figure 7-25). While this comes at a slight cost in delay, it improves the noise immunity significantly. Unless registers are used in a highly-controlled environment (for instance, a custom-designed high-performance datapath), they should be made pseudostatic or static. This holds for all latches and registers discussed in this section.

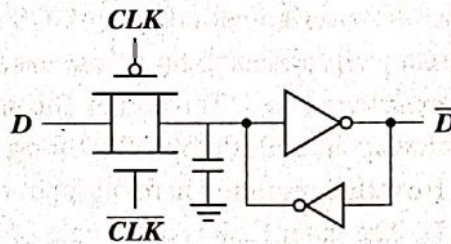


Figure 7-25 Making a dynamic latch pseudostatic.

✓ 7.3.2 C²MOS—A Clock-Skew Insensitive Approach

The C²MOS Register

Figure 7-26 shows an ingenious positive edge-triggered register that is based on a master-slave concept insensitive to clock overlap. This circuit is called the *C²MOS* (Clocked CMOS) register [Suzuki73], and operates in two phases:

1. $CLK = 0$ ($\overline{CLK} = 1$): The first tristate driver is turned on, and the master stage acts as an inverter sampling the inverted version of D on the internal node X . The master stage is in the evaluation mode. Meanwhile, the slave section is in a high-impedance mode, or in a hold mode. Both transistors M_7 and M_8 are off, decoupling the output from the input. The output Q retains its previous value stored on the output capacitor C_{L2} .
2. The roles are reversed when $CLK = 1$: The master stage section is in hold mode (M_3 – M_4 off), while the second section evaluates (M_7 – M_8 on). The value stored on C_{L1} propagates to the output node through the slave stage, which acts as an inverter.

The overall circuit operates as a positive edge-triggered master-slave register very similar to the transmission-gate-based register presented earlier. However, there is an important difference:

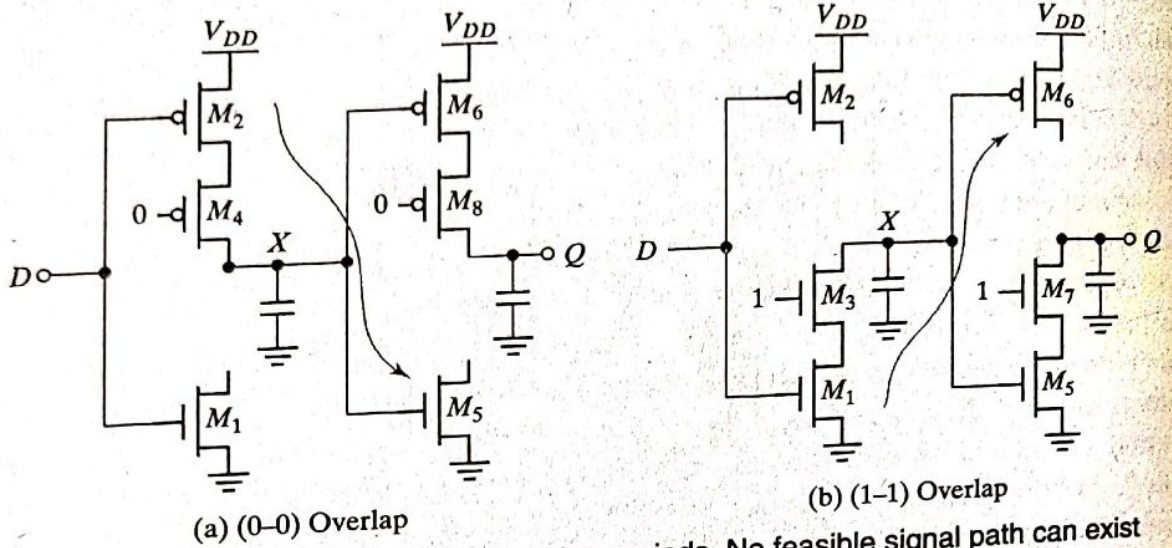


Figure 7-27 C²MOS D FF during overlap periods. No feasible signal path can exist between *In* and *D*, as illustrated by the arrows.

exists a time slot where both the NMOS and PMOS transistors are conducting. This creates a path between input and output that can destroy the state of the circuit. Simulations have shown that the circuit operates correctly as long as the clock rise time (or fall time) is smaller than approximately five times the propagation delay of the register. This criterion is not too stringent, and it is easily met in practical designs. The impact of the rise and fall times is illustrated in Figure 7-28, which plots the simulated transient response of a C²MOS D FF for clock slopes of, respectively, 0.1 and 3 ns. For slow clocks, the potential for a *race condition* exists.

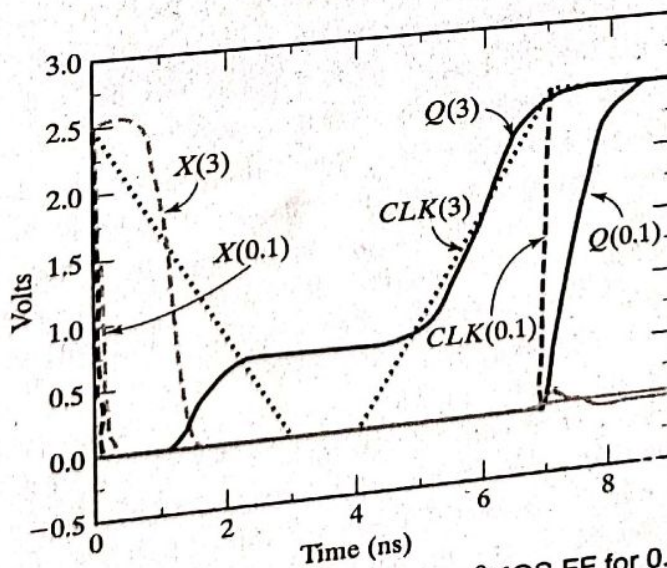


Figure 7-28 Transient response of C²MOS FF for 0.1-ns and 3-ns clock rise/fall times, assuming *ln* = 1.

Dual-Edge Registers

So far, we have focused on edge-triggered registers that sample the input data on only one of the clock edges (rising or falling). It also is possible to design sequential circuits that sample the input on both edges. The advantage of this scheme is that a lower frequency clock—half the original rate—is distributed for the same functional throughput, resulting in power savings in the clock distribution network. Figure 7-29 shows a modification of the C²MOS register enabling sampling on both edges. It consists of two parallel master-slave edge-triggered registers, whose outputs are multiplexed by using tristate drivers.

When clock is high, the positive latch composed of transistors M_1 – M_4 is sampling the inverted D input on node X . Node Y is held stable, since devices M_9 and M_{10} are turned off. On the falling edge of the clock, the top slave latch M_5 – M_8 turns on, and drives the inverted value of X to the Q output. During the low phase, the bottom master latch (M_1 , M_4 , M_9 , M_{10}) is turned on, sampling the inverted D input on node Y . Note that the devices M_1 and M_4 are reused, reducing the load on the D input. On the rising edge, the bottom slave latch conducts and drives the inverted version of Y on node Q . Data thus changes on both edges. Note that the slave latches operate in a complementary fashion—that is, only one of them is turned on during each phase of the clock.

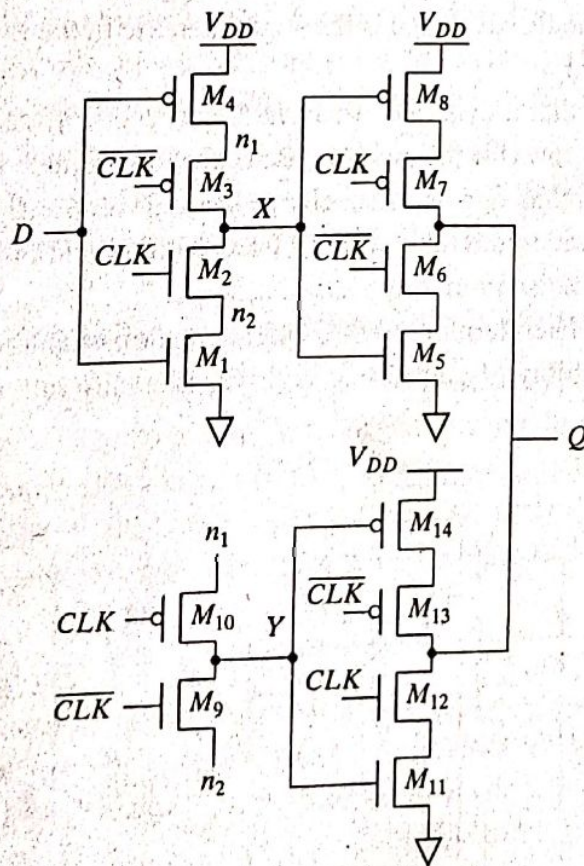


Figure 7-29 C²MOS-based dual-edge triggered register.

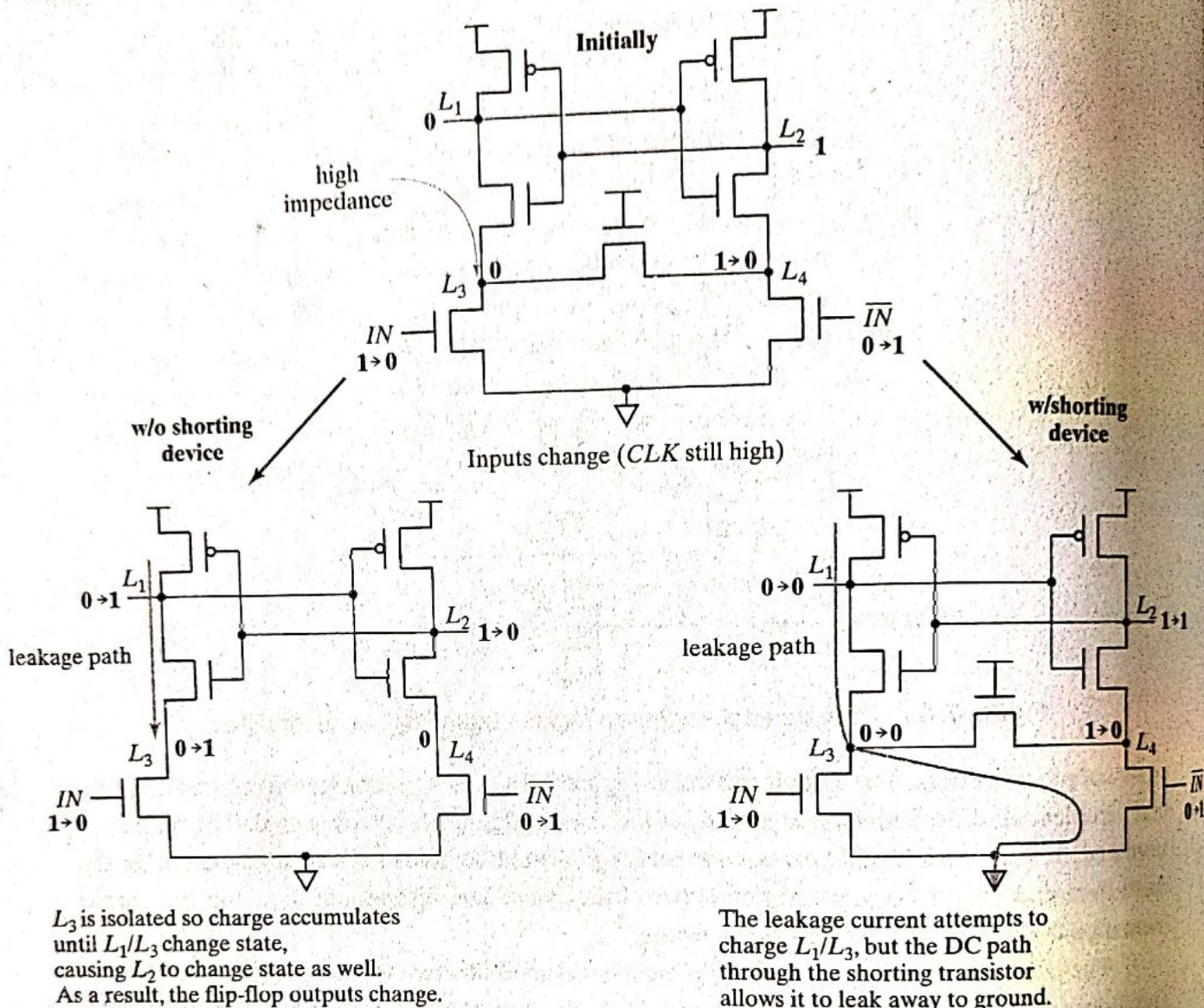


Figure 7-39 The need for the shorting transistor M_4 .

7.5 Pipelining: An Approach to Optimize Sequential Circuits

Pipelining is a popular design technique often used to accelerate the operation of datapaths in digital processors. The concept is explained with the example of Figure 7-40a. The goal of the presented circuit is to compute $\log(|a + b|)$, where both a and b represent streams of numbers (i.e., the computation must be performed on a large set of input values). The minimal clock period T_{min} necessary to ensure correct evaluation is given as

$$T_{min} = t_{c-q} + t_{pd,logic} + t_{su} \quad (7.7)$$

where t_{c-q} and t_{su} are the propagation delay and the setup time of the register, respectively. We assume that the registers are edge-triggered D registers. The term $t_{pd,logic}$ stands for the worst case delay path through the combinational network, which consists of the adder, absolute value, and logarithm functions. In conventional systems (that don't push the edge of technology), the

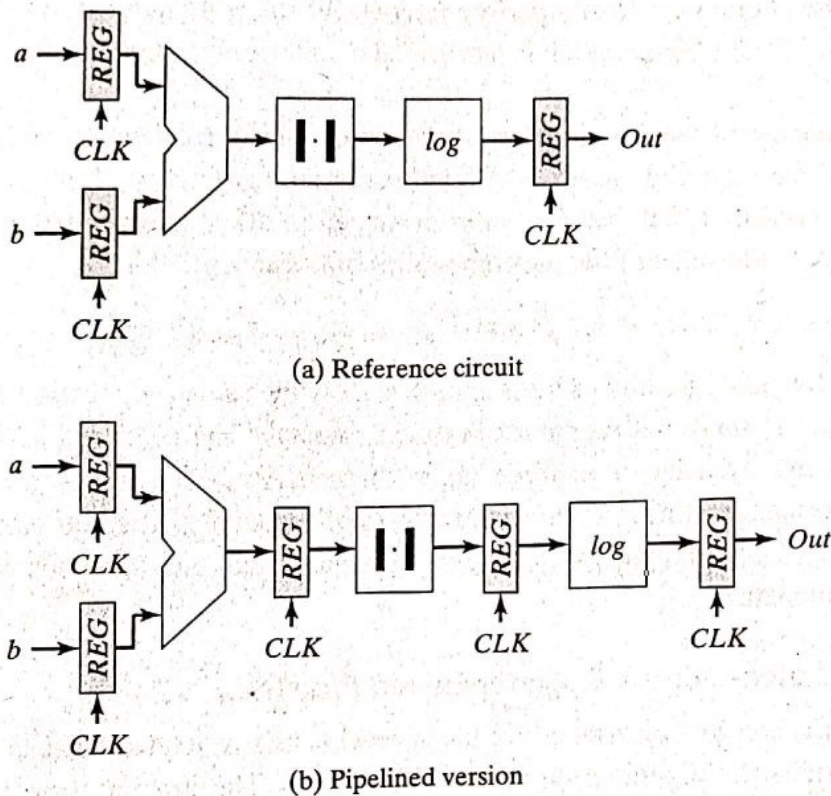


Figure 7-40 Datapath for the computation of $\log(|a + b|)$.

latter delay is generally much larger than the delays associated with the registers and dominates the circuit performance. Assume that each logic module has an equal propagation delay. We note that each logic module is then active for only one-third of the clock period (if the delay of the register is ignored). For example, the adder unit is active during the first third of the period and remains idle (no useful computation) during the other two-thirds of the period. Pipelining is a technique to improve the resource utilization, and increase the functional through-put. Assume that we introduce registers between the logic blocks, as shown in Figure 7-40b. This causes the computation for one set of input data to spread over a number of clock-periods, as shown in Table 7-1. The result for the data set (a_1, b_1) only appears at the output after three clock periods.

Table 7-1 Example of pipelined computations.

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

Chapter 2

FPGA Architectures: An Overview

Field Programmable Gate Arrays (FPGAs) were first introduced almost two and a half decades ago. Since then they have seen a rapid growth and have become a popular implementation media for digital circuits. The advancement in process technology has greatly enhanced the logic capacity of FPGAs and has in turn made them a viable implementation alternative for larger and complex designs. Further, programmable nature of their logic and routing resources has a dramatic effect on the quality of final device's area, speed, and power consumption.

This chapter covers different aspects related to FPGAs. First of all an overview of the basic FPGA architecture is presented. An FPGA comprises of an array of programmable logic blocks that are connected to each other through programmable interconnect network. Programmability in FPGAs is achieved through an underlying programming technology. This chapter first briefly discusses different programming technologies. Details of basic FPGA logic blocks and different routing architectures are then described. After that, an overview of the different steps involved in FPGA design flow is given. Design flow of FPGA starts with the hardware description of the circuit which is later synthesized, technology mapped and packed using different tools. After that, the circuit is placed and routed on the architecture to complete the design flow.

The programmable logic and routing interconnect of FPGAs makes them flexible and general purpose but at the same time it makes them larger, slower and more power consuming than standard cell ASICs. However, the advancement in process technology has enabled and necessitated a number of developments in the basic FPGA architecture. These developments are aimed at further improvement in the overall efficiency of FPGAs so that the gap between FPGAs and ASICs might be reduced. These developments and some future trends are presented in the last section of this chapter.

2.1 Introduction to FPGAs

Field programmable Gate Arrays (FPGAs) are pre-fabricated silicon devices that can be electrically programmed in the field to become almost any kind of digital circuit or system. For low to medium volume productions, FPGAs provide cheaper solution and faster time to market as compared to Application Specific Integrated Circuits (ASIC) which normally require a lot of resources in terms of time and money to obtain first device. FPGAs on the other hand take less than a minute to configure and they cost anywhere around a few hundred dollars to a few thousand dollars. Also for varying requirements, a portion of FPGA can be partially reconfigured while the rest of an FPGA is still running. Any future updates in the final product can be easily upgraded by simply downloading a new application bitstream. However, the main advantage of FPGAs i.e. flexibility is also the major cause of its draw back. Flexible nature of FPGAs makes them significantly larger, slower, and more power consuming than their ASIC counterparts. These disadvantages arise largely because of the programmable routing interconnect of FPGAs which comprises of almost 90% of total area of FPGAs. But despite these disadvantages, FPGAs present a compelling alternative for digital system implementation due to their less time to market and low volume cost.

Normally FPGAs comprise of:

- Programmable logic blocks which implement logic functions.
- Programmable routing that connects these logic functions.
- I/O blocks that are connected to logic blocks through routing interconnect and that make off-chip connections.

A generalized example of an FPGA is shown in Fig. 2.1 where configurable logic blocks (CLBs) are arranged in a two dimensional grid and are interconnected by programmable routing resources. I/O blocks are arranged at the periphery of the grid and they are also connected to the programmable routing interconnect. The “programmable/reconfigurable” term in FPGAs indicates their ability to implement a new function on the chip after its fabrication is complete. The reconfigurability/programmability of an FPGA is based on an underlying programming technology, which can cause a change in behavior of a pre-fabricated chip after its fabrication.

2.2 Programming Technologies

There are a number of programming technologies that have been used for reconfigurable architectures. Each of these technologies have different characteristics which in turn have significant effect on the programmable architecture. Some of the well known technologies include static memory [122], flash [54], and anti-fuse [61].

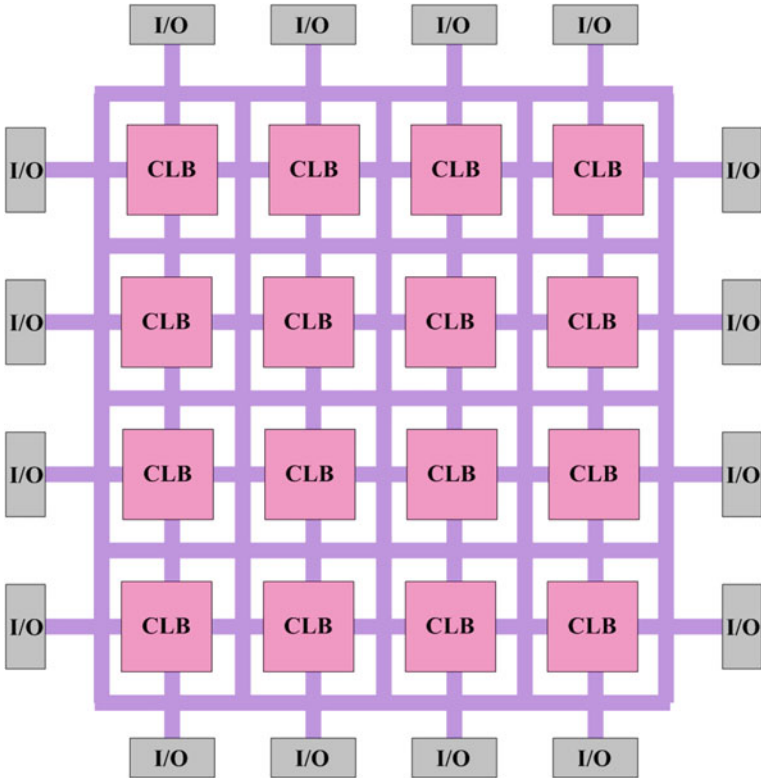


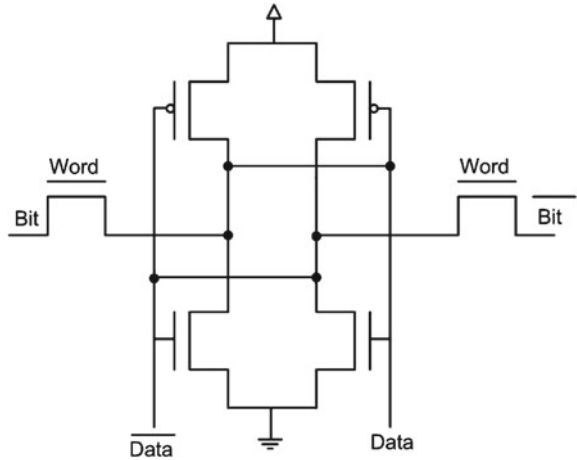
Fig. 2.1 Overview of FPGA architecture [22]

2.2.1 SRAM-Based Programming Technology

Static memory cells are the basic cells used for SRAM-based FPGAs. Most commercial vendors [76, 126] use static memory (SRAM) based programming technology in their devices. These devices use static memory cells which are divided throughout the FPGA to provide configurability. An example of such memory cell is shown in Fig. 2.2. In an SRAM-based FPGA, SRAM cells are mainly used for following purposes:

1. To program the routing interconnect of FPGAs which are generally steered by small multiplexors.
2. To program Configurable Logic Blocks (CLBs) that are used to implement logic functions.

SRAM-based programming technology has become the dominant approach for FPGAs because of its re-programmability and the use of standard CMOS process technology and therefore leading to increased integration, higher speed and lower

Fig. 2.2 Static memory cell

dynamic power consumption of new process with smaller geometry. There are however a number of drawbacks associated with SRAM-based programming technology. For example an SRAM cell requires 6 transistors which makes the use of this technology costly in terms of area compared to other programming technologies. Further SRAM cells are volatile in nature and external devices are required to permanently store the configuration data. These external devices add to the cost and area overhead of SRAM-based FPGAs.

2.2.2 Flash Programming Technology

One alternative to the SRAM-based programming technology is the use of flash or EEPROM based programming technology. Flash-based programming technology offers several advantages. For example, this programming technology is non-volatile in nature. Flash-based programming technology is also more area efficient than SRAM-based programming technology. Flash-based programming technology has its own disadvantages also. Unlike SRAM-based programming technology, flash-based devices can not be reconfigured/reprogrammed an infinite number of times. Also, flash-based technology uses non-standard CMOS process.

2.2.3 Anti-fuse Programming Technology

An alternative to SRAM and flash-based technologies is anti-fuse programming technology. The primary advantage of anti-fuse programming technology is its low area. Also this technology has lower on resistance and parasitic capacitance than other two

programming technologies. Further, this technology is non-volatile in nature. There are however significant disadvantages associated with this programming technology. For example, this technology does not make use of standard CMOS process. Also, anti-fuse programming technology based devices can not be reprogrammed.

In this section, an overview of three commonly used programming technologies is given where all of them have their advantages and disadvantages. Ideally, one would like to have a programming technology which is reprogrammable, non-volatile, and that uses a standard CMOS process. Apparently, none of the above presented technologies satisfy these conditions. However, SRAM-based programming technology is the most widely used programming technology. The main reason is its use of standard CMOS process and for this very reason, it is expected that this technology will continue to dominate the other two programming technologies.

2.3 Configurable Logic Block

A configurable logic block (CLB) is a basic component of an FPGA that provides the basic logic and storage functionality for a target application design. In order to provide the basic logic and storage capability, the basic component can be either a transistor or an entire processor. However, these are the two extremes where at one end the basic component is very fine-grained (in case of transistors) and requires large amount of programmable interconnect which eventually results in an FPGA that suffers from area-inefficiency, low performance and high power consumption. On the other end (in case of processor), the basic logic block is very coarse-grained and can not be used to implement small functions as it will lead to wastage of resources. In between these two extremes, there exists a spectrum of basic logic blocks. Some of them include logic blocks that are made of NAND gates [101], an interconnection of multiplexors [44], lookup table (LUT) [121] and PAL style wide input gates [124]. Commercial vendors like Xilinx and Altera use LUT-based CLBs to provide basic logic and storage functionality. LUT-based CLBs provide a good trade-off between too fine-grained and too coarse-grained logic blocks. A CLB can comprise of a single basic logic element (BLE), or a cluster of locally interconnected BLEs (as shown in Fig. 2.4). A simple BLE consists of a LUT, and a Flip-Flop. A LUT with k inputs (LUT- k) contains 2^k configuration bits and it can implement any k -input boolean function. Figure 2.3 shows a simple BLE comprising of a 4 input LUT (LUT-4) and a D-type Flip-Flop. The LUT-4 uses 16 SRAM bits to implement any 4 inputs boolean function. The output of LUT-4 is connected to an optional Flip-Flop. A multiplexor selects the BLE output to be either the output of a Flip-Flop or the LUT-4.

A CLB can contain a cluster of BLEs connected through a local routing network. Figure 2.4 shows a cluster of 4 BLEs; each BLE contains a LUT-4 and a Flip-Flop. The BLE output is accessible to other BLEs of the same cluster through a local routing network. The number of output pins of a cluster are equal to the total number of BLEs in a cluster (with each BLE having a single output). However, the number of input pins of a cluster can be less than or equal to the sum of input pins required

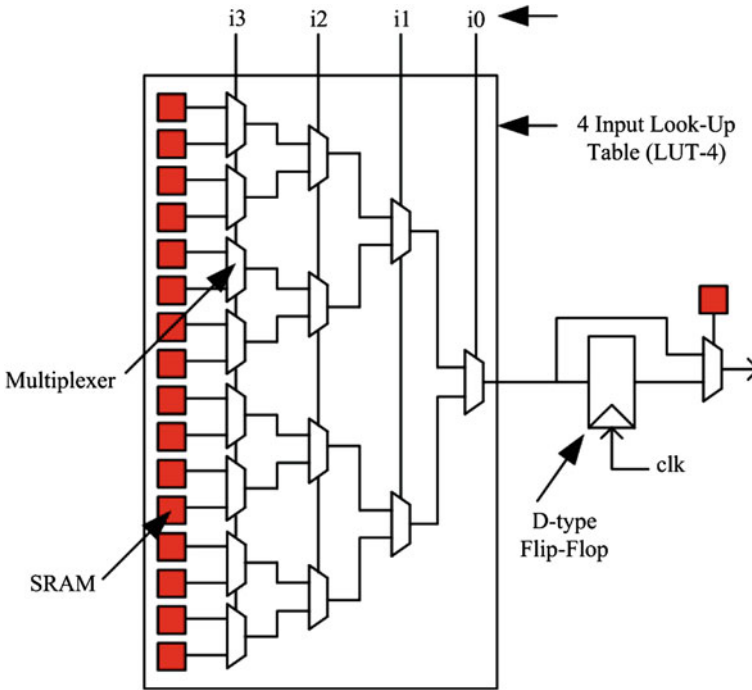


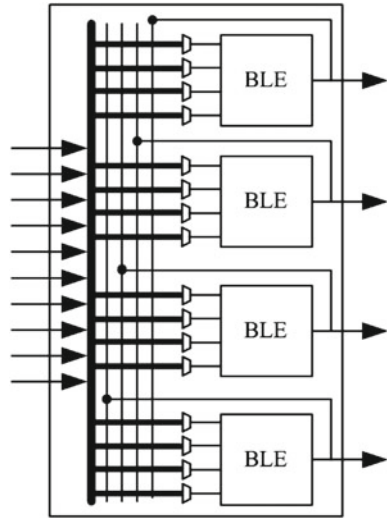
Fig. 2.3 Basic logic element (BLE) [22]

by all the BLEs in the cluster. Modern FPGAs contain typically 4 to 10 BLEs in a single cluster. Although here we have discussed only basic logic blocks, many modern FPGAs contain a heterogeneous mixture of blocks, some of which can only be used for specific purposes. These specific purpose blocks, also referred here as hard blocks, include memory, multipliers, adders and DSP blocks etc. Hard blocks are very efficient at implementing specific functions as they are designed optimally to perform these functions, yet they end up wasting huge amount of logic and routing resources if unused. A detailed discussion on the use of heterogeneous mixture of blocks for implementing digital circuits is presented in Chap. 4 where both advantages and disadvantages of heterogeneous FPGA architectures and a remedy to counter the resource loss problem are discussed in detail.

2.4 FPGA Routing Architectures

As discussed earlier, in an FPGA, the computing functionality is provided by its programmable logic blocks and these blocks connect to each other through programmable routing network. This programmable routing network provides routing

Fig. 2.4 A configurable logic block (CLB) having four BLEs [22]



connections among logic blocks and I/O blocks to implement any user-defined circuit. The routing interconnect of an FPGA consists of wires and programmable switches that form the required connection. These programmable switches are configured using the programmable technology.

Since FPGA architectures claim to be potential candidate for the implementation of any digital circuit, their routing interconnect must be very flexible so that they can accommodate a wide variety of circuits with widely varying routing demands. Although the routing requirements vary from circuit to circuit, certain common characteristics of these circuits can be used to optimally design the routing interconnect of FPGA architecture. For example most of the designs exhibit locality, hence requiring abundant short wires. But at the same time there are some distant connections, which leads to the need for sparse long wires. So, care needs to be taken into account while designing routing interconnect for FPGA architectures where we have to address both flexibility and efficiency. The arrangement of routing resources, relative to the arrangement of logic blocks of the architecture, plays a very important role in the overall efficiency of the architecture. This arrangement is termed here as global routing architecture whereas the microscopic details regarding the switching topology of different switch blocks is termed as detailed routing architecture. On the basis of the global arrangement of routing resources of the architecture, FPGA architectures can be categorized as either hierarchical [4] or island-style [22]. In this section, we present a detailed overview of both routing architectures.

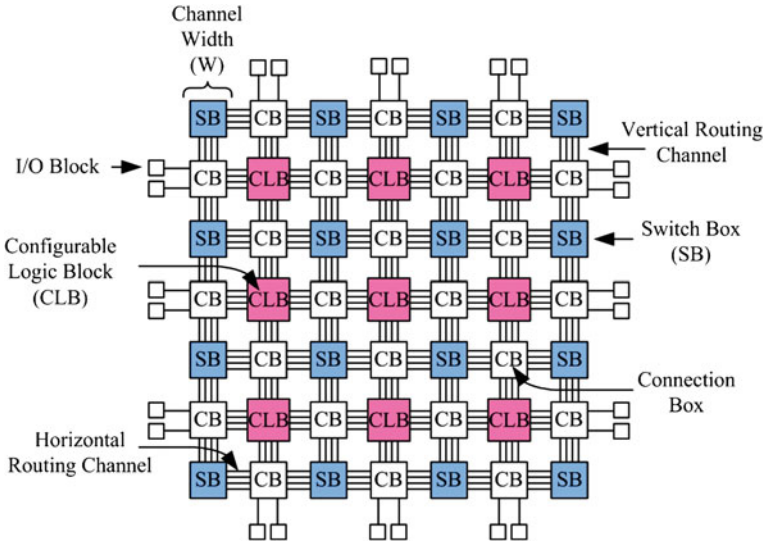


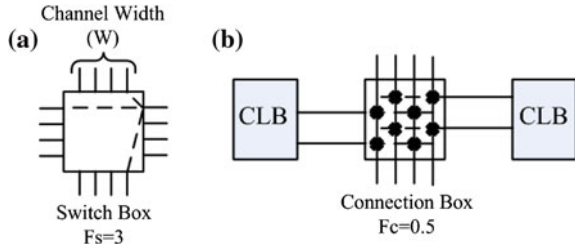
Fig. 2.5 Overview of mesh-based FPGA architecture [22]

2.4.1 Island-Style Routing Architecture

Figure 2.5 shows a traditional island-style FPGA architecture (also termed as mesh-based FPGA architecture). This is the most commonly used architecture among academic and commercial FPGAs. It is called island-style architecture because in this architecture configurable logic blocks look like islands in a sea of routing interconnect. In this architecture, configurable logic blocks (CLBs) are arranged on a 2D grid and are interconnected by a programmable routing network. The Input/Output (I/O) blocks on the periphery of FPGA chip are also connected to the programmable routing network. The routing network comprises of pre-fabricated wiring segments and programmable switches that are organized in horizontal and vertical routing channels.

The routing network of an FPGA occupies 80–90% of total area, whereas the logic area occupies only 10–20% area [22]. The flexibility of an FPGA is mainly dependent on its programmable routing network. A mesh-based FPGA routing network consists of horizontal and vertical routing tracks which are interconnected through switch boxes (SB). Logic blocks are connected to the routing network through connection boxes (CB). The flexibility of a connection box (F_c) is the number of routing tracks of adjacent channel which are connected to the pin of a block. The connectivity of input pins of logic blocks with the adjacent routing channel is called as $F_c(\text{in})$; the connectivity of output pins of the logic blocks with the adjacent routing channel is called as $F_c(\text{out})$. An $F_c(\text{in})$ equal to 1.0 means that all the tracks of adjacent routing channel are connected to the input pin of the logic block. The flexibility of switch box (F_s) is the total number of tracks with which every track entering in the switch

Fig. 2.6 Example of switch and connection box



box connects to. The number of tracks in routing channel is called the channel width of the architecture. Same channel width is used for all horizontal and vertical routing channels of the architecture. An example explaining the switch box, connection box flexibilities, and routing channel width is shown in Fig. 2.6. In this figure switch box has $F_s = 3$ as each track incident on it is connected to 3 tracks of adjacent routing channels. Similarly, connection box has $F_c(in) = 0.5$ as each input of the logic block is connected to 50% of the tracks of adjacent routing channel.

The routing tracks connected through a switch box can be bidirectional or unidirectional (also called as directional) tracks. Figure 2.7 shows a bidirectional and a unidirectional switch box having F_s equal to 3. The input tracks (or wires) in both these switch boxes connect to 3 other tracks of the same switch box. The only limitation of unidirectional switch box is that their routing channel width must be in multiples of 2.

Generally, the output pins of a block can connect to any routing track through pass transistors. Each pass transistor forms a tristate output that can be independently turned on or off. However, single-driver wiring technique can also be used to connect output pins of a block to the adjacent routing tracks. For single-driver wiring, tristate elements cannot be used; the output of block needs to be connected to the neighboring routing network through multiplexors in the switch box. Modern commercial FPGA architectures have moved towards using single-driver, directional routing tracks. Authors in [51] show that if single-driver directional wiring is used instead of bidirectional wiring, 25% improvement in area, 9% in delay and 32% in area-delay can be achieved. All these advantages are achieved without making any major changes in the FPGA CAD flow.

In mesh-based FPGAs, multi-length wires are created to reduce delay. Figure 2.8 shows an example of different length wires. Longer wire segments span multiple blocks and require fewer switches, thereby reducing routing area and delay. However, they also decrease routing flexibility, which reduces the probability to route a hardware circuit successfully. Modern commercial FPGAs commonly use a combination of long and short wires to balance flexibility, area and delay of the routing network.

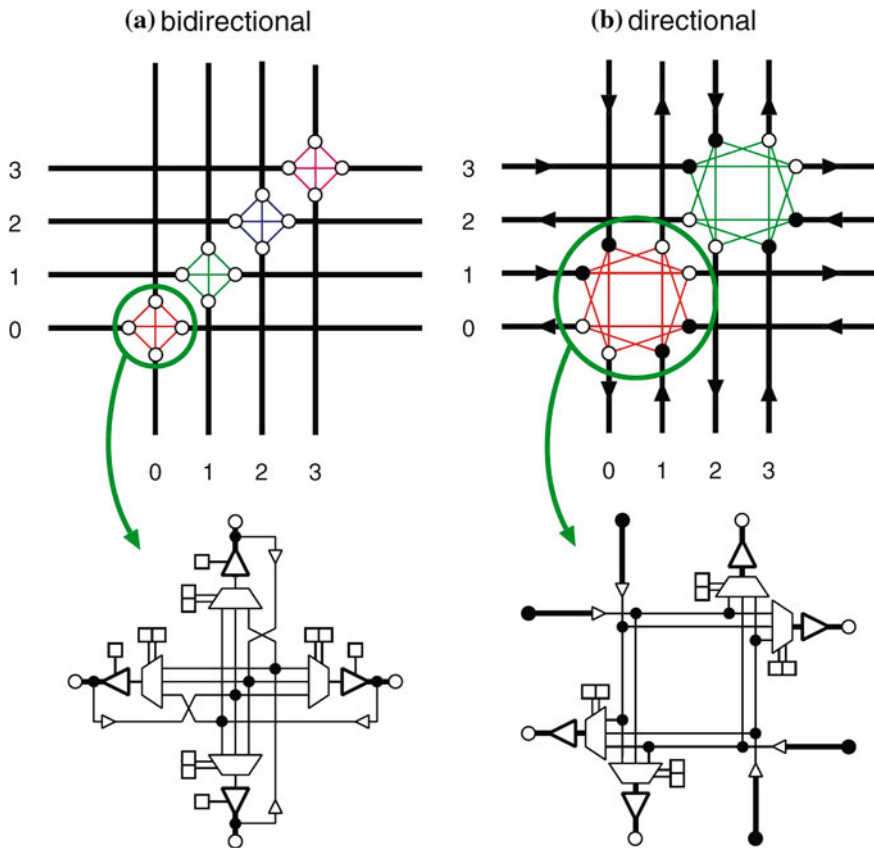


Fig. 2.7 Switch block, length 1 wires [51]

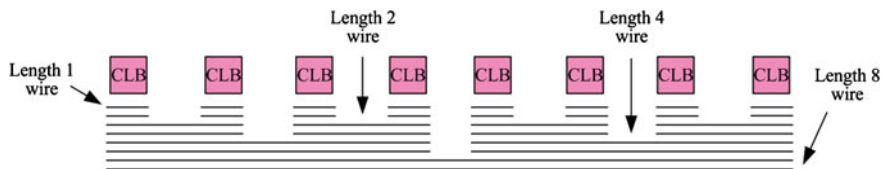


Fig. 2.8 Channel segment distribution

2.4.1.1 Altera’s Stratix II Architecture

Until now, we have presented a general overview about island-style routing architecture. Now we present a commercial example of this kind of architectures. Altera’s Stratix II [106] architecture is an industrial example of an island-style FPGA (Fig. 2.9). The logic structure consists of LABs (Logic Array Blocks), memory blocks, and digital signal processing (DSP) blocks. LABs are used to

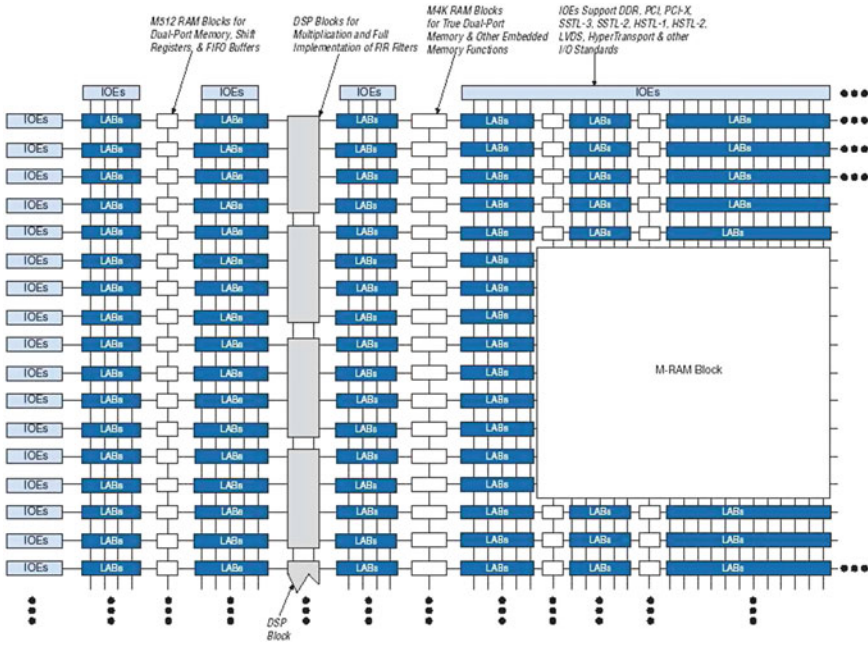


Fig. 2.9 Altera’s stratix-II block diagram

implement general-purpose logic, and are symmetrically distributed in rows and columns throughout the device fabric. The DSP blocks are custom designed to implement full-precision multipliers of different granularities, and are grouped into columns. Input- and output-only elements (IOEs) represent the external interface of the device. IOEs are located along the periphery of the device.

Each Stratix II LAB consists of eight Adaptive Logic Modules (ALMs). An ALM consists of 2 adaptive LUTs (ALUTs) with eight inputs altogether. Construction of an ALM allows implementation of 2 separate 4-input Boolean functions. Further, an ALM can also be used to implement any six-input Boolean function, and some seven-input functions. In addition to lookup tables, an ALM provides 2 programmable registers, 2 dedicated full-adders, a carry chain, and a register-chain. Full-adders and carry chain can be used to implement arithmetic operations, and the register-chain is used to build shift registers. Outputs of an ALM drive all types of interconnect provided by the Stratix II device. Figure 2.10 illustrates a LAB interconnect interface.

Interconnections between LABs, RAM blocks, DSP blocks and the IOEs are established using the Multi-track interconnect structure. This interconnect structure consists of wire segments of different lengths and speeds. The interconnect wire-segments span fixed distances, and run in the horizontal (row interconnects) and vertical (column interconnects) directions. The row interconnects (Fig. 2.11) can be used to route signals between LABs, DSP blocks, and memory blocks in the same row. Row interconnect resources are of the following types:

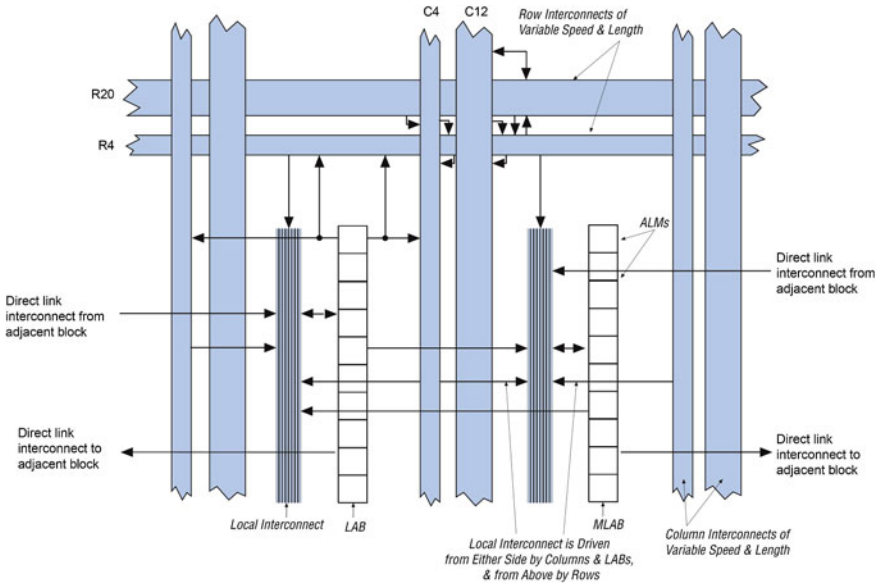


Fig. 2.10 Stratix-II logic array block (LAB) structure

- Direct connections between LABs and adjacent blocks.
- R4 resources that span 4 blocks to the left or right.
- R24 resources that provide high-speed access across 24 columns.

Each LAB owns its set of R4 interconnects. A LAB has approximately equal numbers of driven-left and driven-right R4 interconnects. An R4 interconnect that is driven to the left can be driven by either the primary LAB (Fig. 2.11) or the adjacent LAB to the left.

Similarly, a driven-right R4 interconnect may be driven by the primary LAB or the LAB immediately to its right. Multiple R4 resources can be connected to each other to establish longer connections within the same row. R4 interconnects can also drive C4 and C16 column interconnects, and R24 high speed row resources.

Column interconnect structure is similar to row interconnect structure. Column interconnects include:

- Carry chain interconnects within a LAB, and from LAB to LAB in the same column.
- Register chain interconnects.
- C4 resources that span 4 blocks in the up and down directions.
- C16 resources for high-speed vertical routing across 16 rows.

Carry chain and register chain interconnects are separated from local interconnect (Fig. 2.10) in a LAB. Each LAB has its own set of driven-up and driven-down C4 interconnects. C4 interconnects can also be driven by the LABs that are immediately

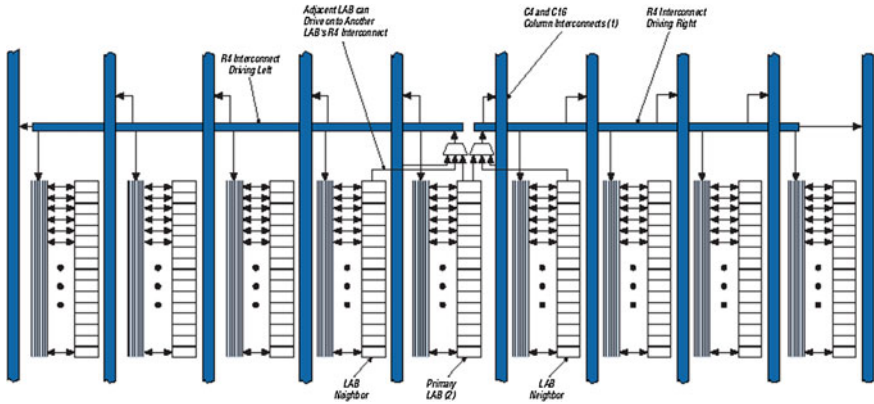


Fig. 2.11 R4 interconnect connections

adjacent to the primary LAB. Multiple C4 resources can be connected to each other to form longer connections within a column, and C4 interconnects can also drive row interconnects to establish column-to-column interconnections. C16 interconnects are high-speed vertical resources that span 16 LABs. A C16 interconnect can drive row and column interconnects at every fourth LAB. A LAB local interconnect structure cannot be directly driven by a C16 interconnect; only C4 and R4 interconnects can drive a LAB local interconnect structure. Figure 2.12 shows the C4 interconnect structure in the Stratix II device.

2.4.2 Hierarchical Routing Architecture

Most logic designs exhibit locality of connections; hence implying a hierarchy in placement and routing of connections between different logic blocks. Hierarchical routing architectures exploit this locality by dividing FPGA logic blocks into separate groups/clusters. These clusters are recursively connected to form a hierarchical structure. In a hierarchical architecture (also termed as tree-based architecture), connections between logic blocks within same cluster are made by wire segments at the lowest level of hierarchy. However, the connection between blocks residing in different groups require the traversal of one or more levels of hierarchy. In a hierarchical architecture, the signal bandwidth varies as we move away from the bottom level and generally it is widest at the top level of hierarchy. The hierarchical routing architecture has been used in a number of commercial FPGA families including Altera Flex10K [10], Apex [15] and ApexII [16] architectures. We assume that Multilevel hierarchical interconnect regroups architectures with more than 2 levels of hierarchy and Tree-based ones.

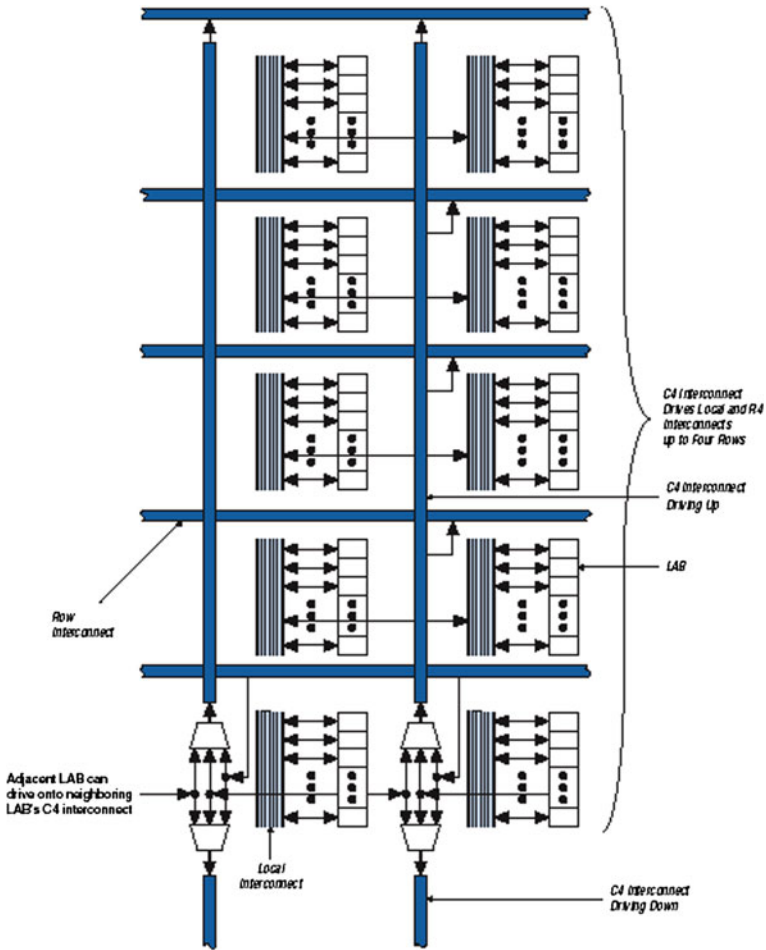


Fig. 2.12 C4 interconnect connections

2.4.2.1 HFPGA: Hierarchical FPGA

In the hierarchical FPGA called HFPGA, LBs are grouped into clusters. Clusters are then grouped recursively together (see Fig. 2.13). The clustered VPR mesh architecture [22] has a Hierarchical topology with only two levels. Here we consider multilevel hierarchical architectures with more than 2 levels. In [1] and [129] various hierarchical structures were discussed. The HFPGA routability depends on switch boxes topologies. HFPGAs comprising fully populated switch boxes ensure 100% routability but are very penalizing in terms of area. In [129] authors explored the HFPGA architecture, investigating how the switch pattern can be partly depopulated while maintaining a good routability.

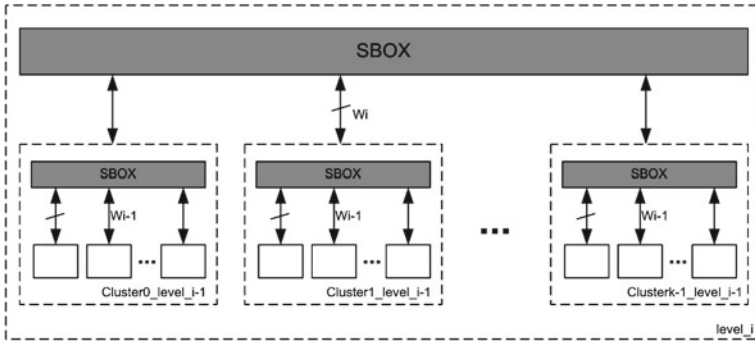


Fig. 2.13 Hierarchical FPGA topology

2.4.2.2 HSRA: Hierarchical Synchronous Reconfigurable Array

An example of an academic hierarchical routing architecture is shown in Fig. 2.14. It has a strictly hierarchical, tree-based interconnect structure. In this architecture, the only wire segments that directly connect to the logic units are located at the leaves of the interconnect tree. All other wire segments are decoupled from the logic structure. A logic block of this architecture consists of a pair of 2-input Look Up Table (2-LUT) and a D-type Flip Flop (D-FF). The input-pin connectivity is based on a choose-k strategy [4], and the output pins are fully connected. The richness of this interconnect structure is defined by its base channel width c and interconnect growth rate p . The base channel width c is defined as the number of tracks at the leaves of the interconnect Tree (in Fig. 2.14, $c = 3$). Growth rate p is defined as the rate at which the interconnect bandwidth grows towards the upper levels. The interconnect growth rate can be realized either using non-compressing or compressing switch blocks. The details regarding these switch blocks is as follows:

- Non-compressing (2:1) switch blocks—The number of tracks at the upper level are equal to the sum of the number of tracks of the children at lower level. For example, in Fig. 2.14, non-compressing switch blocks are used between levels 1, 2 and levels 3, 4.
- Compressing (1:1) switch blocks—The number of tracks at the upper level are equal to the number of tracks of either child at the lower level. For example, in Fig. 2.14, compressing switch blocks are used between levels 2 and 3.

A repeating combination of non-compressing and compressing switch blocks can be used to realize any value of p less than one. For example, a repeating pattern of (2:1, 1:1) switch blocks realizes $p = 0.5$, while the pattern (2:1, 2:1, 1:1) realizes $p = 0.67$. An architecture that has only 2:1 switch blocks provides a growth rate of $p = 1$.

Another hierarchical routing architecture is presented in [132] where the global routing architecture (i.e. the position of routing resources relative to logic resources

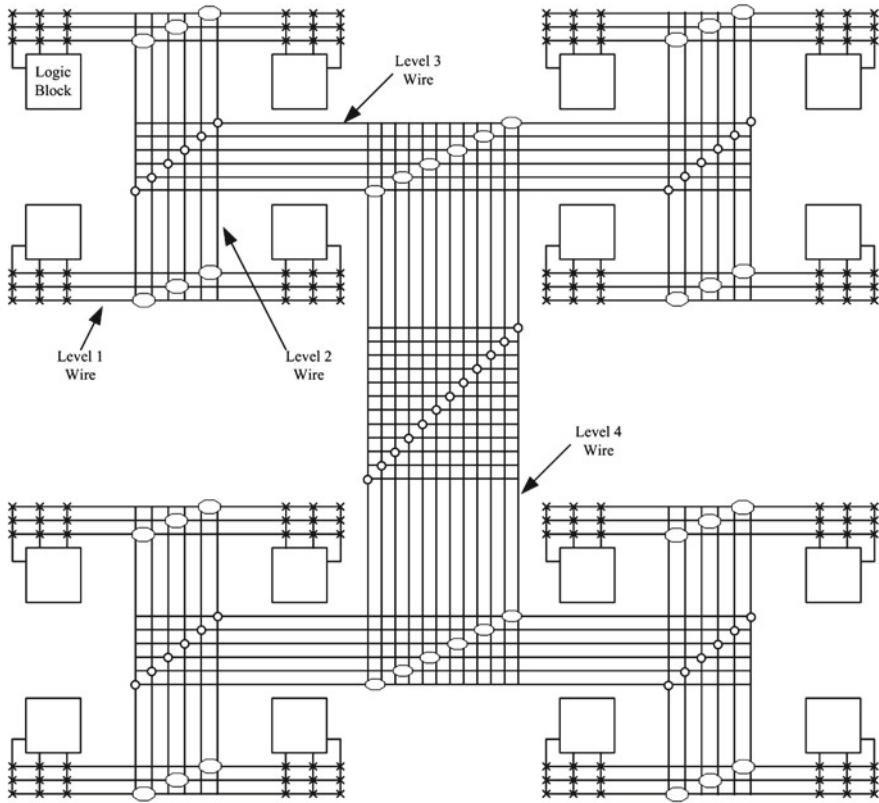
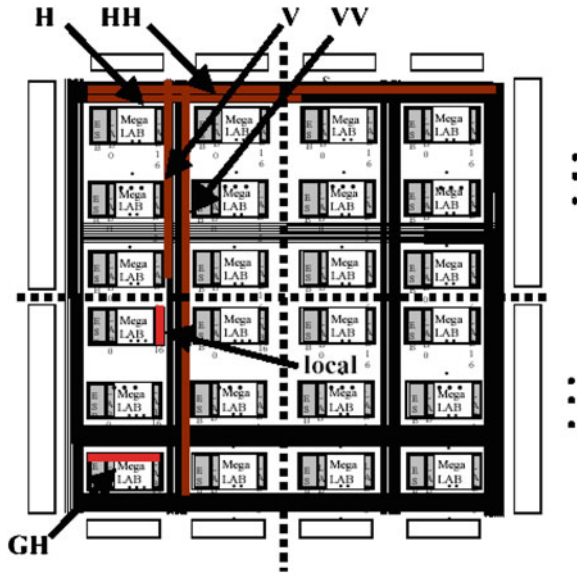


Fig. 2.14 Example of hierarchical routing architecture [4]

of the architecture) remains the same as in [4]. However, there are several key differences at the level of detailed routing architecture (i.e. the way the routing resources are connected to each other, flexibility of switch blocks etc.) that separate the two architectures. For example the architecture shown in Fig. 2.14 has one bidirectional interconnect that uses bidirectional switches and it supports only arity-2 (i.e. each cluster can contain only two sub-clusters). On contrary, the architecture presented in [132] supports two separate unidirectional interconnect networks: one is downward interconnect whereas other is upward interconnect network. Further this architecture is more flexible as it can support logic blocks with different sizes and also the clusters/groups of the routing architecture can have different arity sizes. Further details of this architecture, from now on alternatively termed as tree-based architecture, are presented in next chapter.

Fig. 2.15 The APEX programmable logic Devices [87]



2.4.2.3 APEX: Altera

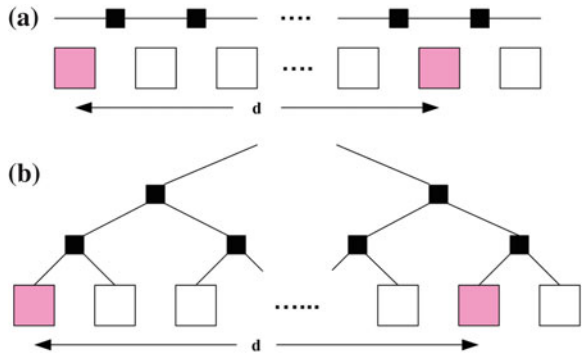
APEX architecture is a commercial product from Altera Corporation which includes 3 levels of interconnect hierarchy. Figure 2.15 shows a diagram of the APEX 20K400 programmable logic device. The basic logic-element (LE) is a 4-input LUT and DFF pair. Groups of 10 LEs are grouped into a logic-array-block or LAB. Interconnect within a LAB is complete, meaning that a connection from the output of any LE to the input of another LE in its LAB always exists, and any signal entering the input region can reach every LE.

Groups of 16 LABs form a MegaLab. Interconnect within a MegaLab requires an LE to drive a GH (MegaLab global H) line, a horizontal line, which switches into the input region of any other LAB in the same MegaLab. Adjacent LABs have the ability to interleave their input regions, so an LE in LAB_i can usually drive LAB_{i+1} without using a GH line. A 20K400 MegaLab contains 279 GH lines.

The top-level architecture is a 4 by 26 array of MegaLabs. Communication between MegaLabs is accomplished by global H (horizontal) and V (vertical) wires, that switch at their intersection points. The H and V lines are segmented by a bidirectional segmentation buffer at the horizontal and vertical centers of the chip. In Fig. 2.15, We denote the use of a single (half-chip) line as H or V and a double or full-chip line through the segmentation buffer as HH or VV. The 20K400 contains 100 H lines per MegaLab row, and 80 V lines per LAB-column.

In this section, so far we have given an overview of the two routing architectures that are commonly employed in FPGAs. Both architectures have their positive and negative points. For example, hierarchical routing architectures exploit the

Fig. 2.16 **a** Number of series switches in a mesh structure
b Number of series switches in a tree structure



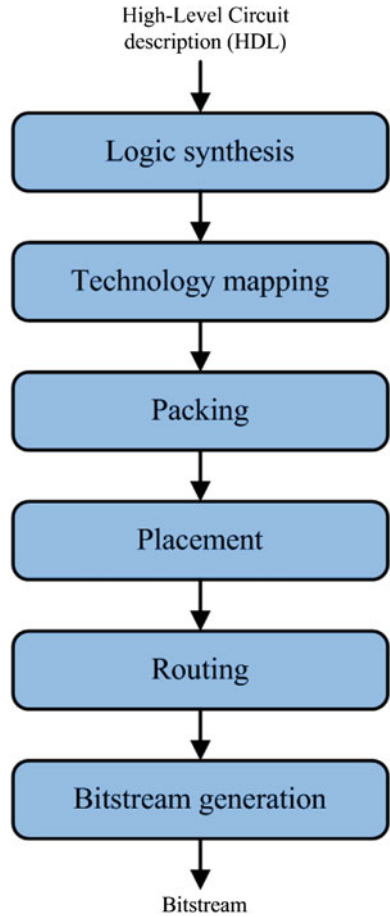
locality exhibited by the most of the designs and in turn offer smaller delays and more predictable routing compared to island-style architectures. The speed of a net is determined by the number of routing switches it has to pass and the length of wires. In a mesh-based architecture, the number of segments increase linearly with manhattan distance d between the logic blocks to be connected. However, for tree-based architecture the distance d between the blocks to be connected increases in a logarithmic manner [82]. This fact is illustrated in Fig. 2.16. On the other hand, scalability is an issue in hierarchical routing architectures and there might be some design mapping issues. But in the case of mesh-based architecture, there are no such issues as it offers a tile-based layout where a tile once formed can be replicated horizontally and vertically to make as large architecture as we wish.

2.5 Software Flow

FPGA architectures have been intensely investigated over the past two decades. A major aspect of FPGA architecture research is the development of Computer Aided Design (CAD) tools for mapping applications to FPGAs. It is well established that the quality of an FPGA-based implementation is largely determined by the effectiveness of accompanying suite of CAD tools. Benefits of an otherwise well designed, feature rich FPGA architecture might be impaired if the CAD tools cannot take advantage of the features that the FPGA provides. Thus, CAD algorithm research is essential to the necessary architectural advancement to narrow the performance gaps between FPGAs and other computational devices like ASICs.

The software flow (CAD flow) takes an application design description in a Hardware Description Language (HDL) and converts it to a stream of bits that is eventually programmed on the FPGA. The process of converting a circuit description into a format that can be loaded into an FPGA can be roughly divided into five distinct steps, namely: synthesis, technology mapping, mapping, placement and routing. The final output of FPGA CAD tools is a bitstream that configures the state of the memory

Fig. 2.17 FPGA software flow



bits in an FPGA. The state of these bits determines the logical function that the FPGA implements. Figure 2.17 shows a generalized software flow for programming an application circuit on an FPGA architecture. A description of various modules of software flow is given in the following part of this section. The details of these modules are generally indifferent to the kind of routing architecture being used and they are applicable to both architectures described earlier unless otherwise specified.

2.5.1 Logic Synthesis

The flow of FPGA starts with the logic synthesis of the netlist being mapped on it. Logic synthesis [26, 27] transforms an HDL description (VHDL or Verilog) into a set of boolean gates and Flip-Flops. The synthesis tools transform the

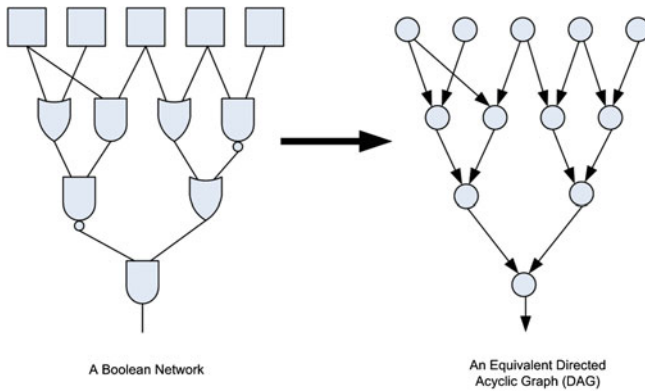


Fig. 2.18 Directed acyclic graph representation of a circuit

register-transfer-level (RTL) description of a design into a hierarchical boolean network. Various technology-independent techniques are applied to optimize the boolean network. The typical cost function of technology-independent optimizations is the total literal count of the factored representation of the logic function. The literal count correlates very well with the circuit area. Further details of logic synthesis are beyond the scope of this book.

2.5.2 Technology Mapping

The output from synthesis tools is a circuit description of Boolean logic gates, flip-flops and wiring connections between these elements. The circuit can also be represented by a Directed Acyclic Graph (*DAG*). Each node in the graph represents a gate, flip-flop, primary input or primary output. Each edge in the graph represents a connection between two circuit elements. Figure 2.18 shows an example of a DAG representation of a circuit. Given a library of cells, the technology mapping problem can be expressed as finding a network of cells that implements the Boolean network. In the FPGA technology mapping problem, the library of cells is composed of k -input LUTs and flip-flops. Therefore, FPGA technology mapping involves transforming the Boolean network into k -bounded cells. Each cell can then be implemented as an independent k -LUT. Figure 2.19 shows an example of transforming a Boolean network into k -bounded cells. Technology mapping algorithms can optimize a design for a set of objectives including depth, area or power. The FlowMap algorithm [64] is the most widely used academic tool for FPGA technology mapping. FlowMap is a breakthrough in FPGA technology mapping because it is able to find a depth-optimal solution in polynomial time. FlowMap guarantees depth optimality at the expense of logic duplication. Since the introduction of FlowMap, numerous technology mappers have been designed that optimize for area and run-time while still maintaining

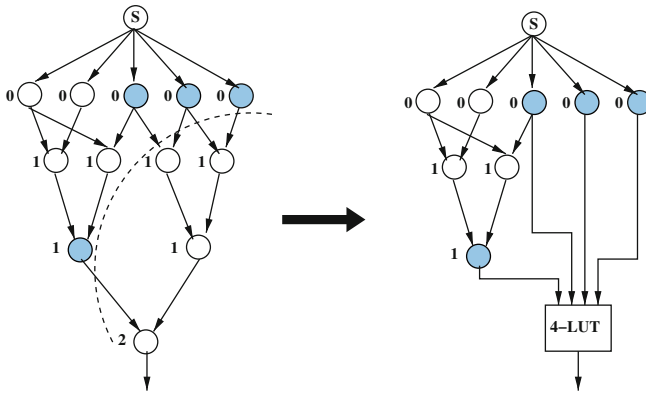


Fig. 2.19 Example of technology mapping

the depth-optimality of the circuit [65–67]. The result of the technology mapping step generates a network of k -bounded LUTs and flip-flops.

2.5.3 Clustering/Packing

The logic elements in a Mesh-based FPGA are typically arranged in two levels of hierarchy. The first level consists of logic blocks (LBs) which are k -input LUT and flip-flop pairs. The second level hierarchy groups k LBs together to form logic blocks clusters. The clustering phase of the FPGA CAD flow is the process of forming groups of k LBs. These clusters can then be mapped directly to a logic element on an FPGA. Figure 2.20 shows an example of the clustering process.

Clustering algorithms can be broadly categorized into three general approaches, namely top-down [39, 78], depth-optimal [84, 100] and bottom-up [14, 17, 43]. Top-down approaches partition the LBs into clusters by successively subdividing the network or by iteratively moving LBs between parts. Depth-optimal solutions attempt to minimize delay at the expense of logic duplication. Bottom-up approaches are generally preferred for FPGA CAD tools due to their fast run times and reasonable timing delays. They only consider local connectivity information and can easily satisfy clusters pin constraints. Top-down approaches offer the best solutions; however, their computational complexity can be prohibitive.

2.5.3.1 Bottom-up Approaches

Bottom-up approaches build clusters sequentially one at a time. The process starts by choosing an LB which acts as a cluster seed. LBs are then greedily selected and added to the cluster, applying various attraction functions. The VPack [14] attraction

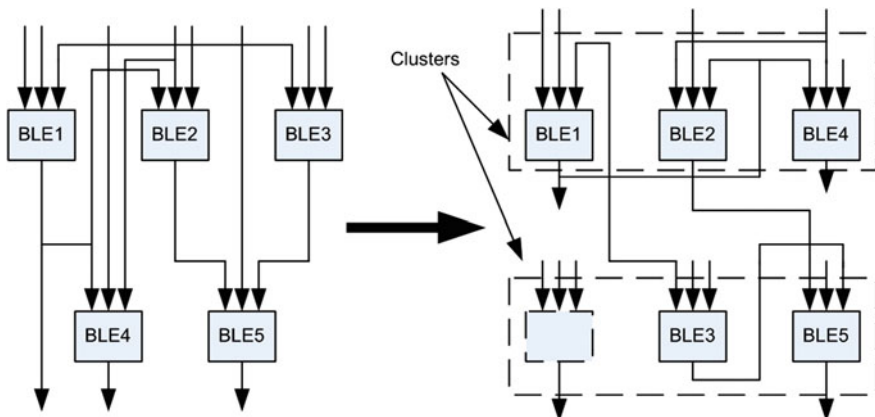


Fig. 2.20 Example of packing

function is based on the number of shared nets between a candidate LB and the LBs that are already in the cluster. For each cluster, the attraction function is used to select a seed LB from the set of all LBs that have not already been packed. After packing a seed LB into the new cluster, a second attraction function selects new LBs to pack into the cluster. LBs are packed into the cluster until the cluster reaches full capacity or all cluster inputs have been used. If all cluster inputs become occupied before this cluster reaches full capacity, a hill-climbing technique is applied, searching for LBs that do not increase the number of inputs used by the cluster. The VPack pseudo-code is outlined in algorithm 2.1.

T-VPack [22] is a timing-driven version of VPack which gives added weight to grouping LBs on the critical path together. The algorithm is identical to VPack, however, the attraction functions which select the LBs to be packed into the clusters are different. The VPack seed function chooses LBs with the most used inputs, whereas the T-VPack seed function chooses LBs that are on the most critical path. VPack's second attraction function chooses LBs with the largest number of connections with the LBs already packed into the cluster. T-VPack's second attraction function has two components for a LB B being considered for cluster C :

$$Attraction(B, C) = \alpha.Crit(B) + (1 - \alpha) \frac{|Nets(B) \cap Nets(C)|}{G} \quad (2.1)$$

where $Crit(B)$ is a measure of how close LB B is to being on the critical path, $Nets(B)$ is the set of nets connected to LB B , $Nets(C)$ is the set of nets connected to the LBs already selected for cluster C , α is a user-defined constant which determines the relative importance of the attraction components, and G is a normalizing factor. The first component of T-VPack's second attraction function chooses critical-path LBs, and the second chooses LBs that share many connections with the LBs already packed into the cluster. By initializing and then packing clusters with


```

UnclusteredLBs = PatternMatchToLBs(LUTs,Registers);
LogicClusters = NULL;
while UnclusteredLBs != NULL do
  C = GetLBwithMostUsedInputs(UnclusteredLBs);
  while | C | < k do
    /*cluster is not full*/
    BestLB = MaxAttractionLegalLB(C,UnclusteredLBs);
    if BestLB == NULL then
      /*No LB can be added to this cluster*/
      break;
    endif
    UnclusteredLBs = UnclusteredLB - BestLB;
    C = C  $\cup$  BestLB;
  endw
  if | C | < k then
    /*Cluster is not full - try hill climbing*/
    while | C | < k do
      BestLB = MinClusterInputIncreaseLB(C,UnclusteredLBs);
      C = C  $\cup$  BestLB;
      UnclusteredLBs = UnclusteredLB - BestLB;
    endw
    if ClusterIsIllegal(C) then
      RestoreToLastLegalState(C,UnclusteredLBs);
    endif
  endif
  LogicClusters = LogicClusters  $\cup$  C;
endw

```

Algorithm 2.1 Pseudo-code of the VPack Algorithm [22]

critical-path LBs, the algorithm is able to absorb long sequences of critical-path LBs into clusters. This minimizes circuit delay since the local interconnect within the cluster is significantly faster than the global interconnect of the FPGA. RPack [43] improves routability of a circuit by introducing a new set of routability metrics. RPack significantly reduced the channel widths required by circuits compared to VPack. T-RPack [43] is a timing driven version of RPack which is similar to T-VPack by giving added weight to grouping LBs on the critical path. iRAC [17] improves the routability of circuits even further by using an attraction function that attempts to encapsulate as many low fanout nets as possible within a cluster. If a net can be completely encapsulated within a cluster, there is no need to route that net in the external routing network. By encapsulating as many nets as possible within clusters, routability is improved because there are less external nets to route in total.

2.5.3.2 Top-down Approaches

The K-way partitioning problem seeks to minimize a given cost function of such an assignment. A standard cost function is net cut, which is the number of hyper-edges that span more than one partition, or more generally, the sum of weights of

such hyperedges. Constraints are typically imposed on the solution, and make the problem difficult. For example some vertices can be fixed in their parts or the total vertex weight in each part must be limited (balance constraint and FPGA clusters size). With balance constraints, the problem of partitioning optimally a hypergraph is known to be NP-hard [85]. However, since partitioning is critical in several practical applications, heuristic algorithms were developed with near-linear runtime. Such move-based heuristics for k-way hypergraph partitioning appear in [24, 34, 110].

Fiduccia-Mattheyses Algorithm

The Fiduccia-Mattheyses (FM) heuristics [34] work by prioritizing moves by gain. A move changes to which partition a particular vertex belongs, and the gain is the corresponding change of the cost function. After each vertex is moved, gains for connected modules are updated.

```

partitioning = initial_solution;
while solution quality improves do
  Initialize gain_container from partitioning;
  solution_cost = partitioning.get_cost();
  while not all vertices locked do
    move = choose_move();
    solution_cost += gain_container.get_gain(move);
    gain_container.lock_vertex(move.vertex());
    gain_update(move);
    partitioning.apply(move);
  endw
  roll back partitioning to best seen solution;
  gain_container.unlock_all();
endw

```

Algorithm 2.2 Pseudo-code for FM Heuristic [38]

The Fiduccia-Mattheyses (FM) heuristic for partitioning hypergraphs is an iterative improvement algorithm. FM starts with a possibly random solution and changes the solution by a sequence of moves which are organized as passes. At the beginning of a pass, all vertices are free to move (unlocked), and each possible move is labeled with the immediate change to the cost it would cause; this is called the gain of the move (positive gains reduce solution cost, while negative gains increase it). Iteratively, a move with highest gain is selected and executed, and the moving vertex is locked, i.e., is not allowed to move again during that pass. Since moving a vertex can change gains of adjacent vertices, after a move is executed all affected gains are updated. Selection and execution of a best-gain move, followed by gain update, are repeated until every vertex is locked. Then, the best solution seen during the pass is adopted as the starting solution of the next pass. The algorithm terminates when a

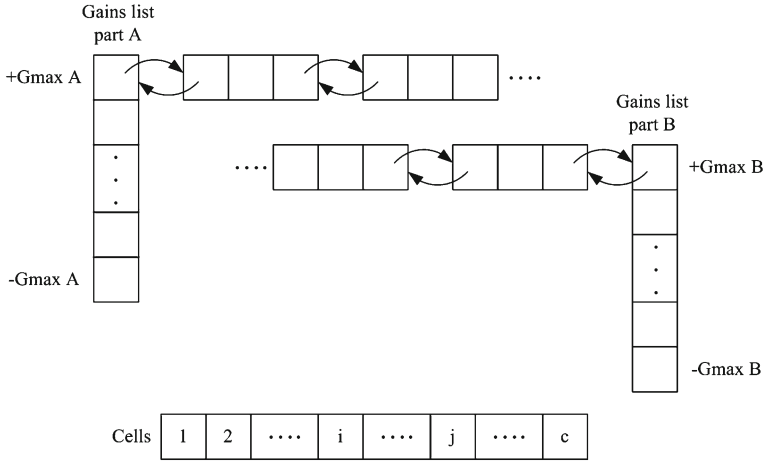


Fig. 2.21 The gain bucket structure as illustrated in [34]

pass fails to improve solution quality. Pseudo-code for the FM heuristic is given in algorithm 2.2.

The FM algorithm has 3 main components (1) computation of initial gain values at the beginning of a pass; (2) the retrieval of the best-gain (feasible) move; and (3) the update of all affected gain values after a move is made. One contribution of Fiduccia and Mattheyses lies in observing that circuit hypergraphs are sparse, and any move's gain is bounded between plus and minus the maximal vertex degree G_{max} in the hypergraph (times the maximal hyperedge weight, if weights are used). This allows prioritizing moves by their gains. All affected gains can be updated in amortized-constant time, giving overall linear complexity per pass [34]. All moves with the same gain are stored in a linked list representing a "gain bucket". Figure 2.21 presents the gain bucket list structure. It is important to note that some gains G may be negative, and as such, FM performs hill-climbing and is not strictly greedy.

Multilevel Partitioning

The multilevel hypergraph partitioning framework was successfully verified by [31, 48, 49] and leads to the best known partitioning results ever since. The main advantage of multilevel partitioning over flat partitioners is its ability to search the solution space more effectively by spending comparatively more effort on smaller coarsened hypergraphs. Good coarsening algorithms allow for high correlation between good partitioning for coarsened hypergraphs and good partitioning for the initial hypergraph. Therefore, a thorough search at the top of the multilevel hierarchy is worthwhile because it is relatively inexpensive when compared to flat partitioning of the original hypergraph, but can still preserve most of the possible improvement.

The result is an algorithmic framework with both improved runtime and solution quality over a completely flat approach. Pseudo-code for an implementation of the multilevel partitioning framework is given in algorithm 2.3.

```

level = 0;
hierarchy[level] = hypergraph;
min_vertices = 200;
while hierarchy[level].vertex_count() > min_vertices do
    next_level = cluster(hierarchy[level]);
    level = level + 1;
    hierarchy[level] = next_level;
endw
partitioning[level] = a random initial solution for top-level hypergraph;
FM(hierarchy[level], partitioning[level]);
while level > 0 do
    level = level - 1;
    partitioning[level] = project(partitioning[level+1], hierarchy[level]);
    FM(hierarchy[level], partitioning[level]);
endw

```

Algorithm 2.3 Pseudo-code for the Multilevel Partitioning Algorithm [38]

As illustrated in Fig. 2.22, multilevel partitioning consists of 3 main components: clustering, top-level partitioning and refinement or “uncoarsening”. During clustering, hypergraph vertices are combined into clusters based on connectivity, leading to a smaller, clustered hypergraph. This step is repeated until obtaining only several hundred clusters and a hierarchy of clustered hypergraphs. We describe this hierarchy, as shown in Fig. 2.22, with the smaller hypergraphs being “higher” and the larger hypergraphs being “lower”. The smallest (top-level) hypergraph is partitioned with a very fast initial solution generator and improved iteratively, for example, using the FM algorithm. The resulting partitioning is then interpreted as a solution for the next hypergraph in the hierarchy. During the refinement stage, solutions are projected from one level to the next and improved iteratively. Additionally, the hMETIS partitioning program [49] introduced several new heuristics that are incorporated into their multilevel partitioning implementation and are reportedly performance critical.

2.5.4 Placement

Placement algorithms determine which logic block within an FPGA should implement the corresponding logic block (instance) required by the circuit. The optimization goals consist in placing connected logic blocks close together to minimize the required wiring (wire length-driven placement), and sometimes to place blocks to balance the wiring density across the FPGA (routability-driven placement) or to maximize circuit speed (timing-driven placement). The 3 major classes of

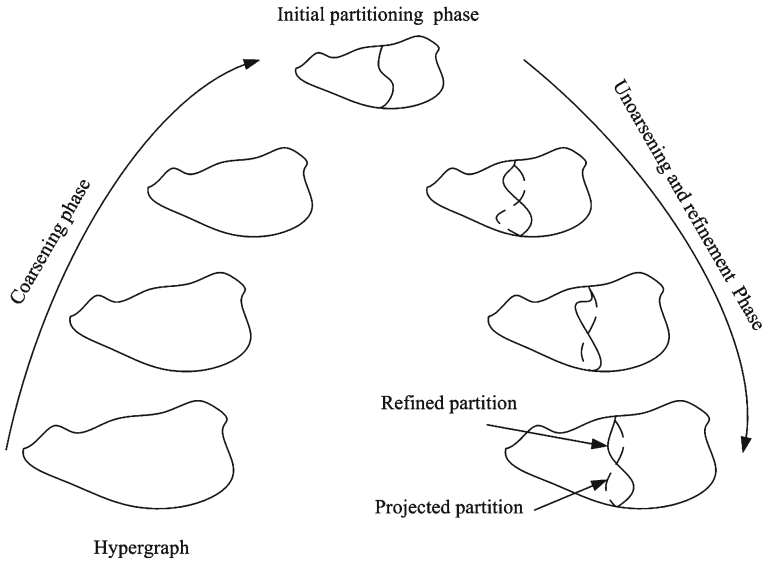


Fig. 2.22 Multilevel hypergraph bisection

placers in use today are min-cut (Partitioning-based) [6, 40], analytic [32, 53] which are often followed by local iterative improvement, and simulated annealing based placers [37, 105]. To investigate architectures fairly we must make sure that our CAD tools are attempting to use every FPGA's feature. This means that the optimization approach and goals of the placer may change from architecture to architecture. Partitioning and simulated annealing approaches are the most common and used in FPGA CAD tools. Thus we focus on both techniques in the sequel.

2.5.4.1 Simulated Annealing Based Approach

Simulated annealing mimics the annealing process used to cool gradually molten metal to produce high-quality metal objects [105]. Pseudo-code for a generic simulated annealing-based placer is shown in algorithm 2.4. A cost function is used to evaluate the quality of a given placement of logic blocks. For example, a common cost function in wirelength-driven placement is the sum over all nets of the half perimeter of their bounding boxes. An initial placement is created by assigning logic blocks randomly to the available locations in the FPGA. A large number of moves, or local improvements are then made to gradually improve the placement. A logic block is selected at random, and a new location for it is also selected randomly. The change in cost function that results from moving the selected logic block to the proposed new location is computed. If the cost decreases, the move is always accepted and the block is moved. If the cost increases, there is still a chance to accept the move, even though it makes the placement worse. This probability of acceptance is

```

S = RandomPlacement();
T = InitialTemperature();
Rlimit = InitialRlimit;
while ExitCriterion() == false do
  while InnerLoopCriterion() == false do
    Snew = GenerateViaMove(S, Rlimit);
    ΔC = Cost(Snew) - Cost(S);
    r = random(0,1);
    if r < e- $\frac{\Delta C}{T}$  then
      S = Snew;
    endif
  endw
  T = UpdateTemp();
  Rlimit = UpdateRlimit();
endw

```

Algorithm 2.4 Generic Simulated Annealing-based Placer [22]

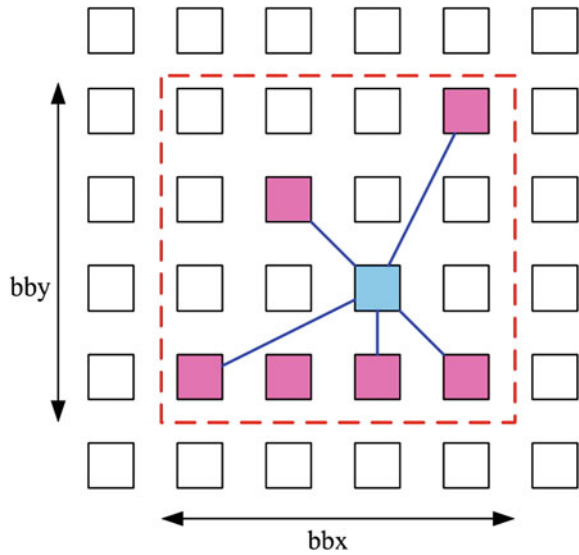
given by $e^{-\frac{\Delta C}{T}}$, where ΔC is the change in cost function, and T is a parameter called temperature that controls probability of accepting moves that worsen the placement. Initially, T is high enough so almost all moves are accepted; it is gradually decreased as the placement improves, in such a way that eventually the probability of accepting a worsening move is very low. This ability to accept hill-climbing moves that make a placement worse allows simulated annealing to escape local minima of the cost function.

The R_{limit} parameter in algorithm 2.4 controls how close are together blocks must be to be considered for swapping. Initially, R_{limit} is fairly large, and swaps of blocks far apart on a chip are more likely. Throughout the annealing process, R_{limit} is adjusted to try to keep the fraction of accepted moves at any temperature close to 0.44. If the fraction of moves accepted, α , is less than 0.44, R_{limit} is reduced, while if α is greater than 0.44, R_{limit} is increased.

In [22], the objective cost function is a function of the total wirelength of the current placement. The wirelength is an estimate of the routing resources needed to completely route all nets in the netlist. Reductions in wirelength mean fewer routing wires and switches are required to route nets. This point is important because routing resources in an FPGA are limited. Fewer routing wires and switches typically are also translated into reductions of the delay incurred in routing nets between logic blocks. The total wirelength of a placement is estimated using a semi-perimeter metric, and is given by Eq. 2.2. N is the total number of nets in the netlist, $bb_x(i)$ is the horizontal span of net i , $bb_y(i)$ is its vertical span, and $q(i)$ is a correction factor. Figure 2.23 illustrates the calculation of the horizontal and vertical spans of a hypothetical net that has 6 terminals.

$$WireCost = \sum_{i=1}^N q(i) \times (bb_x(i) + bb_y(i)) \quad (2.2)$$

Fig. 2.23 Bounding box of a hypothetical 6-terminal net [22]



The temperature decrease rate, the exit criterion for terminating the anneal, the number of moves attempted at each temperature (InnerLoopCriterion), and the method by which potential moves are generated are defined by the annealing schedule. An efficient annealing schedule is crucial to obtain good results in a reasonable amount of CPU time. Many proposed annealing schedules are “fixed” schedules with no ability to adapt to different problems. Such schedules can work well within the narrow application range for which they are developed, but their lack of adaptability means they are not very general. In [86] authors propose an “adaptive” annealing schedule based on statistics computed during the anneal itself. Adaptive schedules are widely used to solve large scale optimization problems with many variables.

2.5.4.2 Partitioning Based Approach

Partitioning-based placement methods, are based on graph partitioning algorithms such as the Fiduccia-Mattheyses (FM) algorithm [34], and Kernighan Lin (KL) algorithm [6]. Partitioning-based placement are suitable to Tree-based FPGA architectures. The partitioner is applied recursively to each hierarchical level to distribute netlist cells between clusters. The aim is to reduce external communications and to collect highly connected cells into the same cluster.

The partitioning-based placement is also used in the case of Mesh-based FPGA. The device is divided into two parts, and a circuit partitioning algorithm is applied to determine the adequate part where a given logic block must be placed to minimize the number of cuts in the nets that connect the blocks between partitions, while leaving highly-connected blocks in one partition.

A divide-and-conquer strategy is used in these heuristics. By partitioning the problem into sub-parts, a drastic reduction in search space can be achieved. On the whole, these algorithms perform in the top-down manner, placing blocks in the general regions which they should belong to. In the Mesh FPGA case, partitioning-based placement algorithms are good from a “global” perspective, but they do not actually attempt to minimize wirelength. Therefore, the solutions obtained are sub-optimal in terms of wirelength. However, these classes of algorithms run very fast. They are normally used in conjunction with other search techniques for further quality improvement. Some algorithms [130] and [95] combine multi-level clustering and hierarchical simulated annealing to obtain ultra-fast placement with good quality. In the following chapters, the partitioning-based placement approach will be used only for Tree-based FPGA architectures.

2.5.5 Routing

The FPGA routing problem consists in assigning nets to routing resources such that no routing resource is shared by more than one net. *Pathfinder* [80] is the current, state-of-the-art FPGA routing algorithm. *Pathfinder* operates on a directed graph abstraction $G(V, E)$ of the routing resources in an FPGA. The set of vertices V in the graph represents the IO terminals of logic blocks and the routing wires in the interconnect structure. An edge between two vertices represents a potential connection between them. Figure 2.24 presents a part of a routing graph in a Mesh-based interconnect.

Given this graph abstraction, the routing problem for a given net is to find a directed tree embedded in G that connects the source terminal of the net to each of its sink terminals. Since the number of routing resources in an FPGA is limited, the goal of finding unique, non-intersecting trees for all the nets in a netlist is a difficult problem.

Pathfinder uses an iterative, negotiation-based approach to successfully route all the nets in a netlist. During the first routing iteration, nets are freely routed without paying attention to resource sharing. Individual nets are routed using *Dijkstra's* shortest path algorithm [111]. At the end of the first iteration, resources may be congested because multiple nets have used them. During subsequent iterations, the cost of using a resource is increased, based on the number of nets that share the resource, and the history of congestion on that resource. Thus, nets are made to negotiate for routing resources. If a resource is highly congested, nets which can use lower congestion alternatives are forced to do so. On the other hand, if the alternatives are more congested than the resource, then a net may still use that resource.

The cost of using a routing resource n during a routing iteration is given by Eq. 2.3.

$$c_n = (b_n + h_n) \times p_n \quad (2.3)$$

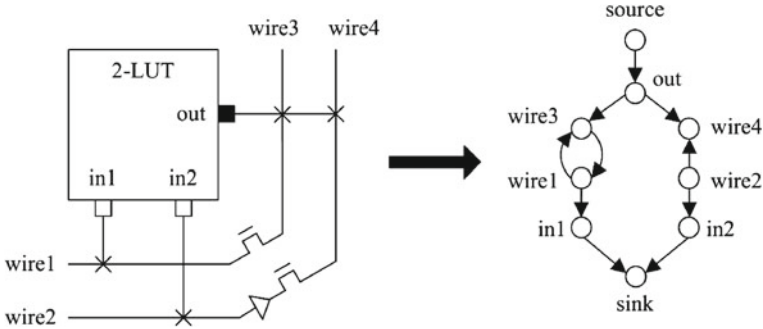


Fig. 2.24 Modeling FPGA architecture as a directed graph [22]

b_n is the base cost of using the resource n , h_n is related to the history of congestion during previous iterations, and p_n is proportional to the number of nets sharing the resource in the current iteration. The p_n term represents the cost of using a shared resource n , and the h_n term represents the cost of using a resource that has been shared during earlier routing iterations. The latter term is based on the intuition that a historically congested node should appear expensive, even if it is slightly shared currently. Cost functions and routing schedule were described in details in [22]. The Pseudo-code of the *Pathfinder* routing algorithm is presented in algorithm 2.5.

```

Let:  $RT_i$  be the set of nodes in the current routing of net  $i$ 
while shared resources exist do
  /*Illegal routing*/
  foreach net,  $i$  do
    rip-up routing tree  $RT_i$ ;
     $RT(i) = s_i$ ;
    foreach sink  $t_{ij}$  do
      Initialize priority queue PQ to  $RT_i$  at cost 0;
      while sink  $t_{ij}$  not found do
        Remove lowest cost node  $m$  from PQ;
        foreach fanout node  $n$  of node  $m$  do
          Add  $n$  to PQ at  $PathCost(n) = c_n + PathCost(m)$ ;
        endfch
      endw
      foreach node  $n$  in path  $t_{ij}$  to  $s_i$  do
        /*backtrace*/
        Update  $c_n$ ;
        Add  $n$  to  $RT_i$ ;
      endfch
    endfch
  endw
  update  $h_n$  for all  $n$ ;
endw
    
```

Algorithm 2.5 Pseudo-code of the *Pathfinder* Routing Algorithm [80]

An important measure of routing quality produced by an FPGA routing algorithm is the critical path delay. The critical path delay of a routed netlist is the maximum delay of any combinational path in the netlist. The maximum frequency at which a netlist can be clocked has an inverse relationship with critical path delay. Thus, larger critical path delays slow down the operation of netlist. Delay information is incorporated into *Pathfinder* by redefining the cost of using a resource n (Eq. 2.4).

$$c_n = A_{ij} \times d_n + (1 - A_{ij}) \times (b_n + h_n) \times p_n \quad (2.4)$$

The c_n term is from Eq. 2.3, d_n is the delay incurred in using the resource, and A_{ij} is the criticality given by Eq. 2.5.

$$A_{ij} = \frac{D_{ij}}{D_{max}} \quad (2.5)$$

D_{ij} is the maximum delay of any combinational path going through the source and sink terminals of the net being routed, and D_{max} is the critical path delay of the netlist. Equation 2.4 is formulated as a sum of two cost terms. The first term in the equation represents the delay cost of using resource n , while the second term represents the congestion cost. When a net is routed, the value of A_{ij} determines whether the delay or the congestion cost of a resource dominates. If a net is near critical (i.e. its A_{ij} is close to 1), then congestion is largely ignored and the cost of using a resource is primarily determined by the delay term. If the criticality of a net is low, the congestion term in Eq. 2.4 dominates, and the route found for the net avoids congestion while potentially incurring delay.

Pathfinder has proved to be one of the most powerful FPGA routing algorithms to date. The negotiation-based framework that trades off delay for congestion is an extremely effective technique for routing signals on FPGAs. More importantly, *Pathfinder* is a truly architecture-adaptive routing algorithm. The algorithm operates on a directed graph abstraction of an FPGA's routing structure, and can thus be used to route netlists on any FPGA that can be represented as a directed routing graph.

2.5.6 Timing Analysis

Timing analysis [99] is used for two basic purposes:

- To determine the speed of circuits which have been completely placed and routed,
- To estimate the slack [68] of each source-sink connection during routing (placement and other parts of the CAD flow) in order to decide which connections must be made via fast paths to avoid slowing down the circuit.

First the circuit under consideration is presented as a directed graph. Nodes in the graph represent input and output pins of circuit elements such as LUTs, registers,

and I/O pads. Connections between these nodes are modeled with edges in the graph. Edges are added between the inputs of combinational logic Blocks (LUTs) and their outputs. These edges are annotated with a delay corresponding to the physical delay between the nodes. Register input pins are not joined to register output pins. To determine the delay of the circuit, a breadth first traversal is performed on the graph starting at sources (input pads, and register outputs). Then the arrival time, $T_{arrival}$, at all nodes in the circuit is computed with the following equation:

$$T_{arrival}(i) = \max_{j \in fanin(i)} \{T_{arrival}(j) + delay(j, i)\}$$

where node i is the node currently being computed, and $delay(j, i)$ is the delay value of the edge joining node j to node i . The delay of the circuit is then the maximum arrival time, D_{max} , of all nodes in the circuit.

To guide a placement or routing algorithm, it is useful to know how much delay may be added to a connection before the path that the connection is on becomes critical. The amount of delay that may be added to a connection before it becomes critical is called the slack of that connection. To compute the slack of a connection, one must compute the required arrival time, $T_{required}$, at every node in the circuit. We first set the $T_{required}$ at all sinks (output pads and register inputs) to be D_{max} . Required arrival time is then propagated backwards starting from the sinks with the following equation:

$$T_{required}(i) = \min_{j \in fanout(i)} \{T_{required}(j) - delay(j, i)\}$$

Finally, the slack of a connection (i, j) driving node, j , is defined as:

$$Slack(i, j) = T_{required}(j) - T_{arrival}(i) - delay(i, j)$$

2.5.7 Bitstream Generation

Once a netlist is placed and routed on an FPGA, bitstream information is generated for the netlist. This bitstream is programmed on the FPGA using a bitstream loader. The bitstream of a netlist contains information as to which SRAM bit of an FPGA be programmed to 0 or to 1. The bitstream generator reads the technology mapping, packing and placement information to program the SRAM bits of Look-Up Tables. The routing information of a netlist is used to correctly program the SRAM bits of connection boxes and switch boxes.

2.6 Research Trends in Reconfigurable Architectures

Until now in this chapter a detailed overview of logic architecture, routing architecture and software flow of FPGAs is presented. In this section, we highlight some of the disadvantages associated with FPGAs and further we describe some of the trends that

are currently being followed to remedy these disadvantages. FPGA-based products are basically very effective for low to medium volume production as they are easy to program and debug, and have less NRE cost and faster time-to-market. All these major advantages of an FPGA come through their reconfigurability which makes them general purpose and field programmable. But, the very same reconfigurability is the major cause of its disadvantages; thus making it larger, slower and more power consuming than ASICs.

However, the continued scaling of CMOS and increased integration has resulted in a number of alternative architectures for FPGAs. These architectures are mainly aimed to improve area, performance and power consumption of FPGA architectures. Some of these propositions are discussed in this section.

2.6.1 Heterogeneous FPGA Architectures

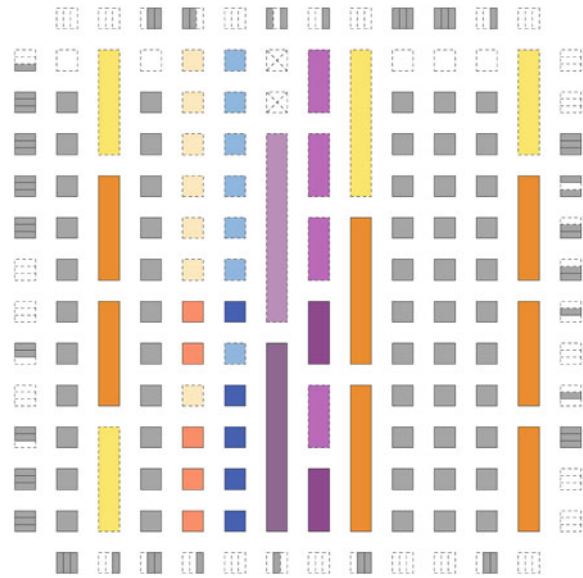
Use of hard-blocks in FPGAs improves their logic density. Hard-Blocks, in FPGAs increase their density, performance and power consumption. There can be different types of hard-blocks like multipliers, adders, memories, floating point units and DSP blocks etc. In this regard, [19] have incorporated embedded floating-point units in FPGAs, [30] have developed virtual embedded block methodology to model arbitrary embedded blocks on existing commercial FPGAs. Here some of the academic and commercial architectures are presented that make use of hard-blocks to improve overall efficiency of FPGAs.

2.6.1.1 Versatile Packing, Placement and Routing VPR

Versatile Packing, Placement and Routing for FPGAs (commonly known as VPR) [14, 22, 120] is the most widely used academic mesh-based FPGA exploration environment. It allows to explore mesh-based FPGA architectures by employing an empirical approach. Benchmark circuits are mapped, placed and routed on a desired FPGA architecture. Later, area and delay of FPGAs are measured to decide best architectural parameters. Different CAD tools in VPR are highly optimized to ensure high quality results.

Earlier version of VPR supported only homogeneous architectures [120]. However, the latest version of VPR known as VPR 5.0 [81] supports hard-blocks (such as multiplier and memory blocks) and single-driver routing wires. Hard-blocks are restricted to be in one grid width column, and that column can be composed of only similar type of blocks. The height of a hard-block is quantized and it must be an integral multiple of grid units. In case a block height is indivisible with the height of FPGA, some grid locations are left empty. Figure 2.25 illustrates a heterogeneous FPGA with 8 different kinds of blocks.

Fig. 2.25 A heterogeneous FPGA in VPR 5.0 [81]



2.6.1.2 Madeo, a Framework for Exploring Reconfigurable Architectures

Madeo [73] is another academic design suite for the exploration of reconfigurable architectures. It includes a modeling environment that supports multi-grained, heterogeneous architectures with irregular topologies. Madeo framework initially allows to model an FPGA architecture. The architecture characteristics are represented as a common abstract model. Once the architecture is defined, the CAD tools of Madeo are used to map a target netlist on the architecture. Madeo uses same placement and routing algorithms as used by VPR [120]. Along with placement and routing algorithms, it also embeds a bitstream generator, a netlist simulator, and a physical layout generator in its design suite. Madeo supports architectural prospection and very fast FPGA prototyping. Several FPGAs, including some commercial architectures (such as Xilinx Virtex family) and prospective ones (such as STMicro LPPGA) have been modeled using Madeo. The physical layout is produced as VHDL description.

2.6.1.3 Altera Architecture

Altera’s Stratix IV [107] is an example of a commercial architecture that uses a heterogeneous mixture of blocks. Figure 2.26 shows the global architectural layout of Stratix IV. The logic structure of Stratix IV consists of LABs (Logic Array Blocks), memory blocks and digital signal processing (DSP) blocks. LABS are distributed symmetrically in rows and columns and are used to implement general purpose logic. The DSP blocks are used to implement full-precision multipliers of different

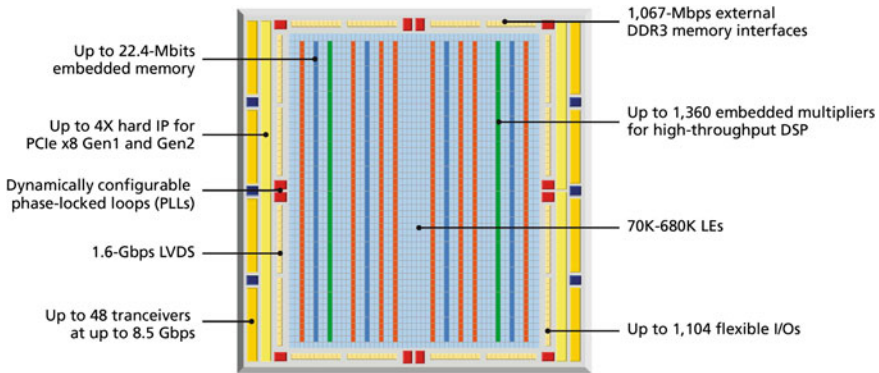


Fig. 2.26 Stratix IV architectural elements

granularities. The memory blocks and DSP blocks are placed in columns at equal distance with one another. Input and Output (I/Os) are located at the periphery of architecture.

Logic array blocks (LABs) and adaptive logic modules (ALMs) provide the basic logic capacity for Stratix IV device. They can be used to configure logic functions, arithmetic functions, and register functions. Each LAB consists of ten ALMs, carry chains, arithmetic chains, LAB control signals, local interconnect, and register chain connection lines. The local interconnect connects the ALMs that are inside same LAB. The direct link allows a LAB to drive into the local interconnect of its left or right neighboring LAB. The register chain connects the output of ALM register to the adjacent ALM register in the LAB. A memory LAB (MLAB) is a derivative of LAB which can be either used just like a simple LAB, or as a static random access memory (SRAM). Each ALM in an MLAB can be configured as a 64×1 , or 32×2 blocks, resulting in a configuration of 64×10 or 32×20 simple dual-port SRAM block. MLAB and LAB blocks always coexist as pairs in Stratix IV families.

The DSP blocks in Stratix IV are optimized for signal processing applications such as Finite Impulse Response (FIR), Infinite Impulse Response (IIR), Fast Fourier Transform functions (FFT) and encoders etc. Stratix IV device has two to seven columns of DSP blocks that can implement different operations like multiplication, multiply-add, multiply-accumulate (MAC) and dynamic arithmetic or logical shift functions. The DSP block supports different multiplication operations such as 9×9 , 12×12 , 18×18 and 36×36 multiplication operations. The Stratix IV devices contain three different sizes of embedded SRAMs. The memory sizes include 640-bit memory logic array blocks (MLABs), 9-Kbit M9K blocks, and 144-Kbit M144K blocks. The MLABs have been optimized to implement filter delay lines, small FIFO buffers, and shift registers. M9K blocks can be used for general purpose memory applications, and M144K are generally meant to store code for a processor, packet buffering or video frame buffering.

2.6.2 FPGAs to Structured Architectures

The ease of designing and prototyping with FPGAs can be exploited to quickly design a hardware application on an FPGA. Later, improvements in area, speed, power and volume production can be achieved by migrating the application design from FPGA to other technologies such as Structured-ASICs. In this regard, Altera provides a facility to migrate its Stratix IV based application design to HardCopy IV [56]. Altera gives provision to migrate FPGA-based applications to Structured-ASIC. Their Structured-ASIC is called as HardCopy [56]. The main theme is to design, test and even initially ship a design using an FPGA. Later, the application circuit that is mapped on the FPGA can be seamlessly migrated to HardCopy for high volume production. Their latest HardCopy-IV devices offer pin-to-pin compatibility with the Stratix IV prototype, making them exact replacements for the FPGAs. Thus, the same system board and softwares developed for prototyping and field trials can be retained, enabling the lowest risk and fastest time-to-market for high-volume production. Moreover, when an application circuit is migrated from Stratix IV FPGA prototype to Hardcopy-VI, the core logic performance doubles and power consumption reduces by half.

The basic logic unit of HardCopy is termed as HCell. It is similar to Stratix IV logic cell (LAB) in the sense that the fabric consists of a regular pattern which is formed by tiling one or more basic cells in a two dimensional array. However, the difference is that HCell has no configuration memory. Different HCell candidates can be used, ranging from fine-grained NAND gates to multiplexors and coarse-grained LUTs. An array of such HCells, and a general purpose routing network which interconnects them is laid down on the lower layers of the chip. Specific layers are then reserved to form via connections or metal lines which are used to customize the generic array into specific functionality. Figure 2.27 illustrates the correspondence between an FPGA and a compatible structured ASIC. There is a one to one layout-level correspondence between MRAMs, phase-lock loops (PLLs), embedded memories, transceivers, and I/O blocks. The soft-logic DSP multipliers and logic cell fabric of the FPGA are re-synthesized to structured ASIC fabric. However, they remain functionally and electrically equivalent in FPGAs and HardCopy ASICs.

Apart from Altera, there are several other companies that provide a solution similar to that of Altera. For example, the eASIC Nextreme [41] uses an FPGA-like design flow to map an application design on SRAM programmable LUTs, which are later interconnected through mask programming of few upper routing layers. Tierlogic [113] is a recently launched FPGA vendor that offers 3D SRAM-based TierFPGA devices for prototyping and early production. The same design solution can be frozen to a TierASIC device with one low-NRE custom mask for error-free transition to an ASIC implementation. The SRAM layer is placed on an upper 3D layer of TierFPGA. Once the TierFPGA design is frozen, the bitstream information is used to create a single custom mask metal layer that will replace the SRAM programming layer.

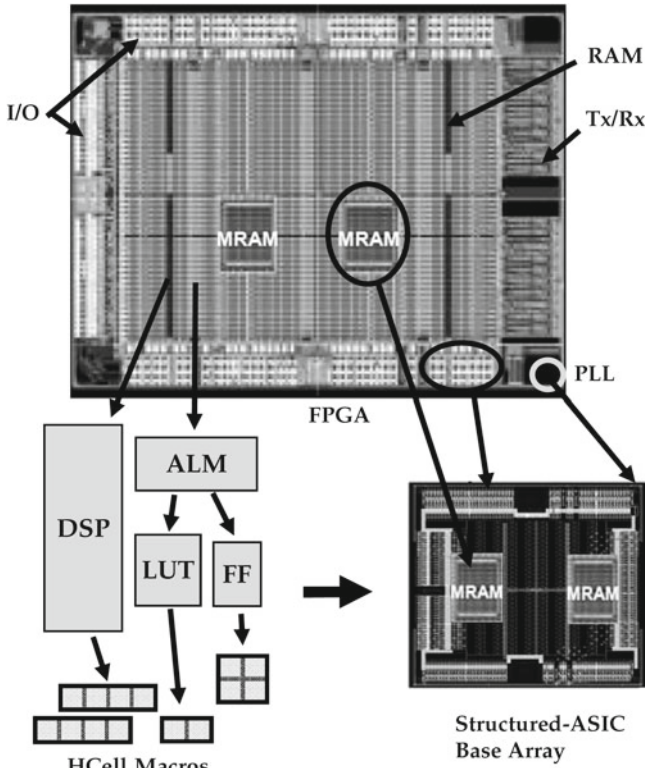


Fig. 2.27 FPGA/Structured-ASIC (HardCopy) Correspondence [59]

2.6.3 Configurable ASIC Cores

Configurable ASIC Core (cASIC) [35] is another example of reconfigurable devices that can implement a limited set of circuits which operate at mutually exclusive times. cASICs are intended as accelerator in domain-specific systems-on-a-chip, and are not designed to replace the entire ASIC-only chip. The host would execute software code, whereas compute-intensive sections can be executed on one or more cASICs. So, to execute the compute intensive sections, cASICs implement only data-path circuits and thus supports full-word blocks only (such as 16-bit wide multipliers, adders, RAMS, etc). Since the application domain of cASICs is more specific, they are significantly smaller than FPGAs. As hardware resources are shared between different netlists, cASICs are even smaller than the sum of the standard-cell based ASIC areas of individual circuits.

2.6.4 Processors Inside FPGAs

Considerable amount of FPGA area can be reduced by incorporating a microprocessor in an FPGA. A microprocessor can execute any less compute intensive task, whereas compute-intensive tasks can be executed on an FPGA. Similarly, a microprocessor based application can have huge speed-up gains if an FPGA is attached with it. An FPGA attached with a microprocessor can execute any compute intensive functionality as a customized hardware instruction. These advantages have compelled commercial FPGA vendors to provide microprocessor in their FPGAs so that complete system can be programmed on a single chip. Few vendors have integrated fixed hard processor on their FPGA (like AVR Processor integrated in Atmel FPSLIC [18] or PowerPC processors embedded in Xilinx Virtex-4 [126]). Others provide soft processor cores which are highly optimized to be mapped on the programmable resources of FPGA. Altera's Nios [90] and Xilinx's Microblaze [88] are soft processor meant for FPGA designs which allow custom hardware instructions. [96] have shown that considerable area gains can be achieved if these soft processors for FPGAs are optimized for particular applications. They have shown that unused instructions in a soft processor can be removed and different architectural tradeoffs can be selected to achieve on average 25% area gain for soft processors required for specific applications. Reconfigurable units can also be attached with microprocessors to achieve execution time speedup in software programs. [28, 70, 104] have incorporated a reconfigurable unit with microprocessors to achieve execution-time speedup.

2.6.5 Application Specific FPGAs

The type of logic blocks and the routing network in an FPGA can be optimized to gain area and performance advantages for a given application domain (controlpath-oriented applications, datapath-oriented applications, etc). These types of FPGAs may include different variety of desired hard-blocks, appropriate amount of flexibility required for the given application domain or bus-based interconnects rather than bit-based interconnects. Authors in [83] have presented a reconfigurable arithmetic array for multimedia applications which they call as CHESS. The principal goal of CHESS was to increase arithmetic computational density, to enhance the flexibility, and to increase the bandwidth and capacity of internal memories significantly beyond the capabilities of existing commercial FPGAs. These goals were achieved by proposing an array of ALUs with embedded RAMs where each ALU is 4-bit wide and supports 16 instructions. Similarly, authors in [42] present a coarse-grained, field programmable architecture for constructing deep computational pipelines. This architecture can efficiently implement applications related to media, signal processing, scientific computing and communications. Further, authors in [128] have used bus-based routing and logic blocks to improve density of FPGAs

for datapath circuits. This is a partial multi-bit FPGA architecture that is designed to exploit the regularity that most of the datapath circuits exhibit.

2.6.6 Time-Multiplexed FPGAs

Time-multiplexed FPGAs increase the capacity of FPGAs by executing different portions of a circuit in a time-multiplexed mode [89, 114]. An application design is divided into different sub-circuits, and each sub-circuit runs as an individual context of FPGA. The state information of each sub-circuit is saved in context registers before a new context runs on FPGA. Authors in [114] have proposed a time-multiplexed FPGA architecture where a large circuit is divided into sub-circuits and each sub-circuit is sequentially executed on a time-multiplexed FPGA. Such an FPGA stores a set of configuration bits for all contexts. A context is shifted simply by using the SRAM bits dedicated to a particular context. The combinatorial and sequential outputs of a sub-circuit that are required by other sub-circuits are saved in context registers which can be easily accessed by sub-circuits at different times.

Time-Multiplexed FPGAs increase their capacity by actually adding more SRAM bits rather than more CLBs. These FPGAs increase the logic capacity by dynamically reusing the hardware. The configuration bits of only the currently executing context are active, the configuration bits for the remaining supported contexts are inactive. Intermediate results are saved and then shared with the contexts still to be run. Each context takes a micro-cycle time to execute one context. The sum of the micro-cycles of all the contexts makes one user-cycle. The entire time-multiplexed FPGA or its smaller portion can be configured to (i) execute a single design, where each context runs a sub-design, (ii) execute multiple designs in time-multiplexed modes, or (iii) execute statically only one single design. Tabula [109] is a recently launched FPGA vendor that provides time-multiplexed FPGAs. It dynamically reconfigures logic, memory, and interconnect at multi-GHz rates with a Spacetime compiler.

2.6.7 Asynchronous FPGA Architecture

Another alternative approach that has been proposed to improve the overall performance of FPGA architecture is the use of asynchronous design elements. Conventionally, digital circuits are designed for synchronous operation and in turn FPGA architectures have focused primarily on implementing synchronous circuits. Asynchronous designs are proposed to improve the energy efficiency of asynchronous FPGAs since asynchronous designs offer potentially lower energy as energy is consumed only when necessary. Also the asynchronous architectures can simplify the design process as complex clock distribution networks become unnecessary.

The first asynchronous FPGA was developed by [57]. It consisted the modified version of previously developed synchronous FPGA architecture. Its logic block was

similar to the conventional logic block with added features of fast feedback and a latch that could be used to initialize an asynchronous circuit. Another asynchronous architecture was proposed in [112]. This architecture is designed specifically for dataflow applications. Its logic block is similar to that of synchronous architecture, along with it consists of units such as split unit which enables conditional forwarding of data and a merge unit that allows for conditional selection of data from different sources. An alternative to fully asynchronous design is a globally asynchronous, locally synchronous approach (GALS). This approach is used by [69] where authors have introduced a level of hierarchy into the FPGA architecture. Standard hard or soft synchronous logic blocks are grouped together to form large synchronous blocks and communication between these blocks is done asynchronously. More recently, authors in [131] have applied the GALS approach on Network on Chip architectures to improve the performance, energy consumption and the yield of future architectures in a synergistic manner.

It is clear that, despite each architecture offering its own benefits, a number of architectural questions remain unresolved for asynchronous FPGAs. Many architectures rely on logic blocks similar to those used for synchronous designs [57, 69] and, therefore, the same architectural issues such as LUT size, cluster size, and routing topology must be investigated. In addition to those questions, asynchronous FPGAs also add the challenge of determining the appropriate synchronization methodology.

2.7 Summary and Conclusion

In this chapter initially a brief introduction of traditional logic and routing architectures of FPGAs is presented. Later, different steps involved in the FPGA design flow are detailed. Finally various approaches that have been employed to reduce few disadvantages of FPGAs and ASICs, with or without compromising their major benefits are described. Figure 2.28 presents a rough comparison of different solutions used to reduce the drawbacks of FPGAs and ASICs. The remaining chapters of this book will focus on the exploration of tree-based FPGA architectures using hard-blocks, tree-based application specific Inflexible FPGAs (ASIF), and their automatic layout generation methods.

This book presents new environment for the exploration of tree-based heterogeneous FPGAs. This environment is used to explore different architecture techniques for tree-based heterogeneous FPGA architecture. This book also presents an optimized environment for mesh-based heterogeneous FPGA. Further, the environments of two architectures are evaluated through the experimental results that are obtained by mapping a number of heterogeneous benchmarks on the two architectures.

Altera [11] has proposed a new idea to prototype, test, and even ship initial few designs on an FPGA, later the FPGA based design can be migrated to Structured-ASIC (known as HardCopy). However, migration of an FPGA-based product to Structured-ASIC supports only a single application design. An ASIF retains this

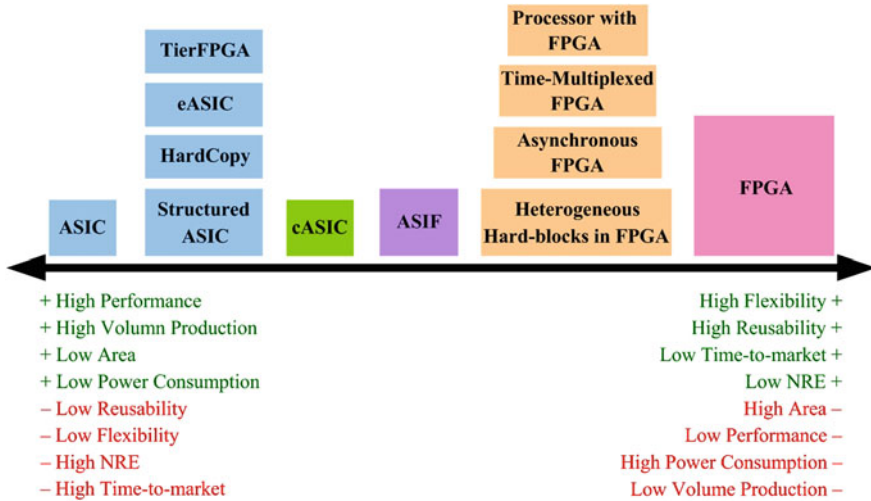


Fig. 2.28 Comparison of different solutions used to reduce ASIC and FPGA drawbacks

property, and can be a possible future extension for the migration of FPGA-based applications to Structured-ASIC. Thus when an FPGA-based product is in the final phase of its development cycle, and if the set of circuits to be mapped on the FPGA are known, the FPGA can be reduced to an ASIF for the given set of application designs. This book presents a new tree-based ASIF and a detailed comparison of tree-based ASIF is performed with mesh-based ASIF. This book also presents automatic layout generation techniques for domain-specific FPGA and ASIF architectures.

Altera FLEX 8000 Block Diagram

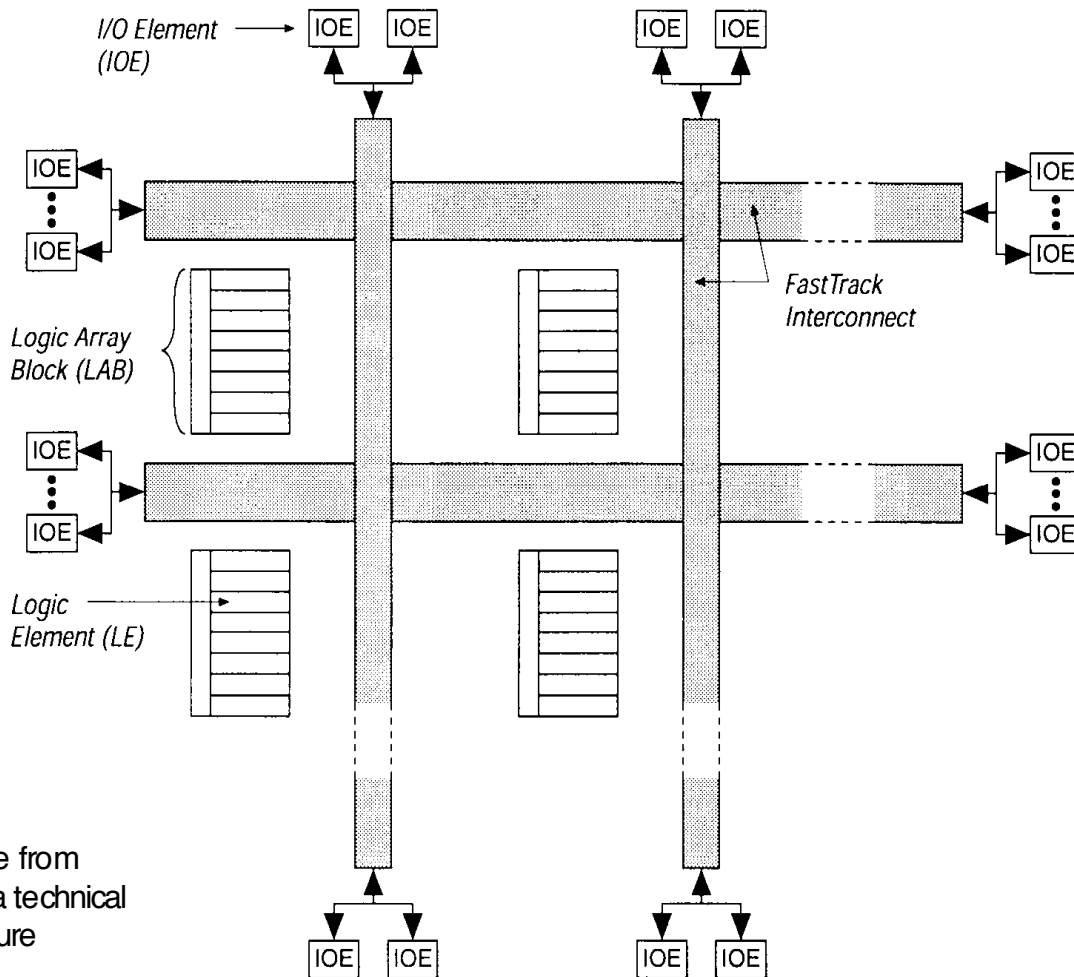


Figure from
Altera technical
literature

- FLEX 8000 chip contains 26–162 LABs
 - Each LAB contains 8 Logic Elements (LEs), so a chip contains 208–1296 LEs, totaling 2,500–16,000 usable gates
 - LABs arranged in rows and columns, connected by FastTrack Interconnect, with I/O elements (IOEs) at the edges

Altera FLEX 8000 Logic Array Block

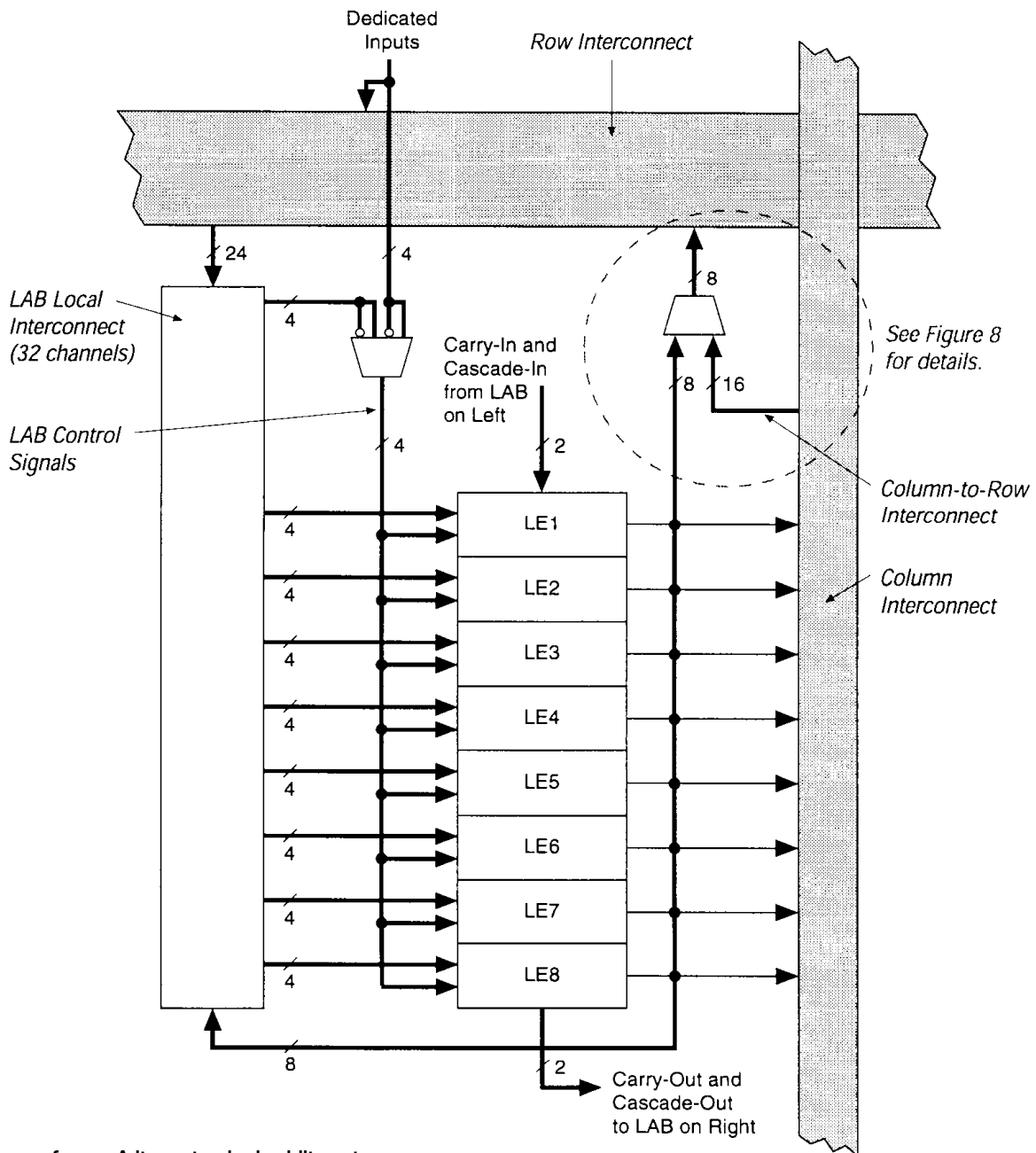


Figure from Altera technical literature

- LAB = 8 LEs, plus local interconnect, control signals, carry & cascade chains

Altera FLEX 8000 Logic Element

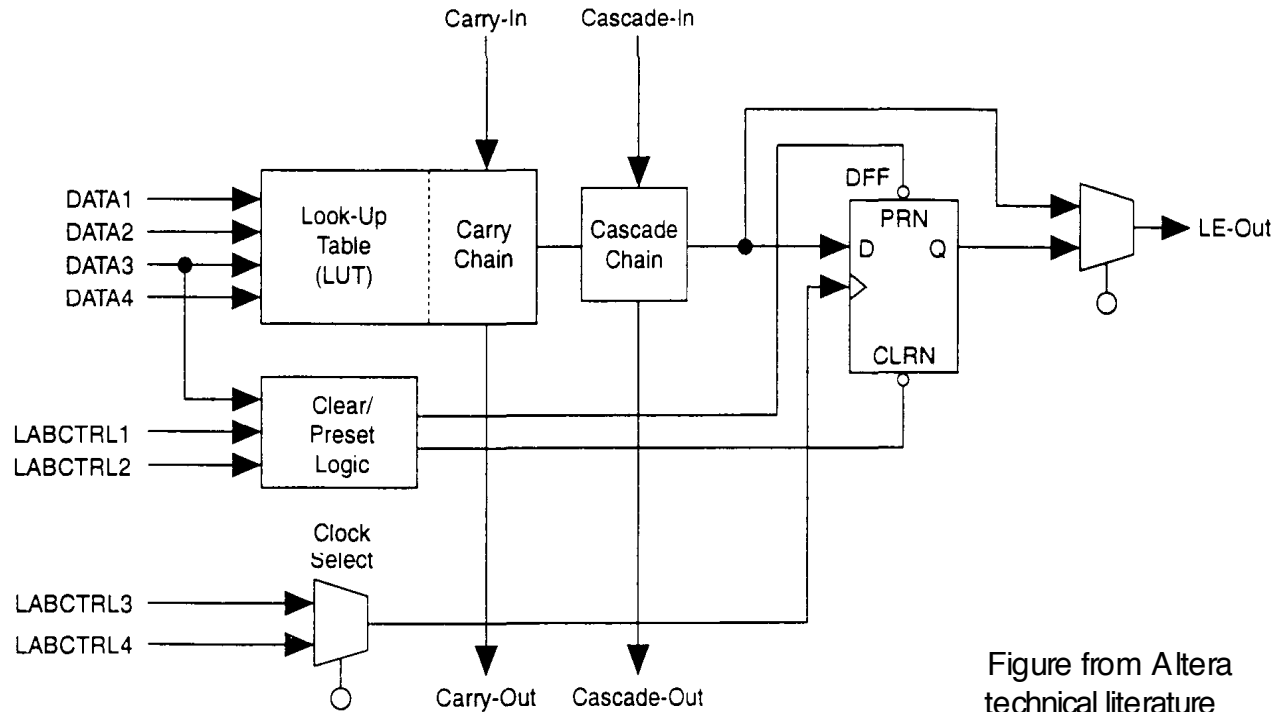


Figure from Altera technical literature

- Each Logic Element (LE) contains:
 - 4-input Look-Up Table (LUT)
 - Can produce any function of 4 variables
 - Programmable flip-flop
 - Can configure as D, T, JR, SR, or bypass
 - Has clock, clear, and preset signals that can come from dedicated inputs, I/O pins, or other LEs
 - Carry chain & cascade chain

Altera FLEX 8000 Carry Chain (Example: n-bit adder)

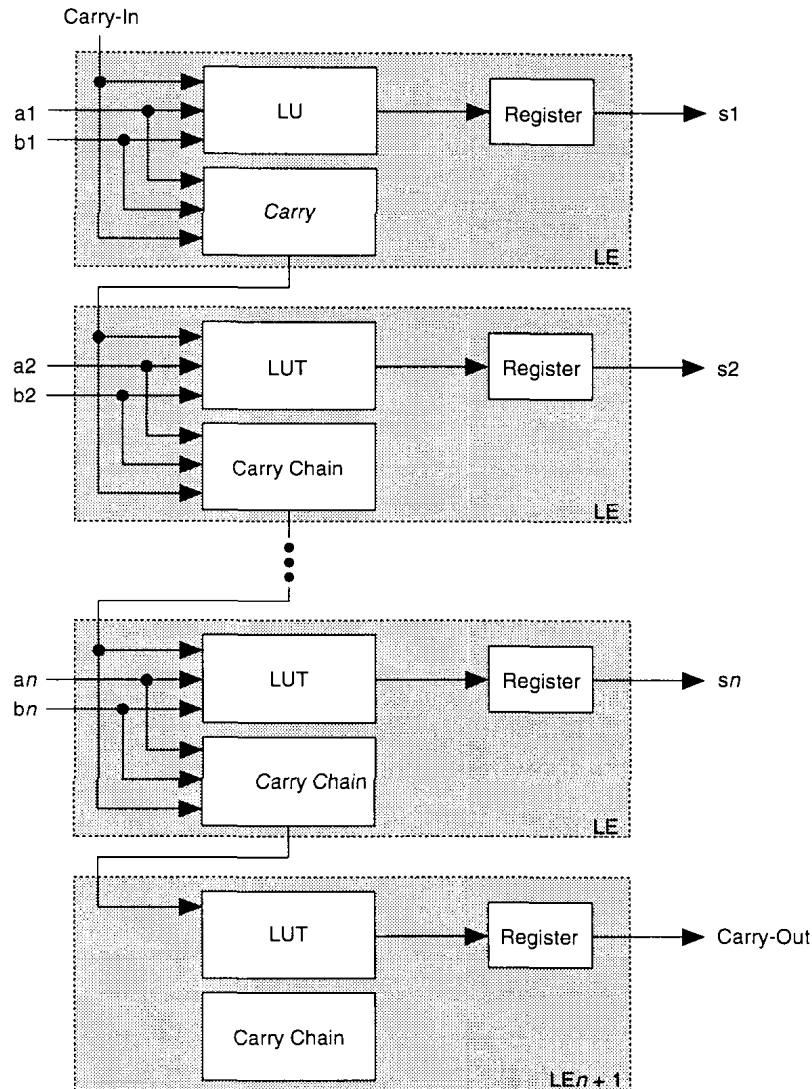
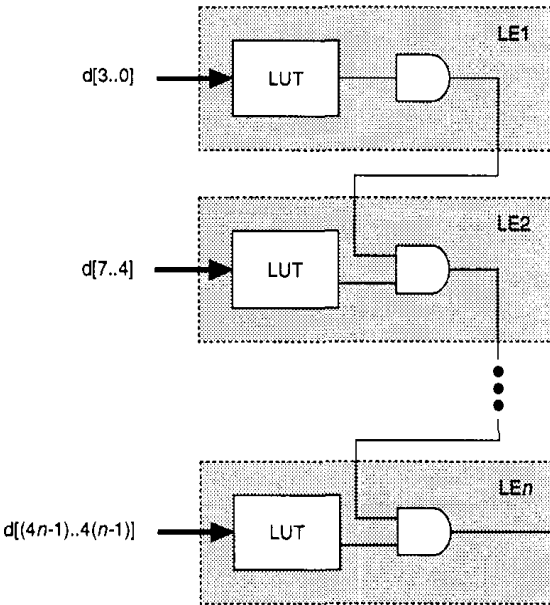


Figure from Altera technical literature

- *Carry chain* provides very fast ($< 1\text{ns}$) carry-forward between LEs
 - Feeds both LUT and next part of chain
 - Good for high-speed adders & counters

Altera FLEX 8000 Cascade Chain

AND Cascade Chain



OR Cascade Chain

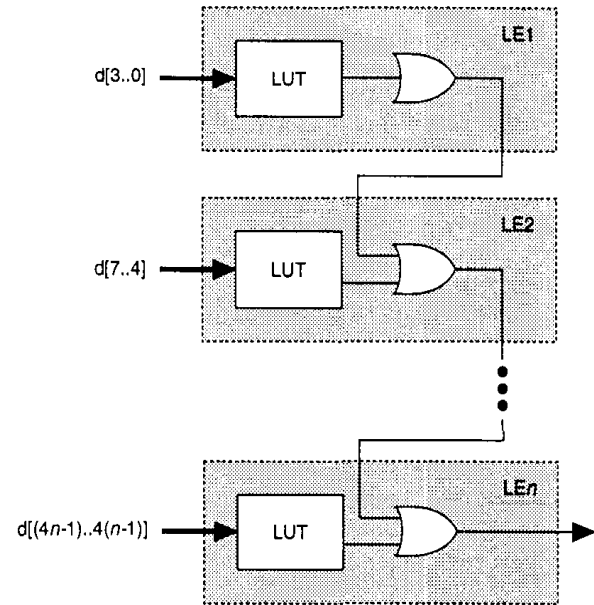


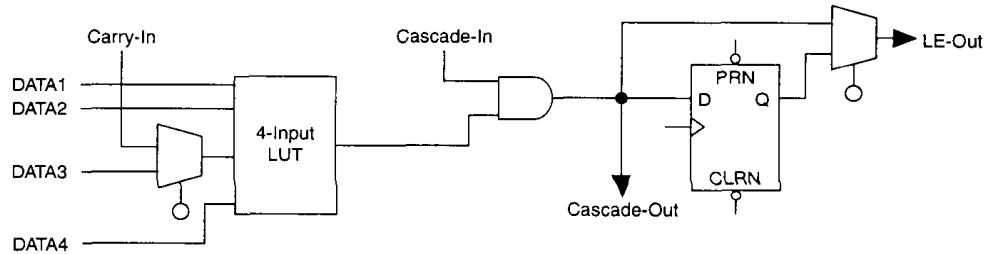
Figure from Altera technical literature

■ *Cascade chain* provides wide fan-in

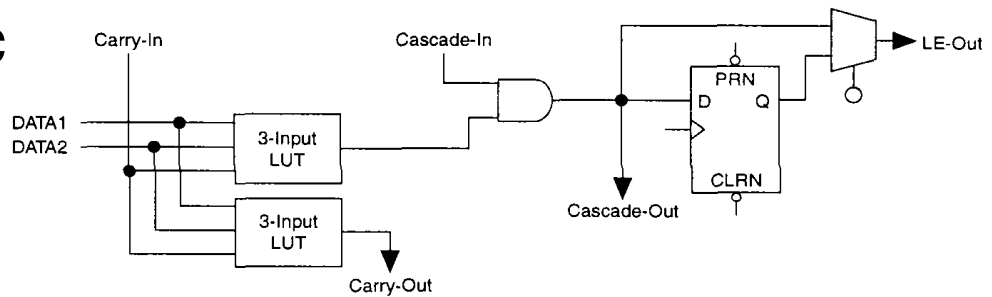
- Adjacent LE's LUTs can compute parts of the function in parallel; cascade chain then serially connects intermediate values
- Can use either a logical AND or a logical OR (using DeMorgan's theorem) to connect outputs of adjacent LEs
- Each additional LE provides 4 more inputs to the width of the function

Altera FLEX 8000 LE Operating Modes

Normal

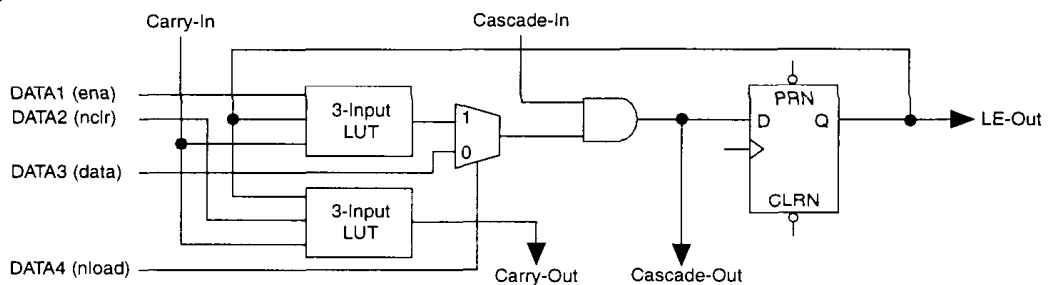


Arithmetic



Up/down Counter

Figure from Altera technical literature



■ Each mode uses LE resources differently

- 7 out of 10 inputs (4 data from LAB local interconnect, feedback from register, and carry-in & cascade-in) go to specific destinations to implement the function
- Remaining 3 provide clock, clear, and preset for register

Altera FLEX 8000 Operating Modes (cont.)

- Normal mode
 - Used for general logic applications, and wide decoding functions that can benefit from the cascade chain

- Arithmetic mode
 - Provides two 3-input LUTs to implement adders, accumulators, and comparators
 - One LUT provides a 3-bit function
 - Other LUT generates a carry bit

- Up/down counter mode
 - Provides counter enable, synchronous up / down control, and data loading options

 - Uses two 3-input LUTs
 - One LUT generates counter data
 - Other LUT generates fast carry bit
 - Use 2-to-1 multiplexer for synchronous data loading, clear and preset for

Altera FLEX 8000 FastTrack Interconnect

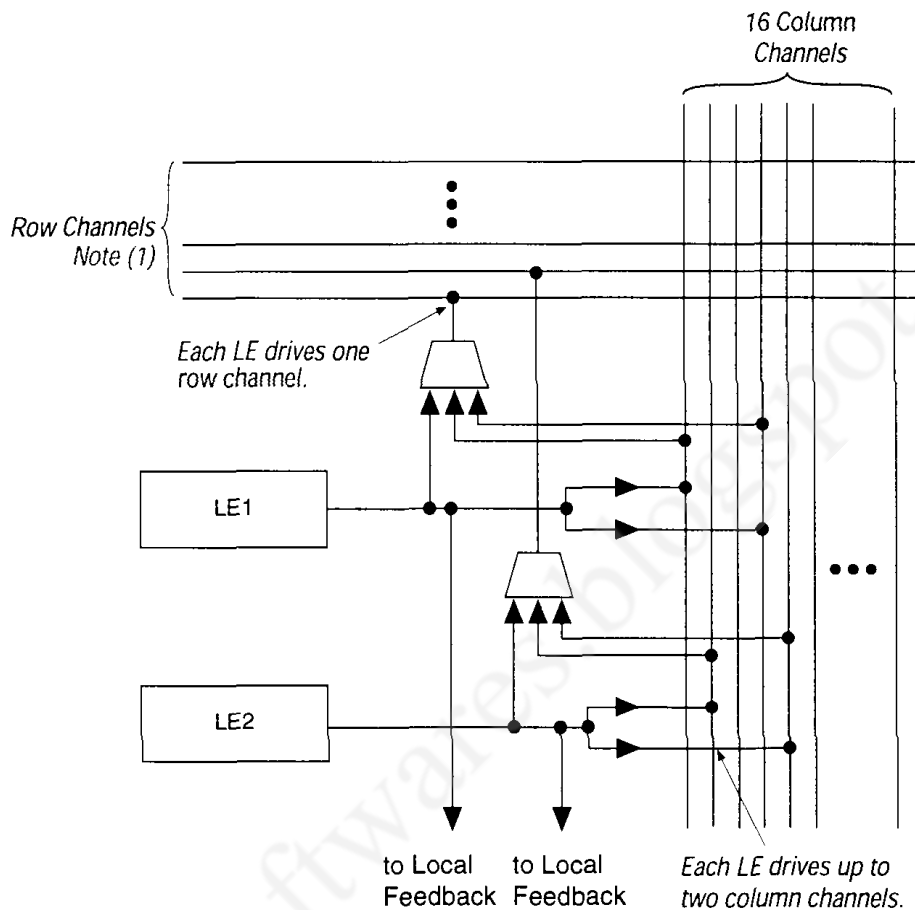


Figure from Altera technical literature

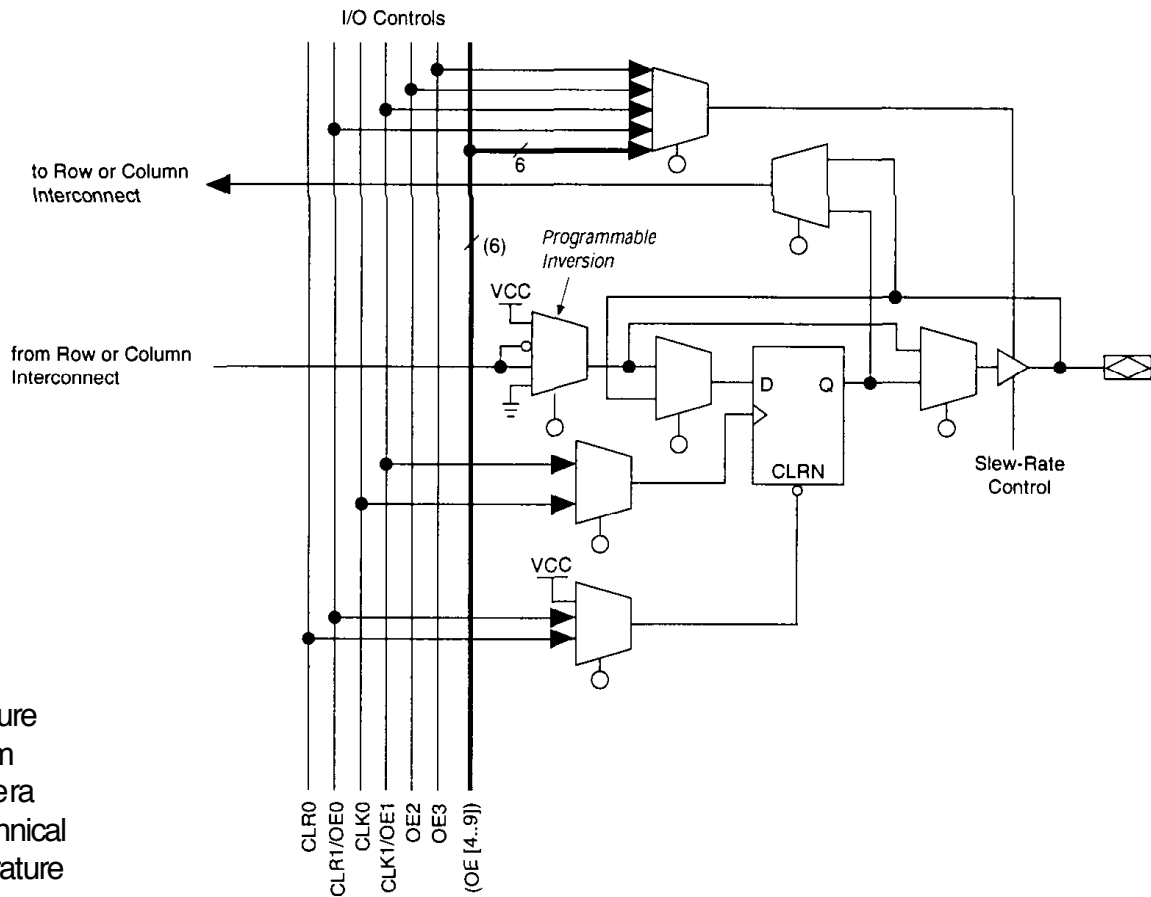
Note:

(1) See Table 4 for the number of row channels.

■ Device-wide rows and columns

- Each LE in LAB drives 2 column (total 16) channels, which connects... that column
- Each LE in LAB drives 1 row channel, which connects to other LABs in that row
 - 3-to-1 muxs connect either LE outputs or column channels to row channels

Altera FLEX 8000 I/O Elements



- Eight I/O Elements (IOEs) are at the end of each row and column
 - Some restrictions on how many row / column channels each IOE connects to
 - Contains a register that can be used for either input or output
 - Associated I/O pins can be used as either input, output, or bidirectional pins

Altera FLEX 8000 Configuration

- Loading the FLEX 8000's SRAM with programming information is called *configuration*, and takes about 100ms
 - After configuration, the device initializes itself (resets its registers, enables its I/O pins, and begins normal operation)
 - Configuration & initialization = command mode, normal operation = user mode
- Six configuration schemes are available:
 - Active serial — FLEX gives configuration EPROM clock signals (not addresses), keeps getting new values in sequence
 - Active parallel up, active parallel down — FLEX 8000 gives configuration EPROM sequence of addresses to read data from
 - Passive parallel synchronous, passive parallel asynchronous, passive serial — passively receives data from some host

Altera FLEX 8000 Block Diagram (Review)

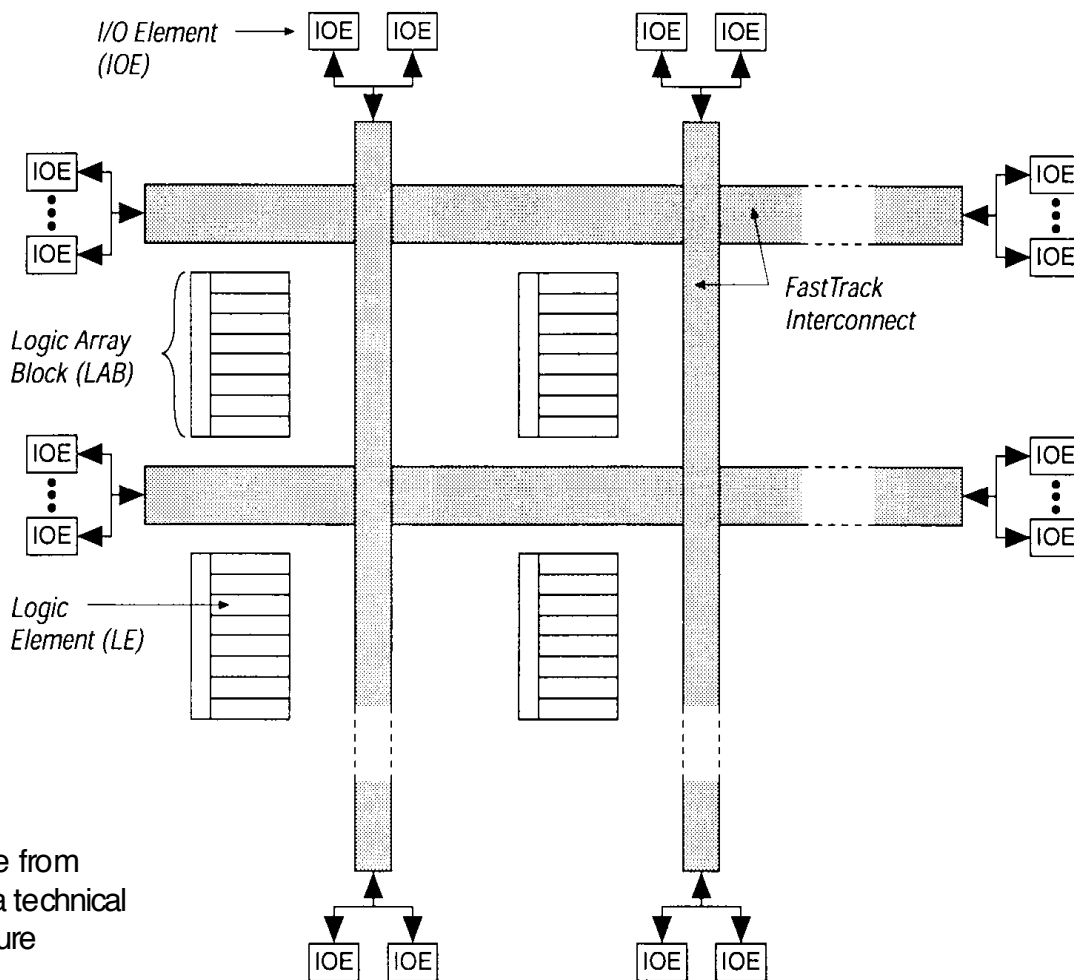


Figure from
Altera technical
literature

- FLEX 8000 chip contains 26–162 LABs
 - Each LAB contains 8 Logic Elements (LEs), so a chip contains 208–1296 LEs, totaling 2,500–16,000 usable gates
 - LABs arranged in rows and columns, connected by FastTrack Interconnect, with I/O elements (IOEs) at the edges

Altera FLEX 10K Block Diagram

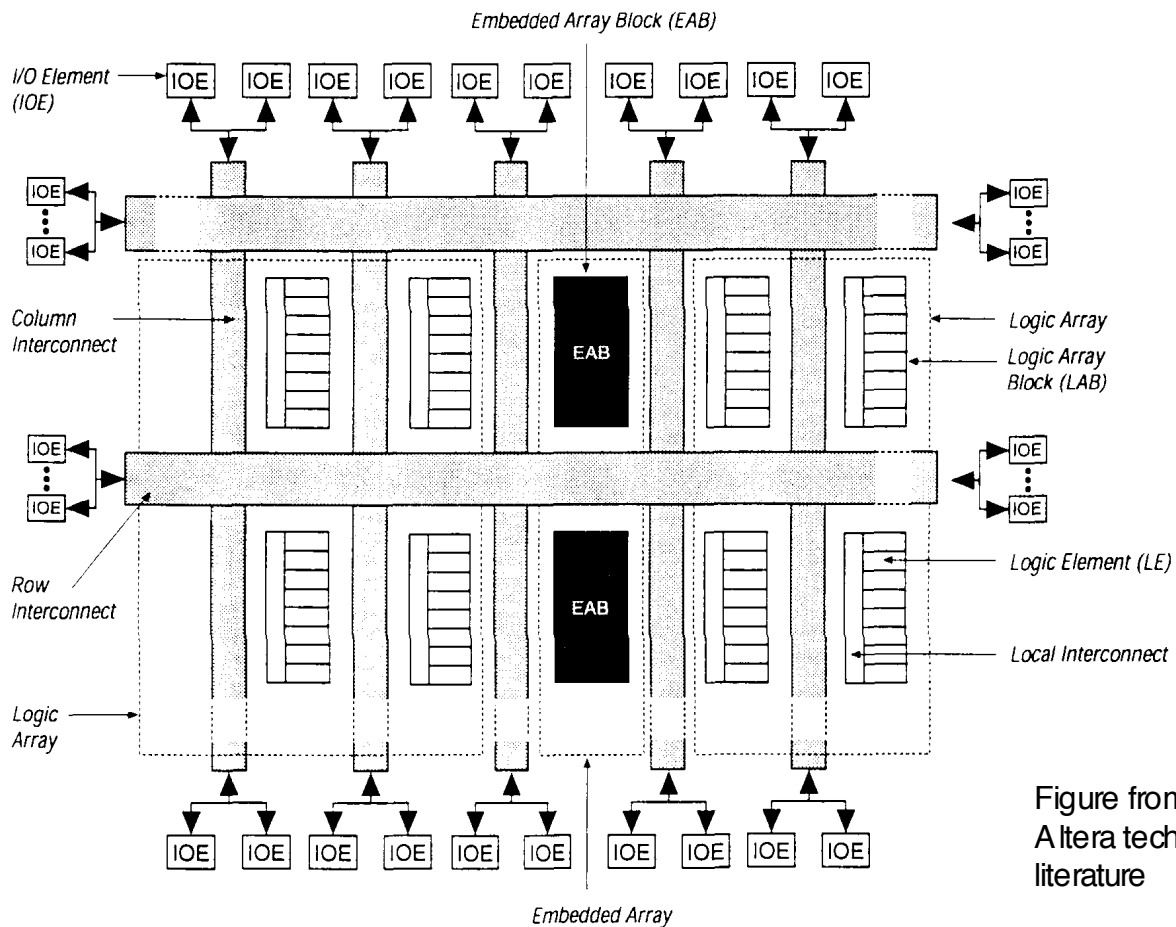


Figure from Altera technical literature

- FLEX 10K chip contains 72–1520 LABs

- Each LAB contains 8 Logic Elements (LEs), so a chip contains 576–12,160 LEs, totaling 10,000–250,000 usable gates

- Each chip also contains 3–20 Embedded Array Blocks (EABs), which can provide 6,164–40,960 bits of RAM

Altera FLEX 10K

Embedded Array Blocks (EABs)

- Each chip contains 3–20 EABs, each of which can be used to implement either logic or memory
- When used to implement logic, an EAB can provide 100 to 600 gate equivalents (in contrast, a LAB provides 96 g.e.'s)
 - Provides a very large LUT
 - Very fast — faster than general logic, since it's only a single level of logic
 - Delay is predictable — each RAM block is not scattered throughout the chip as in some FPGAs
 - Can be used to create complex logic functions such as multipliers (e.g., a 4x4 multiplier with 8 inputs and 8 outputs), microcontrollers, large state machines, and DSPs
 - Each EAB can be used independently, or combined to implement larger functions

Altera FLEX 10K

Embedded Array Blocks (cont.)

- Using EABs to implement memory, a chip can have 6K–40K bits of RAM
 - Each EAB provides 2,048 bits of RAM, plus input and output registers
 - Can be used to implement synchronous RAM, ROM, dual-port RAM, or FIFO
 - Each EAB can be configured in the following sizes:
 - 256x8, 512x4, 1024x2, or 2048x1
 - To get larger blocks, combine multiple EABs:
 - Example: combine two 256x8 RAM blocks to form a 256x16 RAM block
 - Example: combine two 512x4 RAM blocks to form a 512x8 RAM block
 - Can even combine all EABs on the chip into one big RAM block
 - Can combine so as to form blocks up to 2048 words without impacting timing

Altera FLEX 10K Embedded Array Blocks (cont.)

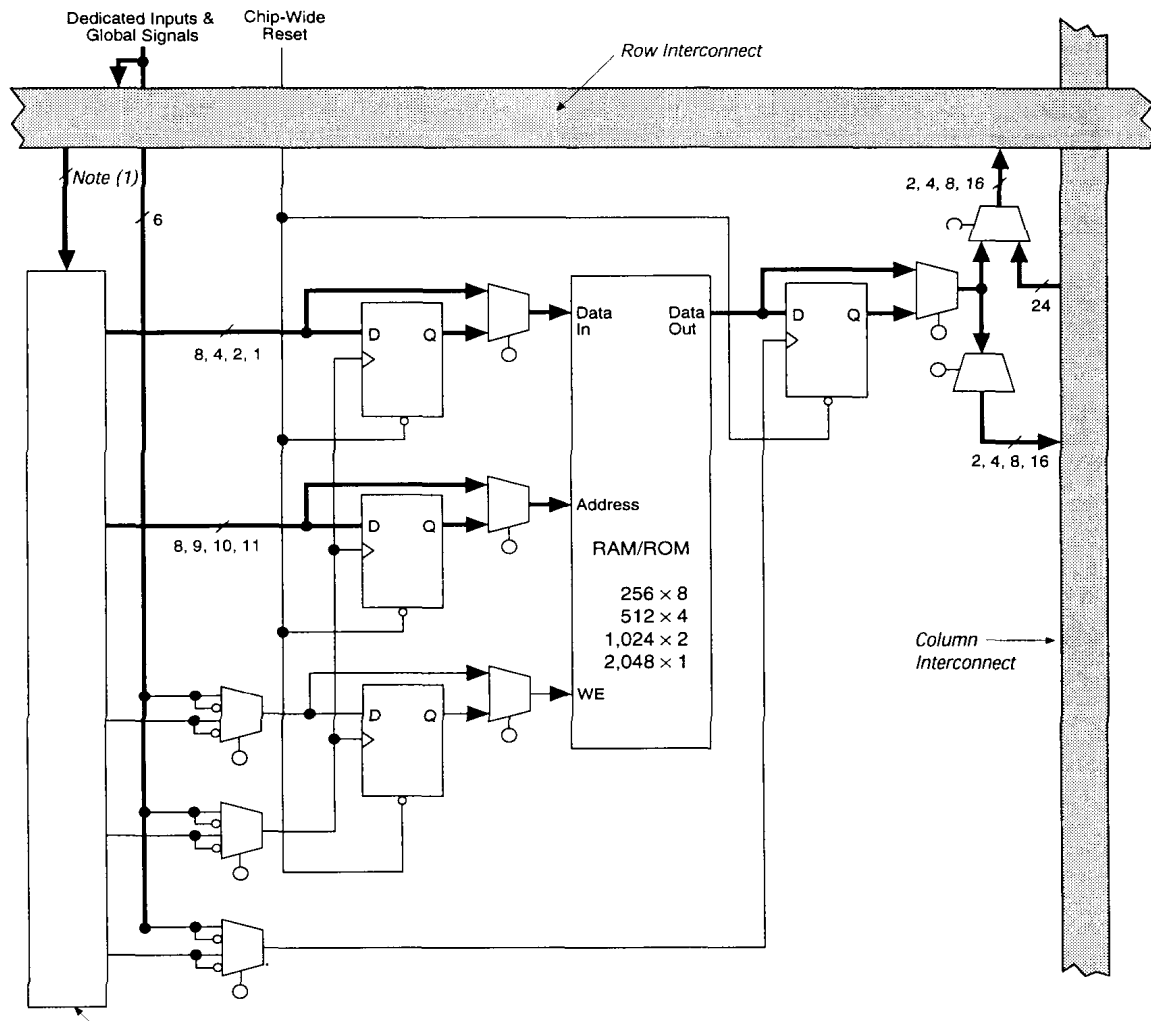


Figure from Altera technical literature

- EAB gets input from a row channel, and can output to up to 2 row channels and 2 column channels
- Input and output buffers are available

Altera APEX 20K Overview

- APEX 20K chip contains:
 - 256–3,456 LABs, each of which contains 10 Logic Elements (LEs), so a chip contains 2,560–51,840 Les, 162,000–2,391,552 usable gates
 - 16–216 Embedded System Blocks (EABs), each of which can provide 32,768–442,368 bits of memory
 - Can implement CAM, RAM, dual-port RAM, ROM, and FIFO

- Organization:
 - MultiCore architecture, combining LUT, product-terms, & memory in one structure
 - Designed for “system on a chip”
 - MegaLAB structures, each of which contains 16 LABs, one ESB, and a MegaLAB interconnect (for routing within the MegaLAB)
 - ESB provides product terms or memory

APEX LABs and Interconnect

- Logic Array Block (LAB)
 - 10 LEs
 - Interleaved local interconnect (each LE connects to 2 local interconnect, each local interconnect connects to 10 LEs)
 - Each LE can connect to 29 other LEs through local interconnect
- Logic Element (LE)
 - 4-input LUT, carry chain, cascade chain, same as FLEX devices
 - Synchronous and asynchronous load and clear logic
- Interconnect
 - MegaLAB interconnect between 16 LABs, etc. inside each MegaLAB
 - FastTrack row and column interconnect between MegaLABs

APEX Embedded System Blocks (ESBs)

- Each ESB can act as a macrocell and provide product terms
 - Each ESB gets 32 inputs from local interconnect, from adjacent LAB or MegaLAB interconnect
 - In this mode, each ESB contains 16 macrocells, and each macrocell contains 2 product terms and a programmable register (parallel expanders also provided)

- Each ESB can also act as a memory block (dual-port RAM, ROM, FIFO, or CAM memory) configured in various sizes
 - Inputs from adjacent local interconnect, which can be driven from MegaLAB or FastTrack interconnect
 - Outputs to MegaLAB and FastTrack, some outputs to local interconnect