**Sri Chandrasekharendra Saraswathi Viswa MahaVidyalaya**

**(Deemed to be University)**

**Department of Electronics and Communication Engineering**

**Lecture Notes**

**on**

# VLSI DESIGN

**Dr. G. Senthil Kumar**

Associate Professor, ECE Department

SCSVMV Deemed University

**Email: gsk_ece@kanchiuniv.ac.in**

# VLSI DESIGN

PROGRAMME : B.E. (Electronics and Communication Engineering)

**COURSE OBJECTIVES**

- To study the fundamentals of MOS transistor and CMOS circuits and their characteristics.
- To understand the concepts of CMOS processing technology and design layouts.
- To discuss the performance trade-offs involved in designing and realizing the circuits in CMOS technology.
- To discuss architectural choices and design and realization of sub-systems.
- To learn the different FPGA architectures and testability of VLSI circuits.

**Prerequisite :**

- Basic knowledge of Electronic Circuits and Digital System Design

**COURSE OUTCOMES**

At the end of the course, the students will be able to:

| CO No. | Course Outcomes | Bloom's Taxonomy (Knowledge/Comprehension/ Application/ Analysis/ Synthesis/Evaluation) |
|---|---|---|
| CO1 | Understand MOS transistor theory and design Inverters | Knowledge / Comprehension / Application / Analysis |
| CO2 | Realize the concepts of CMOS circuits using processing technology and layouts | Knowledge / Comprehension / Application / Analysis/ Synthesis / Evaluation |
| CO3 | Design and realize the circuits in CMOS logic circuits | Knowledge / Comprehension/ Application / Analysis / Synthesis / Evaluation |
| CO4 | Design arithmetic building blocks and memory subsystems. | Knowledge / Comprehension / Application / Analysis |
| CO5 | Apply and implement FPGA design flow and testing. | Knowledge / Comprehension / Application / Analysis |

**Mapping with Program Outcome**

| | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CO1 | S | S | S | M | M | L | | | | | S | S |
| CO2 | S | S | S | S | S | M | | | | | S | S |
| CO3 | S | S | S | S | S | M | | | | | S | S |
| CO4 | S | S | S | S | S | M | | | | | S | S |
| CO5 | S | S | S | S | M | M | | | | | S | S |

S – Strong, M – Medium, L - Low

**Assessment Pattern**

| Bloom's Category | Continuous Assessment Tests | | Terminal Examination (100 marks) |
|---|---|---|---|
| | Ist Internal (30 marks) | IInd Internal (30 marks) | |
| Knowledge | 10 | 8 | 30 |
| Comprehension | 8 | 8 | 30 |
| Application | 4 | 6 | 15 |
| Analysis | 4 | 4 | 15 |
| Synthesis | 4 | 2 | 6 |
| Evaluation | - | 2 | 4 |

# SYLLABUS: VLSI Design

**UNIT – I : INTRODUCTION TO VLSI AND MOS TRANSISTOR THEORY** [10 Hours]

*Evolution of IC Technologies*: SSI, MSI, LSI, VLSI, ULSI, and GLSI, Moore's Law. *MOS Theory*: MOS as switch - NMOS and PMOS, CMOS logic and its features. *NMOS Enhancement Transistor* - Working and Characteristics, Threshold voltage and Body effect of MOS, MOS device design equations (First order effects). *MOS Inverters*: CMOS inverter transfer characteristics, Noise margin. NMOS and pseudo-NMOS inverter, Bi-CMOS Inverter, CMOS transmission gate.

**UNIT – II : CMOS PROCESSING TECHNOLOGY AND LAYOUTS** [10 Hours]

Silicon Semiconductor fabrication technology, Fabrication forms and CMOS (Basic n-WELL process). Layouts and Design rules: Layout based rules, Simple CMOS Stick Layout diagrams - Inverter, NAND, NOR gates and Multiplexer. Scaling: Constant Field, and Constant voltage.

**UNIT – III : MOS CIRCUIT PERFORMANCE AND CMOS LOGIC CIRCUITS** [10 Hours]

Sheet Resistance definition, MOS device capacitances – model, Distributed RC effects. Switching characteristics – Rise time, fall time and delay time. Stage ratio. Simple examples of combinational and sequential circuits using CMOS: NANDI NOR gates, and Compound gates, Latches, and Registers.

**UNIT – IV : SUB SYSTEM DESIGN AND TESTING** [10 Hours]

General system design-Design of ALU subsystems, Adder and Multipliers. Memories - Static RAM, Control logic implementation using PLA's. Testing of VLSI circuits - Need for testing, Fault models, and ATPG. Design for Testability (DFT) - Scan based and Self-test approaches.

**UNIT – V : PROGRAMMABLE LOGICS** [10 Hours]

Basic ROM structures, PLAs, PALs, PLDs, Implementation of traffic light controller using PLD. FPGAs and CPLDs: XILINX and ALTERA series.

---

**ESSENTIAL READING BOOKS :**

1. Neil Weste and Kamran Eshraghian, "Principles of CMOS VLSI Design ", Addison Wesley, 1998.
2. Charles H Roth, Jr., "Digital Systems Design using VHDL", Thomson Learning, 2001

**RECOMMONDED READING BOOKS :**

1. John P. Uyemura, "VLSI Design Principles", John Wiley, 2002
2. E. Fabricious, "Introduction to VLSI design", McGraw-Hill, 1990
3. Wayne Wolf, "Modern VLSI Design", Pearson Education, 2003

**WEB LINKS FOR REFERENCE**

1. https://www.classcentral.com/course/swayam-cmos-digital-vlsi-design-12964
2. https://nptel.ac.in/courses/108/107/108107129/
3.http://access.ee.ntu.edu.tw/course/VLSI_design_90second/pdf/slide/chapter%202%2003-30-2001.pdf

# Notes on
# MOS Transistor Theory
# in VLSI Design – Unit I

**Dr. G. Senthil Kumar,**
Associate Professor,
Dept. of ECE, SCSVMV,
email: gsk_ece@kanchiuniv.ac.in

==============================================================

**OBJECTIVES**:

In this lesson, you will be introduced to the fundamentals of MOS transistor and CMOS circuits and their characteristics.

**CONTENTS**:

## 1. EVOLUTION OF IC TECHNOLOGIES:

### *MOS (Metal Oxide Silicon) Transistor History*

1925: J. Lilienfeld proposed the basic principle of MOS FET (Field Effect Transistor).

1935: O. Heil proposed a similar structure.

1962: P.K. Weimer (RCA) first placed pMOS and nMOS transistors on the same substrate.

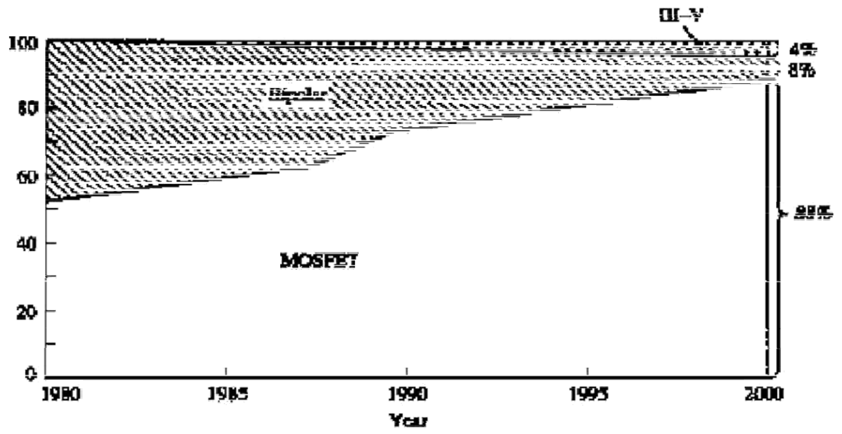
1963: Frank Wanlass (Fairchild) invented invertor, NOR and NAND CMOS gates. This invention starts the era of CMOS low power applications.

1965: The first MOS calculator.

1971: Emergence of nMOS-silicon gate technology.

Note: Early research on MOS technology led to success of bipolar transistor. This in turns leads to a decline of interest in MOS transistor.

1980: The market share of MOSFET exceeds bipolar device.



Transistor was first invented by William. B. Shockley, Walter Brattain and John Bardeen of Bell laboratories. In 1961, first IC was introduced.

***Levels of Integration***:

- Small Scale Integration:- (10-100) transistors => Example: Logic gates
- Medium Scale Integration:- (100-1000) => Example: counters
- Large Scale Integration:- (1000-20000) => Example:8-bit chip
- Very Large Scale Integration:- (20000-1000000) => Example:16 & 32 bit up
- Ultra Large Scale Integration:- (1000000-10000000) => Example: Special processors, virtual reality machines, smart sensors

***Moore's Law***

The number of transistors embedded on the chip doubles after every one and a half years.‖ The number of transistors is taken on the y-axis and the years in taken on the x-axis. The diagram also shows the speed in MHz. the graph given in figure also shows the variation of speed of the chip in MHz.

## 2. MOS TRANSISTOR THEORY

A Metal-Oxide-Semiconductor (MOS) structure is created by superimposing several layers of conducting and insulating materials to form a sandwich-like structure. These structures are manufactured using a series of chemical processing steps involving oxidation of the silicon, selective introduction of dopants, and deposition and etching of metal wires and contacts. Transistors are built on nearly flawless single crystals of silicon, which are available as thin flat circular wafers of 15–30 cm in diameter. CMOS technology provides two types of transistors (also called devices): an n-type transistor (nMOS) and a p-type transistor (pMOS). Transistor operation is controlled by electric fields so the devices are also called Metal Oxide Semiconductor Field Effect Transistors (MOSFETs) or simply FETs. Cross-sections and symbols of these transistors are shown. The n+ and p+ regions indicate heavily doped n- or p-type silicon.

Basic starting material: Single crystal of silicon formed as wafers (4-inch, 6-inch, 8-inch, 12-inch). MOS structure is created by superposing several layers of conducting, insulating, and transistor-forming materials to create a sandwich-like structure by way of a series of chemical processing steps such as: oxidation of the silicon, diffusion of impurities into silicon to give it certain conduction characteristics, and deposition and etching of aluminum on silicon to form interconnection.

***Two types of transistors***

nMOS: with negatively diffused (doped) source and drain on lightly p-doped substrate. pMOS: with positively diffused source and drain on lightly n-doped substrate.

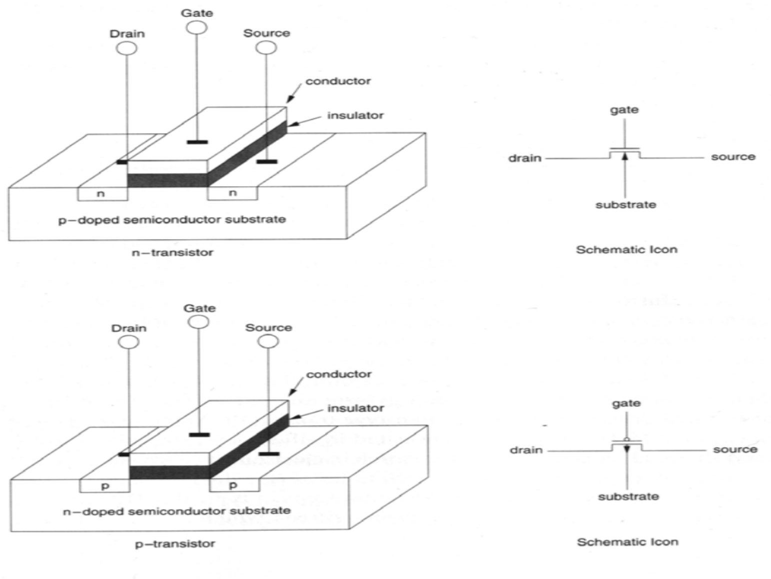


**FIGURE 1.1** Physical structure of MOS transistors and their schematic icons

Four terminals of a transistor: Gate: usually formed by polycrystalline silicon (polysilicon for short). It is a control input that affects the flow of electrical current between source and drain. Source and Drain: Formed by diffusion. They are physically equivalent and the name assignment depends on the direction of current flow. Source provides charges. Drain sinks charges. Substrate: the fourth terminal of MOS transistor and will be discussed later. Note that p-type transistor (pMOS) has n-doped substrate and n-type transistor (nMOS) has p-doped substrate.

Each transistor consists of a stack of the conducting gate, an insulating layer of silicon dioxide (SiO2, better known as glass), and the silicon wafer, also called the substrate, body, or bulk. Gates of early transistors were built from metal, so the stack was called metaloxide-semiconductor, or MOS. Since the 1970s, the gate has been formed from polycrystalline silicon (polysilicon), but the name stuck. (Interestingly, metal gates reemerged in 2007 to solve materials problems in advanced manufacturing processes.) An nMOS transistor is built with a p-type body and has regions of n-type semiconductor adjacent to the gate called the source and drain. They are physically equivalent and for now we will regard them as interchangeable. The body is typically grounded. A pMOS transistor is just the opposite, consisting of p-type source and drain regions with an n-type body. In a CMOS technology with both flavors of transistors, the substrate is either n-type or p-type. The other flavor of transistor must be built in a special well in which dopant atoms have been added to form the body of the opposite type.

## ***MOS TRANSISTOR AS SWITCHES - NMOS AND PMOS SWITCHES***

The gate controls the flow of current between the source and the drain. The gate is a control input: It affects the flow of electrical current between the source and drain. Consider an nMOS transistor. The body is generally grounded so the p–n junctions of the source and drain to body are reverse-biased. If the gate is also grounded, no current flows through the reverse-biased junctions. Hence, we say the transistor is OFF. If the gate voltage is raised, it creates an electric field that starts to attract free electrons to the underside of the Si–SiO2 interface. If the voltage is raised enough, the electrons outnumber the holes and a thin region under the gate called the channel is inverted to act as an n-type semiconductor. Hence, a conducting path of electron carriers is formed from source to drain and current can flow. We say the transistor is ON.

This allows us to treat the MOS transistors as simple on/off switches. For a pMOS transistor, the situation is again reversed. The body is held at a positive voltage. When the gate is also at a positive voltage, the source and drain junctions are reverse-biased and no current flows, so the transistor is OFF. When the gate voltage is lowered, positive charges are attracted to the underside of the Si–SiO2 interface. A sufficiently low gate voltage inverts the channel and a conducting path of positive carriers is formed from source to drain, so the transistor is ON. Notice that the symbol for the pMOS transistor has a bubble on the gate, indicating that the transistor behavior is the opposite of the nMOS.

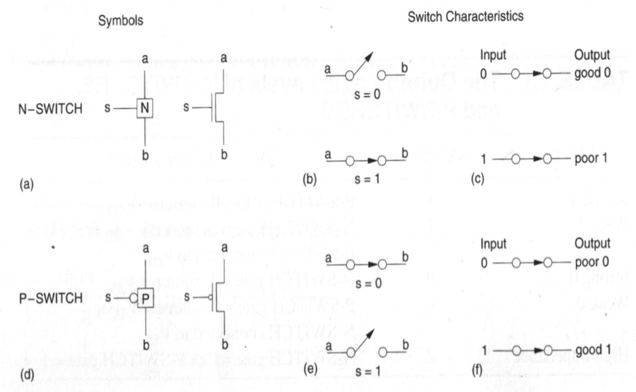


**FIGURE 1.2** nMOS and pMOS switch symbols and characteristics

The positive voltage is usually called VDD or POWER and represents a logic 1 value in digital circuits. In popular logic families of the 1970s and 1980s, VDD was set to 5 volts. Smaller, more recent transistors are unable to withstand such high voltages and have used supplies of 3.3 V, 2.5 V, 1.8 V, 1.5 V, 1.2 V, 1.0 V, and so forth. The low voltage is called GROUND (GND) or VSS and represents a logic 0. It is normally 0 volts.

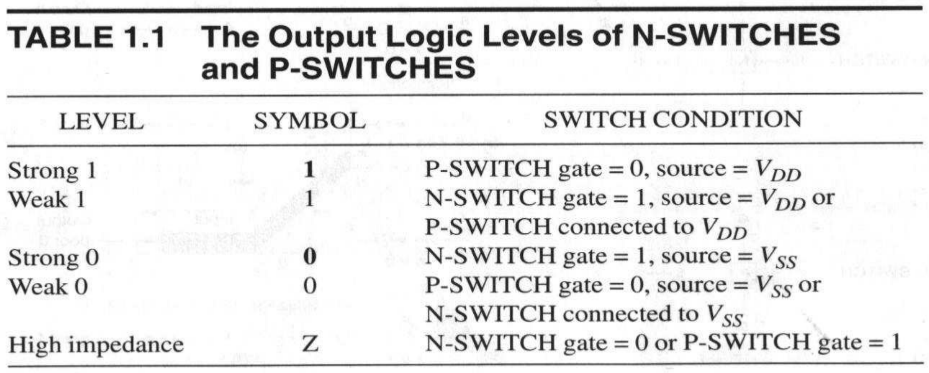


**TABLE 1.1 The Output Logic Levels of N-SWITCHES and P-SWITCHES**

| LEVEL | SYMBOL | SWITCH CONDITION |
|---|---|---|
| Strong 1 | **1** | P-SWITCH gate = 0, source = $V_{DD}$ |
| Weak 1 | 1 | N-SWITCH gate = 1, source = $V_{DD}$ or P-SWITCH connected to $V_{DD}$ |
| Strong 0 | **0** | N-SWITCH gate = 1, source = $V_{SS}$ |
| Weak 0 | 0 | P-SWITCH gate = 0, source = $V_{SS}$ or N-SWITCH connected to $V_{SS}$ |
| High impedance | Z | N-SWITCH gate = 0 or P-SWITCH gate = 1 |

In summary, the gate of an MOS transistor controls the flow of current between the source and drain. Simplifying this to the extreme allows the MOS transistors to be viewed as simple ON/OFF switches. When the gate of an nMOS transistor is 1, the transistor is ON and there is a conducting path from source to drain. When the gate is low, the nMOS transistor is OFF and almost zero current flows from source to drain. A pMOS transistor is just the opposite, being ON when the gate is low and OFF when the gate is high. This switch model is illustrated in Figure 1.10, where g, s, and d indicate gate, source, and drain. This model will be our most common one when discussing circuit behavior.

Logic value system:

1: Between 1.5 and 15 volts

z: High Impedance (a circuit node not connecting to either Power or Ground)

0: Zero volts

Strength of the “1” and “0” signals:

Strength of a signal is measured by its ability to sink or source current.

Power (PWR, VDD): Strongest 1.

Ground (GND, VSS): Strongest 0.

By convention, current is sourced from Power, and Ground sinks current. nMOS switch (N-SWITCH) is closed or ON if the drain and the source are connected . This occurs when there is a “1” on the gate .

Pass a good 0.

Pass a poor 1.

pMOS switch (P-SWITCH) is closed or ON when there is a “0” on the gate. Pass a good 1 and Pass a poor 0.

## *CMOS LOGIC AND ITS FEATURES*

### *Inverter*

Figure shows the schematic and symbol for a CMOS inverter or NOT gate using one nMOS transistor and one pMOS transistor. The bar at the top indicates VDD and the triangle at the bottom indicates GND. When the input A is 0, the nMOS transistor is OFF and the pMOS transistor is ON. Thus, the output Y is pulled up to 1 because it is connected to VDD but not to GND. Conversely, when A is 1, the nMOS is ON, the pMOS is OFF, and Y is pulled down to ‘0.’ This is summarized in Table.

**TABLE 1.1** Inverter truth table

| A | Y |
|---|---|
| 0 | 1 |
| 1 | 0 |

“ Input 0 makes Output 1” suggests a P-SWITCH connected from a “1” source (VDD) to the output. “ Input 1 makes Output 0” suggests an N-SWITCH connected from a “0” source (VSS) to the output.

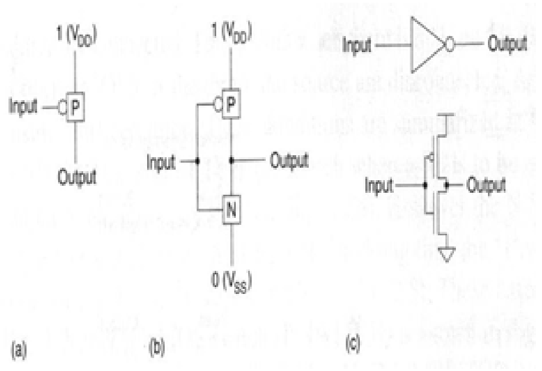
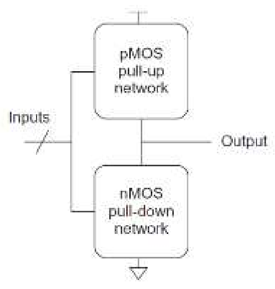


Fig. Inverter Circuit

***CMOS Logic Gates***

The inverter and NAND gates are examples of static CMOS logic gates, also called complementary CMOS gates. In general, a static CMOS gate has an nMOS pull-down network to connect the output to 0 (GND) and pMOS pull-up network to connect the output to 1 (VDD), as shown in Figure. The networks are arranged such that one is ON and the other OFF for any input pattern.



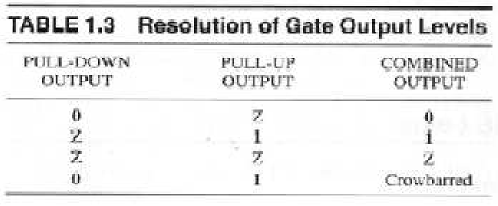
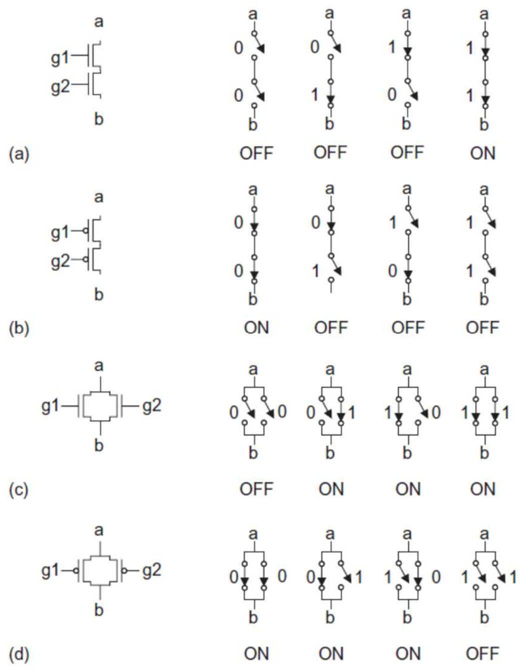


TABLE 1.3 Resolution of Gate Output Levels

| PULL-DOWN OUTPUT | PULL-UP OUTPUT | COMBINED OUTPUT |
|---|---|---|
| 0 | Z | 0 |
| Z | 1 | 1 |
| Z | Z | Z |
| 0 | 1 | Crowbarred |

***Combinational Logic***

The pull-up and pull-down networks in the inverter each consist of a single transistor. The NAND gate uses a series pull-down network and a parallel pullup network. More elaborate networks are used for more complex gates. Two or more transistors in series are ON only if all of the series transistors are ON. Two or more transistors in parallel are ON if any of the parallel transistors are ON. This is illustrated in Figure 1.15 for nMOS and pMOS transistor pairs. By using combinations of these constructions, CMOS combinational gates can be constructed. Although such static CMOS gates are most widely used, explores alternate ways of building gates with transistors.

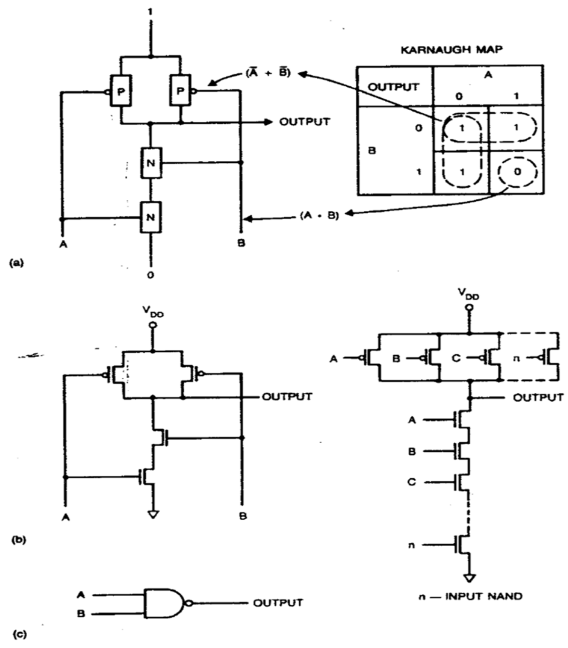


In general, when we join a pull-up network to a pull-down network to form a logic gate as shown in Figure, they both will attempt to exert a logic level at the output. The possible levels at the output are shown in Table. From this table it can be seen that the output of a CMOS logic gate can be in four states. The 1 and 0 levels have been encountered with the inverter and NAND gates, where either the pull-up or pull-down is OFF and the other structure is ON. When both pull-up and pull-down are OFF, the high impedance or floating Z output state results. This is of importance in multiplexers, memory elements, and tristate bus drivers. The crowbarred (or contention) X level exists when both pull-up and pull-down are simultaneously turned ON. Contention between the two networks results in an indeterminate output level and dissipates static power. It is usually an unwanted condition.

***NAND Gate***

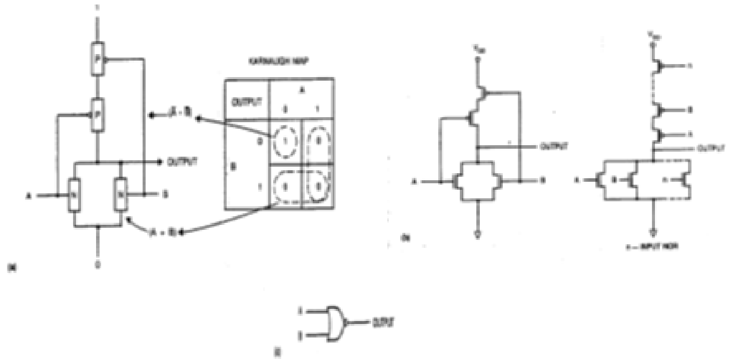
Figure shows a 2-input CMOS NAND gate. It consists of two series nMOS transistors between Y and GND and two parallel pMOS transistors between Y and VDD. If either input A or B is 0, at least one of the nMOS transistors will be OFF, breaking the path from Y to

GND. But at least one of the pMOS transistors will be ON, creating a path from Y to VDD. Hence, the output Y will be 1. If both inputs are 1, both of the nMOS transistors will be ON and both of the pMOS transistors will be OFF. Hence, the output will be 0. The truth table is given in Table and the symbol is shown in Figure (b). Note that by DeMorgan's Law, the inversion bubble may be placed on either side of the gate. In the figures in this book, twolines intersecting at a T-junction are connected.



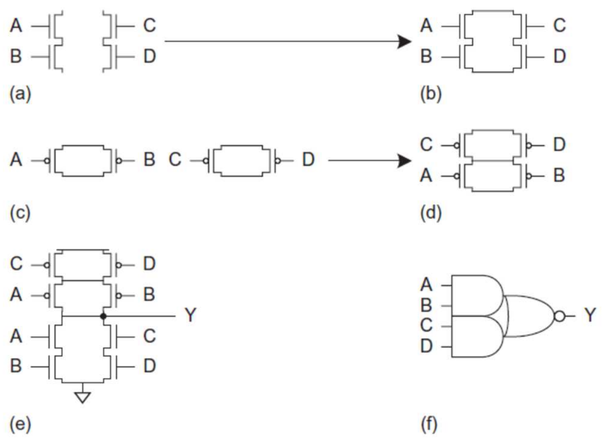
***NOR Gate***

A 2-input NOR gate is shown in Figure. The nMOS transistors are in parallel to pull the output low when either input is high. The pMOS are in series to pull the output high when both inputs are low, as indicated in Table. The output is never crowbarred or left floating.

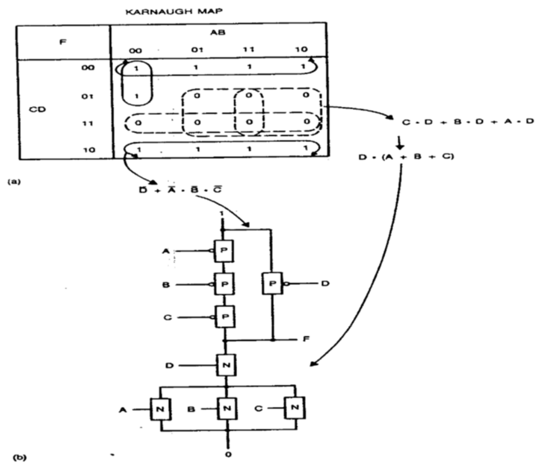


***Compound Gates***

A compound gate performing a more complex logic function in a single stage of logic is formed by using a combination of series and parallel switch structures. For example, the derivation of the circuit for the function ***Y = (A · B) + (C · D)*** is shown in Figure. This function is sometimes called AND-OR-INVERT-22, or AOI22 because it performs the NOR of a pair of 2-input ANDs. For the nMOS pull-down network, take the un-inverted expression ((A · B) + (C · D)) indicating when the output should be pulled to '0.' The AND expressions (A · B) and (C · D) may be implemented by series connections of switches, as shown in Figure 1.18(a). Now ORing the result requires the parallel connection of these two structures, which is shown in Figure 1.18(b).



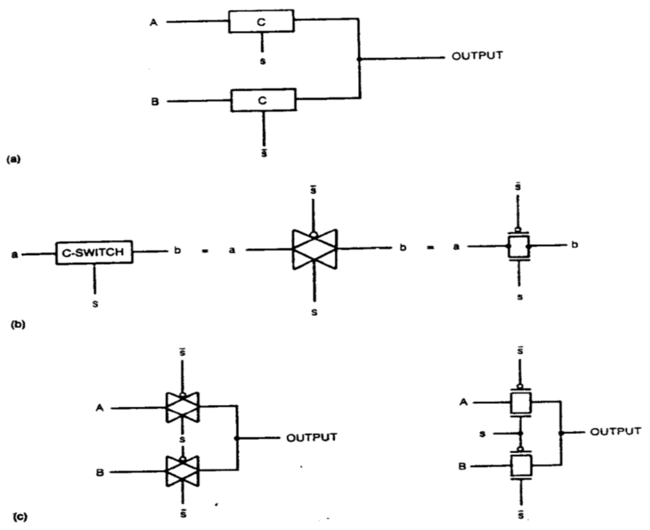
For the pMOS pull-up network, we must compute the complementary expression using switches that turn on with inverted polarity. By DeMorgan's Law, this is equivalent to interchanging AND and OR operations.



Hence, transistors that appear in series in the pull-down network must appear in parallel in the pull-up network. Transistors that appear in parallel in the pulldown network must appear in series in the pull-up network. This principle is called conduction complements and has already been used in the design of the NAND and NOR gates. In the pull-up network, the parallel combination of A and B is placed in series with the parallel combination of C and D. This progression is evident in Figure (c) and Figure (d). Putting the networks together yields the full schematic (Figure (e)).

***Multiplexers***

Multiplexers are key components in CMOS memory elements and data manipulation structures. A multiplexer chooses the output from among several inputs based on a select signal. A 2-input, or 2:1 multiplexer, chooses input D0 when the select is 0 and input D1 when the select is 1. The truth table is given in Table 1.6; the logic function is $Y = S \cdot D0 + S \cdot D1$.



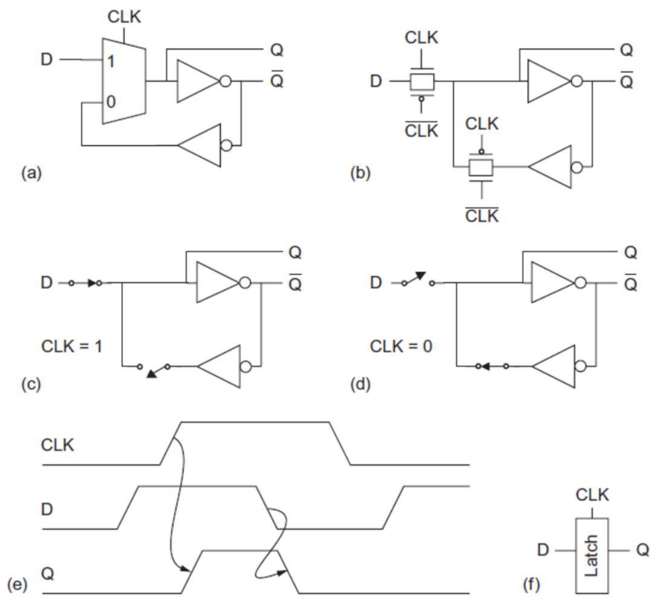
Two transmission gates can be tied together to form a compact 2-input multiplexer, as shown in Figure (a). The select and its complement enable exactly one of the two transmission gates at any given time. The complementary select S is often not drawn in the symbol, as shown in Figure (b).

Again, the transmission gates produce a nonrestoring multiplexer. We could build a restoring, inverting multiplexer out of gates in several ways. One is the compound gate of Figure 1.18(e), connected as shown in Figure (a). Another is to gang together two tristate inverters, as shown in Figure (b). Notice that the schematics of these two approaches are nearly identical, save that the pull-up network has been slightly simplified and permuted in Figure (b).

This is possible because the select and its complement are mutually exclusive. The tristate approach is slightly more compact and faster because it requires less internal wire. Again, if the complementary select is generated within the cell, it is omitted from the symbol

### *Memory-Latches and Registers*

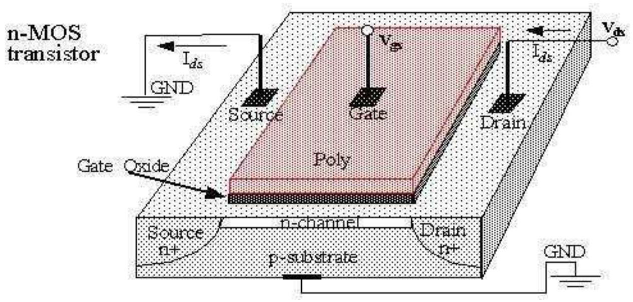
Sequential circuits have memory: their outputs depend on both current and previous inputs. Using the combinational circuits developed so far, we can now build sequential circuits such as latches and flip-flops. These elements receive a clock, CLK, and a data input, D, and produce an output, Q. A D latch is transparent when CLK = 1, meaning that Q follows D. It becomes opaque when CLK = 0, meaning Q retains its previous value and ignores changes in D.



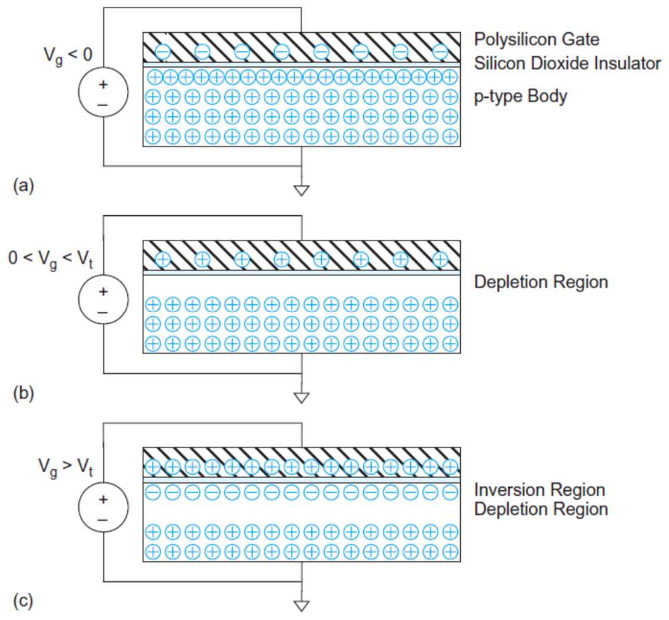
D latch built from a 2-input multiplexer and two inverters is shown in Figure (a). The multiplexer can be built from a pair of transmission gates, shown in Figure (b), because the inverters are restoring. This latch also produces a complementary output, Q. When CLK = 1, the latch is transparent and D flows through to Q (Figure (c)). When CLK falls to 0, the latch becomes opaque. A feedback path around the inverter pair is established (Figure (d)) to hold the current state of Q indefinitely. The D latch is also known as a level-sensitive latch because the state of the output is dependent on the level of the clock signal, as shown in Figure (e). The latch shown is a positive-level-sensitive latch, represented by the symbol in Figure (f ). By inverting the control connections to the multiplexer, the latch becomes negative-level-sensitive.

## 3. NMOS ENHANCEMENT TRANSISTOR

The MOS transistor is a majority-carrier device in which the current in a conducting channel between the source and drain is controlled by a voltage applied to the gate. In an nMOS transistor, the majority carriers are electrons; in a pMOS transistor, the majority carriers are holes. The behavior of MOS transistors can be understood by first examining an isolated MOS structure with a gate and body but no source or drain. Figure shows a simple MOS structure. The top layer of the structure is a good conductor called the gate. Early transistors used metal gates. Transistor gates soon changed to use polysilicon, i.e., silicon formed from many small crystals.



The middle layer is a very thin insulating film of SiO2 called the gate oxide. The bottom layer is the doped silicon body. The figure shows a p-type body in which the carriers are holes. The body is grounded and a voltage is applied to the gate. The gate oxide is a good insulator so almost zero current flows from the gate to the body.



In Figure (a) , a negative voltage is applied to the gate, so there is negative charge on the gate. The mobile positively charged holes are attracted to the region beneath the gate. This is called the accumulation mode. In Figure (b), a small positive voltage is applied to the gate, resulting in some positive charge on the gate. The holes in the body are repelled from the region directly beneath the gate, resulting in a depletion region forming below the gate. In Figure (c), a higher positive potential exceeding a critical threshold voltage Vt is applied, attracting more positive charge to the gate. The holes are repelled further and some free electrons in the body are attracted to the region beneath the gate. This conductive layer of electrons in the p-type body is called the inversion layer. The threshold voltage depends on the number of dopants in the body and the thickness tox of the oxide. It is usually positive, as shown in this example, but can be engineered to be negative.

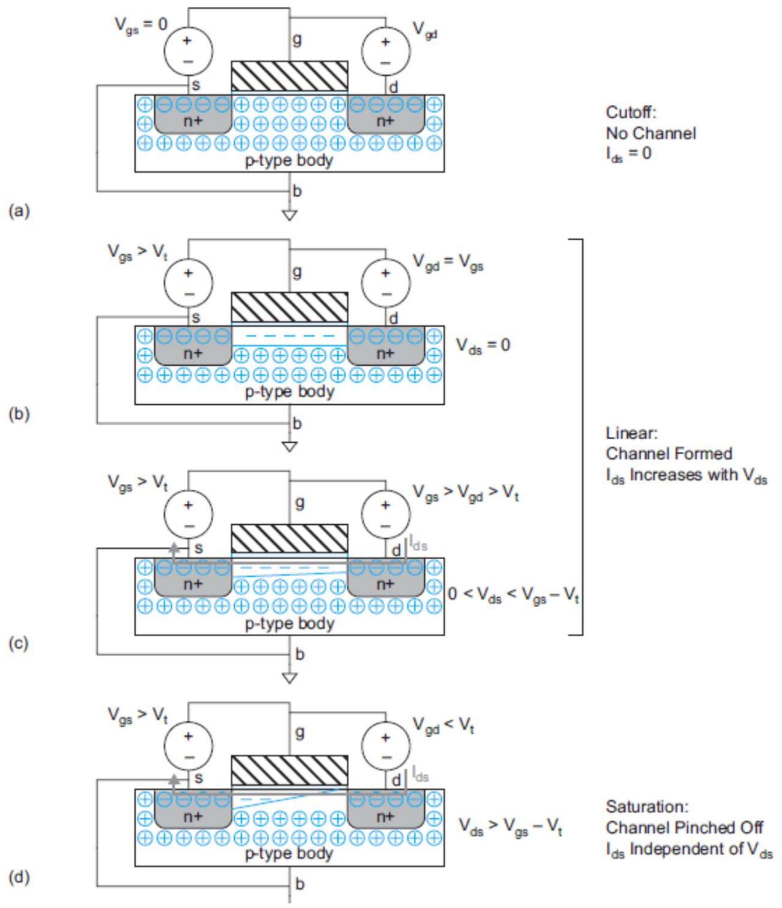


Figure shows an nMOS transistor. The transistor consists of the MOS stack between two n-type regions called the source and drain. In Figure (a), the gate-to-source voltage Vgs is less than the threshold voltage. The source and drain have free electrons. The body has free holes but no free electrons. Suppose the source is grounded. The junctions between the body and the source or drain are zero-biased or reverse-biased, so little or no current flows. We say the transistor is OFF, and this mode of operation is called cutoff. It is often convenient to approximate the current through an OFF transistor as zero, especially in comparison to the current through an ON transistor. Remember, however, that small amounts of current leaking through OFF transistors can become significant, especially when multiplied by millions or billions of transistors on a chip. In Figure (b), the gate voltage is greater than the threshold voltage. Now an inversion region of electrons (majority carriers) called the channel connects the source and drain, creating a conductive path and turning the transistor ON. The number of carriers and the conductivity increases with the gate voltage. The potential difference between drain and source is Vds = Vgs □ Vgd . If Vds = 0 (i.e., Vgs = Vgd), there is no electric field tending to push current from drain to source.

When a small positive potential Vds is applied to the drain (Figure (c)), current Ids flows through the channel from drain to source.2 This mode of operation is termed linear, resistive, triode, nonsaturated, or unsaturated; the current increases with both the drain voltage and gate voltage. If Vds becomes sufficiently large that Vgd < Vt , the channel is no longer inverted near the drain and becomes pinched off (Figure 2.3(d)). However, conduction is still brought about by the drift of electrons under the influence of the positive drain voltage. As electrons reach the end of the channel, they are injected into the depletion region near the drain and accelerated toward the drain. Above this drain voltage the current Ids is controlled only by the gate voltage and ceases to be influenced by the drain. This mode is called saturation.

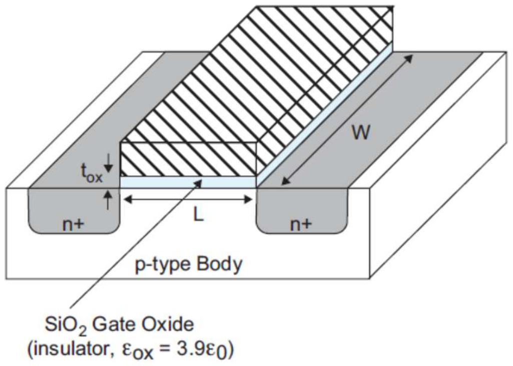
In summary, the nMOS transistor has three modes of operation. If Vgs < Vt , the transistor is cutoff (OFF). If Vgs > Vt , the transistor turns ON. If Vds is small, the transistor acts as a linear resistor in which the current flow is proportional to Vds. If Vgs > Vt and Vds is large, the transistor acts as a current source in which the current flow becomes independent of Vds . The pMOS transistor operates in just the opposite fashion.

***Long-Channel I-V Characteristics & MOS device design equations (First order effects)***

As stated previously, MOS transistors have three regions of operation:

- Cutoff or subthreshold region
- Linear region
- Saturation region

The long-channel model assumes that the current through an OFF transistor is 0. When a transistor turns ON (Vgs > Vt), the gate attracts carriers (electrons) to form a channel. The electrons drift from source to drain at a rate proportional to the electric field between these regions. Thus, we can compute currents if we know the amount of charge in the channel and the rate at which it moves. We know that the charge on each plate of a capacitor is Q = CV.We can model the gate as a parallel plate capacitor with capacitance proportional to area over thickness. The gate has length L and width W and the oxide thickness is tox, as shown in Figure.

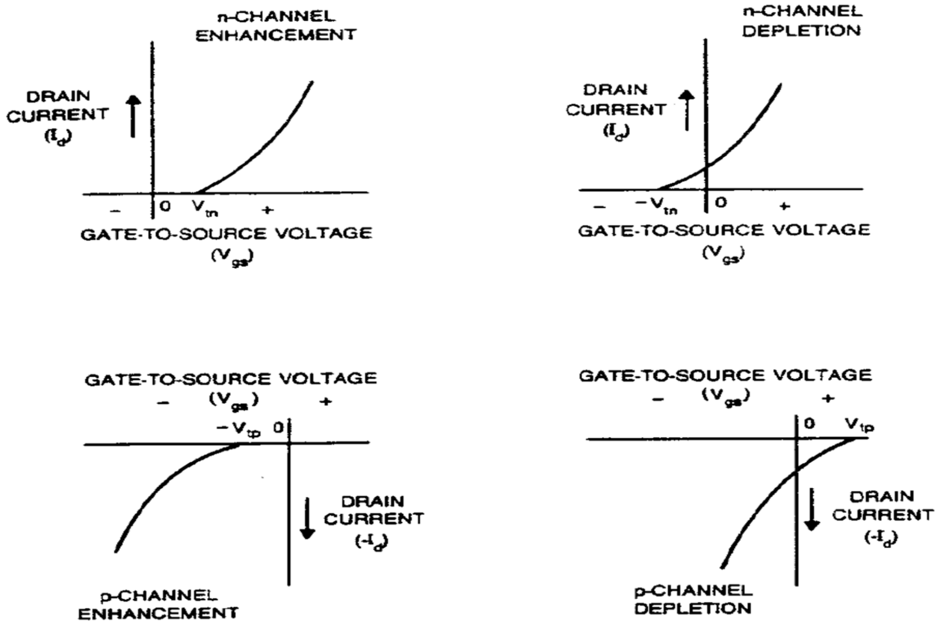


The ideal (first order) equations describing the behaviour of an nMOS device in the three regions are,

$$I_{ds} = \begin{cases} 0 & V_{gs} < V_t & \text{Cutoff} \\ \beta\left(V_{GT} - V_{ds}/2\right)V_{ds} & V_{ds} < V_{\text{dsat}} & \text{Linear} \\ \dfrac{\beta}{2}V_{GT}^2 & V_{ds} > V_{\text{dsat}} & \text{Saturation} \end{cases}$$

Figure (a) shows the I-V characteristics for the transistor. According to the first-order model, the current is zero for gate voltages below Vt. For higher gate voltages, current increases linearly with Vds for small Vds . As Vds reaches the saturation point Vdsat = VGT, current rolls off and eventually becomes independent of Vds when the transistor is saturated. We will later see that the Shockley model overestimates current at high voltage because it does not account for mobility degradation and velocity saturation caused by the high electric fields.

pMOS transistors behave in the same way, but with the signs of all voltages and currents reversed. The I-V characteristics are in the third quadrant, as shown in Figure (b). To keep notation simple in this text, we will disregard the signs and just remember that the current flows from source to drain in a pMOS transistor. The mobility of holes in silicon is typically lower than that of electrons. This means that pMOS transistors provide less current than nMOS transistors of comparable size and hence are slower.



The parameters that affect the magnitude of Ids,

- The distance between source and drain (channel length) and the channel width
- The threshold voltage
- The thickness of the gate oxide layer
- The dielectric constant of the gate insulator.
- The carrier (electron or hole) mobility

### *THRESHOLD VOLTAGE*

So far, we have treated the threshold voltage as a constant. However, Vt increases with the source voltage, decreases with the body voltage, decreases with the drain voltage, and increases with channel length. Until now, we have considered a transistor to be a three-terminal device with gate, source, and drain. However, the body is an implicit fourth terminal. When a voltage Vsb is applied between the source and body, it increases the amount of charge required to invert the channel, hence, it increases the threshold voltage. The threshold voltage can be modeled as

$$V_t = V_{t0} + \gamma\left(\sqrt{\phi_s + V_{sb}} - \sqrt{\phi_s}\right)$$

where Vt 0 is the threshold voltage when the source is at the body potential, $\phi_s$ is the surface potential at threshold and γ is the body effect coefficient, typically in the range 0.4 to 1 V1/2. In turn, these depend on the doping level in the channel, NA. The body effect further degrades the performance of pass transistors trying to pass the weak value (e.g., nMOS transistors passing a ‘1’).

Vt is largely determined at the time of fabrication, rather than by circuit conditions, like Ids. Most are related to the material properties. For example, material parameters that effect Vt include: The gate conductor material (poly vs. metal). The gate insulation material (SiO2). The thickness of the gate material. The channel doping concentration. From equations, threshold voltage may be varied by changing,

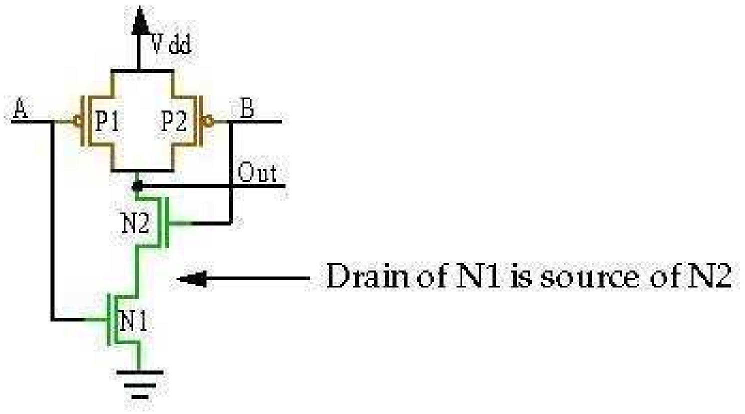
- The doping concentration (NA).
- The oxide capacitance (Cox)
- Surface state charge (Qfc).

Also it is often necessary to adjust Vt. Two methods are common. Change Qfc by introducing a small doped region at the oxide/substrate interface via ion implantation. Change Cox by using a different insulating material for the gate.

## ***BODY EFFECT OF MOS***

In digital circuits, the substrate is usually held at zero. The sources of n-channel devices, for example, are also held at zero, except in cases of series connections, e.g., as shown in figure. The source-to-substrate (Vsb) may increase at this connections, e.g. VsbN1= 0 but VsbN2/= 0 Vsb adds to the channel-substrate potential,

$$V_t = 2\phi_b + \frac{\sqrt{2\varepsilon_{Si}qN_A\left|2\phi_b + V_{sb}\right|}}{C_{ox}} + V_{fb}$$



Moreover, when the source of the nMOS transistor rises, Vsb becomes nonzero. This nonzero source to body potential introduces the body effect that increases the threshold voltage. Using the data from the example in that section, a pass transistor driven with VDD = 1 V would produce an output of only 0.65 V, potentially violating the noise margins of the next stage. Unlike ideal switches, MOS transistors pass some voltage levels better than others. An nMOS transistor passes 0s well, but only pulls up to VDD – Vtn when passing 1s. The pMOS passes 1s well, but only pulls down to |Vtp| when passing 0s. This threshold drop is exacerbated by the body effect, which increases the threshold voltage when the source is at a different potential than the body.

## 4. MOS INVERTERS

### *CMOS INVERTER TRANSFER CHARACTERISTICS*

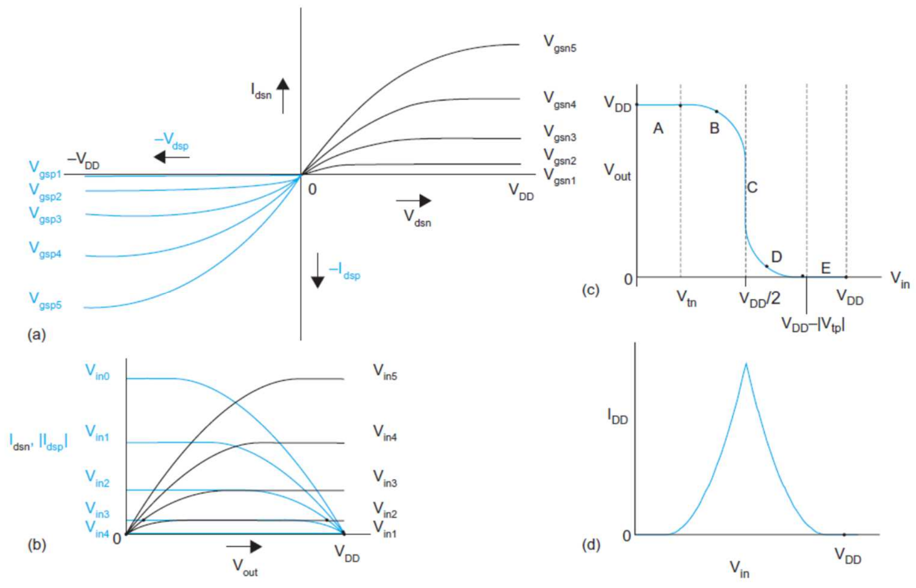
Digital circuits are merely analog circuits used over a special portion of their range. The DC transfer characteristics of a circuit relate the output voltage to the input voltage, assuming the input changes slowly enough that capacitances have plenty of time to charge or discharge. Specific ranges of input and output voltages are defined as valid 0 and 1 logic levels. This section explores the DC transfer characteristics of CMOS gates and pass transistors.

The DC transfer function (Vout vs. Vin) for the static CMOS inverter is shown in Figure. We begin with Table, which outlines various regions of operation for the n- and p-transistors. In this table, Vtn is the threshold voltage of the n-channel device, and Vtp is the threshold voltage of the p-channel device. Note that Vtp is negative. The equations are given both in terms of Vgs /Vds and Vin /Vout. As the source of the nMOS transistor is grounded, Vgsn = Vin and Vdsn = Vout. As the source of the pMOS transistor is tied to VDD, Vgsp = Vin – VDD and Vdsp = Vout – VDD.

**TABLE 2.2** Relationships between voltages for the three regions of operation of a CMOS inverter

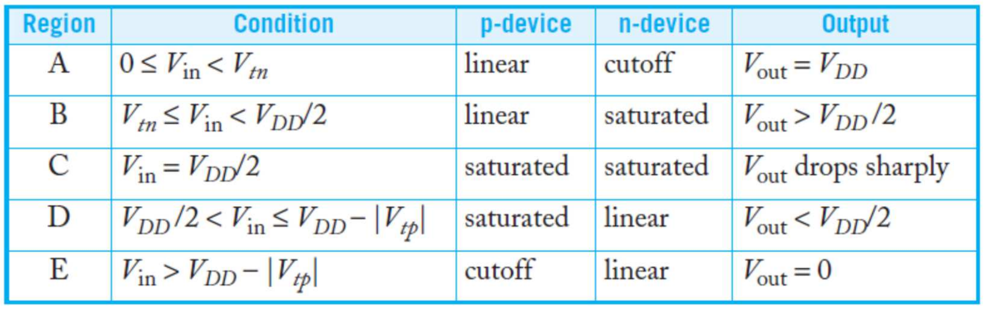
| | Cutoff | Linear | Saturated |
|---|---|---|---|
| nMOS | $V_{gsn} < V_{tn}$ | $V_{gsn} > V_{tn}$ | $V_{gsn} > V_{tn}$ |
| | $V_{\text{in}} < V_{tn}$ | $V_{\text{in}} > V_{tn}$ | $V_{\text{in}} > V_{tn}$ |
| | | $V_{dsn} < V_{gsn} - V_{tn}$ | $V_{dsn} > V_{gsn} - V_{tn}$ |
| | | $V_{\text{out}} < V_{\text{in}} - V_{tn}$ | $V_{\text{out}} > V_{\text{in}} - V_{tn}$ |
| pMOS | $V_{gsp} > V_{tp}$ | $V_{gsp} < V_{tp}$ | $V_{gsp} < V_{tp}$ |
| | $V_{\text{in}} > V_{tp} + V_{DD}$ | $V_{\text{in}} < V_{tp} + V_{DD}$ | $V_{\text{in}} < V_{tp} + V_{DD}$ |
| | | $V_{dsp} > V_{gsp} - V_{tp}$ | $V_{dsp} < V_{gsp} - V_{tp}$ |
| | | $V_{\text{out}} > V_{\text{in}} - V_{tp}$ | $V_{\text{out}} < V_{\text{in}} - V_{tp}$ |

The objective is to find the variation in output voltage (Vout) as a function of the input voltage (Vin). This may be done graphically. Given Vin, we must find Vout subject to the constraint that Idsn =|Idsp|. For simplicity, we assume Vtp = –Vtn and that the pMOS transistor is 2–3 times as wide as the nMOS transistor so □n = □p. The plot shows Idsn and Idsp in terms of Vdsn and Vdsp for various values of Vgsn and Vgsp. Figure (b) shows the same plot of Idsn and |Idsp| now in terms of Vout for various values of Vin. The possible operating points of the inverter, marked with dots, are the values of Vout where Idsn = |Idsp| for a given value of Vin. These operating points are plotted on Vout vs. Vin axes in Figure (c) to show the inverter DC transfer characteristics. The supply current IDD = Idsn = |Idsp| is also plotted against Vin in Figure (d) showing that both transistors are momentarily ON as Vin passes through voltages between GND and VDD, resulting in a pulse of current drawn from the power supply.



The operation of the CMOS inverter can be divided into five regions indicated on Figure c). The state of each transistor in each region is shown in Table. In region A, the nMOS transistor is OFF so the pMOS transistor pulls the output to VDD. In region B, the nMOS transistor starts to turn ON, pulling the output down. In region C, both transistors are in saturation. Notice that ideal transistors are only in region C for Vin = VDD/2 and that the slope of the transfer curve in this example is – in this region, corresponding to infinite gain. Real transistors have finite output resistances on account of channel length modulation, and thus have finite slopes over a broader region C. In region D, the pMOS transistor is partially ON and in region E, it is completely OFF, leaving the nMOS transistor to pull the output down to GND. Also notice that the inverter's current consumption is ideally zero, neglecting leakage, when the input is within a threshold voltage of the VDD or GND rails. This feature is important for low-power operation.

**TABLE 2.3** Summary of CMOS inverter operation



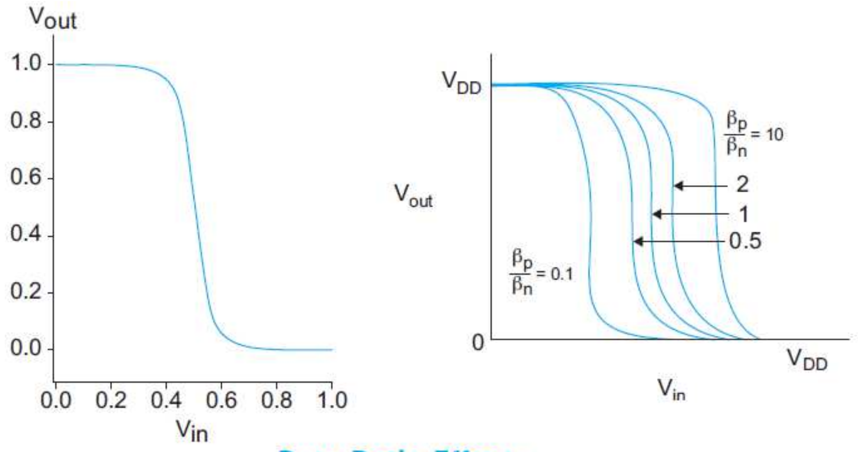
| Region | Condition | p-device | n-device | Output |
|---|---|---|---|---|
| A | $0 \le V_{\text{in}} < V_{tn}$ | linear | cutoff | $V_{\text{out}} = V_{DD}$ |
| B | $V_{tn} \le V_{\text{in}} < V_{DD}/2$ | linear | saturated | $V_{\text{out}} > V_{DD}/2$ |
| C | $V_{\text{in}} = V_{DD}/2$ | saturated | saturated | $V_{\text{out}}$ drops sharply |
| D | $V_{DD}/2 < V_{\text{in}} \le V_{DD} - \lvert V_{tp} \rvert$ | saturated | linear | $V_{\text{out}} < V_{DD}/2$ |
| E | $V_{\text{in}} > V_{DD} - \lvert V_{tp} \rvert$ | cutoff | linear | $V_{\text{out}} = 0$ |

***Beta Ratio Effects***

Figure shows simulation results of an inverter from a 65 nm process. The pMOS transistor is twice as wide as the nMOS transistor to achieve approximately equal betas. Simulation matches the simple models reasonably well, although the transition is not quite as steep because transistors are not ideal current sources in saturation. The crossover point where Vinv = Vin = Vout is called the input threshold. Because both mobility and the magnitude of the threshold voltage decrease with temperature for nMOS and pMOS transistors, the input threshold of the gate is only weakly sensitive to temperature.

We have seen that for □p = □n, the inverter threshold voltage Vinv is VDD/2. This may be desirable because it maximizes noise margins and allows a capacitive load to charge and discharge in equal times by providing equal current source and sink capabilities. Inverters with different beta ratios r = □p /□n are called skewed inverters [Sutherland99]. If r > 1, the inverter is HI-skewed. If r < 1, the inverter is LO-skewed. If r = 1, the inverter has normal skew or is unskewed.

A HI-skew inverter has a stronger pMOS transistor. Therefore, if the input is VDD /2, we would expect the output will be greater than VDD /2. In other words, the input threshold must be higher than for an unskewed inverter. Similarly, a LO-skew inverter has a weaker pMOS transistor and thus a lower switching threshold.



Beta Ratio Effects

Figure explores the impact of skewing the beta ratio on the DC transfer characteristics. As the beta ratio is changed, the switching threshold moves. However, the output voltage transition remains sharp. Gates are usually skewed by adjusting the widths of transistors while maintaining minimum length for speed. However, velocity saturated inverters are more sensitive to skewing because their DC transfer characteristics are not as sharp. DC transfer characteristics of other static CMOS gates can be understood by collapsing the gates into an equivalent inverter. Series transistors can be viewed as a single transistor of greater length. If only one of several parallel transistors is ON, the other transistors can be ignored. If several parallel transistors are ON, the collection can be viewed as a single transistor of greater width.

***NOISE MARGIN***

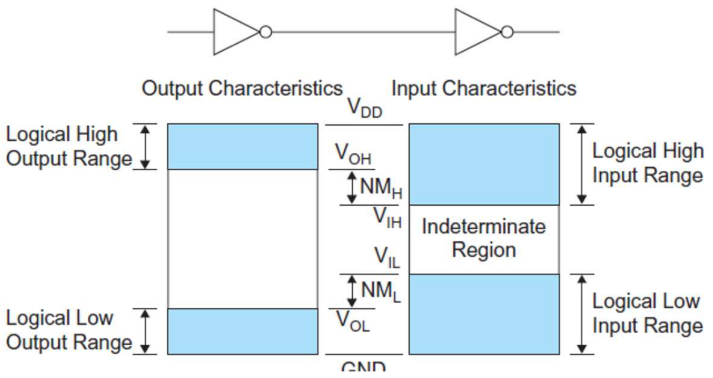
Noise margin is closely related to the DC voltage characteristics. This parameter allows you to determine the allowable noise voltage on the input of a gate so that the output will not be corrupted. The specification most commonly used to describe noise margin (or noise immunity) uses two parameters: the LOW noise margin, NML, and the HIGH noise margin, NMH. With reference to Figure, NML is defined as the difference in maximum LOW input voltage recognized by the receiving gate and the maximum LOW output voltage produced by the driving gate.

$$N_{ML} = V_{IL} - V_{OL}$$

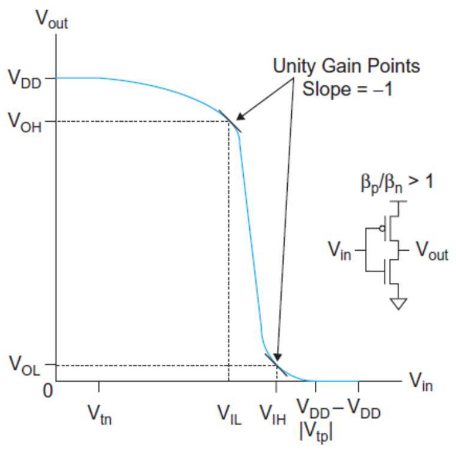
The value of NMH is the difference between the minimum HIGH output voltage of the driving gate and the minimum HIGH input voltage recognized by the receiving gate. Thus,

$$N_{MH} = V_{OH} - V_{IH}$$

where VIH = minimum HIGH input voltage, VIL = maximum LOW input voltage, VOH= minimum HIGH output voltage and VOL = maximum LOW output voltage.



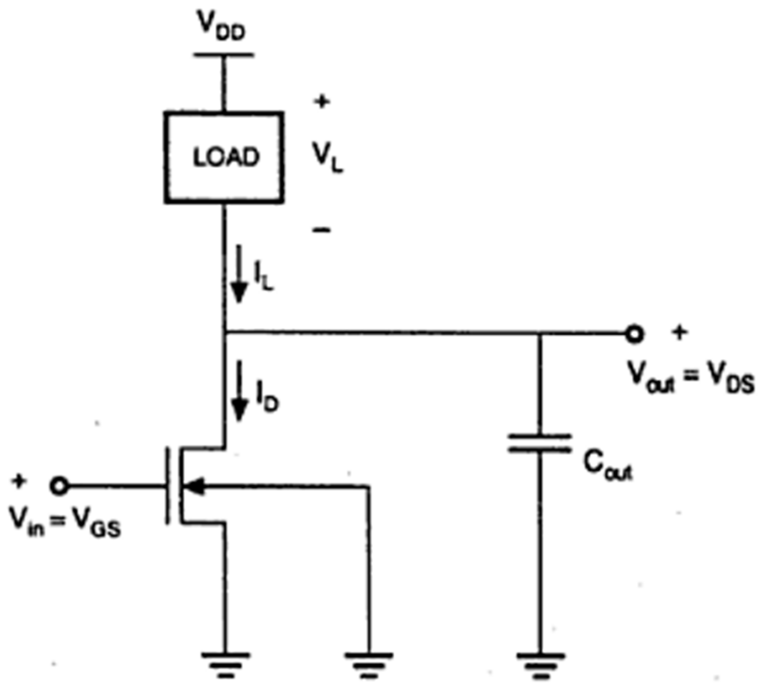
Inputs between VIL and VIH are said to be in the indeterminate region or forbidden zone and do not represent legal digital logic levels. Therefore, it is generally desirable to have VIH as close as possible to VIL and for this value to be midway in the "logic swing," VOL to VOH. This implies that the transfer characteristic should switch abruptly; that is, there should be high gain in the transition region. For the purpose of calculating noise margins, the transfer characteristic of the inverter and the definition of voltage levels VIL, VOL, VIH, and VOH are shown in Figure. Logic levels are defined at the unity gain point where the slope is –1.



If either NML or NMH for a gate are too small, the gate may be disturbed by noise that occurs on the inputs. An unskewed gate has equal noise margins, which maximizes immunity to arbitrary noise sources. If a gate sees more noise in the high or low input state, the gate can be skewed to improve that noise margin at the expense of the other. Note that if |Vtp| = Vtn , then NMH and NML increase as threshold voltages are increased. Quite often, noise margins are compromised to improve speed. Noise sources tend to scale with the supply voltage, so noise margins are best given as a fraction of the supply voltage. A noise margin of 0.4 V is quite comfortable in a 1.8 V process, but marginal in a 5 V process.

***NMOS INVERTER***

The inverter is truly the nucleus of all digital designs. Once its operation and properties are clearly understood, designing more intricate structures such as NAND gates, adders, multipliers, and microprocessors is greatly simplified. The electrical behavior of these complex circuits can be almost completely derived by extrapolating the results obtained for inverters. The analysis of inverters can be extended to explain the behavior of more complex gates such as NAND, NOR, or XOR, which in turn form the building blocks for modules such as multipliers and processors. In this chapter, we focus on one single incarnation of the inverter gate, being the static CMOS inverter — or the CMOS inverter, in short. This is certainly the most popular at present and therefore deserves our special attention.

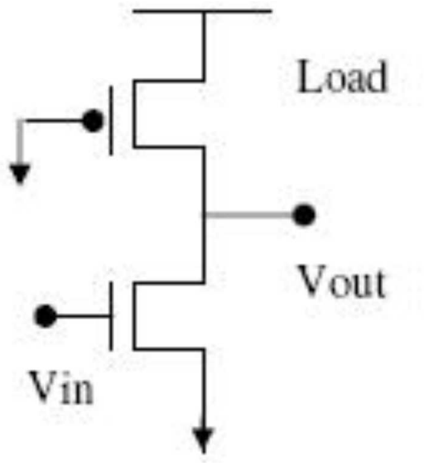


***Principle of Operation***

The logic symbol and truth table of ideal inverter is shown in figure given below. Here A is the input and B is the inverted output represented by their node voltages. Using positive logic, the Boolean value of logic 1 is represented by Vdd and logic 0 is represented by 0. Vth is the inverter threshold voltage, which is Vdd /2, where Vdd is the output voltage. The output is switched from 0 to Vdd when input is less than Vth. So, for 0<Vin<Vth output is equal to logic 0 input and Vth<Vin< Vdd is equal to logic 1 input for inverter. From the given figure, we can see that the input voltage of inverter is equal to the gate to source voltage of nMOS transistor and output voltage of inverter is equal to drain to source voltage of nMOS transistor. The source to substrate voltage of nMOS is also called driver for transistor which is grounded; so VSS = 0. The output node is connected with a lumped capacitance used for VTC.

***PSEUDO-NMOS INVERTER***

The inverter that uses a p-device pull-up or load that has its gate permanently ground. An n-device pull-down or driver is driven with the input signal. This roughly equivalent to use of a depletion load is Nmos technology and is thus called ‘Pseudo-NMOS’. The circuit is used in a variety of CMOS logic circuits. In this, PMOS for most of the time will be linear region. So resistance is low and hence RC time constant is low. When the driver is turned on a constant DC current flows in the circuit.



The CMOS pull up network is replaced by a single pMOS transistor with its gate grounded. Since the pMOS is not driven by signals, it is always ‘on'. The effective gate voltage seen by the pMOS transistor is Vdd. Thus the overvoltage on the p channel gate is always Vdd

-VTp. When the nMOS is turned ‘on', a direct path between supply and ground exists and static power will be drawn. However, the dynamic power is reduced due to lower capacitive loading.

***BI-CMOS INVERTER***

BICMOS logic circuits are made by combining the CMOS and bipolar IC technologies. These ICs combine the advantages of BJT and CMOS transistors in them. We know that the speed of BJT is very high compared to CMOS. However, power dissipation in CMOS is extremely low compared to BJT. By combining such advantages, we construct the BICMOS circuits. Figure shows one configuration of the BICMOS inverter, and Fig. shows its modified version. We see that MOS transistors T3 and T4 form the CMOS inverter logic circuit. We find that T3 and T4 are driven separately from +VDD//VCC rail. With input voltage Vi = 0, the PMOS will conduct and the NMOS will remain OFF. This drives a current through the base of the bipolar junction transistor T1 and turns it ON. This in turn charges the parasitic load capacitance CL at a very fast rate. Thus the output voltage Vo rises very fast, which is usually much faster than the charging of CL by an MOS transistor by itself.

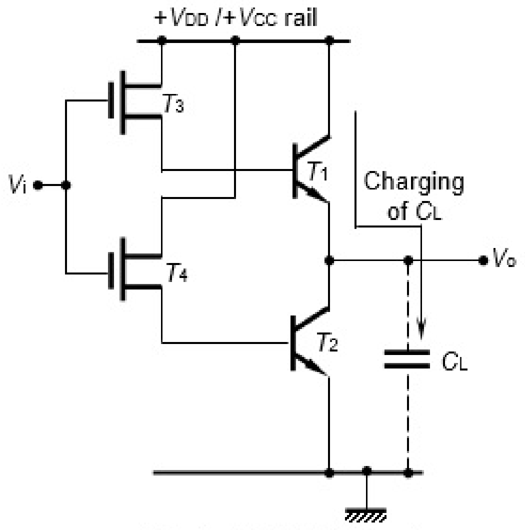


**Fig. 3.43** BICMOS inverter

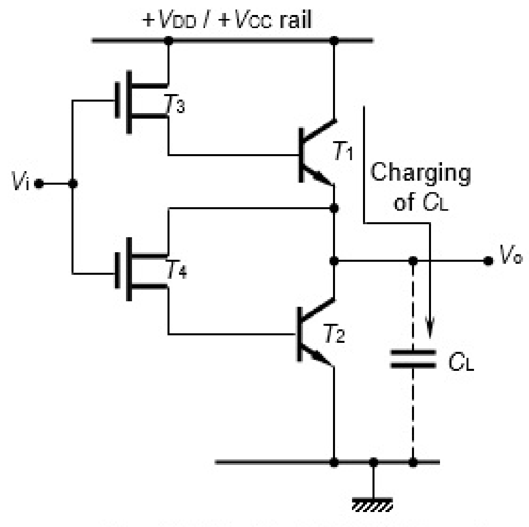


**Fig. 3.44** Modified BICMOS inverter

Now, let Vi = +VDD (≡ logic 1). Then T4 will turn ON and T3 will turn OFF; this drives T2 into the ON-state. Since T3 is OFF, T1 will also remain OFF. In this condition, CL discharges very fast through T2ON. Thus the charging and discharging of CL is through BJTs and hence very fast. The circuit shown in Fig. 3.43 has two major defects. The first of these is that whenever the input is at logic 1, since T3 is ON, there will be a continuous path from +VDD to ground. As a result, steady power drain will occur in this case. This is totally against the advantages of CMOS gates. The second defect is that there is no discharge path for the base currents of T1 and T2. This will therefore reduce the speed of the circuit.

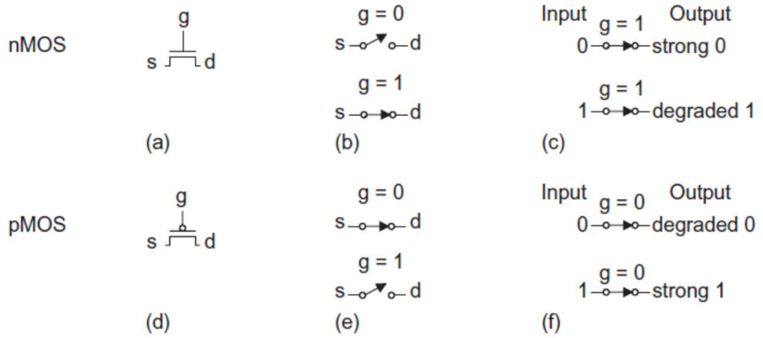
To overcome these problems, we modify the circuit, as shown in Figure. In this case, NMOS transistor T4 has its drain connected to the output terminal rather than to +VDD. As before, when T3 is turned ON, T1 is also turned ON. Now, when T4 is turned ON, we find T2 also to turn ON.. However, since the collector and base of T2 are shorted together through T4ON, the output voltage Vo will now be equal VBES, the saturation base-emitter voltage of T2 (= 0.8 V). Thus in this case, the output swing is between VCC and VBES.

***Features of BICMOS gates:***

1. This has the advantages of both the BJTs and CMOS gates.
2. The power driver (BJT amplifier) in the output stage is capable of driving large loads.
3. The circuit, because of its CMOS input transistors, has high input impedance.
4. The output impedance of the circuit is low.
5. The noise margin is high because of the CMOS input stage.
6. The supply voltage VDD is 5 V.
7. The chip area is small.

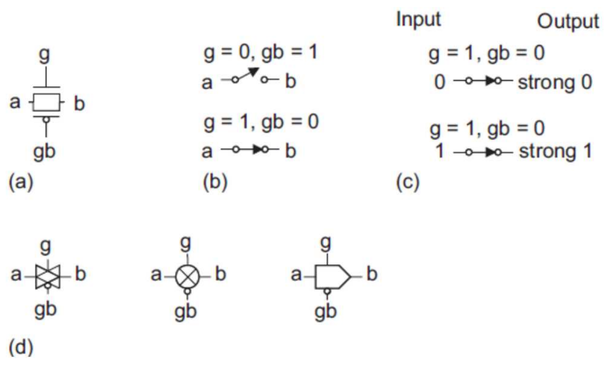
***CMOS TRANSMISSION GATE***

The strength of a signal is measured by how closely it approximates an ideal voltage source. In general, the stronger a signal, the more current it can source or sink. The power supplies, or rails, (VDD and GND) are the source of the strongest 1s and 0s. 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 VDD. We say it passes a degraded or weak 1.



A pMOS transistor again has the opposite behavior, passing strong 1s but degraded 0s. The transistor symbols and behaviors are summarized in Figure with g, s, and d indicating gate, source, and drain.

When an nMOS or pMOS is used alone as an imperfect switch, we sometimes call it a pass transistor. By combining an nMOS and a pMOS transistor in parallel (Figure (a)), we obtain a switch that turns on when a 1 is applied to g (Figure (b)) in which 0s and 1s are both passed in an acceptable fashion (Figure (c)). We term this a transmission gate or pass gate. In a circuit where only a 0 or a 1 has to be passed, the appropriate transistor (n or p) can be deleted, reverting to a single nMOS or pMOS device. Note that both the control input and its complement are required by the transmission gate. This is called double rail logic. Some circuit symbols for the transmission gate are shown in Figure (d).



In all of our examples so far, the inputs drive the gate terminals of nMOS transistors in the pull-down network and pMOS transistors in the complementary pull-up network, as was shown in Figure. Thus, the nMOS transistors only need to pass 0s and the pMOS only pass 1s, so the output is always strongly driven and the levels are never degraded. This is called a fully restored logic gate and simplifies circuit design considerably. In contrast to other forms of logic, where the pull-up and pull-down switch networks have to be ratioed in some manner, static CMOS gates operate correctly independently of the physical sizes of the transistors. Moreover, there is never a path through 'ON' transistors from the 1 to the 0 supplies for any combination of inputs (in contrast to single-channel MOS, GaAs technologies, or bipolar). As we will find in subsequent chapters, this is the basis for the low static power dissipation in CMOS.

## CONCLUSION:

CMOS technology, driven by Moore’s Law, has come to dominate the semiconductor industry. This chapter examined the principles of designing a simple CMOS integrated circuit. MOS transistors can be viewed as electrically controlled switches. Static CMOS gates are built from pull-down networks of nMOS transistors and pull-up networks of pMOS transistors. we have seen that MOS transistors are four-terminal devices with a gate, source, drain, and body. In normal operation, the body is tied to GND or VDD so the transistor can be modeled as a three-terminal device. The transistor behaves as a voltage controlled switch. An nMOS switch is OFF (no path from source to drain) when the gate voltage is below some threshold Vt . The switch turns ON, forming a channel connecting source to drain, when the gate voltage rises above Vt . This chapter has developed more elaborate models to predict the amount of current that flows when the transistor is ON.

Even when the gate voltage is low, the transistor is not completely OFF. Subthreshold current through the channel drops off exponentially for $V_{gs} < V_t$ , but is nonnegligible for transistors with low thresholds. Junction leakage currents flow through the reverse-biased p–n junctions. Tunneling current flows through the insulating gate when the oxide becomes thin enough. We can derive the DC transfer characteristics and noise margins of logic gates using either analytical expressions or a graphical load line analysis or simulation. Static CMOS gates have excellent noise margins. Unlike ideal switches, MOS transistors pass some voltage levels better than others. An nMOS transistor passes 0s well, but only pulls up to VDD – Vtn when passing 1s. The pMOS passes 1s well, but only pulls down to |Vtp| when passing 0s. This threshold drop is exacerbated by the body effect, which increases the threshold voltage when the source is at a different potential than the body.

# Notes on
# CMOS Processing Technology and Layouts
# in VLSI Design – Unit II

**Dr. G. Senthil Kumar,**
Associate Professor,
Dept. of ECE, SCSVMV,
email: gsk_ece@kanchiuniv.ac.in

=================================================================

**OBJECTIVES**:

In this lesson, you will be able realize the concepts of CMOS circuits using processing technology and layouts.

**CONTENTS**:

## 1. SILICON SEMICONDUCTOR FABRICATION TECHNOLOGY:

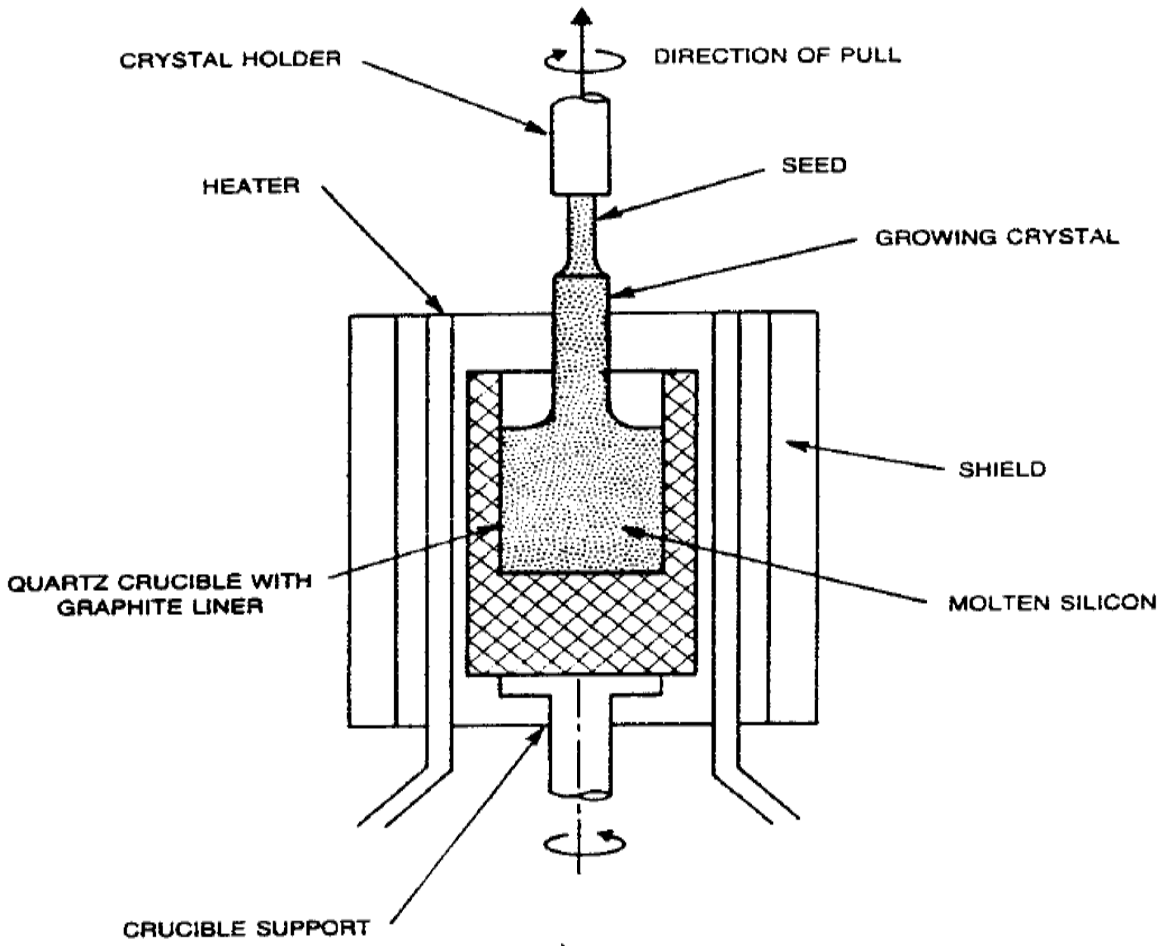
The silicon in its pure or intrinsic state is a semiconductor, having bulk electrical resistance somewhere between that of a conductor and an insulator. The conductivity of silicon can be raised by several orders of magnitude by introducing impurity atoms into the silicon crystal lattice. These dopants can supply either free electrons or holes. Group III impurity elements such as boron that use up electrons are referred to as acceptors because they accept some of the electrons already in the silicon, leaving holes. Similarly, Group V donor elements such as arsenic and phosphorous provide electrons. Silicon that contains a majority of donors is known as n-type, while silicon that contains a majority of acceptors is known as p-type. When n-type and p-type materials are brought together, the region where the silicon changes from n-type to p-type is called a junction. By arranging junctions in certain physical structures and combining them with wires and insulators, various semiconductor devices can be constructed. Over the years, silicon semiconductor processing has evolved sophisticated techniques for building these junctions and other insulating and conducting structures.

***Wafer Formation***

The basic raw material used in CMOS fabs is a wafer or disk of silicon, roughly 75 mm to 300 mm (12 microm—a dinner plate!) in diameter and less than 1 mm thick. Wafers are cut from boules, cylindrical ingots of single-crystal silicon, that have been pulled from a crucible of pure molten silicon. This is known as the Czochralski method and is currently the most common method for producing single-crystal material. Controlled amounts of impurities are added to the melt to provide the crystal with the required electrical properties. A seed crystal is dipped into the melt to initiate crystal growth. The silicon ingot takes on the same crystal orientation as the seed. A graphite radiator heated by radio-frequency induction surrounds the quartz crucible and maintains the temperature a few degrees above the melting point of silicon (1425 °C). The atmosphere is typically helium or argon to prevent the silicon from oxidizing. The seed is gradually withdrawn vertically from the melt while simultaneously being rotated, as shown in Figure 3.2. The molten silicon attaches itself to the seed and recrystallizes as it is withdrawn. The seed withdrawal and rotation rates determine the diameter of the ingot. Growth rates vary from 30 to 180 mm/hour.

Fabrication plants, or fabs, are enormously expensive to develop and operate. In the early days of the semiconductor industry, a few bright physicists and engineers could bring up a fabrication facility in an industrial building at a modest cost and most companies did their own manufacturing. Modern CMOS processing is complex, and while coverage of every nuance is not within the scope of this book, we focus on the fundamental concepts that impact design.

The regions of dopants, polysilicon, metal, and contacts are defined using masks. For instance, in places covered by the mask, ion implantation might not occur or the dielectric or metal layer might be left intact. In areas where the mask is absent, the implantation can occur, or dielectric or metal could be etched away. The patterning is achieved by a process called photolithography, from the Greek photo (light), lithos (stone), and graphe (picture), which literally means “carving pictures in stone using light.” The primary method for defining areas of interest (i.e., where we want material to be present or absent) on a wafer is by the use of photoresists. The wafer is coated with the photoresist and subjected to selective illumination through the photomask. After the initial patterning of photoresist, other barrier layers such as polycrystalline silicon, silicon dioxide, or silicon nitride can be used as physical masks on the chip. This distinction will become more apparent as this chapter progresses.

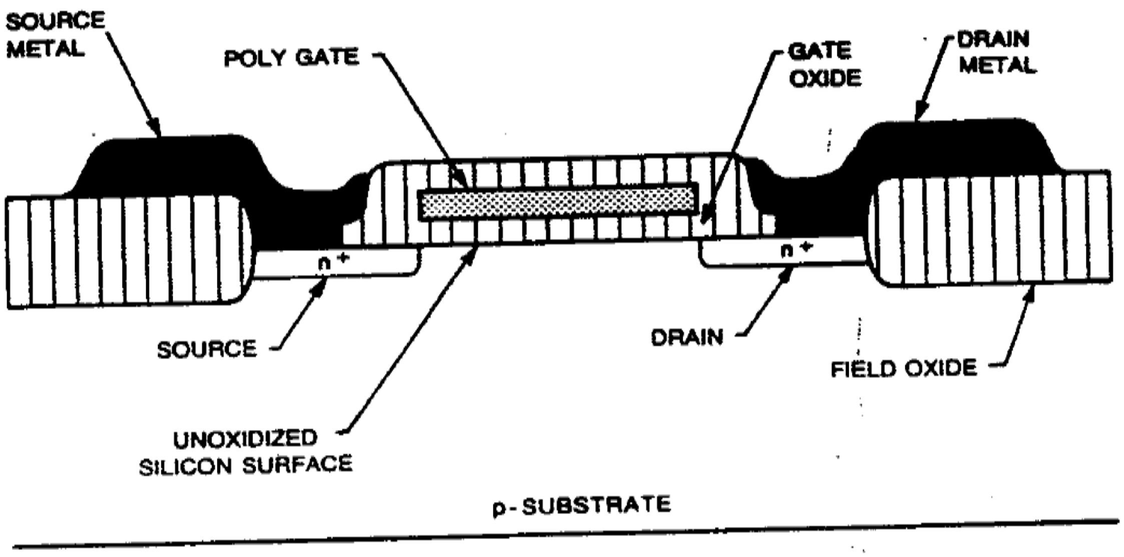


After the seed is dipped into the melt, the seed is gradually withdrawn vertically from the melt while simultaneously being rotated. The molten polycrystalline silicon melts the tip of the seed and as it is withdrawn, refreezing occurs. As the melt freezes, it assumes the single crystal form of the seed. This process is continued until the melt is consumed. The diameter of the ingot is determined by the seed withdrawal rate and the seed rotation rate. Growth rates range from 30 to 180 mm/hour.

Slicing into wafers is usually carried out using internal cutting edge diamond blades. Wafers are usually between 0.25 mm and 1.0 mm thick, depending on their diameter. Following this operation, at least one face is polished to a flat, scratch-free mirror finish.

### 3.1.2 Oxidation

Many of the structures and manufacturing techniques used to make silicon integrated circuits rely on the properties of the oxide of silicon, namely, silicon dioxide ($SiO_2$). Therefore the reliable manufacture of $SiO_2$ is extremely important.



Oxidation of silicon is achieved by heating silicon wafers in an oxidizing atmosphere such as oxygen or water vapor. The two common approaches are:

- Wet oxidation: when the oxidizing atmosphere contains water vapor. The temperature is usually between 900° C and 1000° C. This is a rapid process.
- Dry oxidation: when the oxidizing atmosphere is pure oxygen. Temperatures are in the region of 1200° C, to achieve an acceptable growth rate.

The oxidation process consumes silicon. Since $SiO_2$ has approximately twice the volume of silicon, the $SiO_2$ layer grows almost equally in both vertical directions. This effect is shown in Fig. 3.2 for an n-channel MOS device in which the $SiO_2$ (field oxide) projects above and below the unoxidized silicon surface.

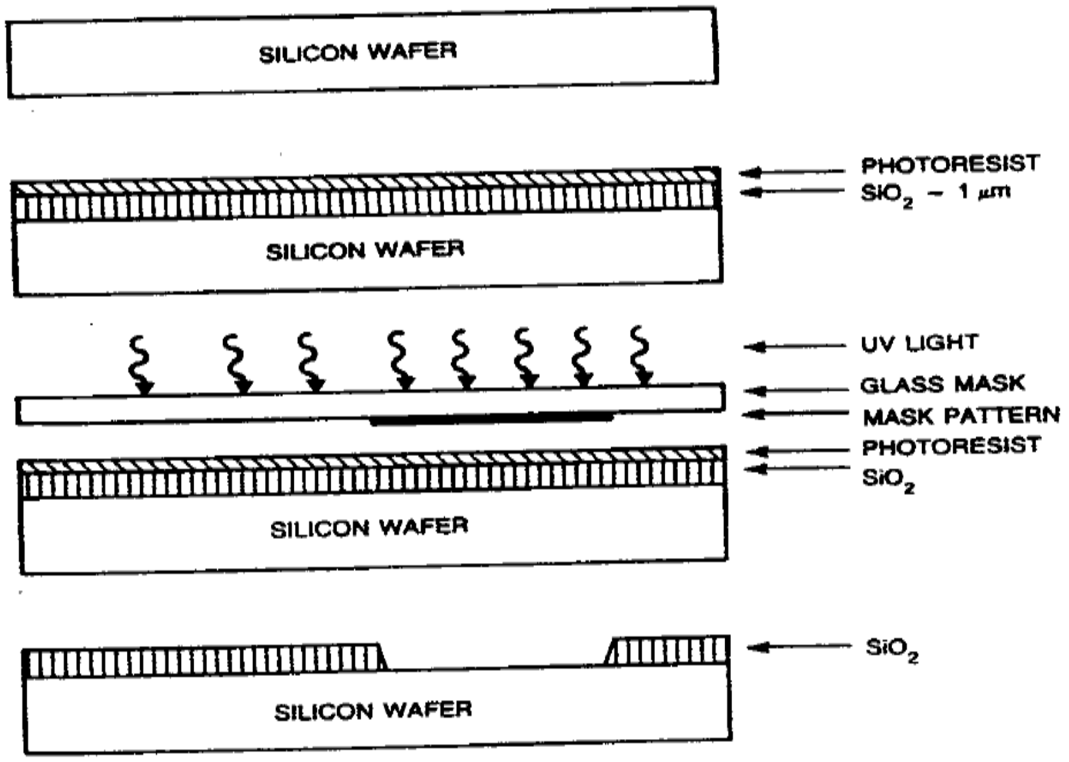
### 3.1.3 Selective diffusion

To create different types of silicon, containing different proportions of donor or acceptor impurities, further processing is required. As these areas are required to be precisely placed and sized, a means of ensuring this is required. The ability of $SiO_2$ to act as a barrier against doping impurities is a vital factor in this process called *selective diffusion*. The $SiO_2$ layer may be used as a *pattern mask*. Areas on the silicon wafer surface where there is an absence of $SiO_2$ allow dopant atoms to pass into the wafer, thus changing the characteristics of the silicon. Areas where $SiO_2$ overlays the silicon act as barriers to the dopant atoms. Thus selective diffusion entails:

- Opening *windows* in a layer of $SiO_2$ grown on the surface of the wafer.
- Removing $SiO_2$, but not $Si$, with a suitable etchant.
- Subjecting exposed $Si$ to a dopant source.

The process used for selectively removing the oxide involves covering the surface of the oxide with an acid resistant coating, except where diffusion windows are needed. The $SiO_2$ is removed using an etching technique. The acid resistant coating is normally a photosensitive organic material called *photoresist* (PR), which can be polymerized by ultraviolet (UV) light. If the UV light is passed through a mask containing the desired pattern, the coating can be polymerized where the pattern is to appear. The unpolymerized areas may be removed with an organic solvent. Etching of exposed $SiO_2$ then may proceed. This process is illustrated in Fig. 3.3. In established processes using PRs in conjunction with UV light sources, diffraction around the edges of the mask patterns and alignment tolerances have limited line widths on the order of about 1.5 $\mu m$ to 2 $\mu m$. However, during recent years, electron beam lithography (EBL) has emerged as a contender for pattern generation and imaging where line widths of the order of 0.5 $\mu m$ with good definition are achievable. The main advantages of EBL pattern generation are:

- Patterns are derived directly from digital data.



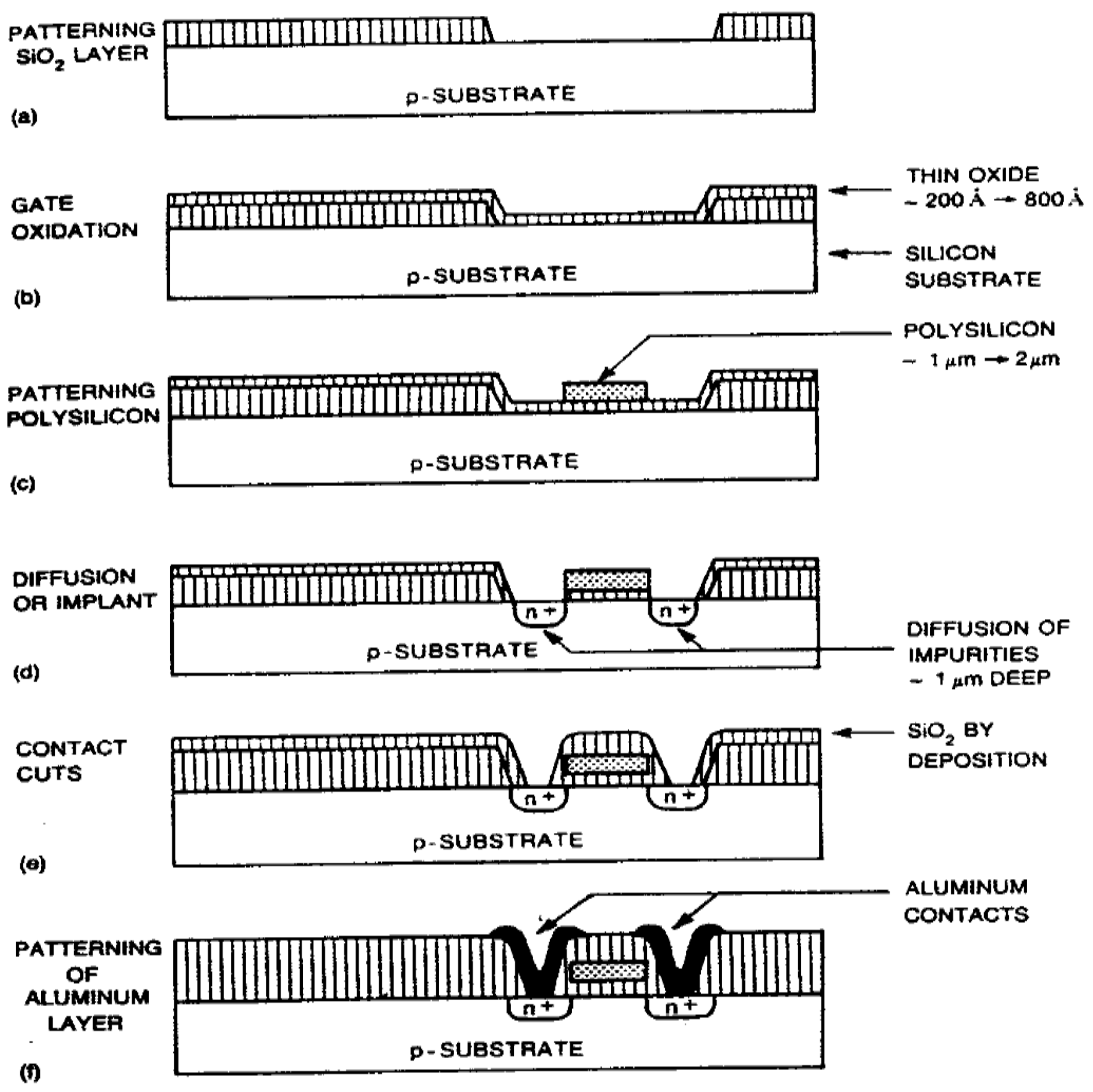
- There are no intermediate hardware images such as recticles or masks; that is, the process may be direct.
- Different patterns may be accommodated in different sections of the wafer without difficulty.
- Changes to patterns can be implemented quickly.

The main disadvantage that has precluded the use of this technique in commercial fabrication lines is the cost of equipment and the large amount of time required to access all points on the wafer.

### 3.1.4 The silicon gate process

So far we have touched on the single crystal form of silicon used in the manufacture of wafers and the oxide used in the manufacture and operation of circuits. Silicon may also be formed in an amorphous form (not having a carefully arranged lattice structure) commonly called *polycrystalline* silicon or *polysilicon*. This is used as an interconnect in silicon ICs and as the gate electrode on MOS transistors. The most significant aspect of using polysilicon as the gate electrode is its ability to be used as a further mask to allow precise definition of source and drain electrodes. This is achieved with minimum gate-to-source/drain overlap, which we will learn improves circuit performance. Polysilicon is formed when silicon is deposited on $SiO_2$ or other surfaces. In the case of an MOS transistor gate electrode, undoped polysilicon is deposited on the gate insulator. Polysilicon and source/drain regions are then normally doped at the same time. Undoped polysilicon has high resistivity. This characteristic is used to provide high value resistors in static memories. The resistivity of polysilicon may be reduced by combining it with a refractory metal (see Section 3.2.5).

The steps involved in a typical silicon gate process entail photomasking and oxide etching, which are repeated a number of times during the processing sequence. Fig. 3.4 shows the processing steps after the initial patterning of the $SiO_2$, which was shown in Fig. 3.3. The wafer is initially covered with a thick layer of $SiO_2$ called the *field oxide*. The field oxide is etched to the silicon surface in areas where transistors are to be placed (Fig. 3.4a). A thin, highly controlled layer of $SiO_2$ is then grown on the exposed silicon surface. This is called the *gate oxide* or *thin oxide* or *thinox* (Fig. 3.4b). Polysilicon is then deposited over the wafer surface and etched to form interconnections and transistor gates. Fig. 3.4c shows the result of an etched polysilicon gate. The exposed thinox (not covered by polysilicon) is then etched away. The complete wafer is then exposed to a dopant source, resulting in two actions (Fig. 3.4d). Diffusion



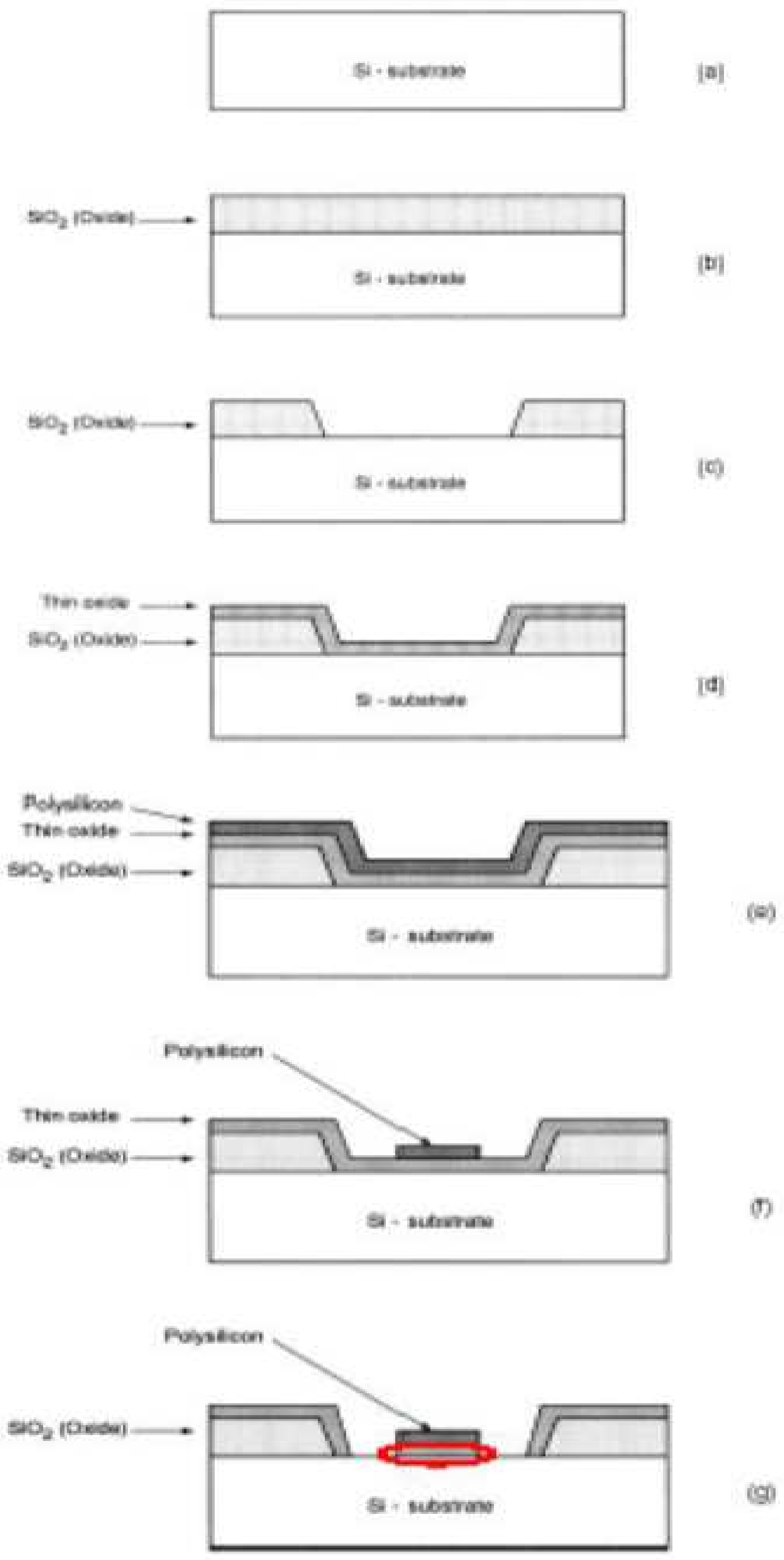
junctions are formed in the substrate and the polysilicon is doped with the particular type of dopant. This reduces the resistivity of the polysilicon. Note that the diffusion junctions form the drain and source of the MOS transistor. They are formed only in regions where the polysilicon gate does not shadow the underlying substrate. This is referred to as a *self-aligned* process because the source and drain do not extend under the gate. Finally, the complete structure is covered with $SiO_2$ and contact holes are etched to make contact with underlying layers (Fig. 3.4e). Aluminum or other metallic interconnect is evaporated and etched to complete the final connection of elements (Fig. 3.4f).

## LAYER REPRESENTATIONS FOR LAYOUTS

### PROCESS

| | p -WELL | n-WELL | TWIN-TUB |
|---|---|---|---|
| – – – – – – – | p -WELL | n-WELL | p -WELL |
| ———— | THINOXIDE | THINOXIDE | THINOXIDE |
| ▨ | POLYSILICON | POLYSILICON | POLYSILICON |
| —··—··— | p-PLUS | p-PLUS | p-PLUS |
| — — — | ALUMINUM (METAL 1) | ALUMINUM | ALUMINUM |
| —····—····— | METAL 2 | METAL 2 | METAL 2 |
| ▩ | CONTACT | CONTACT | CONTACT |
| ▧ | POLYSILICON 2 | POLY 2 | POLY 2 |
| □ | VIA | VIA | VIA |

## *FABRICATION FORMS -NMOS***:**



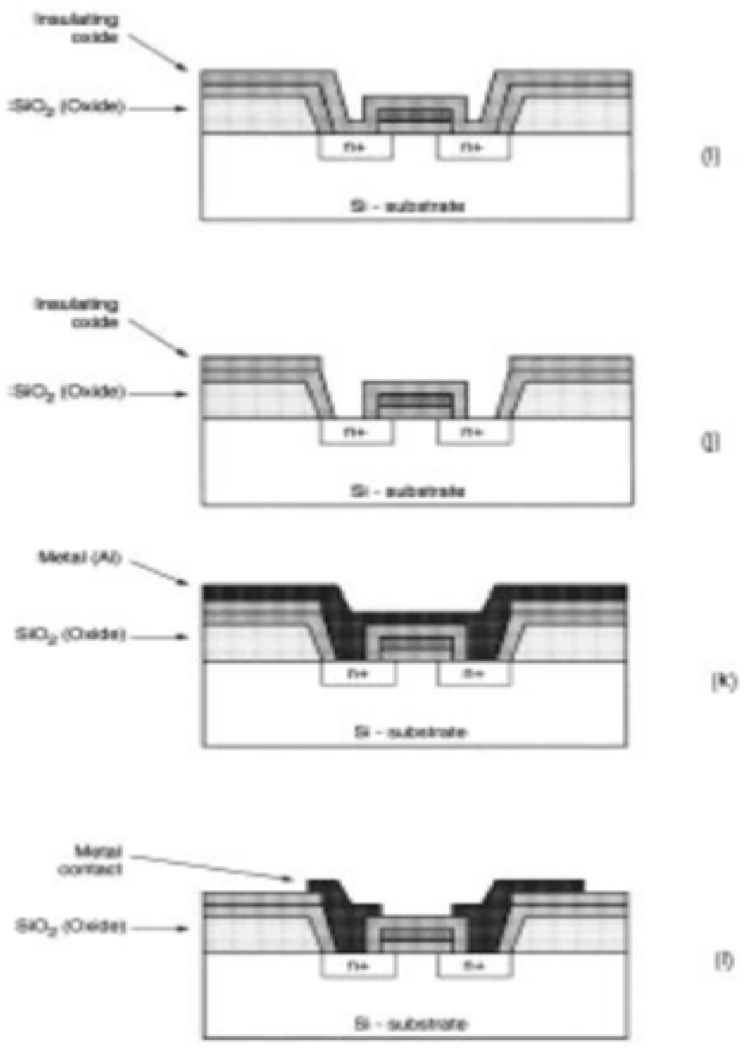


Figure8. NMOS Fabrication process steps

The process starts with the oxidation of the silicon substrate (Fig. 8(a)), in which a relatively thick silicon dioxide layer, also called field oxide, is created on the surface (Fig. 8(b)). Then, the field oxide is selectively etched to expose the silicon surface on which the MOS transistor will be created (Fig. 8(c)). Following this step, the surface is covered with a thin, high-quality oxide layer, which will eventually form the gate oxide of the MOS transistor (Fig. 8(d)). On top of the thin oxide, a layer of polysilicon (polycrystalline silicon) is deposited (Fig. 8(e)). Polysilicon is used both as gate electrode material for MOS transistors and also as an interconnect medium in silicon integrated circuits. Undoped polysilicon has relatively high resistivity. The resistivity of polysilicon can be reduced, however, by doping it with impurity atoms.

After deposition, the polysilicon layer is patterned and etched to form the interconnects and the MOS transistor gates (Fig. 8(f)). The thin gate oxide not covered by polysilicon is also etched away, which exposes the bare silicon surface on which the source and drain junctions are to be formed (Fig. 8(g)). The entire silicon surface is then doped with a high concentration of impurities, either through diffusion or ion implantation (in this case with donor atoms to produce n-type doping). Figure 8(h) shows that the doping penetrates the exposed areas on the silicon surface, ultimately creating two n-type regions (source and drain junctions) in the p-type substrate.

The impurity doping also penetrates the polysilicon on the surface, reducing its resistivity. Note that the polysilicon gate, which is patterned before doping actually defines the precise location of the channel region and, hence, the location of the source and the drain regions. Since this procedure allows very precise positioning of the two regions relative to the gate, it is also called the self-aligned process.

Once the source and drain regions are completed, the entire surface is again covered with an insulating layer of silicon dioxide (Fig. 8 (i)). The insulating oxide layer is then patterned in order to provide contact windows for the drain and source junctions (Fig. 8 (j)). The surface is covered with evaporated aluminum which will form the interconnects (Fig. 8 (k)). Finally, the metal layer is patterned and etched, completing the interconnection of the MOS transistors on the surface (Fig. 8 (l)). Usually, a second (and third) layer of metallic interconnect can also be added on top of this structure by creating another insulating oxide layer, cutting contact (via) holes, depositing, and patterning the metal.

## CMOS Fabrication

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

## P-WELL PROCESS:

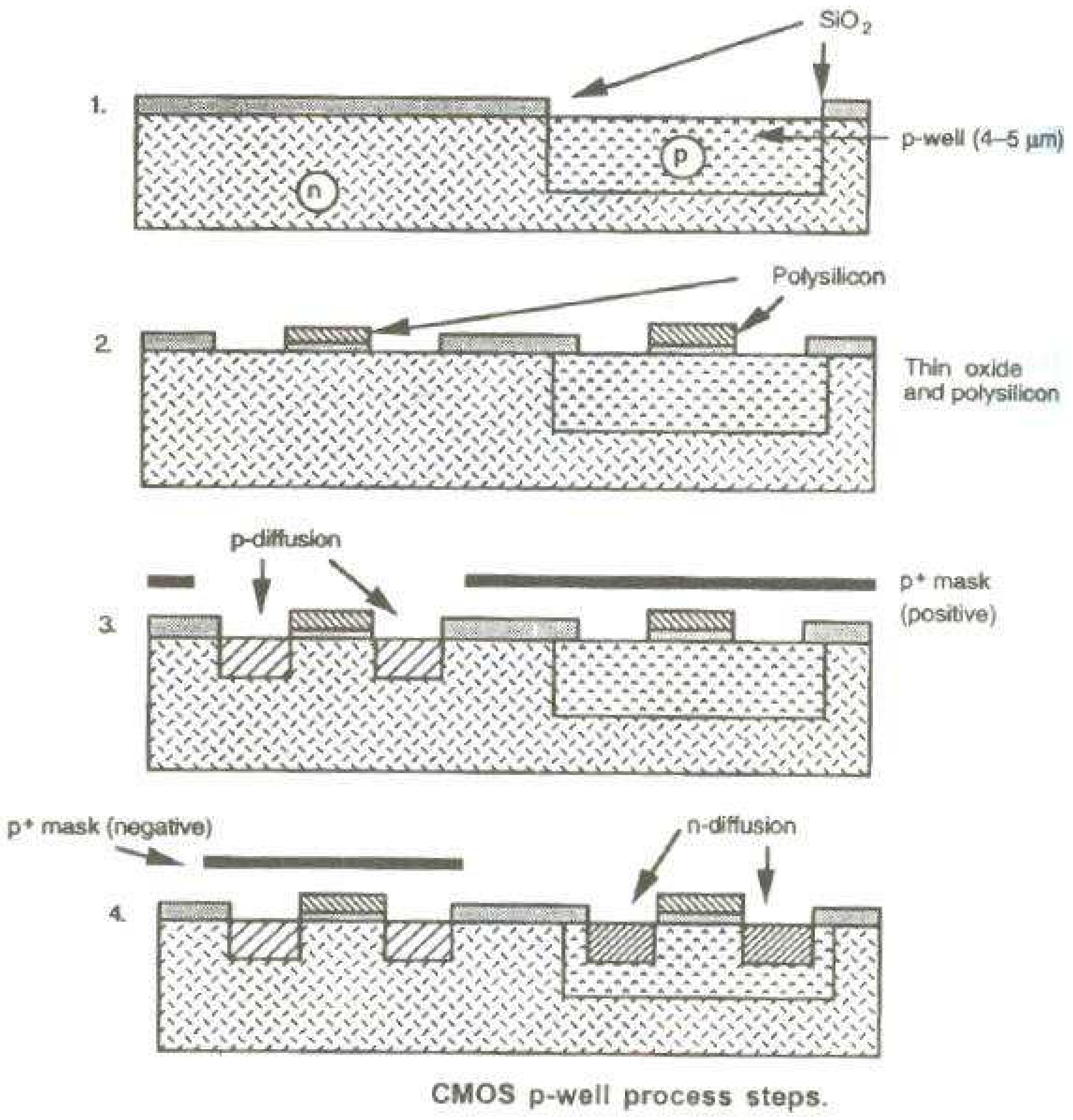


Figure9. CMOS Fabrication (P-WELL) process steps.

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

Mask sequence.

Mask 1:

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

Mask 2:

It defines the thin oxide region (where the thick oxide is to be removed or stripped and thin oxide grown)

Mask 3:

It's used to pattern the polysilicon layer which is deposited after thin oxide. Mask 4: A p+ mask (anded with mask 2) to define areas where p-diffusion is to take place.

Mask 5:

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

Mask 6:

Contact cuts are defined using this mask.

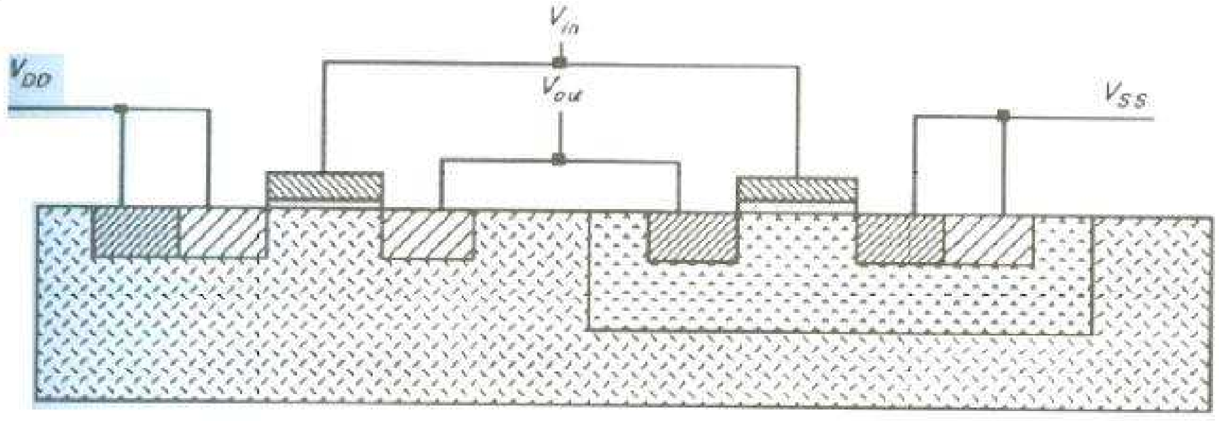
Mask 7:

The metal layer pattern is defined by this mask.

Mask 8:

An overall passivation (over glass) is now applied and it also defines openings for accessing pads.

The cross section below shows the CMOS pwell inverter.

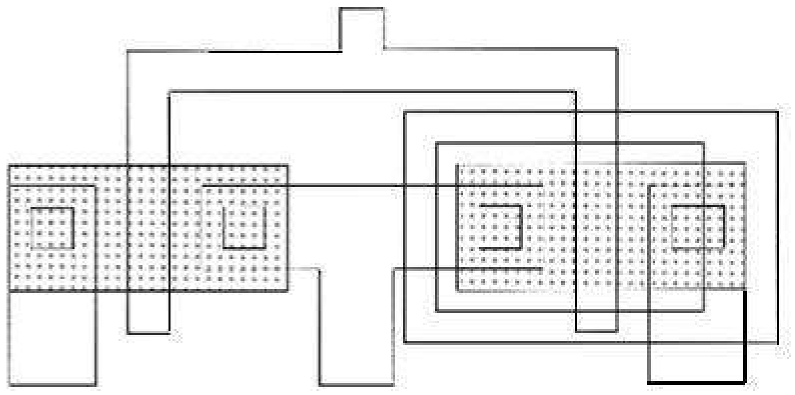


CMOS p-well inverter showing $V_{DD}$ and $V_{SS}$ substrate connections.

Figure10. CMOS inverter (P-WELL)

## 2. BASIC N-WELL PROCESS:

In the following figures, some of the important process steps involved in the fabrication of a CMOS inverter will be shown by a top view of the lithographic masks and a cross-sectional view of the relevant areas. The n-well CMOS process starts with a moderately doped (with impurity concentration typically less than 1015 cm-3) p-type silicon substrate. Then, an initial oxide layer is grown on the entire surface. The first lithographic mask defines the n-well region. Donor atoms, usually phosphorus, are implanted through this window in the oxide. Once the n-well is created, the active areas of the nMOS and pMOS transistors can be defined. Figures 12.1 through 12.6 illustrate the significant milestones that occur during the fabrication process of a CMOS inverter.



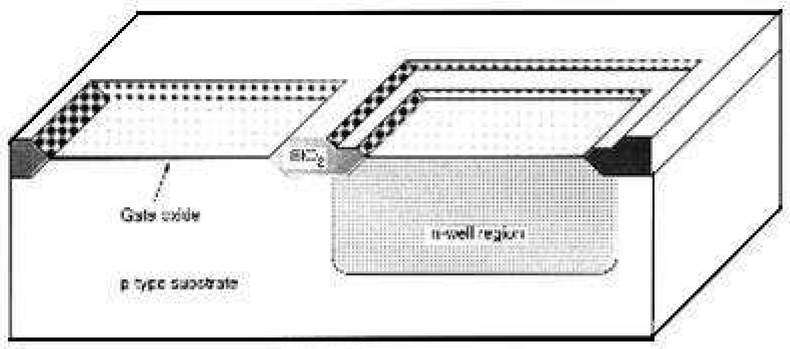


Figure-11.1: Cross  sectional view

Following the creation of the n-well region, a thick field oxide is grown in the areas surrounding the transistor active regions, and a thin gate oxide is grown on top of the active regions.

The thickness and the quality of the gate oxide are two of the most critical fabrication parameters, since they strongly affect the operational characteristics of the MOS transistor, as well as its long-term reliability.

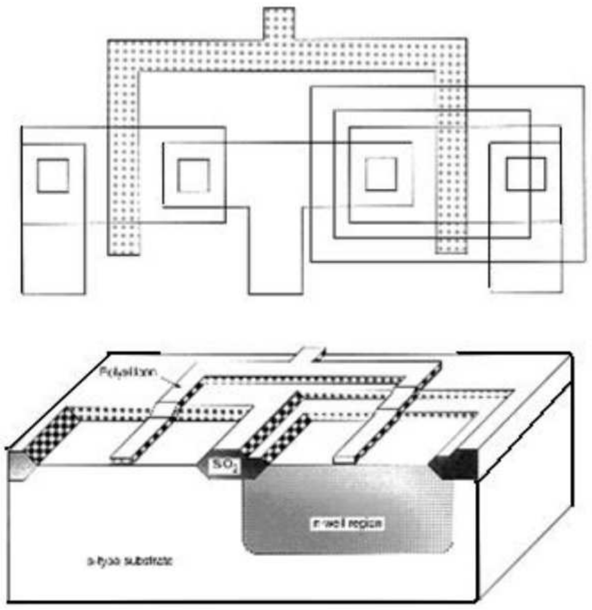


Figure-11.2: Cross sectional view

The polysilicon layer is deposited using chemical vapor deposition (CVD) and patterned by dry (plasma) etching. The created polysilicon lines will function as the gate electrodes of the nMOS and the pMOS transistors and their interconnects. Also, the polysilicon gates act as self-aligned masks for the source and drain implantations that follow this step.

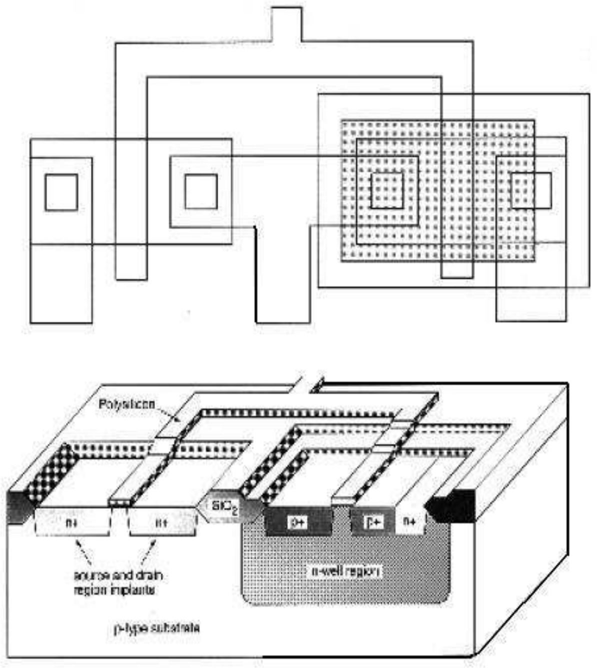


Figure-11.3: Using a set of two masks, the n+ and p+ regions are implanted into the substrate and into the n- well, respectively. Also, the ohmic contacts to the substrate and to the n-well are implanted in this process step.

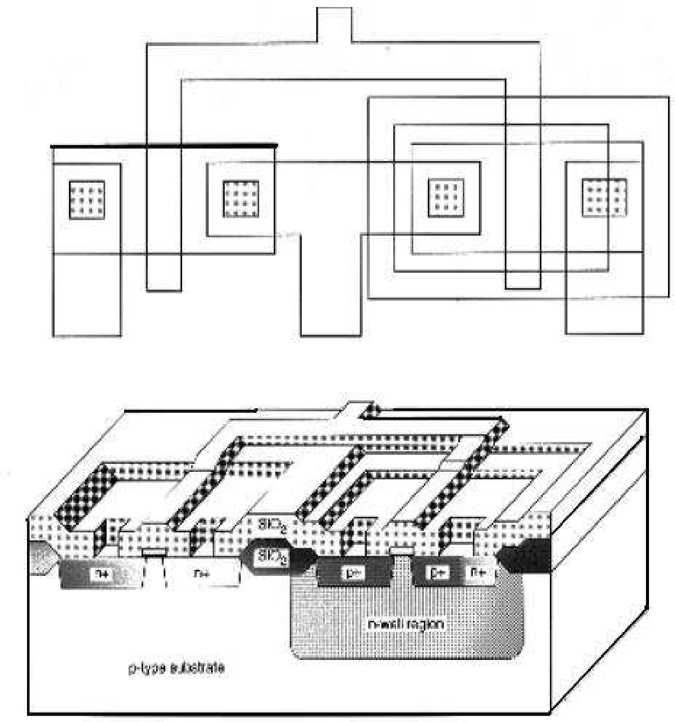
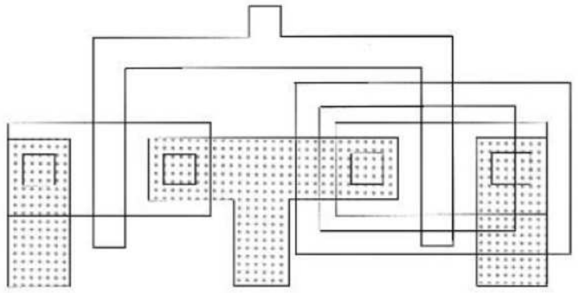


Figure-11.4: An insulating silicon dioxide layer is deposited over the entire wafer using CVD. Then, the contacts are defined and etched away to expose the silicon or polysilicon contact windows. These contact windows are necessary to complete the circuit interconnections using the metal layer, which is patterned in the next step.



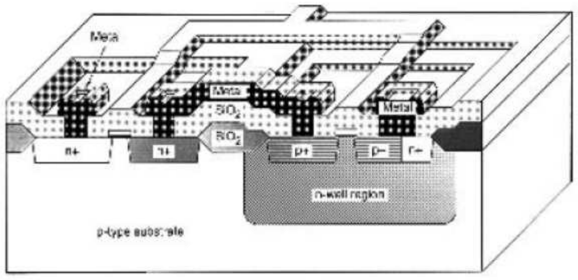
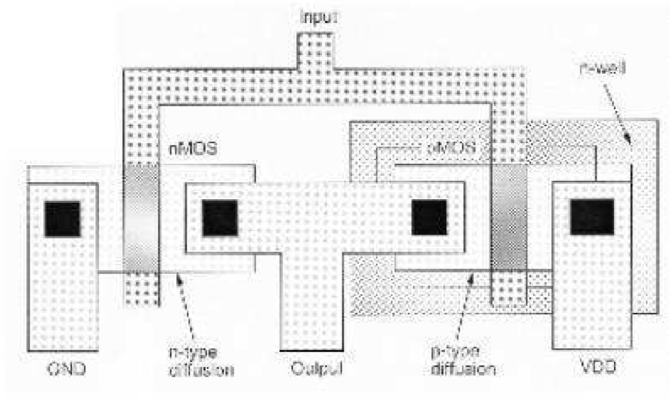


Figure-11.5: Metal (aluminum) is deposited over the entire chip surface using metal evaporation, and the metal lines are patterned through etching. Since the wafer surface is non-planar, the quality and the integrity of the metal lines created in this step are very critical and are ultimately essential for circuit reliability.



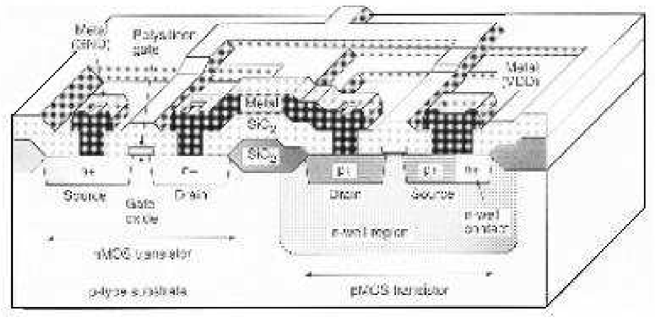


Figure-11.6: The composite layout and the resulting cross-sectional view of the chip, showing one nMOS and one pMOS transistor (built-in n-well), the polysilicon and metal interconnections. The final step is to deposit the passivation layer (for protection) over the chip, except for wire-bonding pad areas.

## Twin-tub process:

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

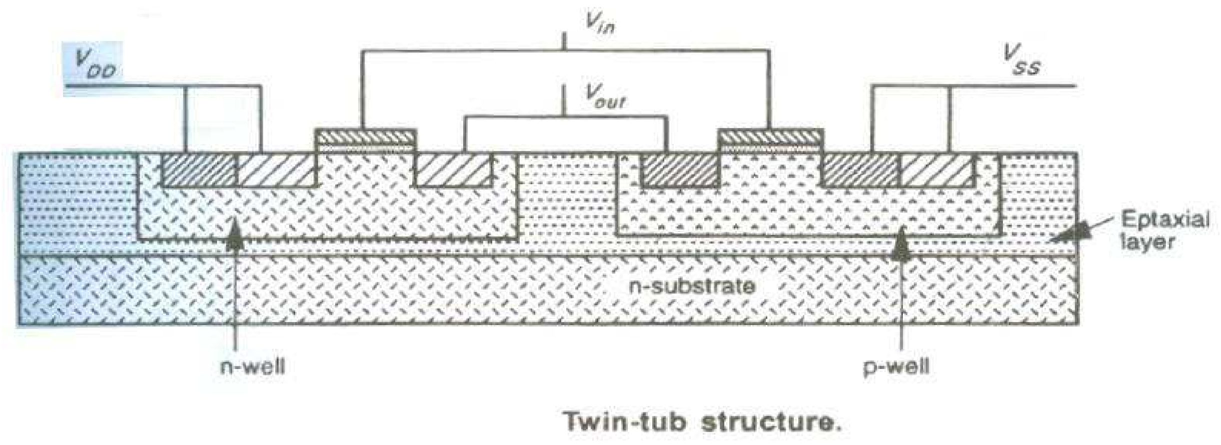


Figure 12 CMOS twin-tub inverter.

NOTE: Twin tub process is one of the solutions for latch-up problem.

# Circuit Design Process

## Introduction:

In this chapter we are going to study how to get the schematic into stick diagrams or layouts.

MOS circuits are formed on four basic layers:

> N-diffusion

> P-diffusion

> Polysilicon

> Metal

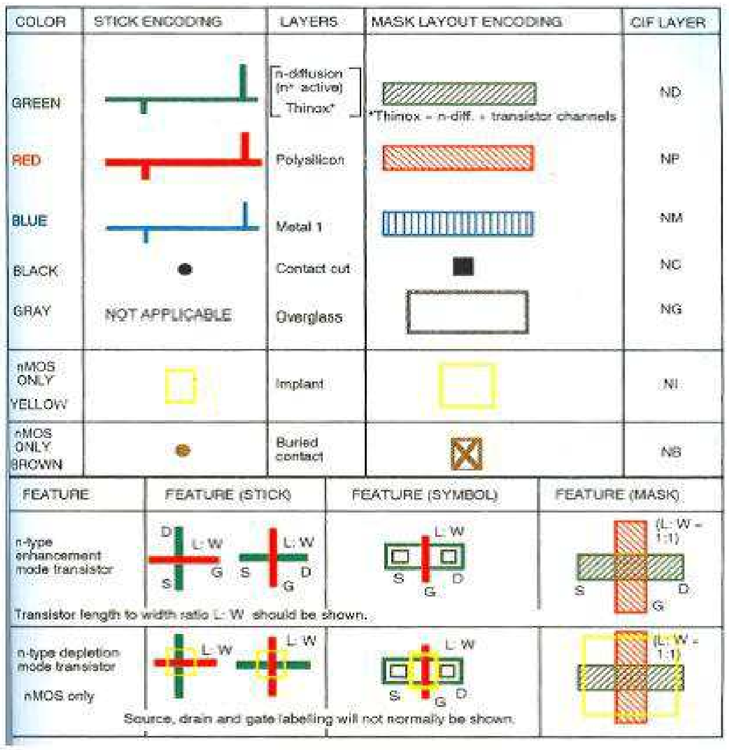
These layers are isolated by one another by thick or thin silicon dioxide insulating layers.

Thin oxide mask region includes n-diffusion / p-diffusion and transistor channel.

## 4. STICK DIAGRAMS:

Stick diagrams may be used to convey layer information through the use of a color code. For example: n-diffusion--green poly--red blue-- metal yellow--implant black--contact areas.

Encodings for NMOS process:



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

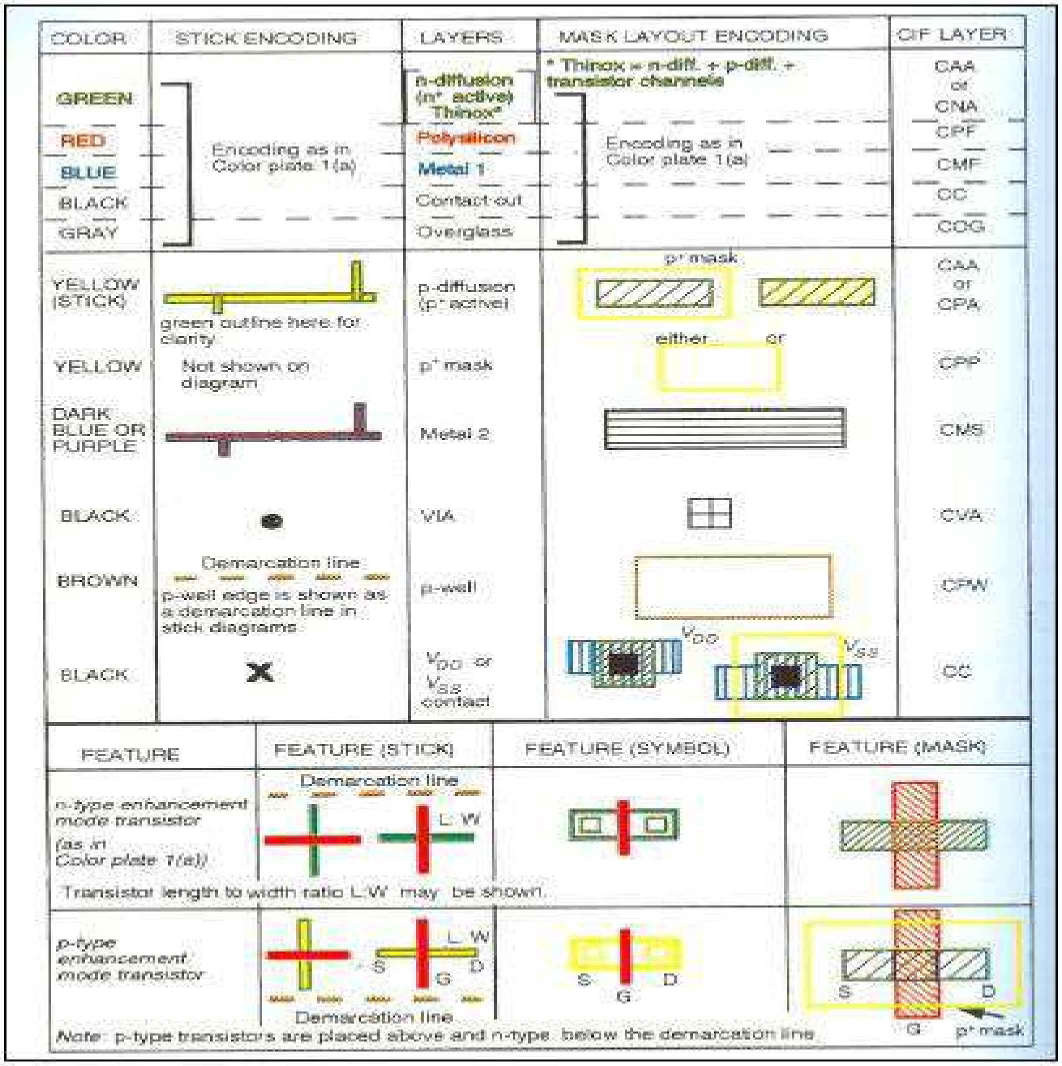
| FEATURE | FEATURE (STICK) | FEATURE (SYMBOL) | FEATURE (MASK) |
|---|---|---|---|
| n-type enhancement mode transistor | D L: W G S; L: W S D G | L: W S G D | (L: W = 1:1) S D G |
| Transistor length to width ratio L: W should be shown. | | | |
| n-type depletion mode transistor nMOS only | L: W; L: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

Figure 1: NMOS encodings.

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

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

**1 Encodings for CMOS process:**



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

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

Figure 2: CMOS encodings.

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

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

## 2 Encoding for BJT and MOSFETs:

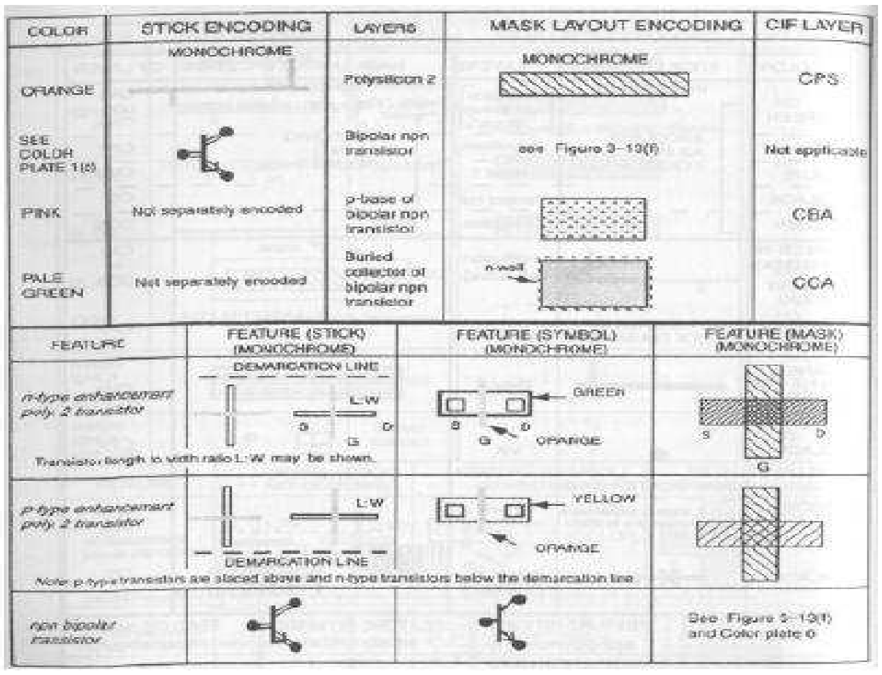

COLOR
STICK ENCODING
LAYERS
MASK LAYOUT ENCODING
CIF LAYER
MONOCHROME
MONOCHROME
ORANGE
Polysilicon 2
CPS
SEE COLOR PLATE 1(d)
Bipolar npn transistor
see Figure 3-13(f)
Not applicable
PINK
Not separately encoded
p-base of bipolar npn transistor
CBA
PALE GREEN
Not separately encoded
Buried collector of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

Figure 3: Bi CMOS encodings.

There are several layers in an nMOS chip:

_ a p-type substrate

_ paths of n-type diffusion

_ a thin layer of silicon dioxide

_ paths of polycrystalline silicon

_ a thick layer of silicon dioxide

_ paths of metal (usually aluminum)

_ a further thick layer of silicon dioxide

With contact cuts through the silicon dioxide where connections are required. The three layers carrying paths can be considered as independent conductors that only interact where polysilicon crosses diffusion to form a transistor. These tracks can be drawn as stick diagrams with _ diffusion in green _ polysilicon in red _ metal in blue using black to indicate contacts between layers and yellow to mark regions of implant in the channels of depletion mode transistors.

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

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

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

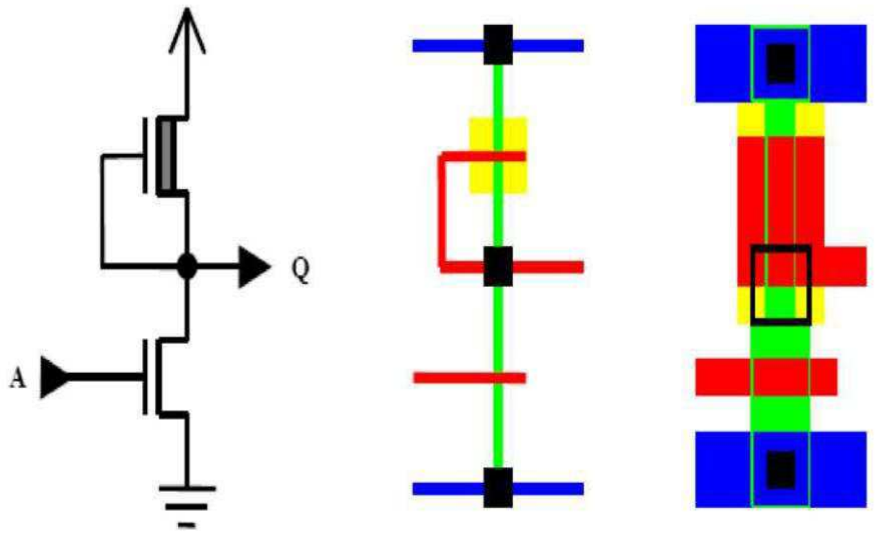


Figure 4: nMOS depletion load inverter.

Figure4 shows schematic, stick diagram and corresponding layout of nMOS depletion load inverter

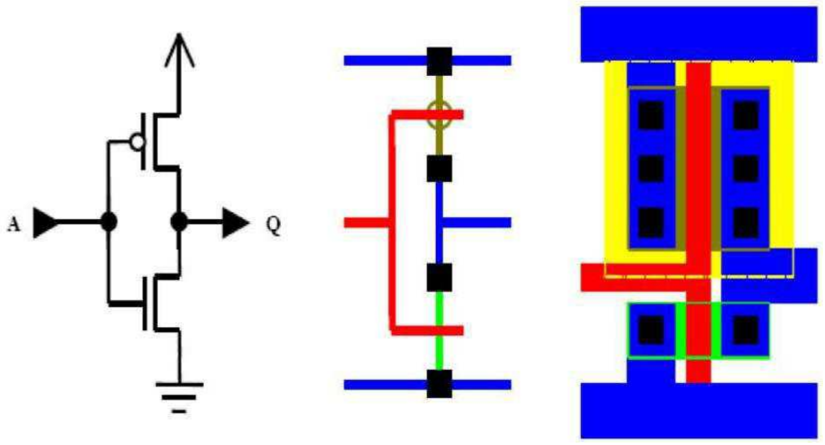


Figure 5: CMOS inverter

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

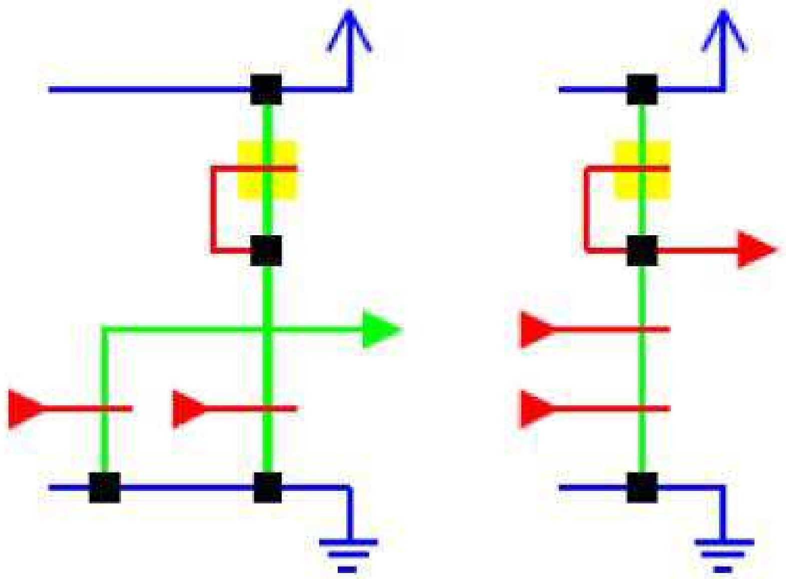


Figure 6 shows the stick diagrams for nMOS NOR and NAND.

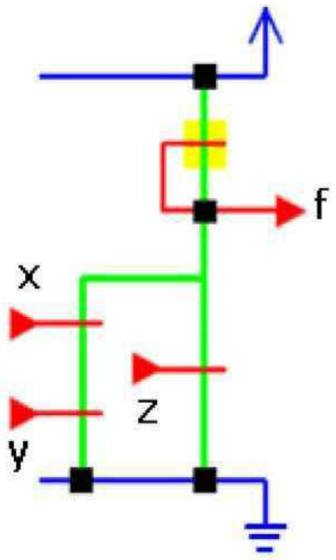


Figure 7: stick diagram of a given function f.

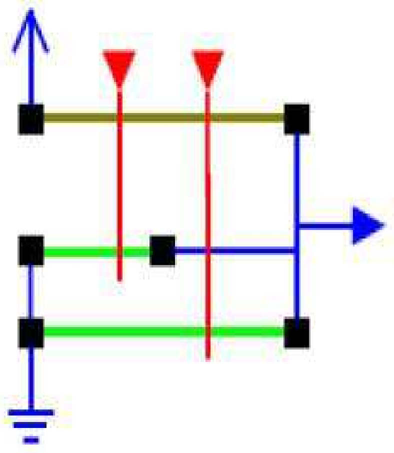


Figure 7

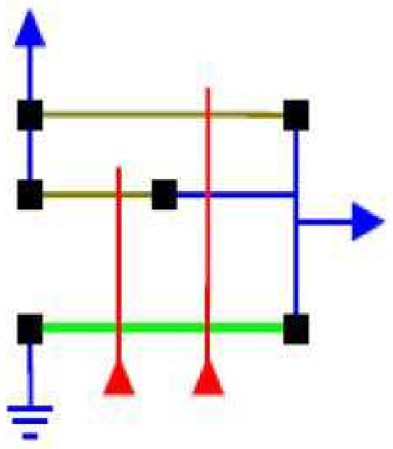


Figure 8

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

Figure 8 shows the stick diagram CMOS NOR and NAND, where we can see that the p diffusion line never touched the n diffusion directly, it is always joined using a blue color metal line.

## NMOS and CMOS Design style:

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

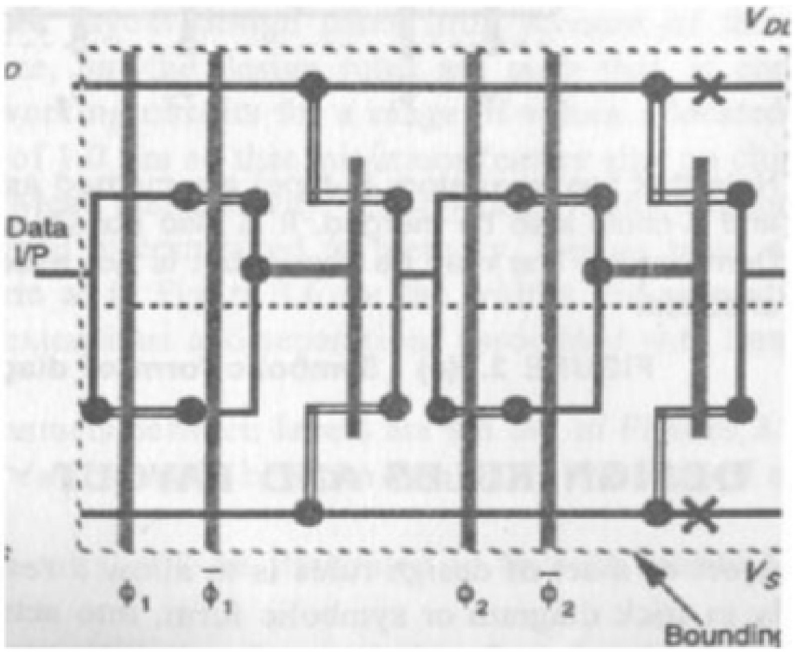


Figure 9: stick diagram of dynamic shift register in CMOS style.

Figure 9 shows the stick diagram of dynamic shift register using CMOS style. Here the output of the TG is connected as the input to the inverter and the same chain continues depending the number of bits.

## 3. DESIGN RULES:

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

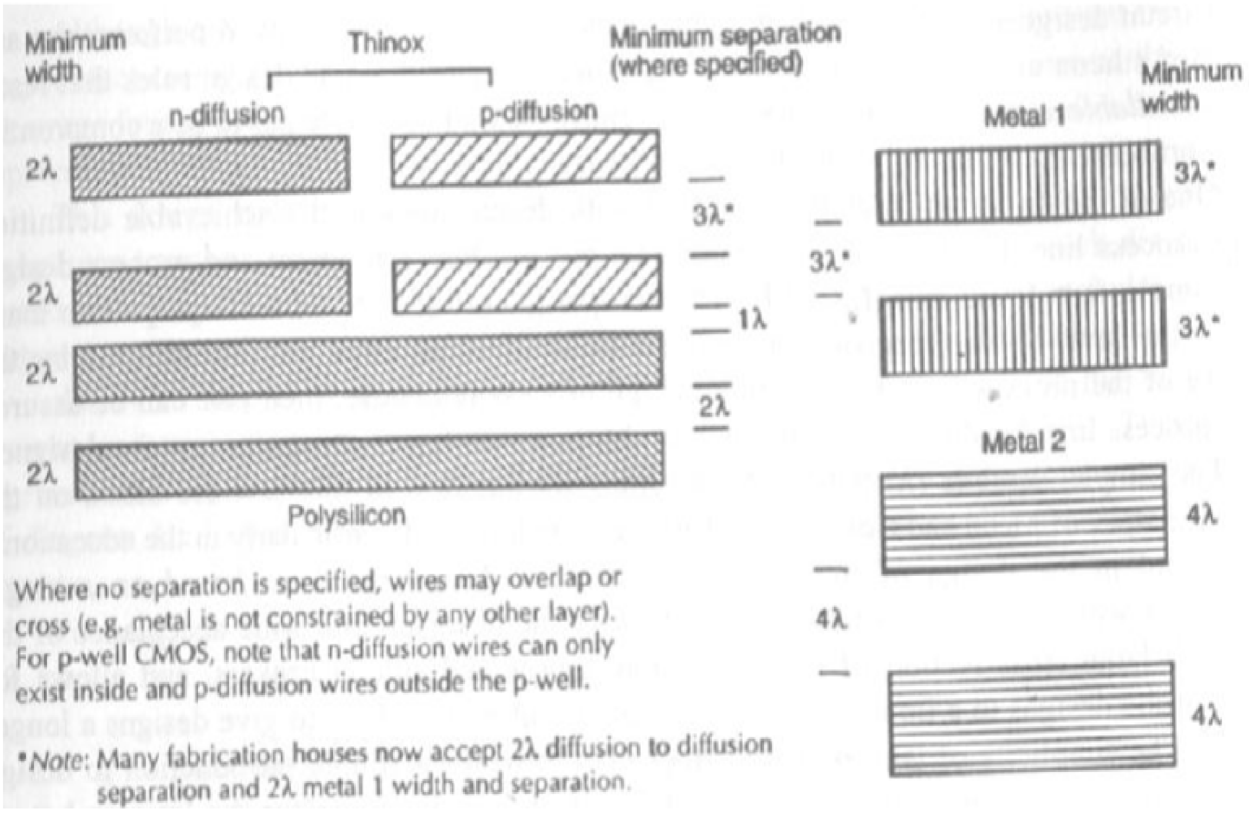
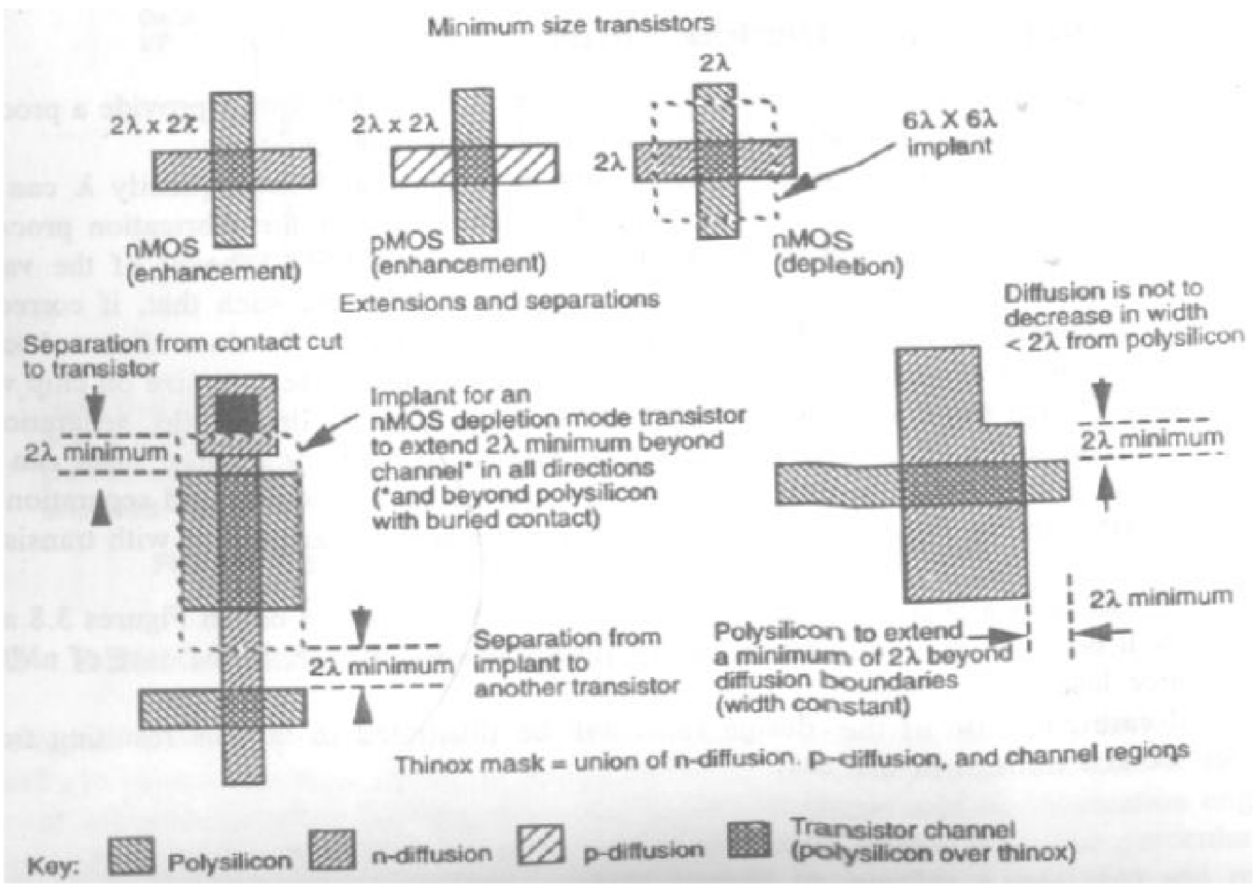

Minimum width
Thinox
Minimum separation (where specified)
Minimum width
n-diffusion
p-diffusion
Metal 1
2λ
3λ*
3λ*
3λ*
2λ
1λ
3λ*
2λ
2λ
2λ
Metal 2
Polysilicon
4λ
4λ
4λ
Where no separation is specified, wires may overlap or cross (e.g. metal is not constrained by any other layer). For p-well CMOS, note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

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

Figure 10 shows the design rule n diffusion, p diffusion, poly, metal1 and metal 2. The n and p diffusion lines is having a minimum width of 2λ and a minimum spacing of 3λ.Similarly we are showing for other layers.



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

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

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

What is Via?

It is used to connect higher level metals from metal connection. The cross section and layout view given figure 13 explain via in a better way.

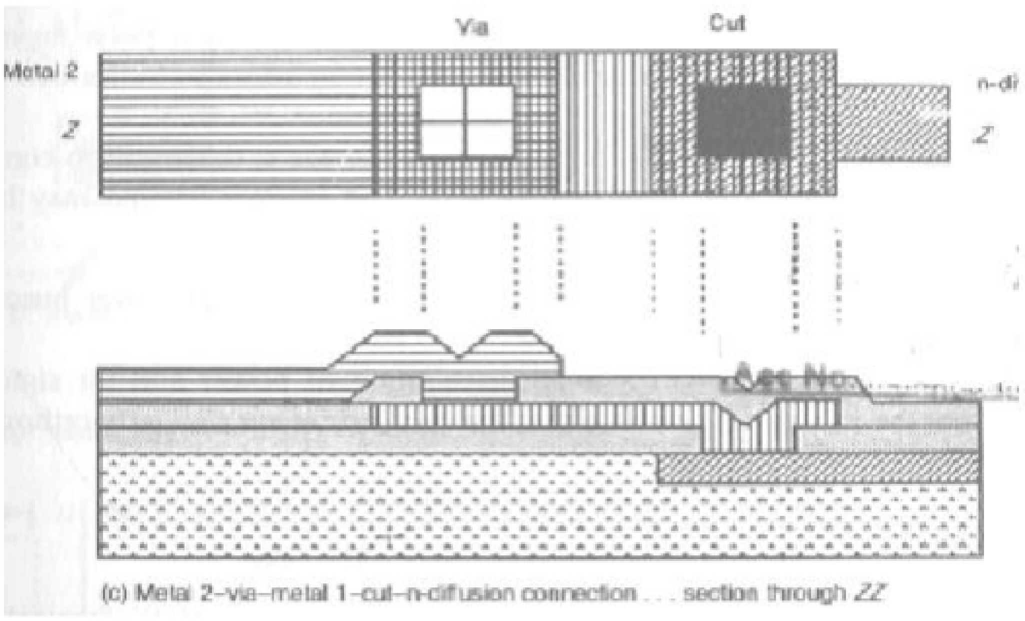

Via
Cut
Metal 2
Z
n-di
Z'
(c) Metal 2–via–metal 1–cut–n-diffusion connection . . . section through ZZ'

Figure 12: cross section showing the contact cut and via

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

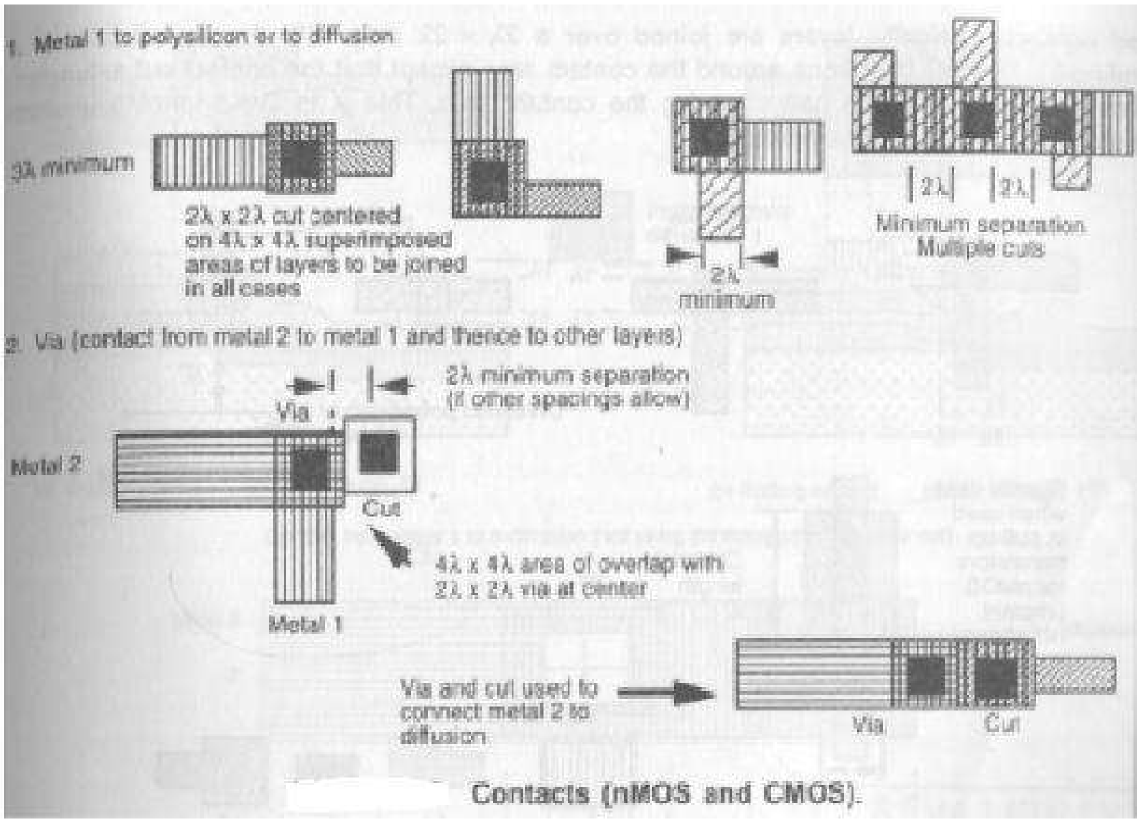

1. Metal 1 to polysilicon or to diffusion
3λ minimum
2λ x 2λ cut centered on 4λ x 4λ superimposed areas of layers to be joined in all cases
2λ minimum
2λ
2λ
Minimum separation Multiple cuts
2. Via (contact from metal 2 to metal 1 and thence to other layers)
Via
2λ minimum separation (if other spacings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

Figure 13: Design rules for contact cuts and vias

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

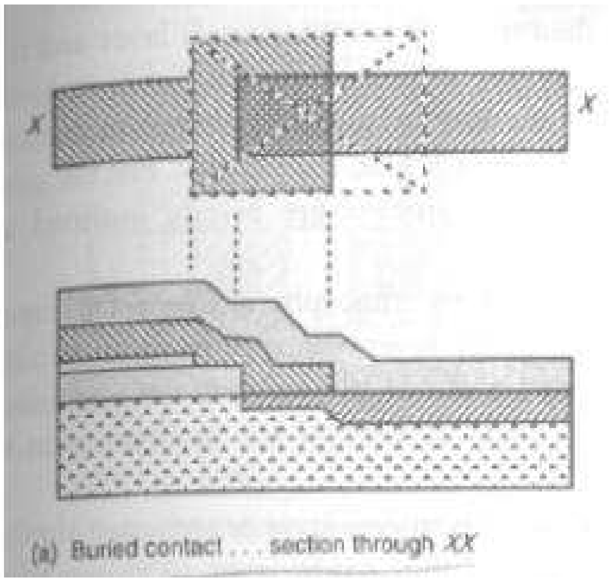


Figure 14: Buried contact.

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

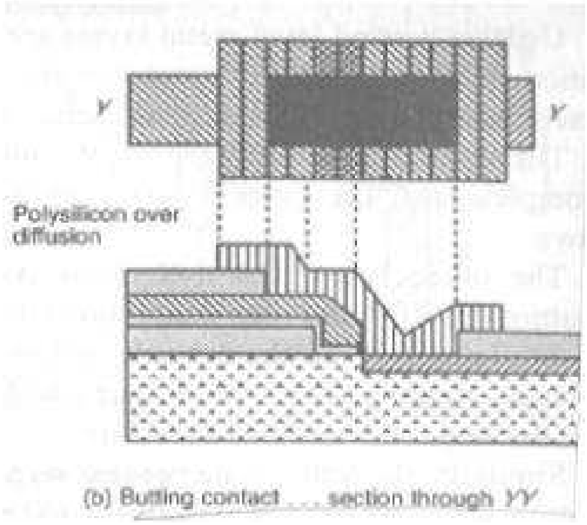

Y
Y
Polysilicon over diffusion
(b) Butting contact . . . section through YY

Figure 15: Butting contact.

## CMOS LAMBDA BASED DESIGN RULES:

Till now we have studied the design rules wrt only NMOS, what are the rules to be followed if we have the both p and n transistor on the same chip will be made clear with the diagram. Figure 16 shows the rules to be followed in CMOS well processes to accommodate both n and p transistors.

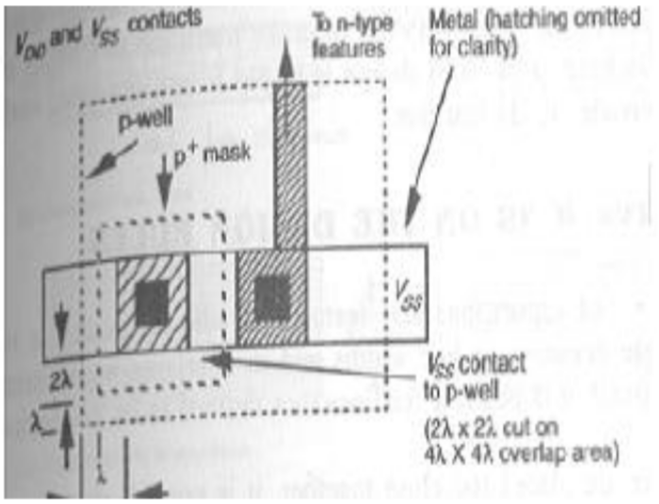

$V_{DD}$ and $V_{SS}$ contacts
To n-type features
Metal (hatching omitted for clarity)
p-well
$p^+$ mask
$V_{SS}$
$V_{SS}$ contact to p-well
(2λ x 2λ cut on 4λ X 4λ overlap area)
2λ
λ

Figure 16: CMOS design rules.

**<u>1 Orbit 2μm CMOS process:</u>**

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

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

Figure 17 shows the design rule for BiCMOS process using orbit 2um process.

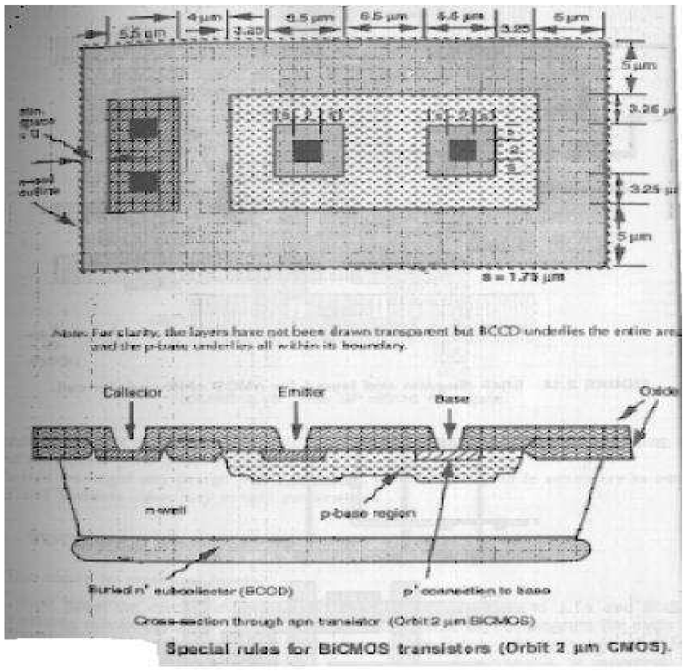

4 μm
5.5 μm
6.5 μm
5 μm
3.25
5 μm
3.25 μm
s = 1.75 μm
Note: For clarity, the layers have not been drawn transparent but BCCD underlies the entire area and the p-base underlies all within its boundary.
Collector
Emitter
Base
Oxide
n-well
p-base region
Buried n⁺ subcollector (BCCD)
p⁺ connection to base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

Figure 17: BiCMOS design rules.

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

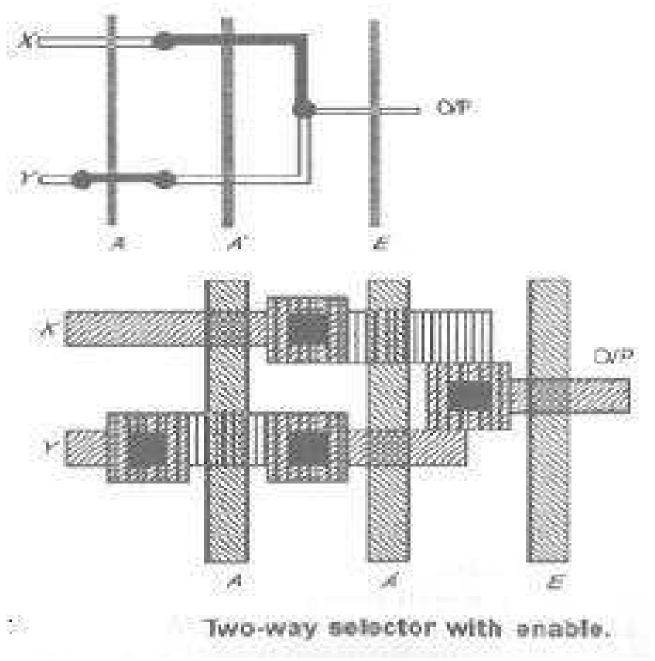


Figure 18: Two way selector stick and layout

## BASIC PHYSICAL DESIGN AN OVERVIEW

The VLSI design flow for any IC design is as follows

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

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

When the devices are represented using these layers, we call it physical design. The design is carried out using the design tool, which requires to follow certain rules. Physical structure is required to study the impact of moving from circuit to layout. When we draw the layout from the schematic, we are taking the first step towards the physical design. Physical design is an important step towards fabrication. Layout is representation of a schematic into layered diagram. This diagram reveals the different layers like ndiff, polysilicon etc that go into formation of the device. At every stage of the physical design simulations are carried out to verify whether the design is as per requirement. Soon after the layout design the DRC check is used to verify minimum dimensions and spacing of the layers. Once the layout is done, a layout versus schematic check carried out before proceeding further. There are different tools available for drawing the layout and simulating it.

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

1. Poly crossing diffusion makes a transistor

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

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

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

## SCHEMATIC AND LAYOUT OF BASIC GATES

### 1. CMOS INVERTER/NOT GATE SCHEMATIC

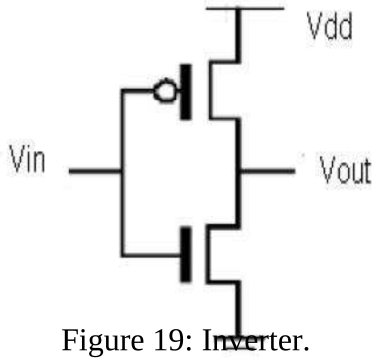


Figure 19: Inverter.

TOWARDS THE LAYOUT

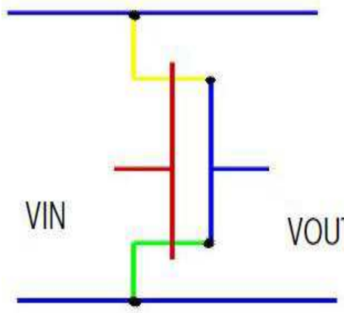


Figure 20: Stick diagram of inverter.

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

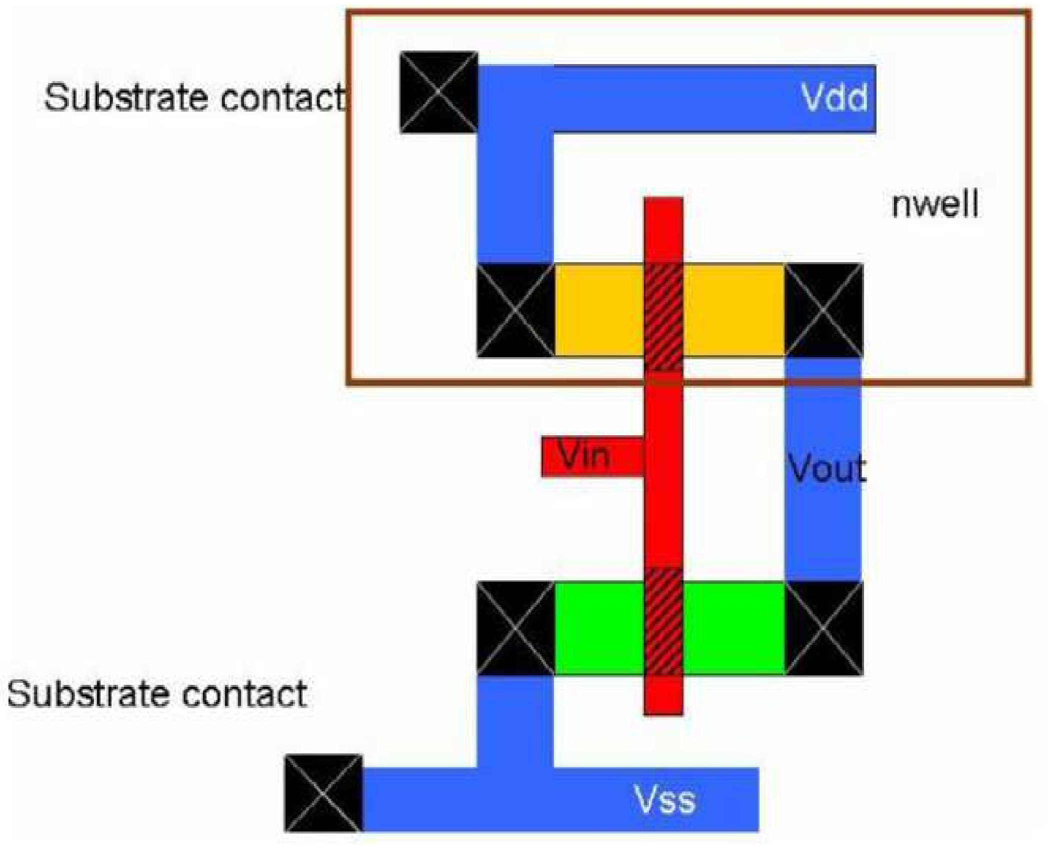


Figure 21: Layout of inverter.

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

## 2. CMOS NAND & NOR GATES SCHEMATIC

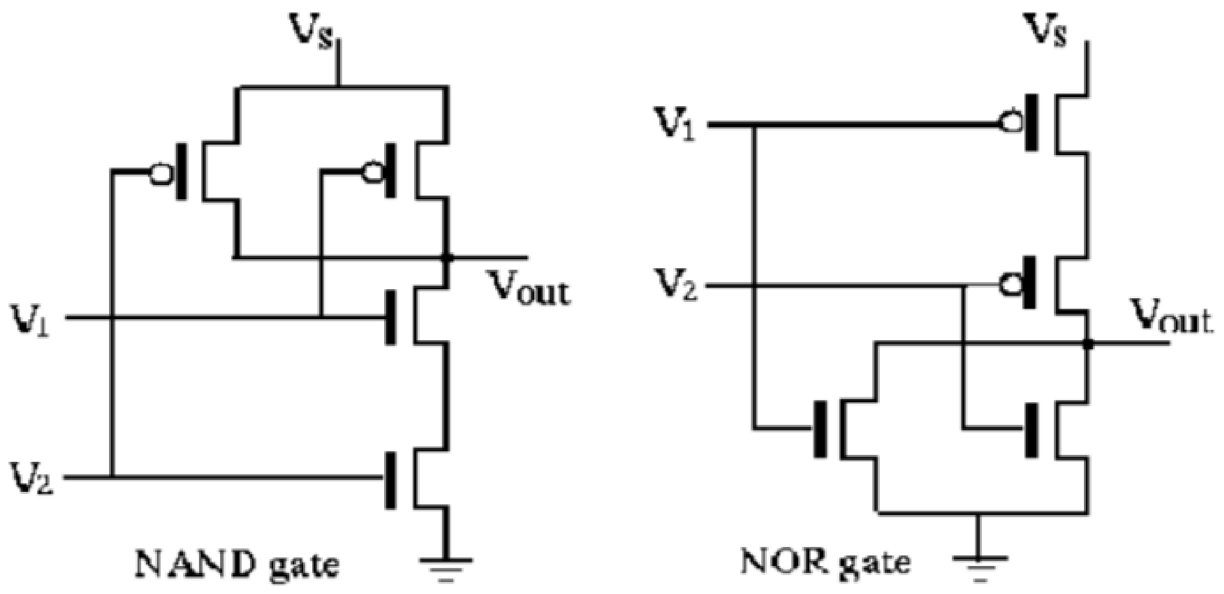


Figure 22: Schematic diagrams of nand and nor gate

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

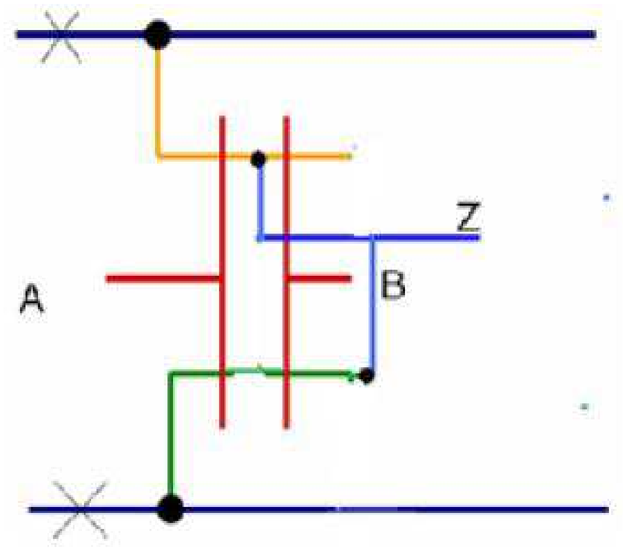


Figure 23: Stick diagrams of nand gate.

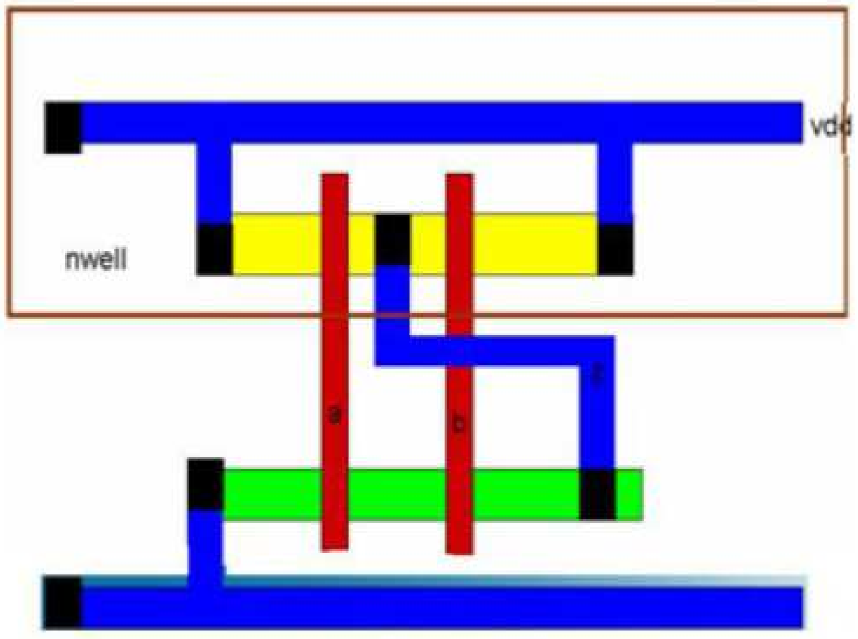


Figure 24: Layout of nand gate.

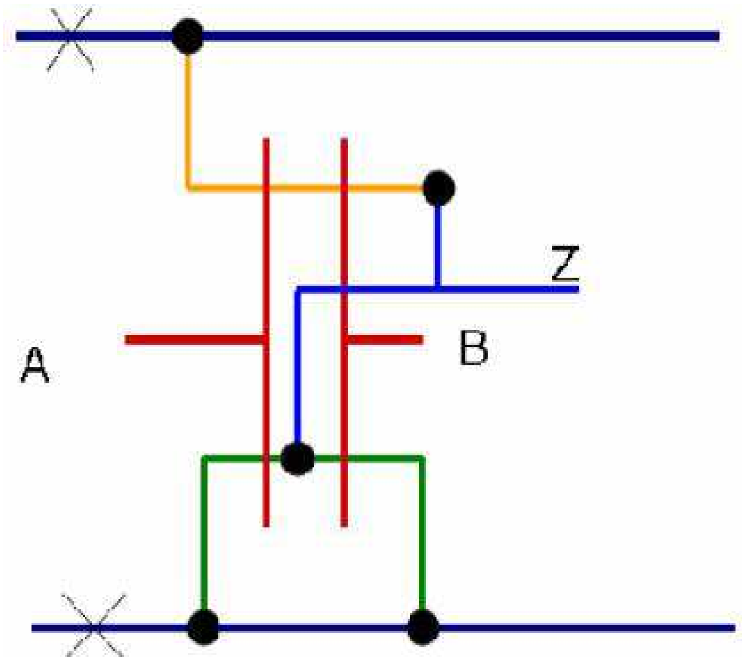


Figure 25: Stick diagram of nor gate.

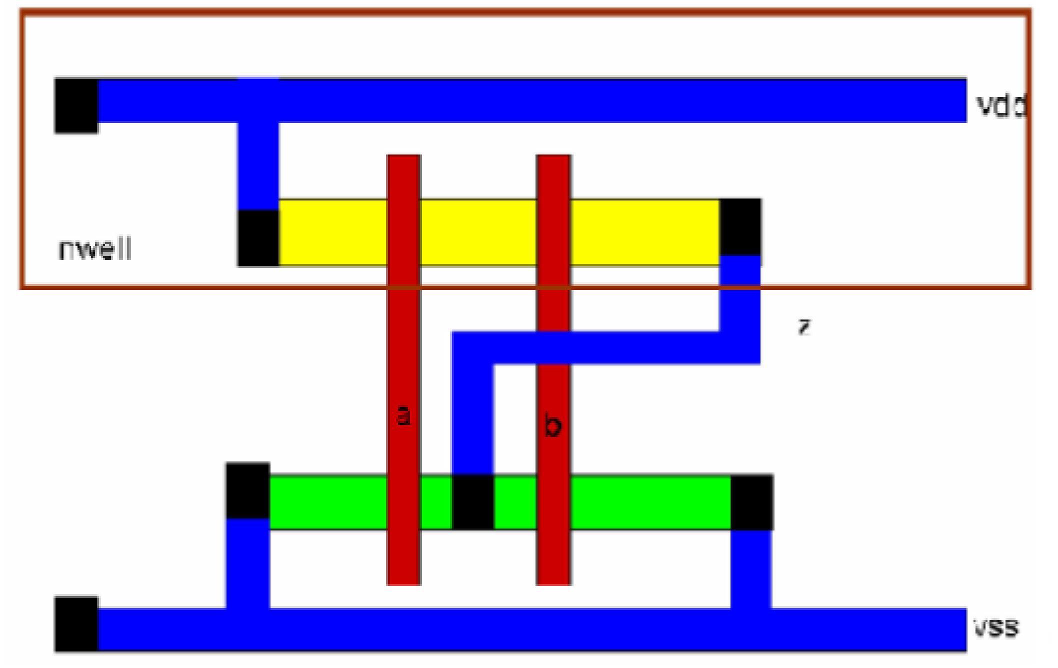


Figure 26: Layout of nor gate.

## GENERAL LAYOUT GUIDELINES

1. The electrical gate design must be completed by checking the following

   a. Right power and ground supplies

   b. Noise at the gate input

   c. Faulty connections and transistors

   d. Improper ratios

   c. Incorrect clocking and charge sharing

2. VDD and the VSS lines run at the top and the bottom of the design

3. Vertical poysilicon for each gate input

4. Order polysilicon gate signals for maximal connection between transistors

5. The connectivity requires to place nmos close to VSS and pmos close to VDD

6. Connection to complete the logic must be made using poly, metal and even metal2

The design must always proceeds towards optimization. Here optimization is at transistor level rather then gate level. Since the density of transistors is large, we could obtain smaller and faster layout by designing logic blocks of 1000 transistors instead of considering a single at a time and then putting them together. Density improvement can also be made by considering optimization of the other factors in the layout.

The factors are

l. Efficient routing space usage. They can be placed over the cells or even in multiple layers.

2. Source drain connections must be merged better.

3. White (blank) spaces must be minimum

4. The devices must be of optimum sizes.

5. Transperent routing can be provided for cell to cell interconnection, this reduces global wiring problems

## LAYOUT OPTIMIZATION FOR PERFORMANCE

l. Vary the size of the transistor according to its position in series. The transistor closest to the output is the smallest. The transistor nearest to the VSS line is the largest. This helps in increasing the performance by 30 %. A three input nand gate with the varying size is shown next.

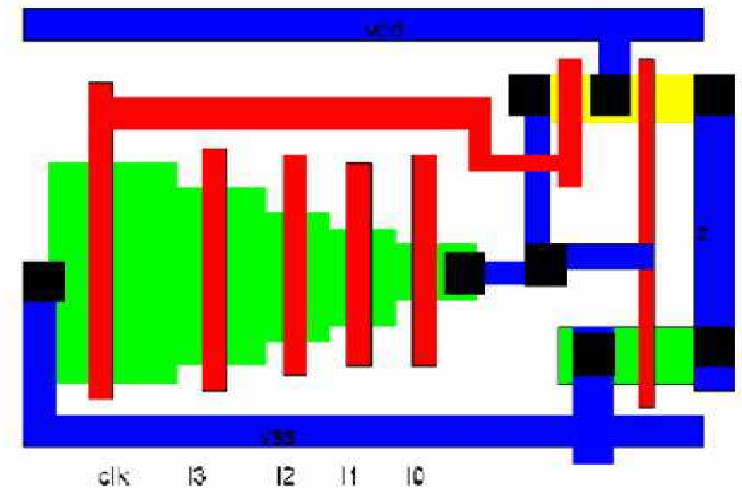


Figure 30: Layout optimization with varying diffusion areas.

2. Less optimized gates could occur even in the case of parallel connected transistors. This is usually seen in parallel inverters, nor & nand. When drains are connected in parallel, we must try and reduce the number of drains in parallel i.e. wherever possible we must try and connect drains in series at least at the output. This arrangement could reduce the capacitance at the output enabling good voltage levels. One example is as shown next.

3.

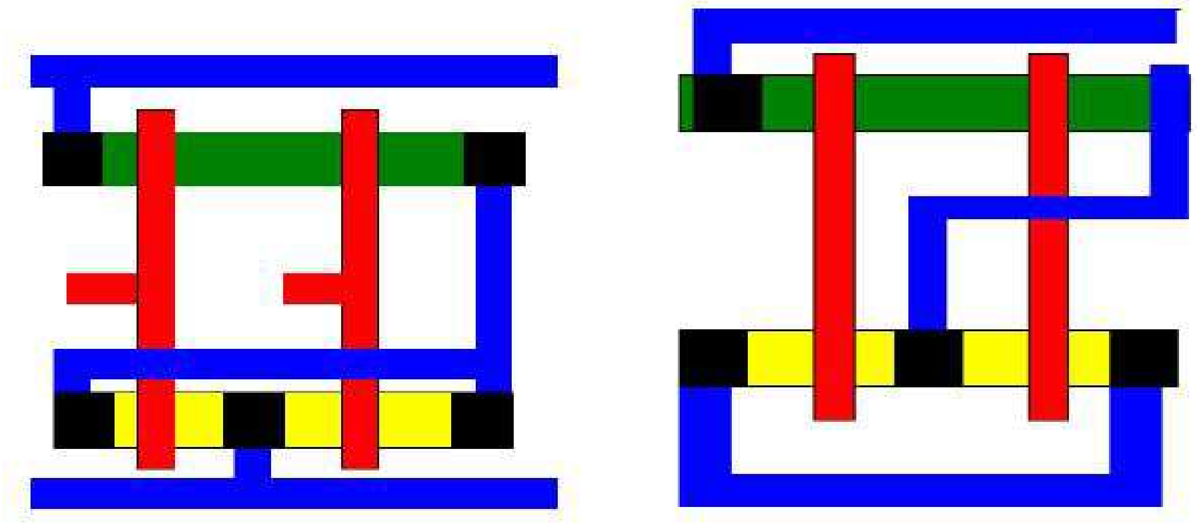
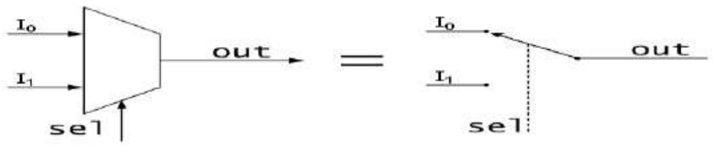


Figure 30: Layout of nor gate showing series and parallel drains.

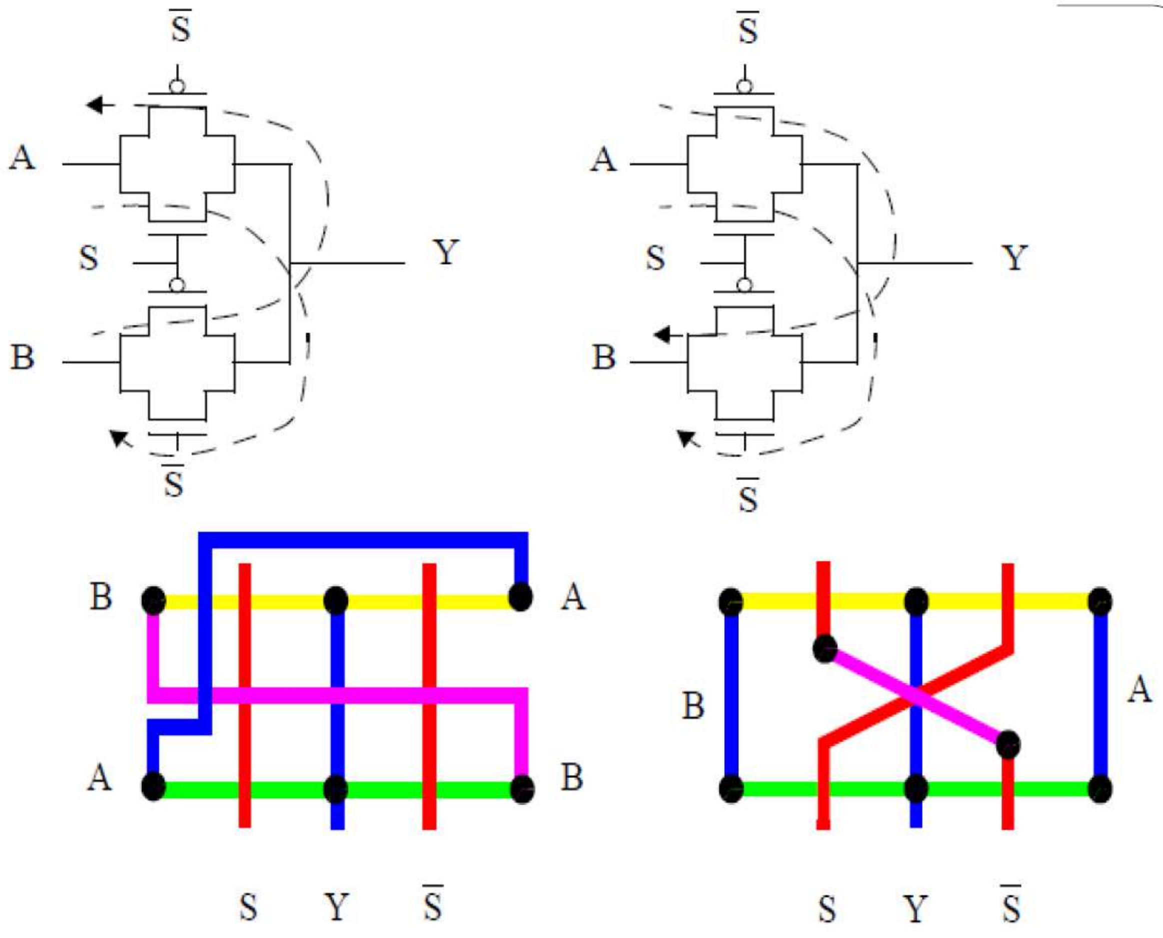
## MULTIPLEXER:

In electronics, a multiplexer or mux (occasionally the terms muldex or muldem are also found for a combination multiplexer-demultiplexer) is a device that performs multiplexing; it selects one of many analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2n inputs has n select lines, which are used to select which input line to send to the output.

An electronic multiplexer makes it possible for several signals to share one device or resource, for example one A/D converter or one communication line, instead of having one device per input signal.



An electronic multiplexer can be considered as a multiple-input, single-output switch, and a demultiplexer as a single-input, multiple-output switch. The schematic symbol for a multiplexer is an isosceles trapezoid with the longer parallel side containing the input pins and the short parallel side containing the output pin. The schematic on the right shows a 2-to-1 multiplexer on the left and an equivalent switch on the right. The sel wire connects the desired input to the output. The stick diagram for Multiplexer is given as follows for the 2 inputs A &B,


$\overline{S}$
A
S
B
Y
$\overline{S}$
$\overline{S}$
A
S
B
Y
$\overline{S}$
B
A
A
B
S
Y
$\overline{S}$
B
A
S
Y
$\overline{S}$

# 5. SCALING OF MOS DEVICES

The VLSI technology is in the process of evolution leading to reduction of the feature size and line widths. This process is called scaling down. The reduction in sizes has generally lead to better performance of the devices. There are certain limits on scaling and it becomes important to study the effect of scaling. The effect of scaling must be studied for certain parameters that effect the performance.

The parameters are as stated below

1.Minimum feature size

2.Number of gates on one chip

3.Power dissipation

4.Maximum operational frequency

5.Die size

6.Production cost .

These are also called as figures of merit

Many of the mentioned factors can be improved by shrinking the sizes of transistors, interconnects, separation between devices and also by adjusting the voltage and doping levels. Therefore it becomes essential for the designers to implement scaling and understand its effects on the performance

There are three types of scaling models used

1.Constant electric field scaling model

2.Constant voltage scaling model

3.Combined voltage and field model

The three models make use of two scaling factors 1/ß and 1/ά . 1/ß is chosen as the scaling factor for Vdd, gate oxide thickness D. 1/ ά is chosen as the scaling factor for all the linear dimensions like length, width etc. the figure next shows the dimensions and their scaling factors

The following are some simple derivations for scaling down the device parameters

**1.Gate area Ag**

Ag= L x W. Since L & W are scaled down by 1/ ά. Ag is scaled down by 1/ $ά^2$

**2.Gate capacitance per unit area**

Co=□o/D, permittivity of sio2 cannot be scaled, hence Co can be scaled 1/1/ß=ß

**3.Gate capacitance Cg**

Cg=CoxA=CoxLxW. Therefore Cg can be scaled by $ßx1/ ά x1/ ά= ß/ ά^2$

**4.Parasitic capacitance**

Cx=Ax/d, where Ax is the area of the depletion around the drain or source. d is the depletion width .Ax is scaled down by 1/ά2 and d is scaled by 1/ά. Hence Cx is scaled by

$1/ ά^2 /1/ ά = 1/ ά$

**5.Carrier density in the channel Qon**

Qon=Co.Vgs

Co is scaled by ß and Vgs is scaled by 1/ ß,hence Qo is scaled by ßx1/ß =1.

**Channel resistance Ro**

Ron = L/Wx1/Qoxμ, μ is mobility of charge carriers . Ro is scaled by1/ά/1/ άx1=1

**Gate delay Td**

Td is proportional to Ro and Cg

Td is scaled by $1x\ ß/ά^2 = ß/ά^2$

**Maximum operating frequency fo**

fo=1/td,therefore it is scaled by $1/ ß/ά^2 = ά^2/ß$

**Saturation current**

Idss= CoμW(Vgs-Vt)/2L, Co scale by ß and

voltages by 1/ ß, Idss is scaled by $ß /ß^2= 1/ß$

**Current Density**

J=Idss/A hence J is scaled by $1/ß/1/ά2 = ά^2 /ß$

## 1.What is Scaling?

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

## 2.Why Scaling?...

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

## 3.FoM for Scaling

Impact of scaling is characterized in terms of several indicators:

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

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

### 3.1 Technology Scaling

Goals of scaling the dimensions by 30%:

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

Double transistor density

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

Die size used to increase by 14% per generation

Technology generation spans 2-3 years

Figure1 to Figure 5 illustrates the technology scaling in terms of minimum feature size, transistor count, prapogation delay, power dissipation and density and technology generations.

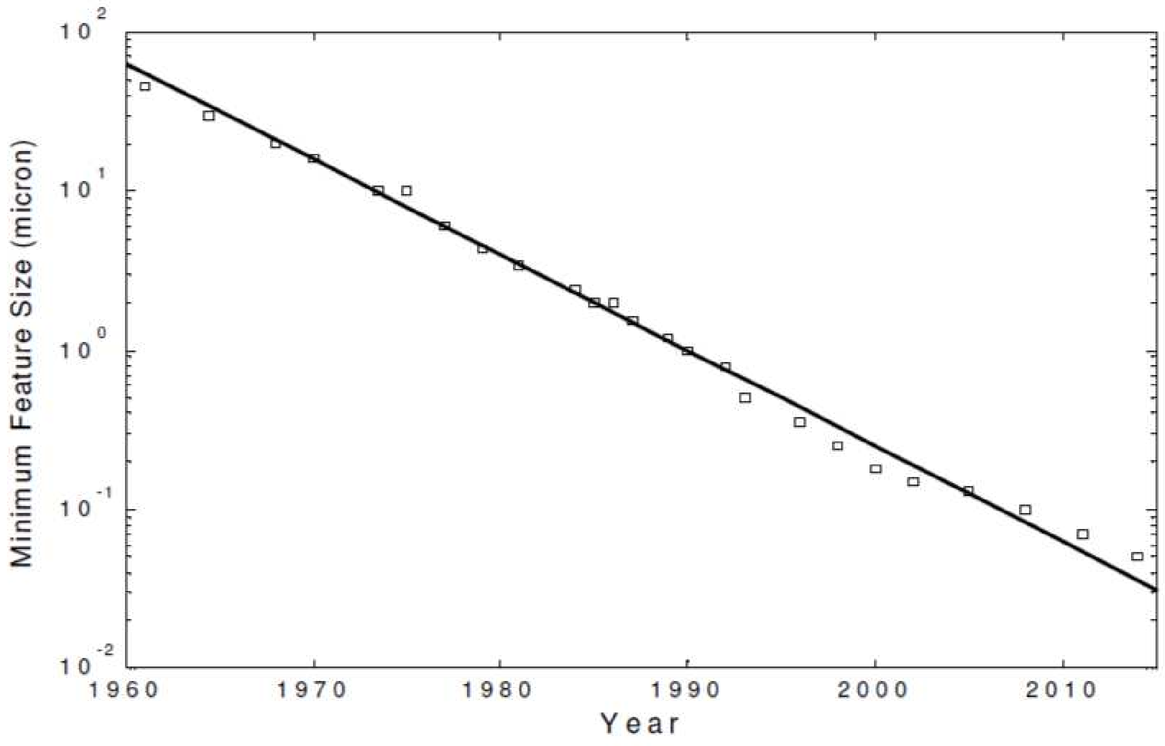


Figure-1:Technology Scaling (1)

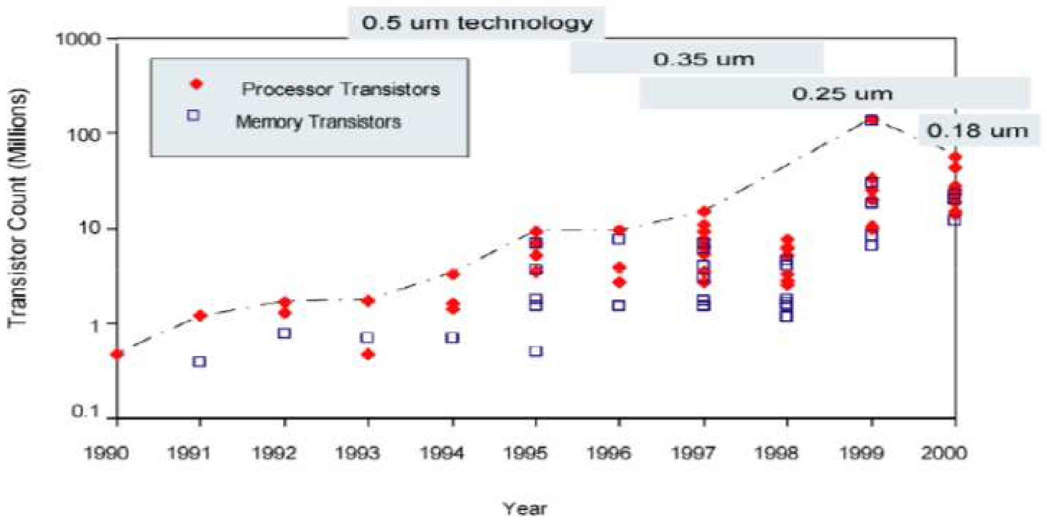
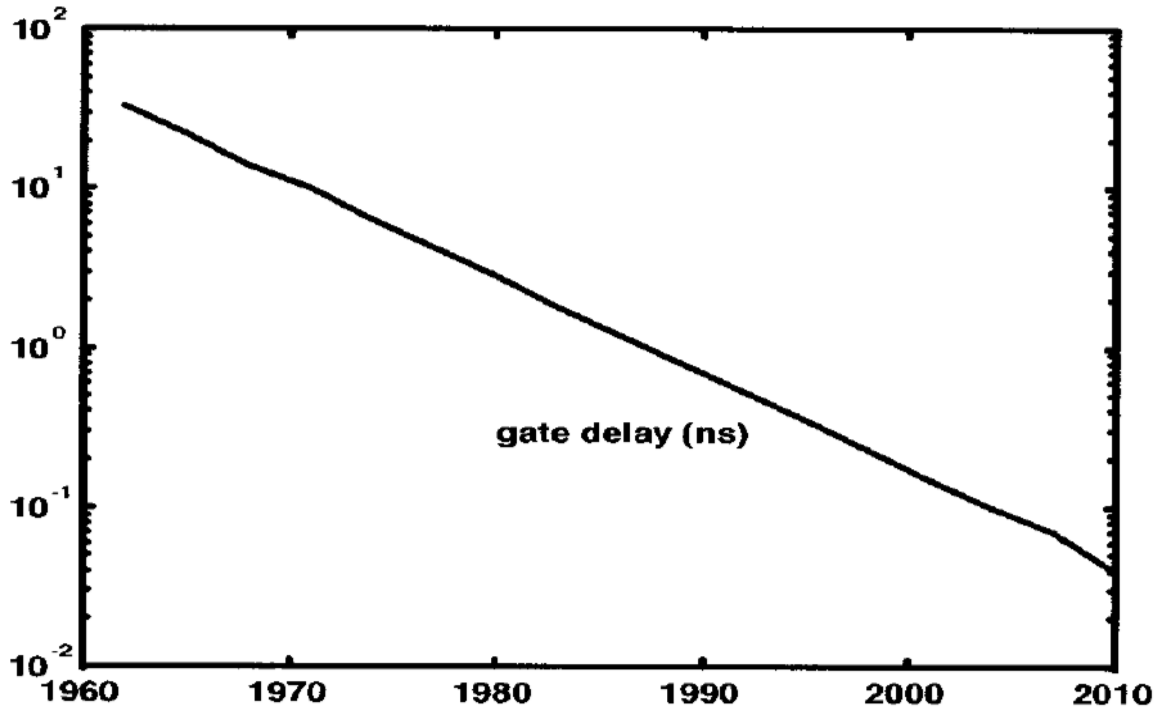


Figure-2:Technology Scaling (2)



**Propagation Delay**

Figure-3:Technology Scaling (3)

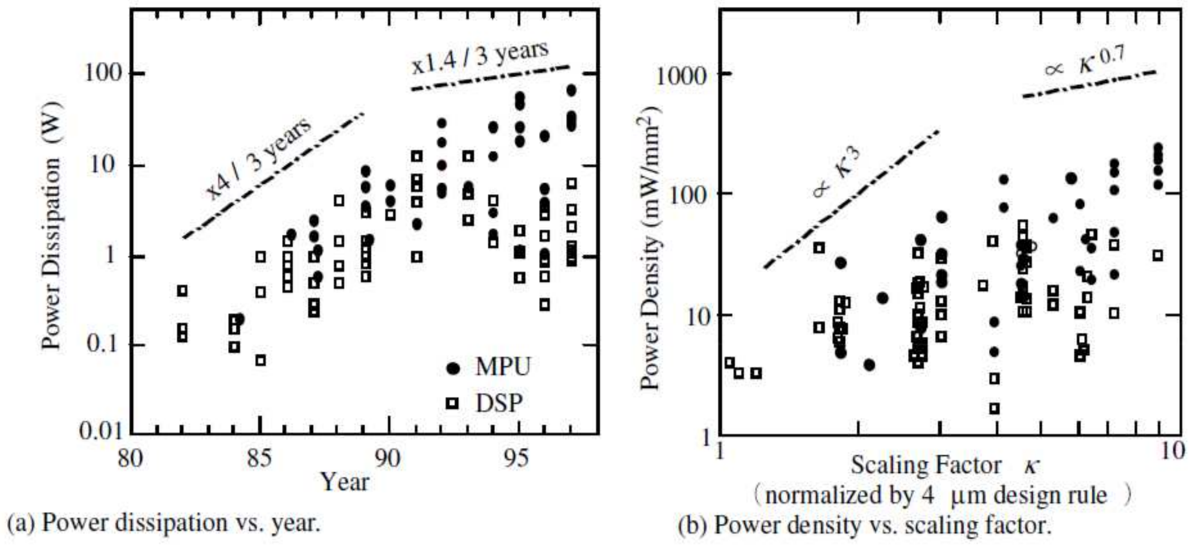


Figure-4:Technology Scaling (4)

**Technology Generations**

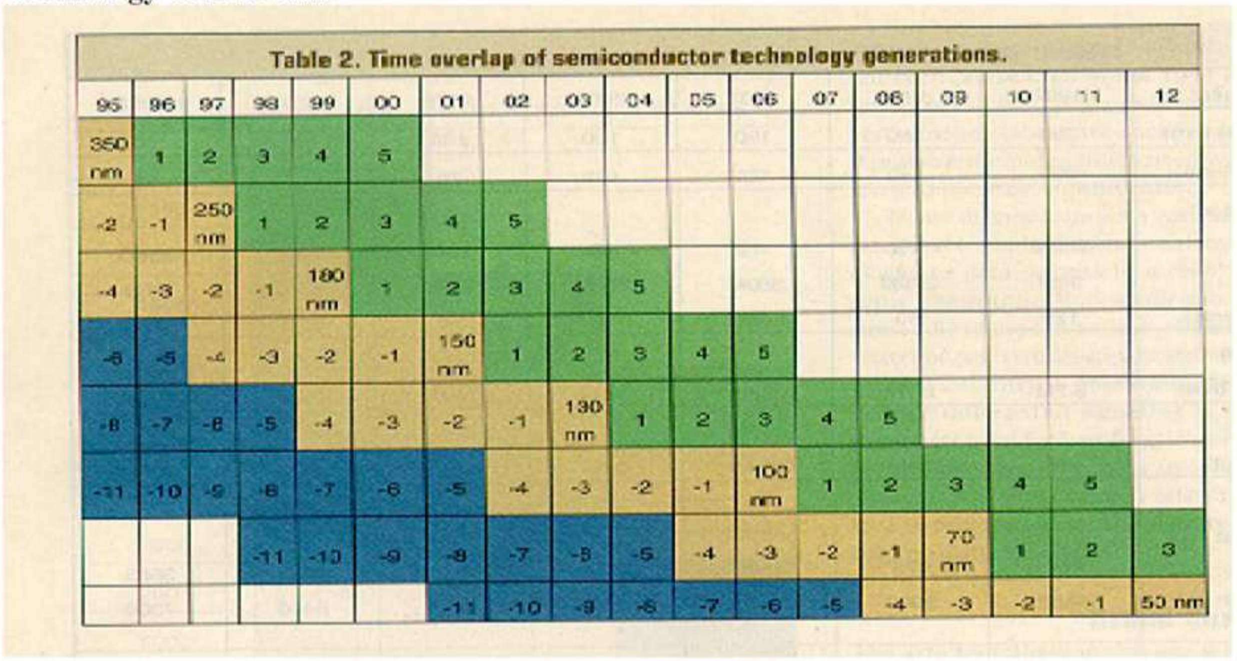


Table 2. Time overlap of semiconductor technology generations.

| 95 | 96 | 97 | 98 | 99 | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 350 nm | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |  |  |  |  |  |
| -2 | -1 | 250 nm | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |  |  |  |
| -4 | -3 | -2 | -1 | 180 nm | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |  |
| -6 | -5 | -4 | -3 | -2 | -1 | 150 nm | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |
| -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 130 nm | 1 | 2 | 3 | 4 | 5 |  |  |  |  |
| -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 100 nm | 1 | 2 | 3 | 4 | 5 |  |
|  |  |  | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 70 nm | 1 | 2 | 3 |
|  |  |  |  |  |  | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 50 nm |

Figure-5:Technology generation

## 4. International Technology Roadmap for Semiconductors (ITRS)

Table 1 lists the parameters for various technologies as per ITRS.

| Year of Introduction | 1999 | 2000 | 2001 | 2004 | 2008 | 2011 | 2014 |
|---|---|---|---|---|---|---|---|
| Technology node [nm] | 180 |  | 130 | 90 | 60 | 40 | 30 |
| Supply [V] | 1.5-1.8 | 1.5-1.8 | 1.2-1.5 | 0.9-1.2 | 0.6-0.9 | 0.5-0.6 | 0.3-0.6 |
| Wiring levels | 6-7 | 6-7 | 7 | 8 | 9 | 9-10 | 10 |
| Max frequency [GHz], Local-Global | 1.2 | 1.6-1.4 | 2.1-1.6 | 3.5-2 | 7.1-2.5 | 11-3 | 14.9 -3.6 |
| Max µP power [W] | 90 | 106 | 130 | 160 | 171 | 177 | 186 |
| Bat. power [W] | 1.4 | 1.7 | 2.0 | 2.4 | 2.1 | 2.3 | 2.5 |

Node years: 2007/65nm, 2010/45nm, 2013/33nm, 2016/23nm

Table 1: ITRS

**CONCLUSION**:

CMOS process options and directions can greatly influence design decisions. Frequently, the combination of performance and cost possibilities in a new process can provide new product opportunities that were not available previously. Similarly, venerable processes can offer good opportunities with the right product. One issue that has to be kept in mind is the ever-increasing cost of having a CMOS design fabricated in a leading-edge process. This in turn is reflected in the types of design and approaches to design that are employed for CMOS chips of the future. For instance, making a design programmable so that it can have a longer product life is a good first start.

# Notes on
# MOS CIRCUIT PERFORMANCE AND CMOS LOGIC CIRCUITS
# in VLSI Design – Unit III

**Dr. G. Senthil Kumar,**
Associate Professor,
Dept. of ECE, SCSVMV,
email: gsk_ece@kanchiuniv.ac.in

==============================================================

**OBJECTIVES**:

In this lesson, you will be understand the design and realize the circuits in CMOS logic circuits.

**CONTENTS**:

## 1. SHEET RESISTANCE:

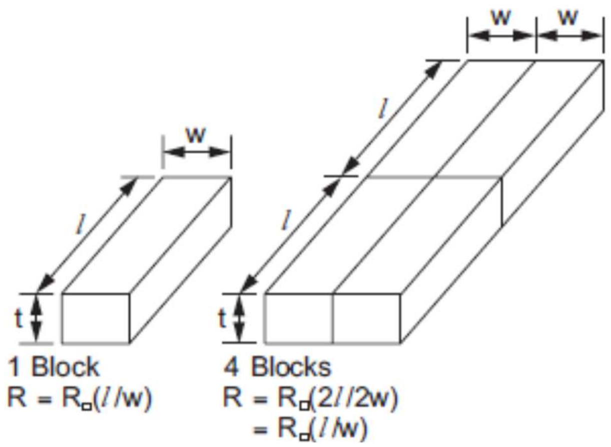
The resistance of a unifor  m slab of conducting material can be expressed as,

$$R = \frac{\rho}{t}\frac{l}{w}$$

where ρ is the resistivity. This expression can be rewritten as

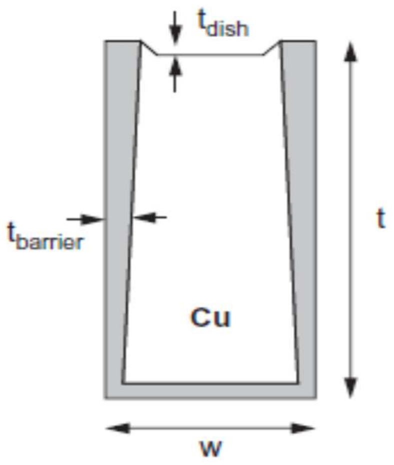
$$R = R_{\square}\frac{l}{w}$$

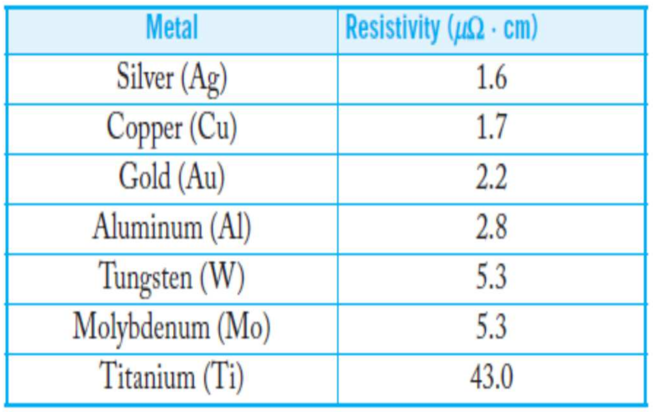
where R = ρ /t is the sheet resistance and has units of ohms/square. Note that a square is a dimensionless quantity corresponding to a slab of equal length and width. This is convenient because resistivity and thickness are characteristics of the process outside the control of the circuit designer and can be abstracted away into the single sheet resistance parameter.



To obtain the resistance of a conductor on a layer, multiply the sheet resistance by the ratio of length to width of the conductor. For example, the resistances of the two shapes in Figure are equal because the length-to-width ratio is the same even though the sizes are different. Nonrectangular shapes can be decomposed into simpler regions for which the resistance is calculated.

Table 6.2 shows bulk electrical resistivities of pure metals at room temperature. The resistivity of thin metal films used in wires tends to be higher because of scattering off the surfaces and grain boundaries,





| Metal | Resistivity ($\mu\Omega \cdot$ cm) |
|---|---|
| Silver (Ag) | 1.6 |
| Copper (Cu) | 1.7 |
| Gold (Au) | 2.2 |
| Aluminum (Al) | 2.8 |
| Tungsten (W) | 5.3 |
| Molybdenum (Mo) | 5.3 |
| Titanium (Ti) | 43.0 |

As shown in Figure, copper must be surrounded by a lower-conductivity diffusion barrier that effectively reduces the wire cross-sectional area and hence raises the resistance. Moreover, the polishing step can cause dishing that thins the metal. Even a 10 nm barrier is quite significant when the wire width is only tens of nanometers. If the average barrier thickness is t barrier and the height is reduced by tdish, the resistance becomes,

$$R = \frac{\rho}{\left(t - t_{\text{dish}} - t_{\text{barrier}}\right)} \frac{l}{\left(w - 2t_{\text{barrier}}\right)}$$

The resistivity of polysilicon, diffusion, and wells is significantly influenced by the doping levels. Polysilicon and diffusion typically have sheet resistances under 10 ohms/square when silicided and up to several hundred ohms/square when unsilicided. Wells have lower doping and thus even higher sheet resistance. These numbers are highly process dependent. Large resistors are often made from wells or unsilicided polysilicon.

Contacts and vias also have a resistance, which is dependent on the contacted materials and size of the contact. Typical values are 2–20 ohms. Multiple contacts should be used to form low-resistance connections, as shown in Figure. When current turns at a right angle or reverses, a square array of contacts is generally required, while fewer contacts can be used when the flow is in the same direction

## *CAPACITANCES ESTIMATION*

The first component of capacitive parasitics we will examine is the MOSFET capacitances. These parasitic components are mainly responsible for the intrinsic delay of logic gates, and they can be modeled with fairly high accuracy for gate delay estimation. The extraction of transistor parasitics from physical structure (mask layout) is also fairly straight forward.

The parasitic capacitances associated with a MOSFET are shown in Fig.7 as lumped elements between the device terminals. Based on their physical origins, the parasitic device capacitances can be classified into two major groups: (1) oxide-related capacitances and (2) junction capacitances. The gate-oxide-related capacitances are Cgd (gate-to-drain capacitance), Cgs (gate-to-source capacitance), and Cgb (gate-to-substrate capacitance). Notice that in reality, the gate-to-channel capacitance is distributed and voltage dependent. Consequently, all of the oxide-related capacitances described here change with the bias conditions of the transistor. Figure 8 shows qualitatively the oxide-related capacitances during cut-off, linear-mode operation and saturation of the MOSFET. The simplified variation of the three capacitances with gate-to-source bias voltage is shown in Fig. 9.

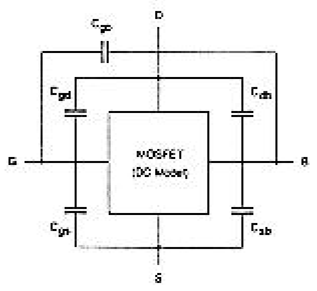


**Figure7:** Lumped representation of parasitic MOSFET capacitances.

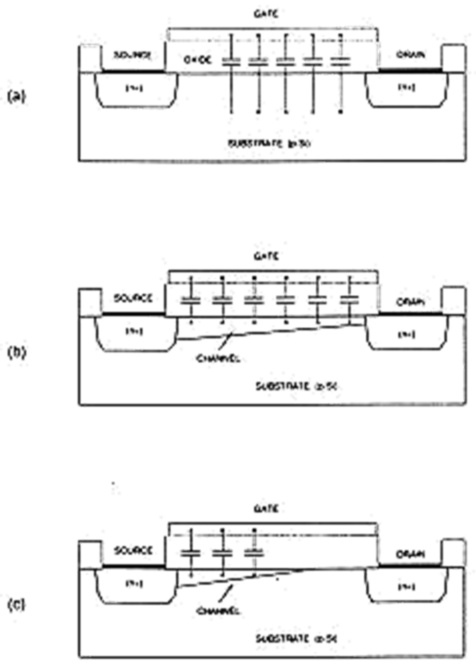


**Figure:** Schematic representation of MOSFET oxide capacitances during (a) cut-off, (b) linear- mode operation, and (c) saturation.

Note that the total gate oxide capacitance is mainly determined by the parallel-plate capacitance between the polysilicon gate and the underlying structures. Hence, the magnitude of the oxide-related capacitances is very closely related to (1) the gate oxide thickness, and (2) the area of the MOSFET gate. Obviously, the total gate capacitance decreases with decreasing device dimensions (W and L), yet it increases with decreasing gate oxide thickness. In sub-micron technologies, the horizontal dimensions (which dictate the gate area) are usually scaled down more easily than the horizontal dimensions, such as the gate oxide thickness. Consequently, MOSFET transistors fabricated using sub-micron technologies have, in general, smaller gate capacitances.

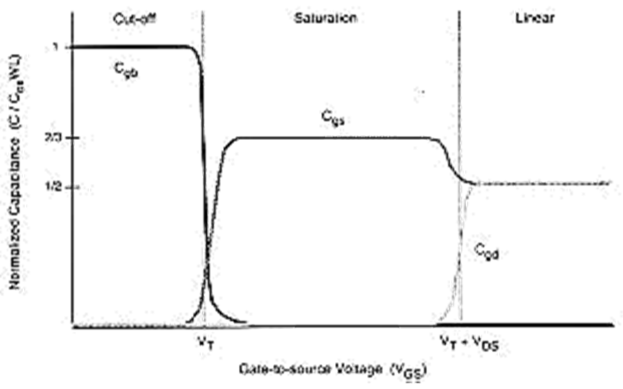


**Figure :** Variation of oxide capacitances as functions of gate-to-source voltage.

Now we consider the voltage-dependent source-to-substrate and drain-to-substrate capacitances, Csb and Cdb. Both of these capacitances are due to the depletion charge surrounding the respective source or drain regions of the transistor, which are embedded in the substrate. Figure 10 shows the simplified geometry of an n-type diffusion region within the p-type substrate. Here, the diffusion region has been approximated by a rectangular box, which consists of five planar pn-junctions. The total junction capacitance is a function of the junction area (sum of all planar junction areas), the doping densities, and the applied terminal voltages. Accurate methods for estimating the junction capacitances based on these data are readily available in the literature, therefore, a detailed discussion of capacitance calculations will not be presented here.

One important aspect of parasitic device junction capacitances is that the amount of capacitance is a linear function of the junction area. Consequently, the size of the drain or the source diffusion area dictates the amount of parasitic capacitance. In sub-micron technologies, where the overall dimensions of the individual devices are scaled down, the parasitic junction capacitances also decrease significantly.

It was already mentioned that the MOSFET parasitic capacitances are mainly responsible for the intrinsic delay of logic gates. We have seen that both the oxide-related parasitic capacitances and the junction capacitances tend to decrease with shrinking device dimensions, hence, the relative significance of intrinsic gate delay diminishes in sub-micron technologies.

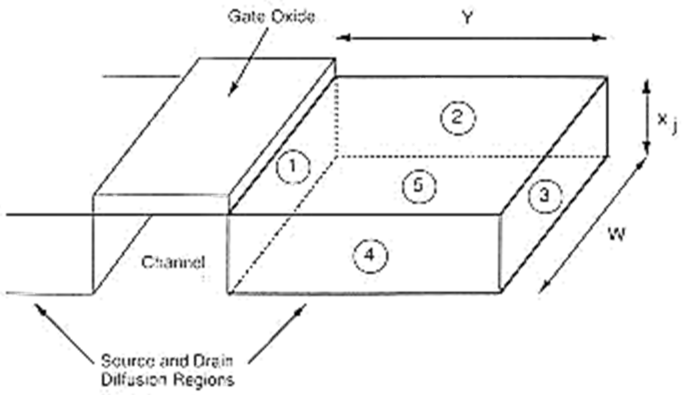
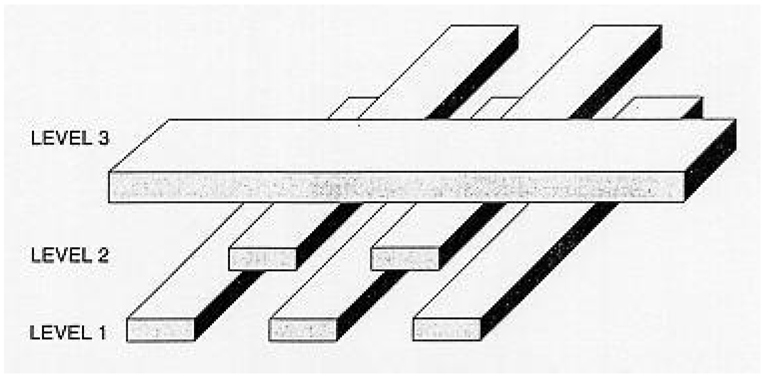


**Figure:** Three-dimensional view of the n-type diffusion region within the p-type substrate.

## *DISTRIBUTED RC EFFECTS*

In a typical VLSI chip, the parasitic interconnect capacitances are among the most difficult parameters to estimate accurately. Each interconnection line (wire) is a three dimensional structure in metal and/or polysilicon, with significant variations of shape, thickness, and vertical distance from the ground plane (substrate). Also, each interconnect line is typically surrounded by a number of other lines, either on the same level or on different levels. Figure 4.11 shows a possible, realistic situation where interconnections on three different levels run in close proximity of each other. The accurate estimation of the parasitic capacitances of these wires with respect to the ground plane, as well as with respect to each other, is obviously a complicated task.



Unfortunately for the VLSI designers, most of the conventional computer-aided VLSI design tools have a relatively limited capability of interconnect parasitic estimation. This is true even for the design tools regularly used for sub-micron VLSI design, where interconnect parasitics were shown to be very dominant. The designer should therefore be aware of the physical problem and try to incorporate this knowledge early in the design phase, when the initial floorplanning of the chip is done.

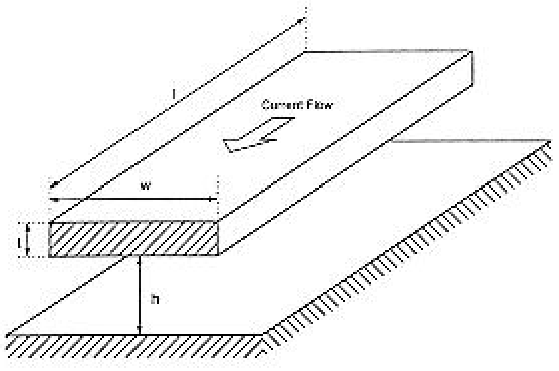


**Figure:** Interconnect segment running parallel to the surface, used for parasitic capacitance estimations.

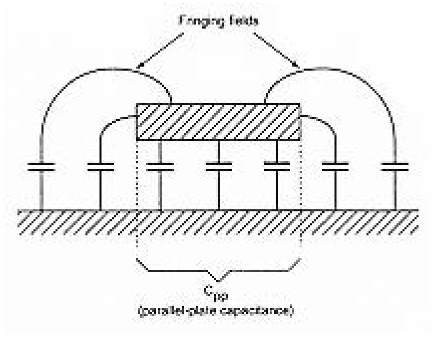


**Figure :** Influence of fringing electric fields upon the parasitic wire capacitance

First, consider the section of a single interconnect which is shown in Fig12. It is assumed that this wire segment has a length of (l) in the current direction, a width of (w) and a thickness of (t). Moreover, we assume that the interconnect segment runs parallel to the chip surface and is separated from the ground plane by a dielectric (oxide) layer of height (h). Now, the correct estimation of the parasitic capacitance with respect to ground is an important issue. Using the basic geometry given in Fig 12, one can calculate the parallel-plate capacitance Cpp of the interconnect segment. However, in interconnect lines where the wire thickness (t) is comparable in magnitude to the ground-plane distance (h), fringing electric fields significantly increase the total parasitic capacitance (Fig).

Figure shows the variation of the fringing-field factor FF = $C_{total}/C_{pp}$, as a function of (t/h), (w/h) and (w/l). It can be seen that the influence of fringing fields increases with the decreasing (w/h) ratio, and that the fringing-field capacitance can be as much as 10-20 times larger than the parallel-plate capacitance. It was mentioned earlier that the sub-micron fabrication technologies allow the width of the metal lines to be decreased somewhat, yet the thickness of the line must be preserved in order to ensure structural integrity. This situation, which involves narrow metal lines with a considerable vertical thickness, is especially vulnerable to fringing field effects.

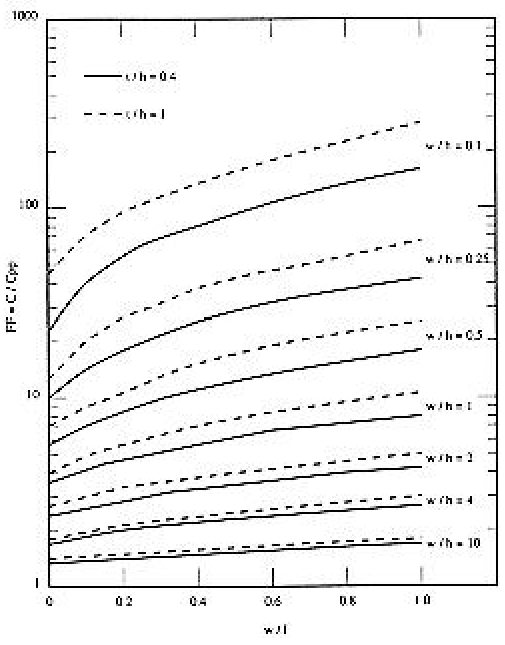


**Figure:** Variation of the fringing-field factor with the interconnect geometry.

A set of simple formulas developed by Yuan and Trick in the early 1980's can be used to estimate the capacitance of the interconnect structures in which fringing fields complicate the parasitic capacitance calculation. The following two cases are considered for two different ranges of line width (w).

$$C=\varepsilon\left[\frac{\left(w-\frac{t}{2}\right)}{h}+\frac{2\pi}{\ln\left(1+\frac{2h}{t}+\sqrt{\frac{2h}{t}\left(\frac{2h}{t}+2\right)}\right)}\right] \quad for \quad w\geq\frac{t}{2}$$

$$C=\varepsilon\left[\frac{w}{h}+\frac{\pi\left(1-0.0543\cdot\frac{t}{2h}\right)}{\ln\left(1+\frac{2h}{t}+\sqrt{\frac{2h}{t}\left(\frac{2h}{t}+2\right)}\right)}+1.47\right]\qquad for\qquad w<\frac{t}{2}$$

(4.2)

These formulas permit the accurate approximation of the parasitic capacitance values to within 10% error, even for very small values of (t/h). Figure 4.15 shows a different view of the line capacitance as a function of (w/h) and (t/h). The linear dash-dotted line in this plot represents the corresponding parallel-plate capacitance, and the other two curves represent the actual capacitance, taking into account the fringing-field effects.

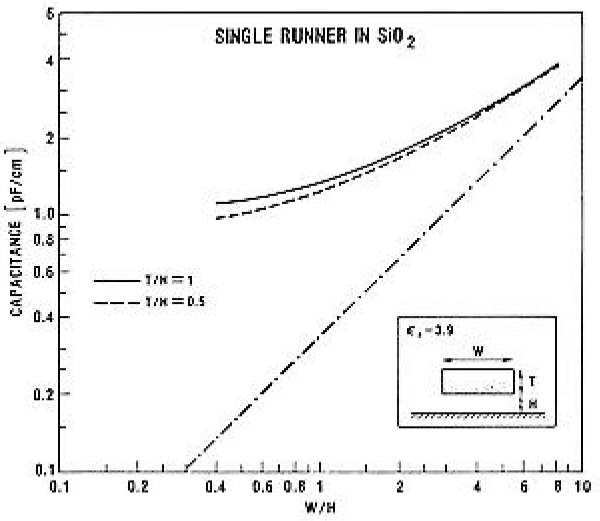


**Figure-15:** Capacitance of a single interconnect, as a function of (w/h) and (t/h).

Now consider the more realistic case where the interconnection line is not "alone" but is coupled with other lines running in parallel. In this case, the total parasitic capacitance of the line is not only increased by the fringing-field effects, but also by the capacitive coupling between the lines. Figure 16 shows the capacitance of a line which is coupled with two other lines on both sides, separated by the minimum design rule. Especially if both of the neighboring lines are biased at ground potential, the total parasitic capacitance of the interconnect running in the middle (with respect to the ground plane) can be more than 20 times as large as the simple parallel-plate capacitance. Note that the capacitive coupling between neighboring lines is increased when the thickness of the wire is comparable to it

Figure shows the cross-section view of a double-metal CMOS structure, where the individual parasitic capacitances between the layers are also indicated. The cross-section does not show a MOSFET, but just a portion of a diffusion region over which some metal lines may pass. The inter-layer capacitances between the metal-2 and metal-1, metal-1 and polysilicon, and metal-2 and polysilicon are labeled as Cm2m1, Cm1p and Cm2p, respectively. The other parasitic capacitance components are defined with respect to the substrate. If the metal line passes over an active region, the oxide thickness underneath is smaller (because of the active area window), and consequently, the capacitance is larger. These special cases are labeled as Cm1a and Cm2a. Otherwise, the thick field oxide layer results in a smaller capacitance value.

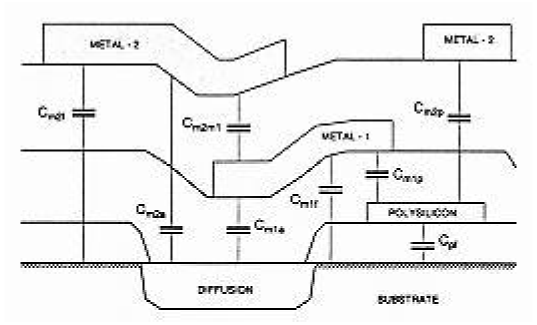


**Figure-17:** Cross-sectional view of a double-metal CMOS structure, showing capacitances between layers.

The vertical thickness values of the different layers in a typical 0.8 micron CMOS technology are given below as an example.

| | | | |
|---|---|---|---|
| Field oxide thickness | 0.52 | um | |
| Gate oxide thickness | 16.0 | nm | |
| Poly 1 thickness | 0.35 | um | (minimum width 0.8 um) |
| Poly-metal oxide thickness | 0.65 | um | |
| Metal 1 thickness | 0.60 | um | (minimum width 1.4 um) |
| Via oxide thickness | 1.00 | um | |
| Metal 2 thickness | 1.00 | um | (minimum width 1.6 um) |
| n+ junction depth | 0.40 | um | |

| | | |
|---|---|---|
| p+ junction depth | 0.40 | um |
| n-well junction depth | 3.50 | um |

The list below contains the capacitance values between various layers, also for a typical 0.8 micron CMOS technology.

| | | | |
|---|---|---|---|
| Poly over field oxide | (area) | 0.066 | fF/um2 |
| Poly over field oxide | (perimeter) | 0.046 | fF/um |
| Metal-1 over field oxide | (area) | 0.030 | fF/um2 |
| Metal-1 over field oxide | (perimeter) | 0.044 | fF/um |
| Metal-2 over field oxide | (area) | 0.016 | fF/um2 |
| Metal-2 over field oxide | (perimeter) | 0.042 | fF/um |
| Metal-1 over poly | (area) | 0.053 | fF/um2 |
| Metal-1 over poly | (perimeter) | 0.051 | fF/um |
| Metal-2 over poly | (area) | 0.021 | fF/um2 |
| Metal-2 over poly | (perimeter) | 0.045 | fF/um |
| Metal-2 over metal-1 | (area) | 0.035 | fF/um2 |
| Metal-2 over metal-1 | (perimeter) | 0.051 | fF/um |

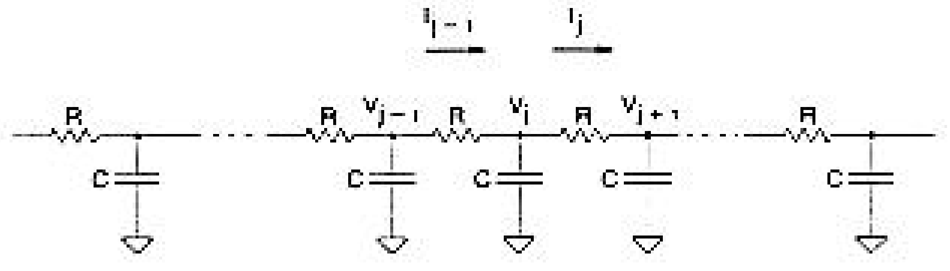
For the estimation of interconnect capacitances in a complicated three-dimensional structure, the exact geometry must be taken into account for every portion of the wire. Yet this requires an unacceptable amount of computation in a large circuit, even if simple formulas are applied for the calculation of capacitances. Usually, chip manufacturers supply the area capacitance (parallel-plate cap) and the perimeter capacitance (fringing-field cap) figures for each layer, which are backed up by measurement of capacitance test structures. These figures can be used to extract the parasitic capacitances from the mask layout. It is often prudent to include test structures on chip that enable the designer to independently calibrate a process to a set of design tools. In some cases where the entire chip performance is influenced by the parasitic capacitance of a specific line, accurate 3-D simulation is the only reliable solution.

***Interconnect Resistance Estimation***

The parasitic resistance of a metal or polysilicon line can also have a profound influence on the signal propagation delay over that line. The resistance of a line depends on the type of material used (polysilicon, aluminum, gold ...), the dimensions of the line and finally, the number and locations of the contacts on that line. Consider again the interconnection line shown in Fig12. The total resistance in the indicated current direction can be found as

$$R_{metal} = \rho \cdot \frac{l}{w \cdot t} = R_{sheet}\left(\frac{l}{w}\right)$$

where the greek letter ro represents the characteristic resistivity of the interconnect material, and Rsheet represents the sheet resistivity of the line, in (ohm/square). For a typical polysilicon layer, the sheet resistivity is between 20-40 ohm/square, whereas the sheet resistivity of silicide is about 2- 4 ohm/square. Using the formula given above, we can estimate the total parasitic resistance of a wire segment based on its geometry. Typical metal-poly and metal-diffusion contact resistance values are between 20-30 ohms, while typical via resistance is about 0.3 ohms.



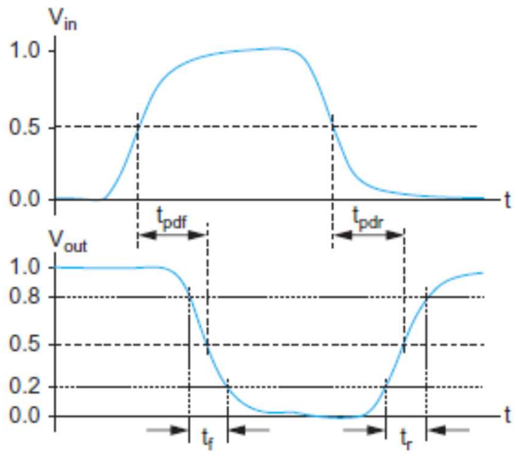
In most short-distance aluminum and silicide interconnects, the amount of parasitic wire resistance is usually negligible. On the other hand, the effects of the parasitic resistance must be taken into account for longer wire segments. As a first-order approximation in simulations, the total lumped resistance may be assumed to be connected in series with the total lumped capacitance of the wire. A much better approximation of the influence of distributed parasitic resistance can be obtained by using an RC-ladder network model to represent the interconnect segment (Fig18). Here, the interconnect segment is divided into smaller, identical sectors, and each sector is represented by an RC-cell. Typically, the number of these RC-cells (i.e., the resolution of the RC model) determines the accuracy of the simulation results. On the other hand, simulation time restrictions usually limit the resolution of this distributed line model.

## 2. SWITCHING CHARACTERISTICS

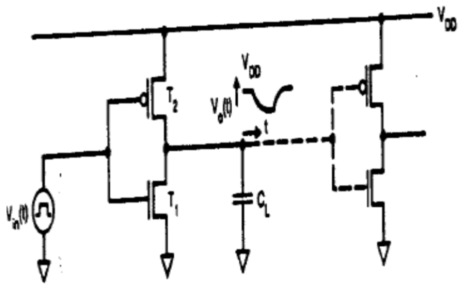
Delay and power are influenced as much by the wires as by the transistors, delves into interconnect analysis and design. A chip is of no value if it cannot reliably accomplish its function, how we achieve robustness in designs.

We begin with a few definitions illustrated in Figure,

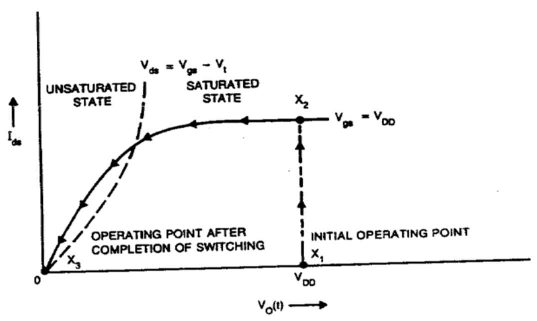
- Propagation delay time, tpd = maximum time from the input crossing 50% to the output crossing 50%
- Contamination delay time, tcd = minimum time from the input crossing 50% to the output crossing 50%
- Rise time, tr = time for a waveform to rise from 20% to 80% of its steady-state value
- Fall time, tf = time for a waveform to fall from 80% to 20% of its steady-state value
- Edge rate, trf = (tr + tf )/2



Intuitively, we know that when an input changes, the output will retain its old value for at least the contamination delay and take on its new value in at most the propagation delay. We sometimes differentiate between the delays for the output rising, tpdr /tcdr , and the output falling, tpdf /tcdf. Rise/fall times are also sometimes called slopes or edge rates. Propagation and contamination delay times are also called max-time and min-time, respectively. The gate that charges or discharges a node is called the driver and the gates and wire being driven are called the load. Propagation delay is usually the most relevant value of interest, and is often simply called delay.



(a)



***Empirical delay model:***

A simulator is used to model the gate. The measured values are backsubstituted into appropriate delay equations. The delay of simple gates may be approximated by the delay of an equivalent inverter. In a simple gate circuit, series connected 'i' number of MOSFETs result in an effective β of;

$$1/\beta_{eff} = 1/\beta_1 + 1/\beta_2 + \ldots + 1/\beta_i$$

In a simple gate circuit, parallel connected 'i' number of MOSFETs result in an effective β of;

$$\beta_{eff} = \min\{\beta_1, \beta_2, \ldots, \beta_i\}$$

Input waveform slope affects the delay: Input signal is not a step function, it has a finite rise and fall times. Input capacitance affects the delay; Gate input capacitance is a function of gate input voltage; MOSFET gate to drain capacitance increases effective input capacitance (bootstrapping).

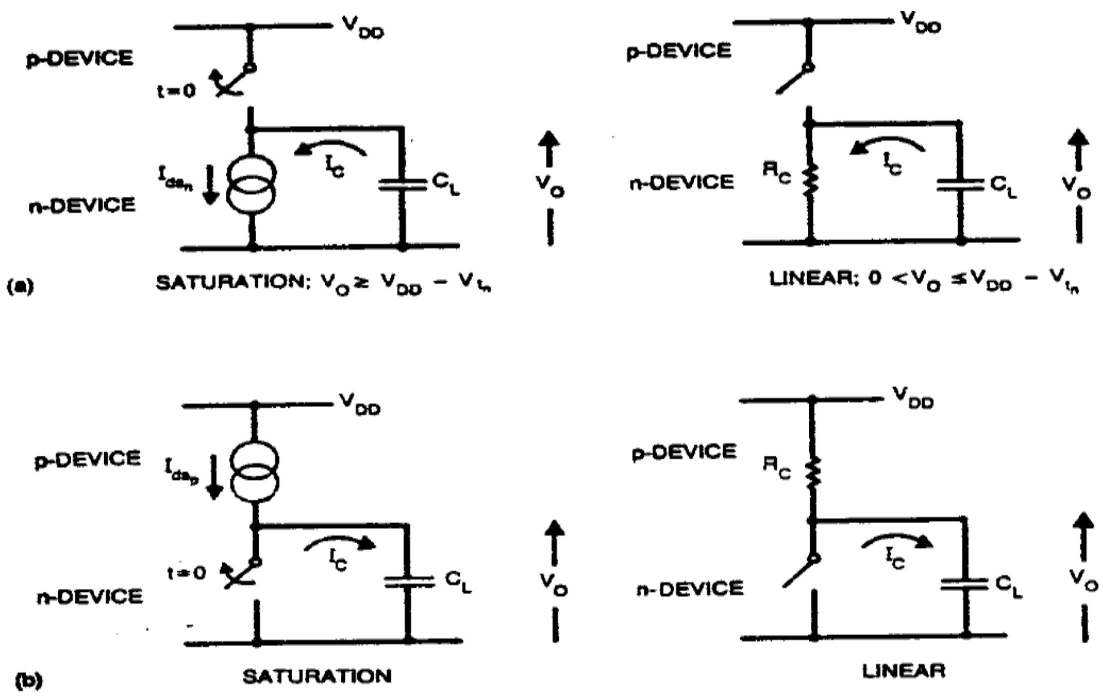
Switch-Level RC models: This is an RC modeling technique that represents transistors as a resistance discharging or charging a capacitance. The models include,

- Simple RC delay model,
- RC-tree model

Macro modeling: Logic gates are simple delay elements. Simulated gate delay characteristics are approximated as;

$$t_d = t_{internal} + k\, t_{output}$$

where k is the loading capacitance, toutput is delay per loading capacitance and tinternal is delay for zero loading capacitance. Body Effect as a dynamic problem: Place the transistors with the latest arriving signals nearest to the gate output. If diffusion is used as wiring then use it for MOSFETs closest to the gate output.



on the $V_{gs} = V_{DD}$ characteristic curve towards point $X_3$ at the origin. From the switching characteristics shown in Fig. 4.15, it is evident that the fall time, $t_f$, consists of two intervals:

1 $t_{f1}$ = period during which the capacitor voltage, $V_o$, drops from $0.9\ V_{DD}$ to $(V_{DD} - V_{t_n})$.
2 $t_{f2}$ = period during which the capacitor voltage, $V_o$, drops from $(V_{DD} - V_{t_n})$ to $0.1\ V_{DD}$.

The equivalent circuits that illustrate the above behavior are shown in Fig. 4.17. From Fig. 4.17a, while in saturation

$$C_L \frac{dV_o}{dt} + \frac{\beta_n}{2}(V_{DD} - V_{t_n})^2 = 0; \qquad V_o \geqslant V_{DD} - V_{t_n}. \tag{4.21}$$

Integrating from $t = t_1$, corresponding to $V_o = 0.9\ V_{DD}$, to $t = t_2$ corresponding to $V_o = (V_{DD} - V_{t_n})$ results in

$$\begin{aligned} t_{f1} &= 2\frac{C_L}{\beta_n(V_{DD} - V_{t_n})^2}\int_{V_{DD}-V_{t_n}}^{0.9V_{DD}} dV_o \\ &= \frac{2C_L(V_{t_n} - 0.1\ V_{DD})}{\beta_n(V_{DD} - V_{t_n})^2}. \end{aligned} \tag{4.22}$$

When the n-device begins to operate in the linear region, the discharge current is no longer constant. The time, $t_{f2}$, taken to discharge the capacitor voltage from $(V_{DD} - V_{t_n})$ to 0.1 $V_{DD}$ can be obtained as before, giving

$$t_{f2} = \frac{C_L}{\beta_n(V_{DD} - V_{t_n})} \int_{0.1V_{DD}}^{V_{DD}-V_{t_n}} \frac{dV_o}{\dfrac{V_o^2}{2(V_{DD} - V_{t_n})} - V_o} \tag{4.23}$$

$$= \frac{C_L}{\beta_n(V_{DD} - V_{t_n})} \ln\left(\frac{19\ V_{DD} - 20\ V_{t_n}}{V_{DD}}\right).$$

Thus the complete term for the fall time, $t_f$ is

$$t_f = 2\frac{C_L}{\beta_n(V_{DD} - V_{t_n})} \times \left[\frac{V_{t_n} - 0.1\ V_{DD}}{V_{DD} - V_{t_n}} + \frac{1}{2}\ln\left(\frac{19\ V_{DD} - 20\ V_{t_n}}{V_{DD}}\right)\right]. \tag{4.24}$$

If we make the assumption that $V_{t_n} \approx 0.2\ V_{DD}$ (in a 5 volt process $V_{t_n} \approx 1$ v $V_{t_p} \approx -1$ v), then $t_f$ can be approximated as

$$t_f \approx 4\frac{C_L}{\beta_n V_{DD}}. \tag{4.25}$$

### 4.4.2 Rise time

Due to the symmetry of the CMOS circuit, a similar approach may be used to obtain the rise time, $t_r$ (Fig. 4.17b). Thus

$$t_r = 2\frac{C_L}{\beta_p(V_{DD} - |V_{t_p}|)} \times \left[\frac{|V_{t_p}| - 0.1\ V_{DD}}{V_{DD} - |V_{t_p}|} + \frac{1}{2}\ln\left(\frac{19\ V_{DD} - 20|V_{t_p}|}{V_{DD}}\right)\right]. \tag{4.26}$$

As before, with $|V_{t_p}| \approx 0.2\ V_{DD}$, Eq. (4.26) reduces to

$$t_r \approx 4\frac{C_L}{\beta_p V_{DD}}. \tag{4.27}$$

For equally sized n- and p-transistors, where $\beta_n = 2\beta_p$

$$t_f = \frac{t_r}{2}. \tag{4.28}$$

Thus the fall time is faster than the rise time, primarily due to different carrier mobilities associated with the p- and n-devices (i.e.,

$\mu_n \approx 2\mu_p$). Therefore, if we want to have approximately the same rise and fall time for an inverter we need to make

$$\frac{\beta_n}{\beta_p} = 1.$$

This implies that the channel width for the p-device must be increased to approximately two times that of the n-device, so

$$W_p = 2W_n.$$

Note that to accurately specify the width ratio required to achieve equal rise and fall times, an accurate ratio of $\mu_n$ and $\mu_p$ must be known. These, in turn, depend on process parameters.

### 4.4.3 Delay time $\tau_d$

In an MOS circuit, the delay of a single gate is dominated by the output rise and fall time. The delay is approximately given by

$$t_{dr} = \frac{t_r}{2} \qquad (4.29)$$
$$t_{df} = \frac{t_f}{2}.$$

The average gate delay for rising and falling transitions is then

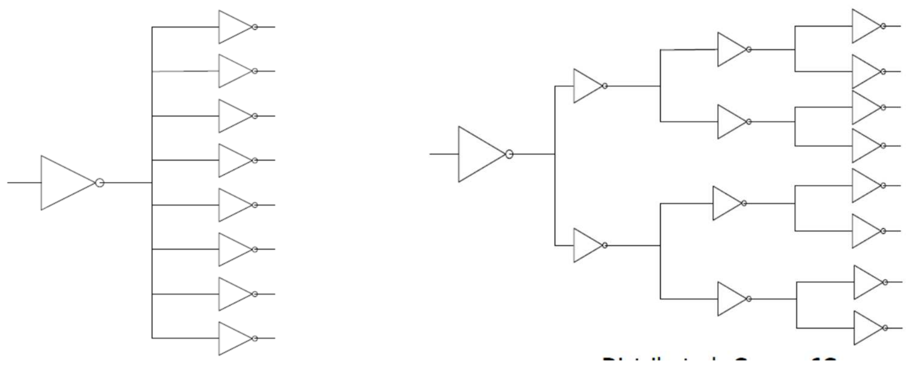
$$\tau_{av} = \frac{t_{df} + t_{dr}}{2} = \frac{t_r + t_f}{4}. \qquad (4.30)$$

## 3. STAGE RATIO

A chain of increasingly larger inverters is usually employed to drive large loads. The ratio by which each stage is increased in size is called the stage ratio.

***Transistor Gate Sizing***

For a simple CMOS inverter, both the static and dynamic characteristics of the circuit were dependent on the transistor properties and *Vt*. It is therefore clear that designers must take care to control these parameters to ensurethat circuits work well. In practice, there is only a limited amount of control available to the designer. The threshold voltage cannot (or should not) be modified at will by the designer. *Vt* is strongly dependent on the doping of the substrate material, the thickness of the gate oxide and the potential in the substrate. The doping in the substrate and the oxide thickness are physical parameters that are determined by the process designers, hence the circuit designer is unable to modify them. The substrate potential can be varied at will by the designer, and this will influence *Vt* via a mechanism known as the body effect. It is unwise to tamper with the substrate potential, and for digital design the bulk connection is always shorted to $VSS$ for NMOS and $VDD$ for PMOS. since the carrier mobility ( $n$ or $p$) is a fixed property of the semiconductor (affected by process), $ox$ is a fixed physical parameter and $tox$ is set by process. The designer is therefore left with the two dimensional parameters, $W$ and $L$. In principle both $W$ and $L$ can be varied at will by the chip designer, provided they stay within the design rules for the minimum size of any feature. However, the smaller $L$, the larger and smaller the gate capacitance, hence the quicker the circuit, so with few exceptions designers always use the smallest possible $L$ available in a process. For example, in a 0.1 m process the gate length will be 0.1 m for virtually every single transistor on the chip. To conclude, the only parameter that the circuit designer can manipulate at will is the transistor gate width, $W$, and much of the following discussion will deal with the effect of modifying $W$ and making the best selection.



Transistor sizing offers at best a linear performance benefit, while supply voltage offers an exponential performance benefit. As a general rule, minimum energy under a performance constraint is thus achieved by using minimum width transistors and raising the supply voltage if necessary from the minimum energy point until the performance is achieved (assuming the performance requirement is low enough that the circuit remains in the subthreshold regime).

If Vt variations from random dopant fluctuations are extremely high, wider transistors might become advantageous to reduce the variability and its attendant risk of high leakage. Also, if one path through a circuit is far more critical than the others, upsizing the transistors in that path for speed might be better than raising the supply voltage to the entire circuit. When minimum-width transistors are employed, wires are likely to contribute the majority of the switching capacitance. To shorten wires, subthreshold cells should be as small as possible; the cell height is generally set by the minimum height of a flip-flop. Good floorplanning and placement is essential.

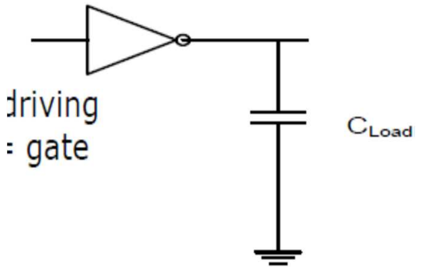
***STAGE RATIO FOR LARGER LOAD***

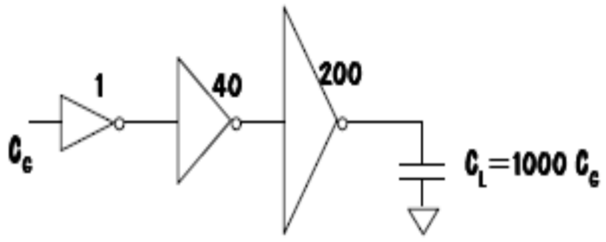
If large loads have to be driven, the delay may increase drastically. Large loads are output capacitances, clock trees, etc.

td = tinv (CL/CG)

Where tinv=is the average delay of a minimum-sized inverter driving another minimum sized inverter, CL = load capacitance& CG = gate capacitance

CL= 1000CG, then td = 1000.tinv





A possibility to reduce the delay, is to use a sequence of n scaled inverters, but not the optimum delay:

td = tinv (40/1)+ tinv (200/40) + tinv (1000/200)= 50tinv

We may decrease the delay by using larger transistors to decrease the resistance. Scaling transistors by factor S results in:

$$\left(\frac{W}{L}\right)_{scaled} = S\left(\frac{W}{L}\right)_{normal}$$

$$R_{scaled} = \frac{R_{scaled}}{S}$$

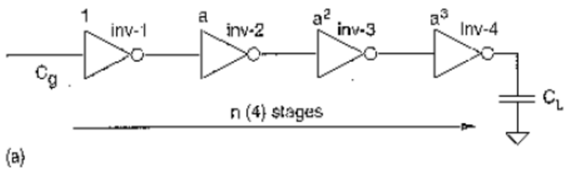
$$t_r = 2.2R_{scaled}\,(C_{in} + C_{load})$$

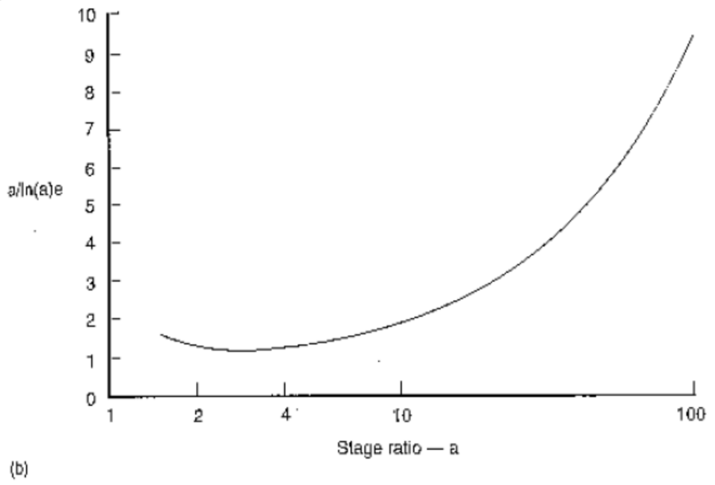
But scaling the transistor also affects the input capacitance of the transistor,

Cin,scaled = S . Cin

The problem is if input signal is placed at inverter 1 what’s the number of stages N and the scaling factor S that will minimize the time needed for the signal to reach Cload td = t1+t2+t3+…+tN

td = R1C2+ R2C3 + R3C4 +…+RN-1CN+RNCload



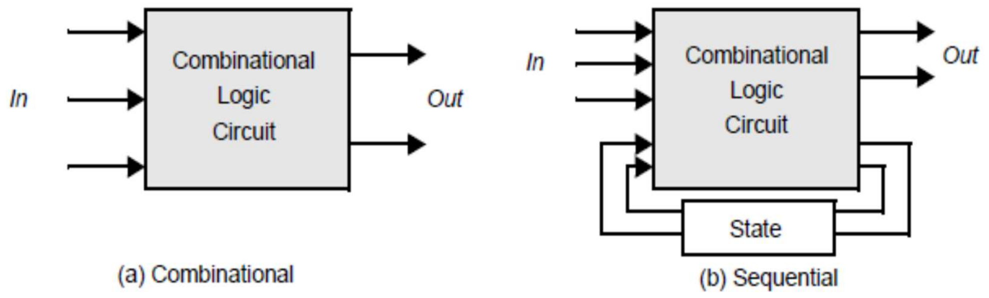


The delay through each stage is atd, where td is the average delay of a minimum-sized inverter driving another minimum-sized inverter. Hence the delay through n stages is natd. If the ratio of the load capacitance to the capacitance of a minimum-sized inverter is R=CL / Cg , then R = an. Hence ln(R)=nln(a). Thus, the variable part of the above equation, normalized to e, is graphed in Figure. The graph shows when a = 2.7 = e would minimize the total delay (dtd/dS = 0). This yields,

S = e =2.71

## 4. SIMPLE EXAMPLES OF COMBINATIONAL AND SEQUENTIAL CIRCUITS USING CMOS

The design considerations for a simple inverter circuit were presented in the previous chapter. In this chapter, the design of the inverter will be extended to address the synthesis of arbitrary digital gates such as NOR, NAND and XOR. The focus will be on combinational logic (or non-regenerative) circuits that have the property that at any point in time, the output of the circuit is related to its current input signals by some Boolean expression (assuming that the transients through the logic gates have settled). No intentional connection between outputs and inputs is present. In another class of circuits, known as sequential or regenerative circuits —to be discussed in a later chapter—, the output is not only a function of the current input data, but also of previous values of the input signals (Figure). This is accomplished by connecting one or more outputs intentionally back to some inputs. Consequently, the circuit "remembers" past events and has a sense of history. A sequential circuit includes a combinational logic portion and a module that holds the state. Example circuits are registers, counters, oscillators, and memory.

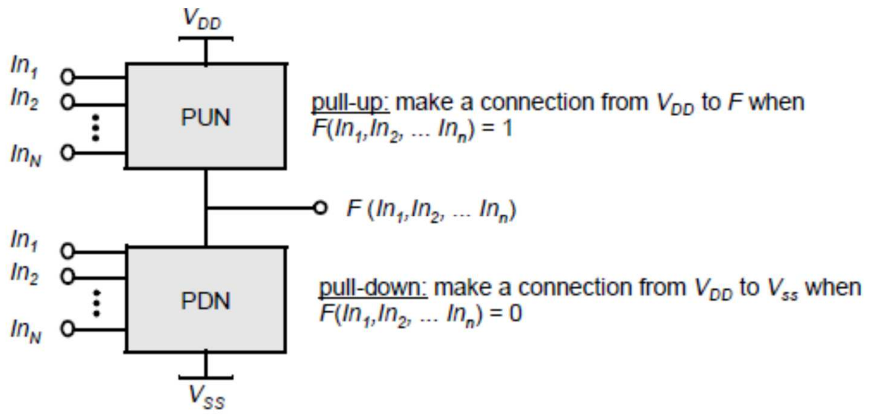


There are numerous circuit styles to implement a given logic function. As with the inverter, the common design metrics by which a gate is evaluated include area, speed, energy and power. Depending on the application, the emphasis will be on different metrics (e.g., in high performance processor, the switching speed of digital circuits is the primary metric while in a battery operated circuit it is the energy dissipation). In addition to these metrics, robustness to noise is also a very important consideration. We will see that certain logic styles (e.g., Dynamic logic) can significantly improve performance, but can be more sensitive to noise. Recently, power dissipation has also become a very important requirement and significant emphasis is placed on understanding the sources of power and approaches to deal with power.

### *Static CMOS Design*

The most widely used logic style is static complementary CMOS. The static CMOS style is really an extension of the static CMOS inverter to multiple inputs. In review, the primary advantage of the CMOS structure is robustness (i.e, low sensitivity to noise), good performance, and low power consumption (with no static power consumption).

The complementary CMOS circuit style falls under a broad class of logic circuits called static circuits in which at every point in time (except during the switching transients), each gate output is connected to either VDD or Vss via a low-resistance path. Also, the outputs of the gates assume at all times the value of the Boolean function implemented by the circuit (ignoring, once again, the transient effects during switching periods). This is in contrast to the dynamic circuit class, that relies on temporary storage of signal values on the capacitance of high-impedance circuit nodes. The latter approach has the advantage that the resulting gate is simpler and faster. On the other hand, its design and operation are more involved than those of its static counterpart, due to an increased sensitivity to noise.


$V_{DD}$
$In_1$
$In_2$
$In_N$
PUN
pull-up: make a connection from $V_{DD}$ to $F$ when $F(In_1,In_2, \ldots In_n) = 1$
$F(In_1,In_2, \ldots In_n)$
$In_1$
$In_2$
$In_N$
PDN
pull-down: make a connection from $V_{DD}$ to $V_{ss}$ when $F(In_1,In_2, \ldots In_n) = 0$
$V_{SS}$

A static CMOS gate is a combination of two networks, called the pull-up network (PUN) and the pull-down network (PDN) (Figure). The figure shows a generic N input logic gate where all inputs are distributed to both the pull-up and pull-down networks. The function of the PUN is to provide a connection between the output and VDD anytime the output of the logic gate is meant to be 1 (based on the inputs). Similarly, the function of the PDN is to connect the output to VSS when the output of the logic gate is meant to be 0. The PUN and PDN networks are constructed in a mutually exclusive fashion such that one and only one of the networks is conducting in steady state. In this way, once the transients have settled, a path always exists between VDD and the output F, realizing a high output ("one"), or, alternatively, between VSS and F for a low output ("zero"). This is equivalent to stating that the output node is always a low-impedance node in steady state. In constructing the PDN and PUN networks, the following observations should be kept in mind,

1) A transistor can be thought of as a switch controlled by its gate signal. An NMOS switch is on when the controlling signal is high and is off when the controlling signal is low. A PMOS transistor acts as an inverse switch that is on when the controlling signal is low and off when the controlling signal is high.

2) The PDN is constructed using NMOS devices, while PMOS transistors are used in the PUN. The primary reason for this choice is that NMOS transistors produce "strong zeros," and PMOS

devices generate “strong ones”. To illustrate this, consider the examples shown in Figure. In Figure 6.3a, the output capacitance is initially charged to VDD. Two possible discharge scenario’s are shown. An NMOS device pulls the output all the way down to GND, while a PMOS lowers the output no further than |VTp| — the PMOS turns off at that point, and stops contributing discharge current. NMOS transistors are hence the preferred devices in the PDN. Similarly, two alternative approaches to charging up a capacitor are shown in Figure 6.3b, with the output load initially at GND. A PMOS switch succeeds in charging the output all the way to VDD, while the NMOS device fails to raise the output above VDD-VTn. This explains why PMOS transistors are preferentially used in a PUN.

3) A set of construction rules can be derived to construct logic functions (Figure). NMOS devices connected in series corresponds to an AND function. With all the inputs high, the series combination conducts and the value at one end of the chain is transfered to the other end. Similarly, NMOS transistors connected in parallel represent an OR function. A conducting path exists between the output and input terminal if at least one of the inpurs is high. Using similar arguments, construction rules for PMOS networks can be formulated. A series connection of PMOS conducts if both inputs are low, representing a NOR function (A.B = A+B), while PMOS transistors in parallel implement a NAND (A+B = A·B.

4) Using De Morgan’s theorems ((A + B) = A·B and A·B = A +B), it can be shown that the pull-up and pull-down networks of a complementary CMOS structure are dual networks. This means that a parallel connection of transistors in the pull-up network corresponds to a series connection of the corresponding devices in the pull-down network, and vice versa. Therefore, to construct a CMOS gate, one of the networks (e.g., PDN) is implemented using combinations of series and parallel devices. The other network (i.e., PUN) is obtained using duality principle by walking the hierarchy, replacing series subnets with parallel subnets, and parallel subnets with series subnets. The complete CMOS gate is constructed by combining the PDN with the PUN.

5) The complementary gate is naturally inverting, implementing only functions such as NAND, NOR, and XNOR. The realization of a non-inverting Boolean function (such as AND OR, or XOR) in a single stage is not possible, and requires the addition of an extra inverter stage.

6) The number of transistors required to implement an N-input logic gate is 2N.

***CMOS NAND GATE***

Figure shows a two-input NAND gate (F = A·B). The PDN network consists of two NMOS devices in series that conduct when both A and B are high. The PUN is the dual network, and consists of two parallel PMOS transistors. This means that F is 1 if A = 0 or B = 0, which is equivalent to F = A·B. The truth table for the simple two input NAND gate is given in Table 6.1. It can be verified that the output F is always connected to either VDD or GND, but never to both at the same time.

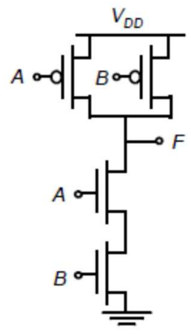
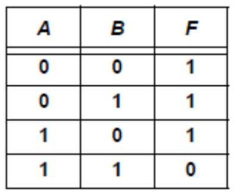


**Table 6.1** Truth Table for 2 input NAND



| A | B | F |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

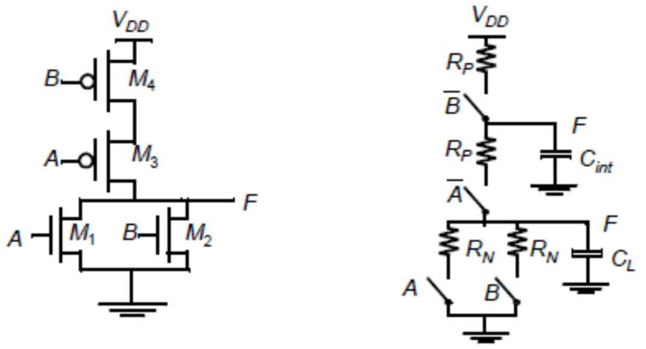
***Design Consideration***:

The important point to take away from the above discussion is that the noise margins are input pattern dependent. For the above example, a smaller input glitch will cause a transition at the output if only one of the inputs makes a transition. Therefore, this condition has a lower low noise margin. A common practice when characterizing gates such as NAND and NOR is to connect all the inputs together. This unfortunately does not represent the worst-case static behavior. The data dependencies should be carefully modelled.

The computation of propagation delay proceeds in a fashion similar to the static inverter. For the purpose of delay analysis, each transistor is modeled as a resistor in series with an ideal switch. The value of the resistance is dependent on the power supply voltage and an equivalent large signal resistance, scaled by the ratio of device width over length, must be used. The logic is transformed into an equivalent RC network that includes the effect of internal node capacitances.

***CMOS NOR GATE***

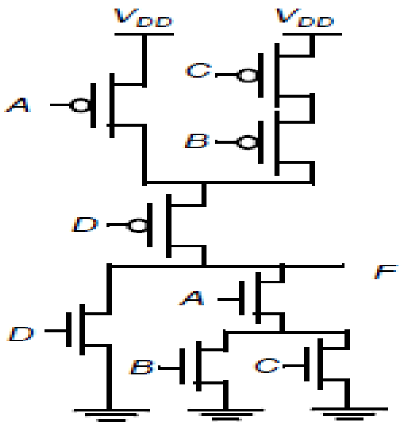
The CMOS implementation of a NOR gate (F = A + B)' is shown in Figure 6.10. The output of this network is high, if and only if both inputs A and B are low. The worst-case pull-down transition happens when only one of the NMOS devices turns on (i.e., if either A or B is high). Assume that the goal is to size the NOR gate such that it has approximately the same delay as an inverter with the following device sizes: NMOS 0.5mm/0.25mm and PMOS 1.5mm/0.25mm.



Since the pull-down path in the worst case is a single device, the NMOS devices (M1 and M2) can have the same device widths as the NMOS device in the inverter. For the output to be pulled high, both devices must be turned on. Since the resistances add, the devices must be made two times larger compared to the PMOS in the inverter (i.e., M3 and M4 must have a size of 3mm/0.25mm). Since PMOS devices have a lower mobility relative to NMOS devices, stacking devices in series must be avoided as much as possible. A NAND implementation is clearly prefered over a NOR implementation for implementing generic logic.

***Compound gates***

Using complementary CMOS logic, consider the synthesis of a complex CMOS gate whose function is F = {D + A· (B +C)}’. The first step in the synthesis of the logic gate is to derive the pull-down network as shown in Figure 6.6a by using the fact that NMOS devices in series implements the AND function and parallel device implements the OR function. The next step is to use duality to derive the PUN in a hierarchical fashion. The PDN network is broken into smaller networks (i.e., subset of the PDN) called sub-nets that simplify the derivation of the PUN. In



(c) complete gate

Figure, the subnets (SN) for the pull-down network are identified At the top level, SN1 and SN2 are in parallel so in the dual network, they will be in series. Since SN1 consists of a single transistor, it maps directly to the pull-up network. On the other hand, we need to recursively apply the duality rules to SN2. Inside SN2, we have SN3 and SN4 in series so in the PUN they will appear in parallel. Finally, inside SN3, the devices are in parallel so they will appear in series in the PUN. The complete gate is shown in Figure 6.6c. The reader can verify that for every possible input cobmination, there always exists a path to either VDD or GND.

## *LATCHES AND REGISTERS*

### *Latch using NAND gates*

Figure shows a SR latch using two NAND2 gates. Note in order to hold (preserve) a state, both of the external trigger inputs must be equal to logic “1”. The state of the circuit can be changed by pulling the set input or reset input to zero. The gate level schematic and the corresponding block diagram representation of the NAND-based SR latch are shown in Figure.

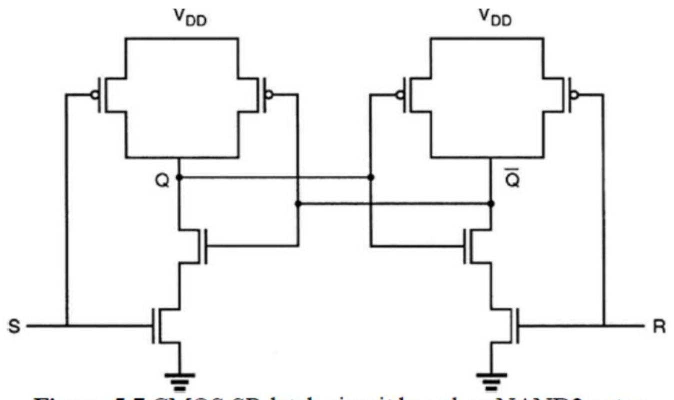
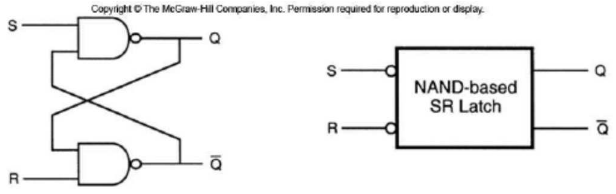


**Figure 5.7** CMOS SR latch circuit based on NAND2 gates


Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

| $S$ | $R$ | $Q_{n+1}$ | $\overline{Q_{n+1}}$ | Operation |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | not allowed |
| 0 | 1 | 1 | 0 | set |
| 1 | 0 | 0 | 1 | reset |
| 1 | 1 | $Q_n$ | $\overline{Q_n}$ | hold |

Note that a NAND-based SR latch responds to an active low input signals, while the NOR-based SR latch responds to an active high inputs. The small circles at the S and R input terminals indicate that the circuit responds to active low input signals. The truth table of the NAND SR latch is also shown in Figure. The same approach used in the timing analysis of the NOR-based SR latches can be applied to NAND-based SR latches. While there is not-allowed input combination for the JK latch, there is still a potential problem. If both inputs are equal to logic "1" during the active phase of the clock pulse, the output of the circuit will oscillate (toggle) continuously until either clock becomes inactive (CK goes to zero), or one of the input signals goes to zero.

To prevent this undesirable timing problem, the clock pulse width must be made smaller than the input to output propagation delay of the JK latch circuit (thus the clock signal must go low before the output has an opportunity to switch again, which prevents uncontrolled oscillation). This clock constraint is difficult to implement for most practical applications.

**CONCLUSION:**

The VLSI designer's challenge is to engineer a system that meets speed requirements while consuming little power or area, operating reliably, and taking little time to design. Circuit simulation is an important tool for calculating delay and will be discussed in depth in Chapter 5, but it takes too long to simulate every possible design; is prone to garbagein, garbage-out mistakes; and doesn't give insight into why a circuit has a particular delay or how the circuit should be changed to improve delay. The designer must also have simple models to quickly estimate performance by hand and explain why some circuits are better than others.

Although transistors are complicated devices with nonlinear current-voltage and capacitance-voltage relationships, for the purpose of delay estimation in digital circuits, they can be approximated quite well as having constant capacitance and an effective resistance R when ON. Logic gates are thus modeled as RC networks. The Elmore delay model estimates the delay of the network as the sum of each capacitance times the resistance through which it must be charged or discharged. Therefore, the gate delay consists of a parasitic delay (accounting for the gate driving its own internal parasitic capacitance) plus an effort delay (accounting for the gate driving an external load). The effort delay depends on the electrical effort (the ratio of load capacitance to input capacitance, also called fanout) and the logical effort (which characterizes the current driving capability of the gate relative to an inverter with equal input capacitance). Even in advanced fabrication processes, the delay vs. electrical effort curve fits a straight line very well. The method of Logical Effort builds on this linear delay model to help us quickly estimate the delay of entire paths based on the effort and parasitic delay of the path. We will use Logical Effort in subsequent chapters to explain what makes circuits fast.

# Notes on
# SUB SYSTEM DESIGN AND TESTING
# in VLSI Design – Unit IV

**Dr. G. Senthil Kumar,**
Associate Professor,
Dept. of ECE, SCSVMV,
email: gsk_ece@kanchiuniv.ac.in

===============================================================

**OBJECTIVES**:

In this lesson, you will be able to discuss architectural choices, design and realization of sub-systems and testing methods of ICs.

**CONTENTS**:

## 1. GENERAL SYSTEM DESIGN

### *General Considerations*

- Lower unit cost
- Higher reliability
- Lower power dissipation, lower weight and lower volume
- Better performance
- Enhanced repeatability
- Possibility of reduced design/development periods

### *Some Problems*

1. How to design complex systems in a reasonable time & with reasonable effort.

2. The nature of architectures best suited to take full advantage of VLSI and the technology
3. The testability of large/complex systems once implemented on silicon

***Some Solution***

Problem 1 & 3 are greatly reduced if two aspects of standard practices are accepted.

1. a) Top-down design approach with adequate CAD tools to do the job

b) Partitioning the system sensibly

c) Aiming for simple interconnections

d) High regularity within subsystem

e) Generate and then verify each section of the design

2. Devote significant portion of total chip area to test and diagnostic facility

3. Select architectures that allow design objectives and high regularity in realization

***Illustration of design processes***

1. Structured design begins with the concept of hierarchy

2. It is possible to divide any complex function into less complex sub functions that is up to leaf cells

3. Process is known as top-down design

4. As a systems complexity increases, its organization changes as different factors become relevant to its creation

5. Coupling can be used as a measure of how much submodels interact

6. It is crucial that components interacting with high frequency be physically proximate, since one may pay severe penalties for long, high-bandwidth interconnects

7. Concurrency should be exploited – it is desirable that all gates on the chip do useful work most of the time

8. Because technology changes so fast, the adaptation to a new process must occur in a short time.

• Conventional circuit symbols

• Logic symbols

• Stick diagram

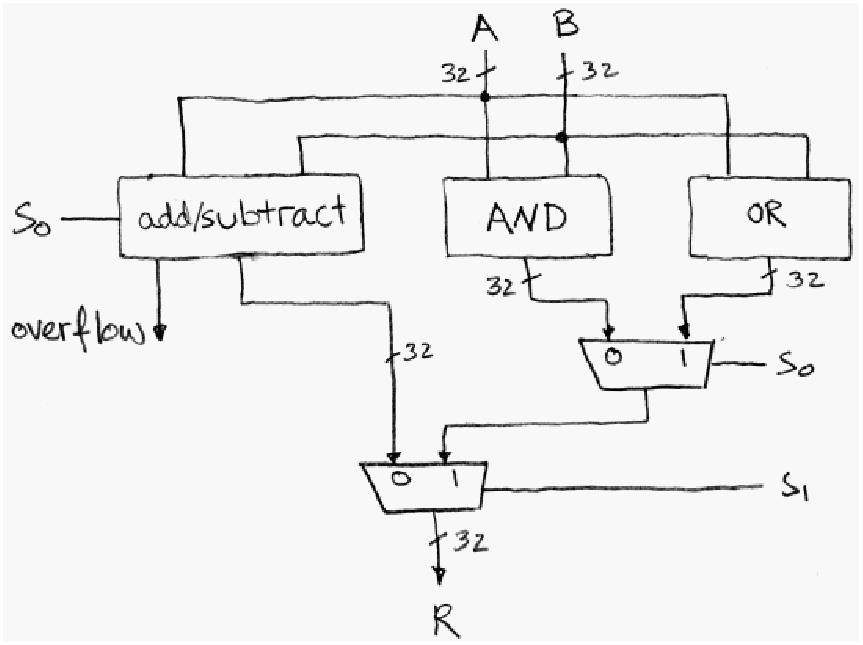
• Any mixture of logic symbols and stick diagram that is convenient at a stage

• Mask layouts

• Architectural block diagrams and floor plans

***DESIGN OF ALU SUBSYSTEMS***

An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer. Modern CPUs contain very powerful and complex ALUs. In addition to ALUs, modern CPUs contain a control unit (CU). Most of the operations of a CPU are performed by one or more ALUs, which load data from input registers. A register is a small amount of storage available as part of a CPU. The control unit tells the ALU what operation to perform on that data, and the ALU stores the result in an output register. The control unit moves the data between these registers, the ALU, and memory.



ALUs are implemented using lower-level components such as logic gates, including and, or, not gates and multiplexers. These building blocks work with individual bits. In principle, an ALU is built from 32 separate 1-bit ALUs. Typically, one constructs separate hardware blocks for each task (e.g., arithmetic and logical operations), where each operation is applied to the 32-bit registers in parallel, and the selection of an operation is controlled by a multiplexer. The advantage of this approach is that it is easy to add new operations to the instruction set, simply by associating an operation with a multiplexer control code. This can be done provided that the mux has sufficient capacity. Otherwise, new data lines must be added to the mux(es), and the CPU must be modified to accomodate these changes.

And/Or Operations. As shown in Figure, a simple (1-bit) ALU operates in parallel, producing all possible results that are then selected by the multiplexer (represented by an oval shape at the output of the and / or gates. The output C is thus selected by the multiplexer. (Note: If the multiplexer were to be applied at the input(s) rather than the output, twice the amount of hardware would be required, because there are two inputs versus one output.)

Different operation as carried out by ALU can be categorized as follows, logical operations − These include operations like AND, OR, NOT, XOR, NOR, NAND, etc.
Bit-Shifting Operations − This pertains to shifting the positions of the bits by a certain number of places either towards the right or left, which is considered a multiplication or division operations.
Arithmetic operations − This refers to bit addition and subtraction. Although multiplication and division are sometimes used, these operations are more expensive to make. Multiplication and subtraction can also be done by repetitive additions and subtractions respectively.

An ALU can be designed by engineers to calculate many different operations. When the operations become more and more complex, then the ALU will also become more and more expensive and also takes up more space in the CPU and dissipates more heat. That is why engineers make the ALU powerful enough to ensure that the CPU is also powerful and fast, but not so complex as to become prohibitive in terms of cost and other disadvantages

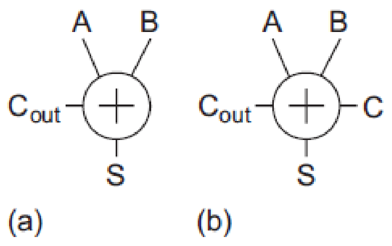
### ***ALU Performance Issues***

When estimating or measuring ALU performance, one wonders if a 32-bit ALU is as fast as a 1-bit ALU - what is the degree of parallelism, and do all operations execute in parallel? In practice, some operations on N-bit operands (e.g., addition with sequential propagation of carries) take O(N) time. Other operations, such as bitwise logical operations, take O(1) time. Since addition can be implemented in a variety of ways, each with a certain level of parallelism, it is wise to consider the possibility of a full adder being a computational bottleneck in a simple ALU. We previously discussed the ripple-carry adder (Figure 3.10) that propagates the carry bit from stage i to stage i+1. It is readily seen that, for an N-bit input, O(N) time is required to propagate the carry to the most significant bit. In contrast, the fastest N-bit adder uses O(log2N) stages in a tree-structured configuration with N-1 one-bit adders. Thus, the complexity of this technique is O(log2N) work. In a sequential model of computation, this translates to O(log2N) time. If one is adding smaller numbers (e.g., up to 10-bit integers with current memory technology), then alookup table can be used that (1) forms a memory address A by concatenating binary representations of the two operands, and (2) produces a result stored in memory that is accessed using A. This takes O(1) time, that is dependent upon memory bandwidth.

***ADDERS***

***Single-Bit Addition***

The half adder of Figure 11.1(a) adds two single-bit inputs, A and B. The result is 0, 1, or 2, so two bits are required to represent the value; they are called the sum S and carry-out Cout. The carry-out is equivalent to a carry-in to the next more significant column of a multibit adder, so it can be described as having double the weight of the other bits. If multiple adders are to be cascaded, each must be able to receive the carry-in. Such a full adder as shown in Figure has a third input called C or Cin.



| $A$ | $B$ | $C_{out}$ | $S$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

The truth tables for the half adder and full adder are given in Tables. For a full adder, it is sometimes useful to define Generate (G), Propagate (P), and Kill (K) signals. The adder generates a carry when Cout is true independent of Cin, so G = A · B. The adder kills a carry when Cout is false independent of Cin, so K = (A · B)’ = (A + B)’. The adder propagates a carry; i.e., it produces a carry-out if and only if it receives a carry-in, when exactly one input is true: P = A (exor) B.

| $A$ | $B$ | $C$ | $G$ | $P$ | $K$ | $C_{out}$ | $S$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | 1 | | | | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| | | 1 | | | | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| | | 1 | | | | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| | | 1 | | | | 1 | 1 |

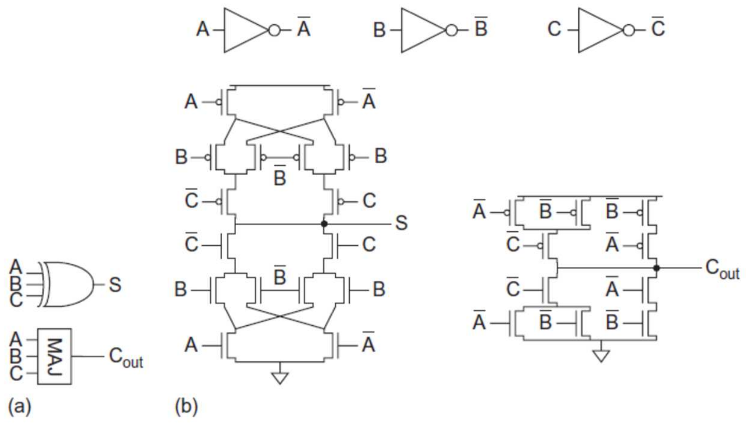
From the truth table, the half adder logic is,

$$S = A \oplus B$$
$$C_{out} = A \cdot B$$

and the full adder logic is,

$$\begin{aligned} S &= A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C + ABC \\ &= (A \oplus B) \oplus C = P \oplus C \\ C_{\text{out}} &= AB + AC + BC \\ &= AB + C(A+B) \\ &= \overline{\overline{A}\overline{B} + \overline{C}(\overline{A}+\overline{B})} \\ &= \text{MAJ}(A, B, C) \end{aligned}$$

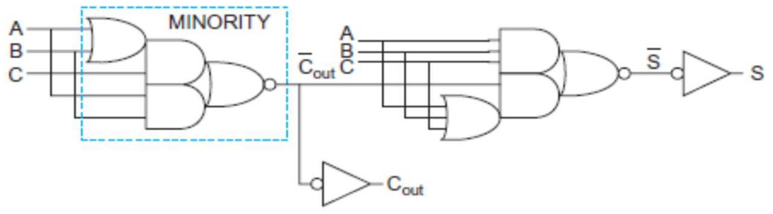
The most straightforward approach to designing an adder is with logic gates. Figure shows a half adder. Figure shows a full adder at the gate (a) and transistor (b) levels. The carry gate is also called a majority gate because it produces a 1 if at least two of the three inputs are 1.



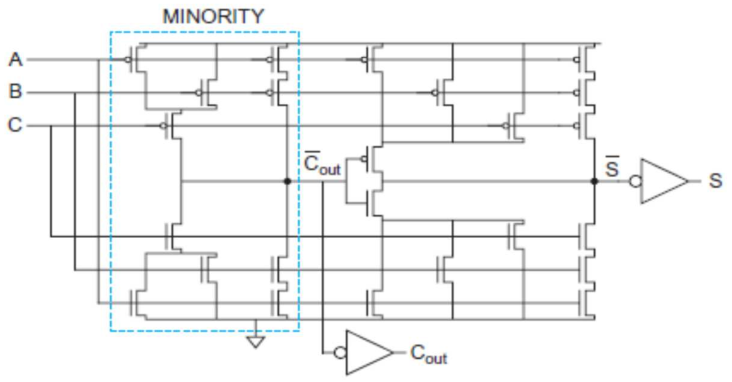
The full adder of Figure employs 32 transistors (6 for the inverters, 10 for the majority gate, and 16 for the 3-input XOR). A more compact design is based on the observation that S can be factored to reuse the Cout term as follows:

$$S = ABC + (A + B + C)\overline{C}_{\text{out}}$$

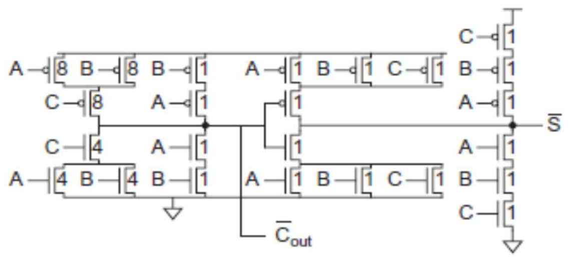
Such a design is shown at the gate (a) and transistor (b) levels in Figure 11.4 and uses only 28 transistors. Note that the pMOS network is identical to the nMOS network rather than being the conduction complement, so the topology is called a mirror adder. This simplification reduces the number of series transistors and makes the layout more uniform. It is possible because the addition function is symmetric ; i.e., the function of complemented inputs is the complement of the function.



(a)



(b)



(c)

The mirror adder has a greater delay to compute S than Cout. In carry-ripple adders (Section 11.2.2.1), the critical path goes from C to Cout through many full adders, so the sizes optimized to favor the critical path using a number of techniques:

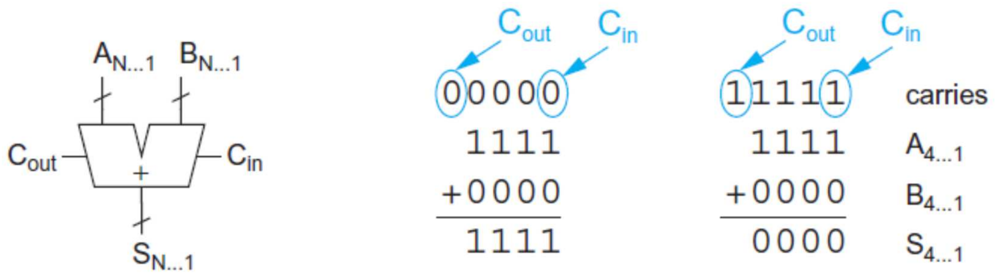
Feed the carry-in signal (C) to the inner inputs so the internal capacitance is already discharged. Make all transistors in the sum logic whose gate signals are connected to the carryin and carry logic minimum size (1 unit). This minimizes the branching effort on the critical path. Keep routing on this signal as short as possible to reduce interconnect capacitance.

Determine widths of series transistors by logical effort and simulation. Build an asymmetric gate that reduces the logical effort from C to Cout at the expense of effort to S. Use relatively large transistors on the critical path so that stray wiring capacitance is a small fraction of the overall capacitance.

Remove the output inverters and alternate positive and negative logic to reduce delay and transistor count to 24.

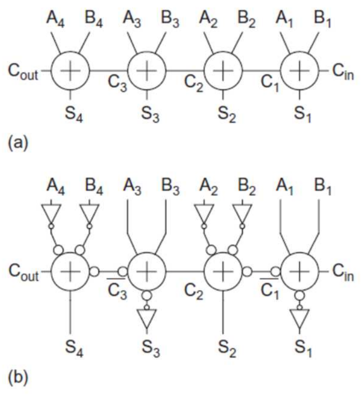
***Carry-Propagate Addition***

N-bit adders take inputs {AN, …, A1}, {BN, …, B1}, and carry-in Cin, and compute the sum {SN, …, S1} and the carry-out of the most significant bit Cout, as shown in Figure. (Ordinarily, this text calls the least significant bit A0 rather than A1. However, for adders, the notation developed on subsequent pages is more graceful if column 0 is reserved to handle the carry.) They are called carry-propagate adders (CPAs) because the carry into each bit can influence the carry into all subsequent bits. For example, Figure shows the addition 11112 + 00002 + 0/1, in which each of the sum and carry bits is influenced by Cin. The simplest design is the carry-ripple adder in which the carry-out of one bit is simply connected as the carry-in to the next. Faster adders look ahead to predict the carry-out of a multibit group. This is usually done by computing group PG signals to indicate whether the multibit group will propagate a carry-in or will generate a carry-out. Long adders use multiple levels of lookahead structures for even more speed.



***Carry-Ripple Adder***

An N-bit adder can be constructed by cascading N full adders, as shown in Figure 11.11(a) for N = 4. This is called a carry-ripple adder (or ripple-carry adder). The carry-out of bit i, Ci, is the carry-in to bit i + 1. This carry is said to have twice the weight of the sum Si. The delay of the adder is set by the time for the carries to ripple through the N stages, so the tC->Cout delay should be minimized.



This delay can be reduced by omitting the inverters on the outputs, as was done in Figure 11.4(c). Because addition is a self-dual function (i.e., the function of complementary inputs is the complement of the function), an inverting full adder receiving complementary inputs produces true outputs. Figure (b) shows a carry ripple  adder built from inverting full adders. Every other stage operates on complementary data. The delay inverting the adder inputs or sum outputs is off the critical ripple-carry path.

***Carry Generation and Propagation***

This section introduces notation commonly used in describing faster adders. Recall that the P (propagate) and G ( generate) signals were defined. We can generalize these signals to describe whether a group spanning bits i…j, inclusive, generate a carry or propagate a carry. A group of bits generates a carry if its carry-out is true independent of the carry in; it propagates a carry if its carry-out is true when there is a carry-in. These signals can be defined recursively for i  k > j as ,

$$G_{i:j} = G_{i:k} + P_{i:k} \cdot G_{k-1:j}$$
$$P_{i:j} = P_{i:k} \cdot P_{k-1:j}$$

with the base case,

$$G_{i:i} \equiv G_i = A_i \cdot B_i$$
$$P_{i:i} \equiv P_i = A_i \oplus B_i$$

In other words, a group generates a carry if the upper (more significant) or the lower portion generates and the upper portion propagates that carry. The group propagates a carry if both the upper and lower portions propagate the carry.2 The carry-in must be treated specially. Let us define C0 = Cin and CN = Cout. Then we can define generate and propagate signals for bit 0 as,

$$G_{0:0} = C_{\text{in}}$$
$$P_{0:0} = 0$$

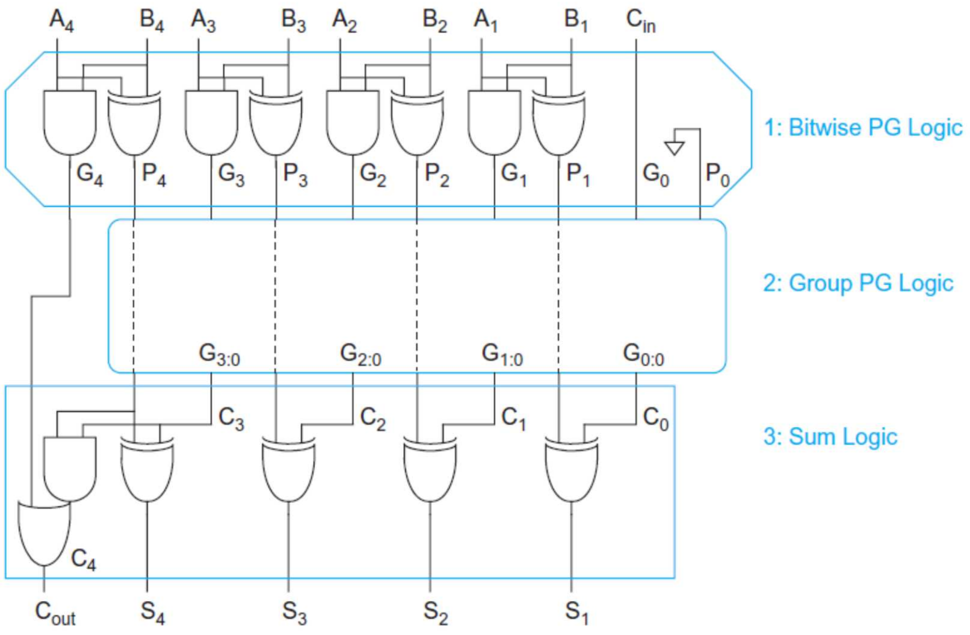
Observe that the carry into bit i is the carry-out of bit i–1 and is Ci–1 = Gi–1:0. This is an important relationship; group generate signals and carries will be used synonymously in the subsequent sections. We can thus compute the sum for bit i using EQ (11.2) as,

$$S_i = P_i \oplus G_{i-1:0}$$

Hence, addition can be reduced to a three-step process:

1. Computing bitwise generate and propagate signals using EQs (11.5) and (11.6)
2. Combining PG signals to determine group generates Gi–1:0 for all N > i > 1 using EQ (11.4)
3. Calculating the sums using EQ (11.7)

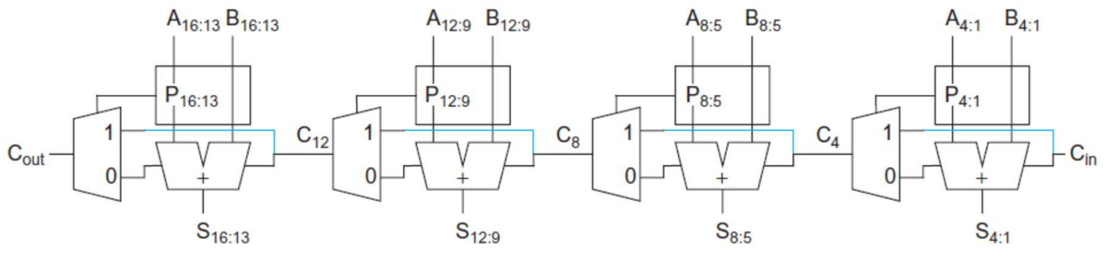
These steps are illustrated in Figure 11.12. The first and third steps are routine, so most of the attention in the remainder of this section is devoted to alternatives for the group PG logic with different trade-offs between speed, area, and complexity. Some of the hardware can be shared in the bitwise PG logic, as shown in Figure.



***Carry-Skip Adder***

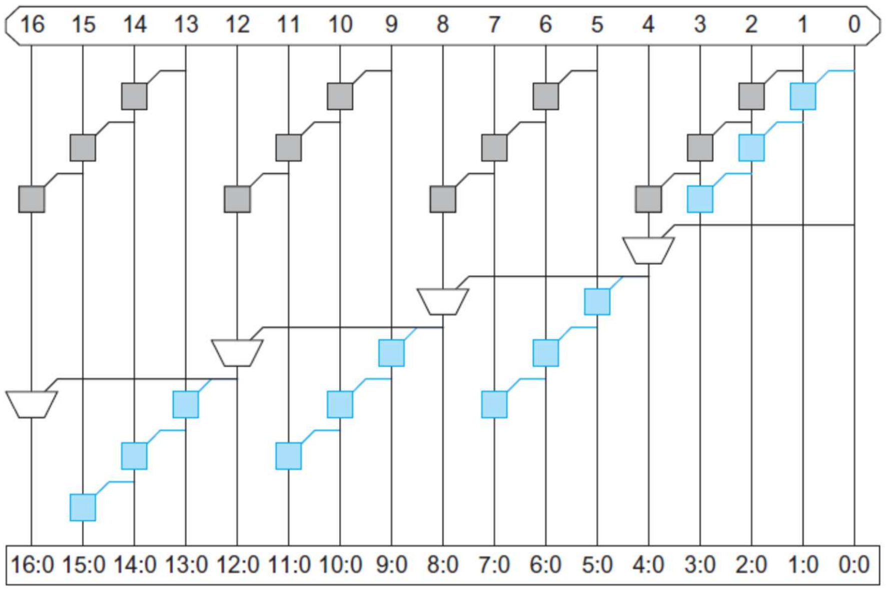
The critical path of CPAs considered so far involves a gate or transistor for each bit of the adder, which can be slow for large adders. The carry-skip (also called carry-bypass) adder, first proposed by Charles Babbage in the nineteenth century and used for many years in mechanical calculators, shortens the critical path by computing the group propagate signals for each carry chain and using this to skip over long carry ripples. Figure shows a carry skip adder built from 4-bit groups. The rectangles compute the bitwise propagate and generate signals (as in Figure, and also contain a 4-input AND gate for the propagate signal of the 4-bit group. The skip multiplexer selects the group carry-in if the group propagate is true or the ripple adder carry-out otherwise.

The critical path through Figure begins with generating a carry from bit 1, and then propagating it through the remainder of the adder. The carry must ripple through the next three bits, but then may skip across the next two 4-bit blocks. Finally, it must ripple through the final 4-bit block to produce the sums. This is illustrated in Figure. The 4-bit ripple chains at the top of the diagram determine if each group generates a carry. The carry skip chain in the middle of the diagram skips across 4-bit blocks. Finally, the 4-bit ripple chains with the blue lines represent the same adders that can produce a carry-out when a carry-in is bypassed to them. Note that the final AND-OR and column 16 are not strictly necessary because Cout can be computed in parallel with the sum XORs using EQ (11.11).



The critical path of the adder from Figures involves the initial PG logic producing a carry out of bit 1, three AND-OR gates rippling it to bit 4, three multiplexers bypassing it to C12, 3 AND-OR gates rippling through bit 15, and a final XOR to produce S16. The multiplexer is an AND22-OR function, so it is slightly slower than the AND-OR function. In general, an N-bit carry-skip adder using k n-bit groups (N = n × k) has a delay of

$$t_{\text{skip}} = t_{pg} + 2(n-1)t_{AO} + (k-1)t_{\text{mux}} + t_{\text{xor}}$$

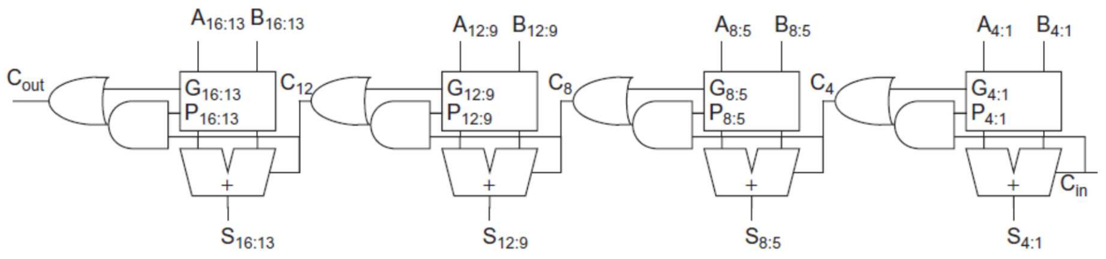


***Carry-Lookahead Adder***

The carry-lookahead adder (CLA) is similar to the carry-skip adder, but computes group generate signals as well as group propagate signals to avoid waiting for a ripple to determine if the first group generates a carry. Such an adder is shown in Figure 11.21 and its PG network is shown in Figure using valency-4 black cells to compute 4-bit group PG signals. In general, a CLA using k groups of n bits each has a delay of

$$t_{\text{cla}} = t_{pg} + t_{pg(n)} + \left[\left(n-1\right)+\left(k-1\right)\right] t_{AO} + t_{\text{xor}}$$

where tpg(n) is the delay of the AND-OR-AND-OR-…-AND-OR gate computing the valency-n generate signal. This is no better than the variable-length carry-skip adder in Figure 11.19 and requires the extra n-bit generate gate, so the simple CLA is seldom a good design choice. However, it forms the basis for understanding faster adders presented in the subsequent sections.

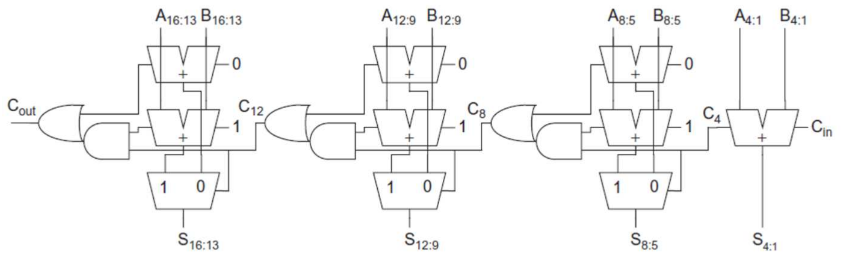


CLAs often use higher-valency cells to reduce the delay of the n-bit additions by computing the carries in parallel. Figure shows such a CLA in which the 4-bit adders are built using Manchester carry chains or multiple static gates operating in parallel.

***Carry-Select Adders***

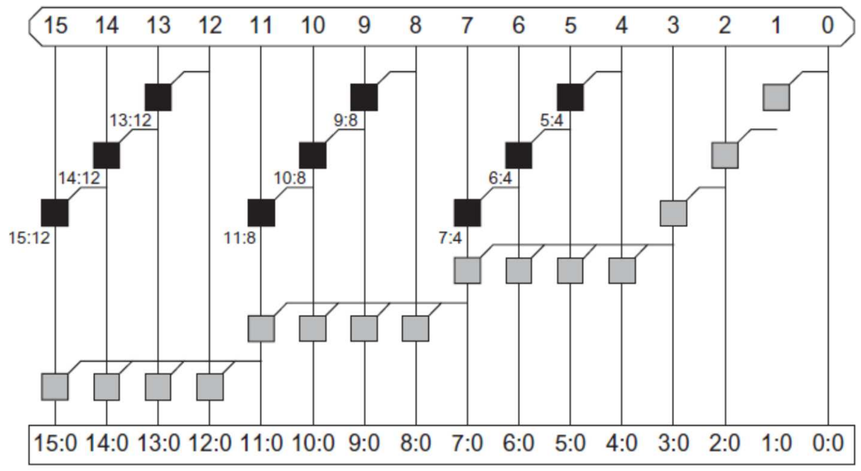
The critical path of the carry-skip and carry-lookahead adders involves calculating the carry into each n-bit group, and then calculating the sums for each bit within the group based on the carry-in. A standard logic design technique to accelerate the critical path is to precompute the outputs for both possible inputs, and then use a multiplexer to select between the two output choices. The carry-select adder shown in Figure  does this with a pair of n-bit adders in each group. One adder calculates the sums assuming a carry-in of 0 while the other calculates the sums assuming a carry-in of 1. The actual carry triggers a multiplexer that chooses the appropriate sum. The critical path delay is

$$t_{\text{select}} = t_{pg} + \left[n + (k-2)\right] t_{AO} + t_{\text{mux}}$$



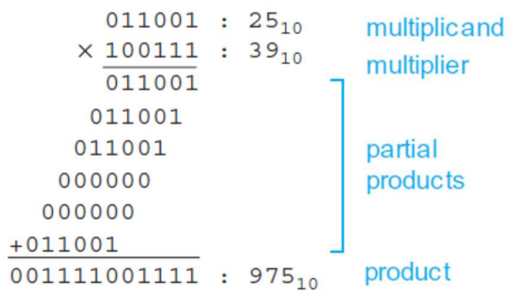
The two n-bit adders are redundant in that both contain the initial PG logic and final sum XOR. It reduces the size by factoring out the common logic and simplifying the multiplexer to a gray cell, as shown in Figure. This is sometimes called a carry increment adder. It uses a short ripple chain of black cells to compute the PG signals for bits within a group. The bits spanned by each group are annotated on the diagram. When the carry-out from the previous group becomes available, the final gray cells in each column determine the carry-out, which is true if the group generates a carry or if the group propagates a carry and the previous group generated a carry. The carry-increment adder has about twice as many cells in the PG network as a carry-ripple adder. The critical path delay is about the same as that of a carry-select adder because a mux and XOR are comparable, but the area is smaller.

$$t_{\text{increment}} = t_{pg} + \left[(n-1) + (k-1)\right] t_{AO} + t_{\text{xor}}$$



## *MULTIPLIERS*

Multiplication is less common than addition, but is still essential for microprocessors, digital signal processors, and graphics engines. The most basic form of multiplication consists of forming the product of two unsigned (positive) binary numbers. This can be accomplished through the traditional technique taught in primary school, simplified to base 2. For example, the multiplication of two positive 6-bit binary integers, 2510 and 3910, proceeds as shown in Figure.



M × N-bit multiplication P = Y × X can be viewed as forming N partial products of M bits each, and then summing the appropriately shifted partial products to produce an M+ N-bit result P. Binary multiplication is equivalent to a logical AND operation. Therefore, generating partial products consists of the logical ANDing of the appropriate bits of the multiplier and multiplicand. Each column of partial products must then be added and, if necessary, any carry values passed to the next column. We denote the multiplicand as,

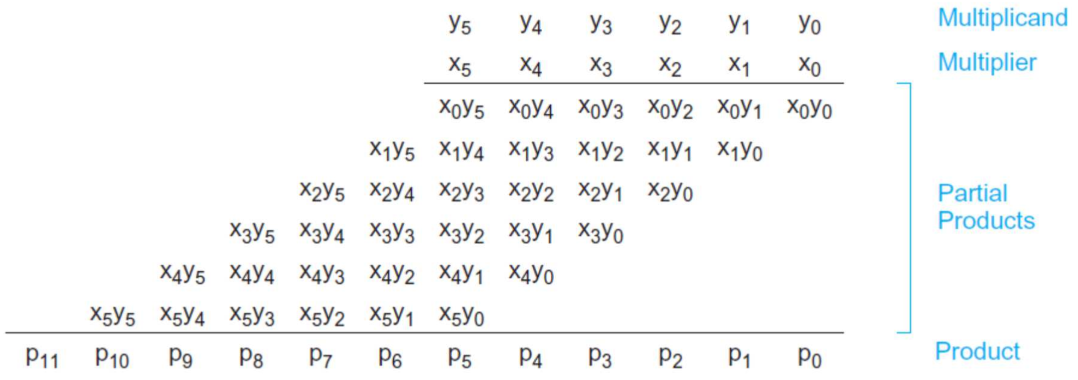
Y = {yM–1, yM–2, …, y1, y0}

and the multiplier as,

X = {xN–1, xN–2, …, x1, x0}.

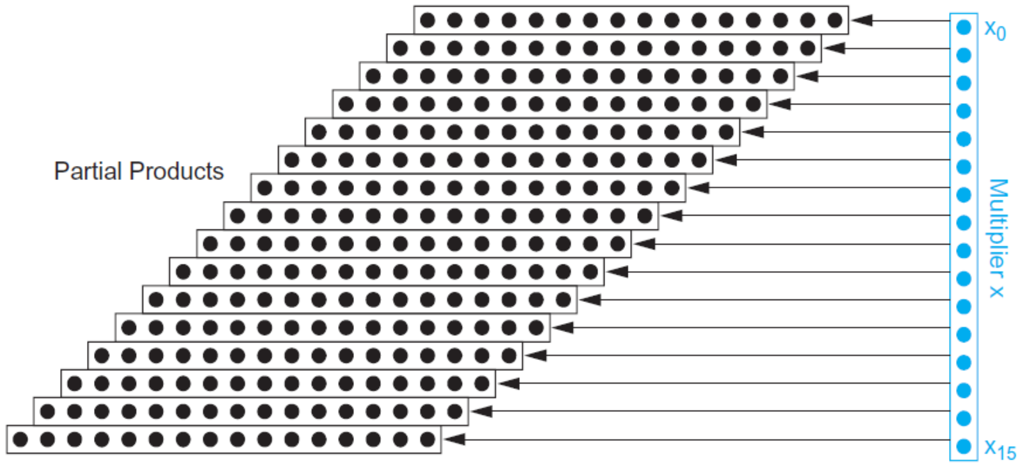
For unsigned multiplication, the product is given in EQ (11.31). Figure 11.72 illustrates the generation, shifting, and summing of partial products in a 6 × 6-bit multiplier.

$$P = \left(\sum_{j=0}^{M-1} y_j 2^j\right)\left(\sum_{i=0}^{N-1} x_i 2^i\right) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} x_i y_j 2^{i+j}$$



$y_5$ $y_4$ $y_3$ $y_2$ $y_1$ $y_0$
$x_5$ $x_4$ $x_3$ $x_2$ $x_1$ $x_0$
$x_0y_5$ $x_0y_4$ $x_0y_3$ $x_0y_2$ $x_0y_1$ $x_0y_0$
$x_1y_5$ $x_1y_4$ $x_1y_3$ $x_1y_2$ $x_1y_1$ $x_1y_0$
$x_2y_5$ $x_2y_4$ $x_2y_3$ $x_2y_2$ $x_2y_1$ $x_2y_0$
$x_3y_5$ $x_3y_4$ $x_3y_3$ $x_3y_2$ $x_3y_1$ $x_3y_0$
$x_4y_5$ $x_4y_4$ $x_4y_3$ $x_4y_2$ $x_4y_1$ $x_4y_0$
$x_5y_5$ $x_5y_4$ $x_5y_3$ $x_5y_2$ $x_5y_1$ $x_5y_0$
$p_{11}$ $p_{10}$ $p_9$ $p_8$ $p_7$ $p_6$ $p_5$ $p_4$ $p_3$ $p_2$ $p_1$ $p_0$
Multiplicand
Multiplier
Partial Products
Product

Large multiplications can be more conveniently illustrated using dot diagrams. Figure shows a dot diagram for a simple 16 × 16 multiplier. Each dot represents a placeholder for a single bit that can be a 0 or 1. The partial products are represented by a horizontal boxed row of dots, shifted according to their weight. The multiplier bits used to generate the partial products are shown on the right.



There are a number of techniques that can be used to perform multiplication. In general, the choice is based upon factors such as latency, throughput, energy, area, and design complexity. An obvious approach is to use an M + 1-bit carry-propagate adder (CPA) to add the first two partial products, then another CPA to add the third partial product to the running sum, and so forth. Such an approach requires N – 1 CPAs and is slow, even if a fast CPA is employed. More efficient parallel approaches use some sort of array or tree of full adders to sum the partial products. We begin with a simple array for unsigned multipliers, and then modify the array to handle signed two's complement numbers using the Baugh-Wooley algorithm. The number of partial products to sum can be reduced using Booth encoding and the number of logic levels required to perform the summation can be reduced with Wallace trees. Unfortunately, Wallace trees are complex to lay out and have long, irregular wires, so hybrid array/tree structures may be more attractive. For completeness, we consider a serial multiplier architecture. This was once popular when gates were relatively expensive, but is now rarely necessary.

***Array Multiplier***

Fast multipliers use carry-save adders (CSAs, see Section 11.2.4) to sum the partial products. A CSA typically has a delay of 1.5–2 FO4 inverters independent of the width of the partial product, while a carry-propagate adder (CPA) tends to have a delay of 4–15+ FO4 inverters depending on the width, architecture, and circuit family.

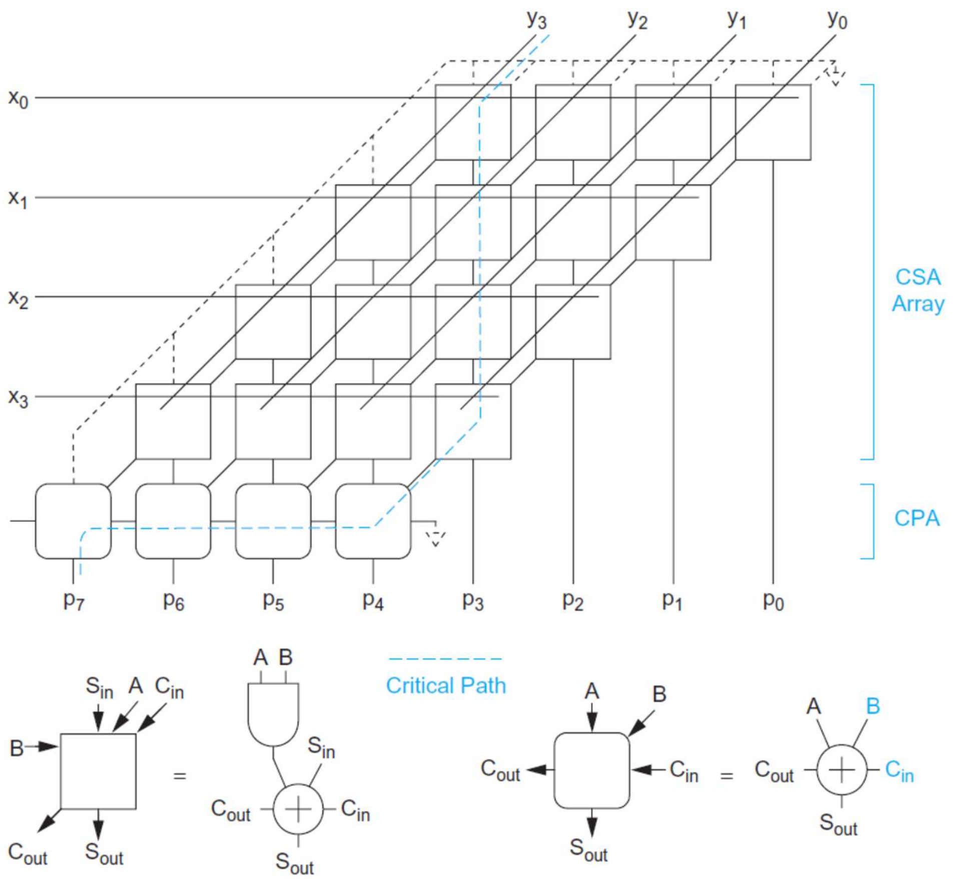
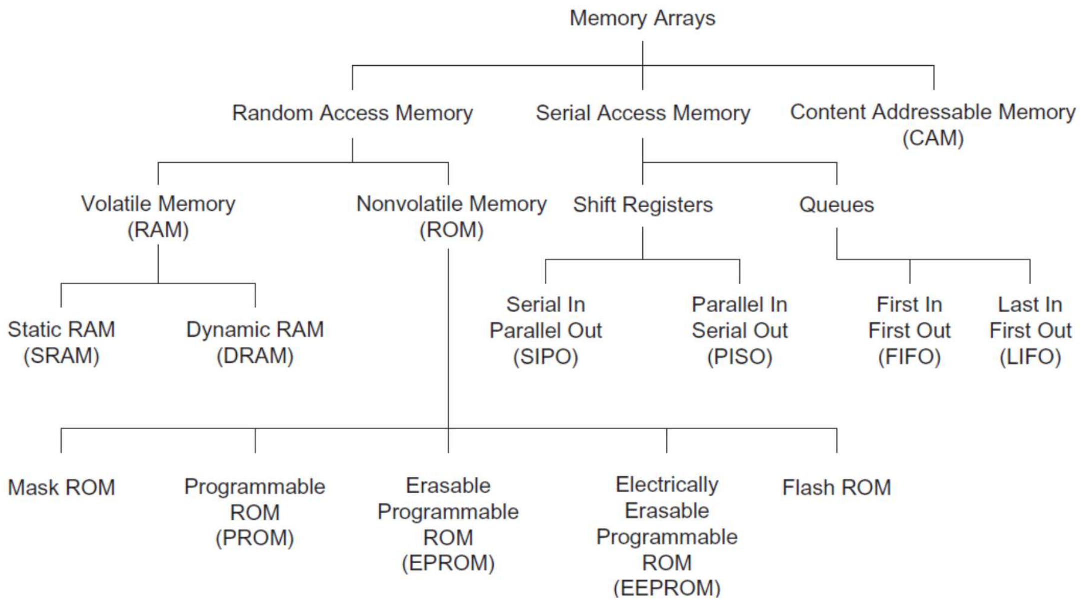


Figure shows a 4 × 4 array multiplier for unsigned numbers using an array of CSAs. Each cell contains a 2-input AND gate that forms a partial product and a full adder (CSA) to add the partial product into the running sum. The first row converts the first partial product into carry-save redundant form. Each later row uses the CSA to add the corresponding partial product to the carry-save redundant result of the previous row and generate a carry-save redundant result. The least significant N output bits are available as sum outputs directly from CSAs. The most significant output bits arrive in carry-save redundant form and require an M-bit carry-propagate adder to convert into regular binary form. In Figure, the CPA is implemented as a carry-ripple adder. The array is regular in structure and uses a single type of cell, so it is easy to design and lay- out. Assuming the carry output is faster than the sum output in a CSA, the critical path through the array is marked on the figure with a dashed line. The adder can easily be pipelined with the placement of registers between rows. In practice, circuits are assigned rectangular blocks in the floorplan so the parallelogram shape wastes space. Figure shows the same adder squashed to fit a rectangular block.

A key element of the design is a compact CSA. This not only benefits area but also helps performance because it leads to short wires with low wire capacitance. An ideal CSA design has approximately equal sum and carry delays because the greater of these two delays limits performance. The mirror adder from Figure is commonly used for its compact layout even though the sum delay exceeds the carry delay. The sum output can be connected to the faster carry input to partially compensate. Note that the first row of CSAs adds the first partial product to a pair of 0s. This leads to a regular structure, but is inefficient. At a slight cost to regularity, the first row of CSAs can be used to add the first three partial products together. This reduces the number of rows by two and correspondingly reduces the adder propagation delay. Yet another way to improve the multiplier array performance is to replace the bottom row with a faster CPA such as a lookahead or tree adder. In summary, the critical path of an array multiplier involves N–2 CSAs and a CPA.

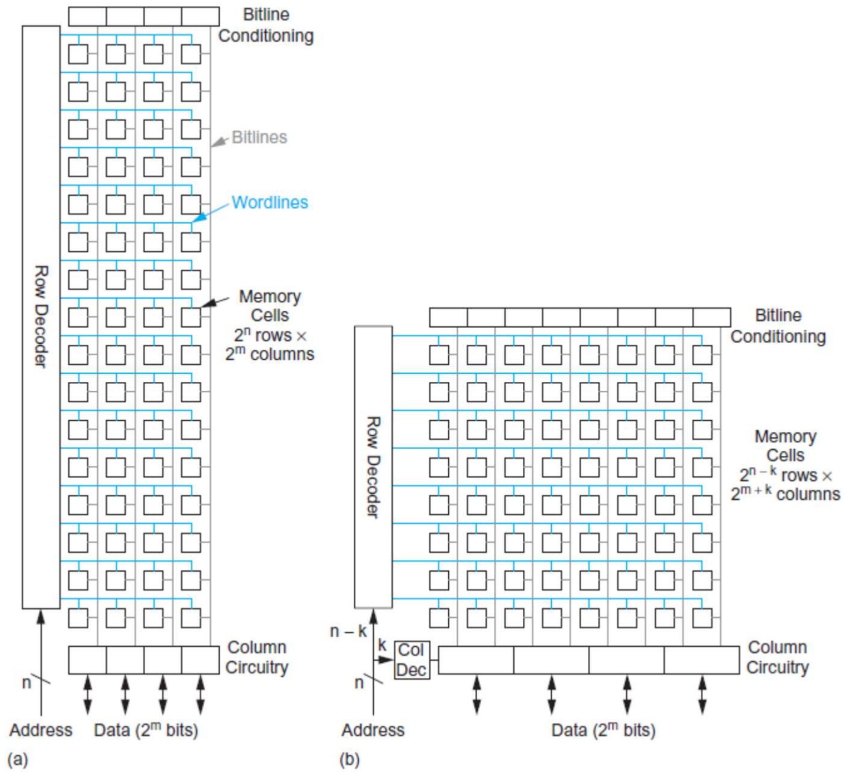
## 2. MEMORIES

Memory arrays often account for the majority of transistors in a CMOS system-on-chip. Arrays may be divided into categories as shown in Figure. Programmable Logic Arrays (PLAs) perform logic rather than storage functions. Random access memory is accessed with an address and has a latency independent of the address. In contrast, serial access memories are accessed sequentially so no address is necessary. Content addressable memories determine which address(es) contain data that matches a specified key.



Random access memory is commonly classified as read-only memory (ROM) or read/write memory (confusingly called RAM). Even the term ROM is misleading because many ROMs can be written as well. A more useful classification is volatile vs. non-volatile memory. Volatile memory retains its data as long as power is applied, while non-volatile memory will hold data indefinitely. RAM is synonymous with volatile memory, while ROM is synonymous with non-volatile memory.

Like sequencing elements, the memory cells used in volatile memories can further be divided into static structures and dynamic structures. Static cells use some form of feedback to maintain their state, while dynamic cells use charge stored on a floating capacitor through an access transistor. Charge will leak away through the access transistor even while the transistor is OFF, so dynamic cells must be periodically read and rewritten to refresh their state. Static RAMs (SRAMs) are faster and less troublesome, but require more area per bit than their dynamic counterparts (DRAMs).



Some non-volatile memories are indeed read-only. The contents of a mask ROM are hardwired during fabrication and cannot be changed. But many non-volatile memories can be written, albeit more slowly than their volatile counterparts. A programmable ROM (PROM) can be programmed once after fabrication by blowing on-chip fuses with a special high programming voltage. An erasable programmable ROM (EPROM) is programmed by storing charge on a floating gate. It can be erased by exposure to ultraviolet (UV) light for several minutes to knock the charge off the gate. Then the EPROM can be reprogrammed. Electrically erasable programmable ROMs (EEPROMs) are similar, but can be erased in microseconds with on-chip circuitry. Flash memories are a variant of EEPROM that erases entire blocks rather than individual bits. Sharing the erase circuitry across larger blocks reduces the area per bit. Because of their good density and easy in system re-programmability, Flash memories have replaced other non-volatile memories in most modern CMOS systems.

Memory cells can have one or more ports for access. On a read/write memory, each port can be read-only, write-only, or capable of both read and write. A memory array contains 2n words of 2m bits each. Each bit is stored in a memory cell. Figure shows the organization of a small memory array containing 16 4-bit words (n = 4, m = 2). Figure (a) shows the simplest design with one row per word and one column per bit. The row decoder uses the address to activate one of the rows by asserting the wordline. During a read operation, the cells on this wordline drive the bitlines, which may have been conditioned to a known value in advance of the memory access. The column circuitry may contain amplifiers or buffers to sense the data. A typical memory array may have thousands or millions of words of only 8–64 bits each, which would lead to a tall, skinny layout that is hard to fit in the chip floorplan and slow because of the long vertical wires. Therefore, the array is often folded into fewer rows of more columns. After folding, each row of the memory contains 2k words, so the array is physically organized as 2n–k rows of 2m+k columns or bits. Figure 12.2(b) shows a two-way fold (k = 1) with eight rows and eight columns. The column decoder controls a multiplexer in the column circuitry to select 2m bits from the row as the data to access. Larger memories are generally built from multiple smaller subarrays so that the wordlines and bitlines remain reasonably short, fast, and low in power dissipation.

***STATIC RAM***

We begin with SRAM, the most widely used form of on-chip memory. SRAM also illustrates all the issues of cell design, decoding, and column circuitry design. Static RAMs use a memory cell with internal feedback that retains its value as long as power is applied. It has the following attractive properties:

Denser than flip-flops

Compatible with standard CMOS processes

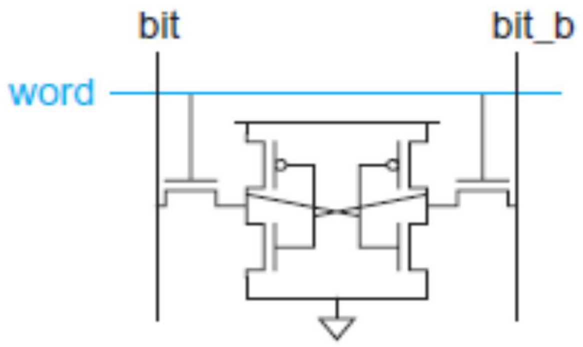
Faster than DRAM

Easier to use than DRAM

For these reasons, SRAMs are widely used in applications from caches to register files to tables to scratchpad buffers. The SRAM consists of an array of memory cells along with the row and column circuitry. This section begins by examining the design and operation of each of these components. It then considers important special cases of SRAMs, including multiported register files, large SRAMs and subthreshold SRAMs.

***SRAM Cells***

A SRAM cell needs to be able to read and write data and to hold the data as long as the power is applied. An ordinary flip-flop could accomplish this requirement, but the size is quite large. Figure shows a standard 6-transistor (6T) SRAM cell that can be an order of magnitude smaller than a flip-flop. The 6T cell achieves its compactness at the expense of more complex peripheral circuitry for reading and writing the cells. This is a good trade-off in large RAM arrays where the memory cells dominate the area. The small cell size also offers shorter wires and hence lower dynamic power consumption.



The 6T SRAM cell contains a pair of weak cross-coupled inverters holding the state and a pair of access transistors to read or write the state. The positive feedback corrects disturbances caused by leakage or noise. The cell is written by driving the desired value and its complement onto the bitlines, bit and bit_b, then raising the wordline, word. The new data overpowers the cross-coupled inverters. It is read by precharging the two bitlines high, then allowing them to float. When word is raised, bit or bit_b pulls down, indicating the data value. The central challenges in SRAM design are minimizing its size and ensuring that the circuitry holding the state is weak enough to be overpowered during a write, yet strong enough not to be disturbed during a read.

SRAM operation is divided into two phases. As described in Section 10.4.6, the phases will be called ϕ1 and ϕ2, but may actually be generated from clk and its complement clkb. Assume that in phase 2, the SRAM is precharged. In phase 1, the SRAM is read or written. Timing diagrams will label the signals as _q1 for qualified clocks (ϕ 1 gated with an enable), _v1 for those that become valid during phase 1, and _s1 for those that remain stable throughout phase 1.

It is no longer common for designers to develop their own SRAM cells. Usually, the fabrication vendor will supply cells that are carefully tuned to the particular manufacturing process. Some processes provide two or more cells with different speed/density trade-offs. Read and write operations and the physical design of the SRAM are as follows.

***Read Operation***

Figure shows a SRAM cell being read. The bitlines are both initially floating high. Without loss of generality, assume Q is initially 0 and thus Q_b is initially 1. Q_b and bit_b both should remain 1. When the wordline is raised, bit should be pulled down through driver and access transistors D1 and A1. At the same time bit is being pulled down, node Q tends to rise. Q is held low by D1, but raised by current flowing in from A1. Hence, the driver D1 must be stronger than the access transistor A1. Specifically, the transistors must be ratioed such that node Q remains below the switching threshold of the P2/D2 inverter. This constraint is called read stability. Waveforms for the read operation are shown in Figure as a 0 is read onto bit. Observe that Q momentarily rises, but does not glitch badly enough to flip the cell.

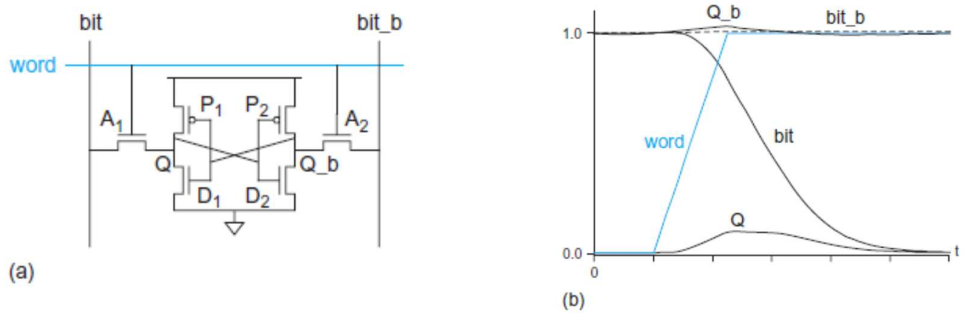
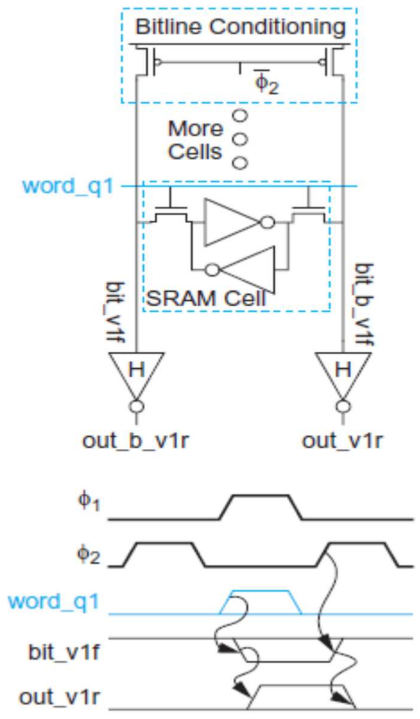


Figure shows the same cell in the context of a full column from the SRAM. During phase 2, the bitlines are precharged high. The wordline only rises during phase 1; hence, it can be viewed as a _q1 qualified clock. Many SRAM cells share the same bitline pair, which acts as a distributed dual-rail footless dynamic multiplexer. The capacitance of the entire bitline must be discharged through the access transistor. The output can be sensed by a pair of HI-skew inverters. By raising the switching threshold of the sense inverters, delay can be reduced at the expense of noise margin. The outputs are dual-rail monotonically rising signals, just as in a domino gate.

***Write Operation***

Figure shows the SRAM cell being written. Again, assume Q is initially 0 and that we wish to write a 1 into the cell. bit is precharged high and left floating. bit_b is pulled low by a write driver. We know on account of the read stability constraint that bit will be unable to force Q high through A1. Hence, the cell must be written by forcing Q_b low through A2. P2 opposes this operation; thus, P2 must be weaker than A2 so that Q_b can be pulled low enough. This constraint is called writability. Once Q_b falls low, D1 turns OFF and P1 turns ON, pulling Q high as desired.



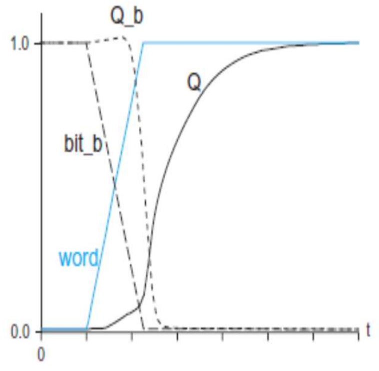
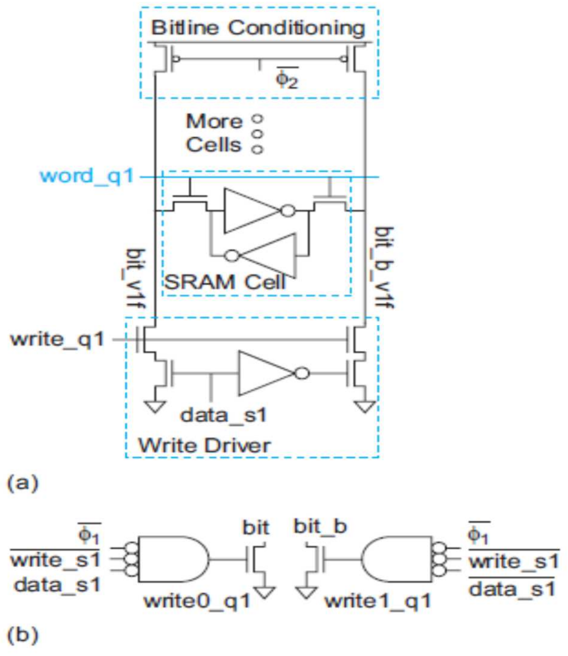


Figure (a) again shows the cell in the context of a full column from the SRAM. During phase 2, the bitlines are precharged high. Write drivers pull the bitline or its complement low during phase 1 to write the cell. The write drivers can consist of a pair of transistors on each bitline for the data and the write enable, or a single transistor driven by the appropriate combination of signals (Figure (b)). In either case, the series resistance of the write driver, bitline wire, and access transistor must be low enough to overpower the pMOS transistor in the SRAM cell. Some arrays use tristate write drivers to improve writability by actively driving one bitline high while the other is pulled low.

***Cell Stability***

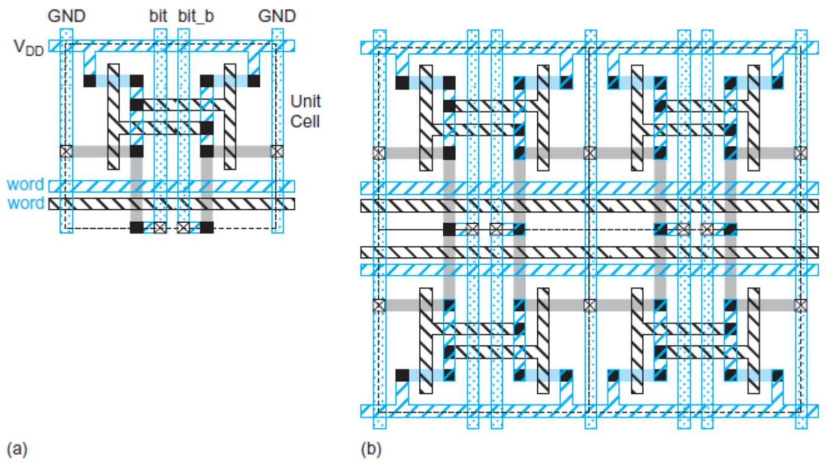
To ensure both read stability and writability, the transistors must satisfy ratio constraints. The nMOS pulldown transistor in the cross-coupled inverters must be strongest. The access transistors are of intermediate strength, and the pMOS pullup transistors must be weak. To achieve good layout density, all of the transistors must be relatively small. For example, the pulldowns could be 8/2 , the access transistors 4/2, and the pullups 3/3. The SRAM cells must operate correctly at all voltages and temperatures despite process variation.

The stability and writability of the cell are quantified by the hold margin, the read margin, and the write margin, which are determined by the static noise margin of the cell in its various modes of operation. A cell should have two stable states during hold and read operation, and only one stable state during write. The static noise margin (SNM) measures how much noise can be applied to the inputs of the two cross-coupled inverters before a stable state is lost (during hold or read) or a second stable state is created (during write).



### *Physical Design*

SRAM cells require clever layout to achieve good density. A traditional design was used until the 90 nm generation, and a lithographically friendly design has been used since. Figure (a) shows a stick diagram of a traditional 6T cell.



The cell is designed to be mirrored and overlapped to share VDD and GND lines between adjacent cells along the cell boundary, as shown in Figure (b). Note how a single diffusion contact to the bit-line is shared between a pair of cells. This halves the diffusion capacitance, and hence reduces the delay discharging the bitline during a read access. The wordline is run in both metal1 and polysilicon; the two layers must occasionally be strapped (e.g., every four or eight cells).

## 3. CONTROL LOGIC IMPLEMENTATION USING PLA's

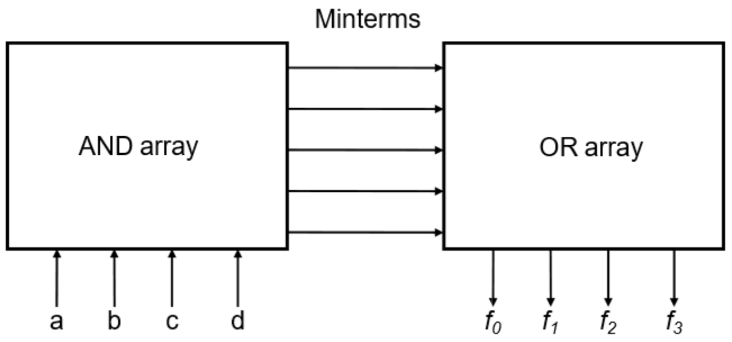
Understand how digital systems may be divided into a data path and control logic. Appreciate the different ways of implementing control logic. FSM design procedure. Draw the state-transition diagram. Check the state diagram. Write state equations. Control logic is a key part of a software program that controls the operations of the program. The control logic responds to commands from the user, and it also acts on its own to perform automated tasks that have been structured into the program. Control logic can be modeled using a state diagram, which is a form of hierarchical state machine. These state diagrams can also be combined with flow charts to provide a set of computational semantics for describing complex control logic.This mix of state diagrams and flow charts is illustrated in the figure on the right, which shows the control logic for a simple stopwatch. The control logic takes in commands from the user, as represented by the event named "START", but also has automatic recurring sample time events, as represented by the event named "TIC".

- Ensure all states are represented, including the IDLE state
- Check that the OR of all transitions leaving a state is TRUE. This is a simple method of determining that there is a way out of a state once entered.
- Verify that the pairwise XOR of all exit transitions is TRUE. This ensures that there are not conflicting conditions that would lead to more than one exit- transition becoming active at any time.
- Insert loops into any state if it is not guaranteed to otherwise change on each cycle.
- Formal FSM verification method
- Perform conformance checking

Two states are definitely equivalent if they have the same outputs for every possible input combination. They have the same next state for every possible input combination (assuming they themselves are equivalent).

## Control – *PLA*

### Structure of a PLA



A PLArepresents an expression of sum-of-product (SOP)

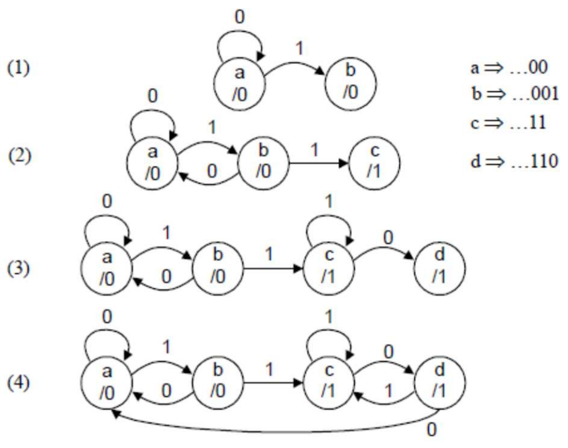
$$f_i = \sum_i m_i(a,b,c,d)$$

$$f_1 = \bar{a}\cdot\bar{b}\cdot c\cdot\bar{d} + \bar{a}\cdot b\cdot c\cdot\bar{d} + a\cdot b\cdot c\cdot d$$

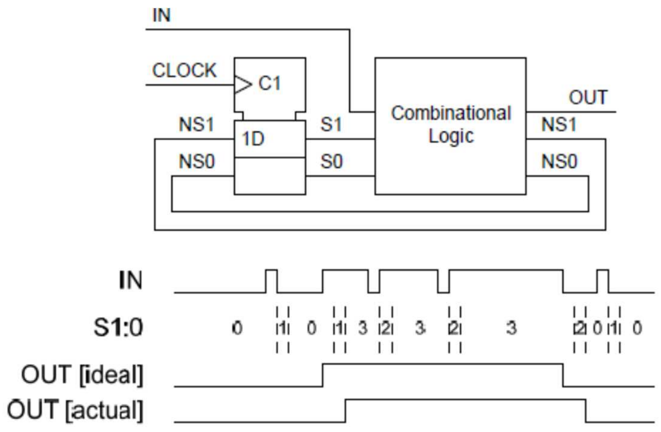
***General Design Procedure***

Construct a sequence of input waveforms that includes all relevant situations. Go through the sequence from the beginning. Each time an input changes, you must decide:

- branch back to a previous state if the current situation is materially identical to a previous one
- create a new state otherwise for each state you must ensure that you have specified
- which state to branch to for every possible input pattern
- what signals to output for every possible input pattern



Use letters a, b,… to label states; we choose numbers later. • Decide what action to take in each state for each of the possible input conditions. Use a Moore machine (i.e. output is constant in each state). Easier to design but needs more states & adds output delay. Assume initially in state "a" and IN has been low for ages.



If IN goes high for two (or more) clock cycles then OUT must go high, whereas if it goes high for only one clock cycle then OUT stays low. It follows that the two histories "IN low for ages" and "IN low for ages then high for one clock" are different because if IN is high for the next clock we need different outputs. Hence we need to introduce state b. If IN goes high for one clock and then goes low again, we canforget it ever changed at all. This glitch on IN will not affect any of our future actions and so we can just return to state a. If on the other hand we are in state b and IN stays high for a second clock cycle, then the output must change. It follows that we need a new state, c. The need for state d is exactly the same as for state b earlier. We reach state d at the end of an output pulse when IN has returned low for one clock cycle. We don't change OUT yet because it might be a false alarm. If we are in state d and IN remains low for a second clock cycle, then it really is the end of the pulse and OUT must go low. We can forget the pulse ever existed and just return to state a.

## 4. TESTING OF VLSI CIRCUITS

### *Need for testing*

Physical defects are likely in manufacturing

– Missing connections (opens)

– Bridged connections (shorts)

– Imperfect doping, processing steps

– Packaging

Yields are generally low

– Yield = Fraction of good die per wafer

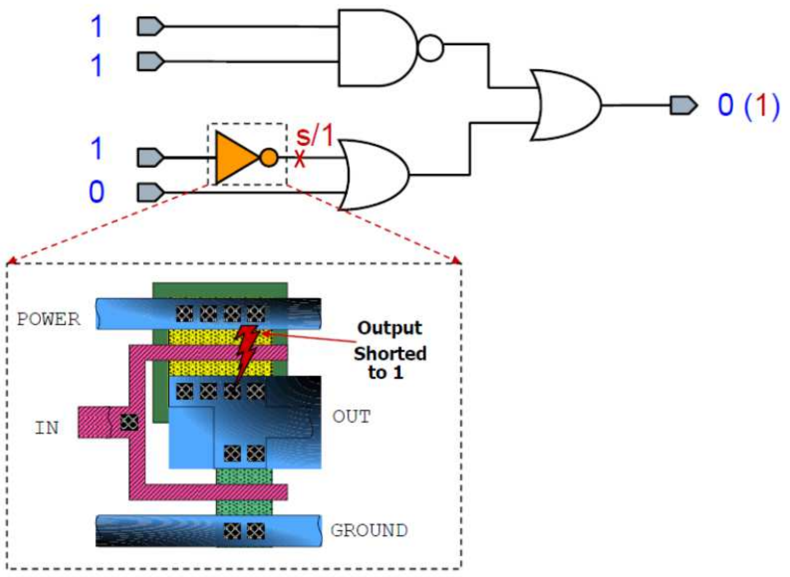
Need to weed out bad die before assembly

Need to test during operation

– Electromagnetic interference, mechanical stress, electromigration, alpha particles

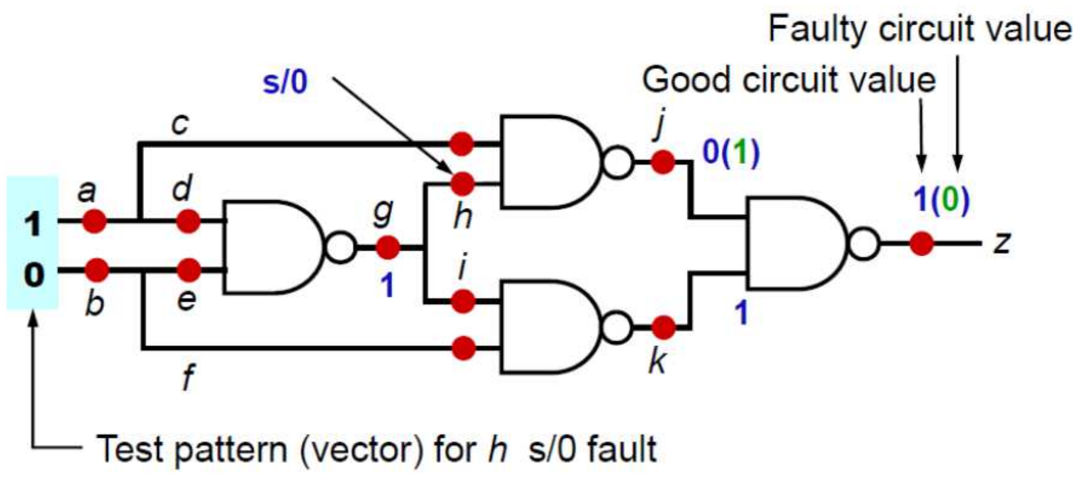
***Fault Models***

To deal with the existence of good and bad parts, it is necessary to propose a fault model; i.e., a model for how faults occur and their impact on circuits. The most popular model is called the Stuck-At model. The Short Circuit/ Open Circuit model can be a closer fit to reality, but is harder to incorporate into logic simulation tools.



Stuck-At Faults In the Stuck-At model, a faulty gate input is modeled as a stuck at zero (Stuck-At-0, S-A- 0) or stuck at one (Stuck-At-l, S-A-l). This model dates from board-level designs, where it was determined to be adequate for modeling faults. Figure 15.11 illustrates how an S-A-0 or S-A-1 fault might occur. These faults most frequently occur due to gate oxide shorts (the nMOS gate to GND or the pMOS gate to VDD) or metal-to-metal shorts.

Short-Circuit and Open-Circuit Faults Other models inc lude stuck-open or shor ted models. Two bridging or shorted faults are shown in Figure. The short S1 results in an S-A-0 fault at input A, while short S2 modifies the function of the gate.



It is evident that to ensure the most accurate modeling, faults should be modeled at the transistor level because it is only at this level that the complete circuit structure is known. For instance, in the case of a simple NAND gate, the intermediate node between the series nMOS transistors is hidden by the schematic. This implies that test generation should ideally take account of possible shorts and open circuits at the switch level. Expediency dictates that most existing systems rely on Boolean logic representations of circuits and stuck-at fault modeling.

***Observability***

The observability of a particular circuit node is the degree to which you can observe that node at the outputs of an integrated circuit (i.e., the pins). This metric is relevant when you want to measure the output of a gate within a larger circuit to check that it operates correctly. Given the limited number of nodes that can be directly observed, it is the aim of good chip designers to have easily observed gate outputs. Adoption of some basic design for test techniques can aid tremendously in this respect. Ideally, you should be able to observe directly or with moderate indirection (i.e., you may have to wait a few cycles) every gate output within an integrated circuit. While at one time this aim was hindered by the expense of extra test circuitry and a lack of design methodology, current processes and design practices allow you to approach this ideal..

***Controllability***

The controllability of an internal circuit node within a chip is a measure of the ease of setting the node to a 1 or 0 state. This metric is of importance when assessing the degree of difficulty of testing a particular signal within a circuit. An easily controllable node would be directly settable via an input pad. A node with little controllability, such as the most significant bit of a counter, might require many hundreds or thousands of cycles to get it to the right state. Often, you will find it impossible to generate a test sequence to set a number of poorly controllable nodes into the right state. It should be the aim of good chip designers to make all nodes easily controllable. In common with observability, the adoption of some simple design for test techniques can aid in this respect tremendously. Making all flip-flops resettable via a global reset signal is one step toward good controllability.

### *Repeatability*

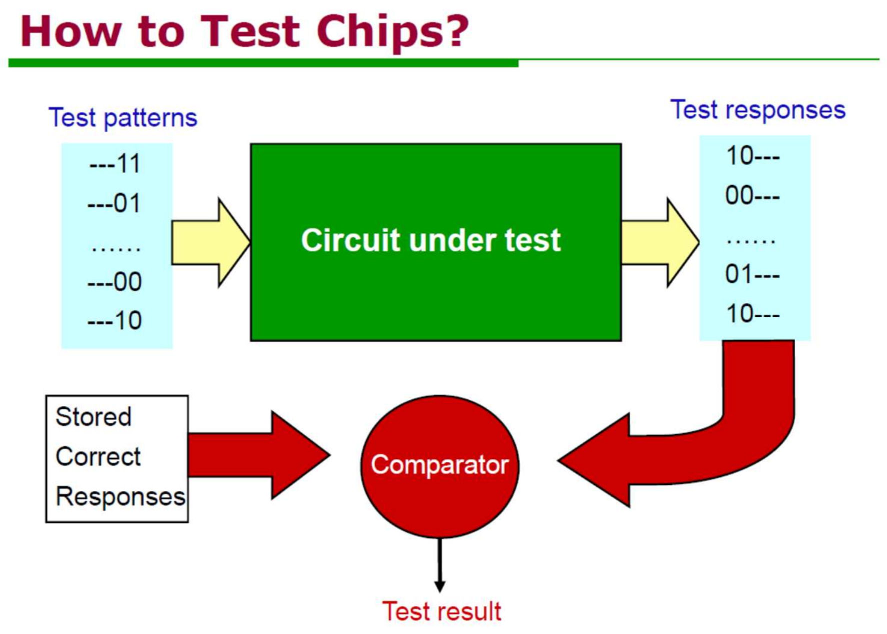
The repeatability of system is the ability to produce the same outputs given the same inputs. Combinational logic and synchronous sequential logic is always repeatable when it is functioning correctly. However, certain asynchronous sequential circuits are nondeterministic. For example, an arbiter may select either input when both arrive at nearly the same time. Testing is much easier when the system is repeatable. Some systems with asynchronous interfaces have a lock-step mode to facilitate repeatable testing.

### *Survivability*

The survivability of a system is the ability to continue function after a fault. For example, error-correcting codes provide survivability in the event of soft errors. Redundant rows and columns in memories and spare cores provide survivability in the event of manufacturing defects. Adaptive techniques provide survivability in the event of process variation. Some survivability features are invoked automatically by the hardware, while others are activated by blowing fuses after manufacturing test.

### *Fault Coverage*

A measure of goodness of a set of test vectors is the amount of fault coverage it achieves. That is, for the vectors applied, what percentage of the chip's internal nodes were checked? Conceptually, the way in which the fault coverage is calculated is as follows. Each circuit node is taken in sequence and held to 0 (S-A-0), and the circuit is simulated with the test vectors comparing the chip outputs with a known good machine—a circuit with no nodes artificially set to 0 (or 1). When a discrepancy is detected between the faulty machine and the good machine, the fault is marked as detected and the simulation is stopped. This is repeated for setting the node to 1 (S-A-1). In turn, every node is stuck (artificially) at 1 and 0 sequentially. The fault coverage of a set of test vectors is the percentage of the total nodes that can be detected as faulty when the vectors are applied. To achieve world-class quality levels, circuits are required to have in excess of 98.5% fault coverage. The Verification Methodology Manual [Bergeron05] is the bible for fault coverage techniques.



Iddq testing is a method for testing CMOS integrated circuits for the presence of manufacturing faults. It relies on measuring the supply current (Idd) in the quiescent state (when the circuit is not switching and inputs are held at static values). The current consumed in the state is commonly called Iddq for Idd (quiescent) and hence the name. Iddq testing uses the principle that in a correctly operating quiescent CMOS digital circuit, there is no static current path between the power supply and ground, except for a small amount of leakage. Many common semiconductor manufacturing faults will cause the current to increase by orders of magnitude, which can be easily detected. This has the advantage of checking the chip for many possible faults with one measurement. Another advantage is that it may catch faults that are not found by conventional stuck-at fault test vectors. Iddq testing is somewhat more complex than just measuring the supply current. If a line is shorted to Vdd, for example, it will still draw no extra current if the gate driving the signal is attempting to set it to '1'. However, a different vector set that attempts to set the signal to 0 will show a large increase in quiescent current, signalling a bad part. Typical Iddq test vector sets may have 20 or so vectors. Note that Iddq test vectors require only controllability, and not observability. This is because the observability is through the shared power supply connection. Iddq testing has many advantages:

- It is a simple and direct test that can identify physical defects.

 The area and design time overhead are very low.

 Test generation is fast.

 Test application time is fast since the vector sets are small.

 It catches some defects that other tests, particularly stuck-at logic tests, do not.

Drawback: Compared to scan testing, Iddq testing is time consuming, and then more expensive, since is achieved by current measurements that take much more time than reading digital pins in mass production.

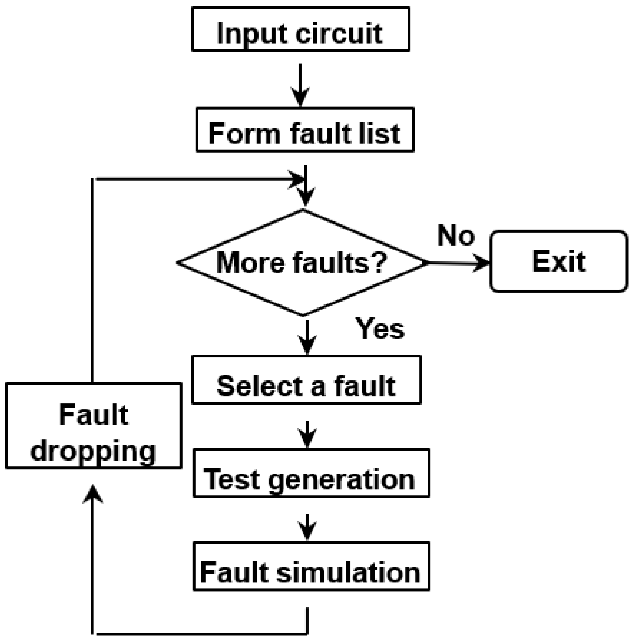
***AUTOMATIC TEST PATTERN GENERATION (ATPG)***

Historically, in the IC industry, logic and circuit designers implemented the functions at the RTL or schematic level, mask designers completed the layout, and test engineers wrote the tests. In many ways, the test engineers were the Sherlock Holmes of the industry, reverse engineering circuits and devising tests that would test the circuits in an adequate manner. For the longest time, test engineers implored circuit designers to include extra circuitry to ease the burden of test generation. Happily, as processes have increased in density and chips have increased in complexity, the inclusion of test circuitry has become less of an overhead for both the designer and the manager worried about the cost of the die. In addition, as tools have improved, more of the burden for generating tests has fallen on the designer. To deal with this burden, Automatic Test Pattern Generation (ATPG) methods have been invented. The use of some form of ATPG is standard for most digital designs.

It is the process of generating test patterns for a given fault model. If we go by exhaustive testing, in the worst case, we may require 2n (where n stands for no. of primary inputs) assignments to be applied for finding test vector for a single stuck-at fault. It is impossible for us to manually use exhaustive testing or path sensitization method to generate a test pattern for chips consisting of millions of transistors. Hence, we need an automated process, a.k.a. Automatic Test Pattern Generation (ATPG).

A cycle of ATPG can generally be divided into two distinct phases: 1) creation of the test; and 2) application of the test.  During the creation of the test, appropriate models for the device circuit are developed at gate or transistor level in such a way that the output responses of a faulty device for a given set of inputs will differ from those of a good device. This generation of test is basically a mathematical process that can be done in three ways: 1) by manual methods; 2) by algorithmic methods (with or without heuristics); and 3) by pseudo-random methods. The software used for complex ATPG applications are quite expensive, but the process of generating a test needs to be done only once at the end of the design process.

When creating a test, the goal should be to make it as efficient in memory space and time requirements as much as possible. As such, the ATPG process must generate the minimum or near minimum set of vectors needed to detect all the important faults of a device. The main considerations for test creation are: 1) the time needed to construct the minimal test set; 2) the size of the pattern generator, or hardware/software system needed to properly stimulate the devices under test; 3) the size of the testing process itself; 4) the time needed to load the test patterns; and 5) the external equipment required (if any).



Examples of ATPG algorithmic methods that are in wide use today include the D-Algorithm, the PODEM, and the FAN. Pattern generation through any of these algorithmic methods require what is known as 'path sensitization.' Path sensitization refers to finding a path in the circuit that will allow an error to show up at an observable output of a device if it is faulty. For example, in a two-input AND gate, sensitizing the path of one input requires the other input to be set to '1'. Most algorithmic generation methods also refer to the notations D and D'. These notations were introduced by the D algorithm and have been adopted by other algorithms since then. D simply stands for a '1' in a good circuit and a '0' in a faulty one. On the other hand, D', which is the opposite of D, stands for a '0' in a good circuit and '1' in a faulty circuit. Thus, propagating a D or D' from the inputs to the output simply means applying a set of inputs to a device to make its output exhibit an 'error' if a fault within the circuit exists.

Algorithmic pattern generation basically consists of the following steps: 1) fault selection, or choosing a fault that needs to be detected; 2) initial assignment, or finding an input pattern that sets up a D or D' at the output of the faulty gate; 3) forward drive, or propagating a D or D' to an observable output using the shortest path possible; 4) justification, or assigning of values to other unassigned inputs in order to justify the assignments made during the forward drive. If an inconsistency arises during justification, backtracking or back propagation is performed, i.e., forward drive is done again using an alternate path. This recursive cycle is performed until the right set of input patterns needed to 'sensitize' a path and propagate the fault to an observable output is determined.

The D algorithm was developed by Roth at IBM in 1966, and was the first 'complete' test pattern algorithm designed to be programmable on a computer. A test algorithm is 'complete' if it is able to propagate a failure to an observable output if a fault indeed exists. As discussed in the previous paragraph, the D algorithm entails finding all sets of inputs to the circuit that will bring out a fault within the circuit. A 'primitive D cube of failure', or PDCF, is a set of inputs that sensitizes a path for a particular fault within a circuit. The 'propagation D cube', or PDC, is a set of inputs that propagates a D from the inputs to the output.

The D algorithm picks all possible PDCF's for the circuit under test and applies them to the circuit with their corresponding PDC's to propagate various faults to the output. While the PDCF's and PDC's are being applied, the 'implied' values for other circuit nodes are tested for consistency, rejecting sets of inputs that cause a circuit violation. The application and testing of various PDCF's and PDC's for a circuit is done repeatedly and recursively, until the minimal set of input patterns necessary to test the circuit for the specified faults is determined.

Commercial ATPG tools can achieve excellent fault coverage. However, they are computation-intensive and often must be run on servers or compute farms with many parallel processors. Some tools use statistical algorithms to predict the fault coverage of a set of vectors without performing as much simulation. Adding scan and built-in self-test, improves the observability of a system and can reduce the number of test vectors required to achieve a desired fault coverage. The algorithmic methods for test generation are examples of 'deterministic' ATPG, since the tests are systematically developed with a definite outcome for the targeted faults.

## ***DESIGN FOR TESTABILITY (DFT)***

Design for Testability The keys to designing circuits that are testable are controllability and observability. Restated, controllability is the ability to set (to 1) and reset (to 0) every node internal to the circuit. Observability is the ability to observe, either directly or indirectly, the state of any node in the circuit. Good observability and controllability reduce the cost of manufacturing testing because they allow high fault coverage with relatively few test vectors. Moreover, they can be essential to silicon debug because physically probing internal signals has become so difficult.

We will first cover three main approaches to what is commonly called Design for Testability (DFT). These may be categorized as follows:

Ad hoc testing

Scan-based approaches

Built-in self-test (BIST)

### ***Ad Hoc Testing***

Ad hoc test techniques, as their name suggests, are collections of ideas aimed at reducing the combinational explosion of testing. They are summarized here for historical reasons. They are only useful for small designs where scan, ATPG, and BIST are not available. A complete scan-based testing methodology is recommended for all digital circuits. Having said that, the following are common techniques for ad hoc testing:

Partitioning large sequential circuits

Adding test points

Adding multiplexers

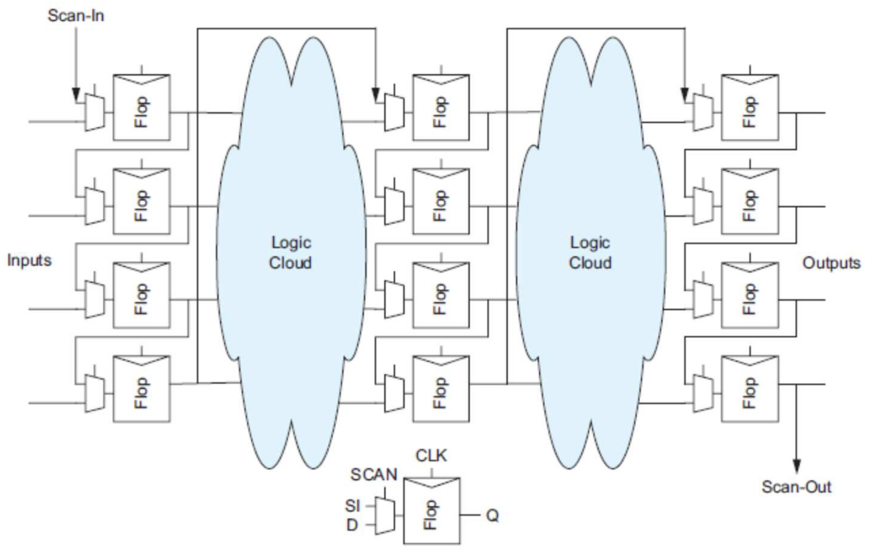
Providing for easy state reset

A technique classified in this category is the use of the bus in a bus-oriented system for test purposes. Each register has been made loadable from the bus and capable of being driven onto the bus. Here, the internal logic values that exist on a data bus are enabled onto the bus for testing purposes. Frequently, multiplexers can be used to provide alternative signal paths during testing. In CMOS, transmission gate multiplexers provide low area and delay overhead. Any design should always have a method of resetting the internal state of the chip within a single cycle or at most a few cycles. Apart from making testing easier, this also makes simulation faster as a few cycles are required to initialize the chip.

In general, ad hoc testing techniques represent a bag of tricks developed over the years by designers to avoid the overhead of a systematic approach to testing, as will be described in the next section. While these general approaches are still quite valid, process densities and chip complexities necessitate a structured approach to testing.

***SCAN BASED TECHNIQUE***

The scan-design strategy for testing has evolved to provide observability and controllability at each register. In designs with scan, the registers operate in one of two modes. In normal mode, they behave as expected. In scan mode, they are connected to form a giant shift register called a scan chain spanning the whole chip. By applying N clock pulses in scan mode, all N bits of state in the system can be shifted out and new N bits of state can be shifted in. Therefore, scan mode gives easy observability and controllability of every register in the system.

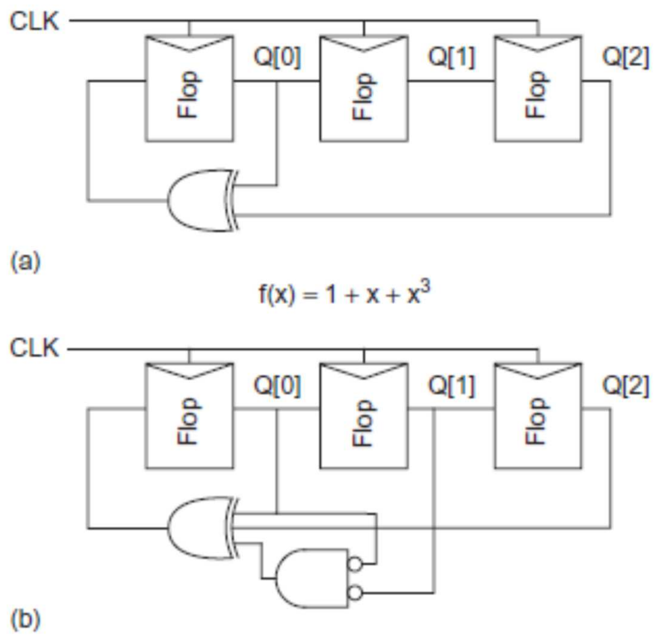


Modern scan is based on the use of scan registers, as shown in Figure. The scan register is a D flip-flop preceded by a multiplexer. When the SCAN signal is deasserted, the register behaves as a conventional register, storing data on the D input. When SCAN is asserted, the data is loaded from the SI pin, which is connected in shift register fashion to the previous register Q output in the scan chain. For the circuit to load the scan chain, SCAN is asserted and CLK is pulsed eight times to load the first two ranks of 4-bit registers with data. SCAN is deasserted and CLK is asserted for one cycle to operate the circuit normally with predefined inputs. SCAN is then reasserted and CLK asserted eight times to read the stored data out. At the same time, the new register contents can be shifted in for the next test. Testing proceeds in this manner of serially clocking the data through the scan register to the right point in the circuit, running a single system clock cycle and serially clocking the data out for observation. In this scheme, every input to the combinational block can be controlled and every output can be observed. In addition, running a random pattern of 1s and 0s through the scan chain can test the chain itself.

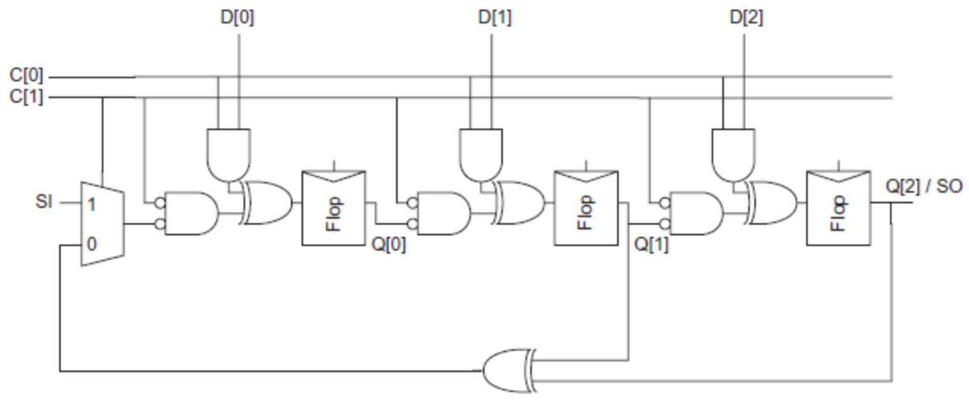
Test generation for this type of test architecture can be highly automated. ATPG techniques can be used for the combinational blocks and, as mentioned, the scan chain is easily tested. The prime disadvantage is the area and delay impact of the extra multiplexer in the scan register. Designers (and managers alike) are in widespread agreement that this cost is more than offset by the savings in debug time and production test cost.

### *SELF-TEST APPROACHES: BUILT-IN SELF-TEST (BIST)*

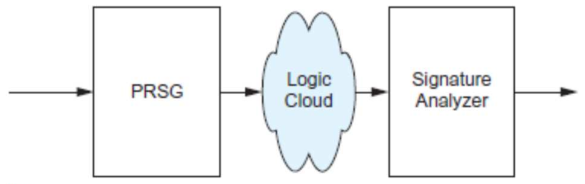
Self-test and built-in test techniques, as their names suggest, rely on augmenting circuits to allow them to perform operations upon themselves that prove correct operation. These techniques add area to the chip for the test logic, but reduce the test time required and thus can lower the overall system cost. [Stroud02] offers extensive coverage of the subject from the implementer's perspective.



One method of testing a module is to use signature analysis or cyclic redundancy checking. This involves using a pseudo-random sequence generator (PRSG) to produce the input signals for a section of combinational circuitry and a signature analyzer to observe the output signals. A PRSG of length n is constructed from a linear feedback shift register (LFSR), which in turn is made of n flip-flops connected in a serial fashion, as shown in Figure (a). The XOR of particular outputs are fed back to the input of the LFSR. An n-bit LFSR will cycle through 2n–1 states before repeating the sequence. LFSRs are discussed further in Section 11.5.4. They are described by a characteristic polynomial indicating which bits are fed back. A complete feedback shift register (CFSR), shown in Figure (b), includes the zero state that may be required in some test situations. An n-bit LFSR is converted to an n-bit CFSR by adding an n – 1 input NOR gate connected to all but the last bit. When in state 0…01, the next state is 0…00. When in state 0…00, the next state is 10…0. Otherwise, the sequence is the same. Alternatively, the bottom n bits of an n + 1-bit LFSR can be used to cycle through the all zeros state without the delay of the NOR gate.



(a)

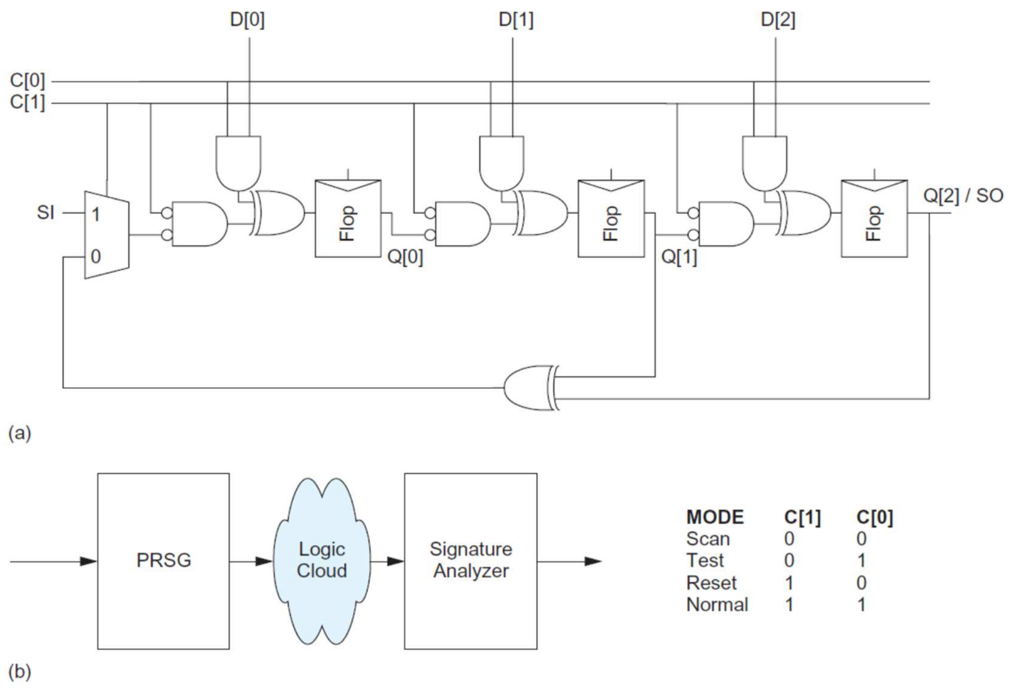


| MODE | C[1] | C[0] |
|---|---|---|
| Scan | 0 | 0 |
| Test | 0 | 1 |
| Reset | 1 | 0 |
| Normal | 1 | 1 |

(b)

A signature analyzer receives successive outputs of a combinational logic block and produces a syndrome that is a function of these outputs. The syndrome is reset to 0, and then XORed with the output on each cycle. The syndrome is swizzled each cycle so that a fault in one bit is unlikely to cancel itself out. At the end of a test sequence, the LFSR contains the syndrome that is a function of all previous outputs. This can be compared with the correct syndrome (derived by running a test program on the good logic) to determine whether the circuit is good or bad. If the syndrome contains enough bits, it is improbable that a defective circuit will produce the correct syndrome.

The combination of signature analysis and the scan technique creates a structure known as BIST—for Built-In Self-Test or BILBO—for Built-In Logic Block Observation [Koenemann79]. The 3-bit BIST register shown in Figure 15.20 is a scannable, resettable register that also can serve as a pattern generator and signature analyzer. C[1:0] specifies the mode of operation. In the reset mode (10), all the flip-flops are synchronously initialized to 0. In normal mode (11), the flip-flops behave normally with their D input and Q output. In scan mode (00), the flip-flops are configured as a 3-bit shift register between SI and SO. Note that there is an inversion between each stage. In test mode (01), the register behaves as a pseudo-random sequence generator or signature analyzer. If all the D inputs are held low, the Q outputs loop through a pseudo-random bit sequence, which can serve as the input to the combinational logic. If the D inputs are taken from the combinational logic output, they are swizzled with the existing state to produce the syndrome. In summary, BIST is performed by first resetting the syndrome in the output register. Then both registers are placed in the test mode to produce the pseudo-random inputs and calculate the syndrome. Finally, the syndrome is shifted out through the scan chain.



| MODE | C[1] | C[0] |
|---|---|---|
| Scan | 0 | 0 |
| Test | 0 | 1 |
| Reset | 1 | 0 |
| Normal | 1 | 1 |

Various companies have commercial design aid packages that automatically replace ordinary registers with scannable BIST registers, check the fault coverage, and generate scripts for production testing. As an example, on a WLAN modem chip comprising roughly 1 million gates, a full at-speed test takes under a second with BIST. This comes with roughly a 7.3% overhead in the core area (but actually zero because the design was pad limited) and a 99.7% fault coverage level. The WLAN modem parts designed in this way were fully tested in less than ten minutes on receipt of first silicon. This kind of test method is incredibly valuable for productivity in manufacturing test generation.

### *Memory BIST*

On many chips, memories account for the majority of the transistors. A robust testing methodology must be applied to provide reliable parts. In a typical MBIST scheme, multiplexers are placed on the address, data, and control inputs for the memory to allow direct access during test. During testing, a state machine uses these multiplexers to directly write a checkerboard pattern of alternating 1s and 0s. The data is read back, checked, then the inverse pattern is also applied and checked. ROM testing is even simpler: The contents are read out to a signature analyzer to produce a syndrome.

## CONCLUSION:

This chapter has presented a range of datapath subsystems. How one goes about designing and implementing a given CMOS chip is largely affected by the availability of tools, the schedule, the complexity of the system, and the final cost goals of the chip. In general, the simplest and least expensive (in terms of time and money) approach that meets the target goals should be chosen. This chapter has summarized the important issues in CMOS chip testing and has provided some methods for incorporating test considerations into chips from the start of the design. Scan is now an indispensable technique to observe and control registers because probing signals directly has become extremely difficult. The importance of writing adequate tests for both the functional verification and manufacturing verification cannot be understated. It is probably the single most important activity in any CMOS chip design cycle and usually takes the longest time no matter what design methodology is used. If one message is left in your mind after reading this chapter, it should be that you are absolutely rigorous about the testing activity surrounding a chip project and it should rank first among any design trade-offs.

# Notes on Programmable Logic Devices (PLDs) in VLSI Design – Unit V

**Dr. G. Senthil Kumar,**
Associate Professor,
Dept. of ECE, SCSVMV,
email: gsk_ece@kanchiuniv.ac.in

==============================================================

**OBJECTIVES**:

In this lesson you will be introduced to some types of Programmable Logic Devices (PLDs):

- PROM, PAL, PLA, CPLDs, FPGAs.
- How to implement digital circuits using these PLDs.

**CONTENTS**:

## 1. INTRODUCTION:

An IC that contains large numbers of gates, flip-flops, etc. that can be configured by the user to perform different functions is called a Programmable Logic Device (PLD). It permits elaborate digital logic designs to be implemented by the user on a single device. The internal logic gates and/or connections of PLDs can be changed/configured by a programming process.

***Comparison: programmable logic Vs fixed logic***

The fixed logic system has circuits whose configurations are permanent. Their instructions perform only a fixed set of operations repeatedly. Once manufactured and programmed, the logic cannot be changed. This system is a fantastic asset for repeated tasks.

But one tiny mistake in the manufacturing process like uploading the wrong code in the device, and the entire system is discarded, and a new design is developed. That's quite some risk that companies aren't willing to take unless necessary. Additionally, fixed logic does not allow the users to expand or build on their existing functionalities.

Thus, we need something more flexible, easy to work, and more cost-efficient. Thus, programmable logic comes to the rescue. It is easy-to-program, affordable and equipped with better features. Inexpensive software is used to develop, code and test the required design. This design is then programmed into a device and tested in a live electronic circuit.

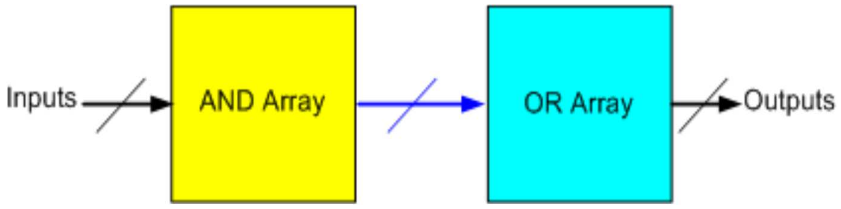
The corresponding performance then decides if the logic needs to be altered, or if the prototype is fit to be determined as the final design itself. The fixed logic system thus offers limited usability; a programmable logic seems more feasible and beneficial.

***Implementing Boolean functions:***

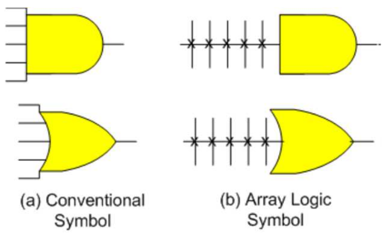
Every Boolean logic can be decomposed into product-of-sum (POS) or sum-of-product by Karnaugh map(k-map),

$$\begin{aligned} S = A \oplus B \oplus C &= \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC \\ &= (A+B+\overline{C})(A+\overline{B}+C)(\overline{A}+B+C)(\overline{A}+\overline{B}+\overline{C}) \end{aligned}$$

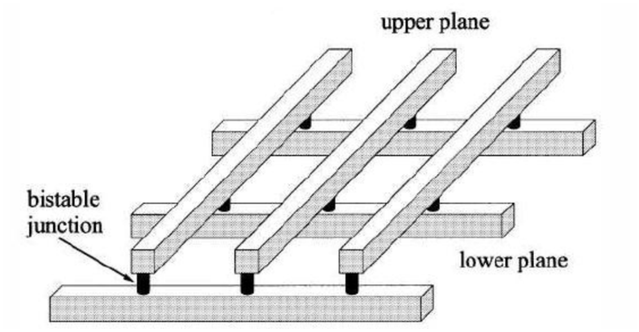
PLDs are typically built with an array of AND gates (AND-array) and an array of OR gates (OR-array) to implement the sum-of-products as shown in figure.



In order to show the internal logic diagram for such technologies in a concise form, it is necessary to have special symbols for array logic. Figure shows the conventional and array logic symbols for a multiple input AND and a multiple input OR gate.



One of the simplest programming technologies is to use fuses. In the original state of the device, all the fuses are intact. Programming the device involves blowing those fuses along the paths that must be removed in order to obtain the particular configuration of the desired logic function. Anti-fuse employs a thin barrier of non-conducting amorphous silicon between two metal conductors. Usually in mesh structure. When a sufficiently high voltage is applied across the amorphous silicon it is turned into a polycrystalline silicon-metal alloy with a low resistance, which is conductive



***Problems of using standard ICs**:* Problems of using standard ICs in logic design are that they require hundreds or thousands of these ICs, considerable amount of circuit board space, a great deal of time and cost in inserting, soldering, and testing. Also require keeping a significant inventory of ICs.

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

***Types of PLDs:***

PLDs are broadly classified into simple and complex programmable logic devices

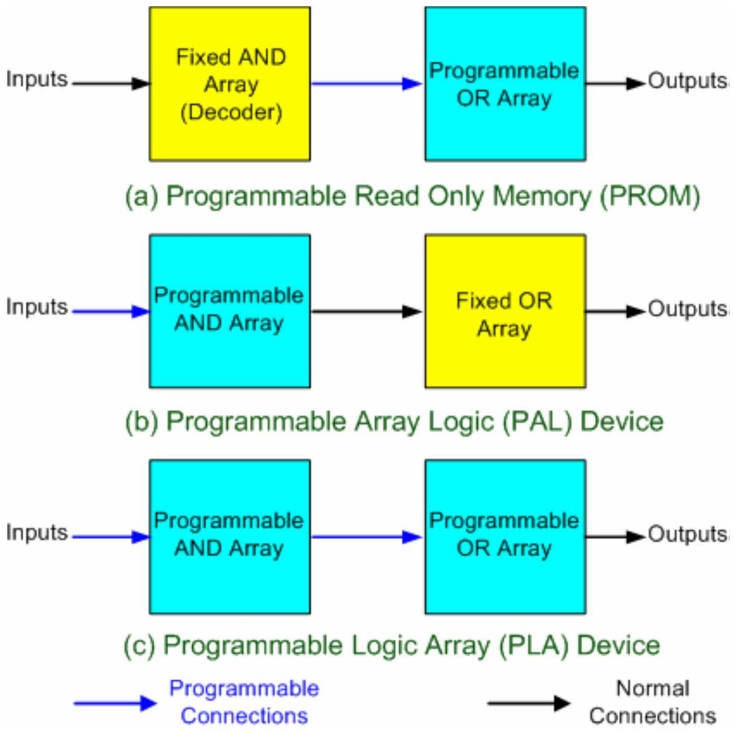
Further, this is grouped as,

- SPLDs (Simple Programmable Logic Devices)
    - ROM (Read-Only Memory)
    - PLA (Programmable Logic Array)
    - PAL (Programmable Array Logic)
    - GAL (Generic Array Logic)

- HCPLD (High Capacity Programmable Logic Device)
    - CPLD (Complex Programmable Logic Device)
    - FPGA (Field-Programmable Gate Array)

***Programmable Connections in PLDs:***

The programmable connections of AND-OR arrays for different types of PLDs are described here. Figure shows the locations of the programmable connections for the three types.



The PROM (Programmable Read Only Memory) has a fixed AND array (constructed as a decoder) and programmable connections for the output OR gates array. The PROM implements Boolean functions in sum-of-minterms form. The PAL (Programmable Array Logic) device has a programmable AND array and fixed connections for the OR array. The PLA (Programmable Logic Array) has programmable connections for both AND and OR arrays. So it is the most flexible type of PLD.

***Applications of Programmable Logic Devices***:

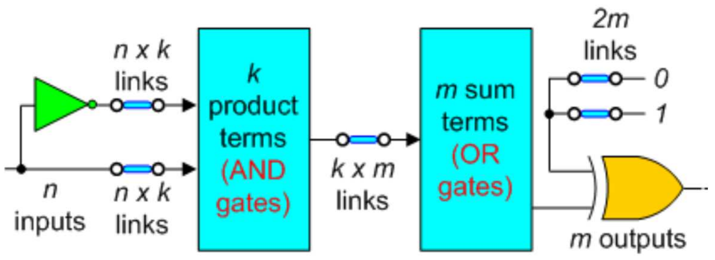
- Glue Logic
- State Machines
- Counters

- Synchronization
- Decoders
- Bus Interfaces
- Parallel-to-Serial
- Serial-to-Parallel

## 2. PROGRAMMABLE LOGIC ARRAY (PLA):

In PLAs, instead of using a decoder as in PROMs, a number (k) of AND gates is used where $k < 2^n$, (n is the number of inputs). Each of the AND gates can be programmed to generate a product term of the input variables and does not generate all the minterms as in the ROM. The AND and OR gates inside the PLA are initially fabricated with the links (fuses) among them. The specific Boolean functions are implemented in sum of products form by opening appropriate links and leaving the desired connections.

A block diagram of the PLA is shown in the figure. It consists of n inputs, m outputs, and k product terms. The product terms constitute a group of k AND gates each of 2n inputs. Links are inserted between all n inputs and their complement values to each of the AND gates. Links are also provided between the outputs of the AND gates and the inputs of the OR gates.

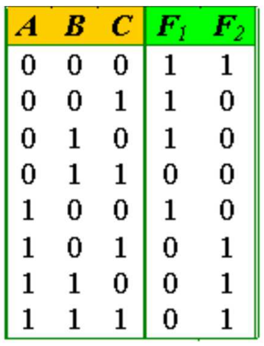


Since PLA has m-outputs, the number of OR gates is m. The output of each OR gate goes to an XOR gate, where the other input has two sets of links, one connected to logic 0 and other to logic 1. It allows the output function to be generated either in the true form or in the complement form. The output is inverted when the XOR input is connected to 1 (since $X \oplus 1 = X^{/}$). The output does not change when the XOR input is connected to 0 (since $X \oplus 0 = X$). Thus, the total number of programmable links is 2n x k + k x m + 2m.

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

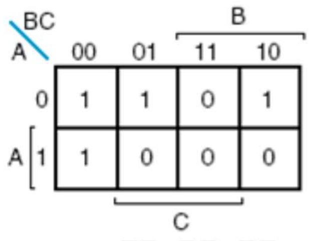
***Example 1:***

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

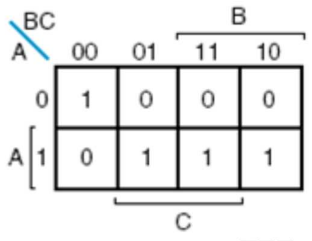


| *A* | *B* | *C* | $F_1$ | $F_2$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 |

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



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



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

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

The combination that gives a minimum number of product terms is,

$F_1' = AB + AC + BC$ or $F_1 = (AB + AC + BC)'$

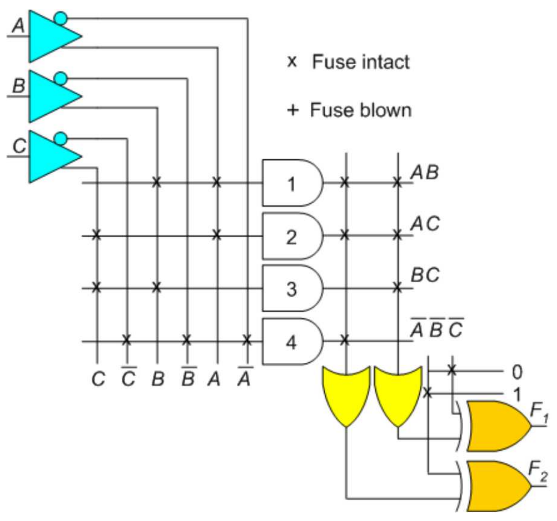
$F_2 = AB + AC + A'B'C'$

This gives only 4 distinct product terms: AB, AC, BC, and A'B'C'. So the PLA table will be as follows,

PLA programming table

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

For each product term, the inputs are marked with 1, 0, or – (dash). If a variable in the product term appears in its normal form (unprimed), the corresponding input variable is marked with a 1. A *1* in the Inputs column specifies a path from the corresponding input to the input of the AND gate that forms the product term. A *0* in the Inputs column specifies a path from the corresponding complemented input to the input of the AND gate. A dash specifies no connection.



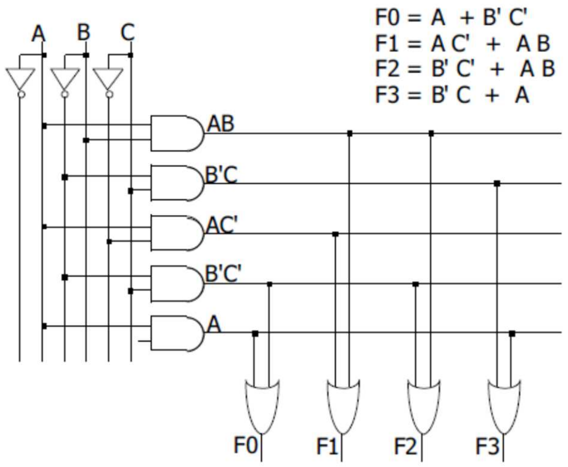
The appropriate fuses are blown and the ones left intact form the desired paths. It is assumed that the open terminals in the AND gate behave like a 1 input.

In the Outputs column, a T (true) specifies that the other input of the corresponding XOR gate can be connected to 0, and a C (complement) specifies a connection to 1.

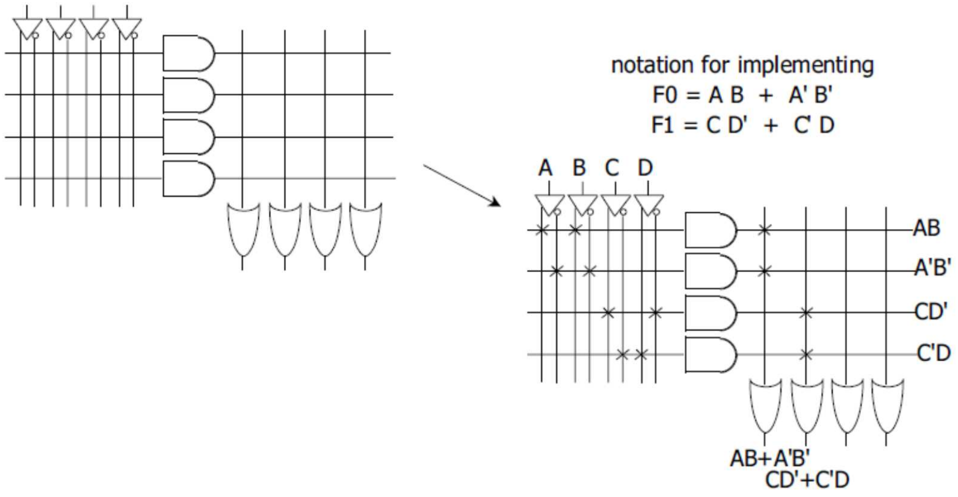
Note that output $F_1$ is the normal (or true) output even though a C (for complement) is marked over it. This is because $F_1$' is generated with AND-OR circuit prior to the output XOR. The output XOR complements the function $F_1$' to produce the true $F_1$ output as its second input is connected to logic 1.

***Example 2:***

All possible connections are available ***before programming*** as follows,



Unwanted connections are blown in the fuse (normally connected, break the unwanted ones) and in the anti-fuse (normally disconnected, make the wanted ones) ***after programming*** **for the given example** as follows,



***Limitations of PLAs***

PLAs come in various sizes. Typical size is 16 inputs, 32 product terms, 8 outputs

- Each AND gate has large fan-in. This limits the number of inputs that can be provided in a PLA
- 16 inputs forms $2^{16}$, possible input combinations; only 32 permitted (since 32 AND gates) in a typical PLA
- 32 AND terms permitted large fan-in for OR gates as well
  - This makes PLAs slower and slightly more expensive than some alternatives to be discussed shortly
- 8 outputs could have shared min-terms, but not required

***Applications of PLA:***

- PLA is used to provide control over datapath.
- PLA is used as a counter.
- PLA is used as a decoders.
- PLA is used as a BUS interface in programmed I/O.

## 3. PROGRAMMABLE READ ONLY MEMORY (PROM):

Read Only Memory (ROM) is a memory device, which stores the binary information permanently. If the ROM has programmable feature, then it is called as Programmable ROM PROM. The user has the flexibility to program the binary information electrically once by using PROM programmer. The input lines to the AND array are hard-wired and the output lines to the OR array are programmable. Thus, we generate $2^n$ product terms using $2^n$ AND gates having n inputs each, using n x $2^n$ decoder. This decoder generates 'n' min-terms. Each AND gate generates one of the possible AND products (i.e., min-terms).

Given a $2^k$ x n ROM, we can implement ANY combinational circuit with at most *k* inputs and at most n outputs. Because,

- k-to-$2^k$ decoder will generate all $2^k$ possible min-terms
- Each of the OR gates must implement a $\sum m()$
- Each $\sum m()$ can be programmed

The procedure for implementing a ROM-based circuit is as follows for the given example,

f(a,b,c) = a'b' + abc

g(a,b,c) = a'b'c' + ab + bc

h(a,b,c) = a'b' + c

and its solution can be obtained as,

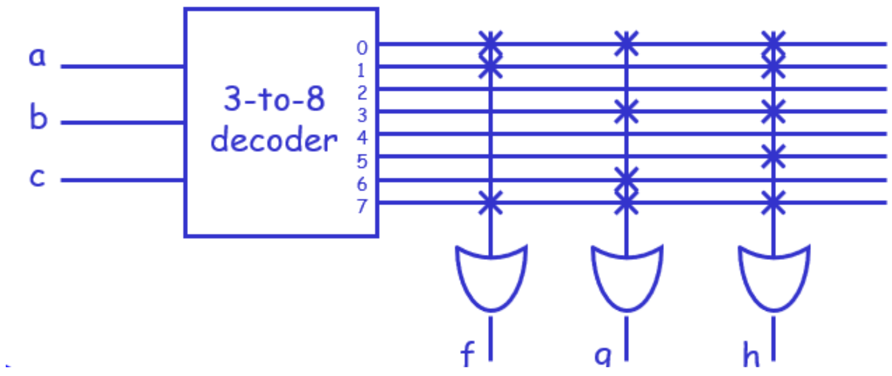
Express f(), g(), and h() in ∑m() format (use truth tables)

Program the ROM based on the 3 ∑m()'s

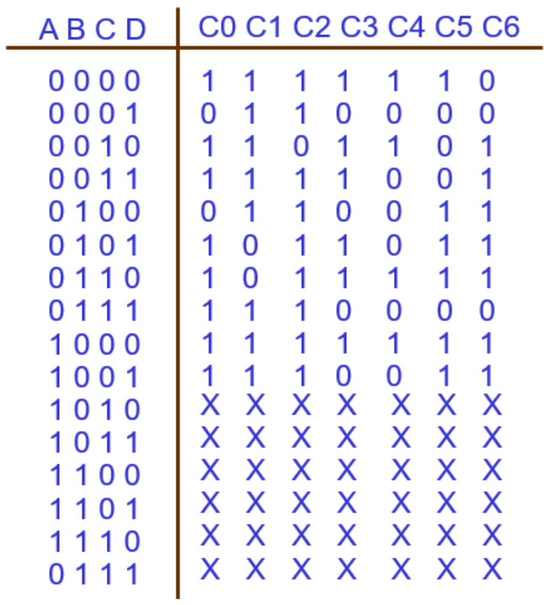
***Example:***

There are 3 inputs and 3 outputs, thus we need a 8x3 ROM block.

- f = ∑m(0, 1, 7)
- g = ∑m(0, 3, 6, 7)
- h = ∑m(0, 1, 3, 5, 7)



Another practical application of PROM device is BCD to 7 Segment Display Controller and the corresponding input and output relationship are shown in the following table.



| A B C D | C0 | C1 | C2 | C3 | C4 | C5 | C6 |
|---|---|---|---|---|---|---|---|
| 0 0 0 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 0 0 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 0 1 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 0 1 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 1 0 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 1 0 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 1 1 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 1 1 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 0 0 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 0 0 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 0 1 0 | X | X | X | X | X | X | X |
| 1 0 1 1 | X | X | X | X | X | X | X |
| 1 1 0 0 | X | X | X | X | X | X | X |
| 1 1 0 1 | X | X | X | X | X | X | X |
| 1 1 1 0 | X | X | X | X | X | X | X |
| 0 1 1 1 | X | X | X | X | X | X | X |

***Comparison: ROM Vs PLA***

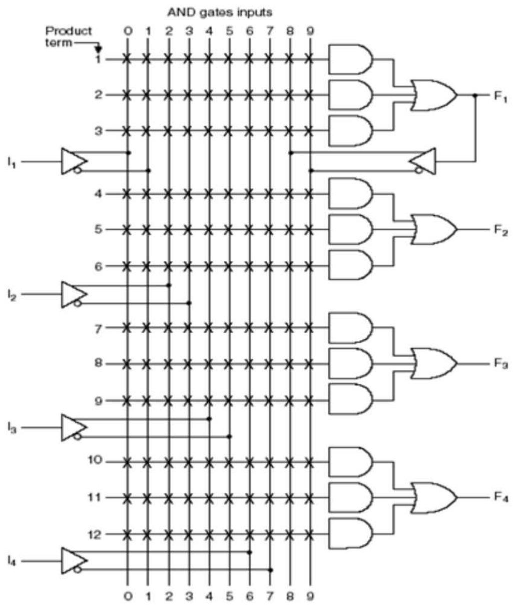
- ❑ ROM approach advantageous when
  - ➢ design time is short (no need to minimize output functions)
  - ➢ most input combinations are needed (e.g., code converters)
  - ➢ little sharing of product terms among output functions
- ❑ ROM problems
  - ➢ size doubles for each additional input (32x4 for Calendar example)
  - ➢ can't exploit don't cares
- ❑ PLA approach advantageous when
  - ➢ design tools are available for multi-output minimization
  - ➢ there are relatively few unique minterm combinations
  - ➢ many minterms are shared among the output functions
  - ➢ Supports multilevel implementation using feedback
- ❑ PAL problems
  - ➢ constrained fan-ins on OR plane
  - ➢ Difficulty of common term re-use??

## 4. PROGRAMMABLE ARRAY LOGIC (PAL):

PAL is a programmable logic device that has Programmable AND array & fixed OR array. The advantage of PAL is that we can generate only the required product terms of Boolean function instead of generating all the min terms by using programmable AND gates. As only AND gates are programmable, the PAL device is easier to program but it is not as flexible as the PLA. Here, the inputs of AND gates are programmable. That means each AND gate has both normal and complemented inputs of variables. So, based on the requirement, we can program any of those inputs. So, we can generate only the required product terms by using these AND gates. Here, the inputs of OR gates are not of programmable type. So, the number of inputs to each OR gate will be of fixed type. Hence, apply those required product terms to each OR gate as inputs. Therefore, the outputs of PAL will be in the form of sum of products form.

The device shown in the below figure has 4 inputs and 4 outputs. Each input has a buffer-inverter gate, and each output is generated by a fixed OR gate. The device has 4 sections, each composed of a 3-wide AND-OR array, meaning that there are 3 programmable AND gates in each section.

Each AND gate has 10 programmable input connections indicating by 10 vertical lines intersecting each horizontal line. The horizontal line symbolizes the multiple input configuration of an AND gate. One of the outputs $\mathbf{F_1}$ is connected to a buffer-inverter gate and is fed back into the inputs of the AND gates through programmed connections.



Designing using a PAL device, the Boolean functions must be simplified to fit into each section. The number of product terms in each section is fixed and if the number of terms in the function is too large, it may be necessary to use two or more sections to implement one Boolean function.

***Example:***

Implement the following Boolean functions using the PAL device as shown above,

W(A, B, C, D) = Σm(2, 12, 13)

X(A, B, C, D) = Σm(7, 8, 9, 10, 11, 12, 13, 14, 15)

Y(A, B, C, D) = Σm(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)

Z(A, B, C, D) = Σm(1, 2, 8, 12, 13)

Simplifying the 4 functions to a minimum number of terms results in the following Boolean functions:

W = ABC’ + A’B’CD’

X = A + BCD

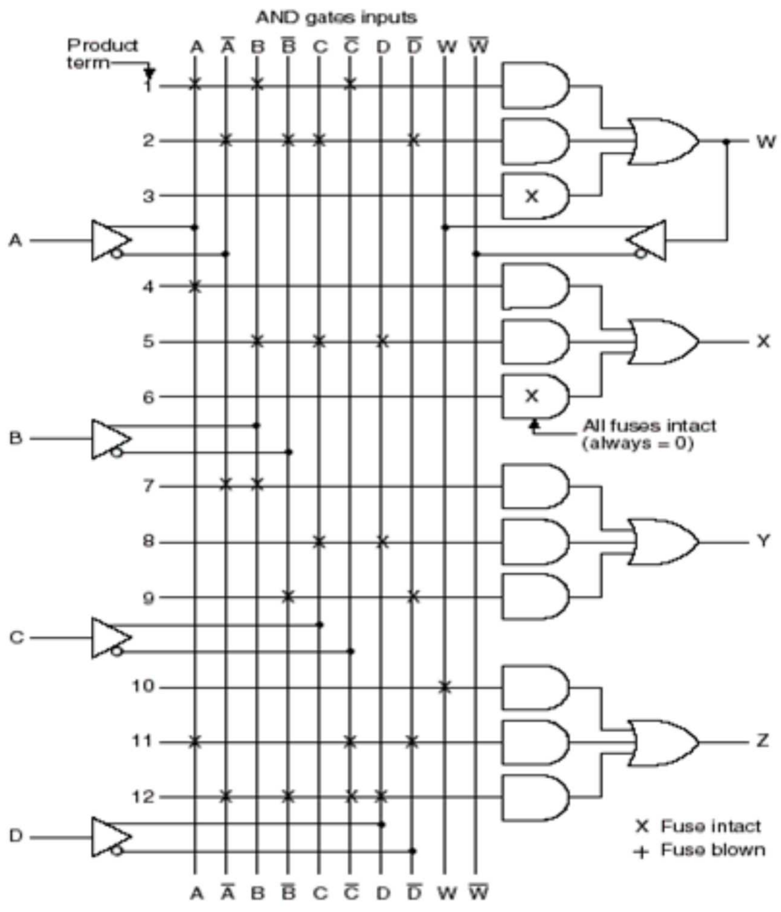
Y = A’B + CD + B’D’

Z = ABC’ + A’B’CD + AC’D’ + A’B’C’D =W +AC’D’ + A’B’C’D

Note that the function for **Z** has four product terms. The logical sum of two of these terms is equal to **W**. Thus, by using **W**, it is possible to reduce the number of terms for **Z** from four to three, so that the function can fit into the given PAL device.

| Product term | AND Inputs | | | | | Outputs |
|---|---|---|---|---|---|---|
| | A | B | C | D | W | |
| 1 | 1 | 1 | 0 | — | — | $W = AB\overline{C}$ |
| 2 | 0 | 0 | 1 | 0 | — | $+\overline{A}\,\overline{B}C\overline{D}$ |
| 3 | — | — | — | — | — | |
| 4 | 1 | — | — | — | — | $X = A$ |
| 5 | — | 1 | 1 | 1 | — | $+BCD$ |
| 6 | — | — | — | — | — | |
| 7 | 0 | 1 | — | — | — | $Y = \overline{A}B$ |
| 8 | — | — | 1 | 1 | — | $+CD$ |
| 9 | — | 0 | — | 0 | — | $+\overline{B}\,\overline{D}$ |
| 10 | — | — | — | — | 1 | $Z = W$ |
| 11 | 1 | — | 0 | 0 | — | $+A\overline{C}\,\overline{D}$ |
| 12 | 0 | 0 | 0 | 1 | — | $+\overline{A}\,\overline{B}\,\overline{C}D$ |

The PAL programming table is similar to the table used for the PLA, except that only the inputs of the AND gates need to be programmed. The following figure shows the connection map for the PAL device, as specified in the programming table.



Since both W and X have two product terms, third AND gate is not used. If all the inputs to this AND gate left intact, then its output will always be 0, because it receives both the true and complement of each input variable i.e., AA’ =0

***Inferences:***

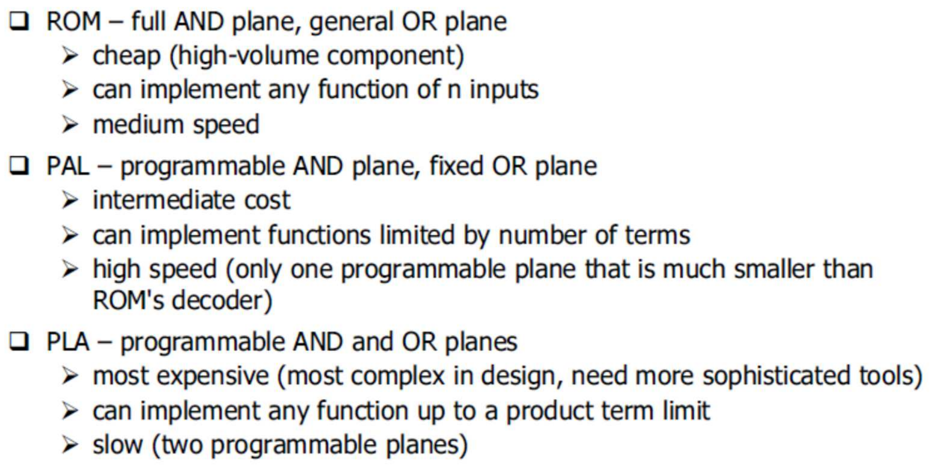
If an I/O pin’s output-control gate produces a constant 1, the output is always enabled, but the pin may still be used as an input too.

Outputs can be used to generate first-pass “*helper terms*” for logic functions that cannot be performed in a single pass with the limited number of AND terms available for a single output.

***Comparison: PAL Vs PLA***

- PALs have the same limitations as PLAs (small number of allowed AND terms) plus they have a fixed OR plane i.e., less flexibility than PLAs
- PALs are simpler to manufacture, cheaper, and faster (better performance)
- PALs also often have extra circuitry connected to the output of each OR gate
    - The OR gate plus this circuitry is called a macro-cell

***Comparison of ROM, PAL and PLA***



- ROM – full AND plane, general OR plane
    - cheap (high-volume component)
    - can implement any function of n inputs
    - medium speed
- PAL – programmable AND plane, fixed OR plane
    - intermediate cost
    - can implement functions limited by number of terms
    - high speed (only one programmable plane that is much smaller than ROM's decoder)
- PLA – programmable AND and OR planes
    - most expensive (most complex in design, need more sophisticated tools)
    - can implement any function up to a product term limit
    - slow (two programmable planes)

**PROGRAMMING SPLDs:**

SPLDs must be programmed so that the switches are in the correct places CAD tools are usually used to do this. A fuse map is created by the CAD tool and then that map is downloaded to the device via a special programming unit.

There are two basic types of programming techniques,

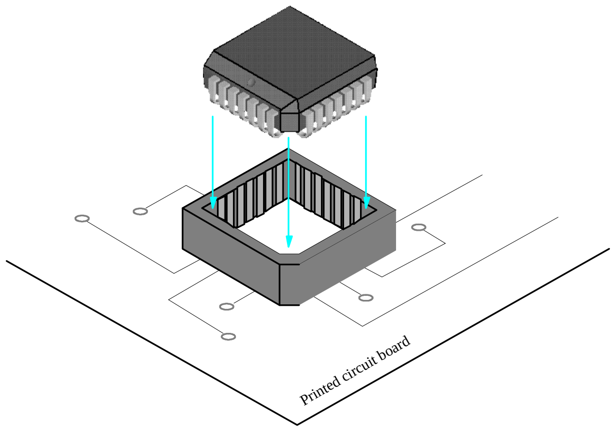
- Removable sockets on a PCB

- In system programming (ISP) on a PCB

This approach is not very common for PLAs and PALs but it is quite common for more complex PLDs.

***Removable SPLD Socket Package:***

The SPLD is removed from the PCB, placed into the unit and programmed there.



***In-System Programming (ISP):***

Used when the SPLD cannot be removed from the PCB. A special cable and PCB connection are required to program the SPLD from an attached computer. Very common approach to programming more complex PLDs like CPLDs, FPGAs, etc.

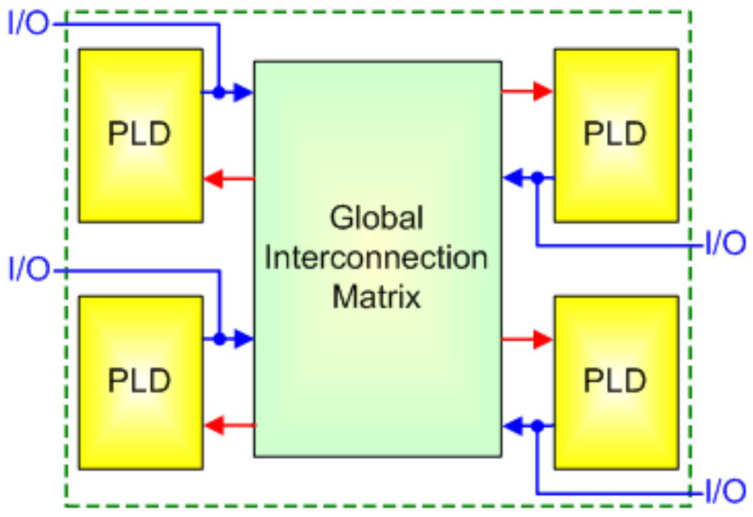
**5. COMPLEX PROGRAMMABLE LOGIC DEVICES (CPLDs):**

A CPLD contains a bunch of PLD blocks whose inputs and outputs are connected together by a global interconnection matrix. A CPLD is an arrangement of many SPLD-like blocks on a single chip. These circuit blocks might be either PAL-like or PLA-like blocks. Thus a CPLD has two levels of programmability: each PLD block can be programmed, and then the interconnections between the PLDs can be programmed.

***Characteristics:***

- They have a higher input to logic gate ratio.
- These devices are denser than SPLDs but have better functional abilities.
- CPLDs are based on EPROM or EEPROM technology.
- If you require a larger number of macrocells for a given application, ranging anywhere between 32 to 1000 macrocells, then a Complex Programmable Logic Device is the solution.

- Thus, we use CPLD in applications involving larger I/Os, but data processing is relatively low.



**CASE STUDY: Xilinx XC9500 CPLD Family:**

The Xilinx XC9500 series is a family of CPLDs with a similar architecture but varying numbers of external input/output (I/O) pins and internal PLDs which is called as function blocks (FBs). Each internal PLD has 36 inputs and 18 macrocells according to the number of chip family. Macro-cells whose outputs are usable only internally are sometimes called buried macrocells.

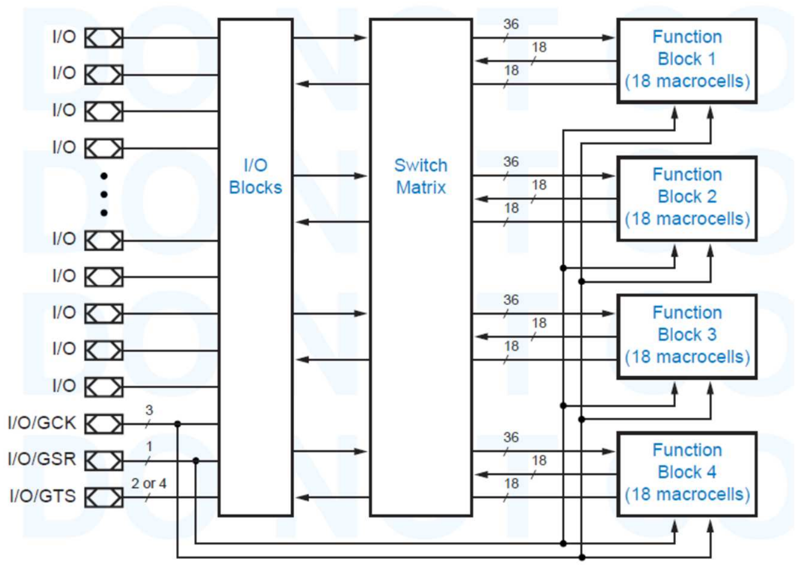
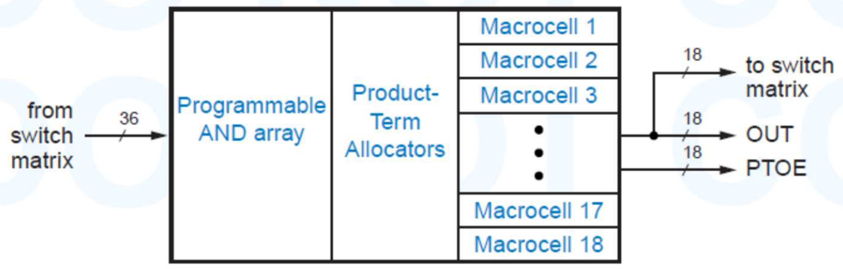


Figure shows the block diagram of the internal architecture of a typical XC9500-family CPLD. The external I/O pins can be used as input, output or bi-directional pins according to device programming. The pins marked I/O/GCK, I/O/GSR and I/O/GTS are special purpose pins. Any of these pins can b e used as global clocks (GCK). The same pin can be used as an asynchronous preset or clear. Two or four pins can be used as global three state controls (GTS).

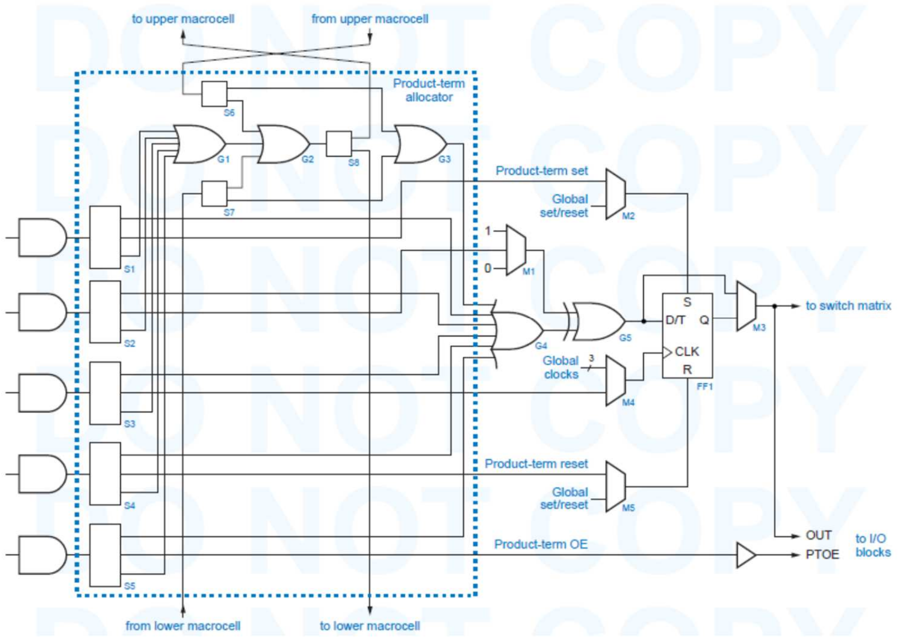
***Function Block Architecture***

The basic structure of an XC9500 FB is shown in following figure. The programmable AND array has just 90 product terms. It has fewer AND terms per microcell. Each FB will receive 36 signals from the switch matrix, the macrocell outputs from each of the FB and the external inputs from the I/O pins are applied to the switching matrix. Each FB has 18 outputs which run “under” the switch matrix and connect to the I/O blocks. These signals are only the output enable signals for the I/O block output drivers



***Macrocell:***

The XC9500 and other CPLDs have product-term allocators that allow a macrocell’s unused product terms to be used by other nearby macrocells in the same FB. Figure shows the logic diagram of the XC9500 product-term allocator and macrocell. In this figure, the rectangular boxes labelled S1–S8 are programmable signal-steering elements that connect their input to one of their two or three outputs. The trapezoidal boxes labeled M1–M5 are programmable multiplexers that connect one of their two to four inputs to their output. The five AND gates associated with the macrocell appear on the left-hand side of the figure. Each one is connected to a signal-steering box whose top output connects the product term to the macrocell’s main OR gate G4. Considering just this, only five product terms are available per macrocell. However, the top, sixth input of G4 connects to another OR gate G3 that receives product terms from the macrocells above and below the current one. Any of the macrocell’s product terms that are not otherwise used can be steered through S1–S5 to be combined in an OR gate G1 whose output can eventually be steered to the macrocell above or below by S8. Before steering, product-term allocator these product terms may be combined with product terms from below or above through S6, S7, and G2. Thus, product terms can be “daisy-chained” through successive macrocells to create larger sums of products. In principle, all 90 product terms in the FB could be combined and steered to one macrocell, although that would leave 17 out of the FB's 18 macrocells with no product terms at all.



DO NOT COPY

***Switch Matrix:***

The Fast CONNECT switch matrix connects signals to the FB inputs. All IOB outputs (corresponding to user pin inputs) and all FB outputs drive the Fast CONNECT matrix. Any of these (up to a FB fan-in limit of 36) may be selected, through user programming, to drive each FB with a uniform delay. The switch matrix is capable of combining multiple internal connections into a single wired-AND output before driving the destination FB. This provides additional logic capability and increases the effective logic fan-in of the destination FB without any additional timing delay.

***I/O Block:***

The I/O Block (IOB) interfaces between the internal logic and the device user I/O pins. Each IOB includes an input buffer, output driver, output enable selection multiplexer, and user programmable ground control. The input buffer is compatible with standard 5V CMOS, 5V TTL, and 3.3V signal levels. The input buffer uses the internal 5V voltage supply (VCCINT) to ensure that the input thresholds are constant and do not vary with the VCCIO voltage.

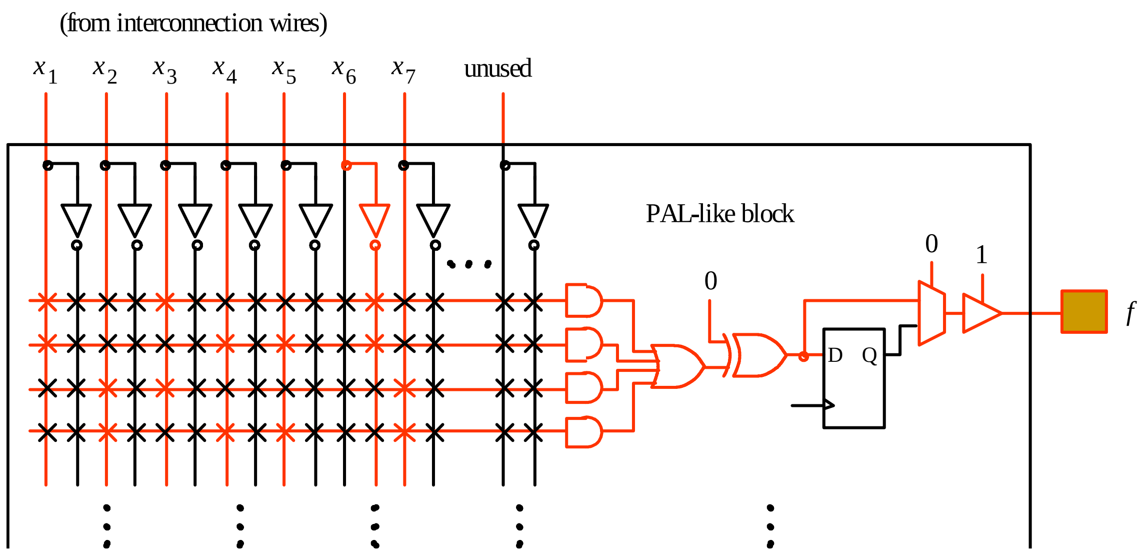
The output enable may be generated from one of four options: a product term signal from the macrocell, any of the global OE signals, always [1], or always [0]. There are two global output enables for devices with up to 144 macrocells, and four global output enables for the rest of the devices. Both polarities of any of the global 3-state control (GTS) pins may be used within the device.

***Features:***

- High-performance
- Large density range: 36 to 288 macrocells with 800 to6,400 usable gates
- 5V in-system programmable
- Endurance of 10,000 program/erase cycles
- Program/erase over full commercial voltage and temperature range
- Enhanced pin-locking architecture
- Flexible 36V18 Function Block
- 90 product terms drive any or all of 18 macrocells within Function Block
- Global and product term clocks, output enables, set and reset signals
- Extensive IEEE Std 1149.1 boundary-scan (JTAG) support
- Programmable power reduction mode in each macrocell
- Slew rate control on individual outputs - User programmable ground pin capability
- Extended pattern security features for design protection
- High-drive 24 mA outputs
- 3.3V or 5V I/O capability

***Example:***

Use a CPLD to implement the function, $f = x_1x_3x_6' + x_1x_4x_5x_6' + x_2x_3x_7 + x_2x_4x_5x_7$



***Applications of CPLD:***

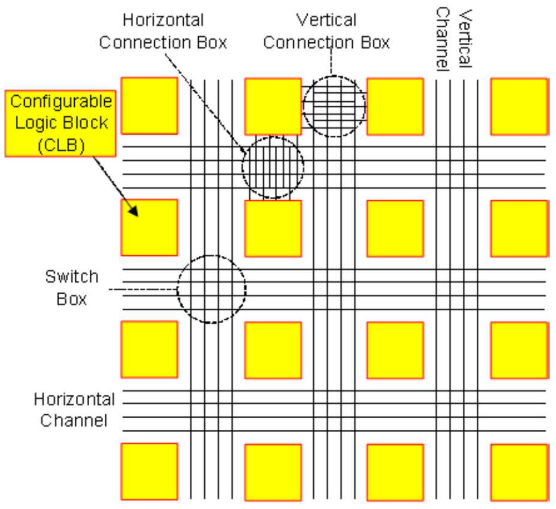
- Complex programmable logic devices are ideal for high performance, critical control applications.
- CPLD can be used in digital designs to perform the functions of boot loader
- CPLD is used for loading the configuration data of a field programmable gate array from non-volatile memory.
- Generally, these are used in small design applications like address decoding
- CPLDs are frequently used many applications like in cost sensitive, battery operated portable devices due to its low size and usage of low power.

## 6. FIELD PROGRAMMABLE GATE ARRAYS (FPGAs):

A Field Programmable Gate Array has an entire logic system integrated on a single chip. It offers excellent flexibility for reprogramming to the system designers. Logic circuitry involving more than a thousand gates use FPGAs. Compared to a normal custom system chip, the FPGA has ten times better integration density.

The FPGA consists of 3 main structures:

1. Programmable logic structure,
2. Programmable routing structure, and
3. Programmable Input/Output (I/O).



***Programmable Logic Structure:***

The programmable logic structure FPGA consists of a 2-dimensional array of configurable logic blocks (CLBs). These logic blocks have a lookup table in which the sequential circuitry is implemented. Each CLB can be configured (programmed) to implement any Boolean function of its input variables. Typically CLBs have between 4-6 input variables. Functions of larger number of variables are implemented using more than one CLB. In addition, each CLB typically contains 1 or 2 FFs to allow implementation of sequential logic.

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

***Configurable Logic Blocks (CLBs):***

*Look-up Table (LUT)-Based CLB :*

The basic unit of look-up table based FPGAs is the configurable logic block. The configurable logic block implements the logic functions. The look-up table based FPGA is the Xilinx 4000-series FPGA. Further, configurable logic block implements functions.

*PLA-Based CLB* :

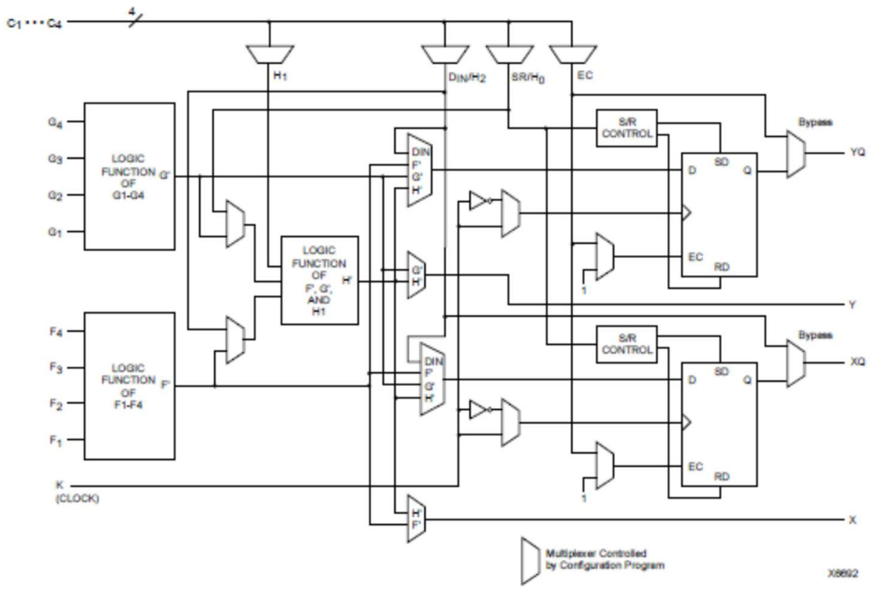
PLA-based FPGA devices are based on conventional PLDs. The important advantage of this structure is the logic circuits are implemented using only a few level logic. To improve integration density logic expander is used.

*Multiplexer-Based CLB :*

In Multiplexer-based FPGAs to implement the logic circuits the multiplexers are used. The main advantage of multiplexer-based FPGA is to provide more functionality by using minimum transistors. Due to large number of inputs, multiplexer-based FPGAs place high demands on routing.

**CASE STUDY: Xilinx 4000 FPGA Family:**

The principle CLB elements are shown in following figure. Each CLB contains a pair of flip-flops and two independent 4-input function generators. These function generators have a good deal of flexibility as most combinatorial logic functions need less than four inputs. Thirteen CLB inputs and four CLB outputs provide access to the functional flip-flops. Configurable Logic Blocks implement most of the logic in an FPGA. Two 4-input function generators (F and G) offer unrestricted versatility. Most combinatorial logic functions need four or fewer inputs. However, a third function generator (H) is provided. The H function generator has three inputs. One or both of these inputs can be the outputs of F and G; the other input(s) are from outside the CLB. The CLB can therefore implement certain functions of up to nine variables, like parity check or expandable-identity comparison of two sets of four inputs.



Each CLB contains two flip-flops that can be used to store the function generator outputs. However, the flip-flops and function generators can also be used independently. DIN can be used as a direct input to either of the two flip-flops. H1 can drive the other flip-flop through the H function generator. Function generator outputs can also be accessed from outside the CLB, using two outputs independent of the flip-flop outputs. This versatility increases logic density and simplifies routing. Thirteen CLB inputs and four CLB outputs provide access to the function generators and flip-flops. These inputs and outputs connect to the programmable interconnect resources outside the block.

Four independent inputs are provided to each of two function generators (F1 - F4 and G1 - G4). These function generators, whose outputs are labelled F' and G', are each capable of implementing any arbitrarily defined Boolean function of four inputs. The function generators are implemented as memory look-up tables. The propagation delay is therefore independent of the function implemented. A third function generator, labelled H', can implement any Boolean function of its three inputs. Two of these inputs can optionally be the F' and G' functional generator out-puts. Alternatively, one or both of these inputs can come from outside the CLB (H2, H0). The third input must come from outside the block (H1).

Signals from the function generators can exit the CLB on two outputs. F' or H' can be connected to the X output. G' or H' can be connected to the Y output. A CLB can be used to implement any of the following functions:

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

Implementing wide functions in a single block reduces both the number of blocks required and the delay in the signal path, achieving both increased density and speed. The versatility of the CLB function generators significantly improves system speed. In addition, the design-software tools can deal with each function generator independently. This flexibility improves cell usage.

The flexibilty and symmetry of the CLB architecture facilitates the placement and routing of a given application. Since the function generators and flip-flops have independent inputs and outputs, each can be treated as a separate entity during placement to achieve high packing density. Inputs, outputs and the functions themselves can freely swap positions within thew CLB to avoid routing congestion during the placement and routing operation.

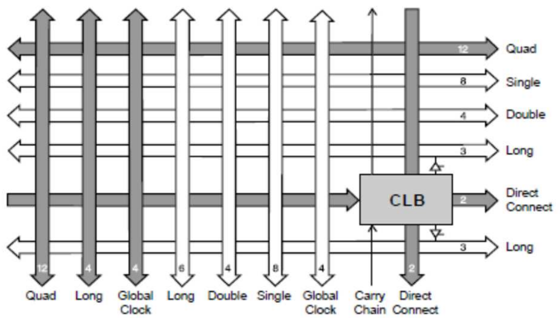
***Programmable Routing Structure:***

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

1. Vertical and horizontal routing channels which consist of different length wires that can be connected together if needed. These channel run vertically and horizontally between columns and rows of CLBs as shown in the Figure.

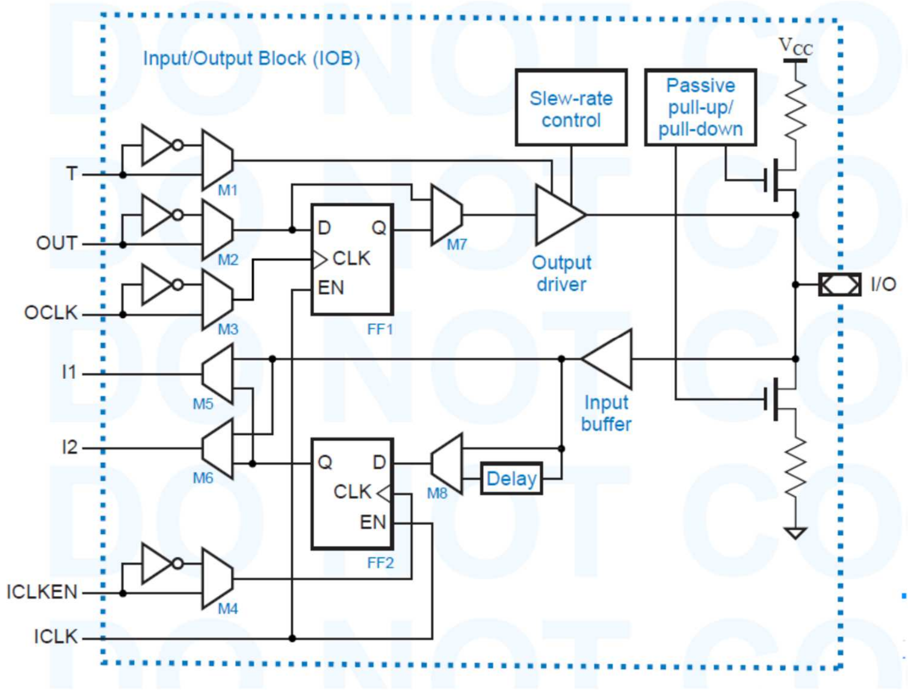
2. Connection boxes, which are a set of programmable links that can connect input and output pins of the CLBs to wires of the vertical or the horizontal routing channels.

3. Switch boxes, located at the intersection of the vertical and horizontal channels. These are a set of programmable links that can connect wire segments in the horizontal and vertical channels.



***Programmable I/O:***

These are mainly buffers that can be configured either as input buffers, output buffers or input/output buffers. They allow the pins of the FPGA chip to function either as input pins, output pins or input/output pins. The IOBs provide a simple interface between the internal user logic and the package pins.



*Input Signals:*

Two paths, labelled I1 and I2, bring input signals into the array. Inputs also connect to an input register that can be programmed as either an edge-triggered flip-flop or a level-sensitive transparent-Low latch. The choice is made by placing the appropriate primitive from the symbol library. The inputs can be globally configured for either TTL (1.2V) or CMOS (2.5V) thresholds.

The two global adjustments of input threshold and output level are independent of each other. There is a slight hysteresis of about 300mV.Seperate clock signals are provided for the input and output registers; these clocks can be inverted, generating either falling-edge or rising-edge triggered flip-flops. As is the case with the CLB registers, a global set/reset signal can be used to set or clear the input and output registers whenever the RESET net is alive.

*Registered Inputs:*

The I1 and I2 signals that exit the block can each carry either the direct or registered input signal. The input and output storage elements in each IOB have a common clock enable input, which through configuration can be activated individually for the input or output flip-flop or both. This clock enable operates exactly like the EC pin on the XC4000E CLB. It cannot be inverted within the IOB.

***Example:***

Use an FPGA with 2 input LUTS to implement the function,

$$f = x_1x_3x_6' + x_1x_4x_5x_6' + x_2x_3x_7 + x_2x_4x_5x_7$$

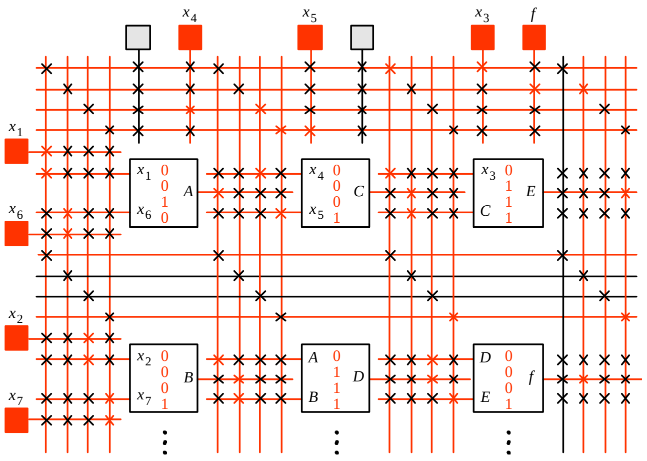
Fan-in of expression is too large for FPGA. This was simple to do in a CPLD

Factor *f* to get sub-expressions with max fan-in = 2

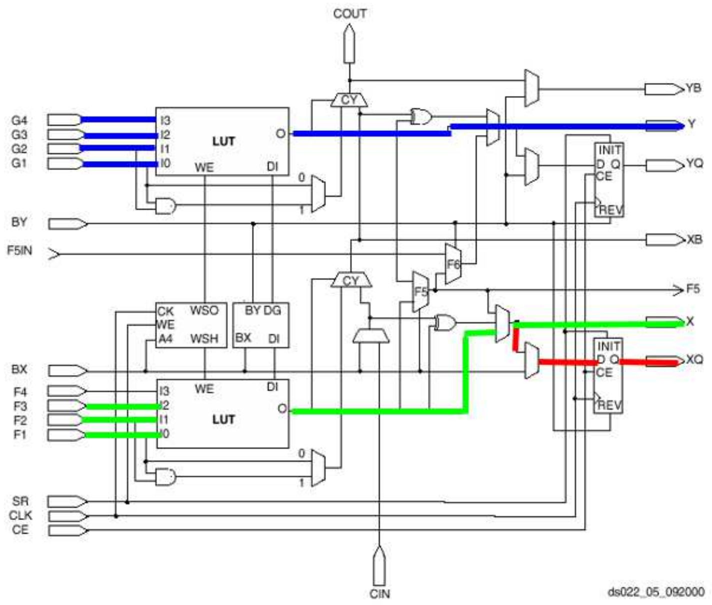
$$f = x_1x_6'(x_3 + x_4x_5) + x_2x_7(x_3 + x_4x_5)$$

$$= (x_1x_6' + x_2x_7)(x_3 + x_4x_5)$$ (Implementation shown in figure)

Could use Shannon's expansion instead. Goal is to build expressions out of 2-input LUTs



***Example for four and three input functions:***



***Applications of FPGAs:***

- Implementation of random logic
    - easier changes at system-level (one device is modified)
    - can eliminate need for full-custom chips
- Prototyping
    - ensemble of gate arrays used to emulate a circuit to be manufactured
    - get more/better/faster debugging done than possible with simulation
- Reconfigurable hardware
    - one hardware block used to implement more than one function
    - functions must be mutually-exclusive in time
    - can greatly reduce cost while enhancing flexibility
    - RAM-based only option
- Special-purpose computation engines
    - hardware dedicated to solving one problem (or class of problems)
    - accelerators attached to general-purpose computers

## CONCLUSION:

Over the last few years, programmable logic suppliers have made such phenomenal technical advances that PLDs are now seen as the logical solution of choice from many designers. One reason for this is that PLD suppliers such as Xilinx are “fabless” companies; instead of owning chip manufacturing foundries, Xilinx out sources that job to partners like IBM Microelectronics and UMC, whose chief occupation is making chips. This strategy allows Xilinx to focus on designing new product architectures, software tools, and intellectual property cores while having access to the most advanced semiconductor process technologies. Advanced process technologies help PLDs in a number of key areas: faster performance, integration of more features, reduced power consumption, and lower cost. Today Xilinx is producing programmable logic devices on a state-of-the-art 0.13-micron low-k copper process, one of the best in the industry.

Just a few years ago, for example, the largest FPGA was measured in tens of thousands of system gates and operated at 40 MHz. Today, however, FPGAs with advanced features offer millions of gates of logic capacity, operate at 300 MHz, and offer a new level of integrated functions such as processors and memory.

Just as significant, PLDs now have a growing library of intellectual property (IP) or cores – these are predefined and tested software modules that customer can use to create system functions instantly inside the PLD. Cores include everything from complex digital signal processing algorithms and memory controllers to bus interfaces and full-blown software-based microprocessors. Such cores save customers a lot of time and expense. It would take customers months to create these functions, further delaying a product introduction.

The value of programmable logic has always been its ability to shorten development cycles for electronic equipment manufacturers and help them get their product to market faster. As PLD suppliers continue to integrate more functions inside their devices, reduce costs, and increase the availability of time-saving IP cores, programmable logic is certain to expand its popularity with digital designers. Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform.