Module 4

Subsystem Design

- This deals with designing of subsystems, which is a small part in a larger system (leaf cell).
- The most basic leaf cell of any digital system is the logic gates and these are seen in different technologies like nMOS, CMOS and BiCMOS.
- While designing high regularity should be maintained. This indicates detailed designing of few leaf cells which can be replicated and interconnected to form system.

Architectural Issues:

- As the complexity of a system increases, the design time also increases. Thus while designing we have to adopt those design methodologies which allows handling complexity with reasonable time and reasonable amount of labor.
- The following are the guidelines/architectural issues that needs to be considered while designing of the system.
    1. Define the requirements carefully and properly.
    2. Partition the overall architecture into appropriate subsystems.
    3. Communication paths should be carefully selected in order to develop sensible interrelationships between the subsystems.
    4. Draw the floor plan of how the system is to map onto the silicon.
    5. Aim for regular structures so that design is largely a matter of replication.
    6. Draw suitable stick or symbolic diagrams of the leaf-cells of the subsystem.
    7. Convert each cell into layout.
    8. Carry out design rule check carefully and thoroughly.
    9. Simulate the performance of each cell/subsystem.

Note: alternate between 2, 3 and 4 can be done as required.

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

- Partitioning of the system so that there are clear subsystems with minimum interdependence and minimum complexity of interconnection between them.
- The design within the subsystems should be simple which helps in cellular design concept. This helps the systems in having few standard cells which are replicated to form high regular structures.

For designing digital systems in MOS technology there are two ways of building the circuits. They are

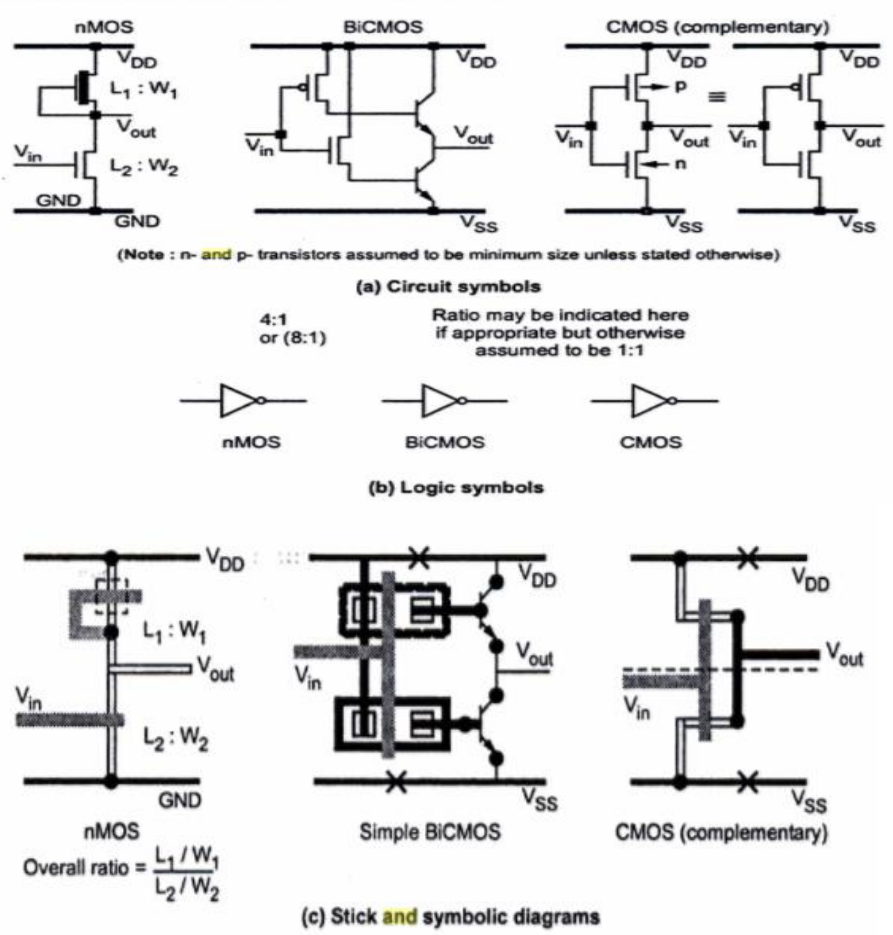
1. Switch logic
2. Gate (restoring) logic.

Gate (restoring) Logic:

- Gate logic is based on the general arrangement of inverter as it is the simplest gate. The inverter can be constructed with nMOS, CMOS or BiCMOS technology. Similar to this NAND and NOR can be constructed. Also AND and OR can be constructed with an inverter for NAND and NOR respectively.
- In nMOS technology the L:W ratio must be considered to get desired Z_{pu}/Z_{pd} ratio.

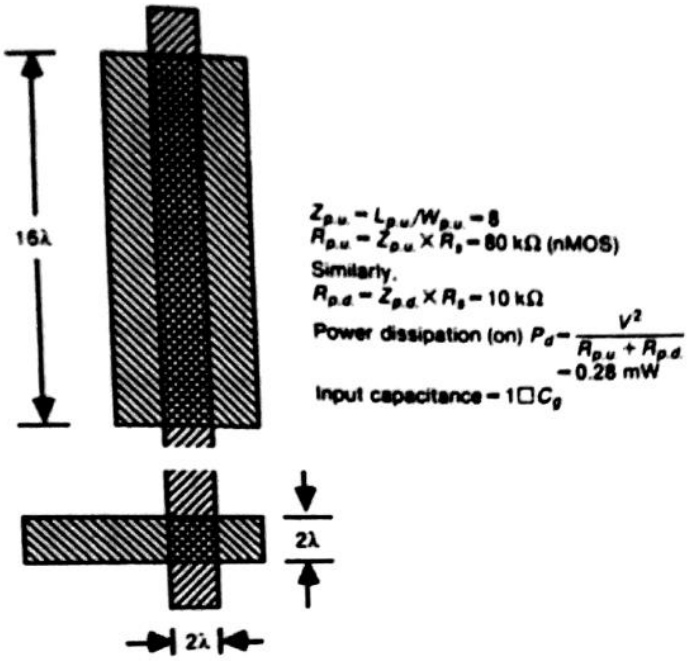
Inverter:

- The inverter circuit in different technology, corresponding stick diagram and symbolic diagram is shown in the Fig.

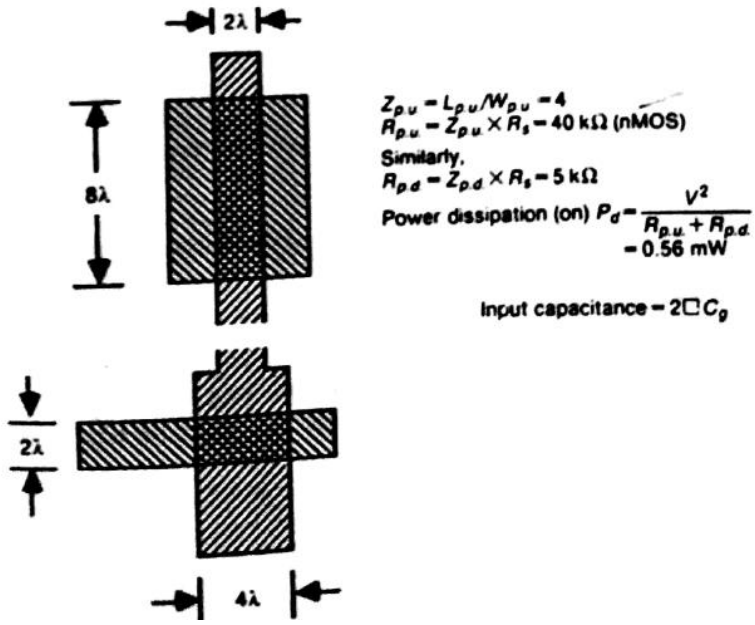


nMOS inverter

- In nMOS inverter the Z_{pu}/Z_{pd} ratio and the channel length to width ratio of each transistors shown.
- With different ratios of pull-up and pull-down several approaches are available.
- With different approaches it results in effecting the power dissipation P_d, area occupied, resistance and the capacitance values.
- The effect of power dissipation and capacitance for different ratios can be seen in the example for nMOS.



nMOS inverter with ratio 8:1, power dissipation is 0.28mW and input capacitance is 1□Cg.



nMOS inverter with ratio 4:1, power dissipation is 0.56mW and input capacitance is 2□Cg.

CMOS inverter

- In CMOS there is static current and thus there is no power dissipation but only at switching there is power dissipation. The power dissipation for fast CMOS logic circuits are considered.

Two-input nMOS, CMOS and BiCMOS NAND Gates:

- Two input NAND gate arrangement is shown in different technologies along with their symbolic representation and stick diagram.

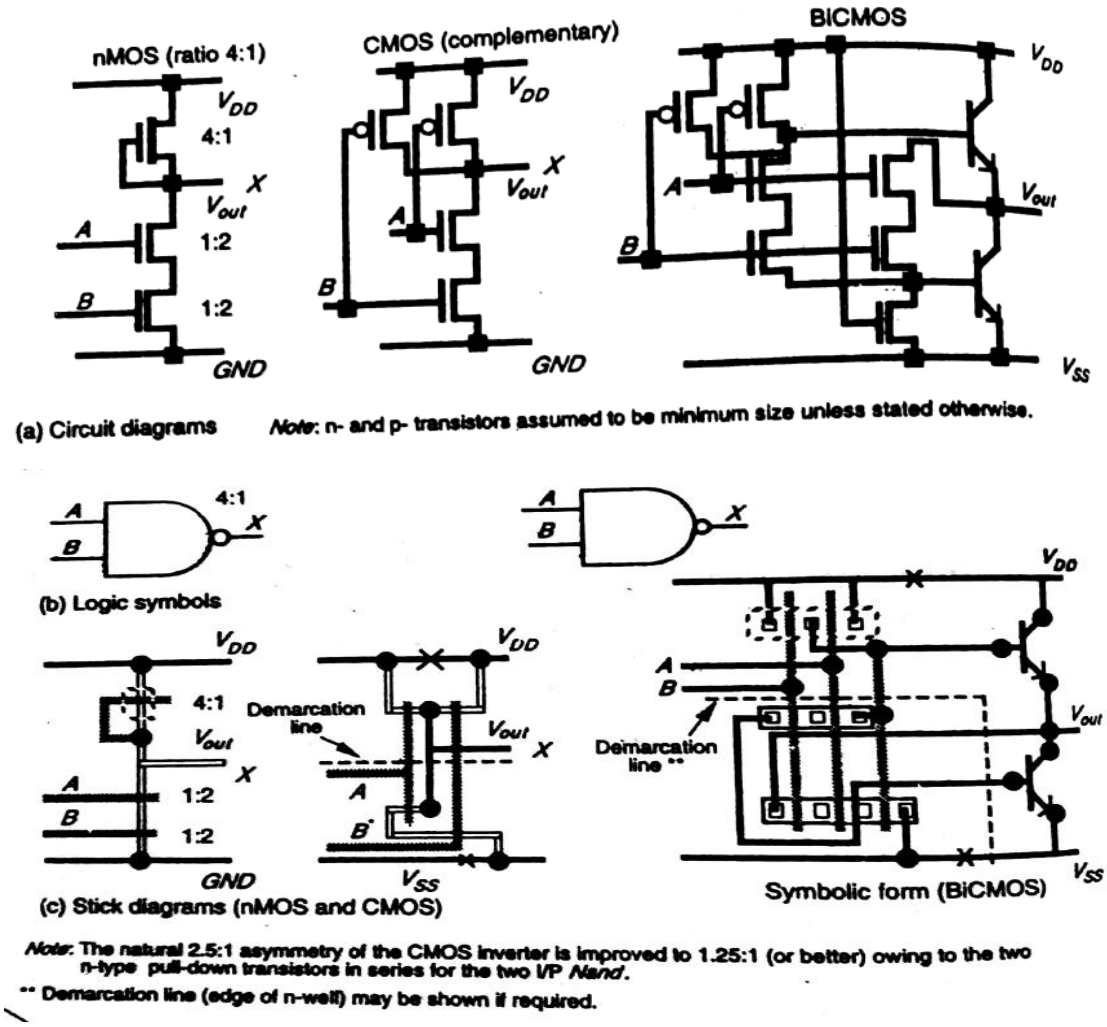


Fig. Two input NAND gate circuit, symbol and stick diagram

nMOS NAND gate

- For nMOS L:W ratio must be carefully chosen so that desired overall Z_{pu}/Z_{pd}
- Here Z_{pd} is contributed by the input transistors connected in series and Z_{pu} is contributed with enhancement mode transistor.
- It is obtained that ratio between Z_{pu} and sum of all pull-down Z_{pd} must be 4:1
- nMOS NAND gate geometry reveals two significant factors.
    - nMOS NAND gate area is greater than that of nMOS inverter. In the pull-down network as the transistors are added in series to provide number of inputs. As the number of inputs are added, corresponding adjustment has to be made in the length of the pull-up transistor in order to maintain the required ratio.
    - The delay in nMOS NAND gate increases in a direct proportion to the number of inputs added. If there are n inputs then the length and resistance of the pull-up transistor must be increased by a factor on n in order to maintain the correct ratio. The delay observed in NAND gate is given by

$$\tau_{Nand} = n\tau_{inv}$$

Where n is the number of inputs and τ_{inv} is the delay related to nMOS inverter.

- ✓ With these properties the nMOS NAND gate is used only when it absolutely necessary and the number of inputs are restricted.

CMOS NAND gate

- No restriction of ratio is seen, but asymmetry is the problem seen in CMOS.
- In order to keep the transfer characteristics symmetrical at $V_{DD}/2$ it is necessary to perform some adjustments in the transistor geometry.

BiCMOS NAND gate

- BICMOS nand gate is complex and difficult for fabrication
- It has considerable load-driving capabilities and useful when driving large capacitive loads or when there is a large fan-out.

Two-input nMOS, CMOS and BiCMOS NOR Gates:

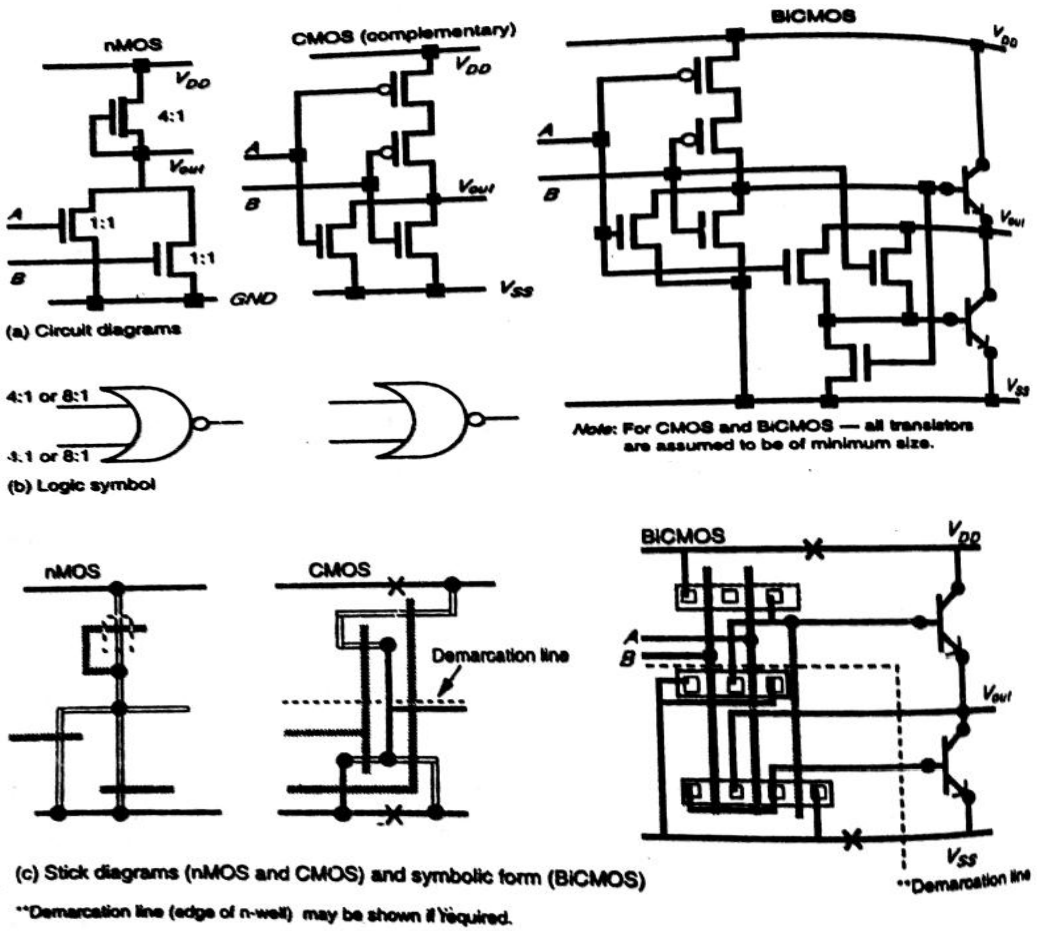


Fig. Two input NOR gate circuit, symbol and stick diagram

- Two input NOR gate arrangement in different technologies along with stick diagram is shown in the above figure.

nMOS NOR gate

- In 2 input NOR gate as one end of the gate provide a path to ground (when it is turned on) from the pull-up network. Thus any one conducting pull-down gate will give characteristics to that of inverter. Hence the ratio of the inverter can be maintained. This is applicable to any number of inputs.
- The area consumed by NOR gate reasonable compared to NAND because with the increase in the number of inputs it is not affecting the dimension of the pull-up device(as seen for NAND gate)
- NOR gate is fast as inverter and is preferable when choice is possible.
- The ratio of Z_{pu} and Z_{pd} depend on the source which is driving the NOR gate, 4:1 ratio if driven by an inverter and 8:1 if driven by one or more pass transistor.

CMOS NOR gate

- CMOS NOR has p-based-transistor network in the pull-up to implement logic 1. Similarly in the pull-down network has n-based transistor to implement logic 0.
- In the pull-up network the p-structure consists of p-transistors connected in series for each input. This results in increasing the resistance.
- Also asymmetry in rise-time and fall-time on capacitive loads is increased and this results in shifting the transfer characteristics which will reduce the noise immunity.
- Thus CMOS NOR gates with more than 2 inputs require adjustment of p or n transistor geometries.

BiCMOS NOR gate

- BICMOS NOR gate is complex and difficult for fabrication
- It has considerable load-driving capabilities and useful when driving large capacitive loads or when there is a large fan-out.

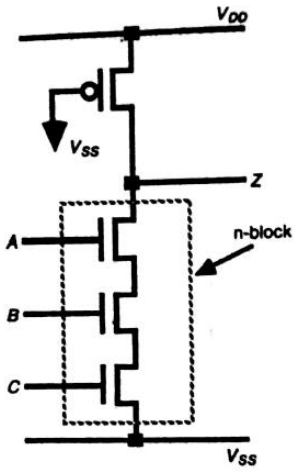
Other Forms of CMOS Logic:

With the availability of both n and p transistors it is possible for the designer to exploit alternatives to inverter based CMOS logic.

Pseudo-nMOS logic

- In nMOS circuit if the pull-up network with depletion mode transistor is replaced by pMOS transistor with its gate connected to V_{ss}. This structure is called as pseudo-nMOS logic.
- This logic helps in reducing the number of transistors seen in the CMOS logic seen for pull-up network.
- Fig. shows 3 input NAND logic using pseudo-nMOS logic.
- As the pMOS is connected to V_{SS} it can be either in saturation or active region and the status of nMOS at pull-doun network depend on the state of input.
- When all the inputs are zero, pMOS will be on and the output is pulled up to V_{DD} and with other data pMOS acts as a current source (saturation). The output is now

the product of the resistance of the pull-down network and current of the pull-up network.



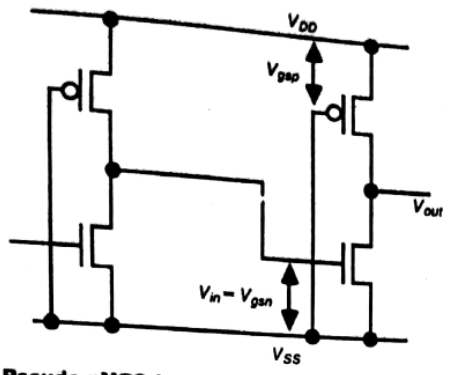
Pseudo-nMOS *Nand* gate.

- In order to obtain required ratio, the arrangement considered is pseudo-nMOS inverter being driven by another pseudo-inverter.
- For analysis taking the condition $V_{in} = V_{DD}/2$, in this condition nMOS device is operating in saturation ($0 < V_{gsn} - V_{tn} < V_{dsn}$) and pMOS is operating in linear /resistive region ($0 < V_{dsp} < V_{gsp} - V_{tp}$).
- Equating the currents of nMOS and pMOS and suitable arrangement, we obtain the equation as

$$V_{inv} = V_{tn} + \frac{(2\mu_p/\mu_n)^{1/2}[(-V_{DD} - V_{tn})V_{dsp} - V_{dsp}^2]^{1/2}}{(Z_{p.u.}/Z_{p.d.})^{1/2}}$$

$$Z_{p.u.} = L_p/W_p$$

$$Z_{p.d.} = L_n/W_n$$



Pseudo-nMOS inverter when driven from a similar inverter.

$$V_{inv} = 0.5V_{DD}$$
$$V_{in} = |V_{tp}| = 0.2V_{DD}$$
$$V_{DD} = 5\ V$$
$$\mu_n = 2.5\ \mu_p$$

We obtain

$$\frac{Z_{p.u.}}{Z_{p.d.}} = \frac{3}{1}$$

- Thus it can concluded as:
  1. As the sheet resistance of the channel of pull-up network (PUN) is about 2.5 times that of pull-down network (PDN) also the ratio of PUN and PDN is 3:1. With this the pseudo-nMOS exhibits resistance of about 85Kohm compared to resistance of 50Kohm with nMOS device. This helps in reducing the power dissipation up-to 60% in comparison with device with nMOS device.
  2. With higher pull-up resistance the delay is larger by 8.5:5 than 4:1 nMOS inverter.

Dynamic CMOS logic:

- The logic is implemented as shown in the circuit.
- It has n block based on the logic to be implemented. Along with this there are 2 more transistors called 'precharge' transistor (pMOS) and 'evaluation' transistor (nMOS).
- Both these transistors are connected to common clock signal 'ϕ'

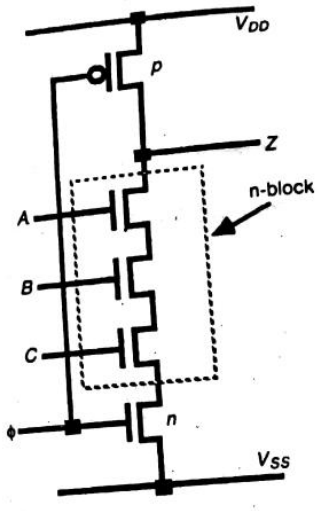


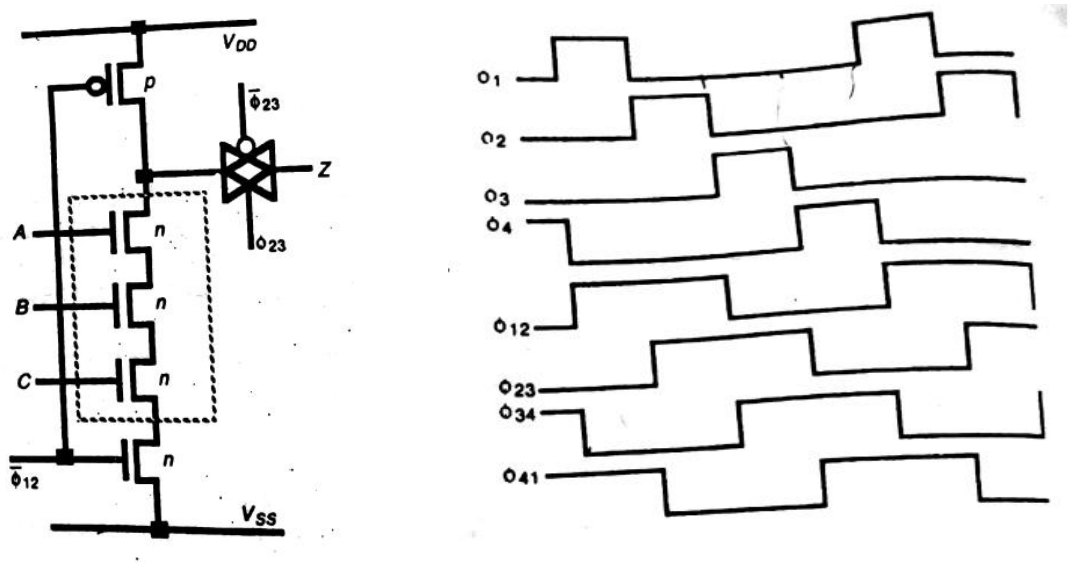
Fig. Dynamic CMOS logic

- During $\phi = 0$, pMOS is ON and nMOS is OFF. As the PDN is not in the ON state, due to the PUN output is pulled to V_{DD} i.e., output is precharged to V_{DD}.

- During $\phi = 1$, pMOS is OFF and nMOS block will be conducting as the evaluation transistor is ON. Thus load capacitor discharges depending on the inputs status of nblock.

Problems seen in dynamic logic

- ✓ Charge sharing problem occurs if the inputs changes during the on period of the clock.
- ✓ Single phase dynamic logic cannot be cascaded because of delay. This delay may result in wrong output results.



(a) (b)

Fig. (a) Possible clock arrangement with TG. (b) Possible 4ϕ and derived clocks

- To overcome the problem in dynamic logic derived clocks must be used.
- It uses 4 phase clock ϕ_1, ϕ_2, ϕ_3 and ϕ_3 to derived clocks ϕ_{12}, ϕ_{23}, ϕ_{34} and ϕ_{41} which are obtained by overlapping the clock signals.
- The circuit is modified with the inclusion of a transmission gate.
- The transmission gate samples the output during evaluate phase and holds the output state until the next stage evaluates the logic.
- For each stage different phase signals are provided for precharge and evaluate and also for the transmission gate.

Clocked CMOS (C^2MOS) logic:

- In this the given logic is implemented in both nMOS and pMOS transistors in the pull-up P block and pull-down N block respectively. This is seen in the fig.
- Along with this 2 additional transistors are included M1 and M2.
- M1 and M2 transistors are provided with clock and its compliment form. Thus both will be either ON or OFF simultaneously.
- When $\phi = 0$, both transistors are OFF and from either pMOS or nMOS network is not connected to the output. Hence output remains in previous state.
- When $\phi = 1$, both transistors are ON and the logic of the input will be evaluated.

- With additional transistors the area increases. Also with additional transistors causes the output rise time and fall time to increase and this results in more delay.

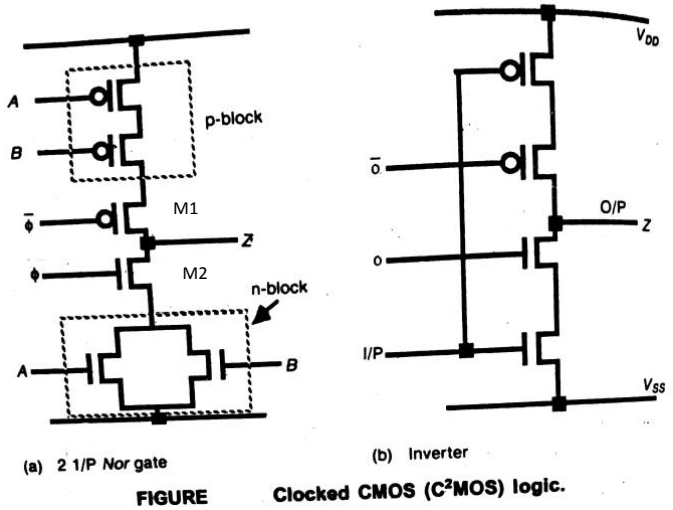


FIGURE Clocked CMOS (C^2MOS) logic.

CMOS Domino logic:

- The problem seen in dynamic logic of cascading can be overcome by using an inverter between the stages as shown in the figure below.
- It is an extension of dynamic CMOS logic.

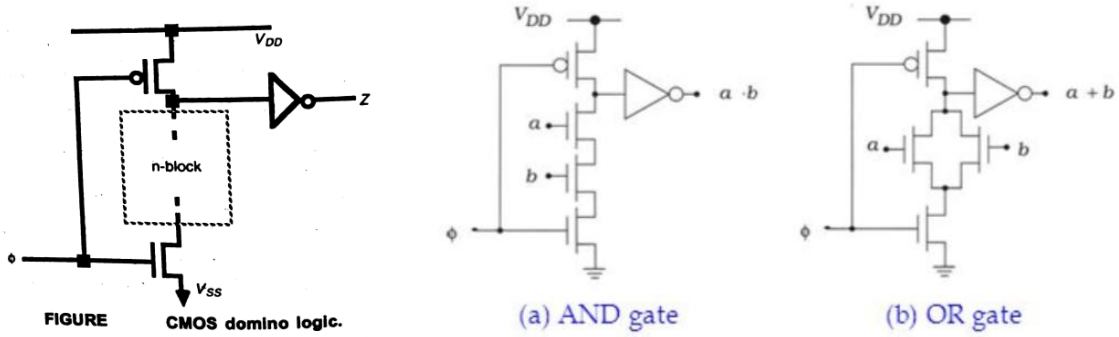


FIGURE CMOS domino logic.

Fig. Domino logic AND an OR gate

- When clk or ϕ = 0 (pre-charge phase), output node of dynamic CMOS logic is pre-charged to V_{DD} and this causes output of inverter low. This low output given to n-block (pull down network – PDN) of next stage causes the nMOS to be in the off state.
- When clk or ϕ = 1 (evaluation phase), the output node either discharges to 0 or remains in the same state depending on the state of the inputs.
- Depending on this output the next output also changes.
- As the output of one stage depends on the output of previous stage, this is similar to the domino effect. Thus this configuration is called as 'domino logic'
- Cascaded stages of domino logic circuit is shown in the Fig.

Note: The inverter output change at most can make transition only one transition i.e., from 0 to 1, but no 1 to 0 transition can take place in the evaluation phase.

Advantages of Domino logic:

- Structures have smaller area than the conventional CMOS logic.
- Parasitic capacitances are smaller so higher operating speeds can be obtained.
- Operation is free from glitches as only 1 to 0 transition is made.
- Only non-inverting structures are possible because of the presence of inverter buffer.

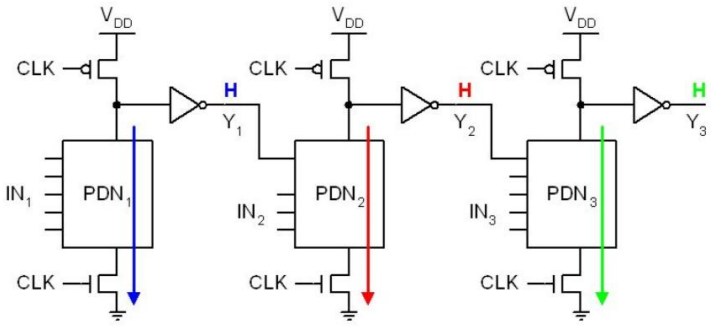


Fig. Cascaded Domino Logic

n-p CMOS logic

- Another variation of dynamic logic is the n-p CMOS logic as shown below.
- This logic is modification of domino logic where the inverter can be avoided and using p-block and n-block alternatively.

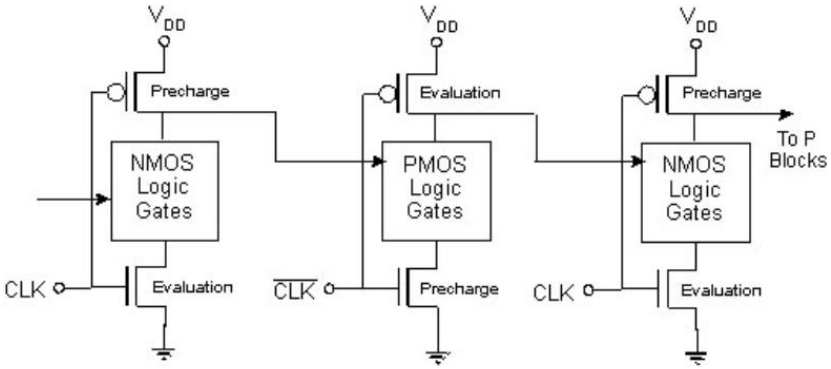


Fig. n-p CMOS logic

- When clk = 0, pre-charge phase, output of I block is HI and is given to p-block logic of II stage. This HI to pMOS will turn it off.
- The clk' (which is 1) is given to II stage and to the nMOS transistor. This causes the nMOS to turn on and output of II stage is goes low.
- This low given to next n-block will turn off the nMOS devices and this continues.
- Thus during pre-charge phase all the n-block and p-block logics will be off.
- When clk = 1, evaluation phase depending on the inputs at the p-block and n-block logic is evaluated and output each stage will remain HI of discharges

NOTE:

- Pseudo nMOS logic is preferred where high output is needed at most of the time. In this condition the static power dissipation is less and also propagation delay becomes

shorter. Common application of pseudo nMOS is in designing of address decoders for memory chips in ROMs.

- The advantage of dynamic CMOS logic over static CMOS or pseudo nMOS logic is that it reduce significantly the surface required to implement logic using complementary pMOS transistors. Also the switching speed is increased.
- Clocked CMOS logic is used for low power dissipation. This logic structure is used to incorporate latches or interface with other form of logic.

Switch Logic

- The switch logic is built on 'pass transistors' or on 'transmission gates'. Switch logic is fast for small arrays and draws no static current from the supply rails. Hence power dissipation of such circuits is small because current flows only on switching.
- Pass transistor can be used as a switch in passing the signals. Switch logic arrangements using basic OR and AND connections along with other arrangements is shown in the fig.

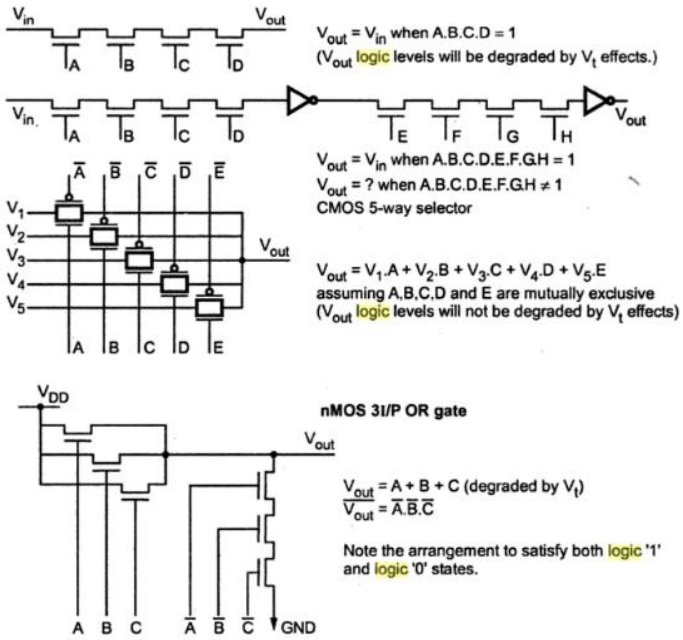


Fig. Some switch logic implementation

- Switch logic can be realized either using n or p pass transistors or from transmission gates.
- The transmission gate are complimentary switches made up of p-pass and n-pass transistor in parallel.
- Simple pass transistors suffer from undesirable threshold voltage effects which gives rise to loss of logic levels as shown in the Fig.

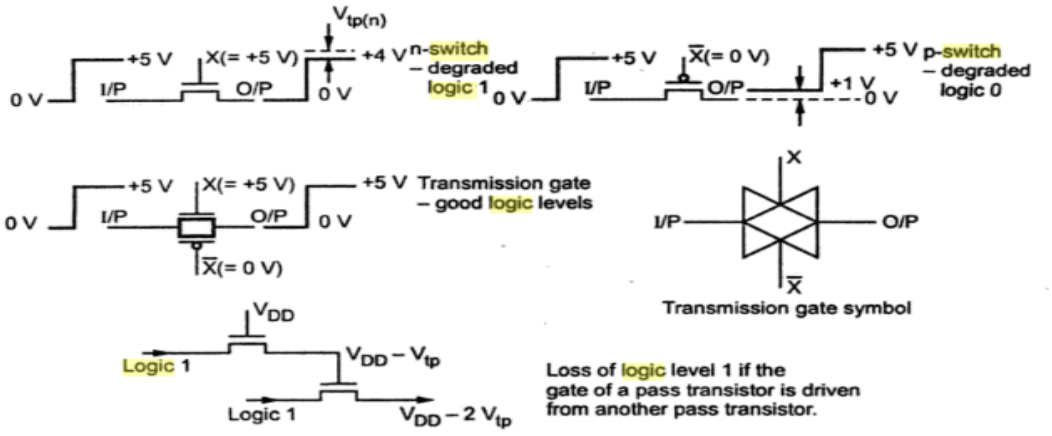


Fig I. Some properties of Pass Transistors and Transmission Gates

- Hence apparently complex transmission gate (TG) is preferred over simple n-switch or p-switch in CMOs applications.
  - The TG is free from any such degradation of logic levels
  - But it occupies more area
  - Requires complementary signals to drive it
  - The 'on' resistance of TG is lower than that of simple pass transistor switches
- Rules or restriction observed when using nMOS switch logic is that no pass transistor gate input must be driven through one or more pass transistors as this will degrade the output. This restriction is also shown in the Fig I.
- The logic levels propagated through pass transistor get degraded by threshold voltage effects. The signal out of the pass transistor T1 is not full logic 1 but rather a voltage that is one transistor threshold below the true logic 1(i.e., V_{DD}-V_{th}). Thus this degraded voltage does not permit the output of T2 to reach an acceptable logic 1 level.

Examples of Structured Design (Combinational Logic)

Multiplexer/data selectors

- A multiplexer or mux is a combinational circuits that selects several analog or digital input signals and forwards the selected input into a single output line.
- A multiplexer of 2^n inputs has n selected lines, are used to select which input line to send to the output.

Implementing 2:1 MUX

- MUX can be designed using various logic.
- The pass-transistor logic attempts to reduce the number of transistors to implement a logic by allowing the primary inputs to drive gate terminals as well as source-drain terminals.
- The implementation of a 2:1 MUX requires 4 transistors (including the inverter required to invert S), while a complementary CMOS implementation would require 6 transistors. The reduced number of devices has the additional advantage of lower capacitance.

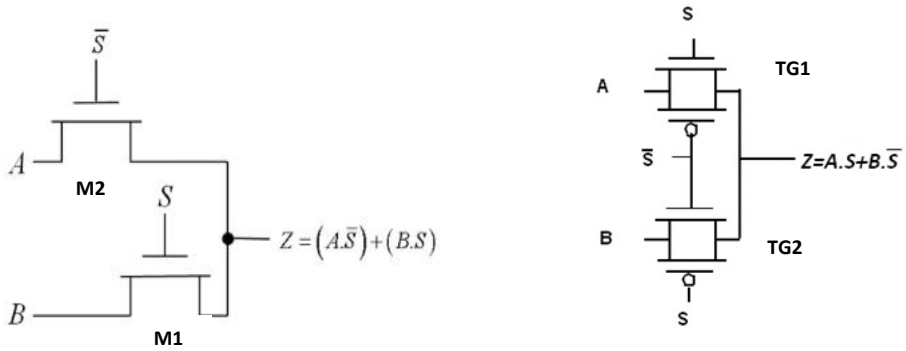


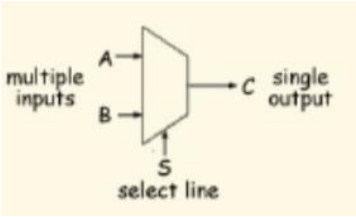
Fig. Pass transistor and transmission gate implementation of 2:1 MUX or data selector

For pass transistor logic

- When S = 0, transistor M1 is OFF and M2 is ON, thus output Z = AS'
- When S = 1, transistor M1 is ON and M1 is OFF, thus output Z = BS

For Transmission gate

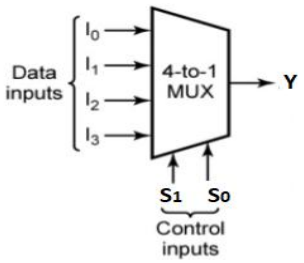
- The transmission gate acts as a bidirectional switch controlled by the gate signal
- When S = 0, TG1 = OFF and TG2 is ON and Z = B.
- When S = 1, TG1 = ON and TG2 is OFF and Z = A.
- Symbolid representation of MUX stick diagram and layout is below



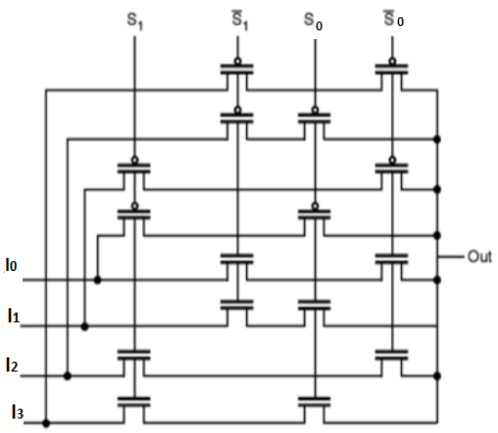
4:1 Multiplexer implementation

- There are 4 inputs - I_0, I_1, I_2, I_3, 2 select lines S_0, S_1 and 1 output line – Y/out.
- The truth table is shown

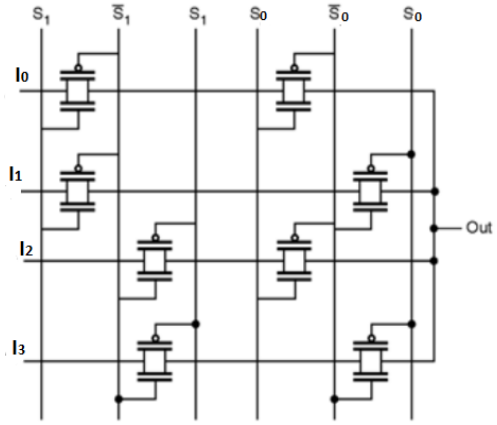
S1	S0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



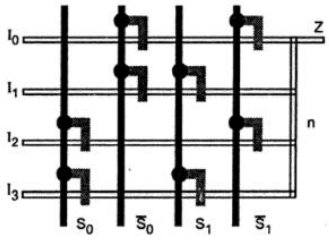
- The output equation for $Y = I_0 S'_1 S'_0 + I_1 S'_1 S_0 + I_2 S_1 S'_0 + I_3 S_1 S_0$
- 4:1 mux implemenation using CMOS technology and transmission Gate also their stick diagrams.



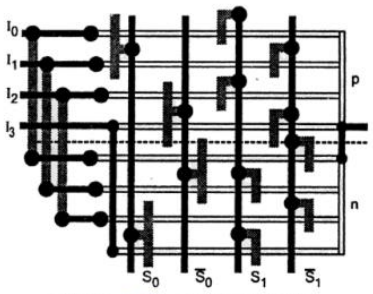
4 : 1 MUX using CMOS logic



4 : 1 MUX using transmission gate



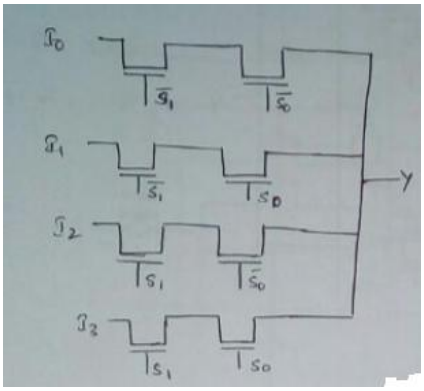
(a) nMOS switches



Note : V_{DD} and V_{SS} contacts are not shown.

(b) Transmission gates (CMOS)

- 4:1 mux can also be implemented using two 2:1 multiplexer



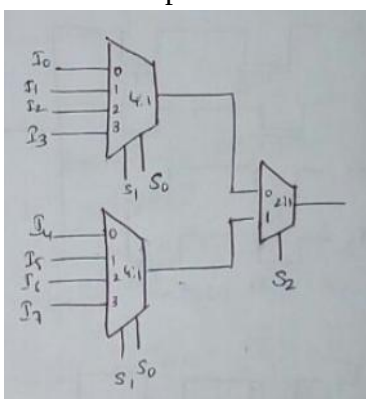


Fig. 4:1 MUX using pass transistor logic and implementation of 8:1 using two 4:1 MUX

- 8:1 mux can be implemented using two 4:1 multiplexer

S2	S1	S0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4

1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

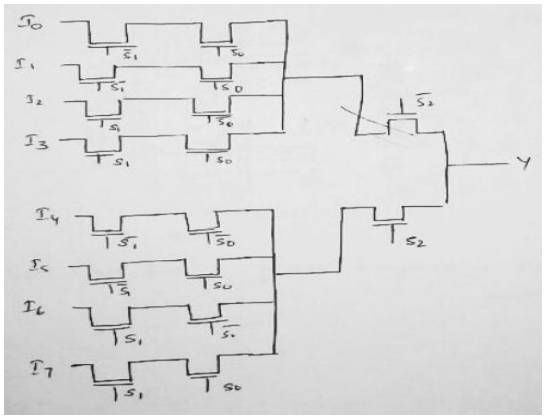
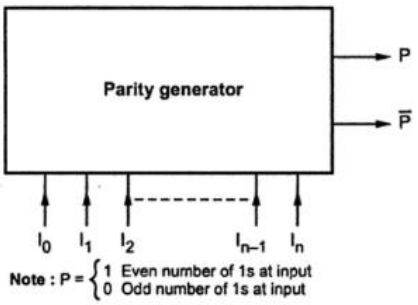


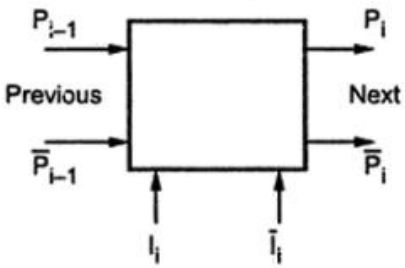
Fig. Truth table for 8:1 MUX and implementation of same using pass transisitor logic

Parity Generator

- The parity generating technique is one of the most widely used error detection techniques for the data transmission.
- In digital systems, when binary data is transmitted and processed, data may be subjected to noise.
- Hence, **parity bit** is added to the word containing data in order to make number of 1s either even or odd. During the transmission of binary data, the message containing the data bits along with parity bit is transmitted from transmitter node to receiver node.
- At the receiving end, the number of 1s in the message is counted and if it doesn't match with the transmitted one, then it means there is an error in the data.
- A parity generator is a combinational logic circuit that generates the parity bit in the transmitter. On the other hand, a circuit that checks the parity in the receiver is called parity checker. A combined circuit or devices of parity generators and parity checkers are commonly used in digital systems to detect the single bit errors in the transmitted data word.
- In even parity, the added parity bit will make the total number of 1s an even amount whereas in odd parity the added parity bit will make the total number of 1s odd amount.
- The basic principle involved in the implementation of parity circuits is that sum of odd number of 1s is always 1 and sum of even number of 1s is always zero. Such error detecting and correction can be implemented by using Ex-OR gates (since Ex-OR gate produce zero output when there are even number of inputs).
- Parity Generator: It is combinational circuit that accepts an n-1 bit stream data and generates the additional bit that is to be transmitted with the bit stream. This additional or extra bit is termed as a parity bit.
- In **even parity** bit scheme, the parity bit is '**0**' if there are **even number of 1s** in the data stream and the parity bit is '**1**' if there are **odd number of 1s** in the data stream.
- In **odd parity** bit scheme, the parity bit is '**1**' if there are **even number of 1**s in the data stream and the parity bit is '**0**' if there are **odd number of 1s** in the data stream. Let us discuss both even and odd parity generators.
- In VLSI a circuit to be designed to indicate parity of a binary number is shown in the Fig. for an (n+1) bit input.

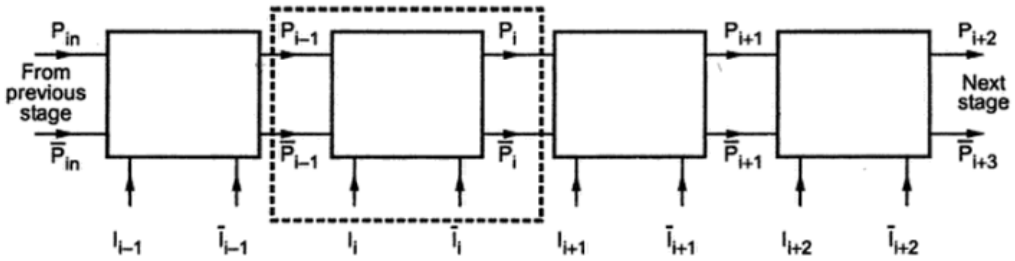


Parity generator basic block diagram



Parity generator-basic one-bit cell

- Since the no. of bits is undefined, a general solution on cascadable bit-wise and a regular structure is shown in Fig.
- A standard or basic one-bit cell from which an n-bit parity generator may be formed. The standard/ basic cell is shown in the Fig.
- The parity information is passed from one cell to next and the parity information is modified or retained depending on the input lines A_i and A'_i



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

Fig. Parity generator - structured design approach

- If $A_i = 1$, parity output P_i will change to P'_{i-1} (i.e., if A_i [input] =1 and P_{i-1} [previous parity] =1, the output parity P_i will change to 0)
- If $A_i = 0$, output parity P_i will remain in the same state of P_{i-1} (i.e., if A_i [input] =0 and P_{i-1} [previous parity] =1, the output parity P_i will remain as 1)
- Suitable arrangement for such cell is implemented using the expression

$$P_i = P'_{i-1} A_i + P_{i-1} A'_i$$

- The same can be implemented using nMOS and CMOS technology. The parity generator symbol and stick diagram for nMOS technology is shown below.

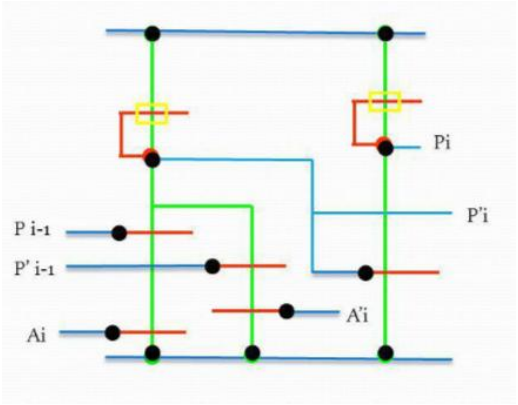
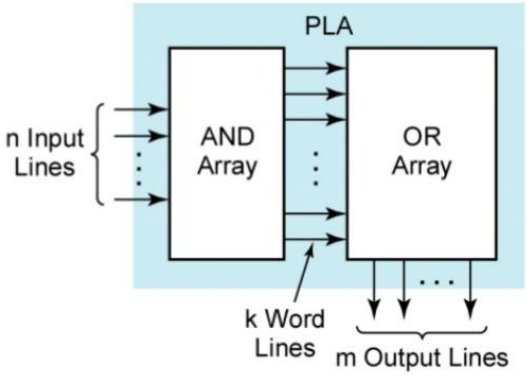


Fig. Symbol for P_i **Fig. nMOS stick diagram for parity generator**

The Programmable Logic Array (PLA)

- PLA describes class of standalone devices that allows users to program the functionalities. The capability of these programmable devices are limited and have been replaced by significantly powerful field programmable gate array (FPGA).
- Fig shows the block diagram of PLA. It consists of two level of combinational logic functions.
- Both the levels of combinational logic used together helps in implementing sum-of-products (SOP) logic functions.
- It performs the same basic function as a ROM. It has n inputs and m outputs and can realize m functions of n variables
- '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.



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

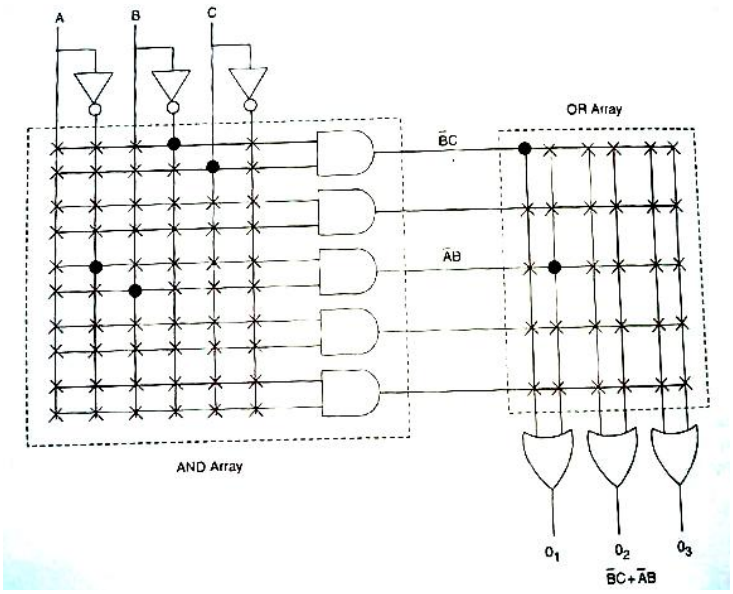
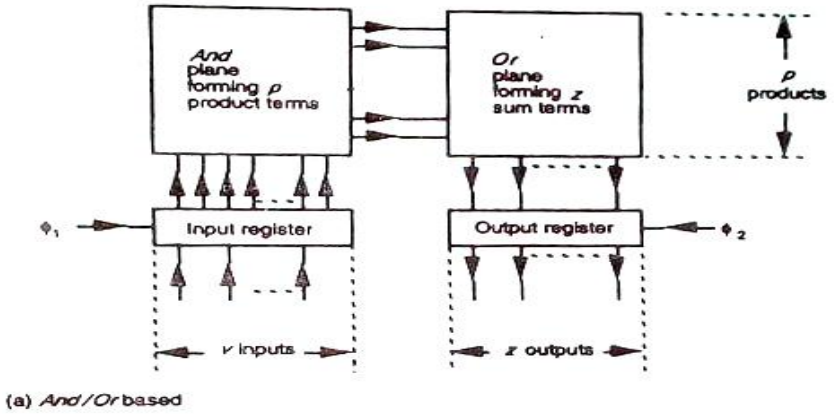


Fig. Architecture of PLA

- 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 need 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.
- Both the implementation is shown below.



(a) *And/Or* based

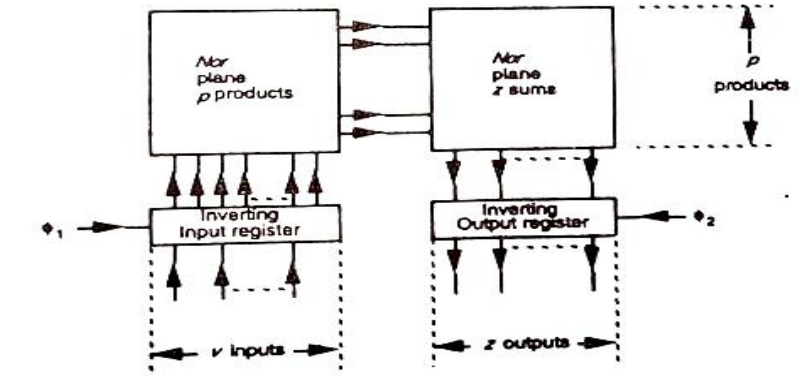


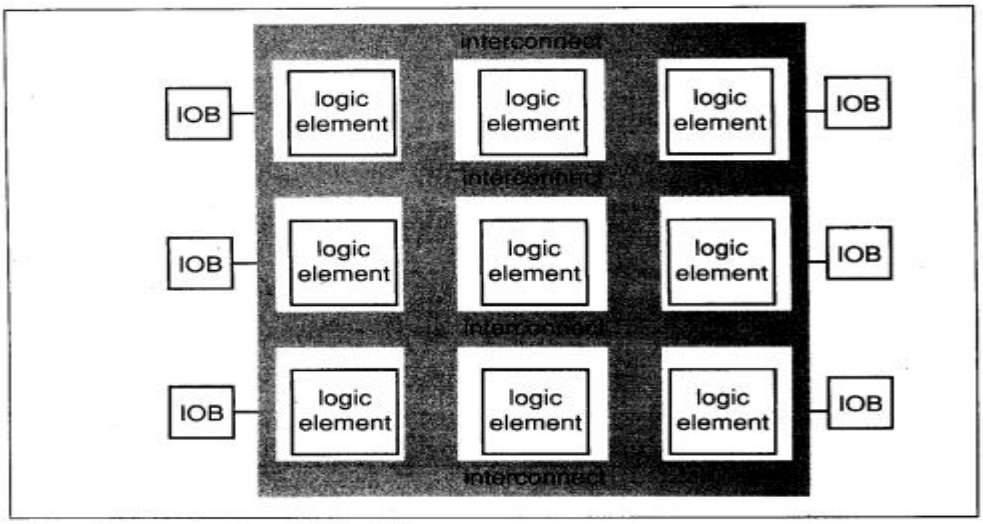
FIGURE C.2 PLA floor plans.

FPGA Based Systems

3.2 FPGA Architecture:

FPGA consists of three major elements.

- ✓ Combinational Logic
- ✓ Interconnect
- ✓ I/O pins

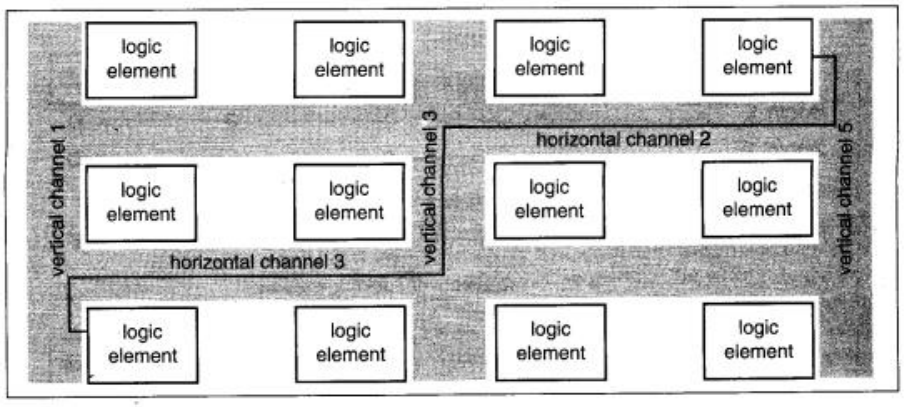


Generic structure of an FPGA fabric.

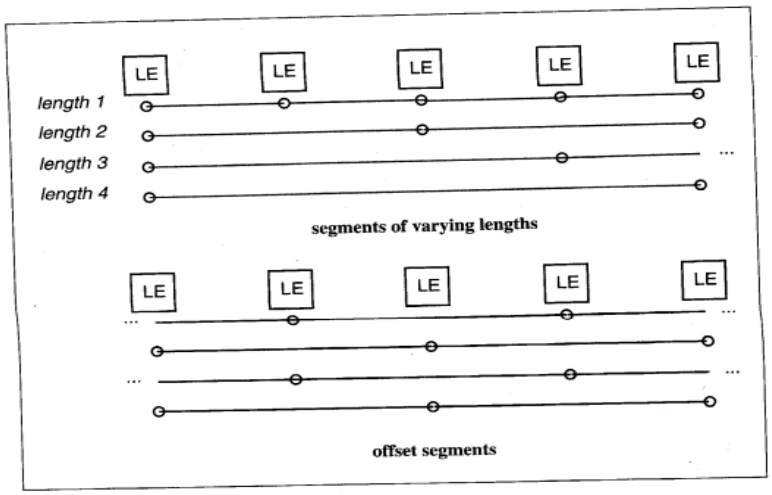
- The combinational logic is divided into small units which is known as logic elements(LE) or combinational logic blocks(CLB).
- LE or CLB usually forms the functions of several logic gates.
- Interconnections between these logic elements are made using programmable interconnects.
- This interconnects 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. These are generally programmable for inputs or outputs and often provides other features such as low power or high speed connection.

FPGA Interconnect:

- 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 aspect because wiring is global property of logic design.
- Connection between logic elements requires complex paths since LE's are arranged in two dimensional structure as shown in below figure.



- Wires are typically organized in wiring channels or routing channels which runs horizontally and vertically throughout the chip.
- Each channel contains several wires 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.
- 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 below shows connections of varying length as measured in unit LE's: 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.



- All FPGA's need to be programmed or configured.
- There are three major circuit technologies for configuring an FPGA: SRAM, Antifuse and flash.

4.1 Physical Design of FPGA.

Physical design is divided into two major phases.

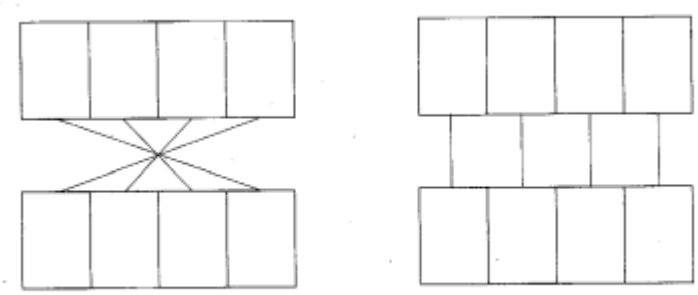
- Placement: it determines the position of logic elements and I/O pads.
- Routing: selects the paths for connection between logic elements and I/O pads.

- 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 is the obvious metric 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 compute, so tools generally use surrogates to estimate delay.

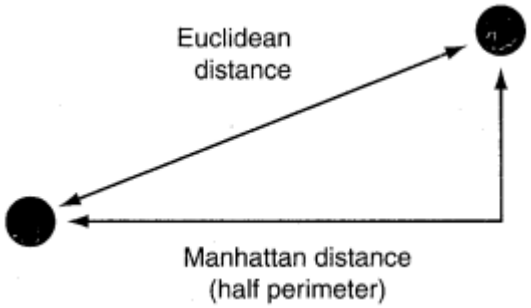
Placement

The separation of placement and routing raises an important problem: how to judge the quality of placement? We cannot afford to execute a complete routing for every placement to judge its quality: 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.



bad placement good placement

- 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. In above figure shows two placements, one with longer total rat's nest connection and one with shorter total interconnections.



- There are several ways to measure distance as shown in figure: Euclidean distance is the direct line between the two. We can also measure Manhattan distance, also known as half perimeter.
- Manhattan distance is more reflective of final wire length, it is also easier to compute because unlike Euclidean distance it does not require computing a square root.
- There are many algorithms for partitioning but these algorithms can generally divided into different approaches: Bottom up and Top Down.
- Bottom up methods are generally referred as clustering methods. These cluster together nodes to create partitions.
- Top Down methods divide nodes into groups that are than further divided. These methods are known as portioning methods.
- Dividing all the components into two partitions doesn't give very fine direction for placement, the partitioning is usually repeated by creating subpartitions of the original partition, a process known as hierarchical partitioning.
- Clustering algorithms build groups of nodes from the bottom up in contrast to the Top – Down approach taken by partitioning. A small number of nodes create initial clusters. Nodes then added clusters based upon their connections with other nodes.

Routing

- Routing selects paths for connections that must be made between logic elements and I/O pads.
- In 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.
- Routing has two major cost metrics: wire length and delay. Wire length approximates utilization of routing resources and delay 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.
- 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.
- 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.

1.1 Goals and Techniques

Logical function to be performed is only one goal that must be met by FPGA or any digital design. Many other attributes must be satisfied for project to be successful:

- ✓ **Performance:** Logic must run at required rate. It is measured in many ways, such as throughput and latency. Clock rate often measures performance.
- ✓ **Power/energy:** chip must run within an energy or power budget. Energy consumption is clearly critical in battery powered systems
- ✓ **Design time:** FPGAs have standard parts, have several advantages in design time. They can be used as prototypes, can be programmed quickly and can be used as part in final design.

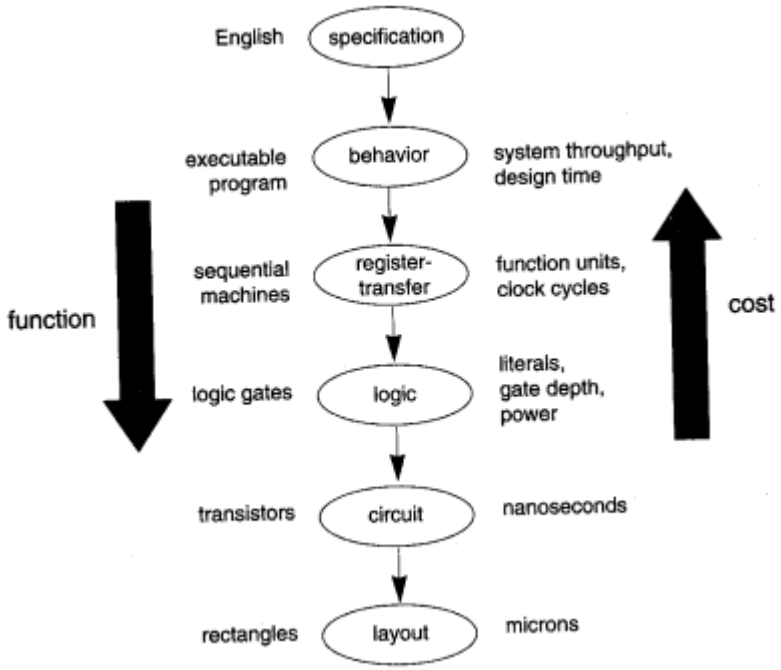
- ✓ **Design cost:** design time is one important component in design cost, but other factors such as required support tools may be considered. FPGA tools are less expensive than custom VLSI tools.
- ✓ **Manufacturing cost:** It is the cost of replicating the system many times. FPGAs are generally more expensive than ASICs due to overhead of programming. But the fact that they are standard parts helps to reduce their cost.

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.