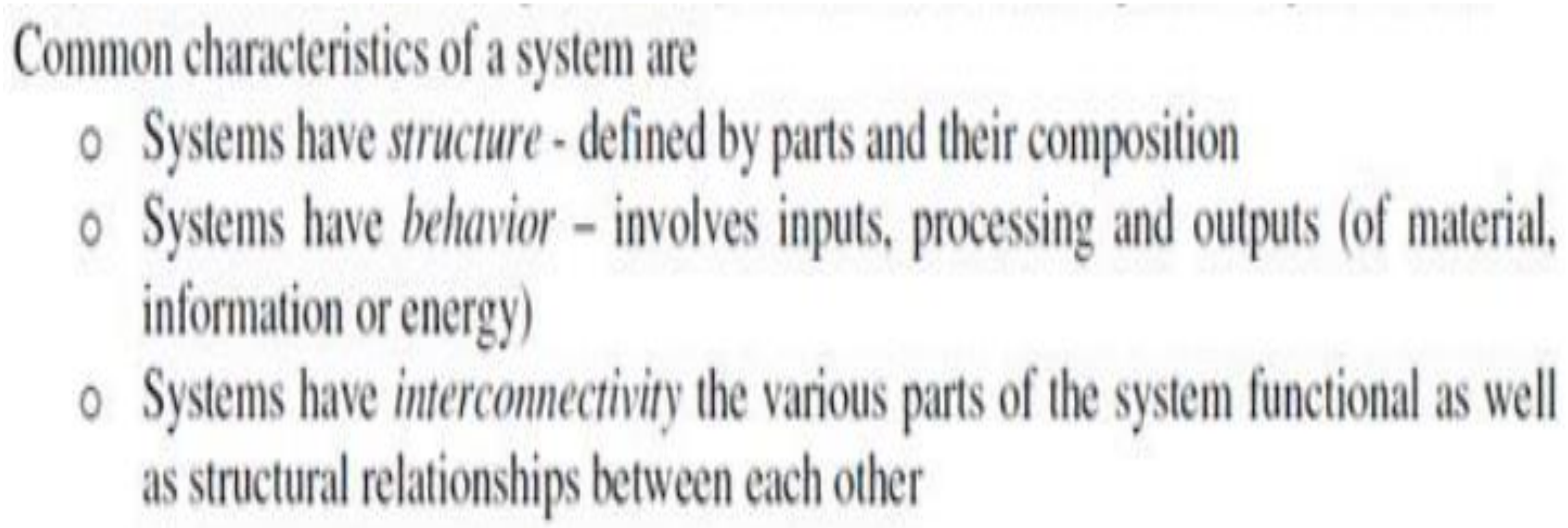
ACS College of Engineering
Approved by AICTE New Delhi, Affiliated to VTU, Belagavi
(A Unit of RajaRajeswari Group of Institutions)

NAAC 'A' Accredited

MODULE -4
Sub system design and PLA
ACSCE

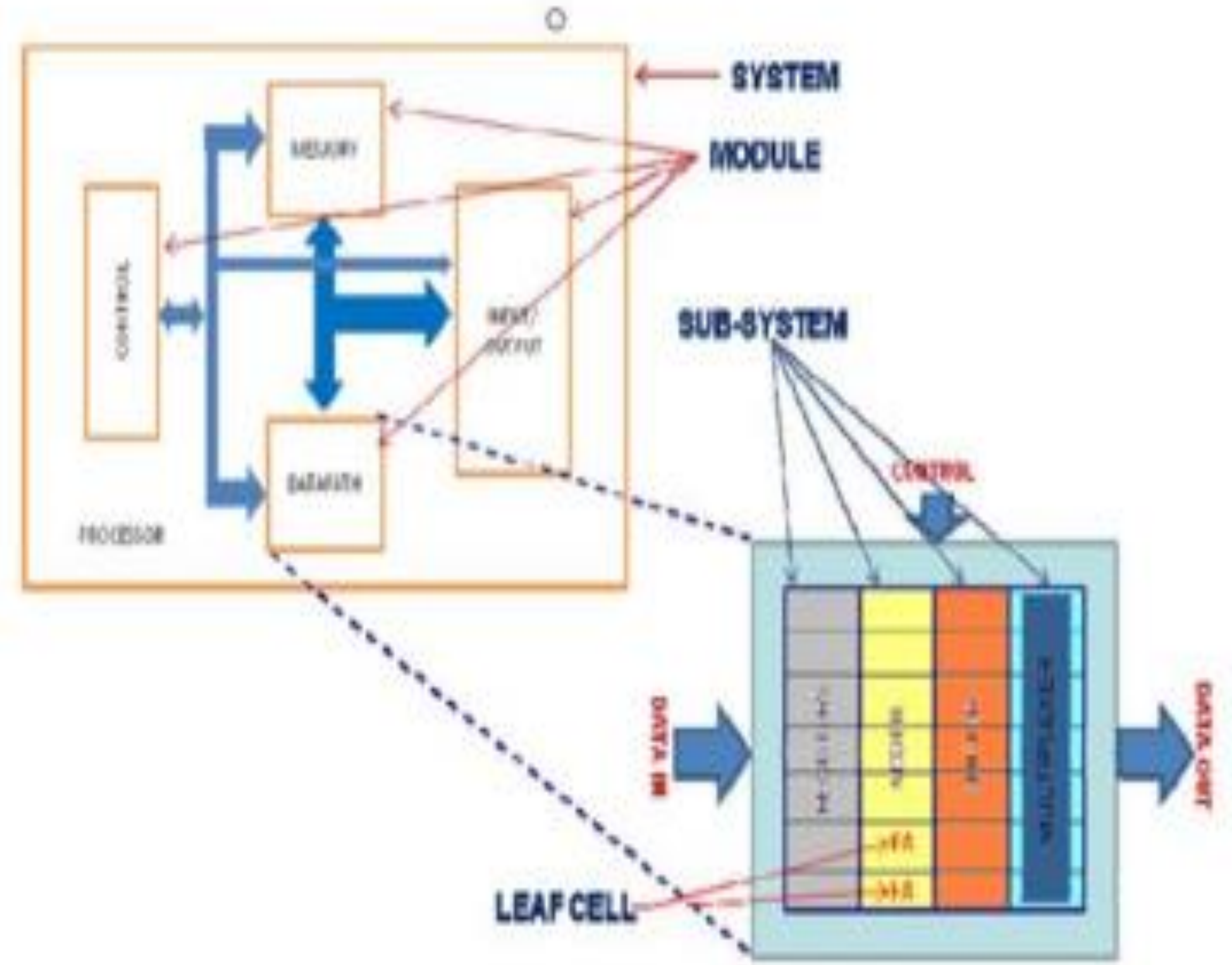
SYSTEM and SUB SYSTEM

- System is a set of interdependent entities/modules /leaf cell which interact among themselves to form a complete unit.
- Large systems are composed of subsystems known as leaf cells.
- Most basic leaf cells are inverter, NAND,CMOS ….


Common characteristics of a system are

- Systems have *structure* - defined by parts and their composition
- Systems have *behavior* – involves inputs, processing and outputs (of material, information or energy)
- Systems have *interconnectivity* the various parts of the system functional as well as structural relationships between each other

A system


SYSTEM
MODULE
MEMORY
CONTROL
INPUT/OUTPUT
DATAPATH
PROCESSOR
SUB-SYSTEM
CONTROL
DATA IN
DATA OUT
ADDER
LEAF CELL

ARCHITECTURAL ISSUES OF SUBSYSTEM DESIGN

- In any VLSI system, logical and systematic approach is essential.

  For ex:

  In designing an MSI logic circuit comprising of

  a) 500 transistors - 2 engineer-months are required, similarly in designing
  b) 500,000 transistor - 2,000 engineer-months or 170-engineers - years are required.

- Thus as **complexity increases**, **design time** exponentially **increases.**

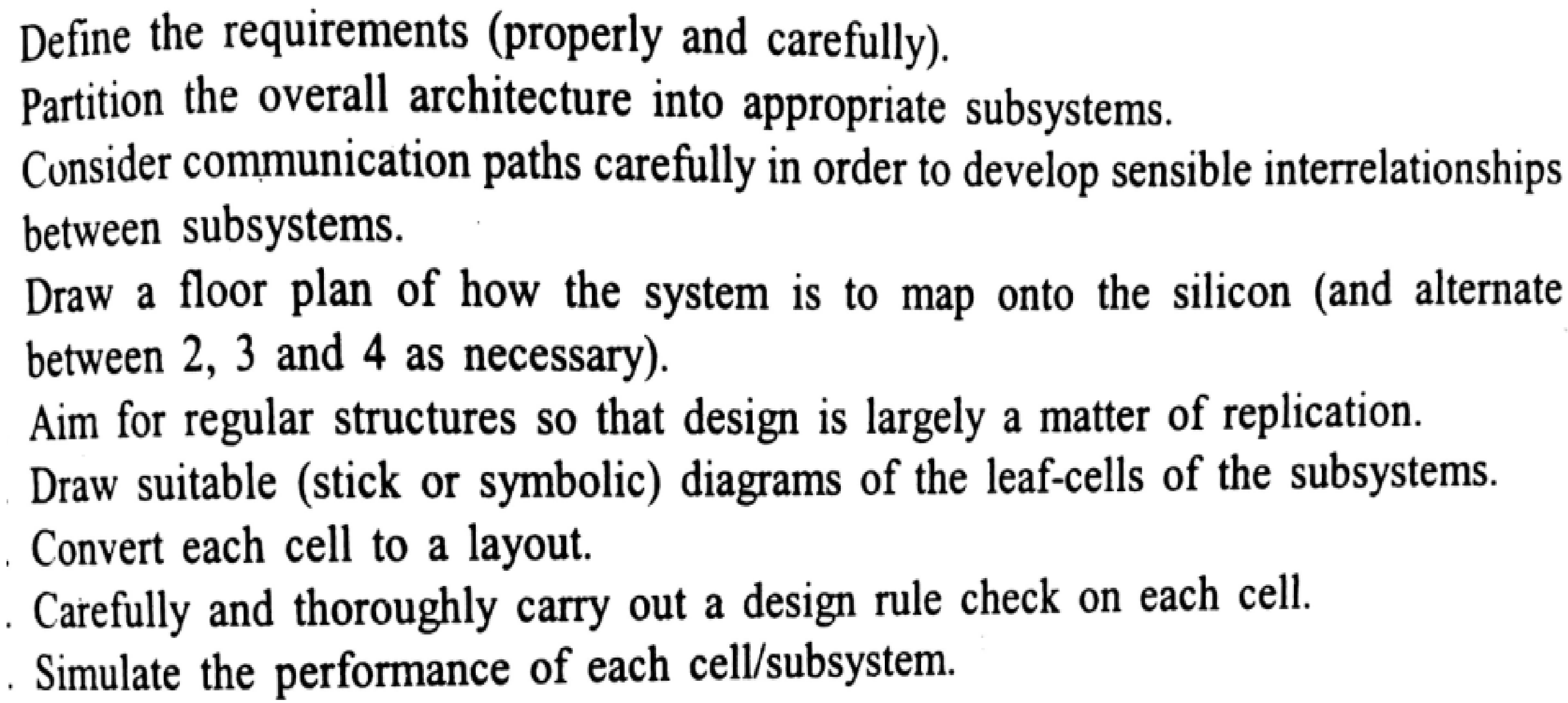
  Therefore,

  we must **adopt design methods** which handle **complexity**

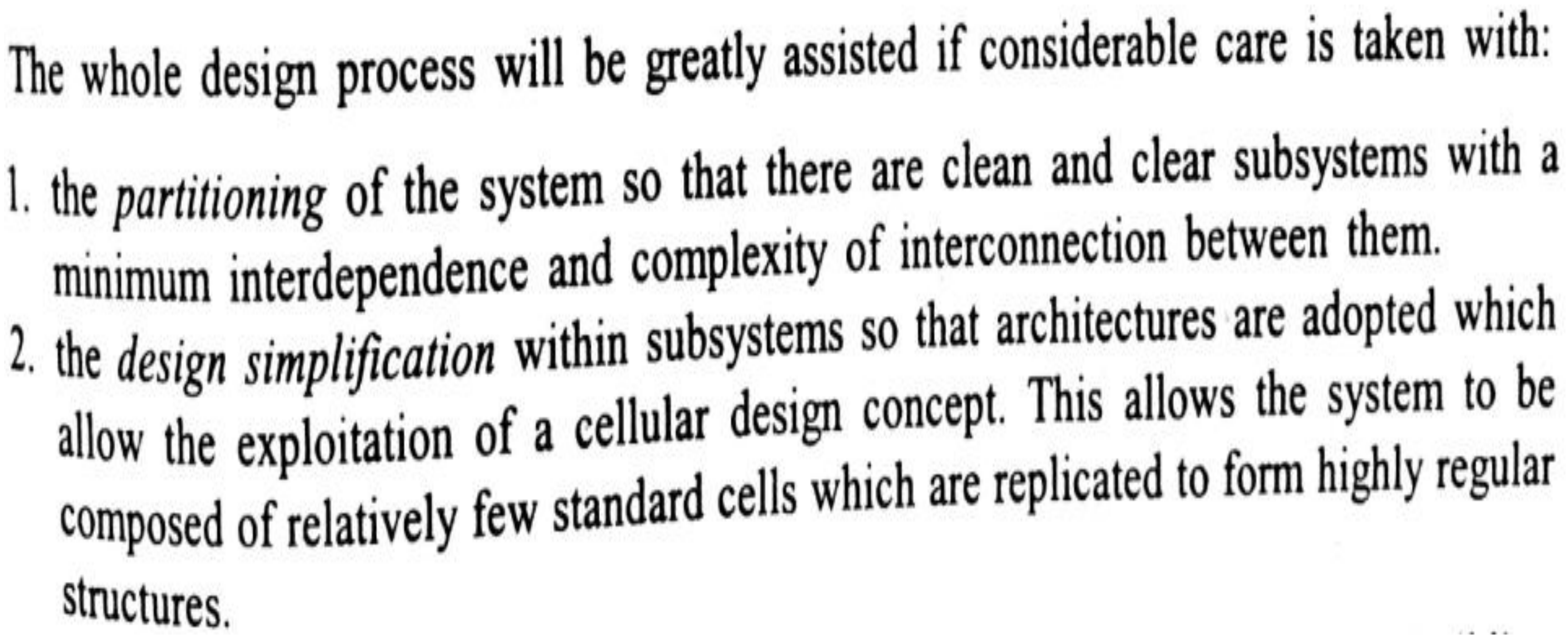
  with a less time and

  with less labor.

GUIDELINES TO BE FOLLOWED AT THE SUBSYSTEM OR LEAF-CELL



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


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

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

SWITCH LOGIC and **GATE(RESTORING LOGIC)** are the 2 basic ways of build logic circuits in digital systems in MOS technology.

SWITCH LOGIC: It is based on "Pass Transistor" or "Transmission Gate".

Pass transistor

- Buffer used in MSI = PT used in VLSI.
- One form of logic that is popular in nMOs rich ckt is PTLogic.
- It occupies minimum space.
- One of the alternative for conventional logic
- PT Logic will not allow us to have a direct path between Vcc and GND . Here ,current only flows on switching. Therefore, power dissipation is small.
- Minimum size of PT is 2λ x 2λ.
- PT can be used as either Pull Up Transistor or Pull down Transistor.

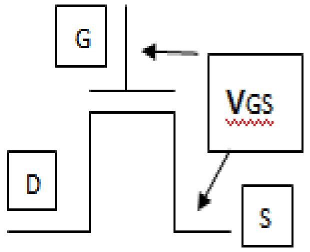
Pass Transistor

- Transistors can be thought as a switch controlled by its gate signal.
- PT are single FET's that pass signal b/w drain to source, instead of fixed power supply value.
- It cannot pass entire Voltage range.
- PT are used in designing regular arrays such as ROM, PLA's and Multiplexers.

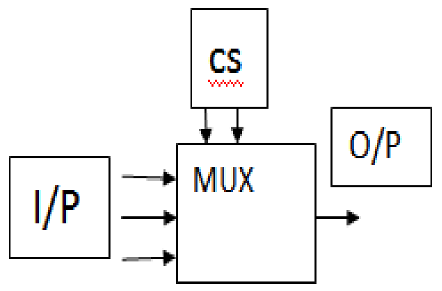
➢ $V_O=V_{DD}-V_T$ (nMOS Transistor).

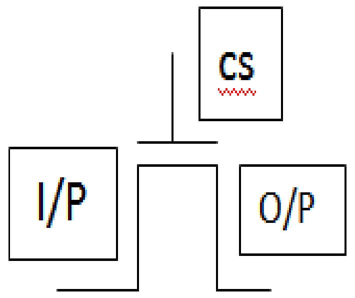
➢ $V_O=V_{DD}+V_T$ (pMOS Transistor).

➢ It is similar to model shown below:



MOSFET AS PASS TRANSISTOR





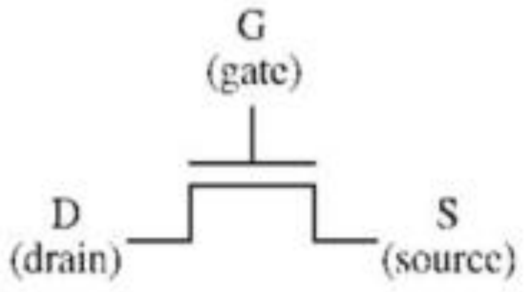
Depending on Control Signal at gate, Vin is passed to output. A transistor connected in this manner is called Pass transistor.

Advantages of PT:

occupies less area than logic gateS
3 Terminal device
require no DC power
DO not have direct short b/w Vcc and gnd.

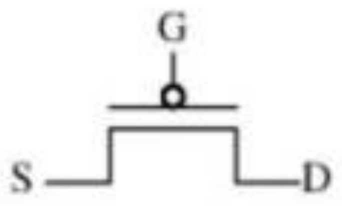
MOSFET as SWITCH

MOSFET as a Switch



*n*MOS transistor:
Closed (conducting) when
Gate = 1 (Vdd, 5V)

Open (non-conducting) when
Gate = 0 (ground, 0V)



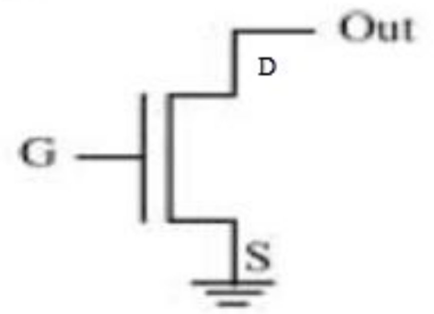
*p*MOS transistor:
Closed (conducting) when
Gate = 0 (ground, 0V)

Open (non-conducting) when
Gate = 1 (Vdd, 5V)

- We can view MOS transistors as electrically controlled switches
- Voltage at gate controls path from source to drain

MOSFET as Pull up and Pull down transistor

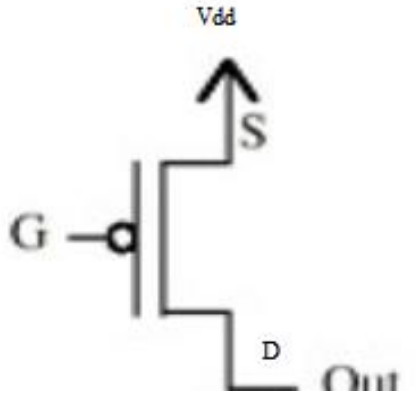
For *n*MOS switch, source is typically tied to ground and is used to *pull-down* signals:



when Gate = 1, Out = 0, (OV)

when Gate = 0, Out = Z (high impedance)

For *p*MOS switch, source is typically tied to Vdd, used to *pull* signals *up*:

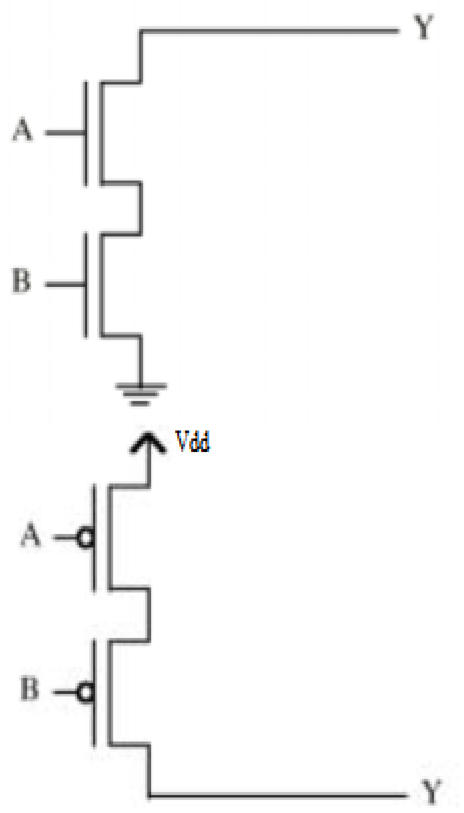


when Gate = 0, Out = 1 (Vdd)

when Gate = 1, Out = Z (high impedance)

SERIES AND PARALLEL CONNECTION OF SWITCHES.

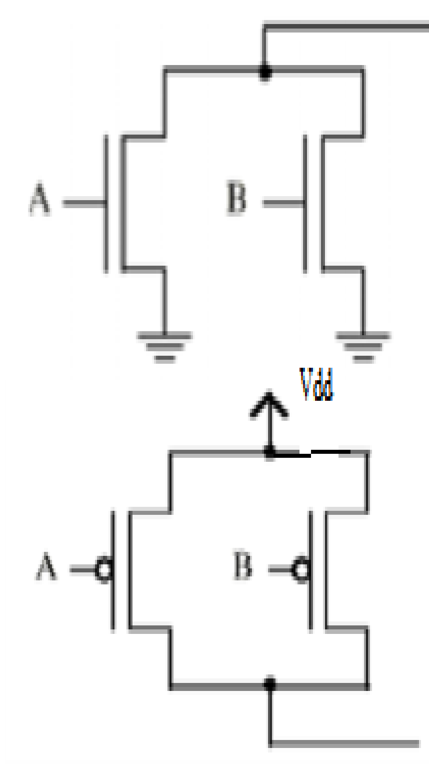
Series connection of Switches..



$Y = 0$, if $\underbrace{A \text{ and } B = 1}_{A \cdot B}$

$Y = 1$, if $\underbrace{A \text{ and } B = 0}_{\overline{A} \cdot \overline{B}}$

Parallel connection of Switches..

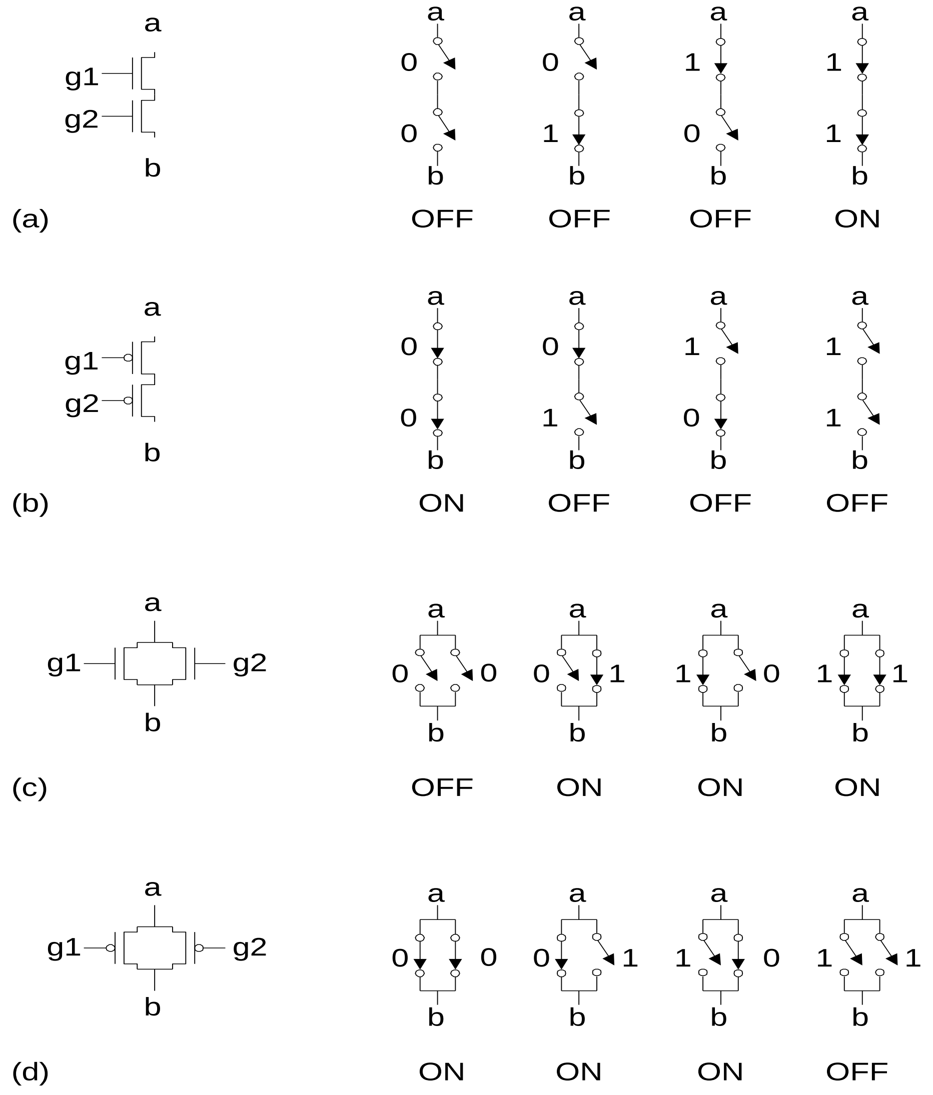


$Y = 0$, if $\underbrace{A \text{ or } B = 1}_{A + B}$

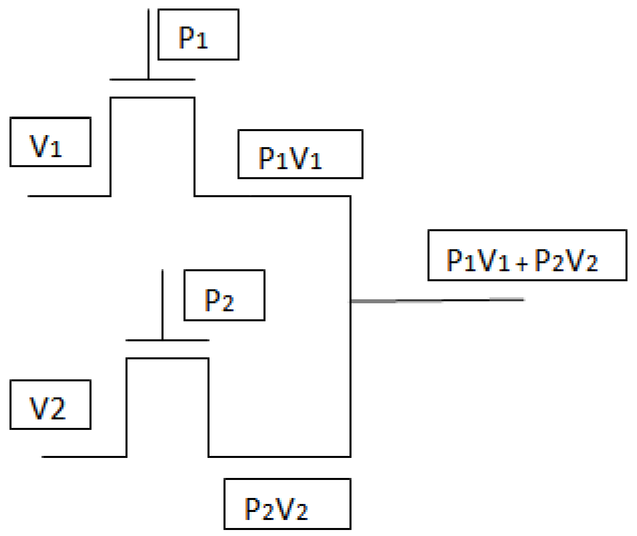
$Y = 1$ if $\underbrace{A \text{ or } B = 0}_{\overline{A} + \overline{B}}$

Series and Parallel Switches

- nMOS: 1 = ON
- pMOS: 0 = ON
- *Series*: both must be ON
- *Parallel*: either can be ON



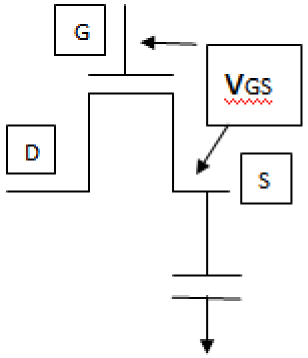
MODEL FOR Pass Transistor logic



More examples from Book –I of VLSI - HBB NOTES.
PAGE NO 25 TO 27

PT as AND gate, XNOR,CMOS XNOR,CMOS 4 way selector
A general 2 variable functional block, 4:1 Mux, 2 way selector, 2:1 MUX,

Signal degradation in nMOS tr



- Tr here behaves like a 'R' and 'C'.
- When G=1, Vin = V_d = V_s = V_o. The 'C' charges and V_o increases.
- This reduces V_{gs}.
- When Vgs becomes < V_{tn},Vgs < Vtn,tr= off=C.O
- Tr enters Cutoff and C cannot charge.
- Maximum output voltage ,
- V_o= V_{GG} –V_{tn} = 5-1.5 = 3.5Volts approximately.
- **Disadvantage:**
- **o/p of nMos tr is always < 5V.**
- **i.e logic 1 level in nMOs < 5V.**
- **This implies that transmission of Logic 1 is degraded as it passes through the gate.**
- **This is called signal degradation.**
- If source tries to increase the voltage , device will cutoff.

Page 17 , book -1

nMOS:

- An nMOS transistor is an almost perfect switch when passing a 0 and thus we say it passes a strong 0.
- However, the nMOS transistor is imperfect at passing a 1. The high
- voltage level is somewhat less than V DD . We say it
- passes a degraded or weak signal.

Transmission of Logic 1 is degraded as it passes through the nmos.

$V_O = V_{DD} - V_T$ (nMOS tr.)

- Vin = 0 , Vo= 0 - 0.7 = - 0.7
- Vin= 5, Vout = 4.3 < 5 V

Nmos is bad in transmitting ‘1’ or 5V and good at transmitting zero.

pMOS

- Transmission of Logic 0 is degraded as it passes through the p device. $V_O = V_{DD} + |V_T|$
- Vin = 0 , Vo=0+0.7 = 0.7
- Vin= 5,Vout = 5.7 >5 V
- pMOS tr is good at transmitting ‘1’ and bad in transmitting ‘0 ‘.
- A pMOS transistor again has the opposite behavior, passing
- strong 1s but degraded 0s.

PASS TRANSISTOR AS SWITCHES.

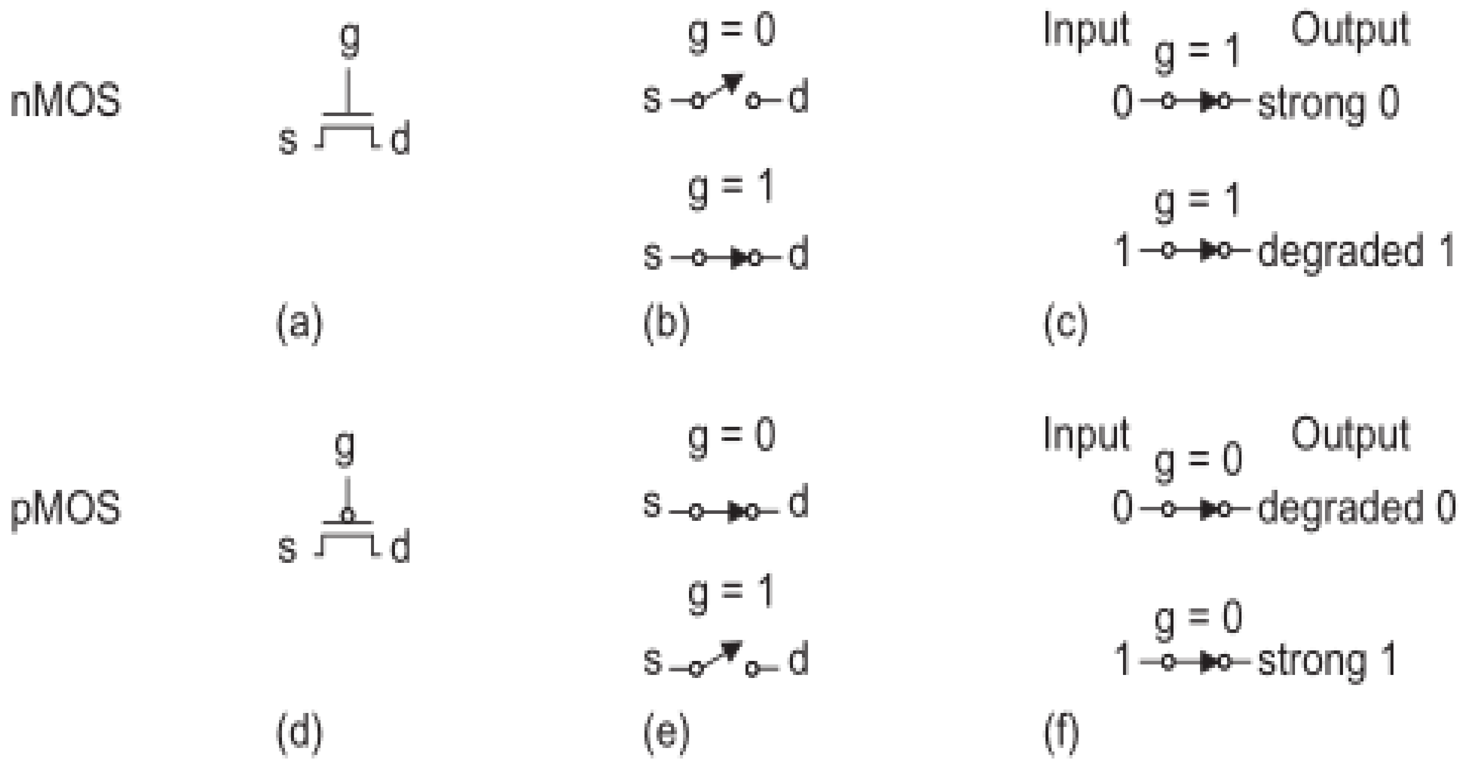
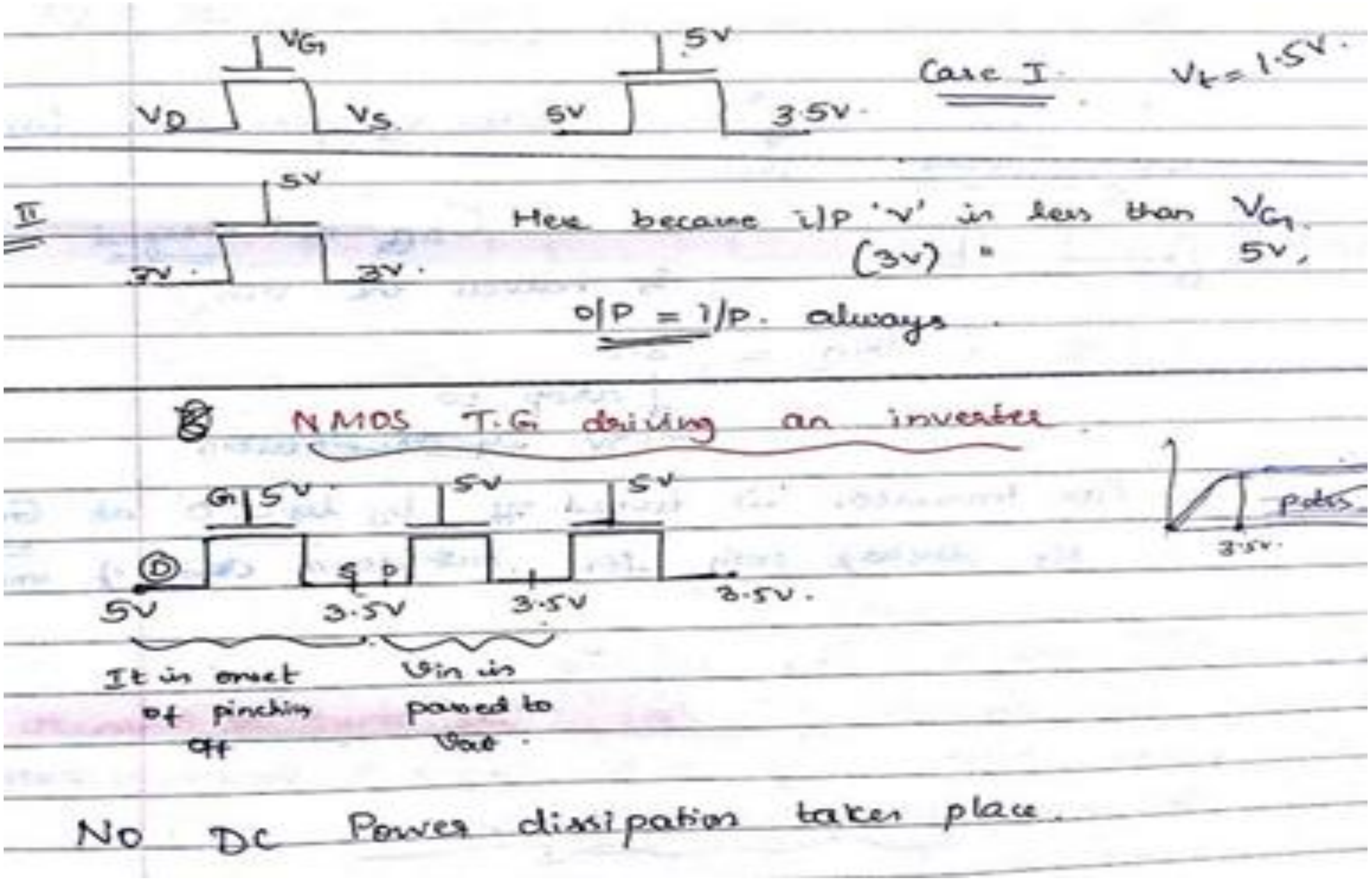


FIGURE Pass transistor strong and degraded outputs

Case: I and II


VG
VD
VS
5V
5V
3.5V
5V
3V.
3V.
G 5V
D
5V
S D
3.5V
5V
3.5V
5V
3.5V.
3.5V.
pass

Case I $V_t = 1.5V$.

II

Here because i/P 'v' is less than V_G.
(3v) = 5v,

o/P = i/P. always.

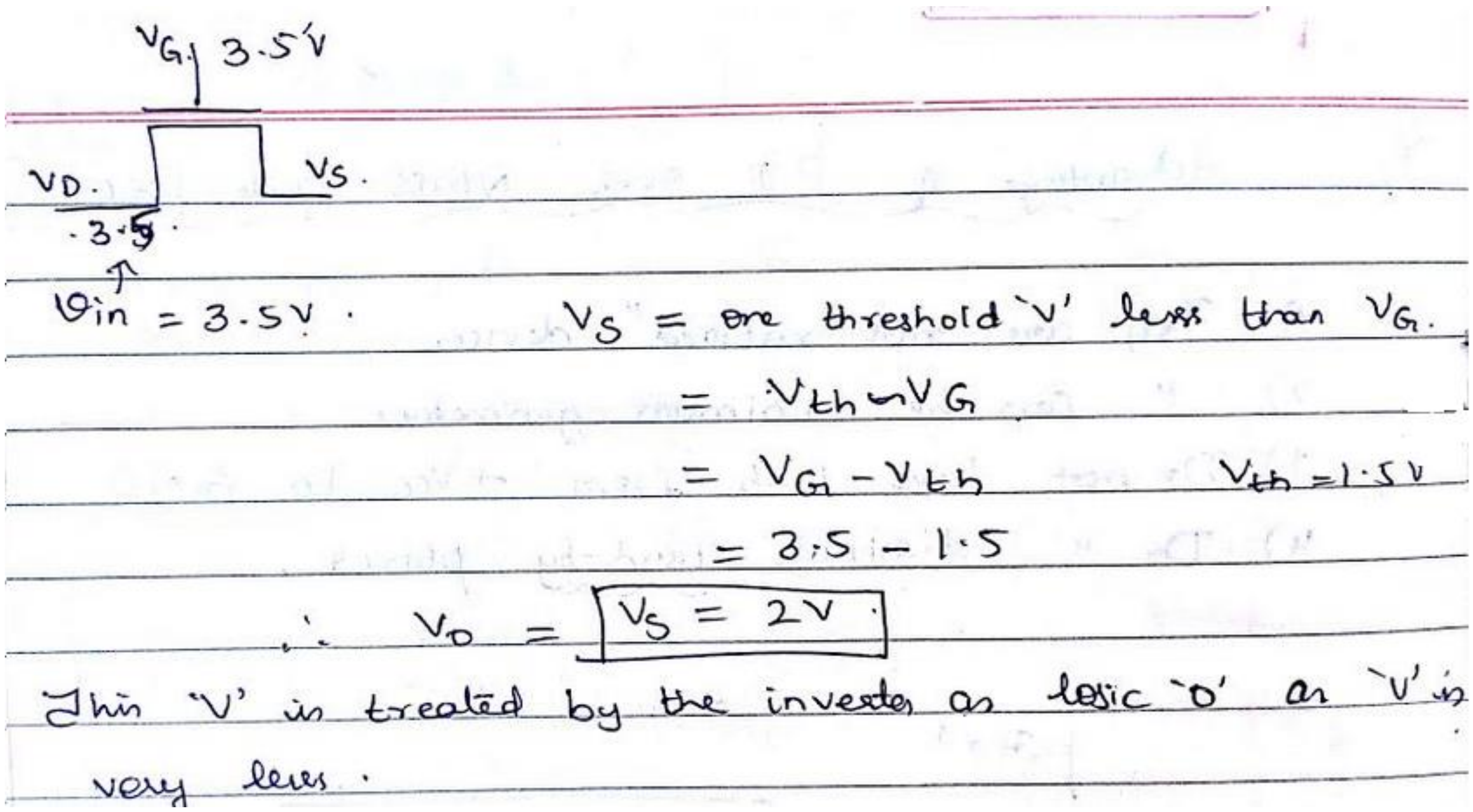
NMOS T.G driving an inverter

It is onset of pinching off

Vin is passed to Vout.

No DC Power dissipation takes place.

Case III



V_G 3.5V

V_D . V_S .

.3.5.

$v_{in} = 3.5V$.

V_S = one threshold 'V' less than V_G.

$= V_{th} \sim V_G$

$= V_G - V_{th}$ $\quad V_{th} = 1.5V$

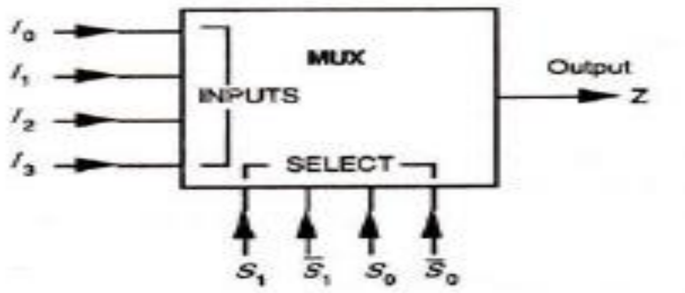
$= 3.5 - 1.5$

$\therefore V_D = \boxed{V_S = 2V.}$

This 'V' is treated by the inverter as logic '0' as 'V' is very less.

4:1 mux using pass transistor

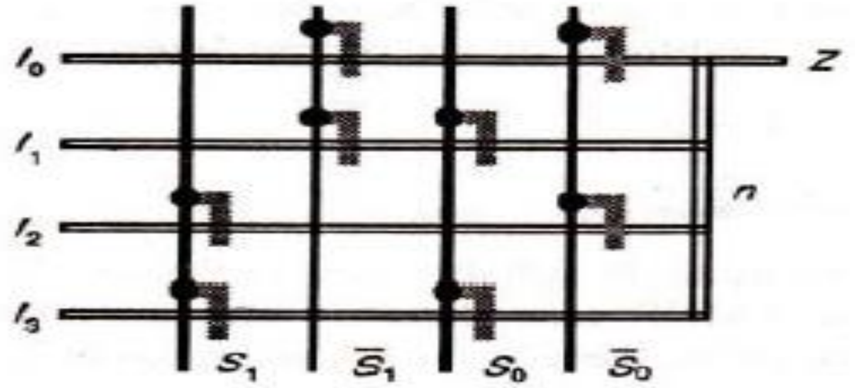
4to 1 MUX using pass transistor



Truth table

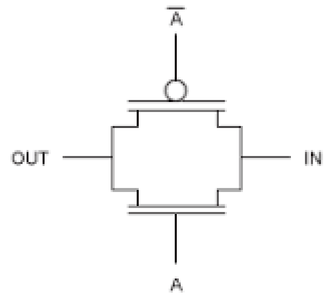
S_1	S_0	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

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



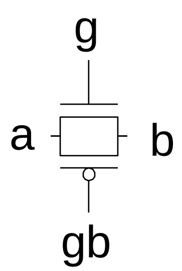
Transmission gate/ Complementary switches

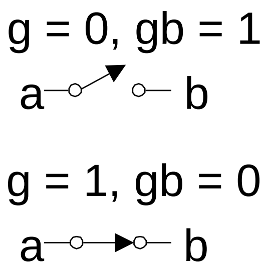
- When an **nMOS** or **pMOS** is used alone as an imperfect switch, we sometimes call it a pass transistor.
- TG is used to eliminate the undesirable threshold voltage effects which give rise to the loss of logic levels in PT.
- A **transmission gate**, or analog switch, is defined as an electronic element that will selectively block or pass a signal level from the input to the output.
- This solid-state switch is obtained by combining an nMOS and a pMOS transistor in parallel.
- They transmit entire voltage from input to output.

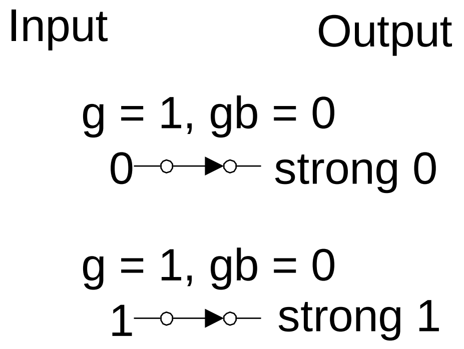


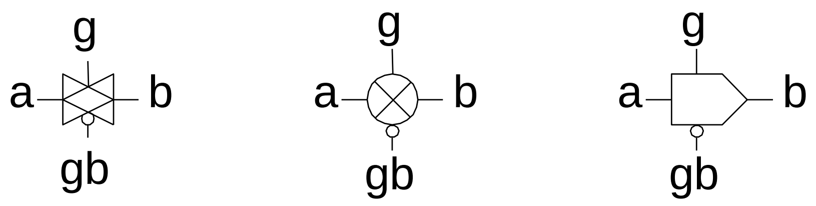
Transmission Gates- WORKING

- Pass transistors produce degraded outputs
- *Transmission gates* pass both ‘0’ and ‘1’ well when g=1 and gb=0.
- When g=0,gb=1,it enters high impedance state.









Advantages : No degradations of signals.

D.A:

More area is occupied .

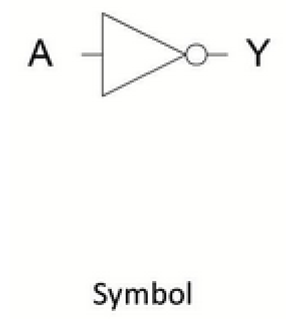
It requires 2 FET’s and one inverter.

GATE (restoring LOGIC)

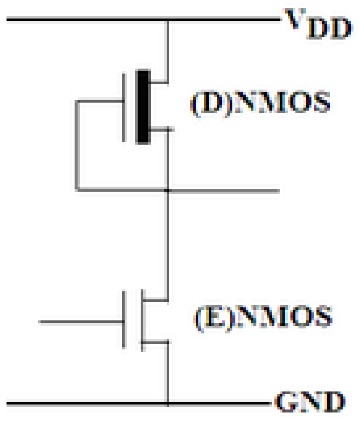
- It is based on general arrangements circuits.
- Ex: Inverter ,NAND,NOR and OR.

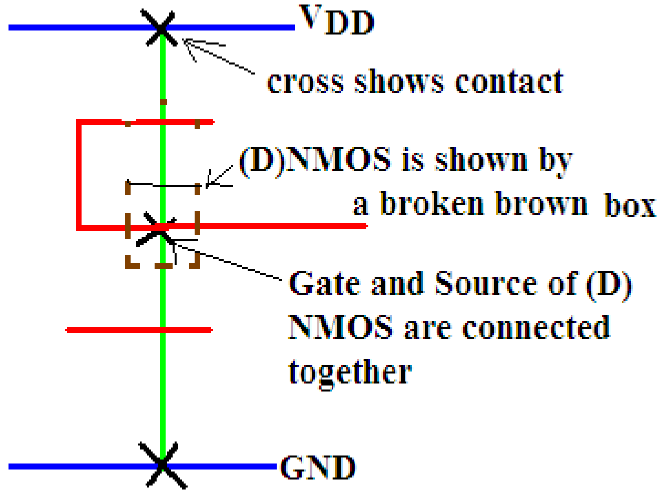
nMOS INVERTER

4:1 or 8:1



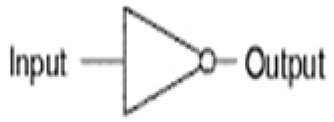
CIRCUIT SYMBOL



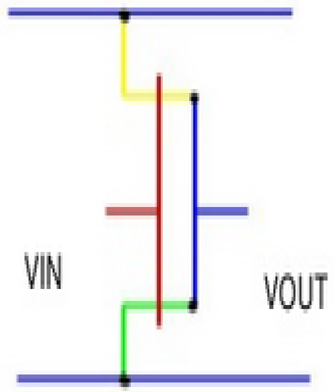

VDD
cross shows contact
(D)NMOS is shown by a broken brown box
Gate and Source of (D) NMOS are connected together
GND

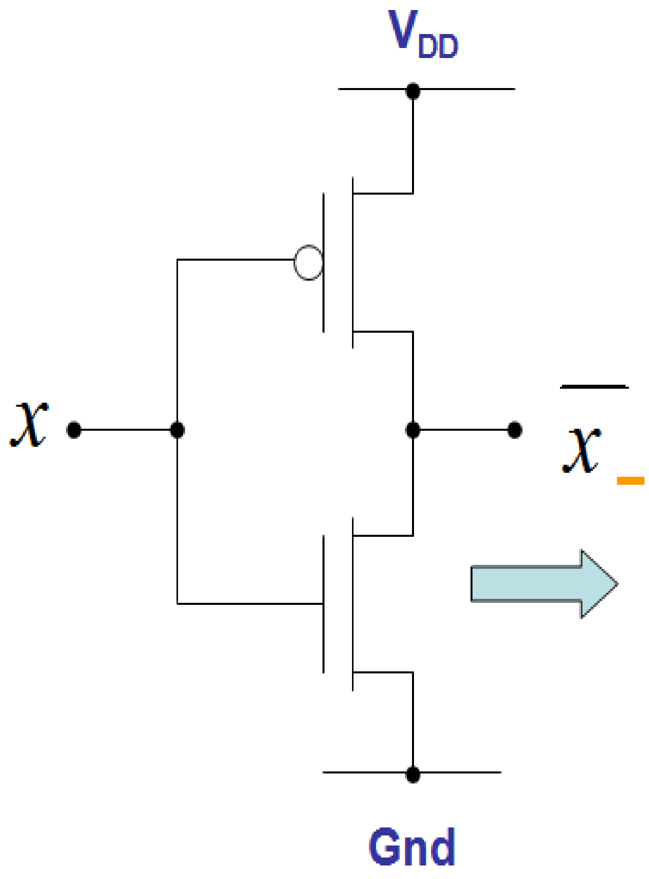
CMOS INVERTER

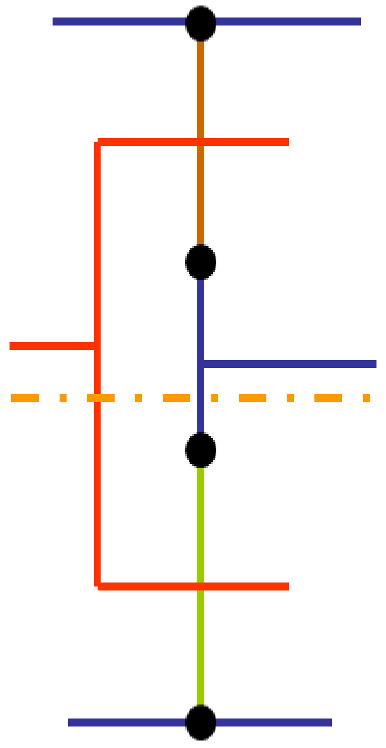
INVERTER



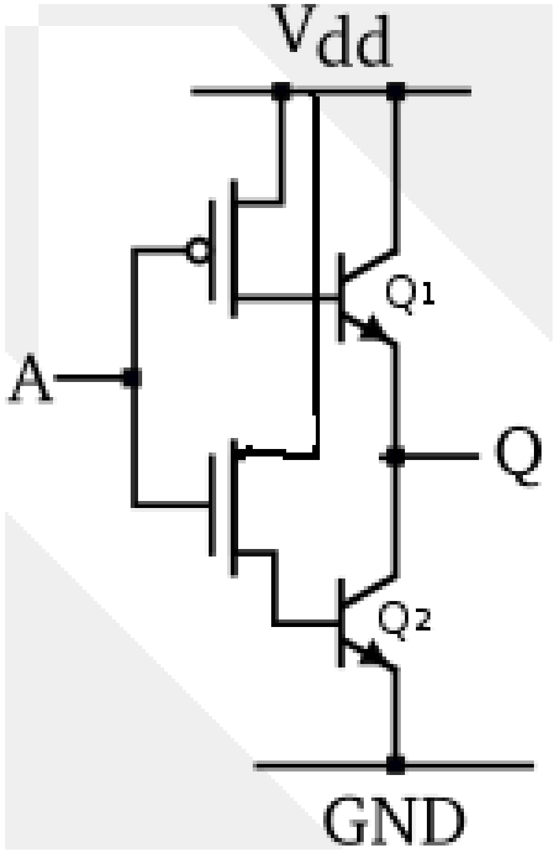
Input	Output
1	0
0	1







Bi- CMOS INVERTER colour plate 3.1d



We combine,
low static power consumption of CMOS and high current driving capability of BJT,
BiCMOS configuration combines best of two worlds.

Page number 20,21, book- 1 vlsi

- Nand gate
- Nor gate

Examples for structured design (Combinational logic)

PARITY GENERATOR (PG)- Basics

- It is a concept to detect errors.
- A parity bit is commonly used for error detection during the transmission of digital signals.
- Parity systems: Odd parity and Even parity
- Parity bit is added to original message signal to make it either even or odd parity.
- Exclusive-OR and exclusive-NOR gates are used in applications such as parity checking, binary comparison and controlled complementing circuits.
- Electrical noise in the transmission of binary information can cause errors. Parity can detect these types of errors.

The Exclusive-OR Gate-Basics

- The output is HIGH if either one or the other inputs are HIGH., *but not both.*
- *o/p is high for odd number of ones.*

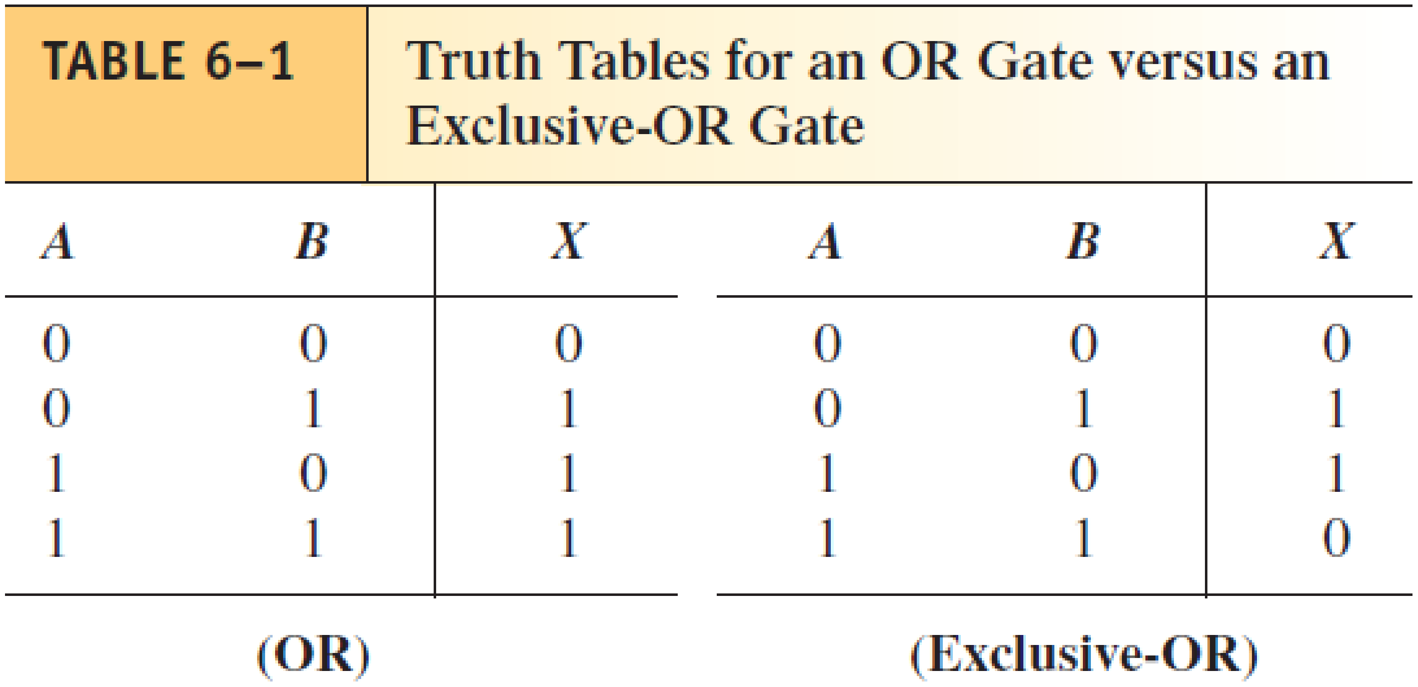


TABLE 6–1 Truth Tables for an OR Gate versus an Exclusive-OR Gate

A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

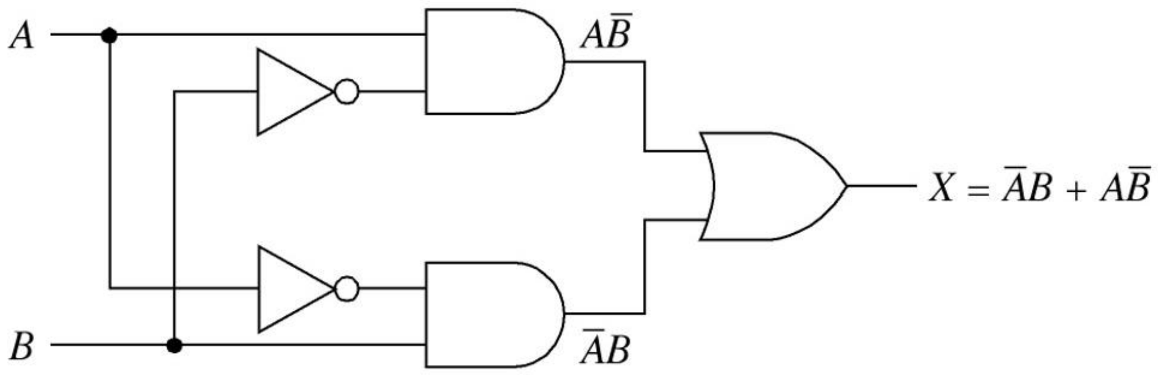
(OR)

A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

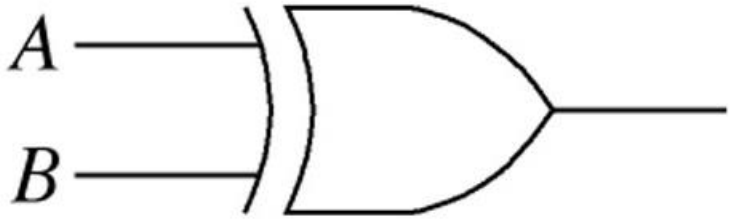
(Exclusive-OR)

The Exclusive-OR Gate- **Basics**

Logic circuits for the exclusive-OR function.

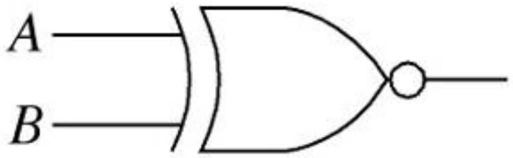


The Exclusive-OR Gate



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

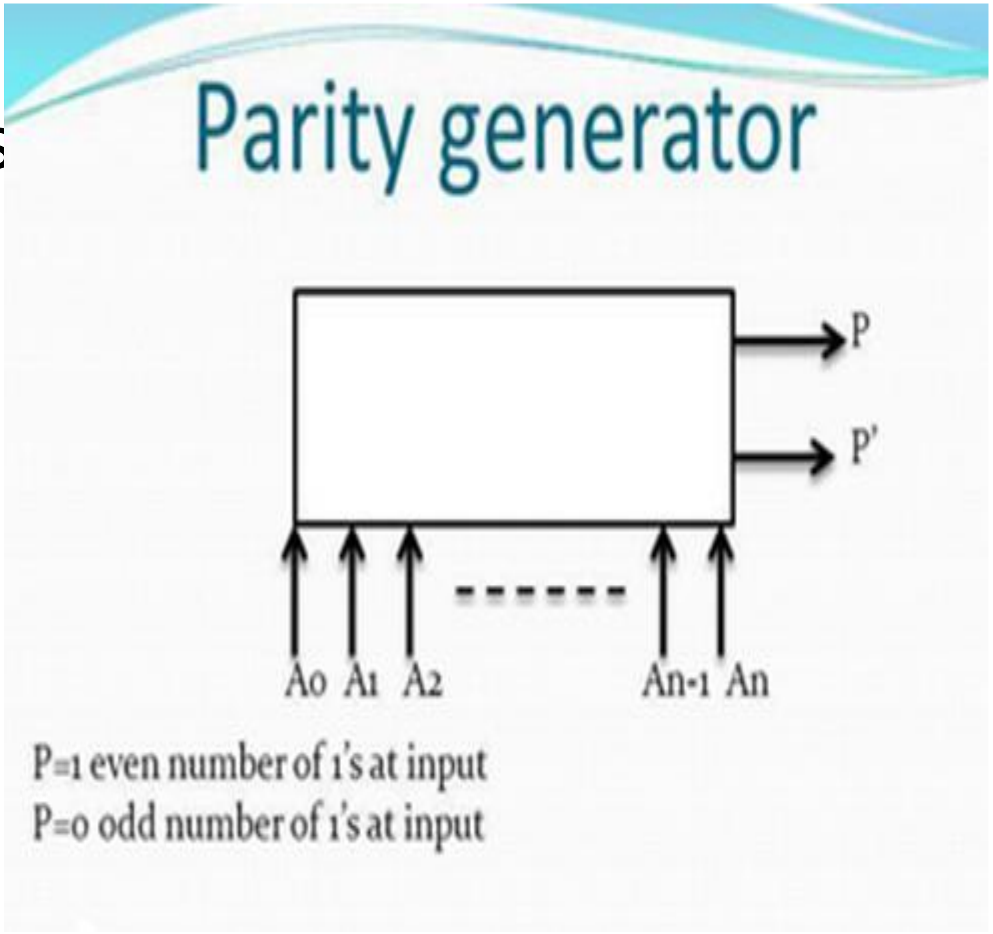
The Exclusive- NOR Gate



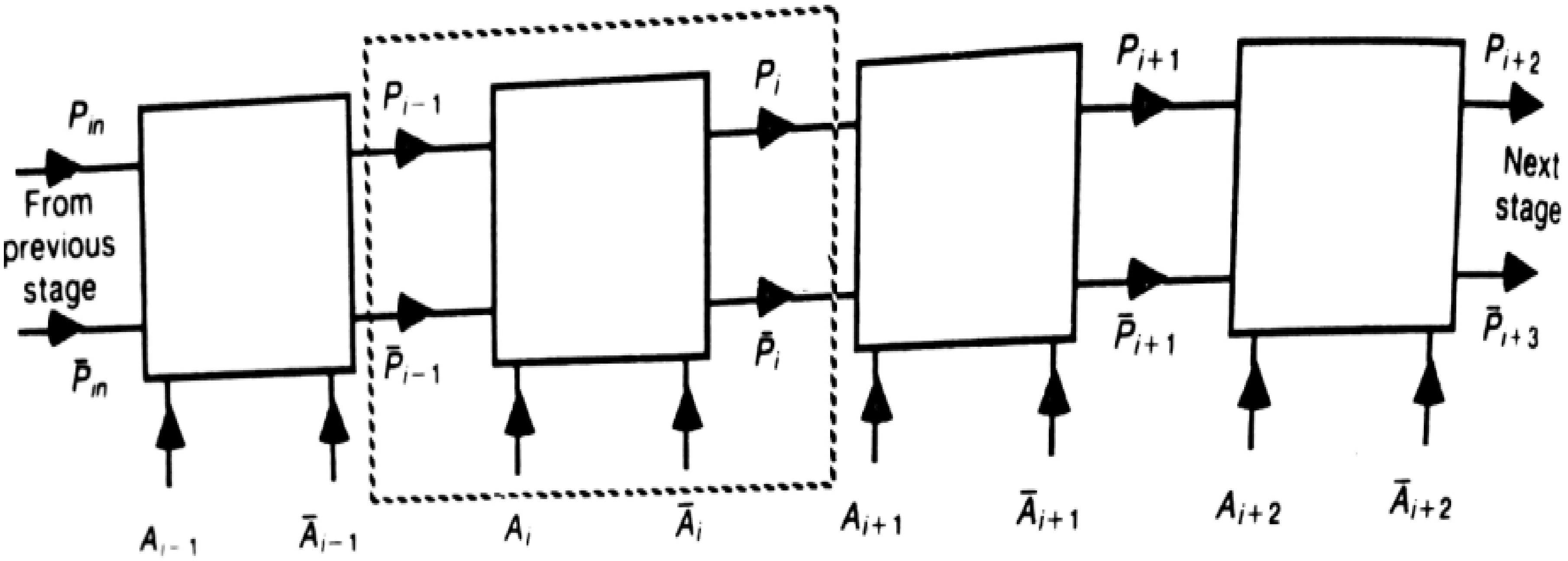
$$X = \overline{A \oplus B} = AB + \overline{A}\,\overline{B}$$

Basic block diagram for PG

- (n+1)bit input
- n = no of n inputs
- 1 bit is = parity bit


Parity generator
P
P'
A0 A1 A2
An-1 An
P=1 even number of 1's at input
P=0 odd number of 1's at input

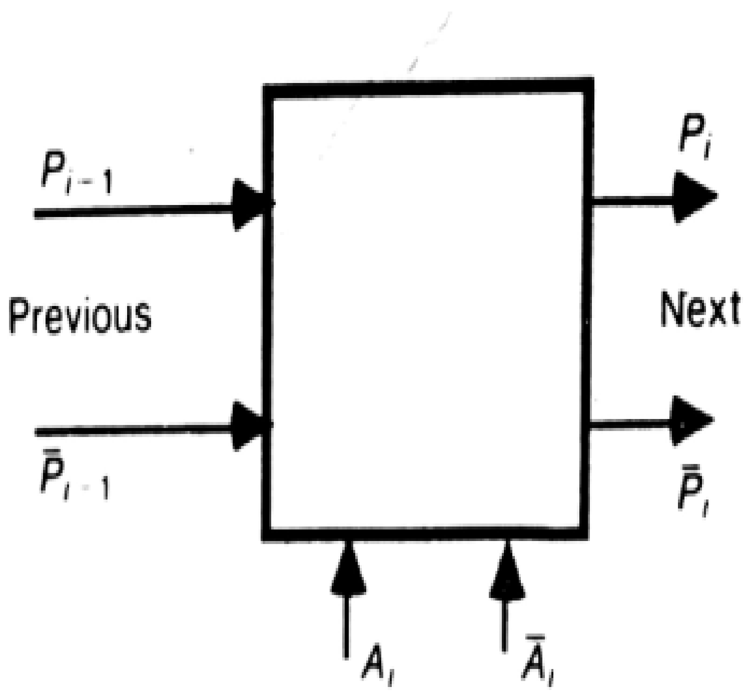
Page 152,fig 6.16



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

Parity generator—structured design approach.

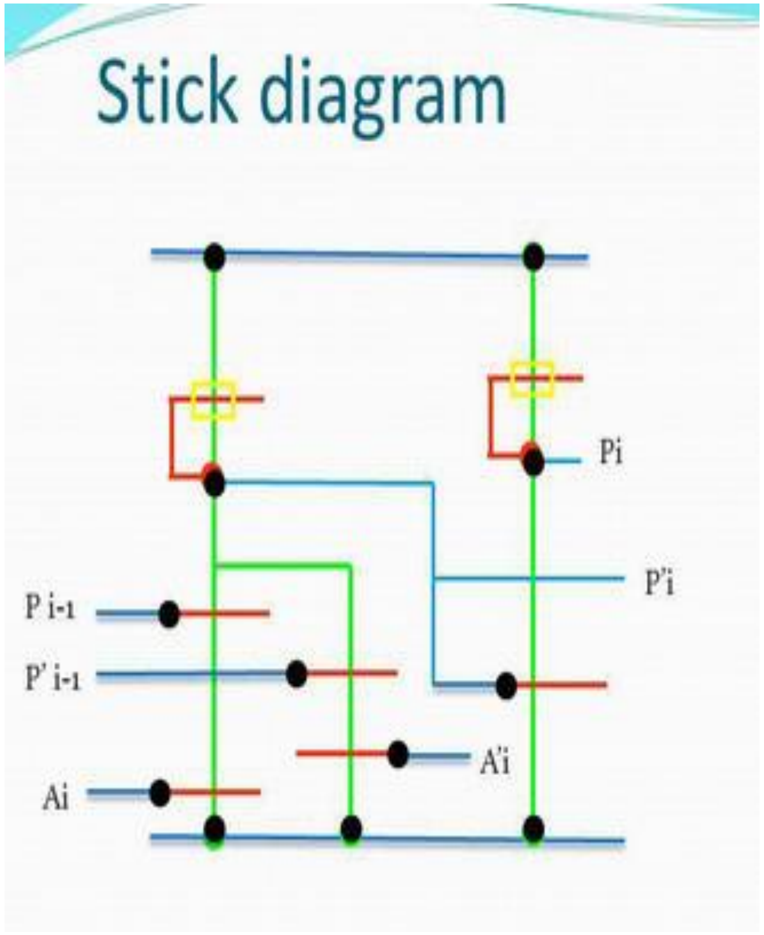
ONE BIT PARITY CELL PAGE 152 , FIG 6.17



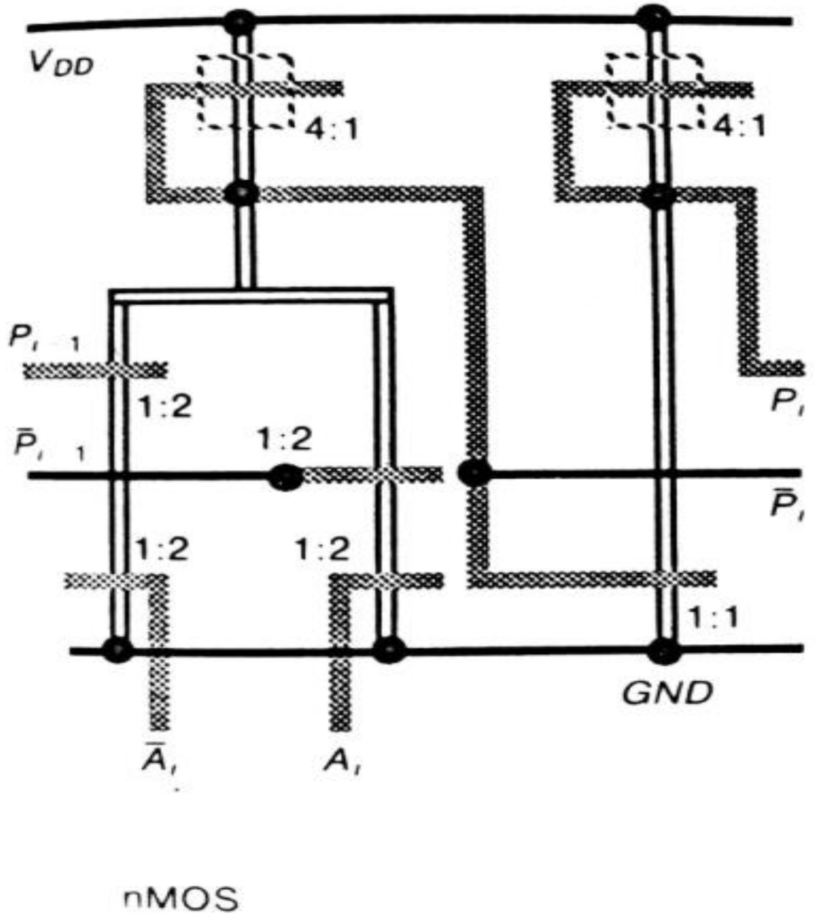
Parity generator—basic one-bit cell.

- Here ,parity information is passed from one cell to next cell and is modified or not , depending on input Ai and $\bar{A}i$.
- $P_i = (P'_{i-1})\ (Ai) + (P_{i-1})\ (A'i)$
- X-OR gate is used between inputs and Parity of previous stage.
- When writing stick diagram, cell boundary must be chosen in each case so that cells may be cascaded at will.

Stick diagram for one bit nMOS PG.

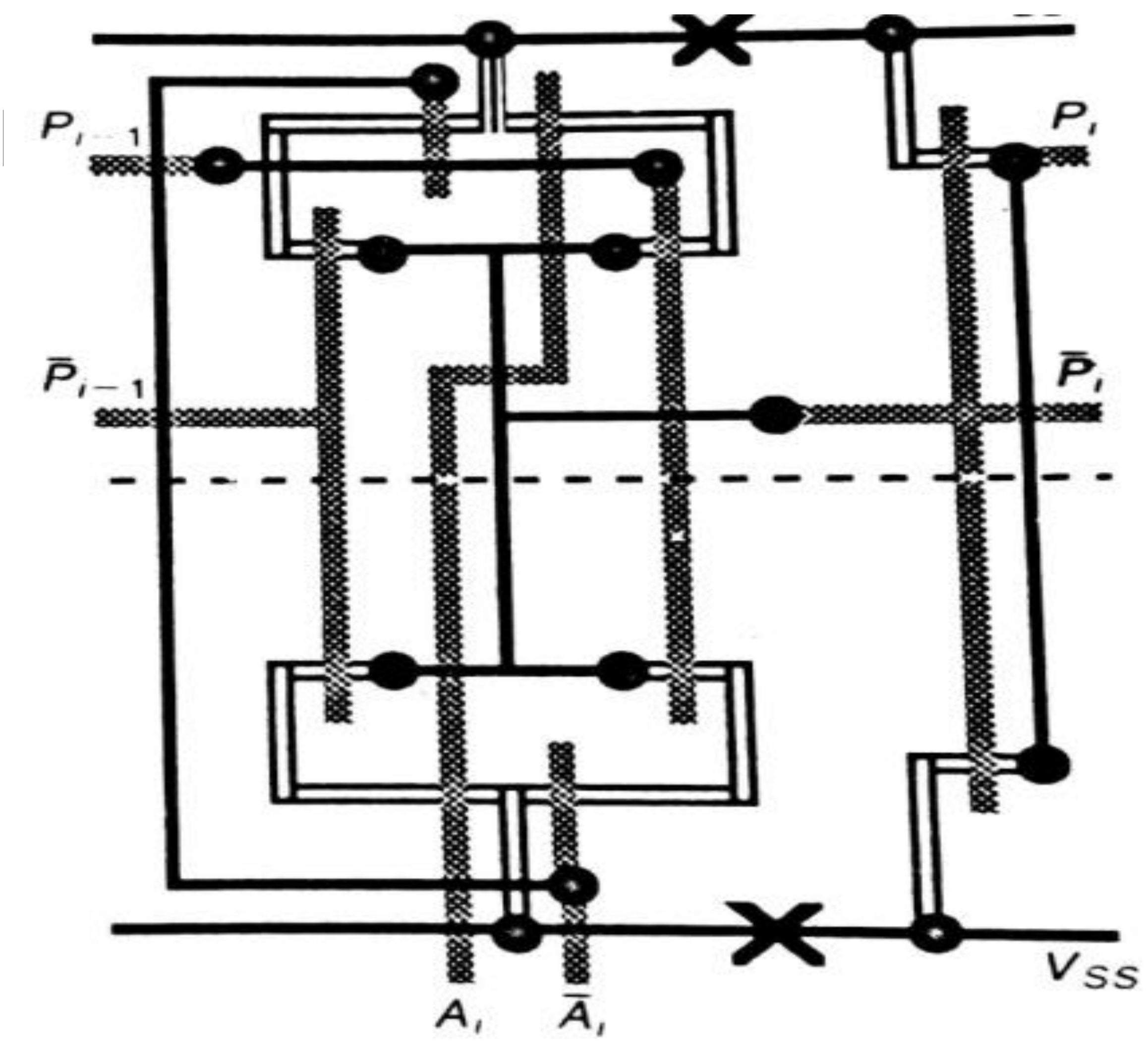


- Fig 6.18 a



Stick diagram for one bit CMOS PG

- Fig 6.18

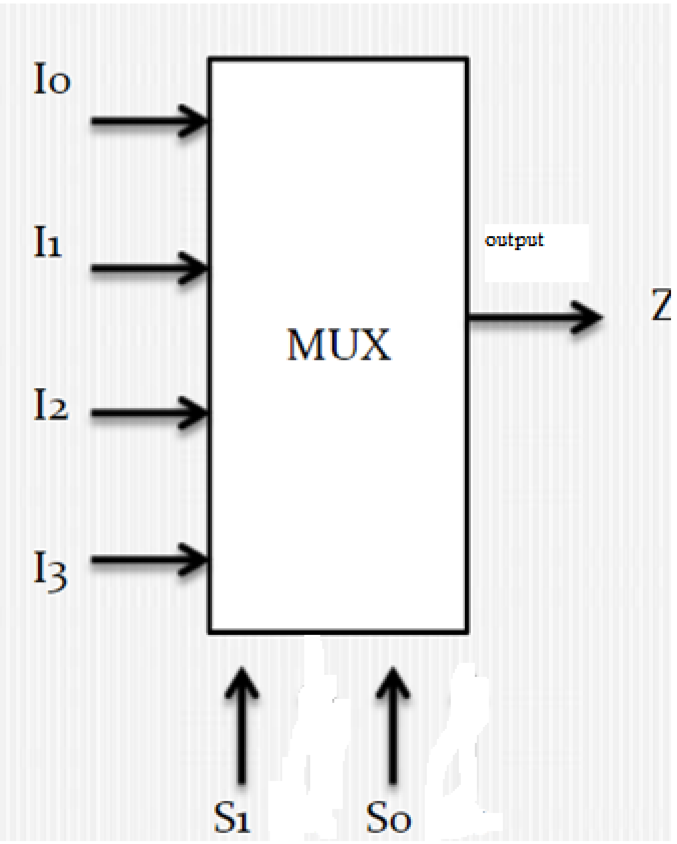


CMOS

MULTIPLEXER / DATA SELECTOR

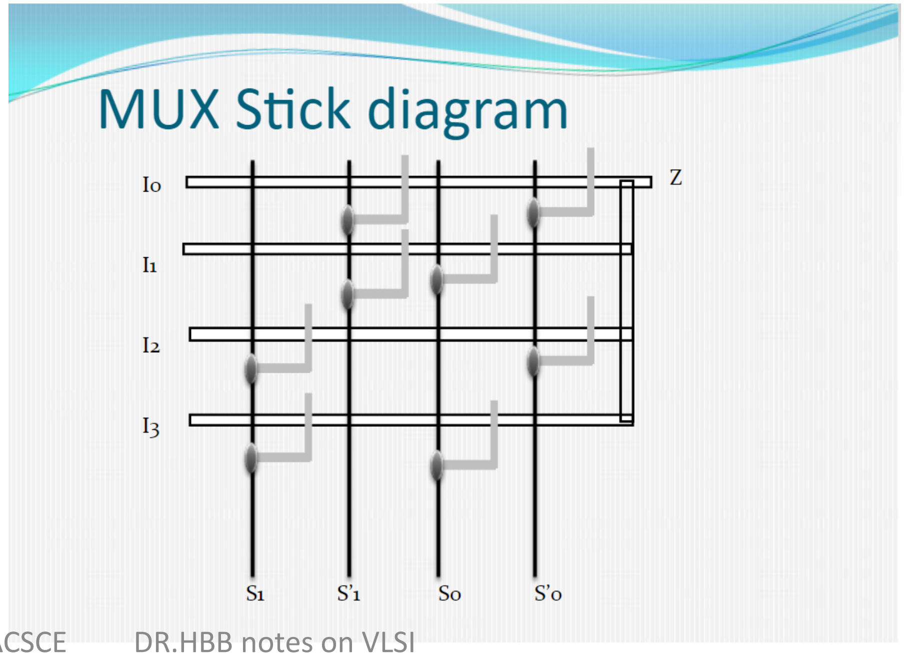
- The *multiplexer OR* “MUX” is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal.
- It can be either digital circuits or analogue types using transistors, MOSFET’s or relays to switch one of the voltage or current inputs through to a single output.
- They are available in a number of standard configuration in TTL or other logic families.

4 WAY MULTIPLEXER.



Select lines

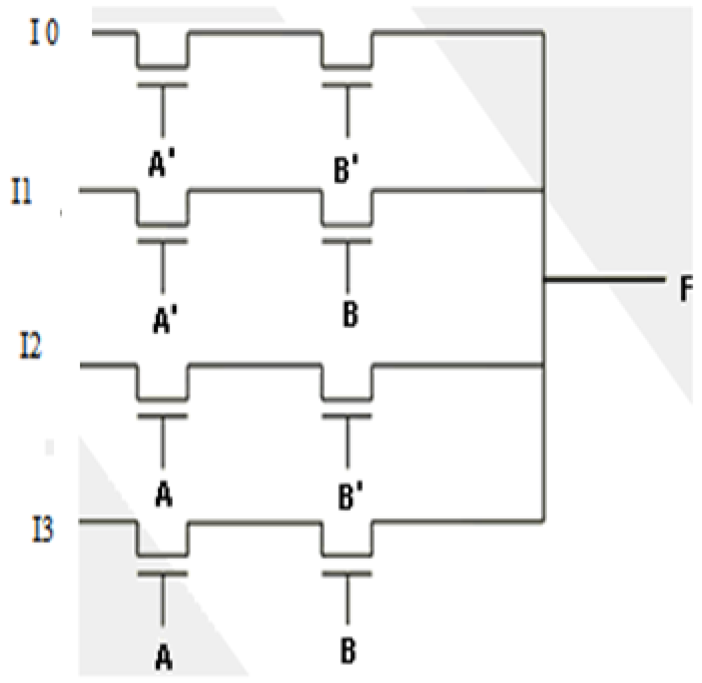
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

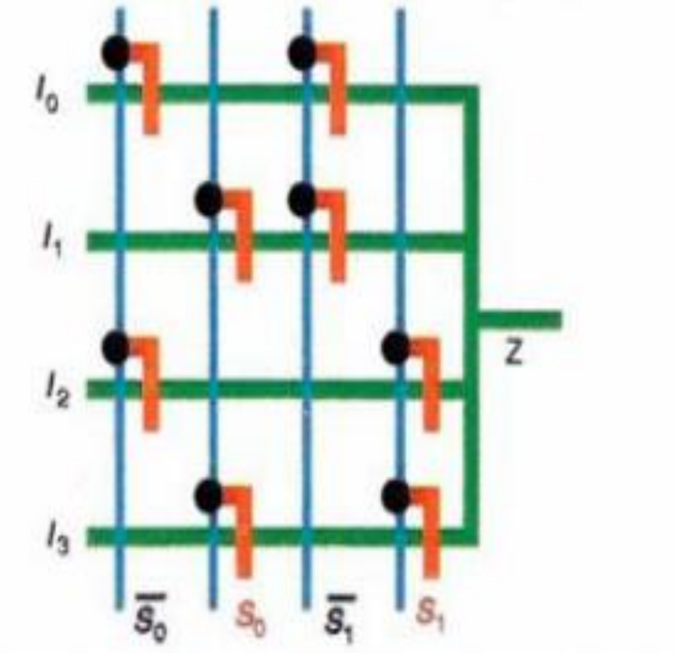


MUX Stick diagram

Implementation using pass transistor

$$Z = \left(I0\, \overline{S}_0 . \overline{S}_1 \right) + \left(I1\ \overline{S}_0 . S_1 \right) + \left(I2\ S_0 . \overline{S}_1 \right) + \left(I3\ S_0 . S_1 \right)$$





n-type pass transistor based 4-way MUX.

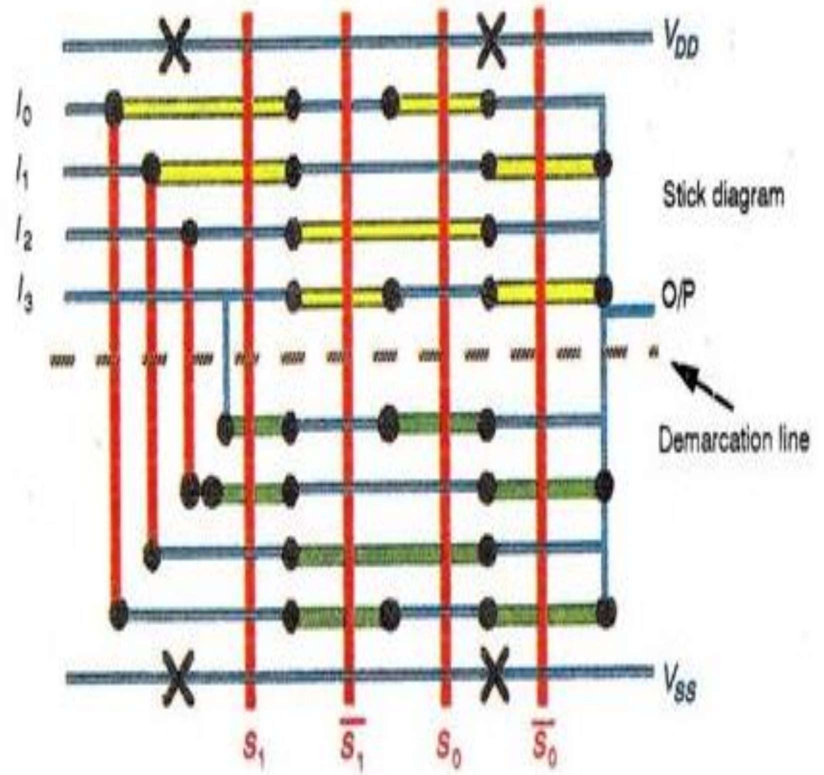
transmission gate based CMOS stick diagram - 6.24 b- place colour plate

Note :

All n transistor are below the demarcation line and close to Vss rail to configure p-well and Vss contacts.

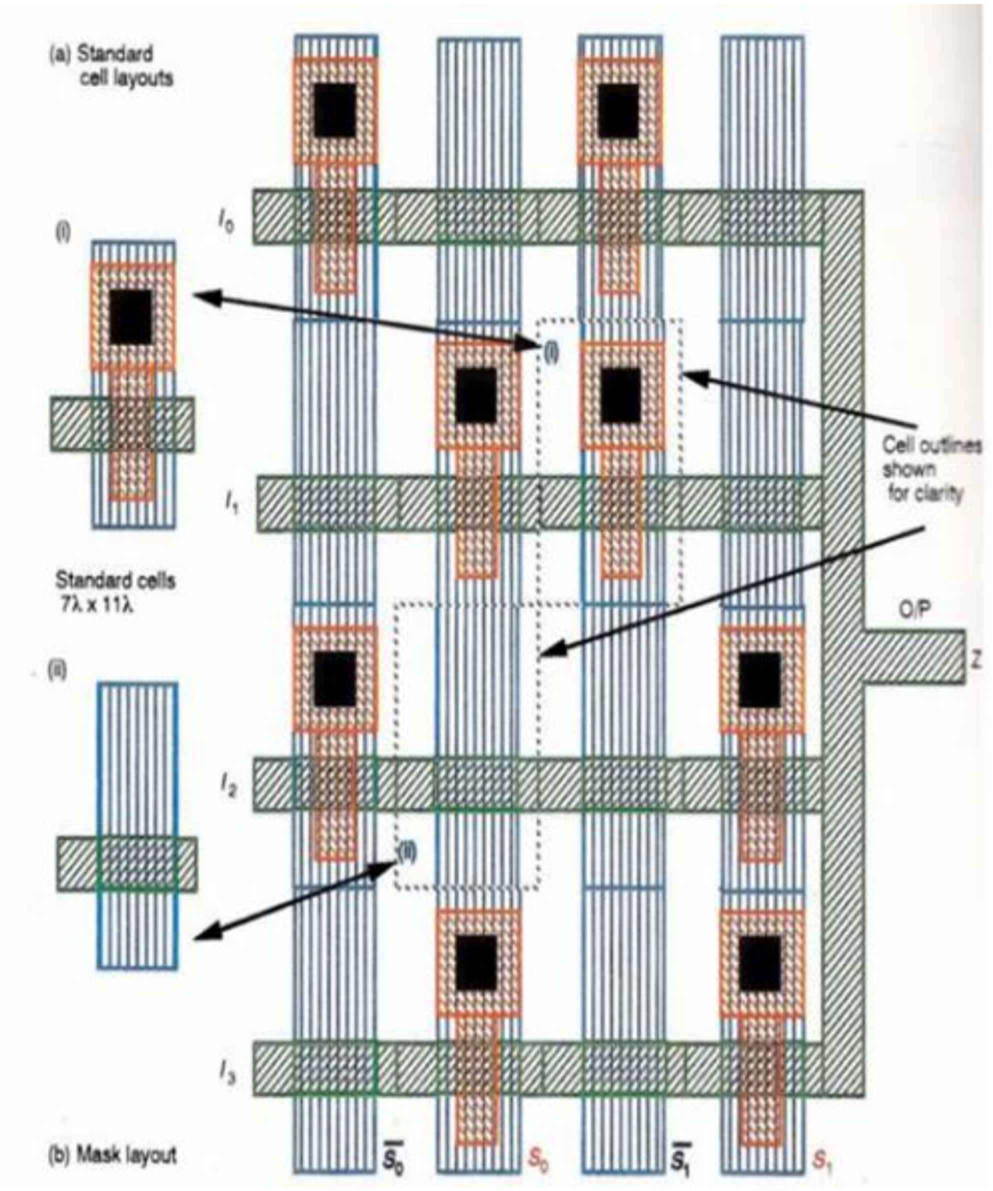
All p transistor are above the demarcation line and close to Vdd rail .

Logic level 1 will not be degraded.

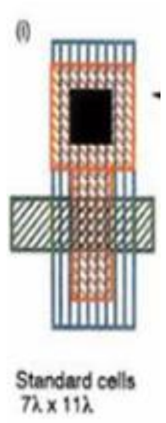


COLOR PLATE CMOS transmission gate based 4-way MUX.

FOUR WAY N-SWITCH BASED MULTIPLEXER LAYOUT



Standard cell = 7λ x 11λ



PLA – Programmable logic array

Contents covered - I

The Programmable Logic Array (PLA)- VLSI Design"- Douglas A. Pucknell& Kamran Eshraghian, PHI 3rd,Edition (original Edition – 1994). Slide 2- **11**

Contents covered – II

FIELD PROGRAMMABLE GATE ARRAYS

FPGA ARCHITECTURE

PHYSICAL DESIGN OF FPGA

"FPGA Based System Design"- Wayne Wolf, Pearson Education, Technology and Engineering, 2004 . Slide 22 - 49

The Programmable Logic Array (PLA)

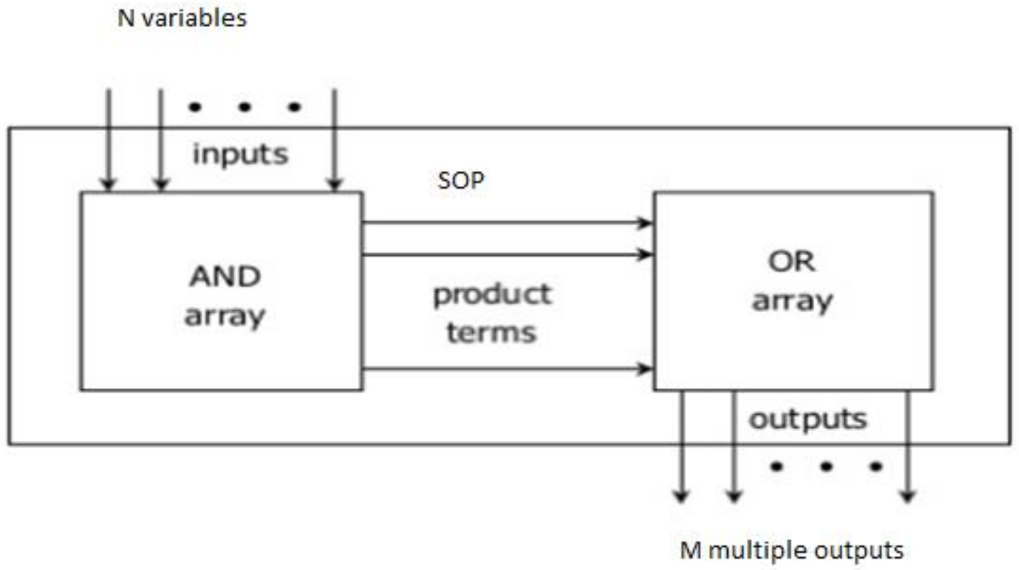
Text book :

“Basic VLSI Design”- Douglas A. Pucknell& Kamran Eshraghian, PHI 3rd,Edition (original Edition – 1994).

The Programmable Logic Array (PLA) –introduction

A simple solution to mapping of **irregular combinational logic functions** into **regular Structures** is provided by PLA.

It provides the designers a regular way of implementing / providing multiple ‘m’ outputs with n inputs variables using sum-of-products (SOP) logic functions.



'AND' plane is responsible for the generation of all product terms needed to form logic functions. 'OR' plane OR's (sums) the selected product terms together to form the desired logic functions.

The Programmable Logic Array (PLA)

- Personalized" by making or breaking connections among the gates.
- It performs the same basic function as a ROM.
- Pre-fabricated building block of many AND/OR gates (or NOR, NAND).
- It consists of two level of combinational logic functions .
- PLA describes class of standalone devices that allows users to program the functionalities.
- Presently, we have PLA with 14 i/p variables, 96 product terms , 8 o/p functions. Such elements are programmed by manufacturer or field programmed by user to meet requirements.
- The capability of these programmable devices are limited and have been replaced by significantly powerful field programmable gate array (FPGA).

■

1) In Vlsi design, however, Custom PLA's can be readily designed and must be programmed during the design process . Thus, for the VLSI designer , PLA's are tailored to specific task with little wastage of function or space .
.
2)PLA structure is normally expanded, or modified during design.

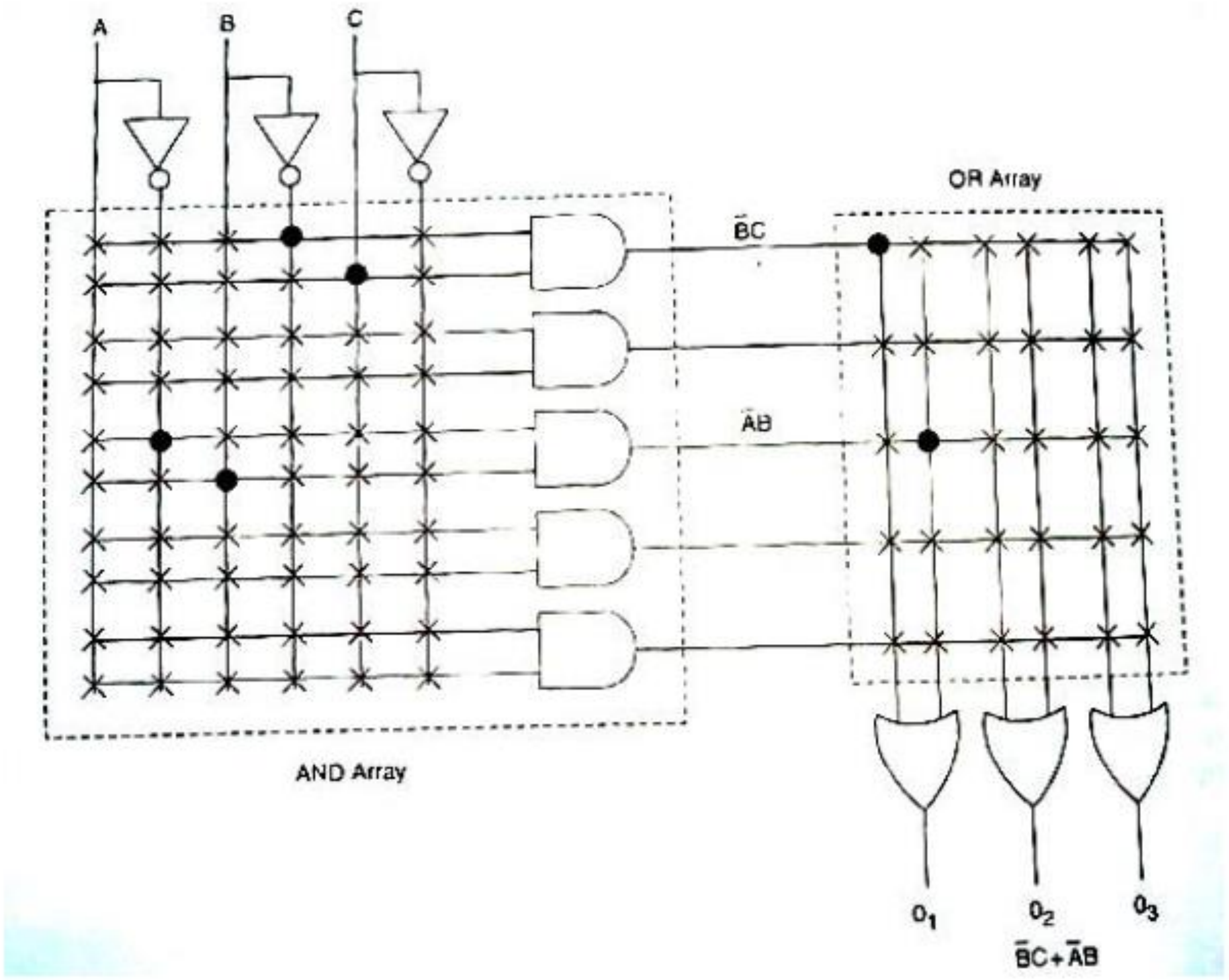
3) In vlsi design, our objectives is to map circuits onto silicon to meet specifications Floor plan indicates PLA mapping onto chips. It also gives notional areas and relative disposition of particular circuits and subsystems.
.
De-morgan's theorem is used to manipulate AND-OR combinational logic requirements into NOR form. .

Depending on which of the AND/OR logic arrays is programmable, we have three basic organizations

ORGANIZATION	AND ARRAY	OR ARRAY
PAL	PROG.	FIXED
PROM	FIXED	PROG.
PLA	PROG.	PROG.

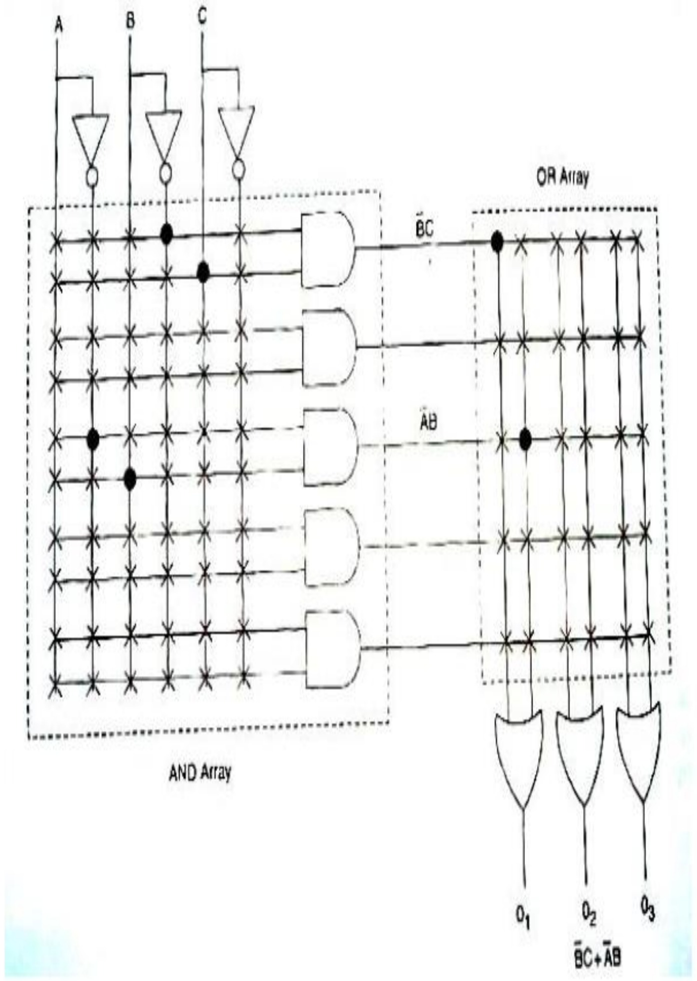
Programmable Array Logic – PAL

General architecture of PLA



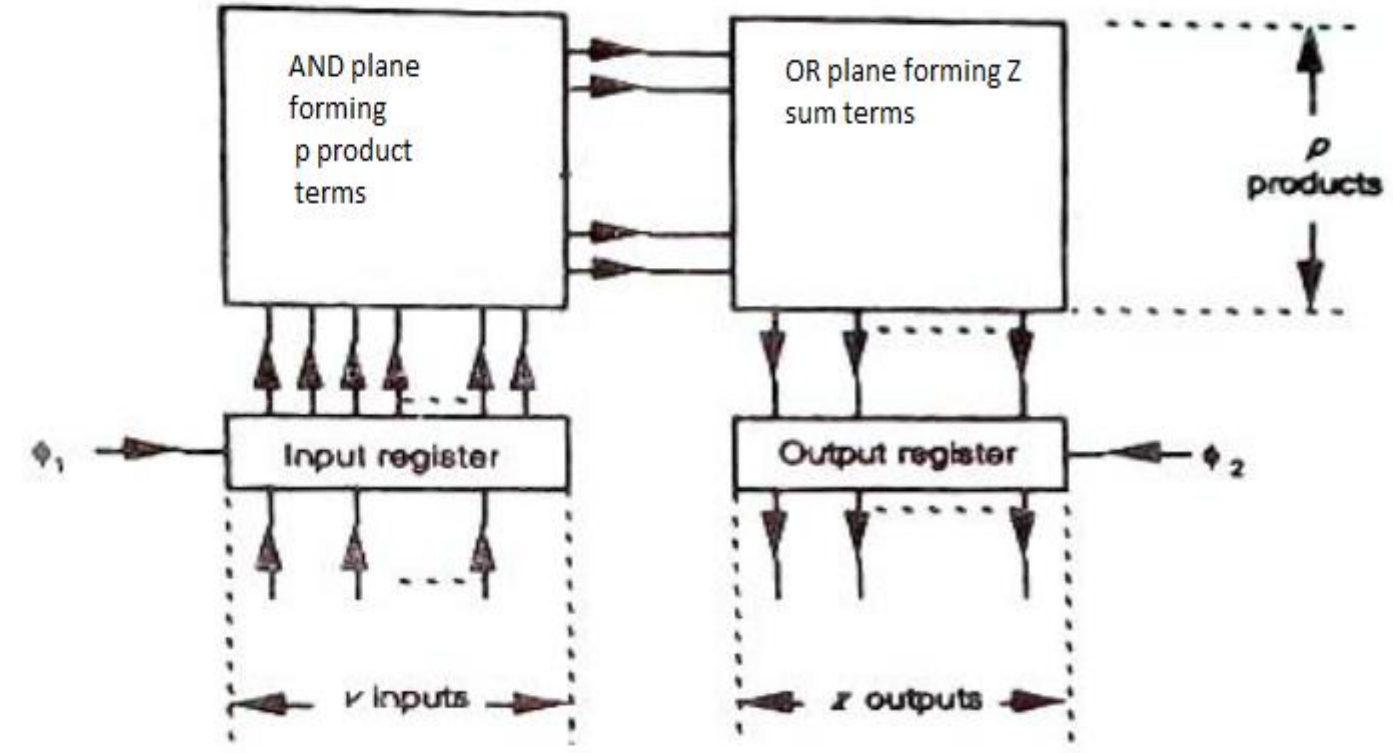
General architecture of PLA

- Fig. shows 3 input and 3 output PLA. Along with 3 inputs, its compliment is also available.
- With this, it is possible to generate many functions of product terms in AND array.
- Inputs are A, B & C and outputs are O_1, O_2 & O_3 .
- X indicates no connection and desired product term can be obtained by making relevant connection in the AND array (shown with dot)
- Similarly connection can be made in OR array.
- For obtaining O_1 = B'C + A'B, the connections are shown in the Fig.



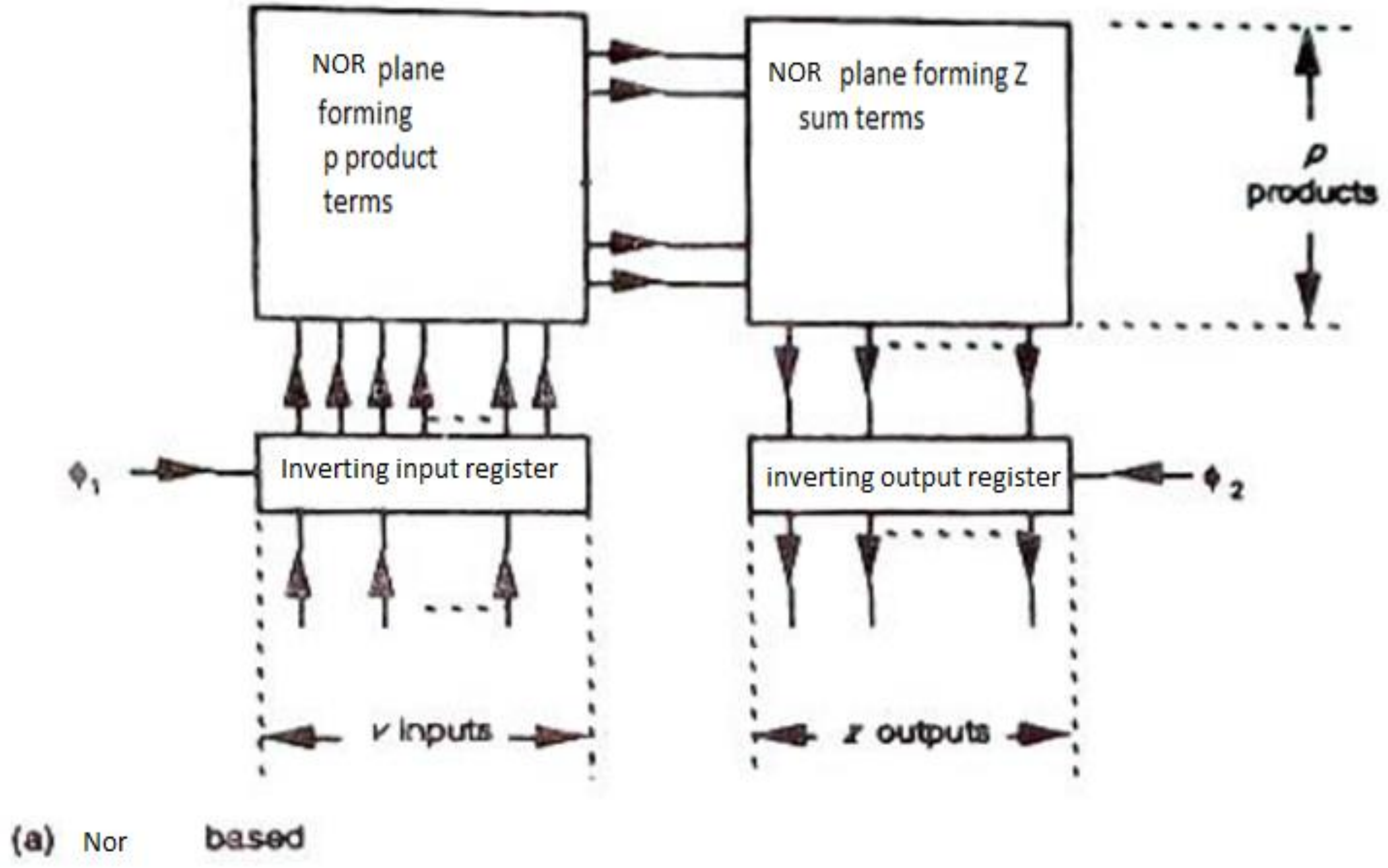
- In VLSI design, **objective is to map circuits onto Si to meet the specifications.**
- In circuit implementation for AND and OR array needs NAND and NOT logic and NOR and NOT logic respectively. But this includes **more fabrication steps**.
- However, this can be **simplified by implementing the logic in NOR** logic. Thus AND and OR array can be implemented using NOR logic.
- If the output of NOR is complimented, then we get the OR logic. Similarly if both the inputs of AND is complimented and given to NOR it gives AND logic

PLA floor plan for AND/OR based



(a) *And/Or* based

PLA floor plan for NOR based

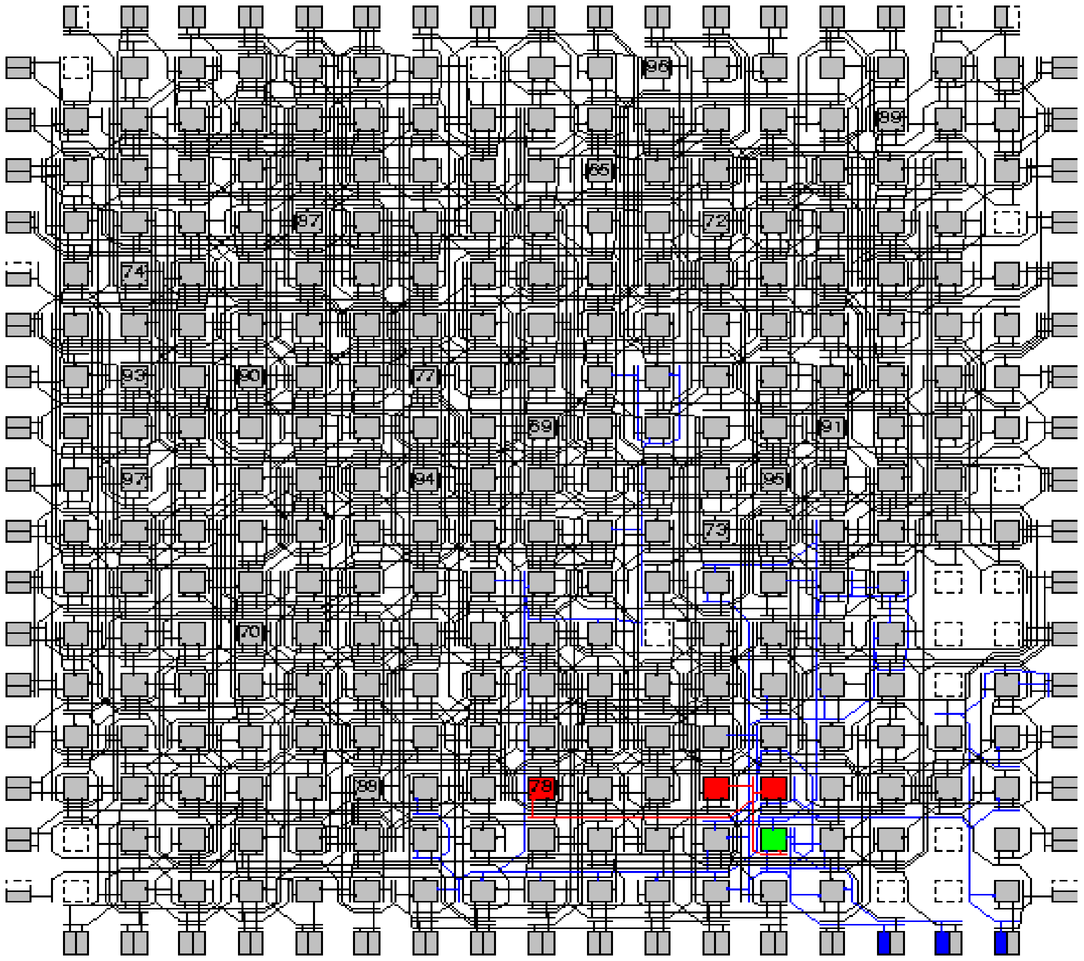

NOR plane forming p product terms
NOR plane forming Z sum terms
p products
ϕ1
Inverting input register
inverting output register
ϕ2
v inputs
z outputs

(a) Nor based

FPGA –
FIELD PROGRAMMABLE GATE ARRAYS

"FPGA Based System Design"- Wayne Wolf, Pearson Education, Technology and Engineering, **2004**

FPGA -FIELD PROGRAMMABLE GATE ARRAYS


Routing succeeded with a channel width factor of 7.

•

FPGA

- FPGA plays the complementary role of microprocessor.
- **Standard parts**: They are not designed for any particular functions but are programmed by customer for a particular purpose.
- **Implement multilevel logic**:

  They use both programmable logic blocks and Programmable interconnect to build the multi level logic functions.
- **FABRIC: Combination of logic and interconnect.**
- They are used in all sorts of digital systems. Because of its

  a) high speed Telecommunications equipment

  b) as video accelerators in home personal video recorders (PVR)'s.
- FPGA is far better than smaller programmable device like PLD's.

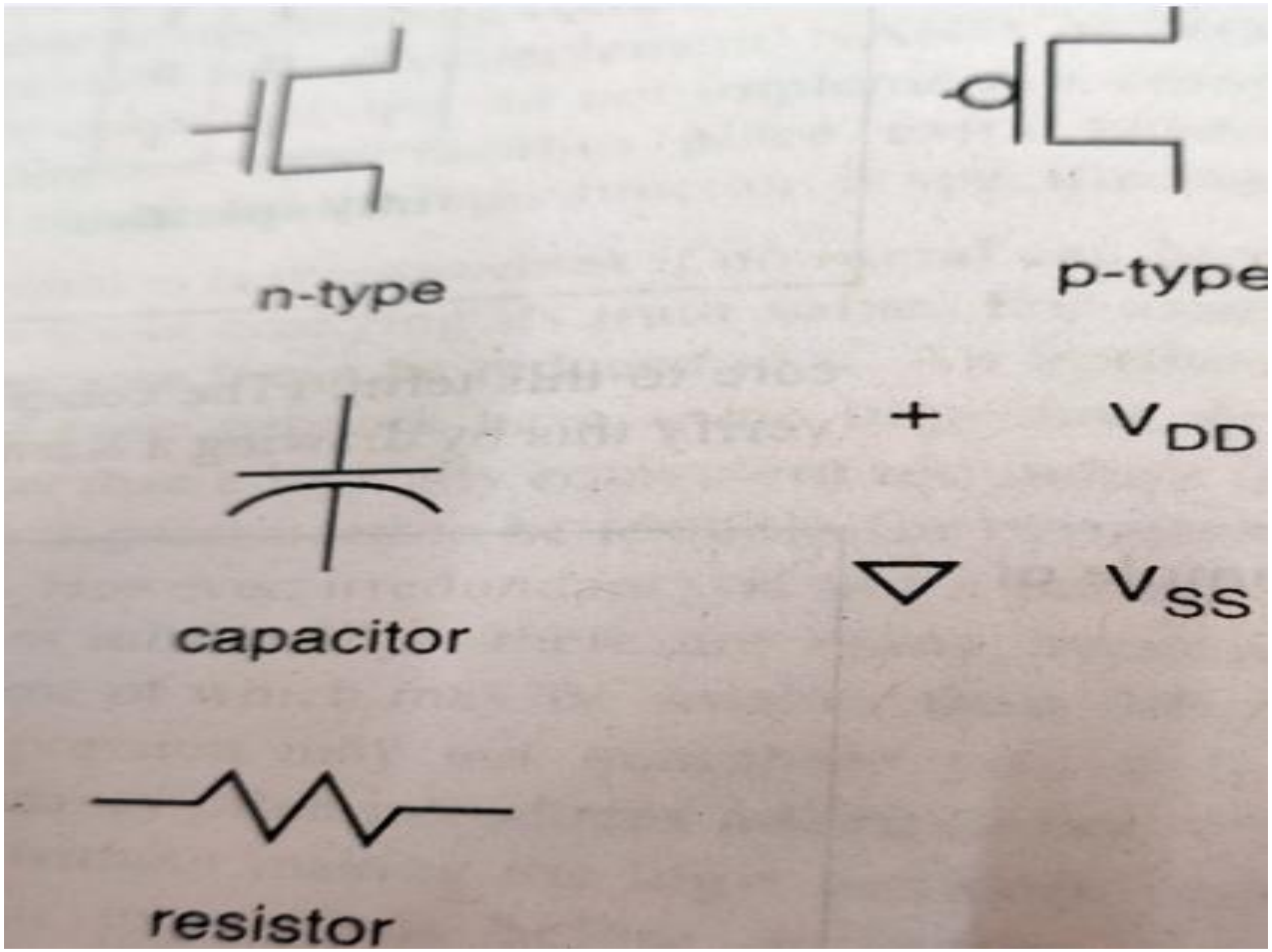
$$\begin{tikzcd} & \textbf{FPGA} \arrow[dl] \arrow[dr] & \\ \textbf{Permanently programmed} & & \textbf{Re-programmed / Reconfigurable Devices} \end{tikzcd}$$

❑ Device need not be thrown away every time a change is made.

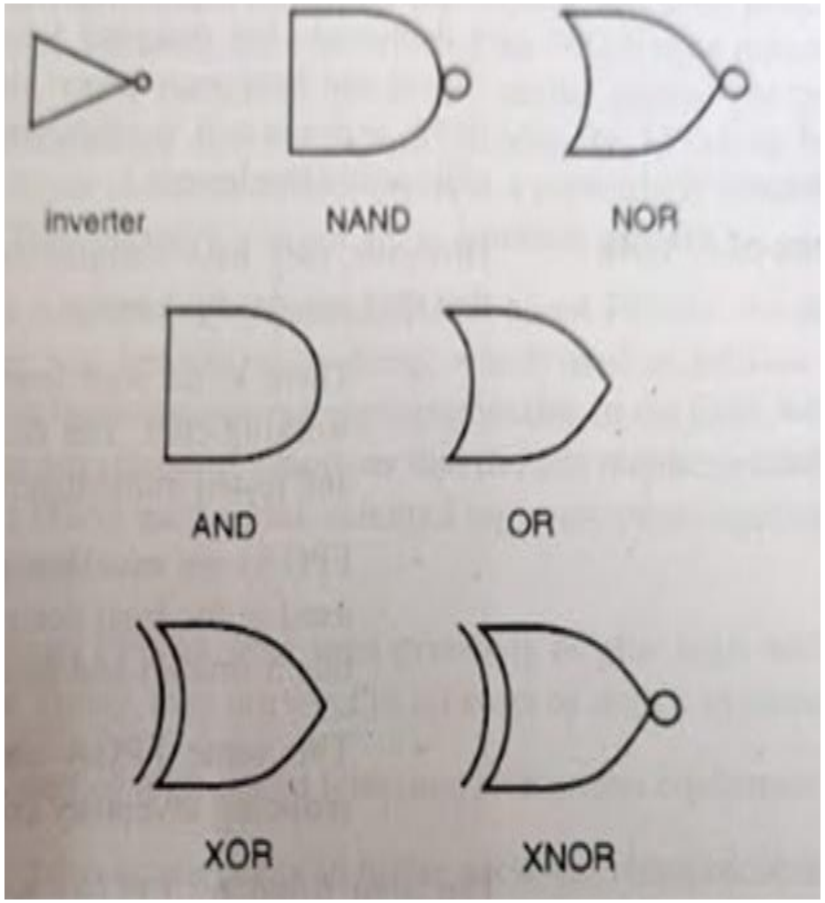
❑It can be reprogrammed on the fly during system operation.

Thus ,one hardware to perform several diff functions.
Thus systems can be operated in diff modes.

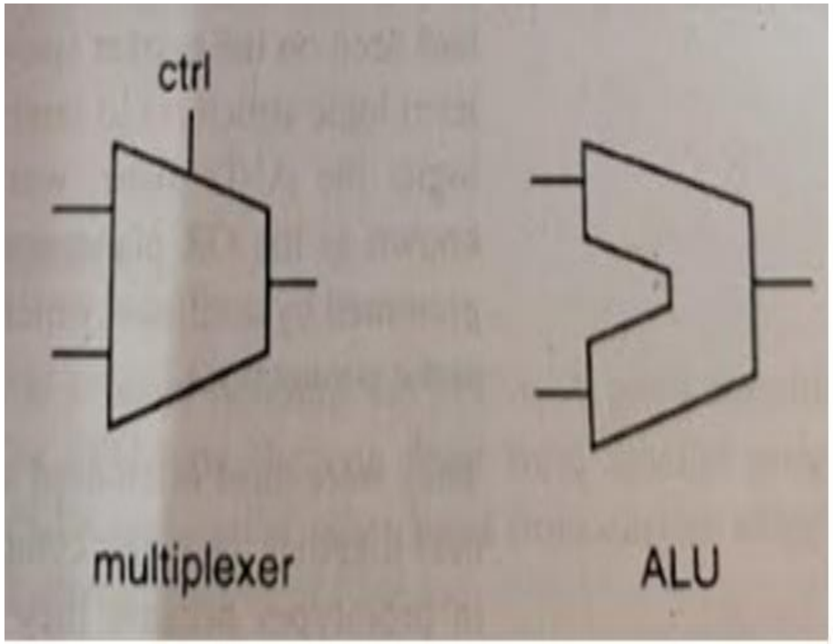
Schematics and Logic Functions



Schematics Symbols For Logic Gates



Schematics Symbols For Register Transfer Components



Digital Design and FPGAs

Microprocessor or

General Purpose operating system

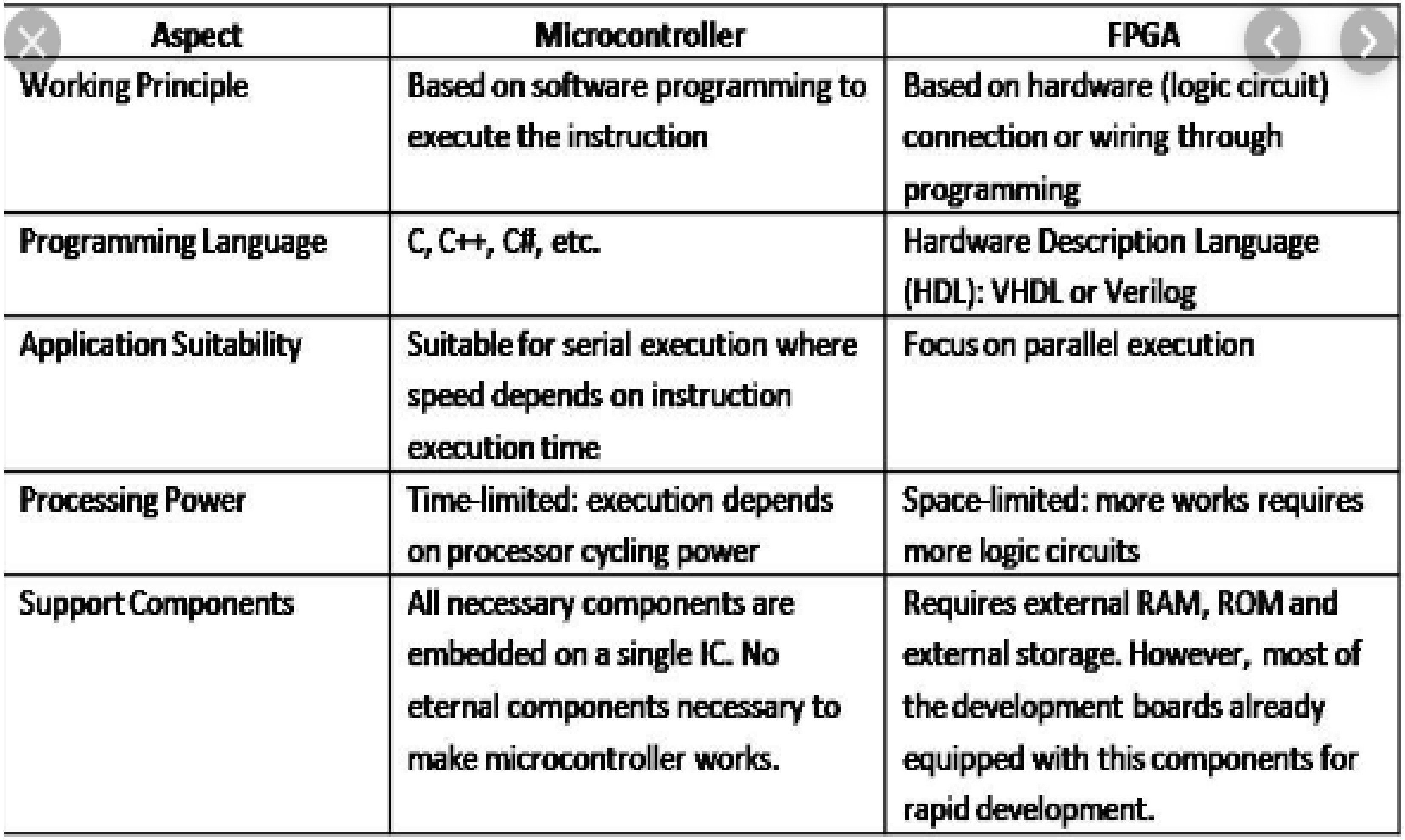
- Rely on S/w to implement functions.
- Slower
- More power consumption than custom chips

FPGA /ASIC-

Application specific integrated circuit

- Not custom parts
- Slower
- Burn more power than custom logic.
- Expensive

Difference between



Aspect	Microcontroller	FPGA
Working Principle	Based on software programming to execute the instruction	Based on hardware (logic circuit) connection or wiring through programming
Programming Language	C, C++, C#, etc.	Hardware Description Language (HDL): VHDL or Verilog
Application Suitability	Suitable for serial execution where speed depends on instruction execution time	Focus on parallel execution
Processing Power	Time-limited: execution depends on processor cycling power	Space-limited: more works requires more logic circuits
Support Components	All necessary components are embedded on a single IC. No eternal components necessary to make microcontroller works.	Requires external RAM, ROM and external storage. However, most of the development boards already equipped with this components for rapid development.

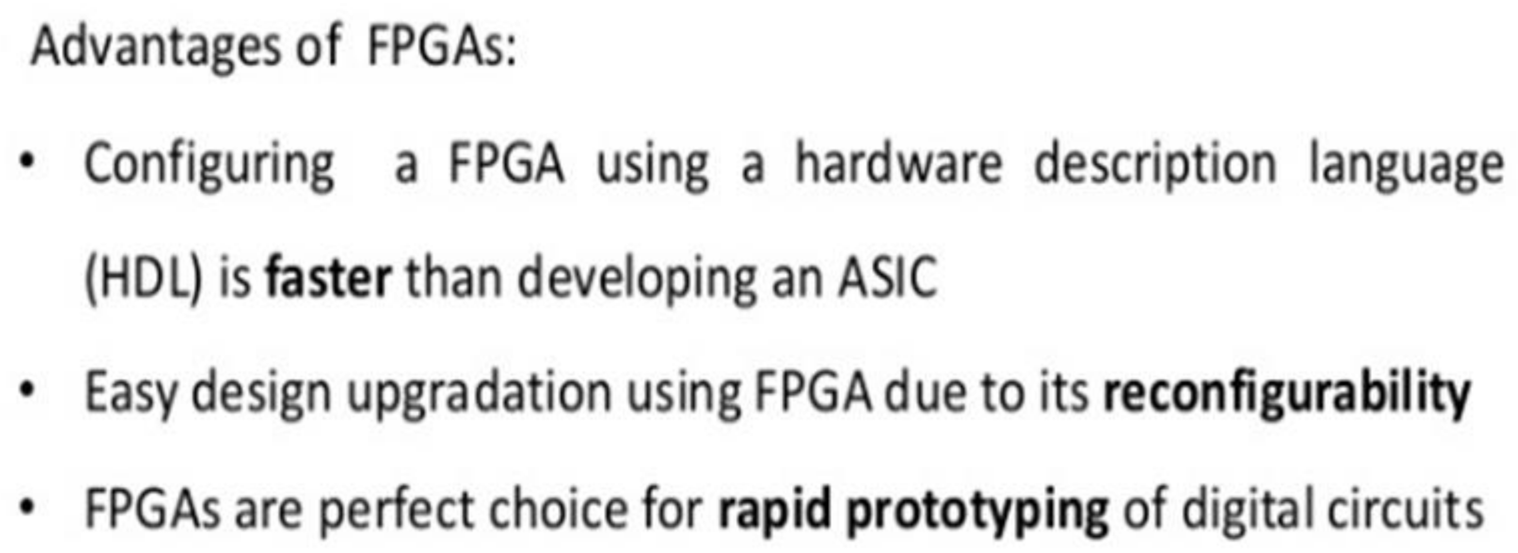
FPGA

Benefits

- Better performance than General Purpose Processors
  - Even though clock frequency may be 50-200MHz
- Easier to design than Full Custom
- Easier to test than Full Custom
- Good for prototyping Full Custom

Drawbacks

- Not a Production-Grade piece of hardware
  - No application uses 100% of everything available on the FPGA
  - Some FPGA's reset on power loss, and need to be reprogrammed



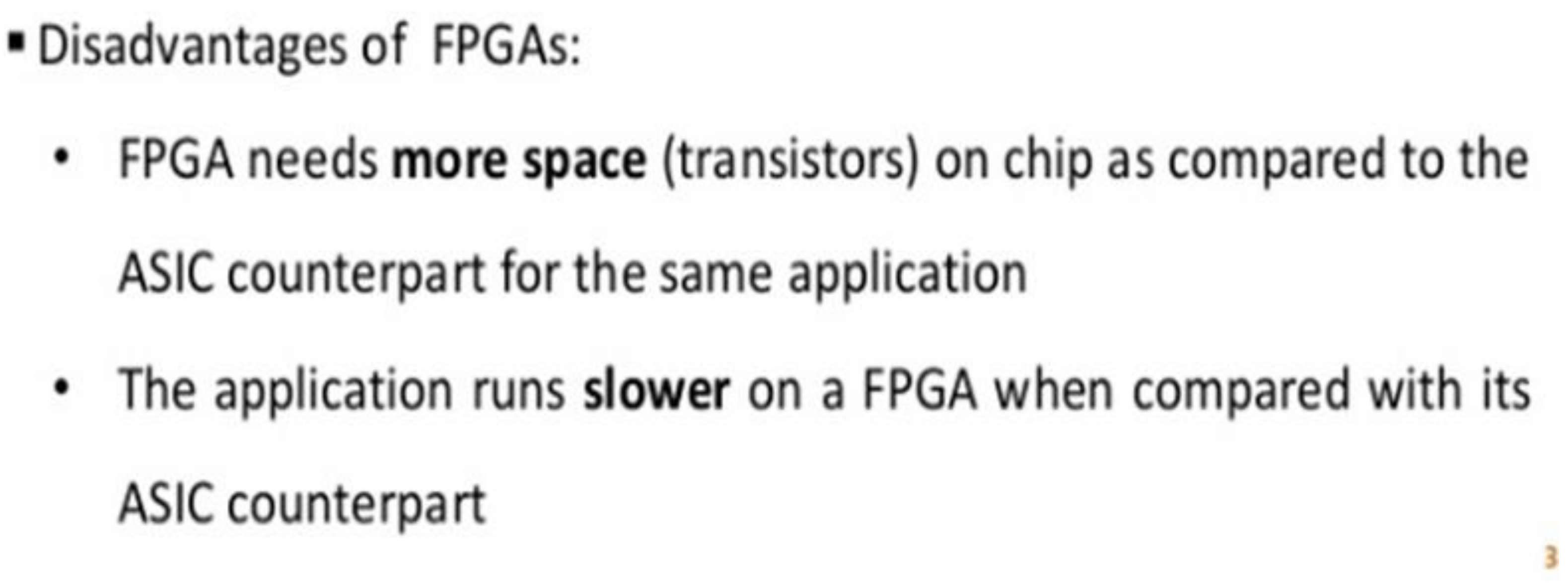
Advantages of FPGAs:

- Configuring a FPGA using a hardware description language (HDL) is **faster** than developing an ASIC
- Easy design upgradation using FPGA due to its **reconfigurability**
- FPGAs are perfect choice for **rapid prototyping** of digital circuits

when FPGA is used in final design, the jump from prototype to product is much smaller and easier to negotiate.

The same FPGA can be used in several diff designs reducing cost.

The design can be programmed into FPGA and tested immediately.



- Disadvantages of FPGAs:
  - FPGA needs **more space** (transistors) on chip as compared to the ASIC counterpart for the same application
  - The application runs **slower** on a FPGA when compared with its ASIC counterpart

3

Alternative to FPGA- ASIC – Application Specific IC

- ASIC is designed to implement a particular logical function/ designed for particular purpose.
- ASIC uses predesigned layouts for logic gates.
- Design of ASIC ----Mask ---- used to fabricate IC.
- ASIC – after months – fabricated----tested.
- Consumes low power.
- Faster than FPGA
- Cheaper.
- FPGAs use more transistors for a given function than ASICs, but an FPGA can be designed in days compared to the year long design cycle required for garden variety FPGAs.

FPGA -Based system design

Goals and techniques

The logical function to be performed is only one of the goals that must be met by an FPGA or any digital system design. Many other attributes must be satisfied for the project to be successful.

1) **Performance :** Logic must run at desired rate.It is measured in many ways, such as throughput and latency. Clock rate often measures performance.

2) **Power / Energy : C**hip must run within an energy or power budget. Energy consumption is very imp in battery powered system.

- **3) Design time :** FPGAs have standard parts, have several advantages in design time. As prototypes ,programming should be very quick. It effects design cost also.

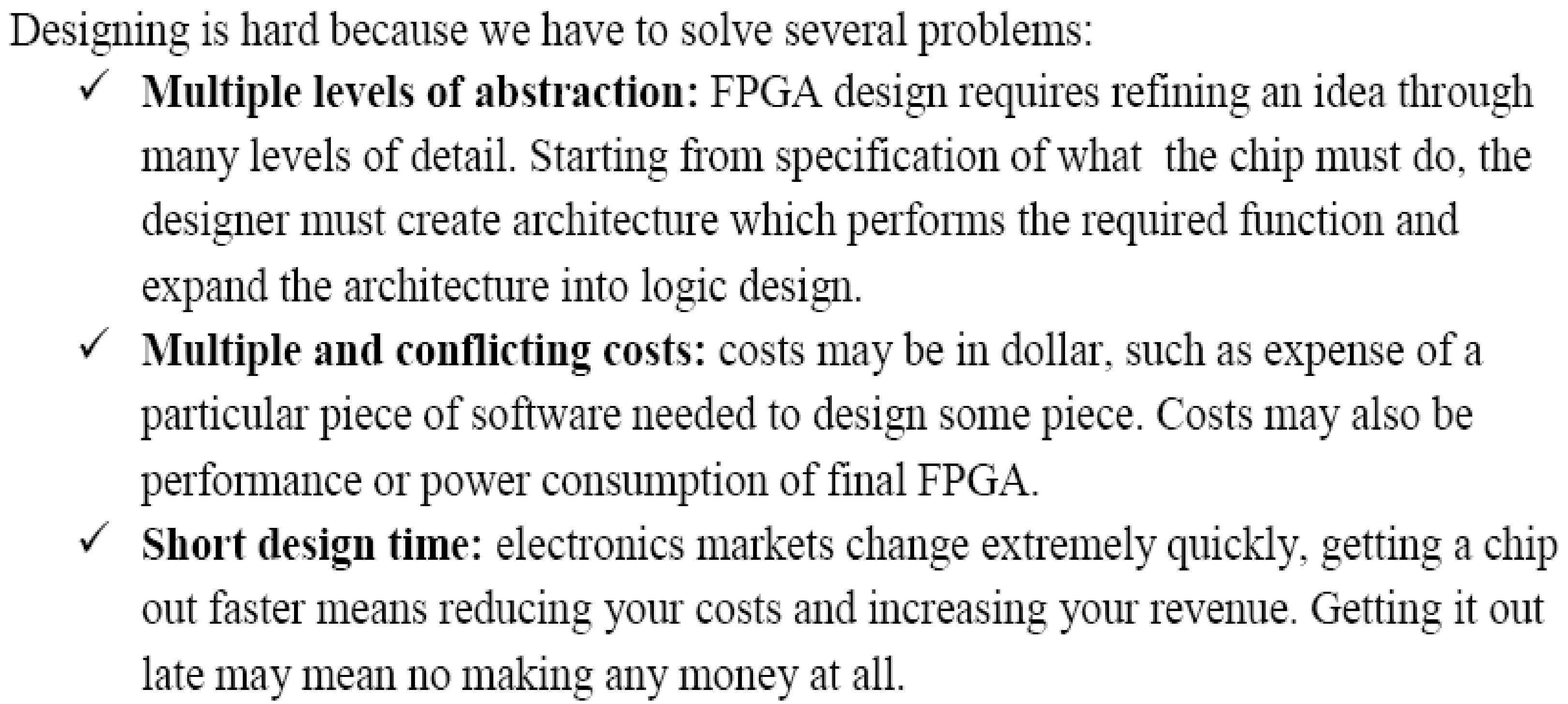
4) **Design cost : D**esign time is one important component in design cost, but other factors such as required support tools may be considered. FPGA tools are less expensive than custom VLSI tools.

5) **Manufacturing cost** : Its cost of replicating the system many times. FPGAs are generally more expensive than ASICs due to overhead of programming. But the fact that they are standard parts helps to reduce their cost .

Design Challenges

Design is particularly hard because we must solve several problems.

- **Multiple levels of abstractions**
- **Multiple and conflicting costs**
- **Short design time**

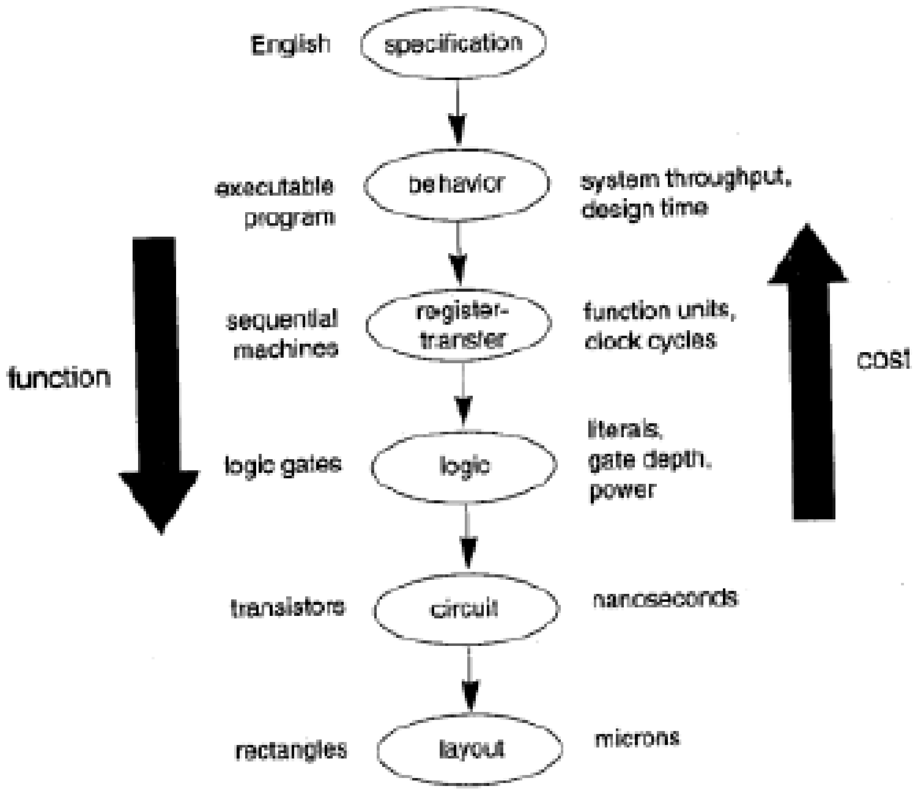


Designing is hard because we have to solve several problems:

- ✓ **Multiple levels of abstraction:** FPGA design requires refining an idea through many levels of detail. Starting from specification of what the chip must do, the designer must create architecture which performs the required function and expand the architecture into logic design.
- ✓ **Multiple and conflicting costs:** costs may be in dollar, such as expense of a particular piece of software needed to design some piece. Costs may also be performance or power consumption of final FPGA.
- ✓ **Short design time:** electronics markets change extremely quickly, getting a chip out faster means reducing your costs and increasing your revenue. Getting it out late may mean no making any money at all.

1.2 Design Abstraction

- ✓ Design abstraction is critical to hardware system design.
- ✓ Hardware designer use multiple levels of design abstraction to manage the design process and ensure that they meet major design goals, such as speed and power consumption.
- ✓ Design abstraction of the FPGA is as shown in below figure.
  - **Behavior:** Explains detailed, executable description of what the chip should do, but it won't describe how it should do it. Example a C program will not mimic the clock cycle-by- clock cycle behavior of the chip, but it will allow us to describe in detail what needs to be computed, error and boundary conditions etc.
  - **Register transfer:** The system's time behavior is fully specified- we know the allowed input and output values on every clock cycle but the logic isn't specified as gates. The system is specified as Boolean functions stored in abstract memory elements. Only delay and area estimates can be made from Boolean functions.
  - **Logic:** The system is designed in terms of Boolean logic gates, latches and flip-flops.
  - **Configuration:** The logic must be placed into logic elements around FPGA and the proper connections must be made between those logic elements. Placement and routing performs these important steps.



Abstraction of FPGA design

- Design always requires working down from the top of abstraction hierarchy and up from least abstract description. Work must begin by adding details to abstraction – top-down design add functional detail. Top–down design decisions are made with limited information.
- Bottom – up analysis and design percolates cost information back to higher levels of abstraction: for instance, we may use more accurate delay information from circuit design to redesign the logic. But most designs requires cycles of Top _Down design followed by bottom – up redesign.

Assignments:

Section 1.4.2 (page 31,32)

1.4.4 (37,38,39)

Designing is hard because we have to solve several problems:

- ✓ **Multiple levels of abstraction:** FPGA design requires refining an idea through many levels of detail. Starting from specification of what the chip must do, the designer must create architecture which performs the required function and expand the architecture into logic design.
- ✓ **Multiple and conflicting costs:** costs may be in dollar, such as expense of a particular piece of software needed to design some piece. Costs may also be performance or power consumption of final FPGA.
- ✓ **Short design time:** electronics markets change extremely quickly, getting a chip out faster means reducing your costs and increasing your revenue. Getting it out late may mean no making any money at all.

FPGA FABRICS - Basic structure

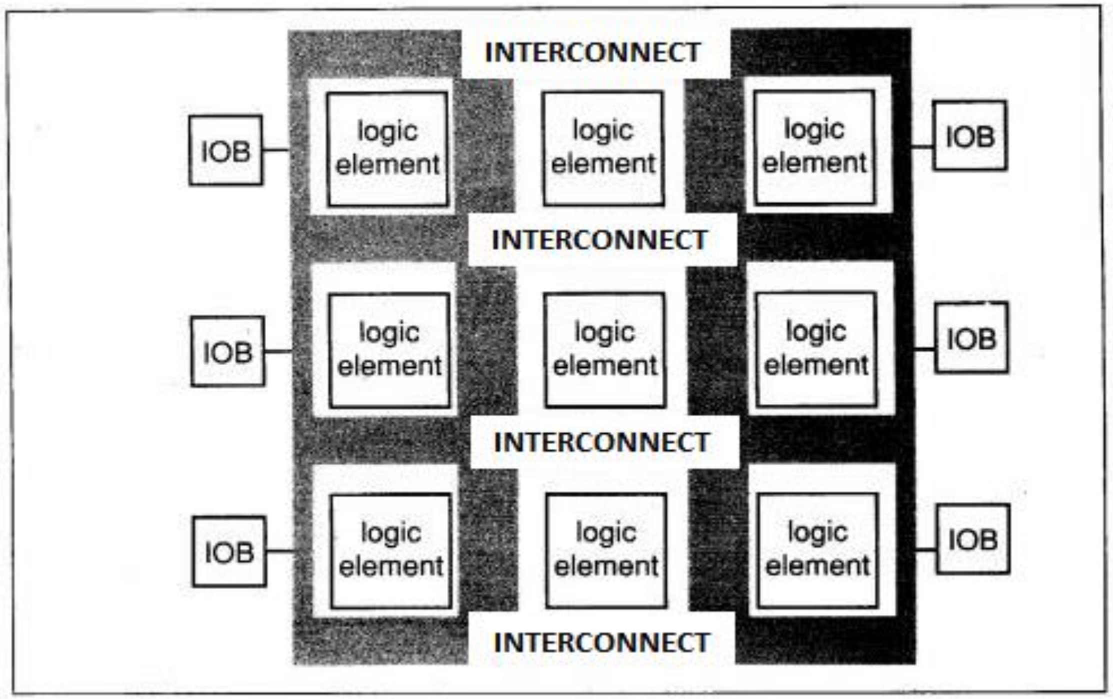
3.2 - FPGA Architecture

- FPGA consists of three major elements.

Combinational Logic

Interconnect

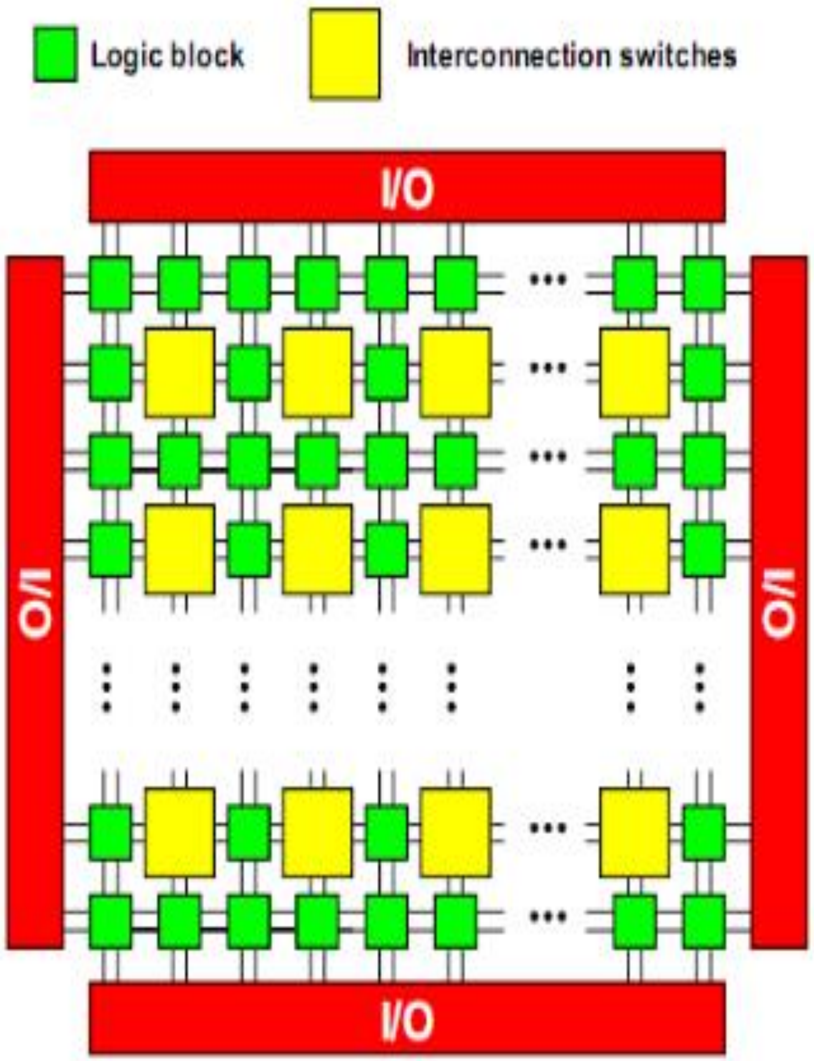
I/O pins



Generic structure of an FPGA fabric.

FPGA building blocks:

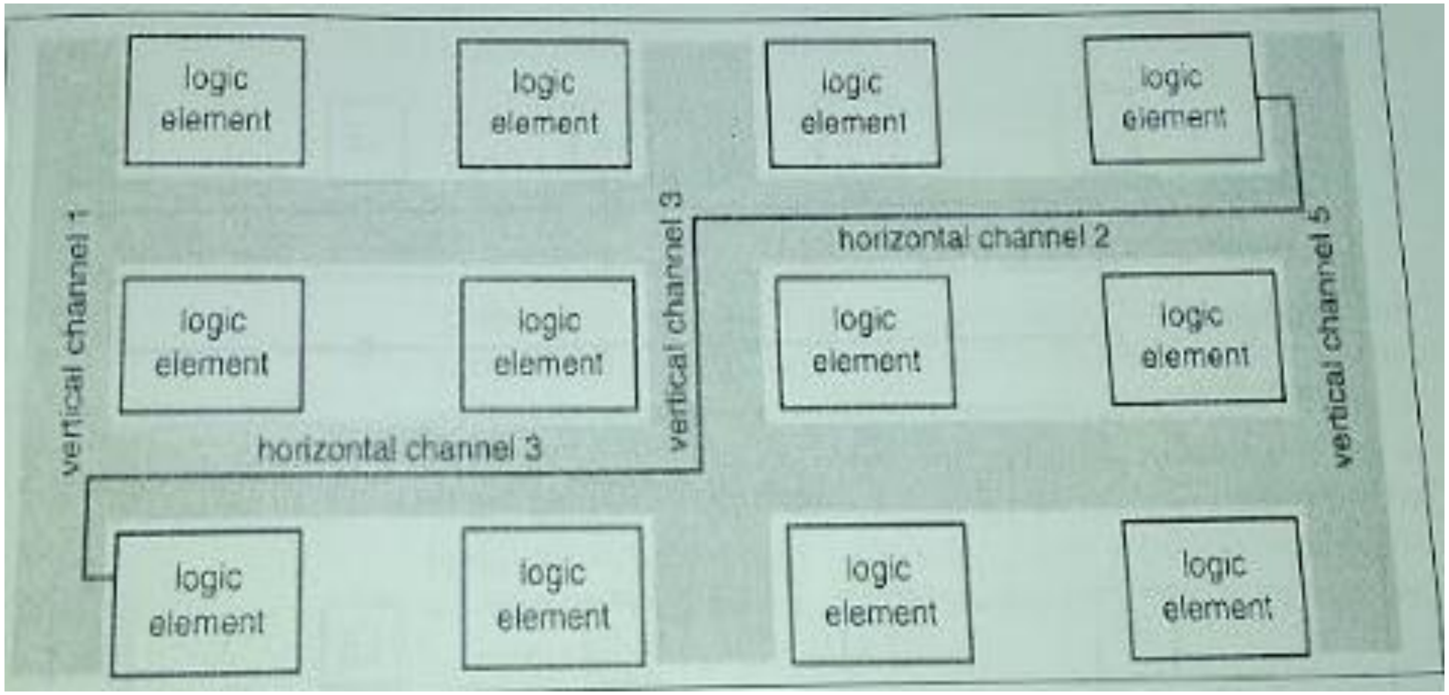
- ***Programmable logic blocks***
  Implement combinatorial and sequential logic
- ***Programmable interconnect***
  Wires to connect inputs and outputs to logic blocks
- ***Programmable I/O blocks***
  Special logic blocks at the periphery of device for external connections



FPGA Architecture

- The combinational logic is divided into small units which is known as logic elements(LE) or combinational logic blocks(CLB).
- LE or CLB usually forms the functions of several logic gates.
- Interconnections between these logic elements are made using programmable interconnects.
- This interconnects are logically organized into channels or other units.
- FPGA offers several interconnects depending on the distance between CLB’s that are to be connected: clock signals are provided with their own interconnection networks.
- I/O pins are referred as I/O blocks (IOB’s).
- These are generally programmable for inputs or outputs and often provides other features such as low power or high speed connection.

FPGA Interconnect s



The FPGA designer should rely on pre-designed wiring, unlike custom VLSI designer cannot design wires as needed.

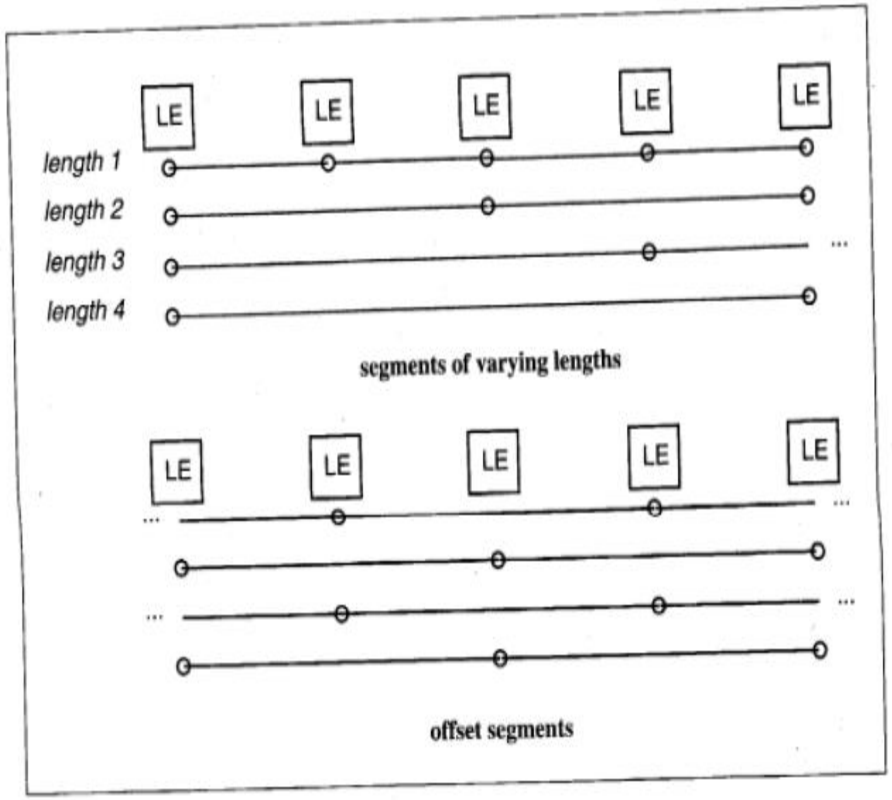
The interconnection system of FPGA is one of the most complex aspects because wiring is global property of logic design.

Connection Paths

- Connection between logic elements requires complex paths since LE’s are arranged in two dimensional structure as shown in below in previous slide.
- We therefore need to make connections not just between LEs and wires but also between the wire themselves.
- Wires are typically organized in **wiring channels or routing channels** which runs horizontally and vertically throughout the chip.
- Each channel contains several wires ; human designer or program chooses which wire will carry signal in each channel.
- Connection must be made between wires to carry signal from one point to another.

- Ex: Net in figure starts from output of LE in upper-right-hand corner travels down **vertical channel 5** until it reaches **horizontal channel 2,** then moves down **vertical channel 3** to **horizontal channel 3**, then it uses **vertical channel 1** to reach the input of **LE at Lower-left-corner**.

Segmented wiring



top signal of **length 1** goes to next LE,
the second signal goes to second LE and so on.
This organization is Known as **segment wiring structure**.

FPGA channel must provide wires of variety of lengths for designer to make all the required connection between logic elements.
Since LE's are organized in regular array, we can arrange wires going from one LE to another.
Figure above shows connections of varying length as measured in unit LE's:

FPGA Configuration.

- All FPGA’s need to be programmed or configured.
- **Three major circuit technologies for configuring an FPGA**:

a) SRAM

b) Antifuse

c) flash.

- No matter which circuits we use, all the major elements of FPGA - Logic elements, interconnect and i/O pins needs to be confugured.

4.8 Physical Design of FPGA

Configuring an FPGA means placing the logic into LE in fabric and choose paths .Delay and Energy consumption of LE plays a major role.

Physical design is divided into two major phases.

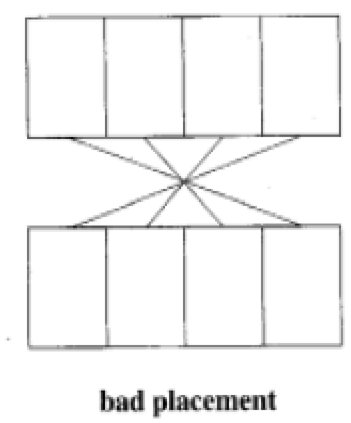
- **Placement:** it determines the position of logic elements and I/O pads.
- **Routing**: selects the paths for connection between logic elements and I/O pads.
- These two phases interact – one placement of the logic may not be routable whereas a different placement of the same logic can be routed. But this division allows us to make physical design problem for tractable.
- We use several different metrics to judge the quality of a Placement or Routing.

- **Size matters** as we are concerned about whether we can fit complete design on to the chip.
- In FPGA, **size** is closely tied with routing.The number of logic elements required is determined by logic synthesis. If we cannot find legal routing for given placement, we may need to change placement.
- **Delay** is also critical measure in most design. A long delay is not critical, whereas a relatively short delay path that is critical must be carefully considered.
- *Detailed delay characteristics* are somewhat expensive to calculate, so tools are used for the same.

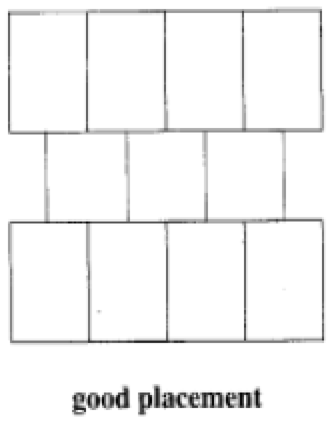
Placement

- The separation of placement and routing raises an important problem: how to judge the quality of placement?
- We cannot judge the placement quality by 2 measures (area and delay).
- We cannot afford to execute a complete routing for every placement to judge its quality:
- There fore, *we need some metric which* estimates the quality of routing.
- Different algorithms use different metrics, but few simple metrics suggests important properties of placement algorithms.

Two types of placements



Longer rat's nest interconnections



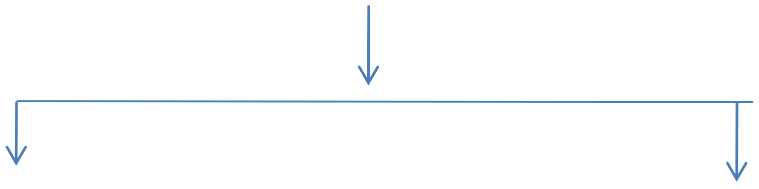
Shorter rats nest interconnections

•One way to judge the area of layout before routing is to measure total distance between interconnection.

•Of course, we can't measure total wire length without routing the wires, but we can estimate length of a wire by measuring distance between pins it connects.

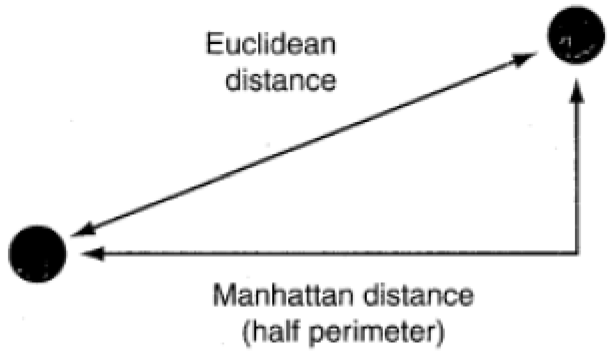
When straight lines between connected pins are plotted ,***results is known as rat's nest plot.*** ***Larger connections means larger layouts.***

SEVERAL WAYS TO MEASURE DISTANCE



Euclidean distance is the direct line between the two.

Manhattan distance, also known as half perimeter



Clustering Vs Partitioning

Many algorithms for there for partitioning

Bottom up

These methods are generally referred as **clustering methods**. These cluster together nodes to create partitions.

Fig 4-44 pp-289

Top Down

Top Down methods divide nodes into groups that are than further divided. These methods are known as partitioning methods.

- **Fig 4-45 pp290**

Placement by clustering .

- Page 292 second pyara

Placement by partitioning

Partitioning process for mini cut bisecting criterion is as shown.

Here nodes are components & edges are connections b/w components.

Goal : to separate the graph nodes into 2 partition with nearly equal nodes with minimum edges.

In fig ,we can see 5 wires crossing partition boundary.

When we swapping ,2 nodes reduces the net cut count to one. Thus a significant reduction.

- 4-46 pp291

Routing

- Routing selects paths for connections that must be made between logic elements and I/O pads.
- In an FPGA, the interconnection resources are predetermined by the architecture of the FPGA fabric.
- A connection must be made by finding sequence of routing resources, all of which unused and which share connections such that continuous path can be made from source to sink.

Routing is generally divided into two phases:

- **Global Routing** selects general path through the chip but does not determine exact wire segments to be used.
- **Detailed Routing** selects the exact set of wires to be used for each connections.

Costs in routing

Routing has two major cost metrics:

wire length

Delay.

Wire length approximates utilization of routing resources. We may not use more routing resorces for wires and

D**elay** may be measured by looking at the delay on paths with largest number of levels of logic

or

by looking to nets whose delay is close to maximum allowed value .

Global routing

- The principal job during global routing of FPGAs is to balance the requirements of various nets.
- Nets are routed one at a time, so the order in which nets are routed affects final result.
- *Net may have one of the two problems*:

1) it may not be routable because there is no room available to make connection or

2) it may take a path that incurs too much delay.

- These problems are harder to solve in FPGAs than in custom chip designs because routing resources are pre determined .
- Connections must be composed of re-designed paths that may include larger wire segments.
- Most FPGA have diff categories of wiring ,each with very diff CH .

Wire ordering

- Many ways have developed to determine the order in which wires are routed.
- A good heuristic for initial ordering is to route most delay critical nets first:
- one may also want to start with large fanout nets since they consume many routing resources.
- In general , a wire may be routed more than once before it finds its final route. ***Ripup*** *and* ***reroute*** *is one simple strategy for choosing the order in which to route nets.*