

**SRI CHANDRASEKHARENDRA SARASWATHI VISWA
MAHAVIDYALAYA**

(University U/S 3 of UGC Act 1956)

Accredited with “A” Grade by NAAC

ENATHUR, KANCHIPURAM - 631561



DEPARTMENT : EIE/MECHATRONICS
YEAR/SEM : FOURTH / SEVENTH
SUBJECT : VLSI DESIGN
SUBJECT CODE : EC7T1
UNIT : I TO V

Prepared by

S. S. SARAVANA KUMAR,

Assistant Professor

**Department of Electronics and Instrumentation Engineering
Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya
Enathur, Kanchipuram - 631561**

UNIT V PROGRAMMABLE LOGIC'S

(9 Hours)

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

COURSE OUTCOME

The students should be able to:

- CO1.** Understand the basic number system and Boolean algebra. **CO2.** Understand the basics of combinational and Sequential circuits. **CO3.** Know about Flip flops and their designing.
CO4. Analyze about State reduction techniques and various hazards present in the circuit.
CO5. Understanding the concepts of VHDL programming for designing Digital circuits.

TEXT 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.

REFERENCE BOOKS

1. VLSI Design Principles- John P. Uyemura, John Wiley, 2002
2. E. Fabricious , Introduction to VLSI design, McGraw-Hill 1990
3. Wayne Wolf, Modern VLSI Design, Pearson Education 2003A. Anand Kumar, “Switching Theory and Logic Design” – PHI, 2nd Edition

UNIT - I

INTRODUCTION TO VLSI AND MOS TRANSISTOR THEORY

PRE MCQ:

1. VLSI technology uses _____ to form integrated circuit.
 - a) Transistor
 - b) Switches
 - c) Diodes
 - d) Buffers

2. Medium scale integration has _____
 - a) ten logic gates
 - b) fifty logic gates
 - c) hundred logic gates
 - d) thousands logic gates

3. As die size shrinks, the complexity of making the photomasks _____
 - a) decreases
 - b) increases
 - c) remains the same
 - d) cannot be determined

4. _____ Architecture is used to design VLSI.
 - a) system on a device
 - b) single open circuit
 - c) system on a circuit
 - d) system on a chip

5. In depletion mode, source and drain are connected by _____
 - a) V_{dd}
 - b) V_{ss}
 - c) Conducting Channel
 - d) Insulating Channel

6. In enhancement mode, device is in _____ condition
 - a. insulating
 - b. conducting
 - c. partially conducting
 - d. non conducting

7. Inversion layer in enhancement mode consists of excess of _____
- a) positive carriers
 - b) negative carriers
 - c) both in equal quantity
 - d) neutral carriers
8. MOS transistors consist of which of the following?
- a) semiconductor layer
 - b) metal layer
 - c) layer of silicon-di-oxide
 - d) all of the mentioned
9. In MOS transistors _____ is used for their gate.
- a) metal
 - b) polysilicon
 - c) silicon-di-oxide
 - d) gallium
10. The depletion mode n-MOS differs from enhancement mode n-MOS in:
- a) Channel Length
 - b) Switching time
 - c) Threshold voltage
 - d) None of the mentioned

THEORY:

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.

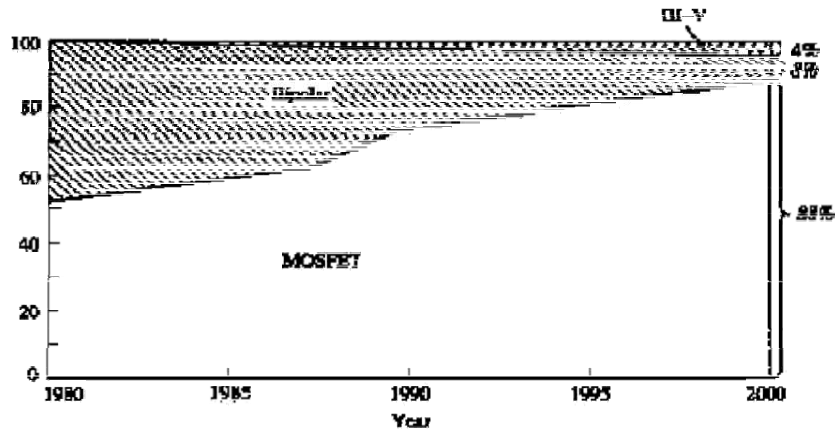
1963: Frank Wanlass (Fairchild) invented inverter, NOR and NAND CMOS gates. This invention starts the era of CMOS low power applications.

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 turn 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

I. 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

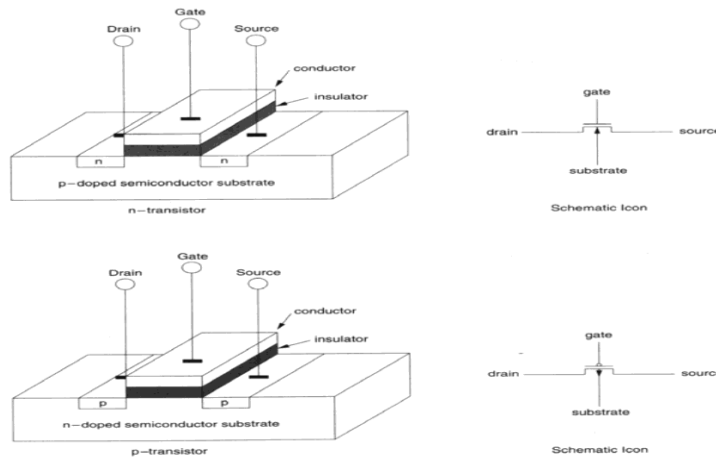


FIGURE 1.1 Physical structure of MOS transistors and their schematic icons

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. 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 (SiO_2 , 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.

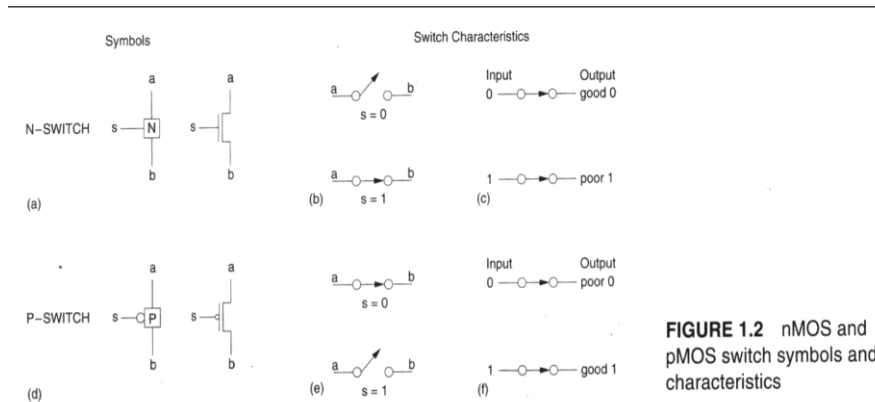


FIGURE 1.2 nMOS and pMOS switch symbols and characteristics

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

TABLE 1.1 Inverter truth table

A	Y
0	1
1	0

down to ‘0.’ This is summarized in Table.

“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.

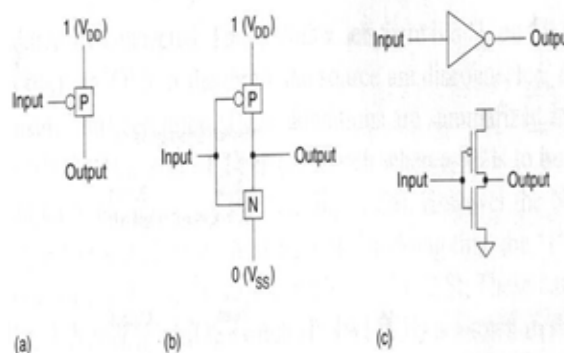


Fig. Inverter Circuit

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.

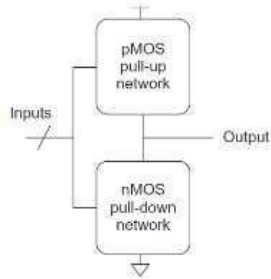
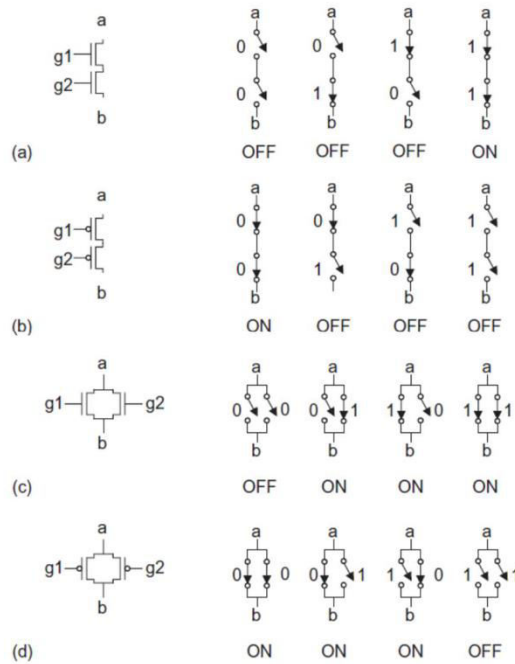


TABLE 1.3 Resolution of Gate Output Levels

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.

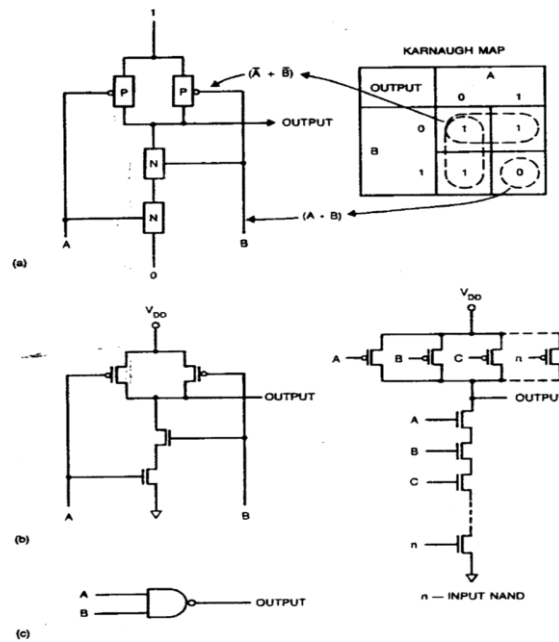


In general, when we join a pull-up network to a pull-down network to form a logic gate as shown in Figure, they both will attempt to exert a logic level at the output. The possible levels at the output are shown in Table. From this table it can be seen that the output of a CMOS logic gate can be in four states. The 1 and 0 levels have been encountered with the inverter and NAND gates, where both the pull-up and 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.

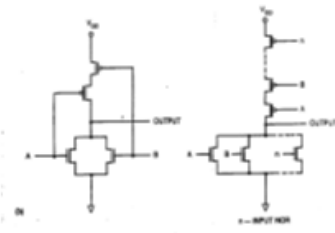
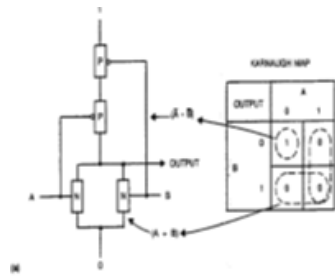
NAND Gate

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.



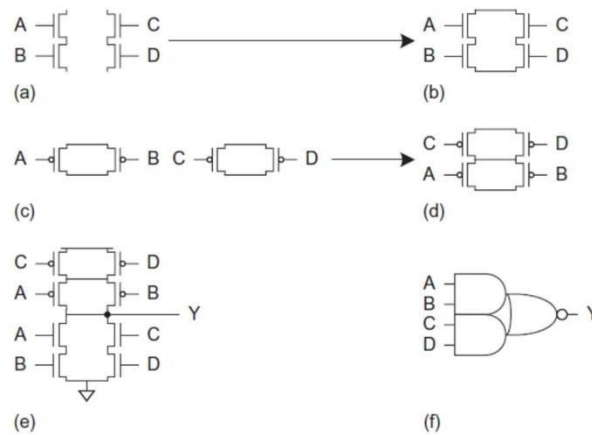
NOR Gate

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.

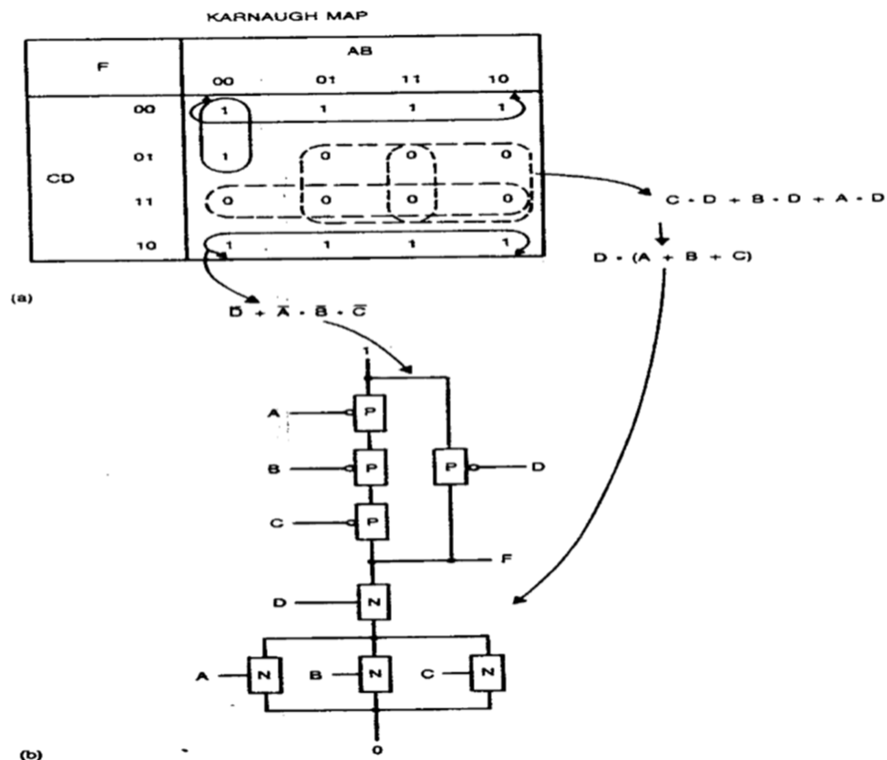


Compound Gates

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 $Y = (A \cdot B) + (C \cdot D)$ 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 $((A \cdot B) + (C \cdot D))$ indicating when the output should be pulled to '0.' The AND expressions $(A \cdot B)$ and $(C \cdot D)$ 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).



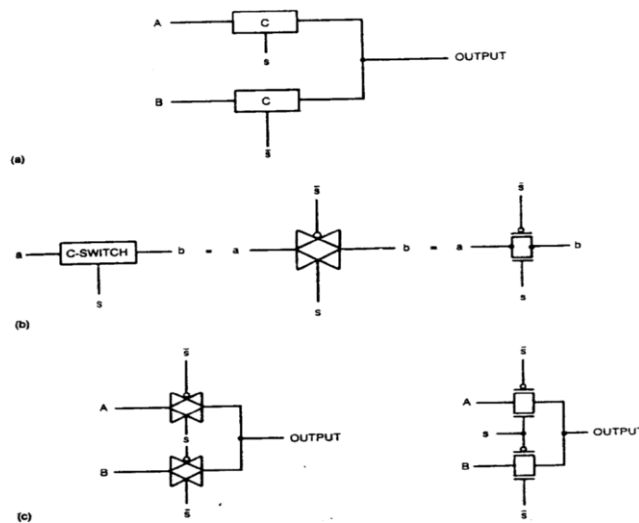
For the pMOS pull-up network, we must compute the complementary expression using switches that turn on with inverted polarity. By DeMorgan's Law, this is equivalent to interchanging AND and OR operations.



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 $Y = S \cdot D0 + \bar{S} \cdot D1$.

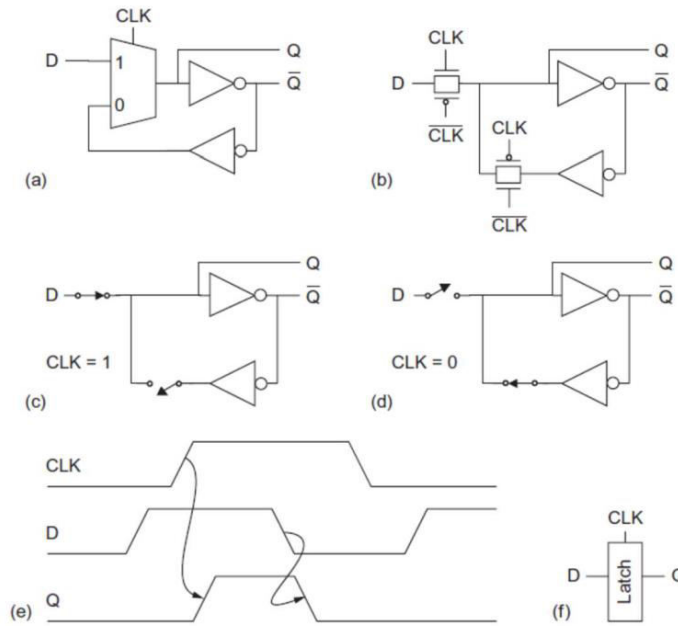


Two transmission gates can be tied together to form a compact 2-input multiplexer, as shown in Figure (a). The select and its complement enable exactly one of the two transmission gates at any given time. The complementary select S is often not drawn in the symbol, as shown in Figure (b).

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 $CLK = 1$,

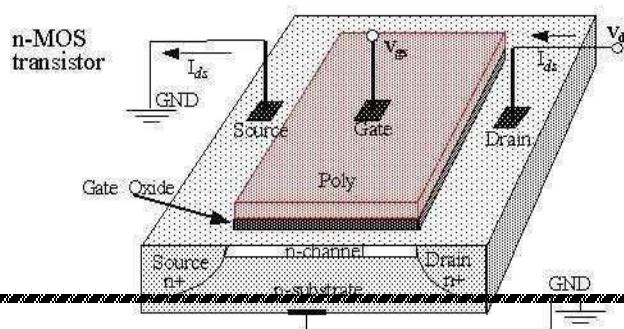


meaning that Q follows D. It becomes opaque when $CLK = 0$, meaning Q retains its previous value and ignores changes in D.

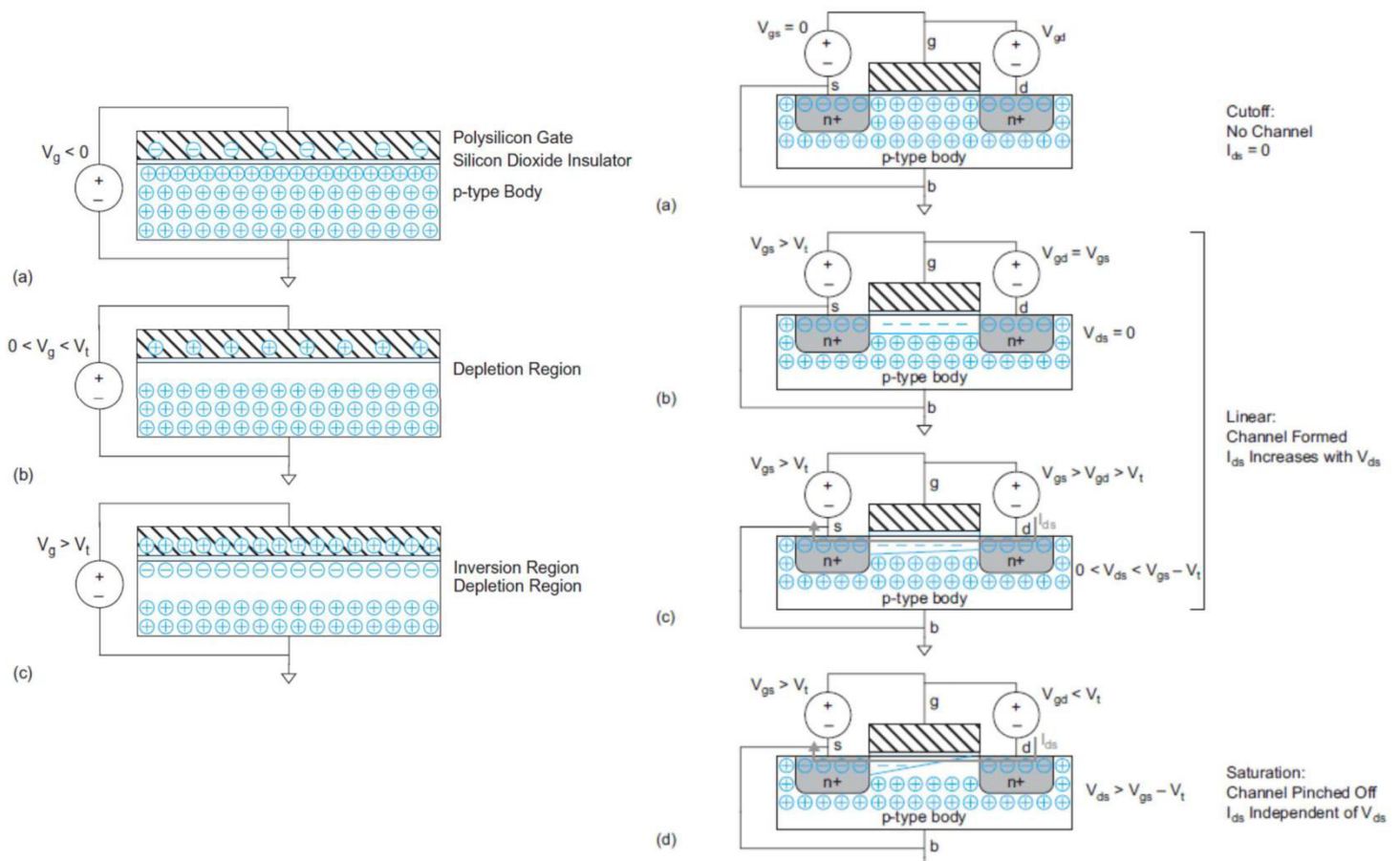
D latch built from a 2-input multiplexer and two inverters is shown in Figure (a). The multiplexer can be built from a pair of transmission gates, shown in Figure (b), because the inverters are restoring. This latch also produces a complementary output, \bar{Q} . When $CLK = 1$, the latch is transparent 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.

2. 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 V_t 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 t_{ox} 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 V_{gs} 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 $V_{ds} = V_{gs} - V_{gd}$. If $V_{ds} = 0$ (i.e., $V_{gs} = V_{gd}$), there is no electric field tending to push current from drain to source.

When a small positive potential V_{ds} is applied to the drain (Figure (c)), current I_{ds} 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 V_{ds} becomes sufficiently large that $V_{gd} < V_t$, 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 I_{ds} is controlled only by the gate voltage and ceases to be influenced by the drain. This mode is called saturation.

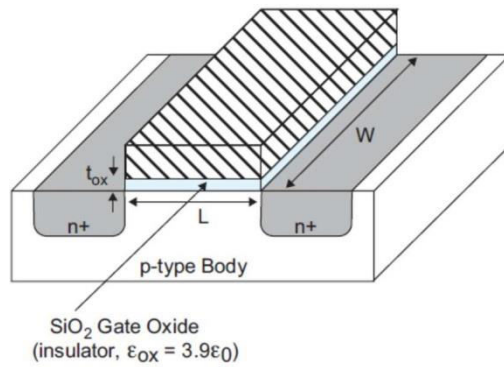
In summary, the nMOS transistor has three modes of operation. If $V_{gs} < V_t$, the transistor is cutoff (OFF). If $V_{gs} > V_t$, the transistor turns ON. If V_{ds} is small, the transistor acts as a linear resistor in which the current flow is proportional to V_{ds} . If $V_{gs} > V_t$ and V_{ds} is large, the transistor acts as a current source in which the current flow becomes independent of V_{ds} . 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

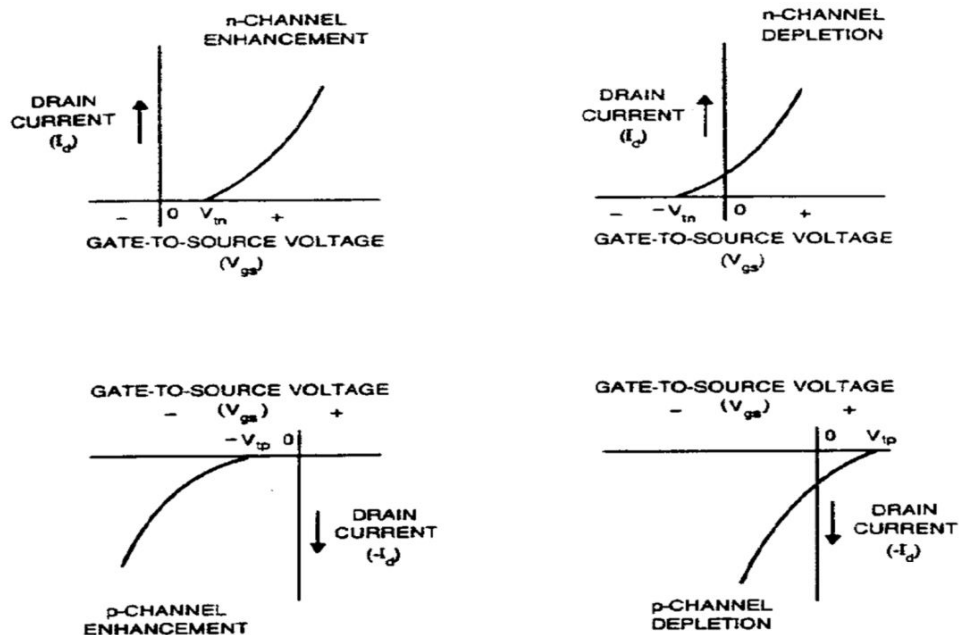
The long-channel model assumes that the current through an OFF transistor is 0. When a transistor turns ON ($V_{gs} > V_t$), the gate attracts carriers (electrons) to form a channel. The electrons drift from source to drain at a rate proportional to the electric field between these regions. Thus, we 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 t_{ox} , as shown in Figure.



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

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

Figure (a) shows the I-V characteristics for the transistor. According to the first-order model, the current is zero for gate voltages below V_t . For higher gate voltages, current increases linearly with V_{ds} for small V_{ds} . As V_{ds} reaches the saturation point $V_{dsat} = V_{GT}$, current rolls off and eventually becomes independent of V_{ds} 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 I_{ds} ,

- 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, V_t 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 V_{sb} 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 V_{t0} is the threshold voltage when the source is at the body potential, ϕ_s is the surface potential at threshold and γ is the body effect coefficient, typically in the range 0.4 to 1 V^{1/2}. In turn, these depend on the doping level in the channel, N_A . The body effect further degrades the performance of pass transistors trying to pass the weak value (e.g., nMOS transistors passing a '1').

V_t is largely determined at the time of fabrication, rather than by circuit conditions, like I_{ds} . Most are related to the material properties. For example, material parameters that effect V_t 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 (N_A).
- The oxide capacitance (C_{ox})
- Surface state charge (Q_{fc}).

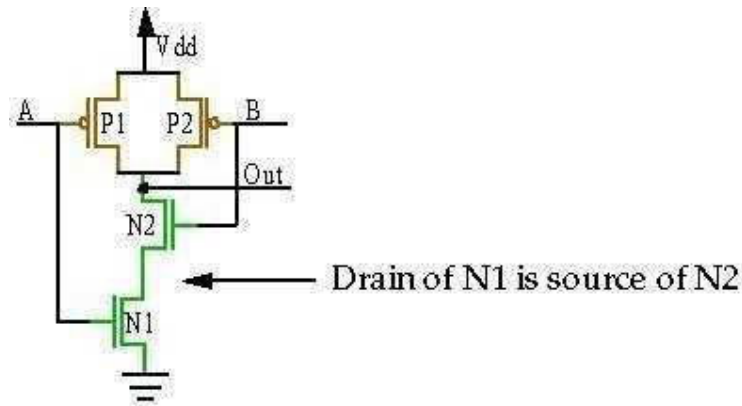
Also it is often necessary to adjust V_t . Two methods are common. Change Q_{fc} by introducing a small doped region at the oxide/substrate interface via ion implantation. Change C_{ox} by using a different insulating material for the gate.

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 (V_{sb}) may increase

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

at this connections, e.g. $V_{sbN1} = 0$ but $V_{sbN2} = V_{sb}$ adds to the channel-substrate potential,



Moreover, when the source of the nMOS transistor rises, V_{sb} 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 $V_{DD} = 1$ 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 $V_{DD} - V_{tn}$ when passing 1s. The pMOS passes 1s well, but only pulls down to $|V_{tp}|$ 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.

3. MOS INVERTERS

CMOS INVERTER TRANSFER CHARACTERISTICS

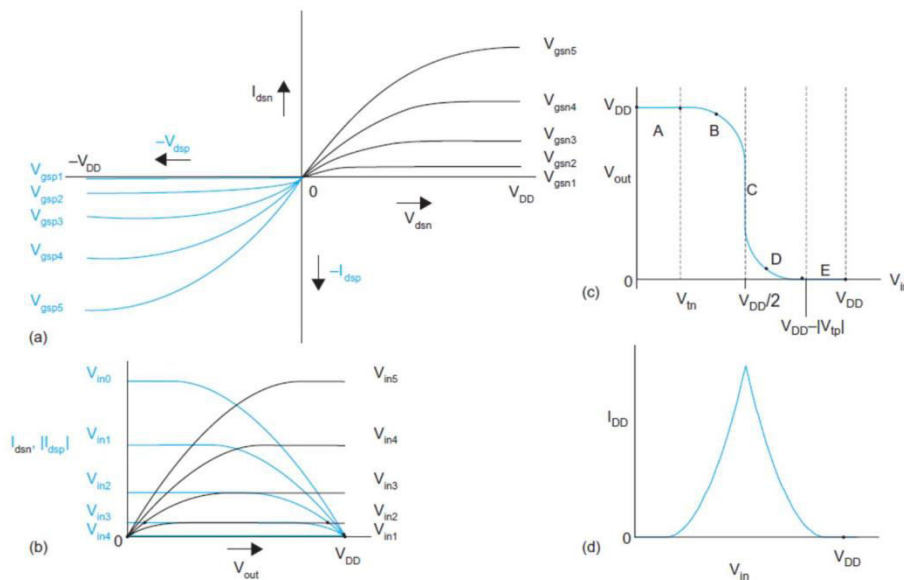
Digital circuits are merely analog circuits used over a special portion of their range. The DC transfer characteristics of a circuit relate the output voltage to the input voltage, assuming the input changes slowly enough that capacitances have plenty of time to charge or discharge. Specific ranges of 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 (V_{out} vs. V_{in}) 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, V_{tn} is the threshold voltage of the n-channel device, and V_{tp} is the threshold voltage of the p-channel device. Note that V_{tp} is negative. The equations are given both in terms of V_{gs} / V_{ds} and V_{in} / V_{out} . As the source of the nMOS transistor is grounded, $V_{gsn} = V_{in}$ and $V_{dsn} = V_{out}$. As the source of the pMOS transistor is tied to V_{DD} , $V_{gsp} = V_{in} - V_{DD}$ and $V_{dsp} = V_{out} - V_{DD}$.

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

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

The objective is to find the variation in output voltage (V_{out}) as a function of the input voltage (V_{in}). This may be done graphically. Given V_{in} , we must find V_{out} subject to the constraint that $I_{dsn} = |I_{dsp}|$. For simplicity, we assume $V_{tp} = -V_{tn}$ and that the pMOS transistor is 2–3 times as wide as the nMOS transistor so $\mu_n = \mu_p$. The plot shows I_{dsn} and I_{dsp} in terms of V_{dsn} and V_{dsp} for various values of V_{gsn} and V_{gsp} . Figure (b) shows the same plot of I_{dsn} and $|I_{dsp}|$ now in terms of V_{out} for various values of V_{in} . The possible operating points of the inverter, marked with dots, are the values of V_{out} where $I_{dsn} = |I_{dsp}|$ for a given value of V_{in} . These operating points are plotted on V_{out} vs. V_{in} axes in Figure (c) to show the inverter DC transfer characteristics. The supply current $I_{DD} = I_{dsn} = |I_{dsp}|$ is also plotted against V_{in} in Figure (d) showing that both transistors are momentarily ON as V_{in} passes through voltages between GND and V_{DD} , 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 V_{DD} . 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 $V_{in} = V_{DD}/2$ and that the slope of the transfer curve in this example is -1 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

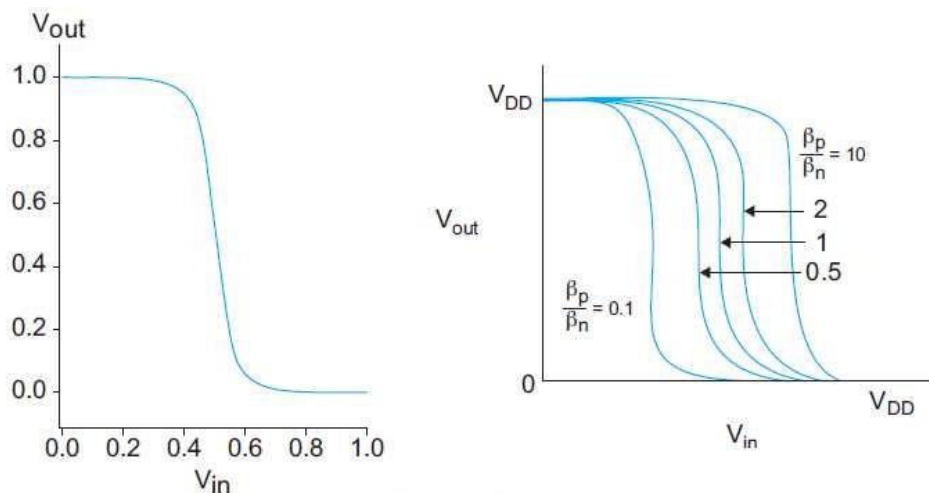
Region	Condition	p-device	n-device	Output
A	$0 \leq V_{in} < V_{tn}$	linear	cutoff	$V_{out} = V_{DD}$
B	$V_{tn} \leq V_{in} < V_{DD}/2$	linear	saturated	$V_{out} > V_{DD}/2$
C	$V_{in} = V_{DD}/2$	saturated	saturated	V_{out} drops sharply
D	$V_{DD}/2 < V_{in} \leq V_{DD} - V_{tp} $	saturated	linear	$V_{out}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

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 $V_{inv} = V_{in} = V_{out}$ 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 $\beta_p = \beta_n$, the inverter threshold voltage V_{inv} is $V_{DD}/2$. 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 $r = \beta_p / \beta_n$ 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 $V_{DD}/2$, we would expect the output will be greater than $V_{DD}/2$. 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

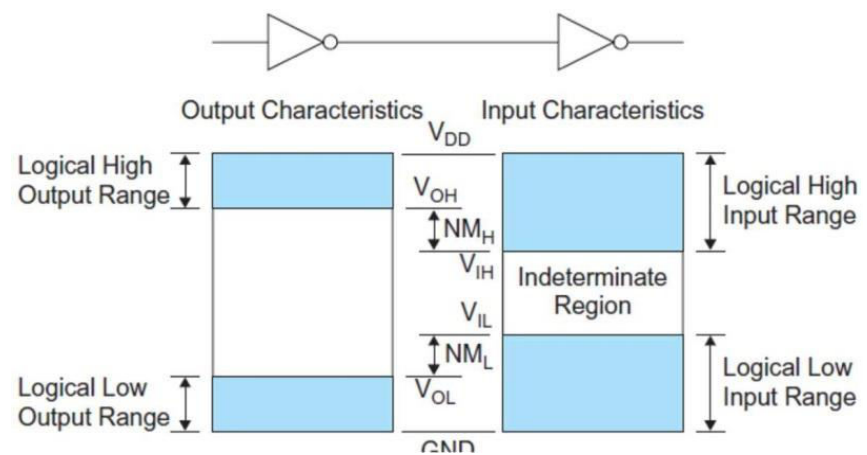
Noise margin is closely related to the DC voltage characteristics. This parameter allows you to determine the allowable noise voltage on the input of a gate so that the output will not be corrupted. The specification most commonly used to describe noise margin (or noise immunity) uses two parameters: 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.

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

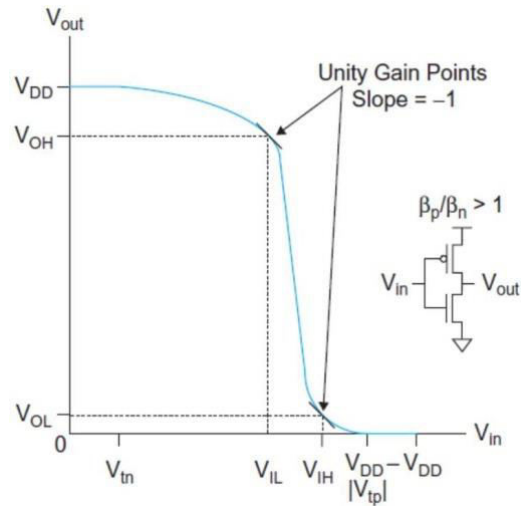
The value of NMH is the difference between the minimum HIGH output voltage of the driving gate and the minimum HIGH input voltage recognized by the receiving gate. Thus,

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

where V_{IH} = minimum HIGH input voltage, V_{IL} = maximum LOW input voltage, V_{OH} = minimum HIGH output voltage and V_{OL} = maximum LOW output voltage.



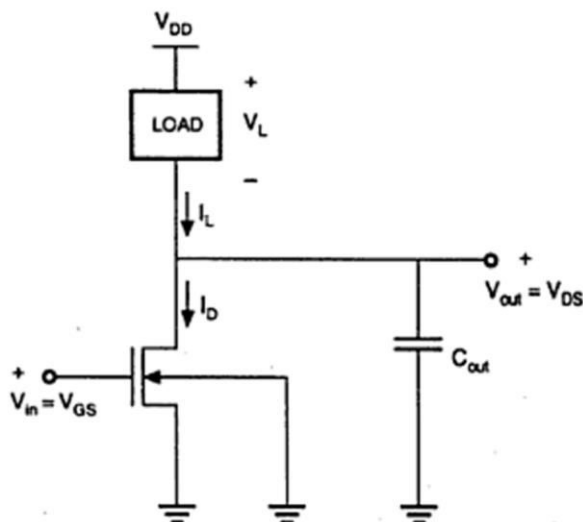
Inputs between V_{IL} and V_{IH} are said to be in the indeterminate region or forbidden zone and do not represent legal digital logic levels. Therefore, it is generally desirable to have V_{IH} as close as possible to V_{IL} and for this value to be midway in the "logic swing," V_{OL} to V_{OH} . This 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 V_{IL} , V_{OL} , V_{IH} , and V_{OH} 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 $|V_{tp}| = V_{tn}$, 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.

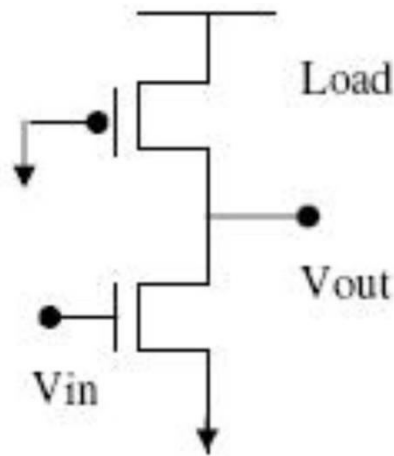
Principle of Operation

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 V_{dd} and logic 0 is represented by 0. V_{th} is the inverter threshold voltage, which is $V_{dd}/2$, where V_{dd} is the output voltage. The output is switched from 0 to V_{dd} when input is less than V_{th} . So, for $0 < V_{in} < V_{th}$ output is equal to logic 0 input and $V_{th} < V_{in} < V_{dd}$ 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 $V_{SS} = 0$. 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 V_{dd} . Thus the overvoltage on the p channel gate is always $V_{dd} - V_{Tp}$. 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 V_{DD}/V_{CC} rail. With input voltage $V_i = 0$, 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 C_L at a very fast rate. Thus the output voltage V_o rises very fast, which is usually much faster than the charging of C_L by an MOS transistor by itself.

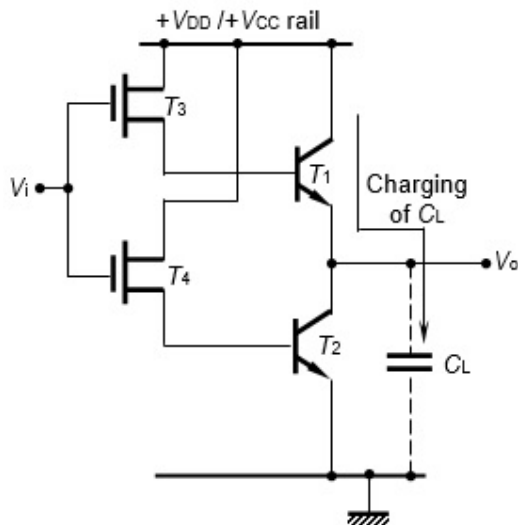


Fig. 3.43 BICMOS inverter

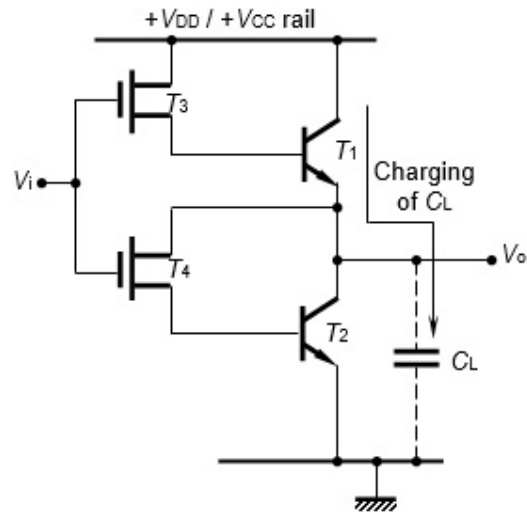


Fig. 3.44 Modified BICMOS inverter

Now, let $V_i = +V_{DD}$ (\equiv 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, C_L discharges very fast through T2 ON. Thus the charging and discharging of C_L 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.

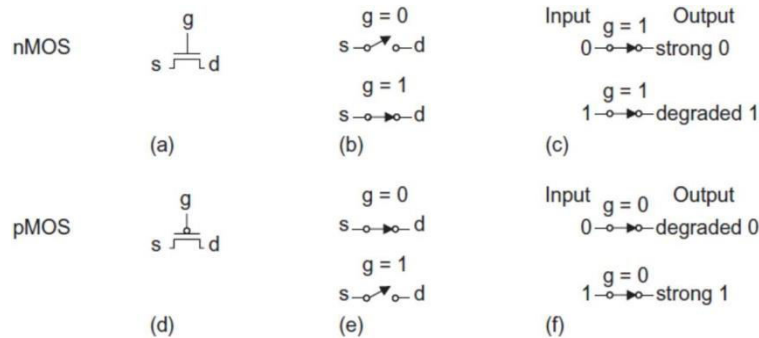
To overcome these problems, we modify the circuit, as shown in Figure. In this case, NMOS transistor T4 has its drain connected to the output terminal rather than to +VDD. As before, when T3 is turned ON, T1 is also turned ON. Now, when T4 is turned ON, we find T2 also to turn ON. However, since the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

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.

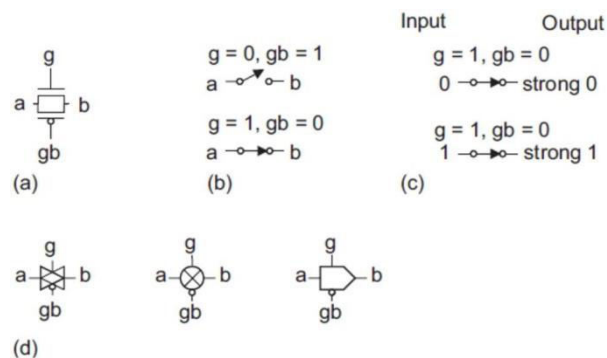
CMOS TRANSMISSION GATE

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 V_t . The switch turns ON, forming a channel connecting source to drain, when the gate voltage rises above V_t . 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 $V_{DD} - V_{tn}$ when passing 1s. The pMOS passes 1s well, but only pulls down to $|V_{tp}|$ 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.

POST MCQ

1. The difficulty in achieving high doping concentration leads to _____
 - a) error in concentration
 - b) error in variation**
 - c) error in doping
 - d) distribution error
2. nMOS devices are formed in _____
 - a) p-type substrate of high doping level
 - b) n-type substrate of low doping level
 - c) p-type substrate of moderate doping level**
 - d) n-type substrate of high doping level
3. What is the condition for non saturated region?
 - a) $V_{ds} = V_{gs} - V_t$
 - b) V_{gs} lesser than V_t
 - c) V_{ds} lesser than $V_{gs} - V_t$**
 - d) V_{ds} greater than $V_{gs} - V_t$
4. What is the condition for non conducting mode?
 - a) V_{ds} lesser than V_{gs}
 - b) V_{gs} lesser than V_{ds}
 - c) $V_{gs} = V_{ds} = 0$
 - d) $V_{gs} = V_{ds} = V_s = 0$**

5. What is the condition for linear region?
 - a) V_{gs} lesser than V_t
 - b) V_{gs} greater than V_t**
 - c) V_{ds} lesser than V_{gs}
 - d) V_{ds} greater than V_{gs}
6. The direction of electric field when the gate voltage is zero:
 - a) Metal to semiconductor**
 - b) Semiconductor to metal
 - c) No electric field exists
 - d) None of the mentioned
7. The threshold voltage depends on:
 - a) The workfunction difference between gate and channel
 - b) The gate voltage component to change surface potential
 - c) The gate voltage component to offset the depletion charge and fixed charges in gate oxide
 - d) All of the mentioned**
8. The Lower Noise Margin is given by:
 - a) $V_{OL} - V_{IL}$
 - b) $V_{IL} - V_{OL}$**
 - c) $V_{IL} \sim V_{OL}$ (Difference between V_{IL} and V_{OL} , depends on which one is greater)
 - d) All of the Mentioned
9. If V_{IL} of the 2nd gate is higher than V_{OL} of the 1st gate, then logic output 0 from the 1st gate is considered as:
 - a) Logic input 1
 - b) Uncertain
 - c) Logic input 0**
 - d) None of the mentioned
10. Thermal noise is generated from MOSFET by:
 - a) Conduction of charge carriers in the channel**
 - b) Electric field across the gate and channel
 - c) Capacitance of the gate oxide
 - d) Substrate bias effect

UNIT – II

COMBINATIONAL CIRCUITS

PRE MCQ:

1. Oxidation process is carried out using _____
 - a) **hydrogen**
 - b) low purity oxygen
 - c) sulphur
 - d) nitrogen
2. Photoresist layer is formed using _____
 - a) high sensitive polymer
 - b) **light sensitive polymer**
 - c) polysilicon
 - d) silicon di oxide
3. _____ is sputtered on the whole wafer.
 - a) silicon
 - b) calcium
 - c) potassium
 - d) **aluminium**
4. What is Lithography?
 - a) **Process used to transfer a pattern to a layer on the chip**
 - b) Process used to develop an oxidation layer on the chip
 - c) Process used to develop a metal layer on the chip
 - d) Process used to produce the chip
5. Positive photo resists are used more than negative photo resists because _____
 - a) **Negative photo resists are more sensitive to light, but their photo lithographic resolution is not as high as that of the positive photo resists**
 - b) Positive photo resists are more sensitive to light, but their photo lithographic resolution is not as high as that of the negative photo resists
 - c) Negative photo resists are less sensitive to light
 - d) Positive photo resists are less sensitive to light

THEORY:

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.

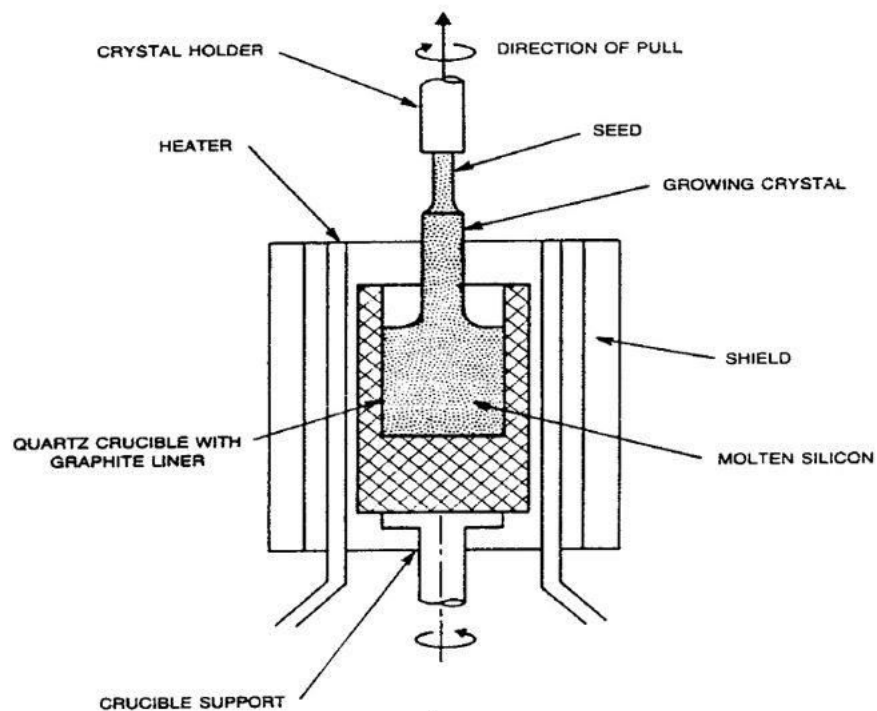
Wafer Formation

The basic raw material used in CMOS fabs is a wafer or disk of silicon, roughly 75 mm to 300 mm (12 inches—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

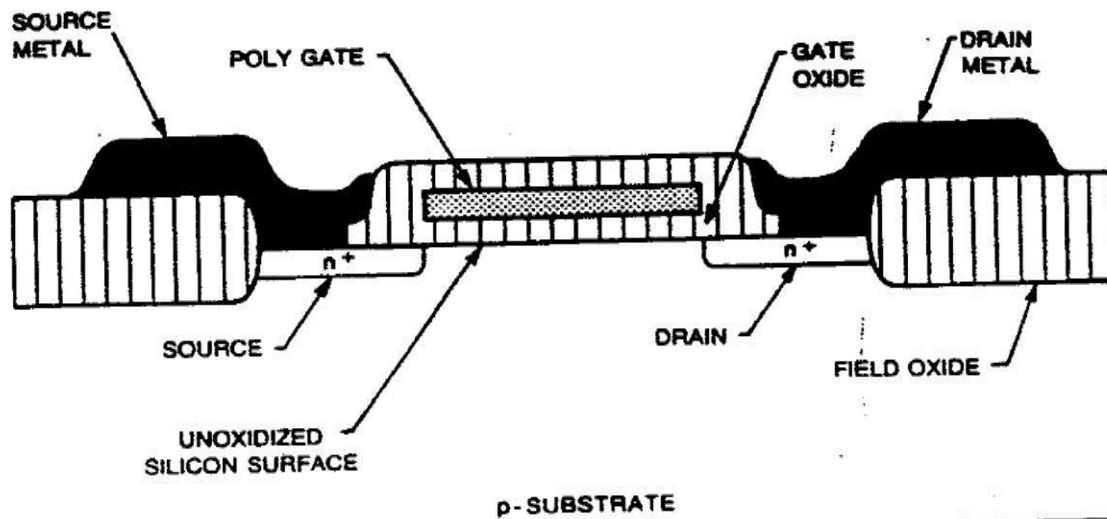
called photolithography, from the Greek photo (light), lithos (stone), and graphe (picture), which literally means “carving pictures in stone using light.” The primary method for defining areas of interest (i.e., where we want material to be present or absent) on a wafer is by the use of photoresists. 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 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 1.0mm thick, depending on their diameter. Following this operation, at least one face is polished to a flat, scratch-free mirror finish.

OXIDATION

Many of the structures and manufacturing techniques used to make silicon integrated 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.

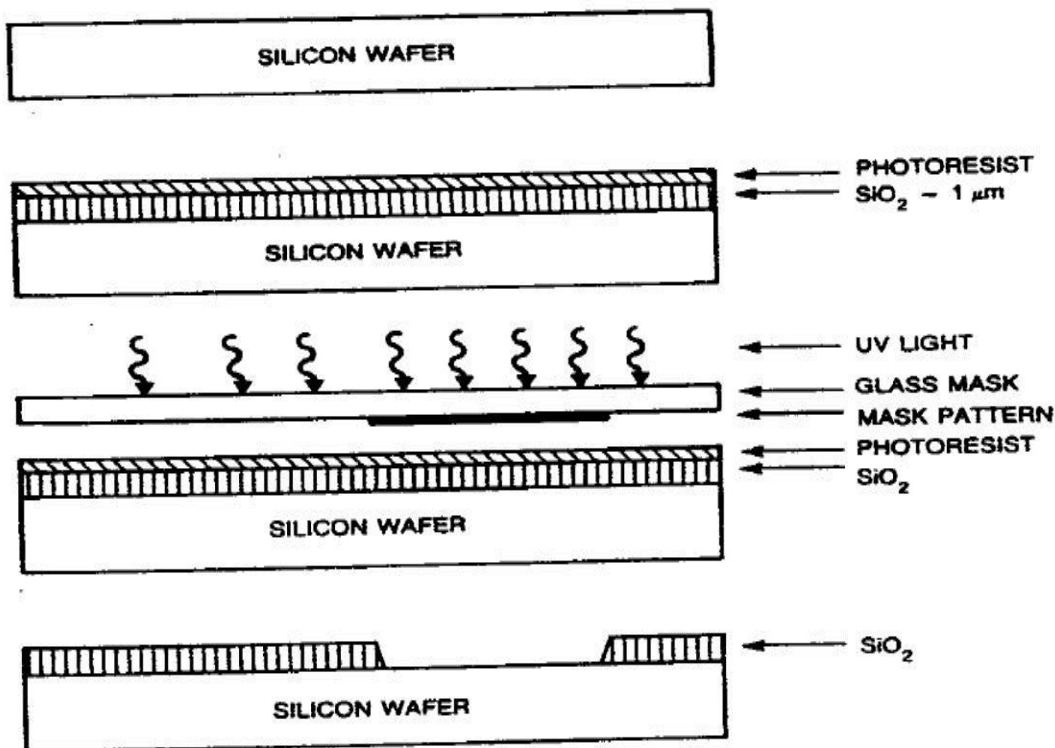
3.1.3 Selective diffusion

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 $1.5 \mu\text{m}$ to $2 \mu\text{m}$. 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 $0.5 \mu\text{m}$ with good definition are achievable. The main advantages of EBL pattern generation are:

- Patterns are derived directly from digital data.



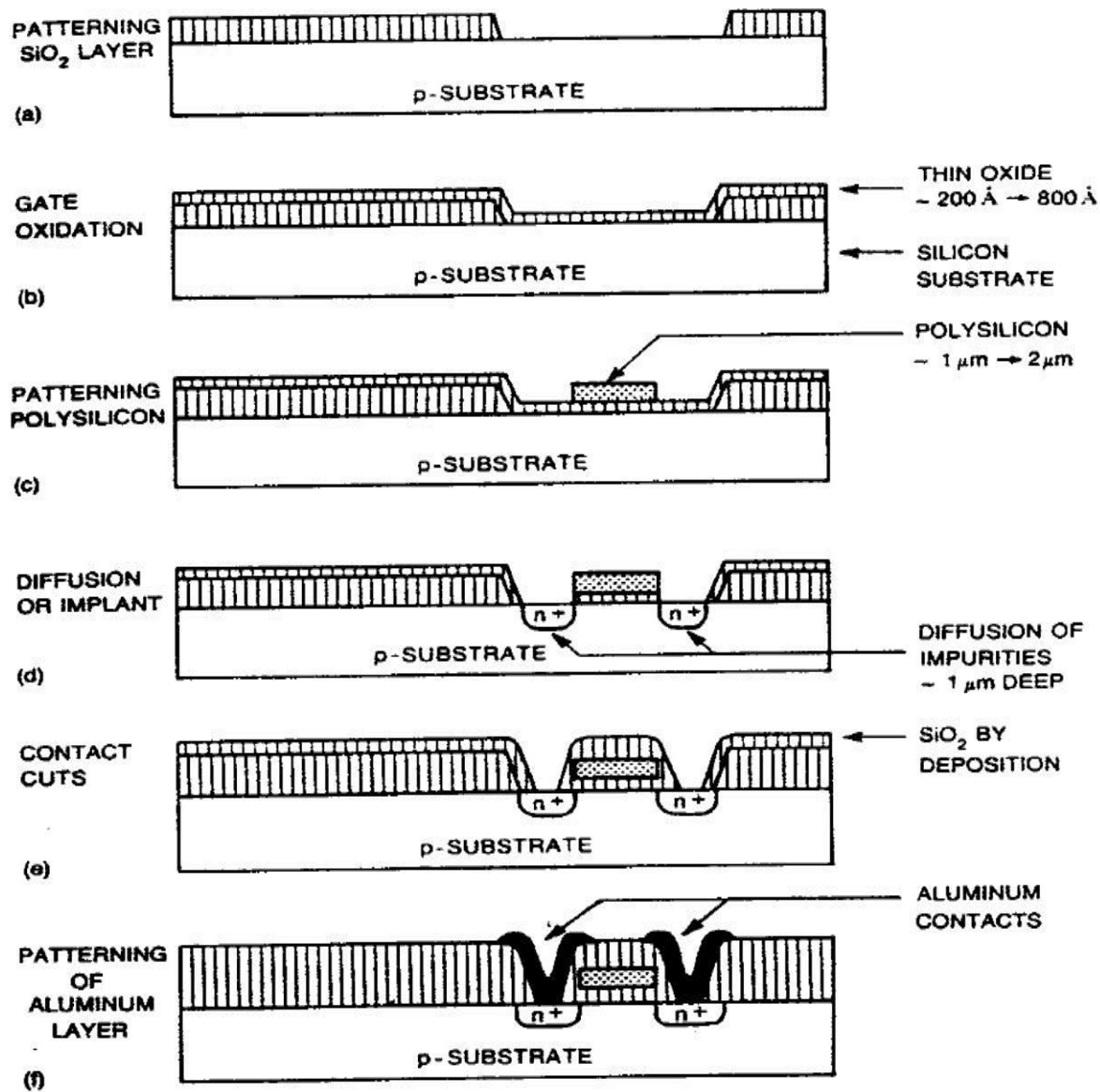
- There are no intermediate hardware images such as recticles or masks; that is, the process may be direct.
- Different patterns may be accommodated in different sections of the wafer without difficulty.
- Changes to patterns can be implemented quickly.

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).



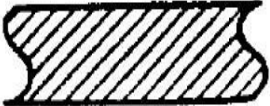






The steps involved in a typical silicon gate process entail photomasking and oxide etching, which are repeated a number of times during the processing sequence. Fig. 3.4 shows the processing steps after the initial patterning of the SiO_2 , which was shown in Fig. 3.3. The wafer is initially covered 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

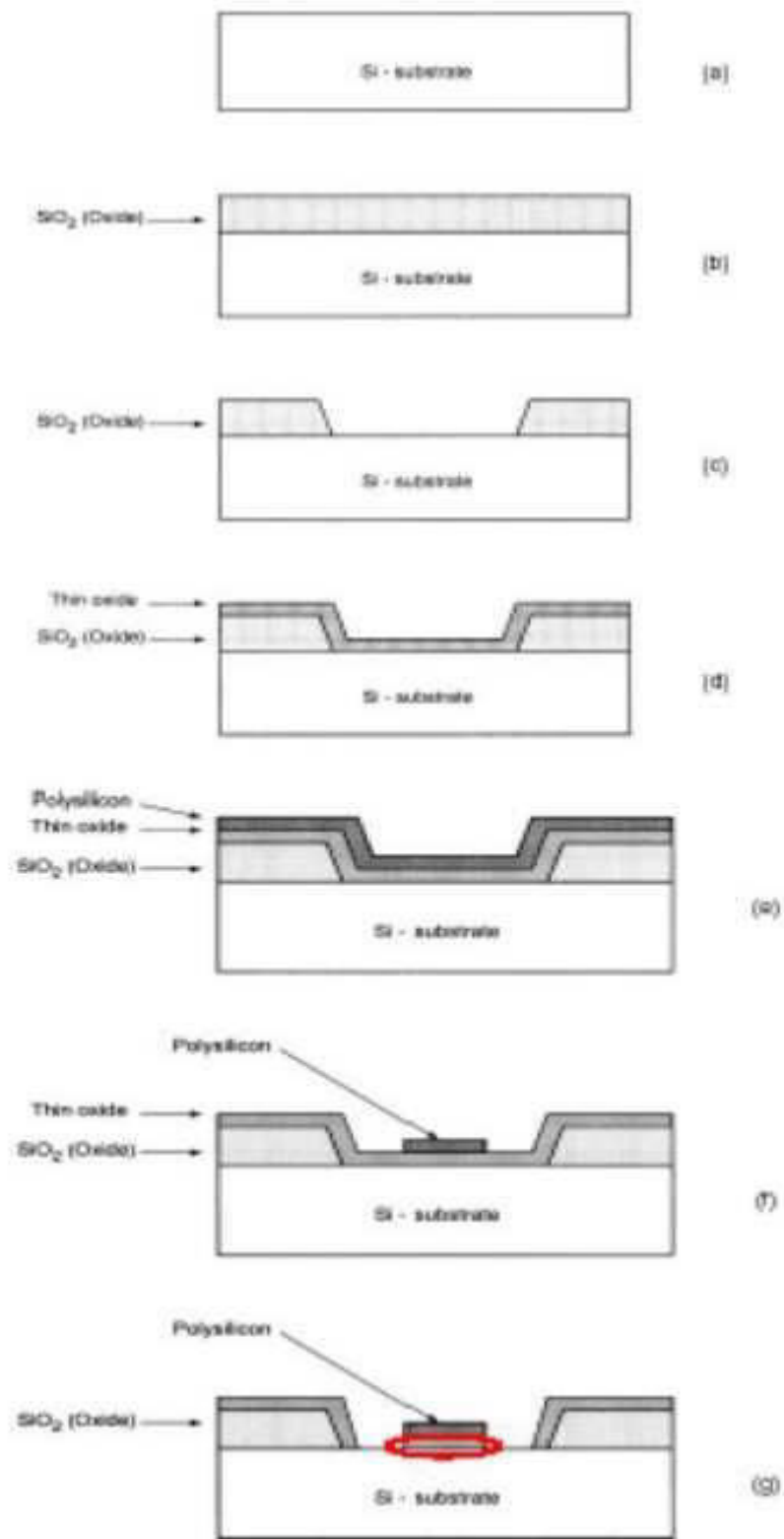


junctions are formed in the substrate and the polysilicon is doped with the particular type of dopant. This reduces the resistivity of the polysilicon. Note that the diffusion junctions form the drain and source of the MOS transistor. They are formed only in regions where the polysilicon gate does not 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:

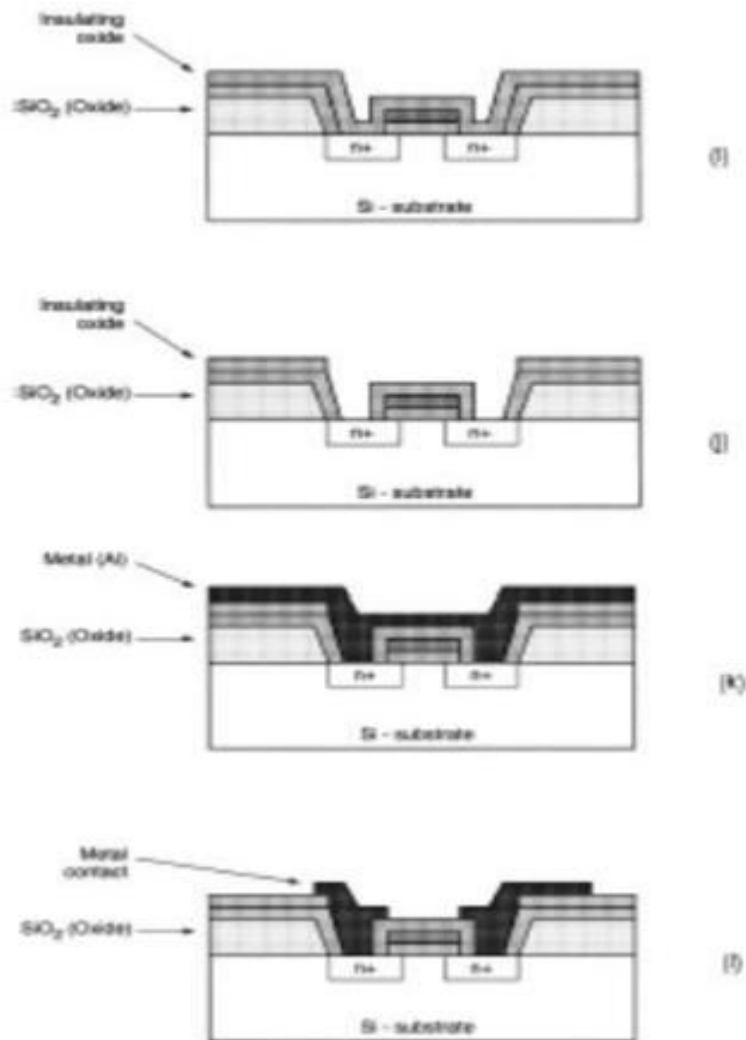


Figure8. NMOS Fabrication process steps

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:

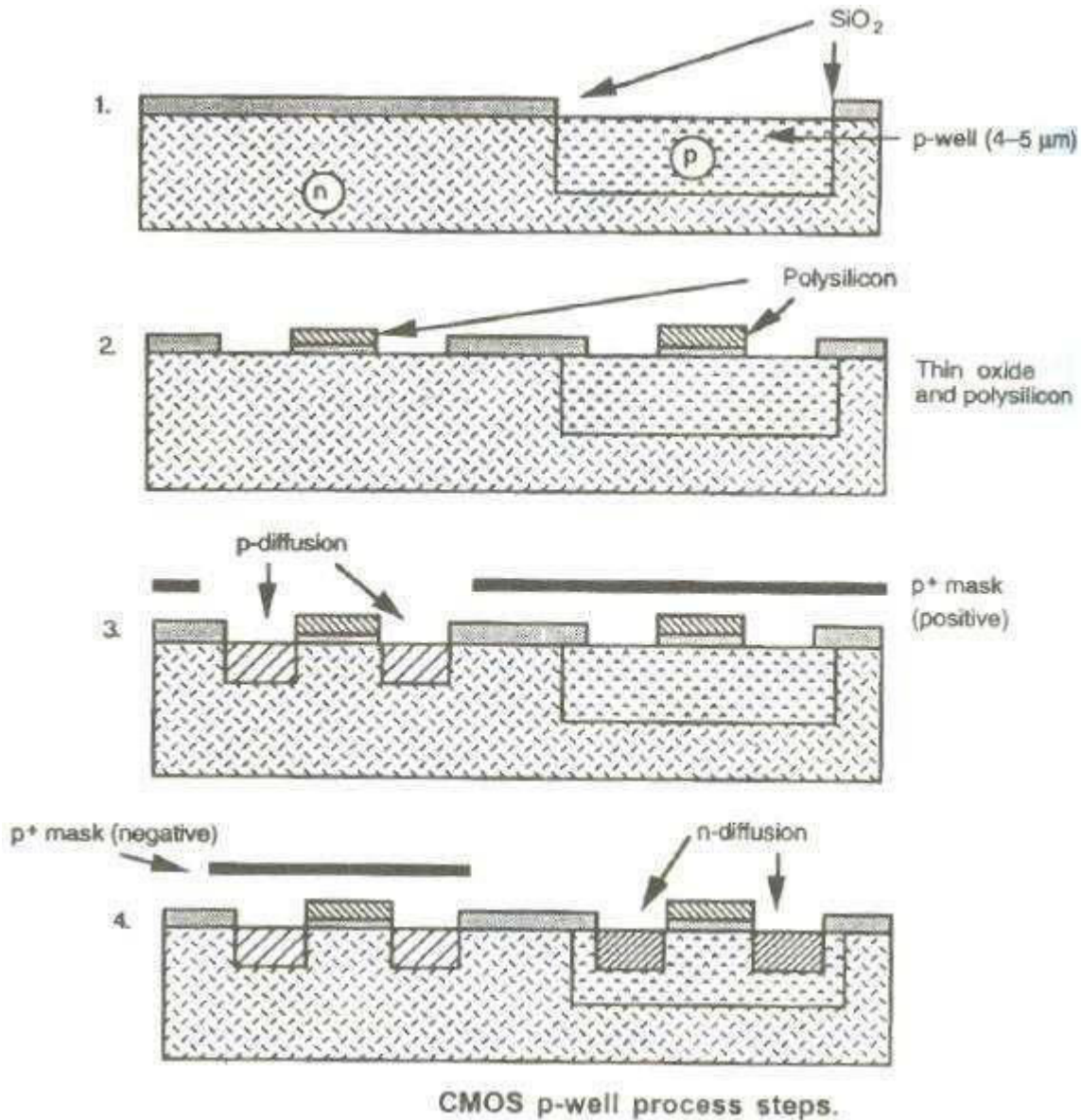


Figure9. CMOS Fabrication (P-WELL) process steps.

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 the 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: Ap+ 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.

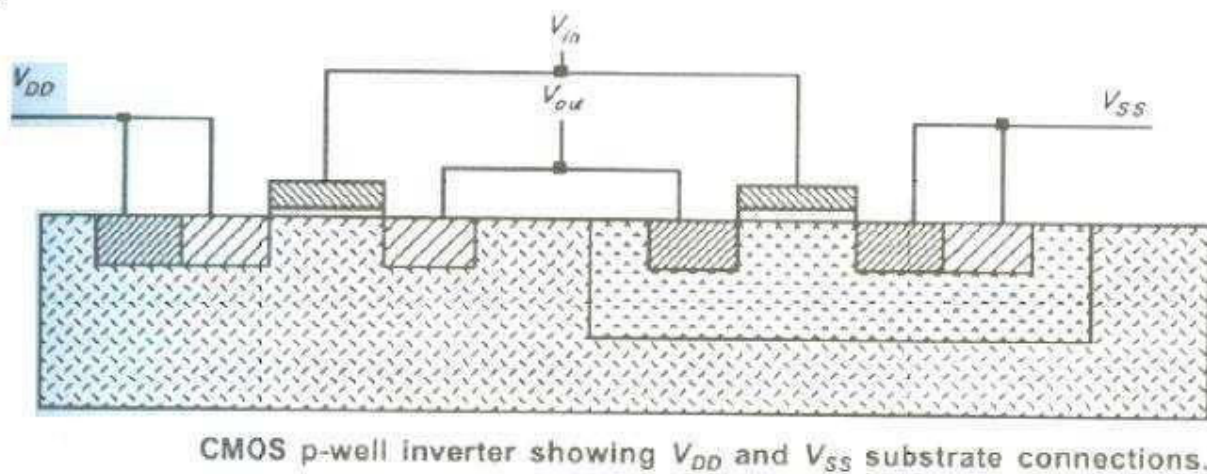


Figure 10. CMOS inverter (P-WELL)

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 10^{15} 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.

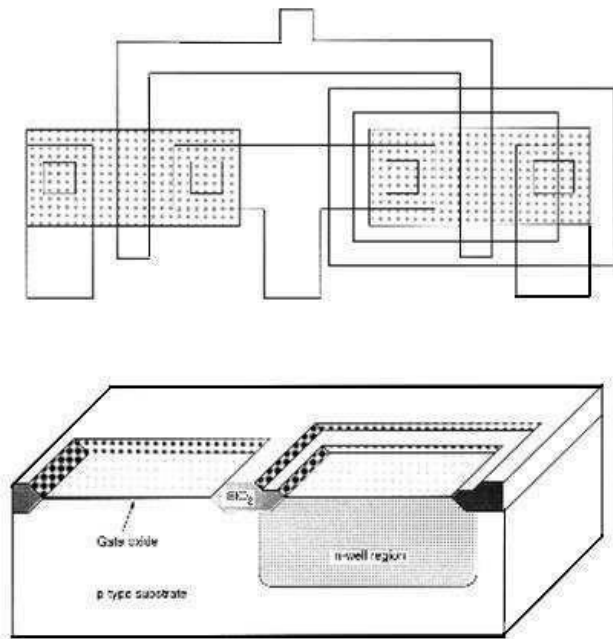


Figure-11.1: Cross sectional view

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.

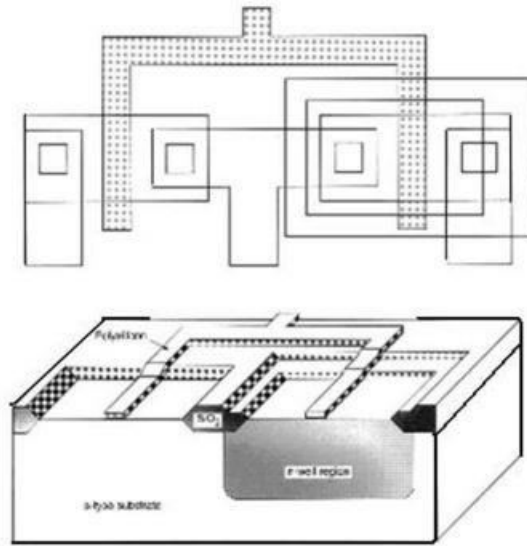


Figure-11.2: Cross sectional view

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 MOS 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.

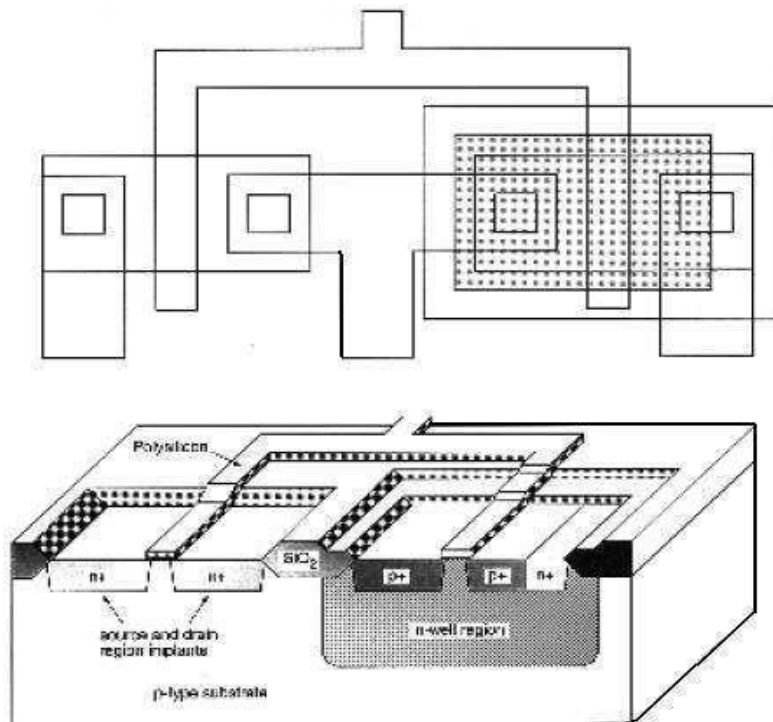


Figure-11.3: Using a set of two masks, the n+ and p+ regions are implanted into the substrate and into the n-well, respectively. Also, the ohmic contacts to the substrate and to the n-well are implanted in this process step.

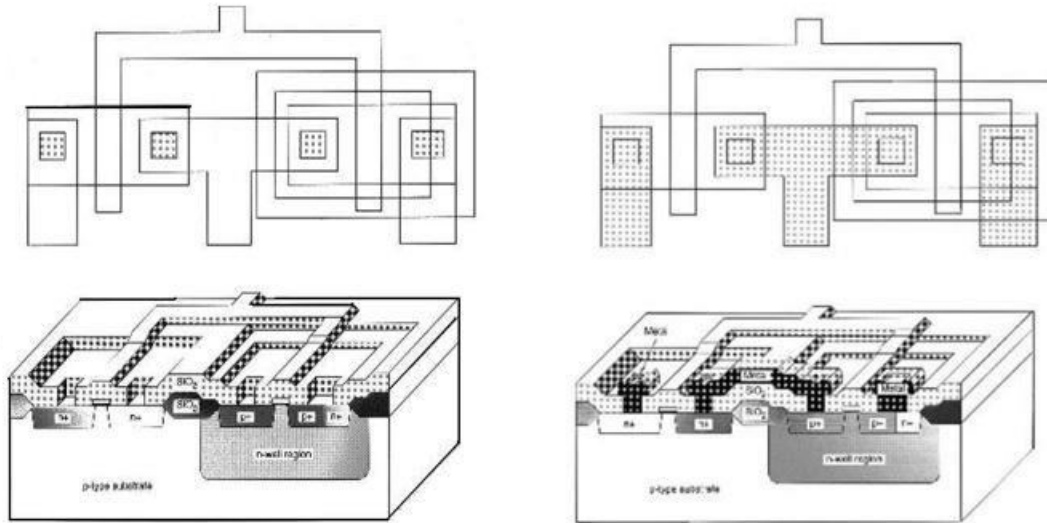


Figure-11.4: An insulating silicon dioxide layer is deposited over the entire wafer using CVD. Then, the contacts are defined and etched away to expose the silicon or polysilicon contact windows. These contact windows are necessary to complete the circuit interconnections using the metal layer, which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

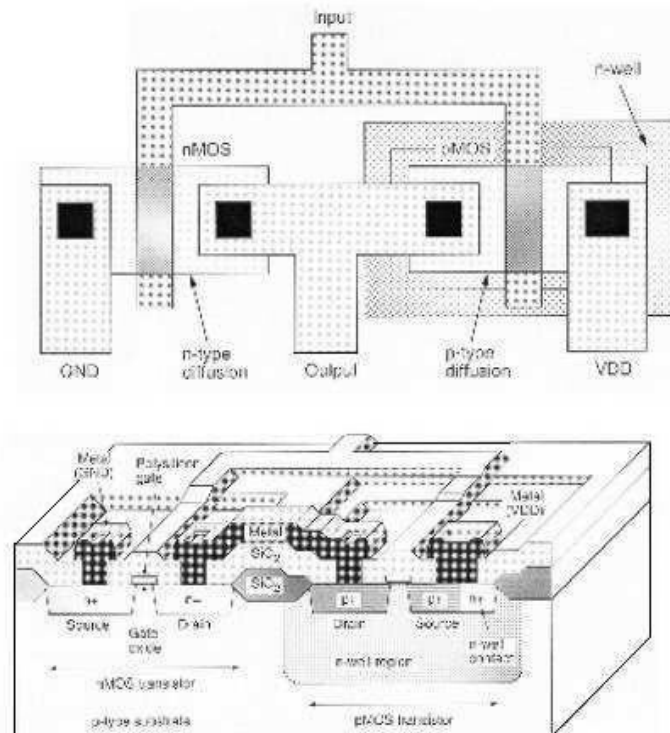


Figure-11.6: The composite layout and the resulting cross-sectional view of the chip, showing one nMOS and one pMOS transistor (built-in n-well), the polysilicon and metal interconnections. The final step is to deposit the passivation layer (for protection) over the chip, except for wire- bonding pad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

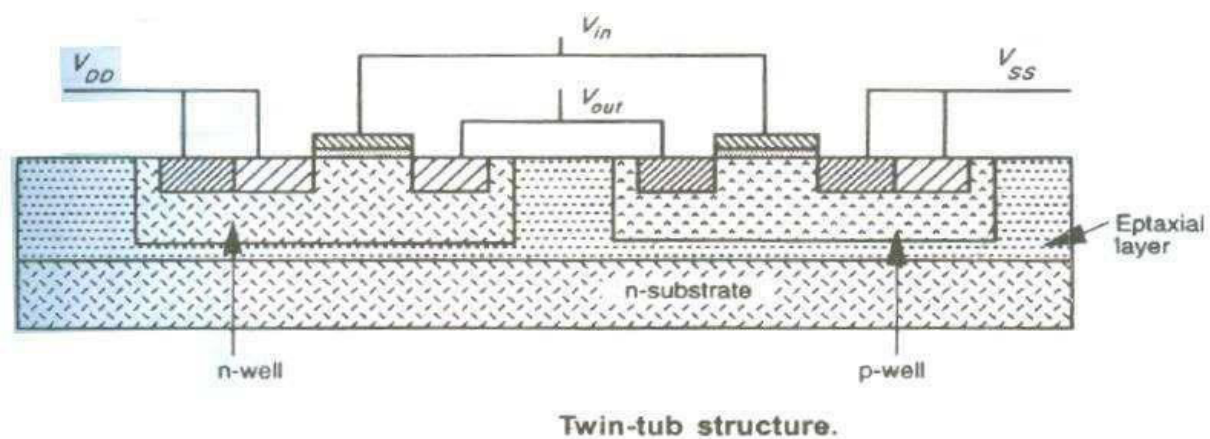


Figure 12 CMOS twin-tub inverter.

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. Foreexample: 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*		ND
RED		Polysilicon		NP
BLUE		Metal 1		NM
BLACK		Contact cut		NC
GRAY	NOT APPLICABLE	Oxerglass		NG
nMOS ONLY YELLOW		Implant		NI
nMOS ONLY BROWN		Buried contact		NB
FEATURE	FEATURE (STICK)	FEATURE (SYMBOL)	FEATURE (MASK)	
n-type enhancement mode transistor				
Transistor length to width ratio L:W should be shown.				
n-type depletion mode transistor nMOS only				
Source, drain and gate 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 1 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	GIF LAYER
GREEN	Encoding as in Color plate 1(a)	n-diffusion (n ⁺ active) Thinox*	* Thinox = n-diff. + p-diff. + transistor channels	CAA or CNA
RED		Polysilicon	Encoding as in Color plate 1(e)	CPF
BLUE		Metal 1		CMF
BLACK		Contact-out		CC
GRAY		Overglass		COG
YELLOW (STICK)	 green outline here for clarity	p-diffusion (p ⁺ active)		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		CC
FEATURE	FEATURE (STICK)	FEATURE (SYMBOL)	FEATURE (MASK)	
n-type enhancement mode transistor (as in Color plate 1(a)) Transistor length to width ratio L/W may be shown.	 Demarcation line			
p-type enhancement mode transistor Note: p-type transistors are placed above and n-type below the demarcation line.	 Demarcation line	 S G D	 S G D p+ mask	

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
ORANGE	MONOCHROME	Polysilicon 2	MONOCHROME	CPS
SEE COLD PLATE 1(e)		Bipolar n-p-n transistor	See Figure 3-19(f)	Not applicable
PINK	Not separately encoded	p-base of bipolar n-p-n transistor		CBA
PALE GREEN	Not separately encoded	Buried collector of bipolar n-p-n transistor		CCA
FEATURE	FEATURE (STICK) (MONOCHROME)	FEATURE (SYMBOL) (MONOCHROME)	FEATURE (MASK) (MONOCHROME)	
<i>n</i> -type enhancement poly 2 transistor				
Transistor length to width ratio L:W may be shown.				
<i>p</i> -type enhancement poly 2 transistor				
Note: <i>p</i> -type transistors are placed above and <i>n</i> -type transistors below the demarcation line				
non bipolar transistor				See Figure 3-19(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:

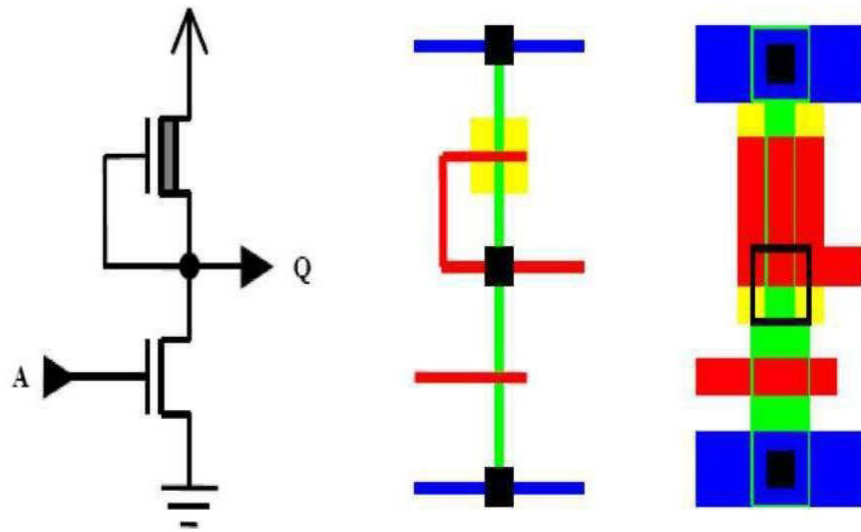
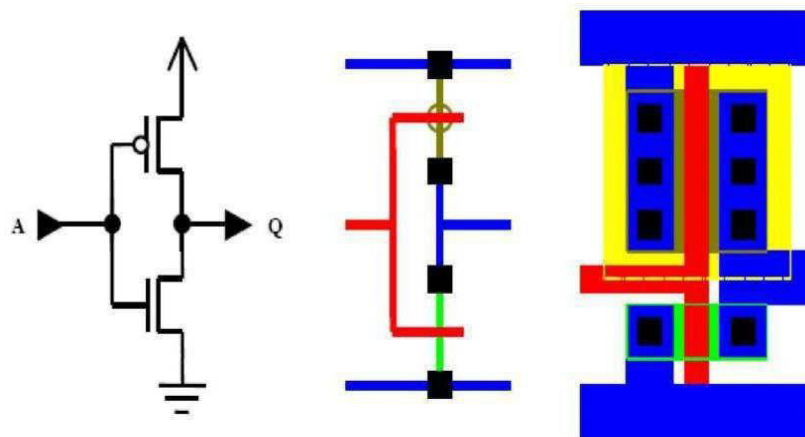


Figure 4: nMOS depletion load inverter.

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.

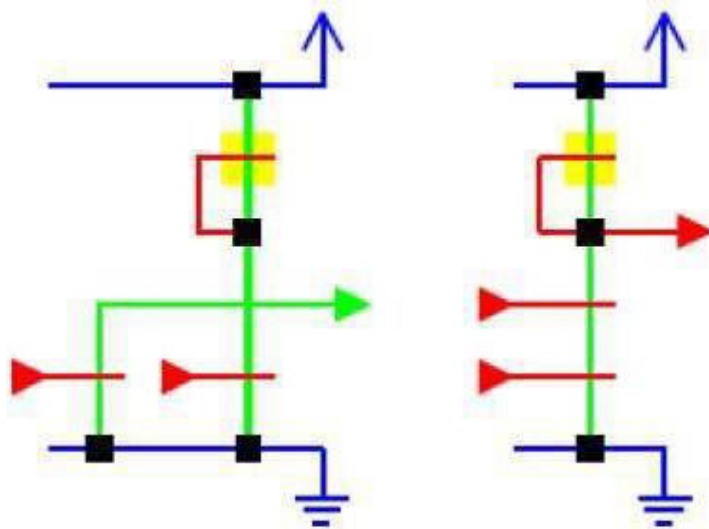


Figure 6 shows the stick diagrams for nMOS NOR and NAND.

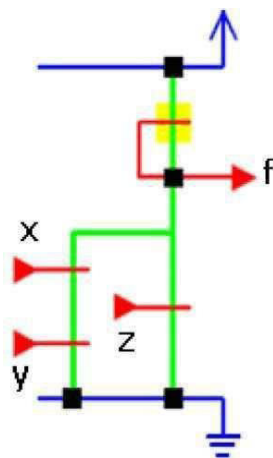


Figure 7: stick diagram of a given function f .

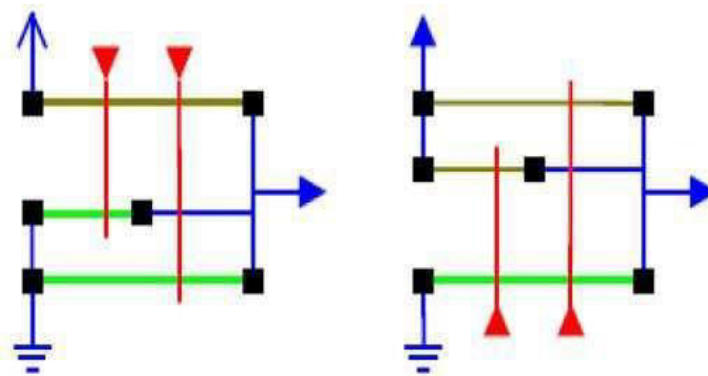


Figure 7

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 line are the n transistors. Following stick shows CMOS circuit example in monochrome where we utilize the demarcation line.

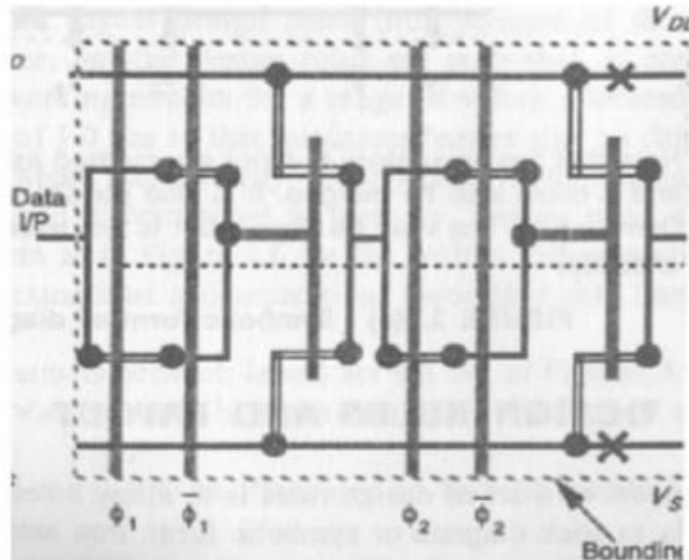


Figure 9: stick diagram of dynamic shift register in CMOS style.

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.

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 $2X$. 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 $2X$, and a minimum spacing of $3X$. If we select a $2\text{ }\mu\text{m}$ technology (i.e., $X = 1\text{ }\mu\text{m}$), the above rules are translated to a minimum width of $2\text{ }\mu\text{m}$ and a minimum spacing of $3\text{ }\mu\text{m}$. On the other hand, if a $1\text{ }\mu\text{m}$ technology (i.e., $X = 0.5\text{ }\mu\text{m}$) is selected, then the same width and spacing rules are now specified as $1\text{ }\mu\text{m}$ and $1.5\text{ }\mu\text{m}$, respectively.

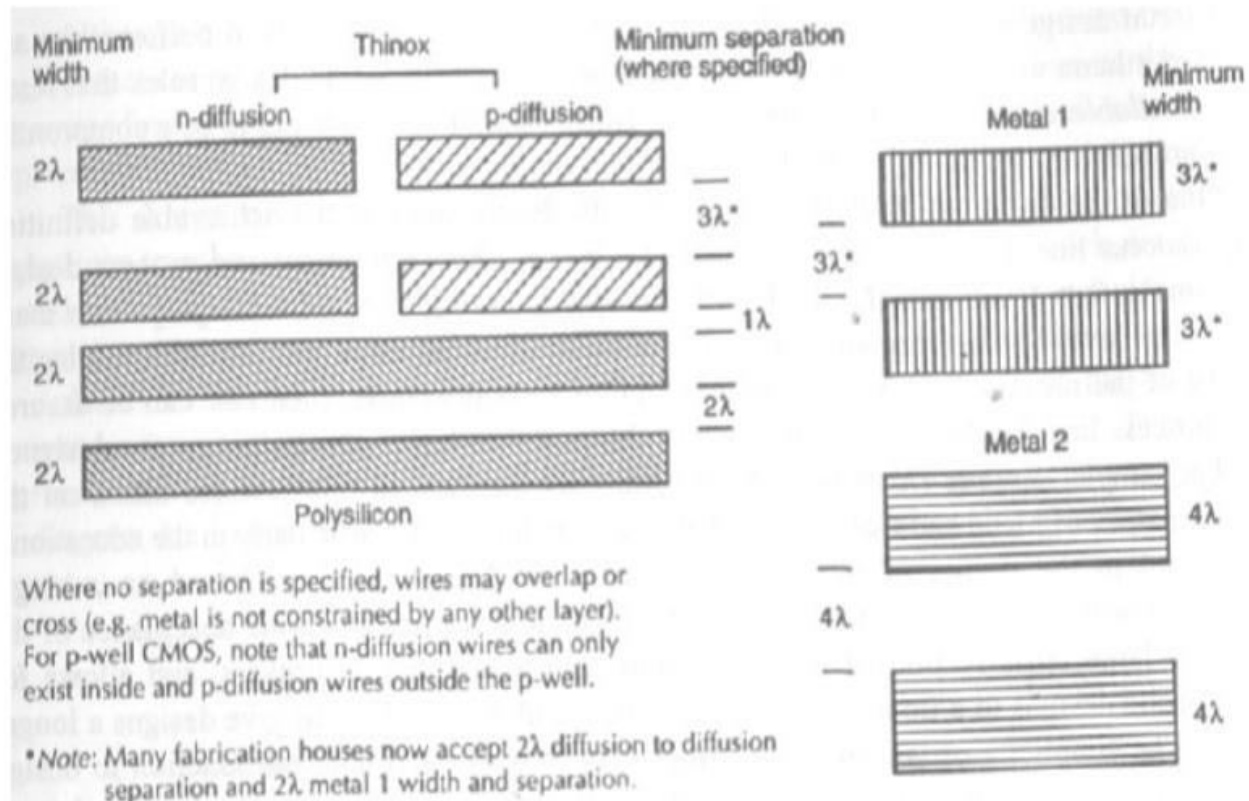


Figure 10: Design rules for the diffusion layers and metal layers.

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

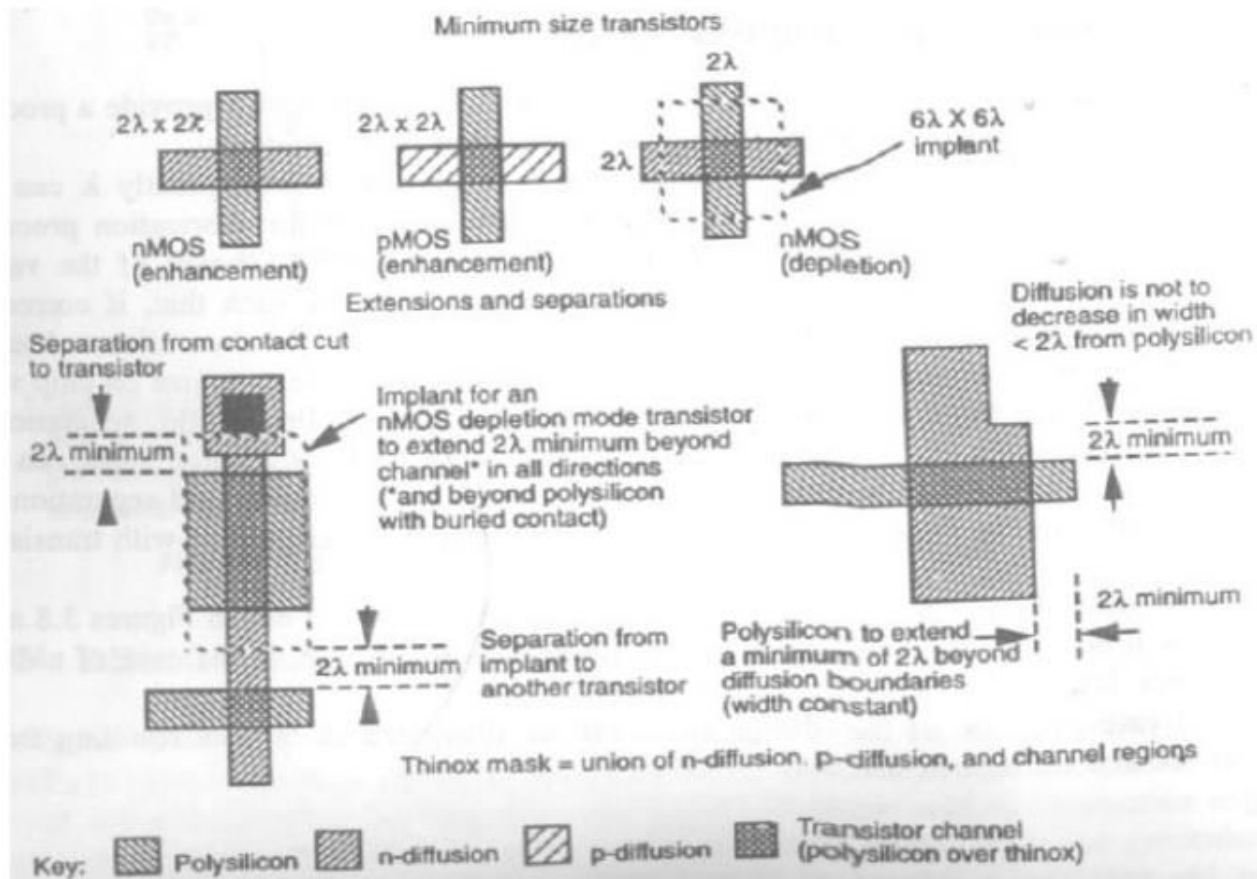


Figure 11: Design rules for transistors and gate over hang distance.

Figure shows the design rule for the transistor, and it also shows that the poly should extend for a minimum of 7λ 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.

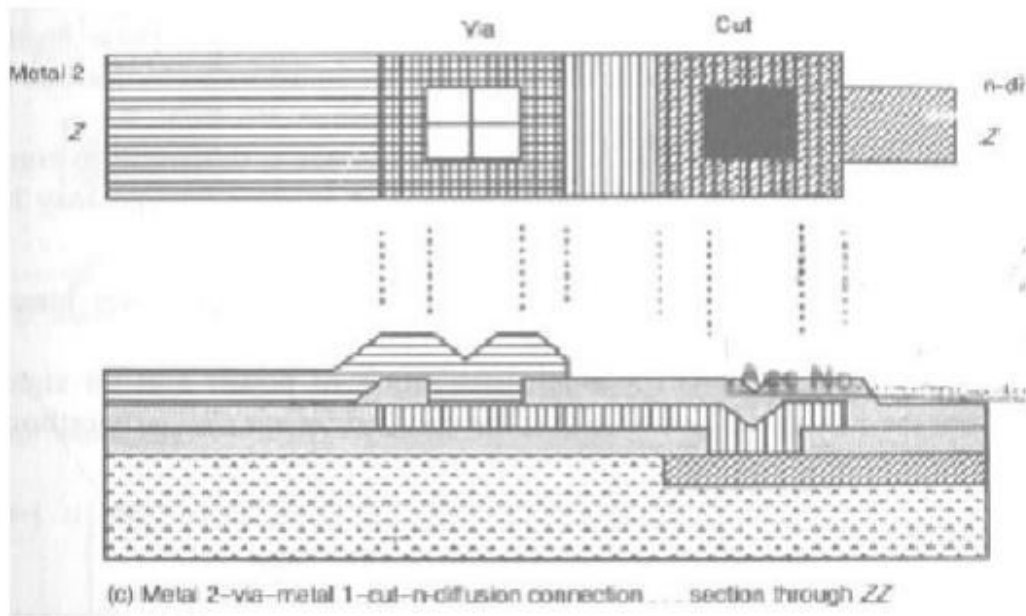
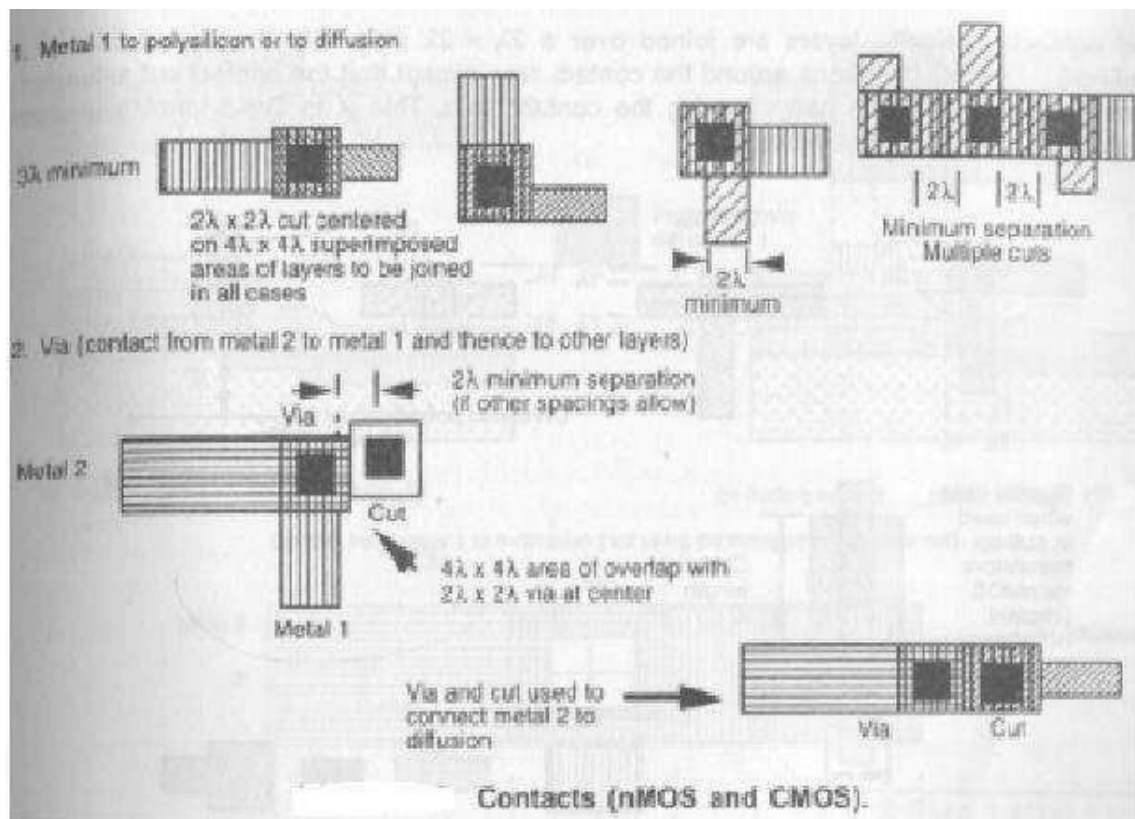


Figure 12: cross section showing the contact cut and via

Figure shows the design rules for contact cuts and Vias. The design rule for contact



is minimum $2\lambda \times 2\lambda$ and same is applicable for a Via.

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.

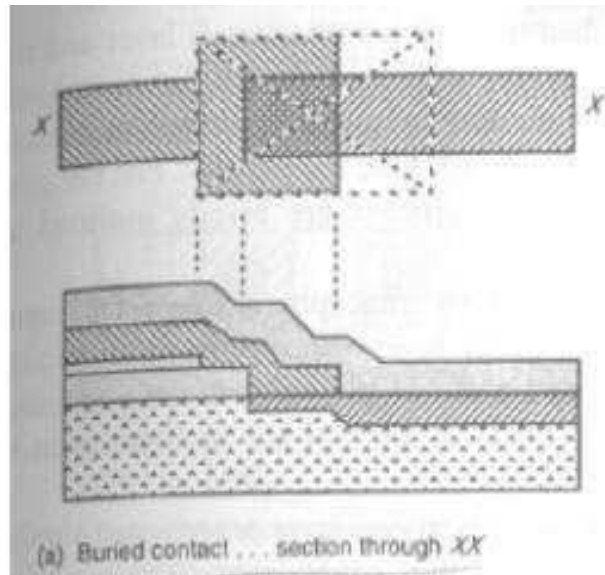


Figure 14: Buried contact.

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

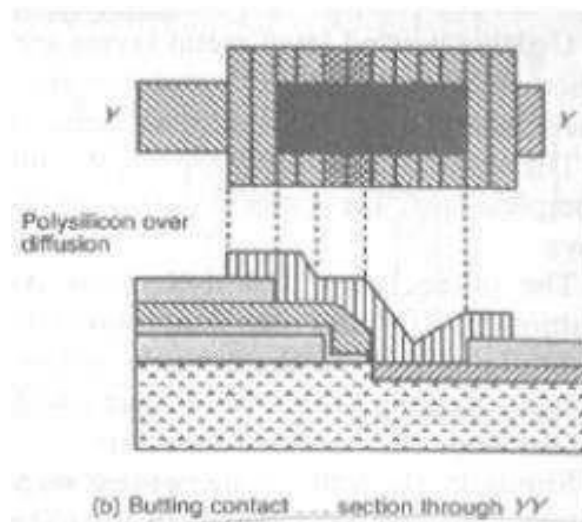


Figure 15: Butting contact.

CMOS LAMBDA BASED DESIGN RULES:

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

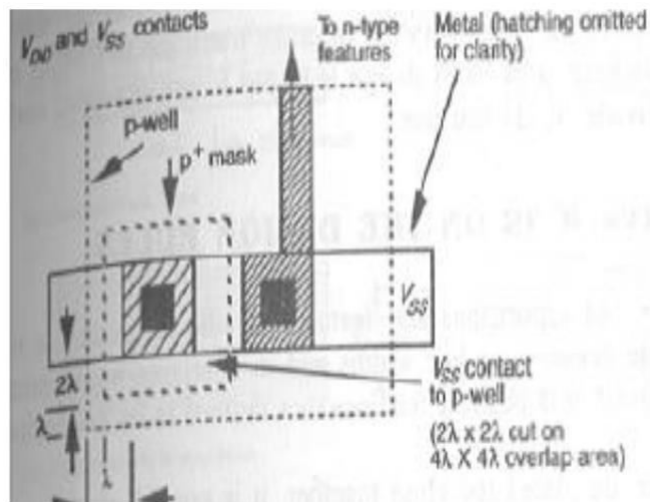


Figure 16: CMOS design rules.

1 Orbit 2 μ m CMOS process:

In this process all the spacing between each layers and dimensions will be in terms micrometer. The 2 μ m here represents the feature size. All the design rules 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.

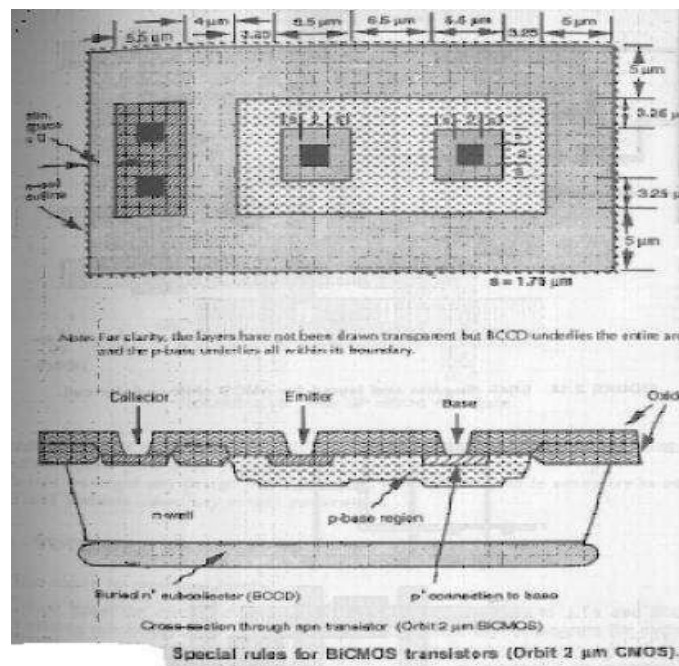


Figure 17: BiCMOS design rules.

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

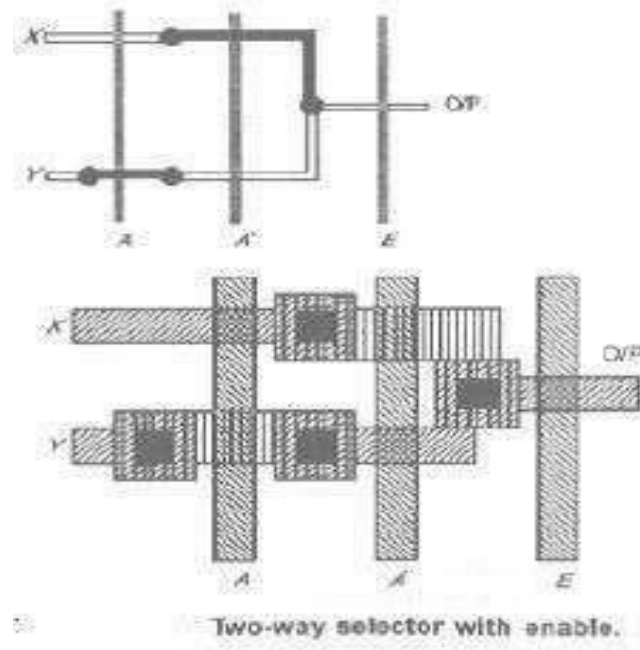


Figure 18: Two way selector stick and layout

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.

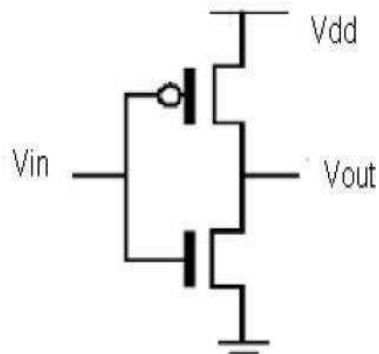
The simplest way to begin a layout representation is to draw the stick diagram. But as the complexity increases it is not possible to draw the stick diagrams. For beginners it is easy to draw the stick diagram and then proceed with the layout for the basic digital gates. We will have a look at some of the things we should know before starting the layout. In the schematic representation lines drawn between device terminals represent interconnections and any non-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 is 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 at least 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



TOWARDS THE LAYOUT

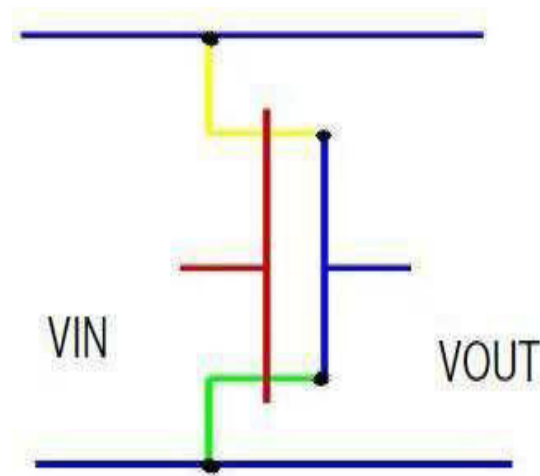


Figure 20: Stick diagram of inverter.

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 $V_{in} = 1$, Nmos (green) is on it pulls V_{out} to V_{ss} , 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.

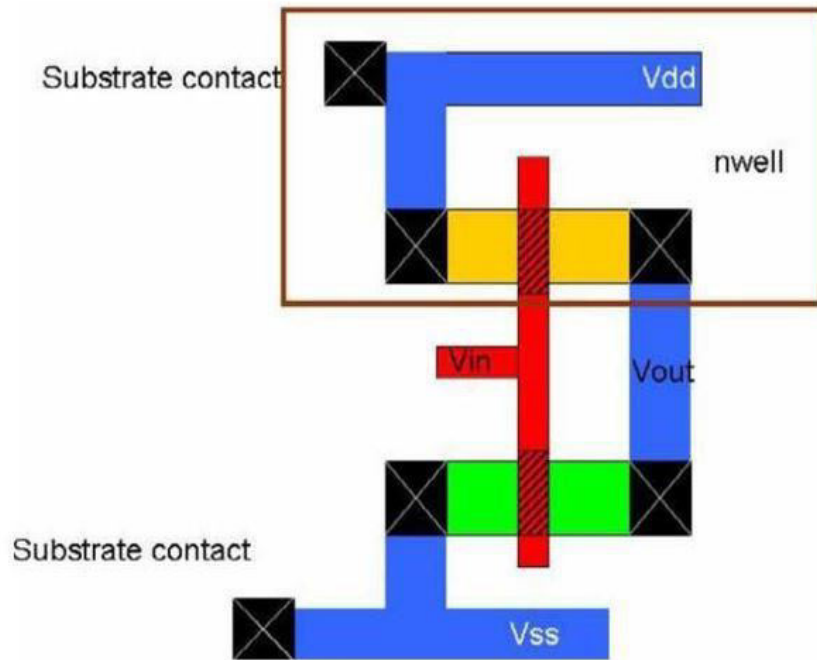


Figure 21: Layout of inverter.

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

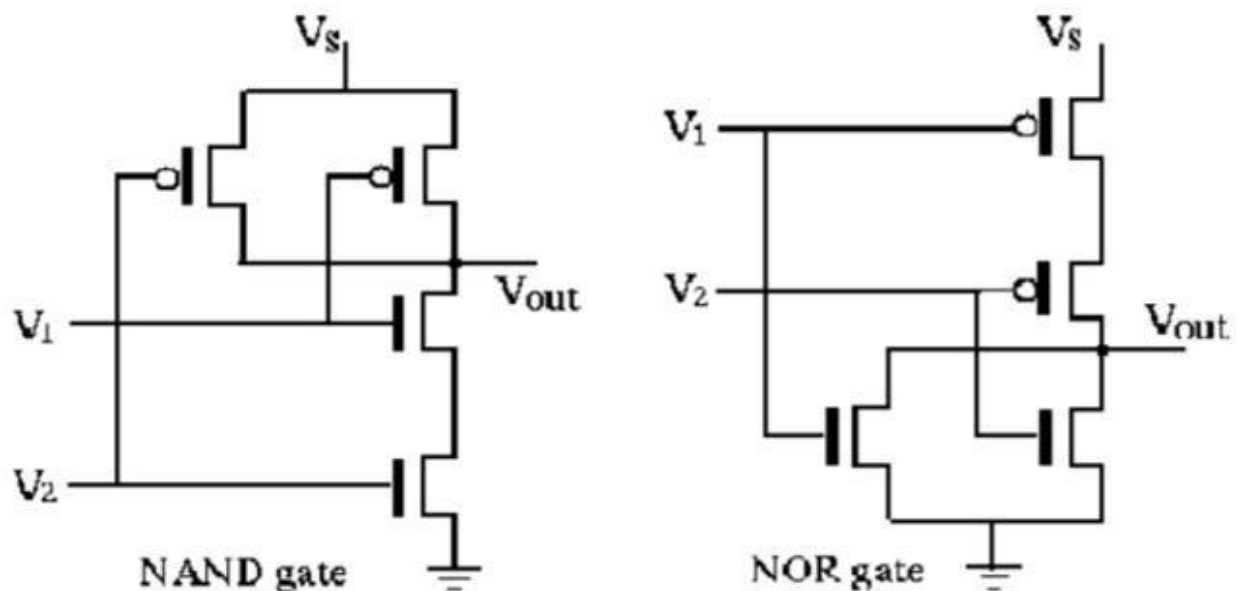
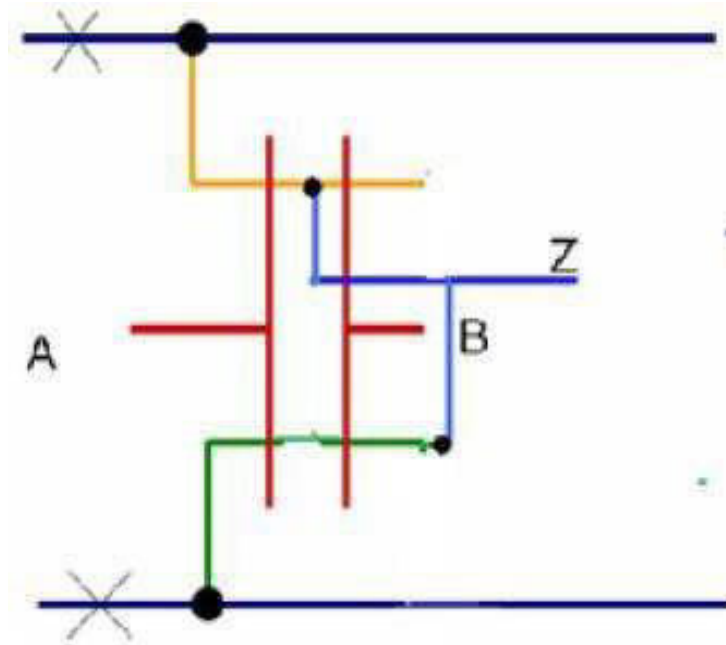


Figure 22: Schematic diagrams of nand and nor gate

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.

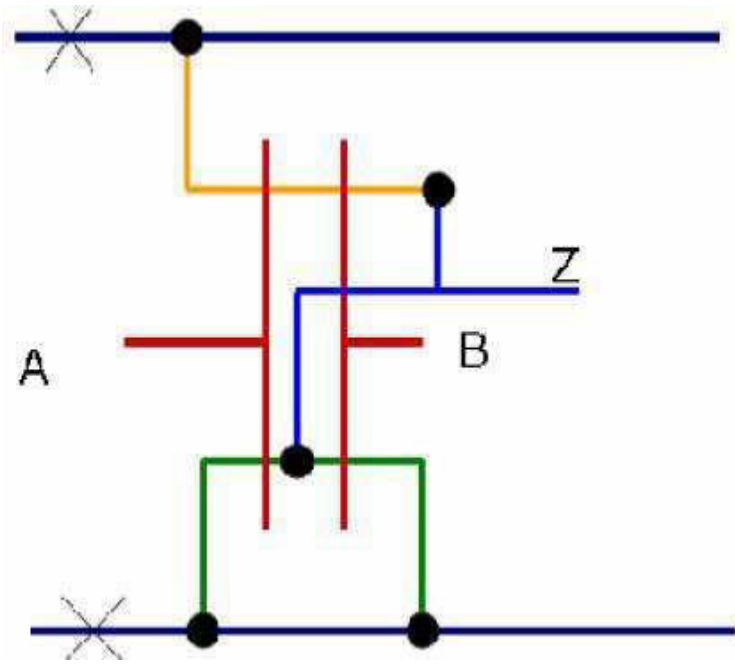
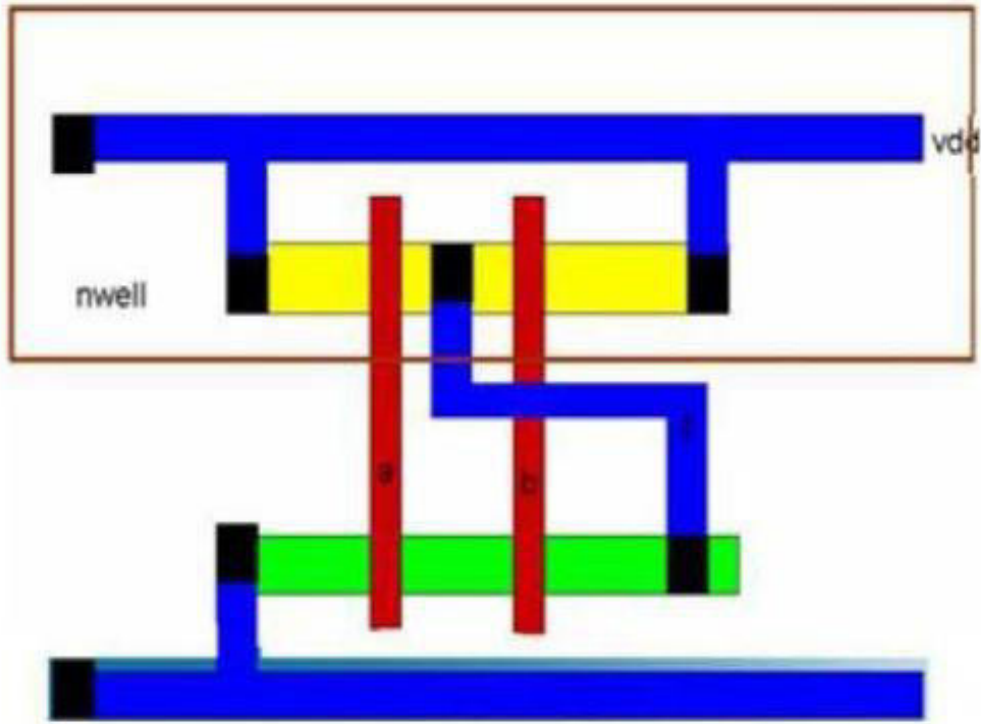


Figure 24: Layout of nand gate.

Figure 25: Stick diagram of nor gate.

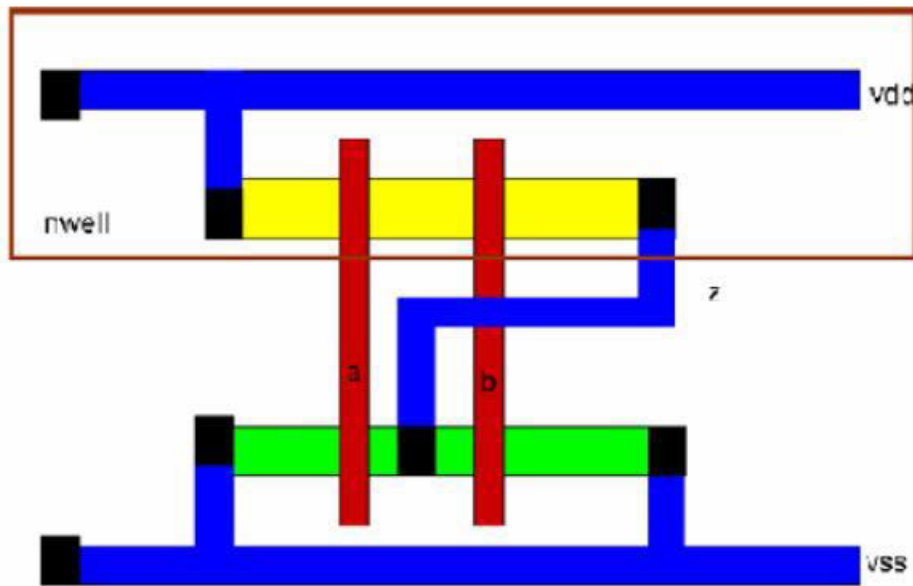


Figure 26: Layout of nor gate.

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 polysilicon 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 proceed towards optimization. Here optimization is at transistor level rather than 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.

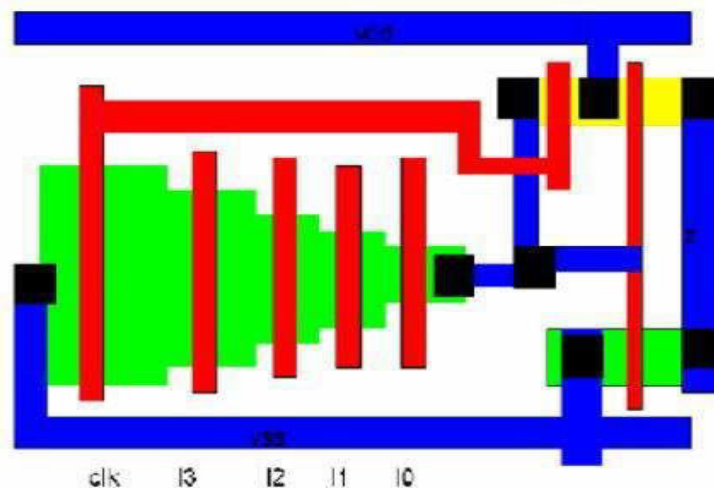
The factors are

- I. 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. Transparent routing can be provided for cell to cell interconnection, this reduces global wiring problems

LAYOUT OPTIMIZATION FOR PERFORMANCE

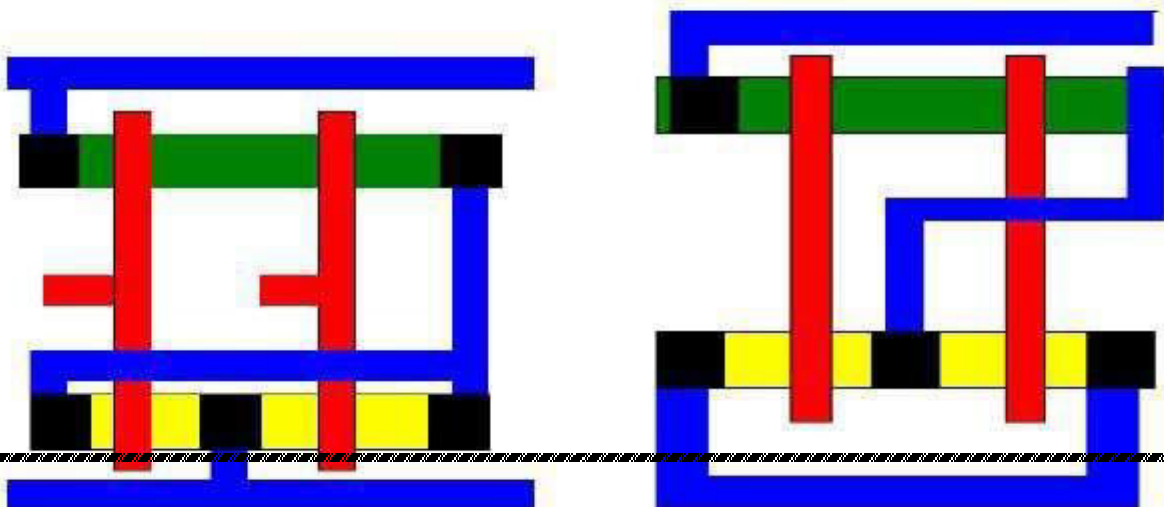
1. Vary the size of the transistor according to its position in series. The transistor closest to the output is the smallest. The transistor nearest to the VSS line is the largest. This helps in increasing the performance by 30 %. A three input nand gate with the varying size is shown



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



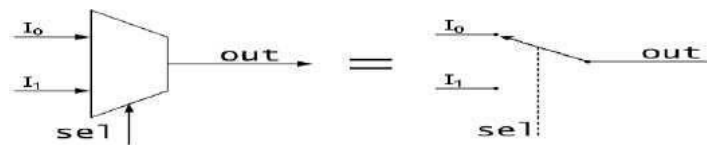
try and reduce the number of drains in parallel i.e. wherever possible we must try and connect drains in series at least at the output. This arrangement could reduce the capacitance at the output enabling good voltage levels. One example is as shown next.

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

MULTIPLEXER:

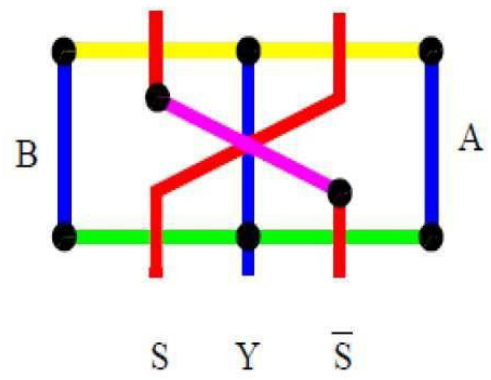
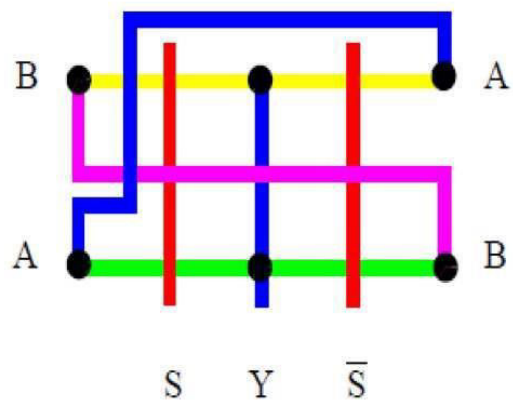
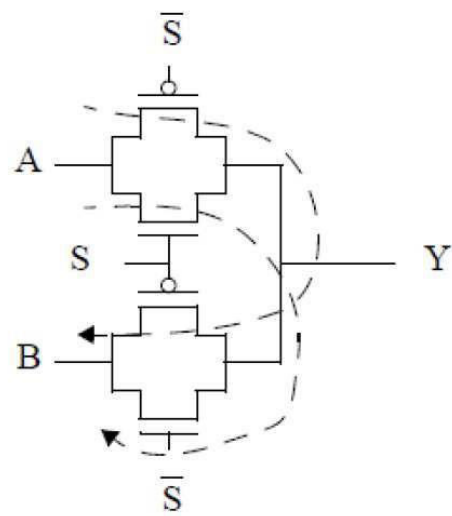
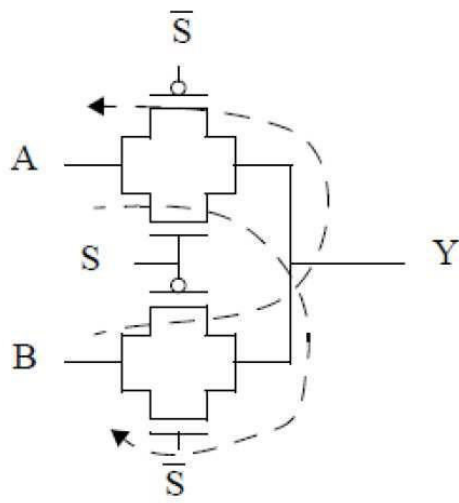
In electronics, a multiplexer or mux (occasionally the terms muldex or muldem are also found for a combination multiplexer-demultiplexer) is a device that performs multiplexing; it selects one of many analog or digital input signals and forwards the selected input into a single line. A multiplexer of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



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 $1/\beta$ and $1/\alpha$. $1/\beta$ is chosen as the scaling factor for V_{dd} , gate oxide thickness D . $1/\alpha$ 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 A_g

$A_g = L \times W$. Since L & W are scaled down by $1/\alpha$. A_g is scaled down by $1/\alpha^2$

2. Gate capacitance per unit area

$C_o = \epsilon_o/D$, permittivity of SiO_2 cannot be scaled, hence C_o can be scaled $1/1/\beta = \beta$

3. Gate capacitance C_g

$C_g = C_{ox}A = C_{ox}LxW$. Therefore C_g can be scaled by $\beta x 1/\alpha x 1/\alpha = \beta/\alpha^2$

4. Parasitic capacitance

$C_x = A_x/d$, where A_x is the area of the depletion around the drain or source. d is the depletion width. A_x is scaled down by $1/\alpha^2$ and d is scaled by $1/\alpha$. Hence C_x is scaled by

$$1/\alpha^2 / 1/\alpha = 1/\alpha$$

5. Carrier density in the channel Q_{on}

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

C_o is scaled by β and V_{gs} is scaled by $1/\beta$, hence Q_o is scaled by $\beta x 1/\beta = 1$.

Channel resistance R_o

$R_{on} = L/Wx 1/Q_{ox}\mu$, μ is mobility of charge carriers. R_o is scaled by $1/\alpha / 1/\alpha x 1 = 1$

Gate delay T_d

T_d is proportional to R_o and C_g

$$T_d \text{ is scaled by } 1x \beta/\alpha^2 = \beta/\alpha^2$$

Maximum operating frequency f_o

$f_o = 1/t_d$, therefore it is scaled by $1/\beta/\alpha^2 = \alpha^2/\beta$

Saturation current

$I_{dss} = C_o\mu W(V_{gs} - V_t)/2L$, C_o scale by β and voltages by $1/\beta$, I_{dss} is scaled by $\beta/\beta^2 = 1/\beta$

Current Density

$J = I_{dss}/A$ hence J is scaled by $1/\beta / 1/\alpha^2 = \alpha^2/\beta$

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

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

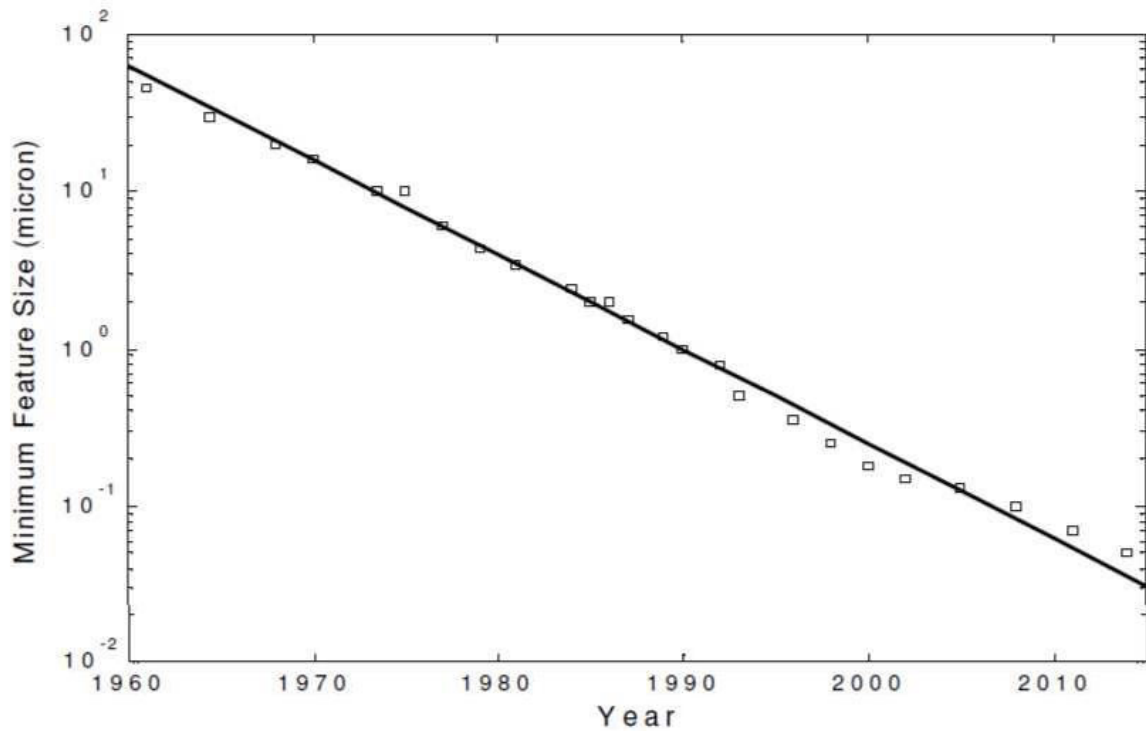


Figure-1: Technology Scaling (1)

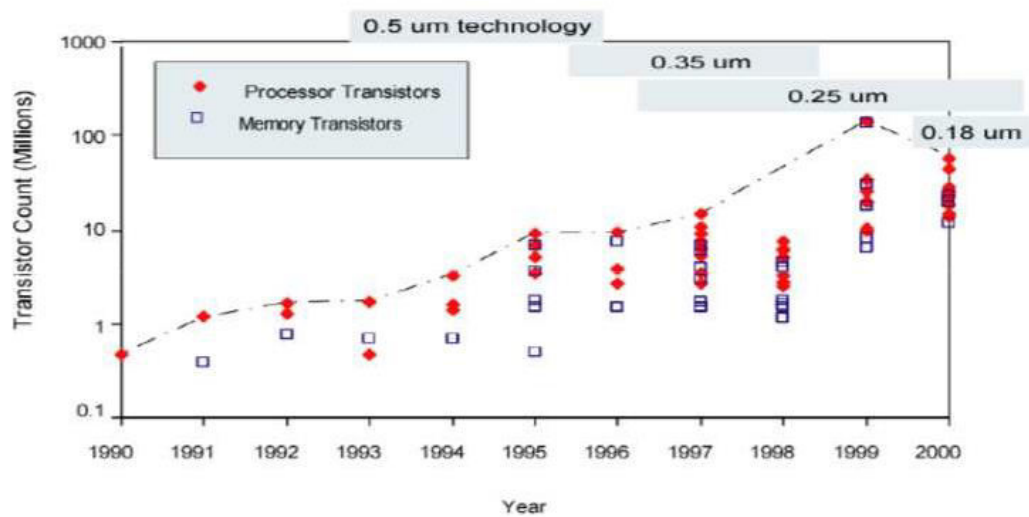


Figure-2: Technology Scaling (2)

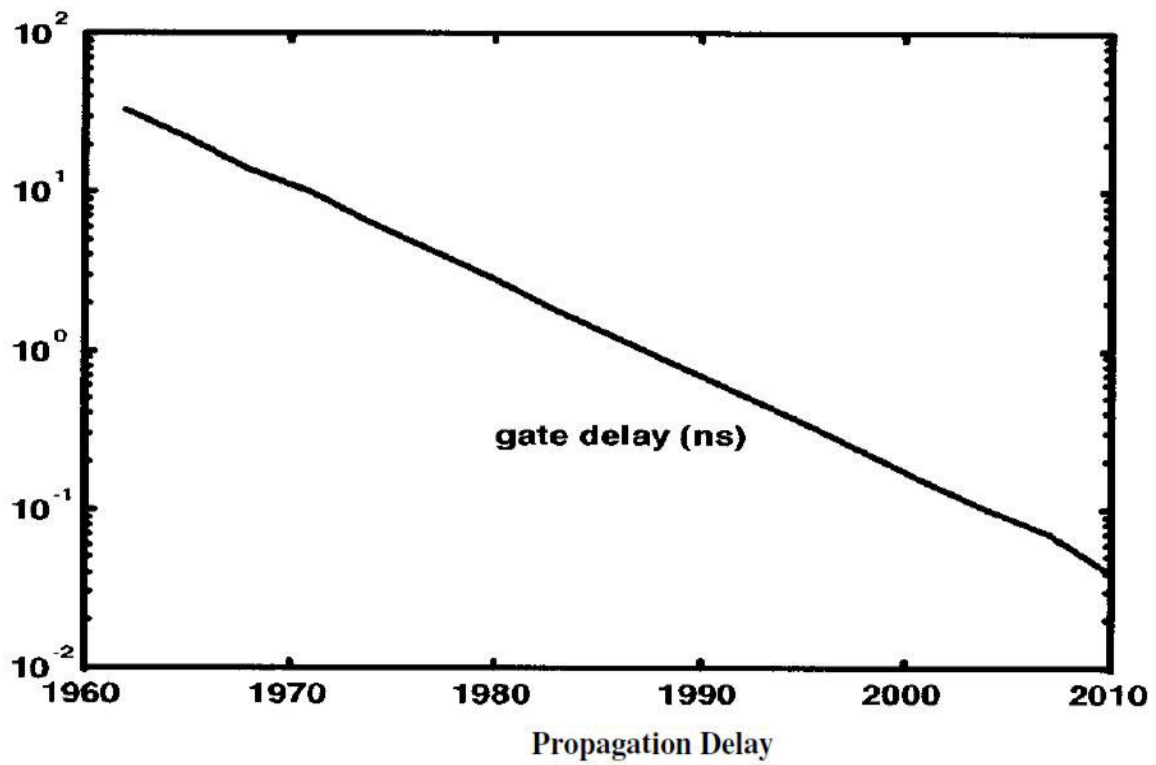


Figure-3:Technology Scaling (3)

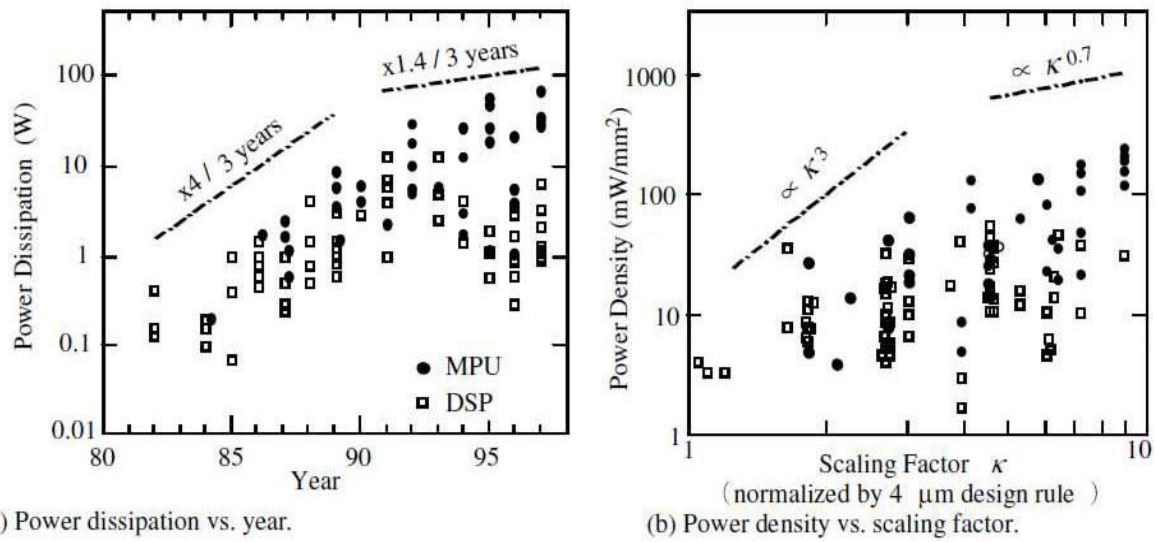


Figure-4:Technology Scaling (4)

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												
250 nm	-2	-1		1	2	3	4	5										
180 nm	-4	-3	-2	-1		1	2	3	4	5								
150 nm	-6	-5	-4	-3	-2	-1		1	2	3	4	5						
130 nm	-8	-7	-6	-5	-4	-3	-2	-1		1	2	3	4	5				
100 nm	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1		1	2	3	4	5	
70 nm				-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1		1	2	3
50 nm							-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	

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

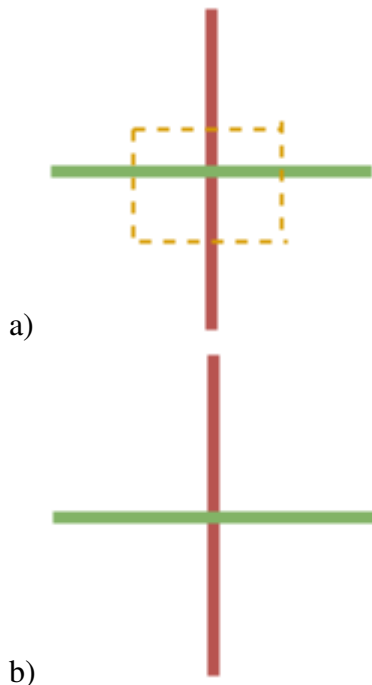
Table 1: ITRS

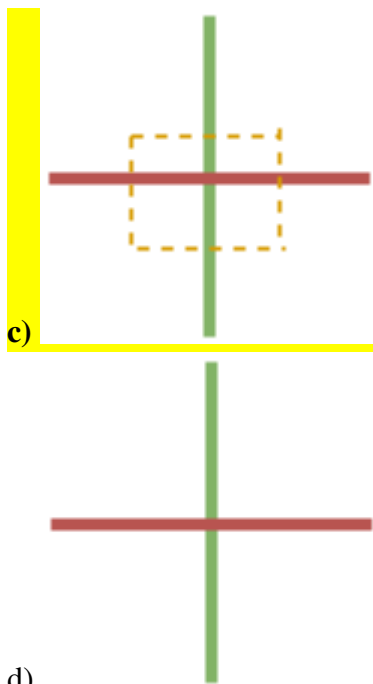
CONCLUSION:

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.

POST MCQ:

1. The isolated active areas are created by technique known as _____
 - a) Etched field-oxide isolation
 - b) Local Oxidation of Silicon
 - c) Etched field-oxide isolation or Local Oxidation of Silicon**
 - d) None of the mentioned
2. Implant is represented using _____
 - a) black, dark line
 - b) black, dotted line
 - c) yellow, dark line
 - d) yellow, dotted line**
3. How is nMOS depletion mode transistor represented?





- d)
4. The width of n-diffusion and p-diffusion layer should be?
 a) 3λ
b) 2λ
 c) λ
 d) 4λ
5. What should be the width of metal 1 and metal 2 layers?
 a) $3\lambda, 3\lambda$
 b) $2\lambda, 3\lambda$
c) $3\lambda, 4\lambda$
 d) $4\lambda, 3\lambda$

MOS CIRCUIT PERFORMANCE AND CMOS LOGIC CIRCUITS

PRE MCQ:

1. For 2 micron technology, what is the R_s value for polysilicon?
 - a) 10-40
 - b) 20-50
 - c) 15-30**
 - d) 15-100
2. What is the relationship between channel resistance and sheet resistance?
 - a) $R = R_s$
 - b) $R = Z * R_s$**
 - c) $R = Z/R_s$
 - d) $R = R_s/Z$
3. Z can be given as the ration of _____
 - a) lower channel by upper channel
 - b) upper channel by lower channel**
 - c) all of the mentioned
 - d) none of the mentioned
4. A feature size square has _____
 - a) $L > W$
 - b) $W > L$
 - c) $L = W$**
 - d) $L > d$
5. Relative area for $L = 20\lambda$ and $W = 3\lambda$ is?
 - a) 10
 - b) 15**
 - c) 1/15
 - d) 1/10

THEORY

SHEET RESISTANCE:

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

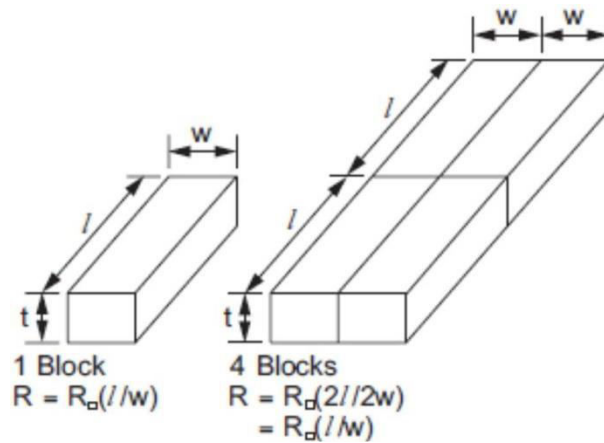
$$R = \frac{\rho l}{t w}$$

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

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

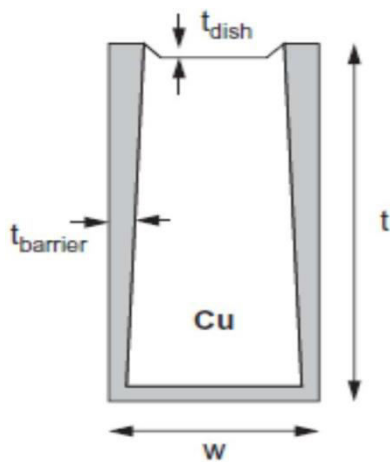
where $R = \rho / t$ is the sheet resistance and has units of ohms/square. Note that a square is a dimensionless quantity corresponding to a slab of equal length and width. This is convenient

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 ($\mu\Omega \cdot \text{cm}$)
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 t_{dish} , the resistance becomes,

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

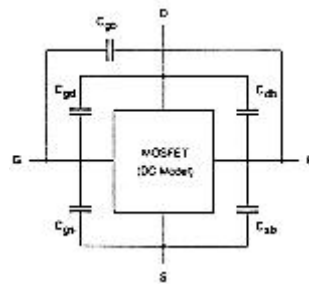
The resistivity of polysilicon, diffusion, and wells is significantly influenced by the doping levels. Polysilicon and diffusion typically have sheet resistances under 10 ohms/square when silicided and up to several hundred ohms/square when unsilicided. Wells have lower doping and thus even higher sheet 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 C_{gd} (gate-to-drain capacitance), C_{gs} (gate-to-source capacitance), and C_{gb} (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.

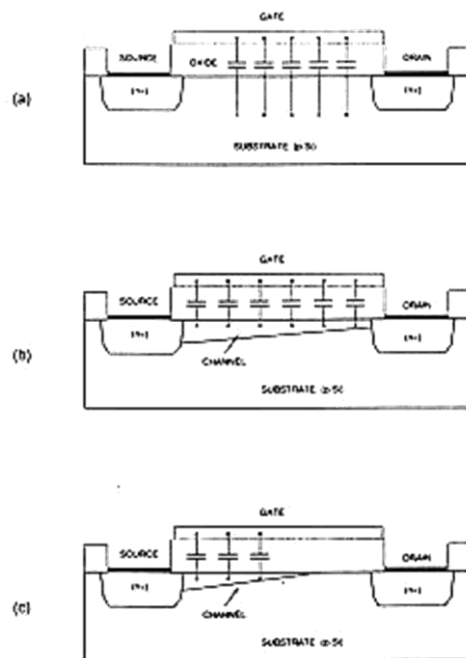


Figure: Schematic representation of MOSFET oxide capacitances during (a) cut-off, (b) linear- mode operation, and (c) saturation

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.

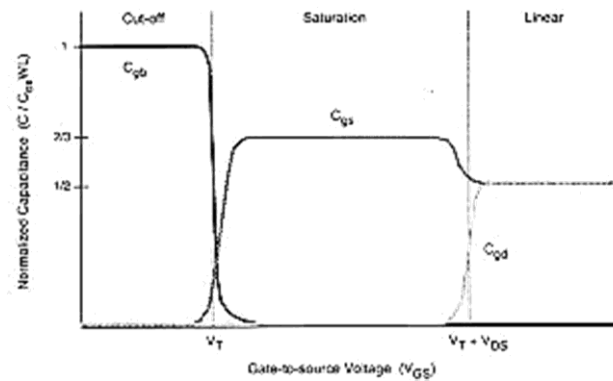
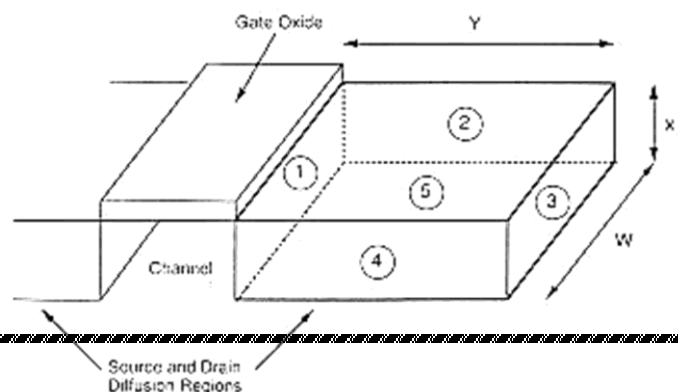


Figure : Variation of oxide capacitances as functions of gate-to-source voltage.

Now we consider the voltage-dependent source-to-substrate and drain-to-substrate capacitances, C_{sb} and C_{db} . 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



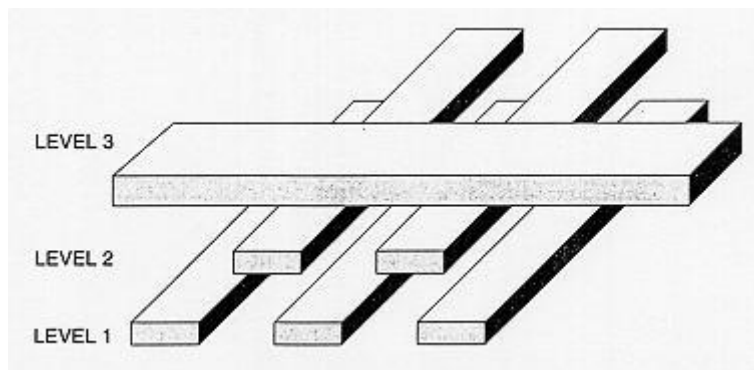
literature, therefore, a detailed discussion of capacitance calculations will not be presented here.

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

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.

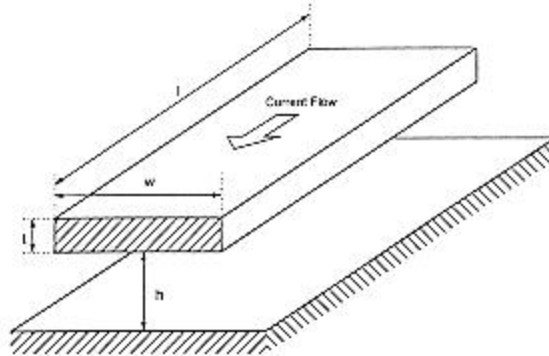
DISTRIBUTED RC EFFECTS

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 floor planning of the chip is done.

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

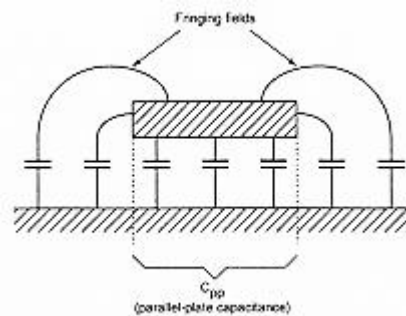


Figure : Influence of fringing electric fields upon the parasitic wire capacitance First, consider the section of a single interconnect which is shown in Fig12. It is assumed that this wire segment has a length of (l) in the current direction, a width of (w) and a thickness of (t). Moreover, we assume 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 C_{pp} 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.

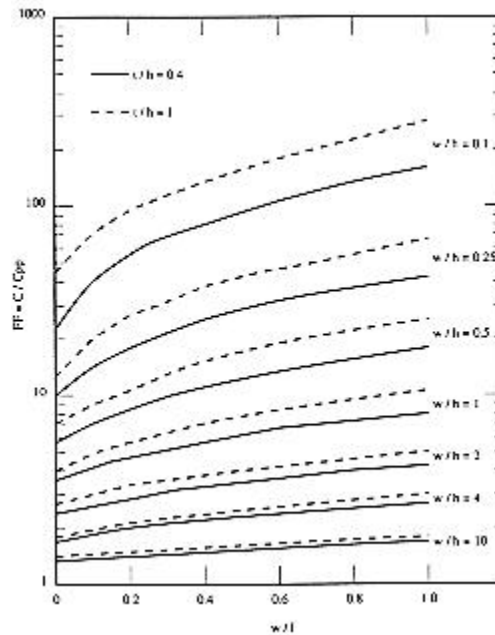


Figure: Variation of the fringing-field factor with the interconnect geometry.

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

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

line width (w).

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.

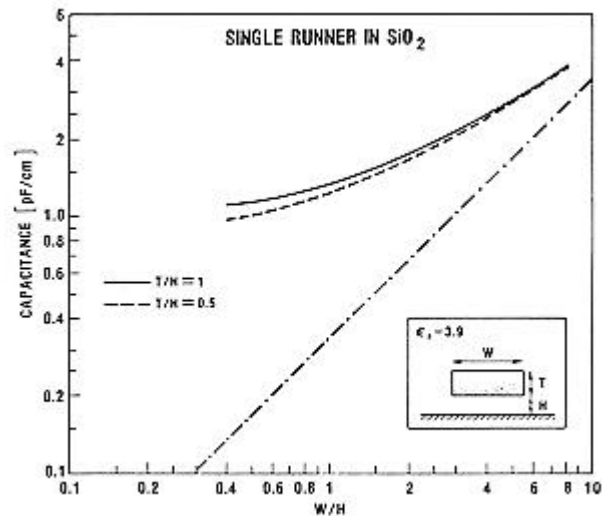
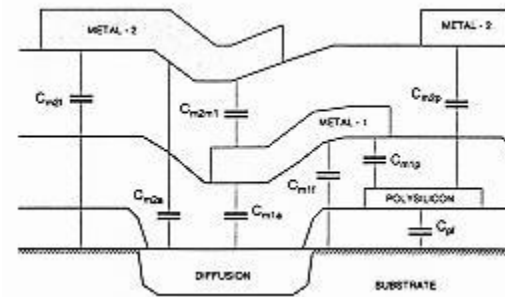


Figure-15: Capacitance of a single interconnect, as a function of (w/h) and (t/h) .

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 C_{m2m1} , C_{m1p} and C_{m2p} , 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.



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 atypical 0.8 micron CMOS technology.

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

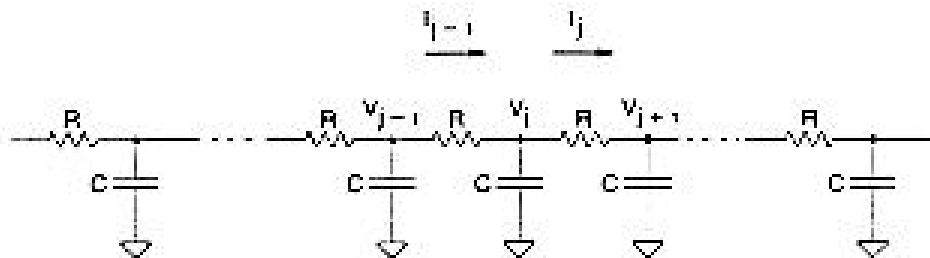
For the estimation of interconnect capacitances in a complicated three-dimensional structure, the exact geometry must be taken into account for every portion of the wire. Yet this requires an unacceptable amount of computation in a large circuit, even if simple formulas are applied for the calculation 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 ρ represents the characteristic resistivity of the interconnect material, and R_{sheet} 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.



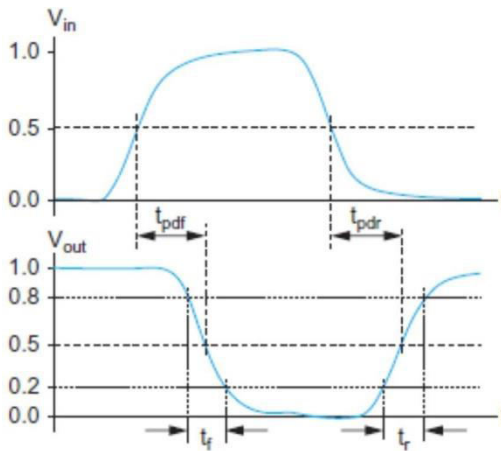
In most short-distance aluminum and silicide interconnects, the amount of parasitic wire resistance is usually negligible. On the other hand, the effects of the parasitic resistance must be taken into account for longer wire segments. As a first-order approximation in simulations, the total lumped resistance 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.

1. SWITCHING CHARACTERISTICS

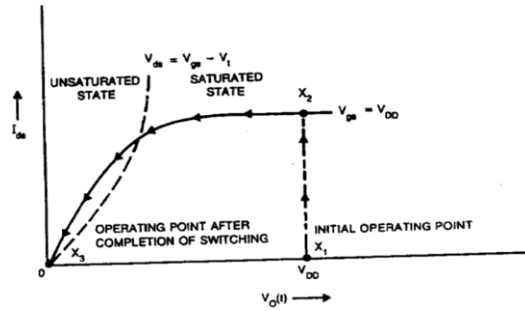
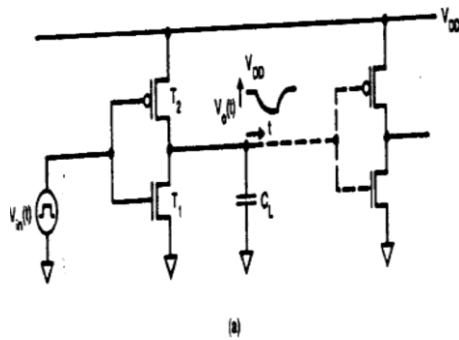
Delay and power are influenced as much by the wires as by the transistors, delves into interconnect analysis and design. A chip is of no value if it cannot reliably accomplish its function, how we achieve robustness in designs.

We begin with a few definitions illustrated in Figure,

- Propagation delay time, t_{pd} = maximum time from the input crossing 50% to the output crossing 50%
- Contamination delay time, t_{cd} = minimum time from the input crossing 50% to the output crossing 50%
- Rise time, t_r = time for a waveform to rise from 20% to 80% of its steady-state value
- Fall time, t_f = time for a waveform to fall from 80% to 20% of its steady-state value
- Edge rate, $t_{rf} = (t_r + t_f)/2$



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, t_{pdr}/t_{cdr} , and the output falling, t_{pdf}/t_{cdf} . 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.



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;

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

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

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

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).

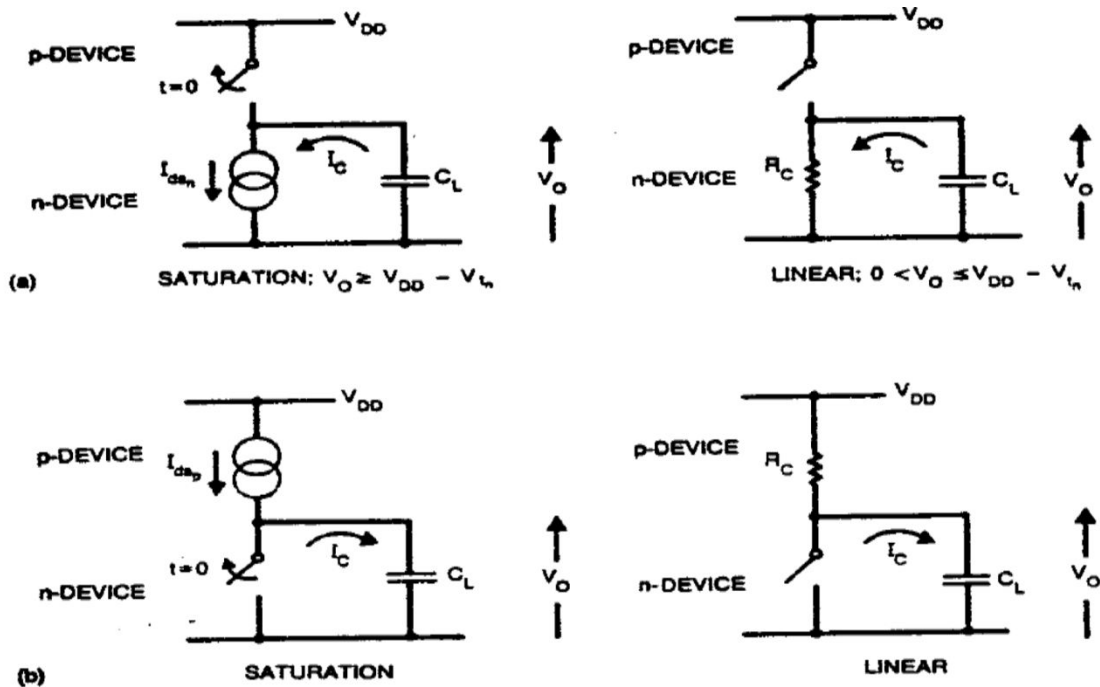
Switch-Level RC models: This is an RC modeling technique that represents transistors as a resistance discharging or charging a capacitance. The models include,

- Simple RC delay model,
- RC-tree model

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

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

where k is the loading capacitance, t_{output} is delay per loading capacitance and t_{internal} 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 $(V_{DD} - V_{tn})$.
- 2 t_{f2} = period during which the capacitor voltage, V_o , drops from $(V_{DD} - V_{tn})$ 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

$$C_L \frac{dV_o}{dt} + \frac{\beta_n}{2} (V_{DD} - V_{tn})^2 = 0; \quad V_o \geq V_{DD} - V_{tn}. \quad (4.21)$$

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

$$t_{f1} = 2 \frac{C_L}{\beta_n (V_{DD} - V_{tn})^2} \int_{V_{DD} - V_{tn}}^{0.9 V_{DD}} dV_o \quad (4.22)$$

$$= \frac{2 C_L (V_{tn} - 0.1 V_{DD})}{\beta_n (V_{DD} - V_{tn})^2}.$$

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 $(V_{DD} - V_{tn})$ to $0.1 V_{DD}$ can be obtained as before, giving

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

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

Thus the complete term for the fall time, t_f is

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

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

$$t_f \approx 4 \frac{C_L}{\beta_n V_{DD}}. \quad (4.25)$$

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

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

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

$$t_r \approx 4 \frac{C_L}{\beta_p V_{DD}}. \quad (4.27)$$

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

$$t_f = \frac{t_r}{2}. \quad (4.28)$$

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 μ_n and μ_p must be known. These, in turn, depend on process parameters.

4.4.3 Delay time τ_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

$$t_{dr} = \frac{t_r}{2} \tag{4.29}$$

$$t_{df} = \frac{t_f}{2}.$$

The average gate delay for rising and falling transitions is then

$$\tau_{av} = \frac{t_{df} + t_{dr}}{2} \tag{4.30}$$

$$= \frac{t_r + t_f}{4}.$$

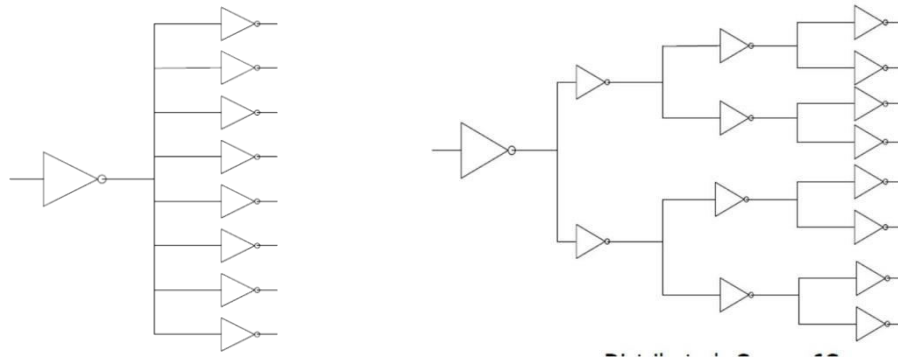
2. STAGE RATIO

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 V_t . It is therefore clear that designers must take care to control these parameters to ensure that 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. V_t 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 V_t 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 V_{SS} for NMOS and V_{DD}

for PMOS. Since the carrier mobility (n or p) is a fixed property of the semiconductor (affected by process), αx is a fixed physical parameter and $t_{\alpha x}$ 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 μ m process the gate length will be 0.1 μ 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 V_t 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 floor planning and placement is essential.

STAGE RATIO FOR LARGER LOAD

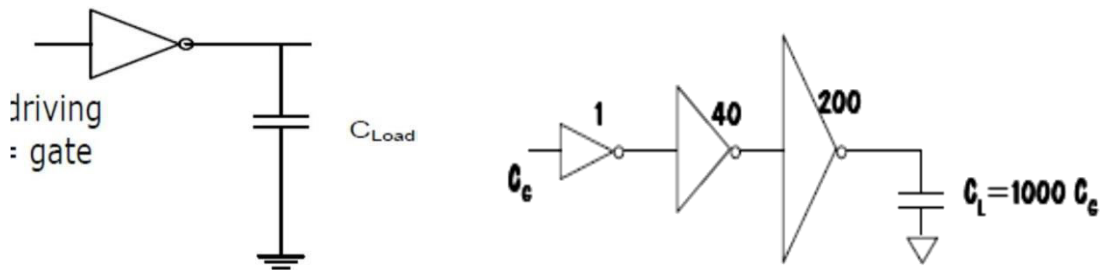
If large loads have to be driven, the delay may increase drastically. Large loads are

output capacitances, clock trees, etc.

$$t_d = t_{inv} (C_L/C_G)$$

Where t_{inv} is the average delay of a minimum-sized inverter driving another minimum sized inverter, C_L = load capacitance & C_G = gate capacitance

$$C_L = 1000C_G, \text{ then } t_d = 1000.t_{inv}$$



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

$$t_d = t_{inv} (40/1) + t_{inv} (200/40) + t_{inv} (1000/200) = 50t_{inv}$$

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

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

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

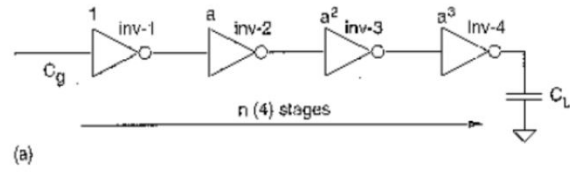
$$t_r = 2.2R_{scaled} (C_{in} + C_{load})$$

But scaling the transistor also affects the input capacitance of the transistor, $C_{in,scaled} = S \cdot C_{in}$

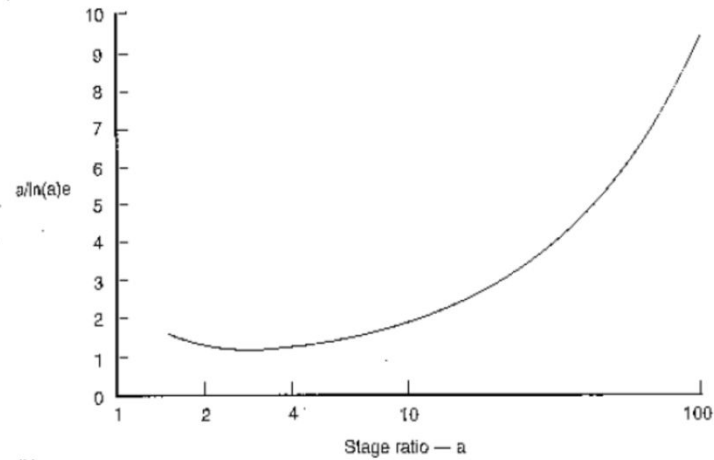
The problem is if input signal is placed at inverter 1 what's the number of stages N and the scaling factor S that will minimize the time needed for the signal to reach C_{load}

$$t_d = t_1 + t_2 + t_3 + \dots + t_N$$

$$t_d = R_1C_2 + R_2C_3 + R_3C_4 + \dots + R_{N-1}C_N + R_N C_{load}$$



(a)



(b)

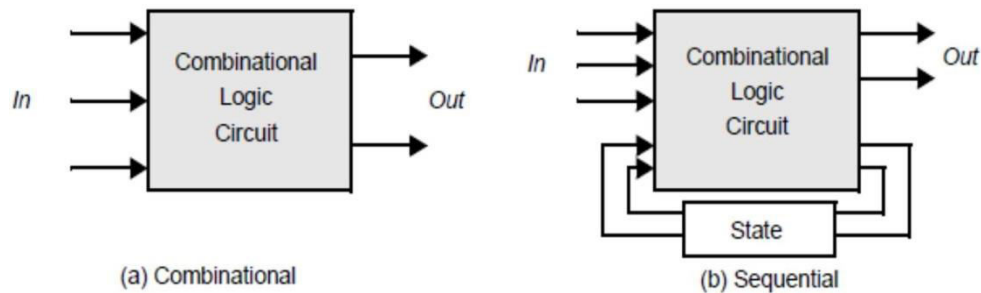
The delay through each stage is atd , where t_d is the average delay of a minimum-sized inverter driving another minimum-sized inverter. Hence the delay through n stages is $natd$. If the ratio of the load capacitance to the capacitance of a minimum-sized inverter is $R = C_L / C_g$, then $R = an$. Hence $\ln(R) = n \ln(a)$. 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,

$$S = e = 2.71$$

3. 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.

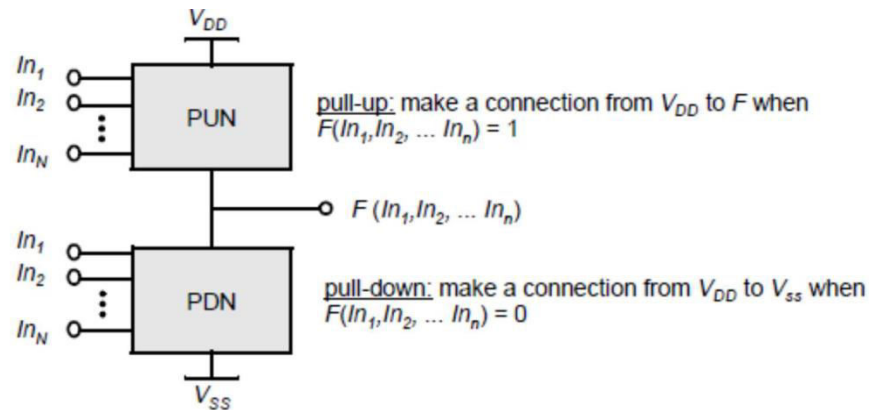


There are numerous circuit styles to implement a given logic function. As with the inverter, the common design metrics by which a gate is evaluated include area, speed, energy and power. Depending on the application, the emphasis will be on different metrics (e.g., in high performance processor, the 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, which relies on temporary storage of signal values on the capacitance of high-impedance circuit nodes. The latter approach has the advantage that the resulting gate is simpler and faster. On the other hand, its design and operation are more involved than those of its static counterpart, due to



an increased sensitivity to noise.

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 V_{DD} anytime the output of the logic gate is meant to be 1 (based on the inputs). Similarly, the function of the PDN is to connect the output to V_{SS} when the output of the logic gate is meant to be 0. The PUN and PDN networks are constructed in a mutually exclusive fashion such that one and only one of the networks is conducting in steady state. In this way, once the transients have settled, a path always exists between V_{DD} and the output F , realizing a high output (“one”), or, alternatively, between V_{SS} and F for a low output (“zero”). This is equivalent to stating that the output node is always a low-impedance node in steady state. 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 V_{DD} . 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 $|V_{Tp}|$ — 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 $VDD - V_{Tn}$. 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 transferred to the other end. Similarly, NMOS transistors connected in parallel represent an OR function. A conducting path exists between the output and input terminal if at least one of the inputs is high. Using similar arguments, construction rules for PMOS networks can be formulated. A series connection of PMOS conducts if both inputs are low, representing a NOR function ($A \cdot B = A + B$), while PMOS transistors in parallel implement a NAND ($A + B = A \cdot B$).

4) Using De Morgan's theorems ($(A + B) = A \cdot B$ and $A \cdot B = A + B$), 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 ($F = A \cdot B$). The PDN network consists of two NMOS devices in series that conduct when both A and B are high. The PUN is the dual network, and consists of two parallel PMOS transistors. This means that F is 1 if $A = 0$ or $B = 0$, which is equivalent to $F = A \cdot B$. The truth table for the simple two input NAND gate is given in Table

It can be verified that the output F is always connected to either VDD or GND, but never to both at the same time.

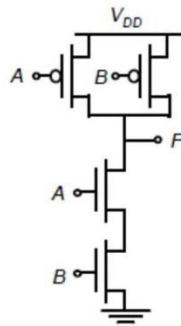


Table 6.1 Truth Table for 2 input NAND

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

Design Consideration:

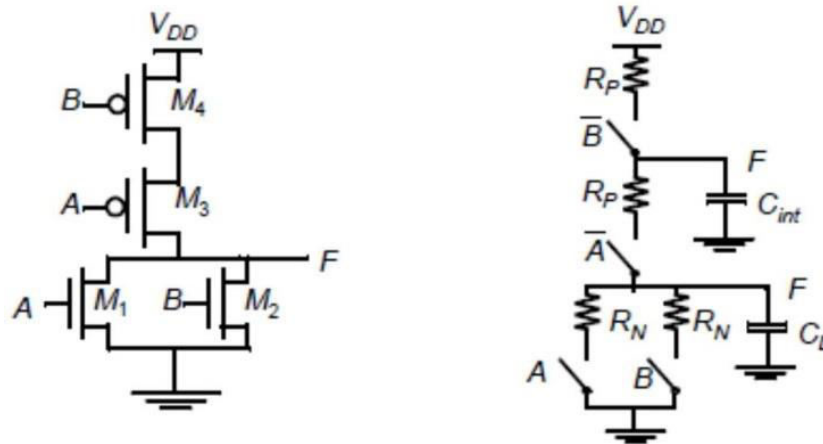
The important point to take away from the above discussion is that the noise margins are input pattern dependent. For the above example, a smaller input glitch will cause a transition at the output if only one of the inputs makes a transition. Therefore, this condition has a lower low noise margin. A 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

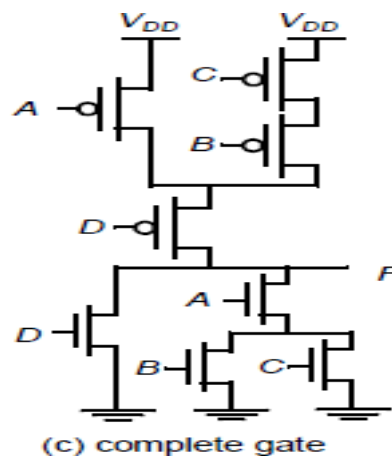
an inverter with the following device sizes: NMOS 0.5mm/0.25mm and PMOS 1.5mm/0.25mm.



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



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 combination, 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.

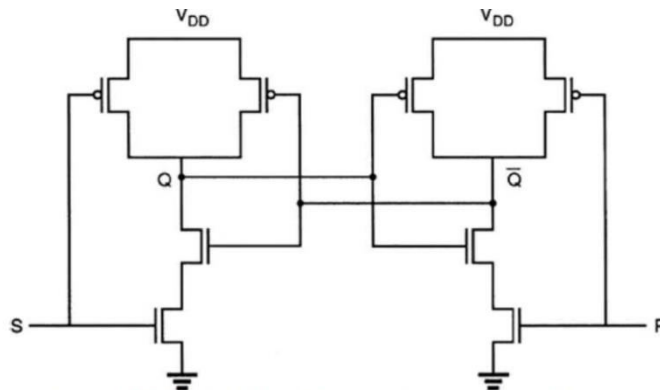
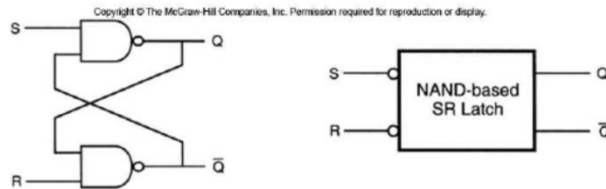


Figure 5.7 CMOS SR latch circuit based on NAND2 gates



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.

POST MCQ:

1. The ratio of rise time to fall time can be equated to _____
a) β_n/β_p
b) β_p/β_n
c) $\beta_p*\beta_n$
d) $\beta_p/2\beta_n$
2. Rise time and fall time is _____ to load capacitance CL.
a) directly proportional
b) inversely proportional
c) exponentially equal
d) not related
3. The value μ_n is equal to _____
a) μ_p
b) $0.5\mu_p$
c) $1.5\mu_p$
d) $2.5\mu_p$
4. What is the total delay of an nMOS pair?
a) $f\tau$
b) $2f\tau$
c) $5f\tau$
d) $4f\tau$
5. Register cell consists of
a) inverter
b) pass transistor
c) inverter & pass transistor
d) none of the mentioned

UNIT IV
ASYNCHRONOUS SEQUENTIAL LOGIC

PRE MCQ:

1. Approach used for design process are
 - a) circuit symbols
 - b) logic symbols
 - c) stick diagrams
 - d) all of the mentioned**
2. Two metal layers can be joined by using
 - a) contact cut
 - b) wire
 - c) via**
 - d) glass
3. The bottom subfunction is called as
 - a) lower function
 - b) low cell
 - c) leaf cell**
 - d) bottom cell
4. Regularity is the ratio of
 - a) total transistors in the chip to total transistors that must be designed in detail**
 - b) total transistors that must be designed in detail to total transistors in a chip
 - c) total transistors to total components
 - d) total charge storage components to charge dissipating components
5. Design gives a detailed
 - a) logic circuit design
 - b) topology of communication**
 - c) colour codes of the layers
 - d) functions of layers

THEORY:

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.

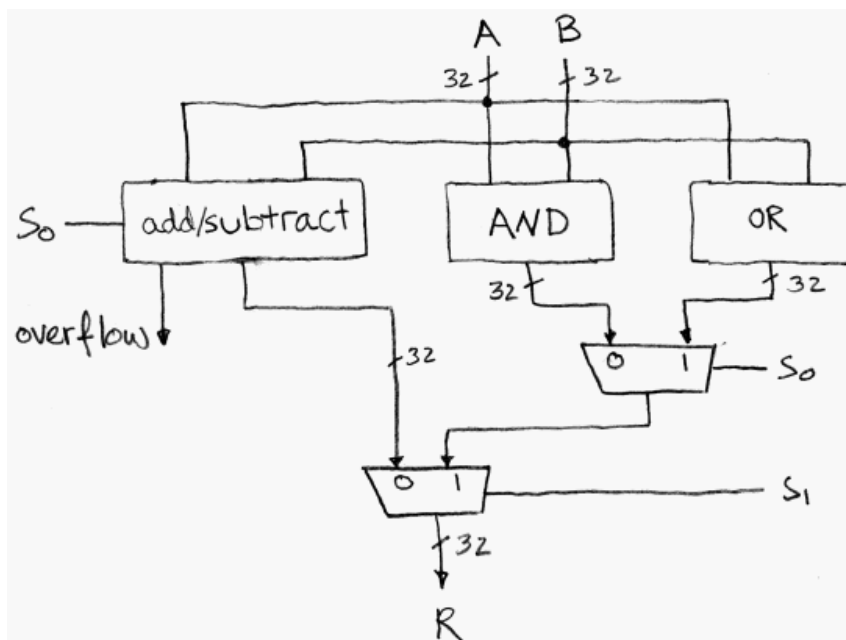
1.
 - a) Top-down design approach with adequate CAD tools to do the job
 - b) Partitioning the system sensibly
 - c) Aiming for simple interconnections
 - d) High regularity within subsystem
 - e) Generate and then verify each section of the design
2. Devote significant portion of total chip area to 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 accommodate 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

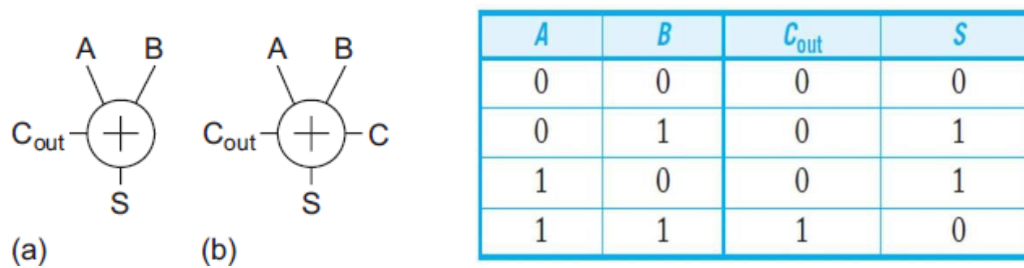
When estimating or measuring ALU performance, one wonders if a 32-bit ALU is as fast as a 1-bit ALU - what is the degree of parallelism, and do all operations execute in parallel? In practice, some operations on N-bit operands (e.g., addition with sequential propagation of carries) take $O(N)$ time. 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 $O(\log_2 N)$ stages in a tree-structured configuration with $N-1$ one-bit adders. Thus, the complexity of this technique is $O(\log_2 N)$ work. In a sequential model of computation, this translates to $O(\log_2 N)$ time. If one is adding smaller numbers (e.g., up to 10-bit integers with current memory technology), then a lookup 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 C_{out} . 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 C_{in} .



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 C_{out} is true independent of C_{in} , so $G = A \cdot B$. The adder kills a carry when C_{out} is false independent of C_{in} , so $K = (A \cdot B)' = (A + B)'$. 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: $P = A \text{ (exor) } B$.

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,

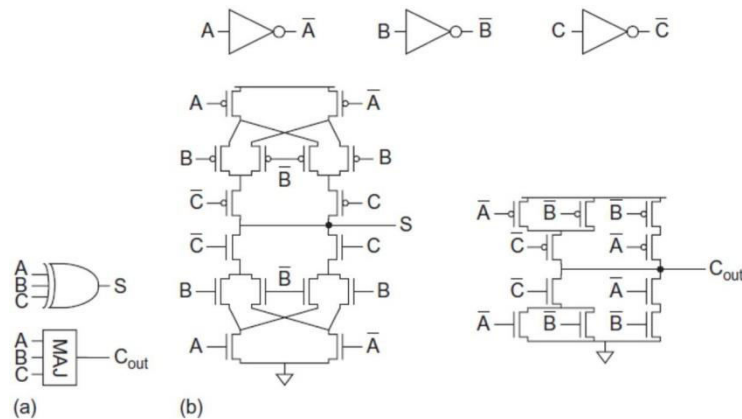
$$S = A \oplus B$$

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

and the full adder logic is,

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

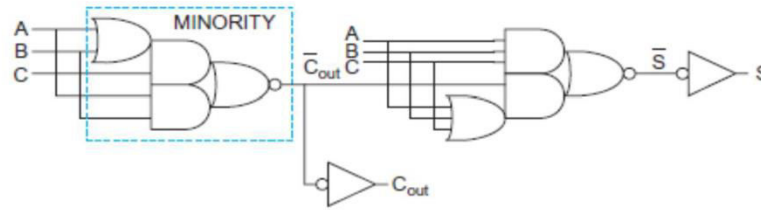
The most straightforward approach to designing an adder is with logic gates. Figure shows a half adder. Figure shows a full adder at the gate (a) and transistor (b) levels. The carry gate is also called a majority gate because it produces a 1 if at least two of the three inputs are 1.



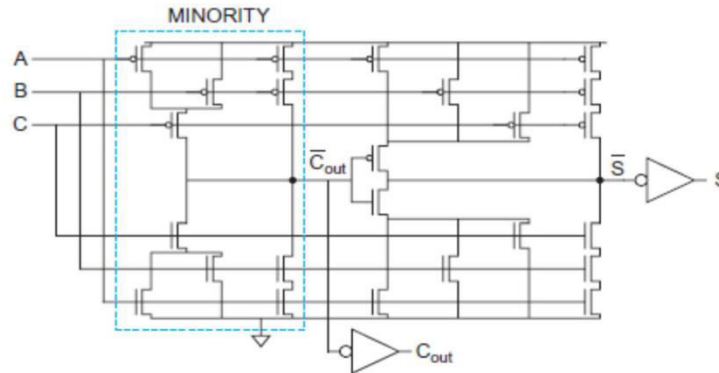
The full adder of Figure employs 32 transistors (6 for the inverters, 10 for the majority gate, and 16 for the 3-input XOR). A more compact design is based on the observation that S can be factored to reuse the Cout term as follows:

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

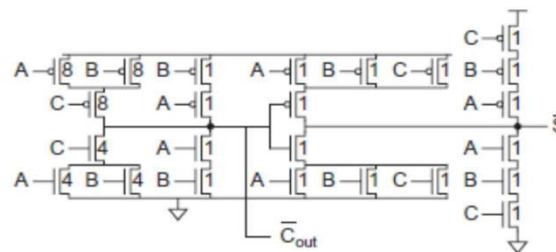
Such a design is shown at the gate (a) and transistor (b) levels in Figure 11.4 and uses only 28 transistors. Note that the pMOS network is identical to the nMOS network rather than being the conduction complement, so the topology is called a mirror adder. This simplification reduces the number of series 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 C_{out} . In carry-ripple adders (Section 11.2.2.1), the critical path goes from C to C_{out} through many full adders, so the sizes optimized to favor the critical path using a number of techniques:

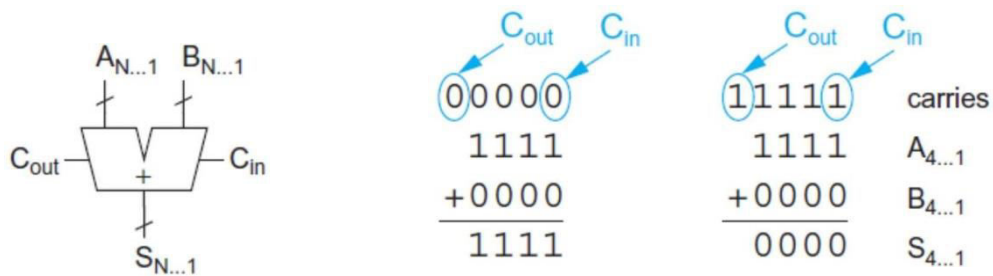
Feed the carry-in signal (C) to the inner inputs so the internal capacitance is already discharged. Make all transistors in the sum logic whose gate signals are connected to the carryin and carry logic minimum size (1 unit). This minimizes the branching effort on the critical path. Keep routing on this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

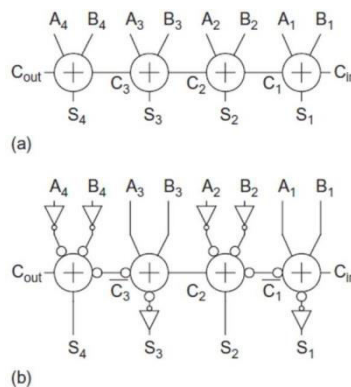
Carry-Propagate Addition

N-bit adders take inputs $\{A_N, \dots, A_1\}$, $\{B_N, \dots, B_1\}$, and carry-in C_{in} , and compute the sum $\{S_N, \dots, S_1\}$ and the carry-out of the most significant bit C_{out} , as shown in Figure. (Ordinarily, this text calls the least significant bit A_0 rather than A_1 . 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 C_{in} . 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 , C_i , is the carry-in to bit $i + 1$. This carry is said to have twice the weight of the sum S_i . The delay of the adder is set by the time for the carries to ripple through the N stages, so the $t_{C \rightarrow C_{out}}$ 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 $i \leq k < j$ 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 $C_0 = C_{in}$ and $C_N = C_{out}$. Then we can define generate and propagate signals for bit 0 as,

$$G_{0:0} = C_{in}$$

$$P_{0:0} = 0$$

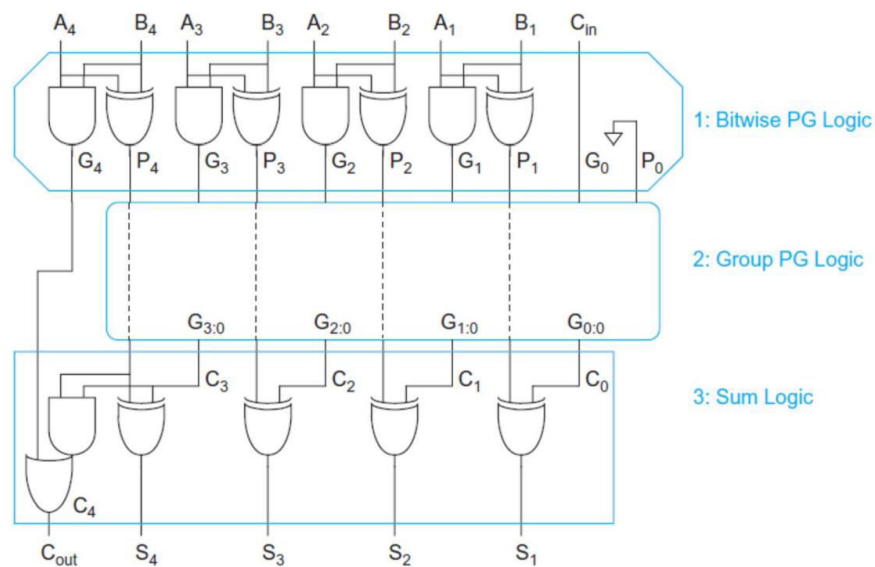
Observe that the carry into bit i is the carry-out of bit $i-1$ and is $C_{i-1} = G_{i-1:0}$. This is an important relationship; group generate signals and carries will be used synonymously in the subsequent sections. We can thus compute the sum for bit i using EQ (11.2) as,

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

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 $G_{i-1:0}$ 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.

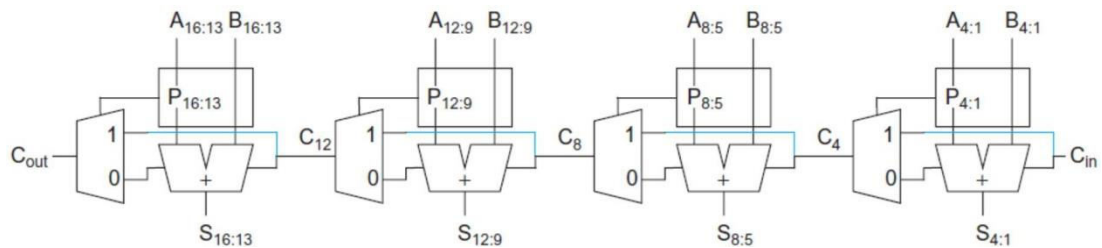


Carry-Skip Adder

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.

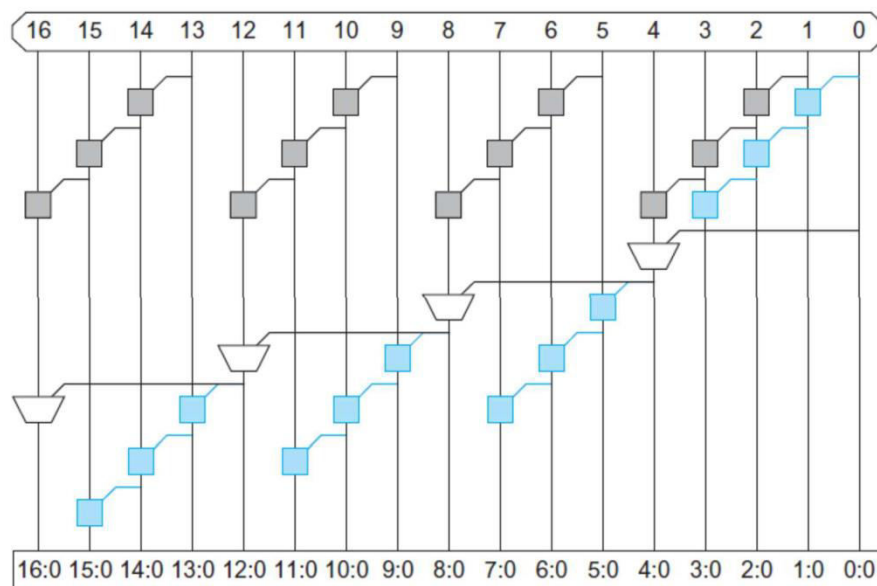
The critical path through Figure begins with generating a carry from bit 1, and then propagating it through the remainder of the adder. The carry must ripple through the next three

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-

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



skip adder using k n-bit groups ($N = n \times k$) has a delay of

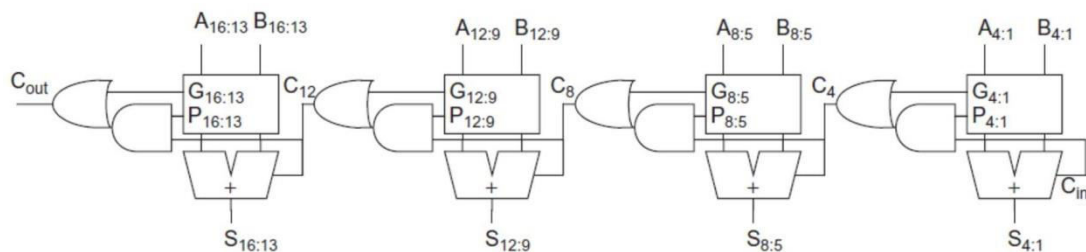
Carry-Lookahead Adder

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

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

using k groups of n bits each has a delay of

where $t_{pg(n)}$ 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.

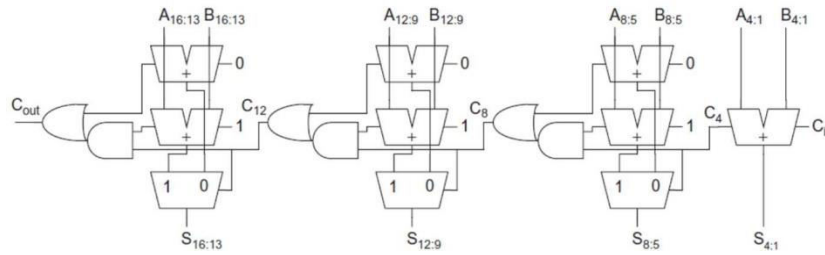


CLAs often use higher-valency cells to reduce the delay of the n -bit additions by computing the carries in parallel. Figure shows such a CLA in which the 4-bit adders are built using Manchester carry chains or multiple static gates operating in parallel.

Carry-Select Adders

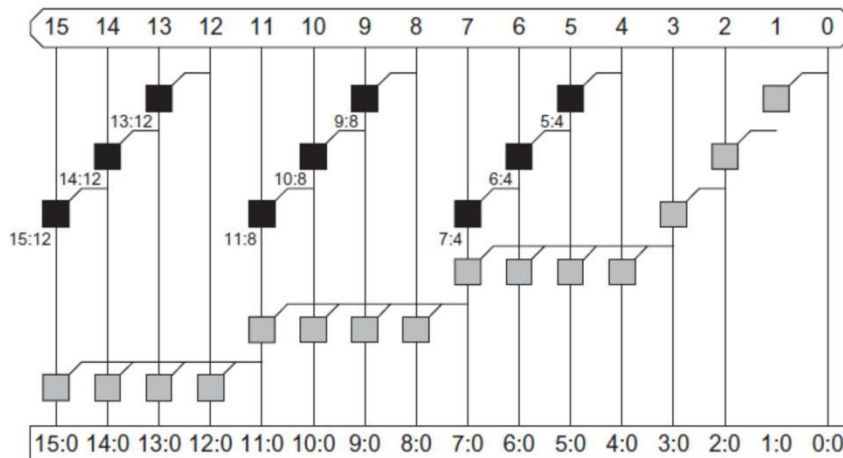
The critical path of the carry-skip and carry-lookahead adders involves calculating the carry into each n -bit group, and then calculating the sums for each bit within the group based on the carry-in. A standard logic design technique to accelerate the critical path is to precompute the outputs for both 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

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



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

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



comparable, but the area is smaller.

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.

	011001	:	25 ₁₀		}	multiplicand multiplier partial products product
×	100111	:	39 ₁₀			
	011001					
	011001					
	000000					
	000000					
	+011001					
	001111001111		:	975 ₁₀		

$M \times N$ -bit multiplication $P = Y \times X$ 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,

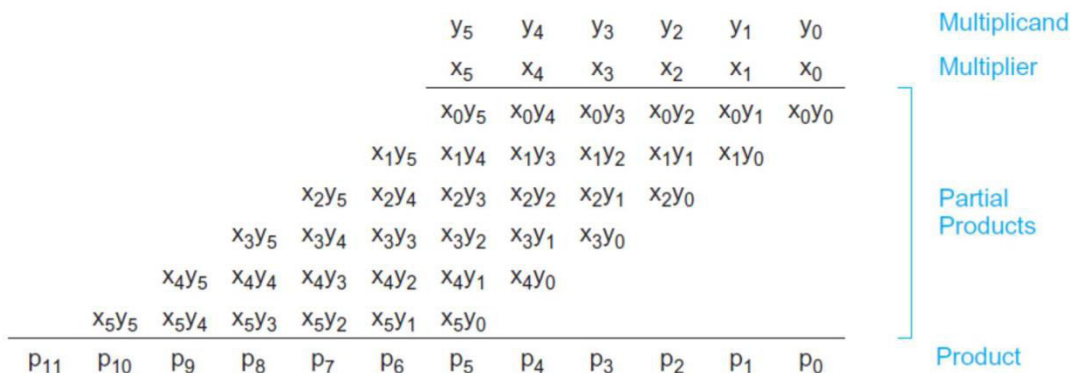
$$Y = \{y_{M-1}, y_{M-2}, \dots, y_1, y_0\}$$

and the multiplier as,

$$X = \{x_{N-1}, x_{N-2}, \dots, x_1, x_0\}.$$

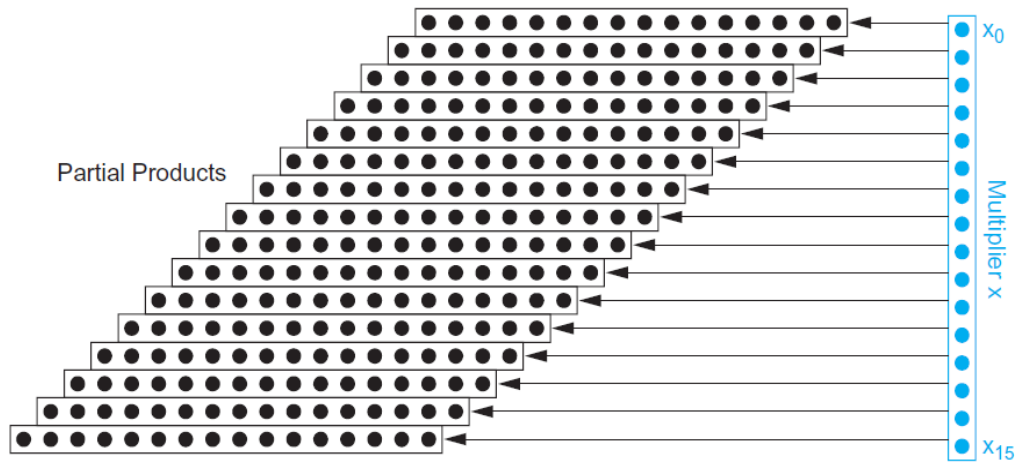
For unsigned multiplication, the product is given in EQ (11.31). Figure 11.72 illustrates the

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



generation, shifting, and summing of partial products in a 6×6 -bit multiplier.

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.

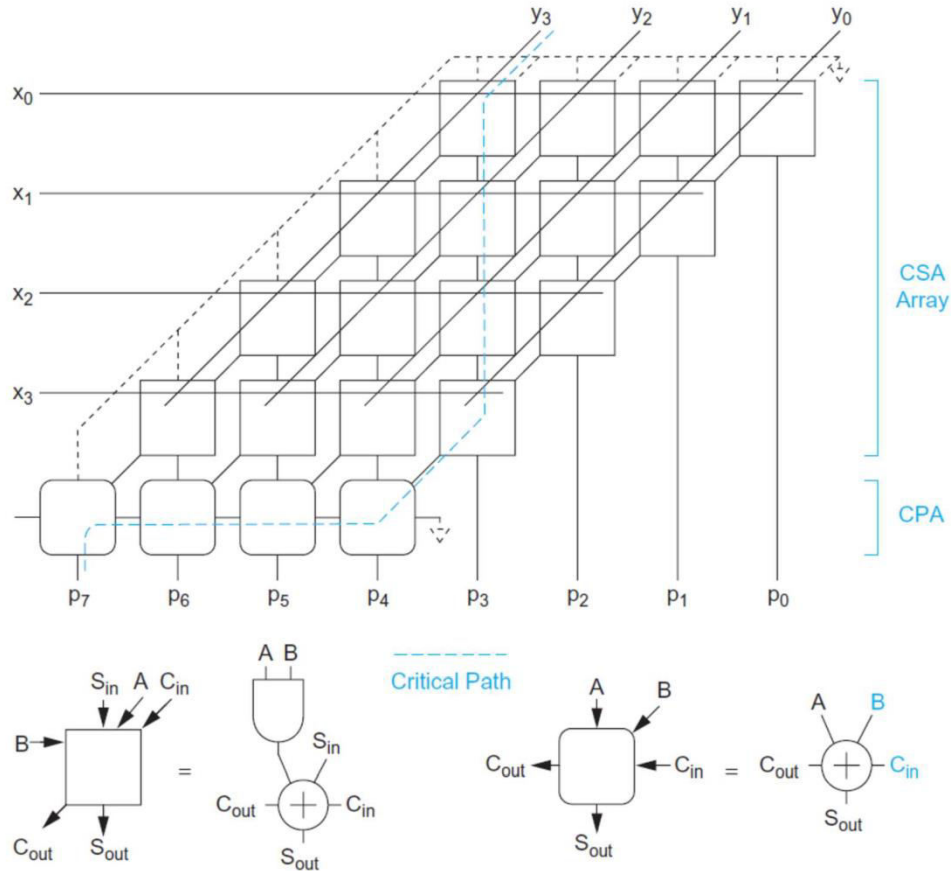
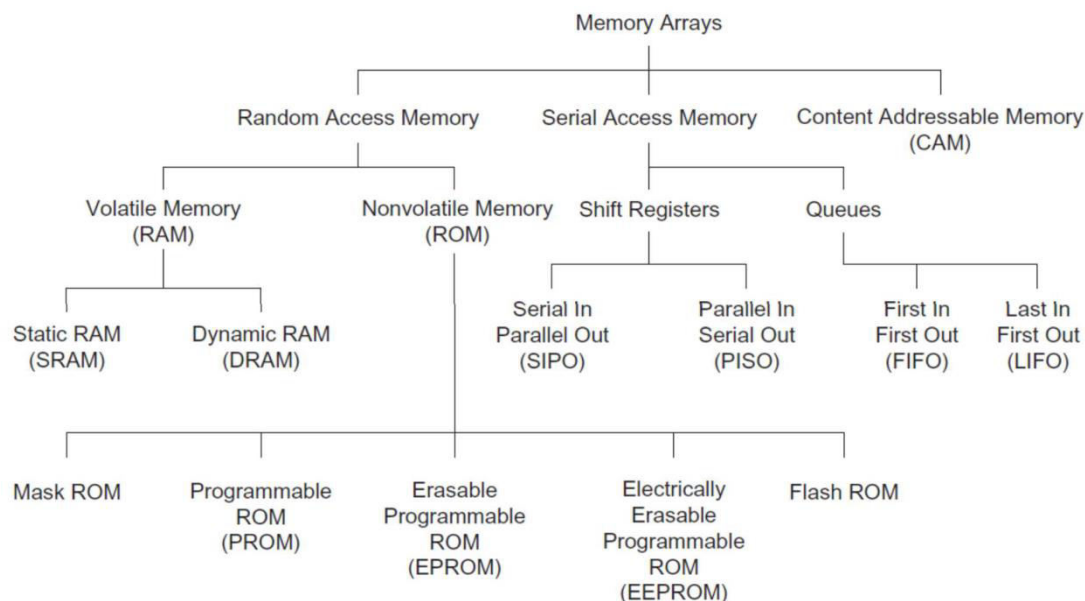


Figure shows a 4×4 array multiplier for unsigned numbers using an array of CSAs. Each cell contains a 2-input AND gate that forms a partial product and a full adder (CSA) to add the partial product into the running sum. The first row converts the first partial product into carry-save redundant form. 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 layout. 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

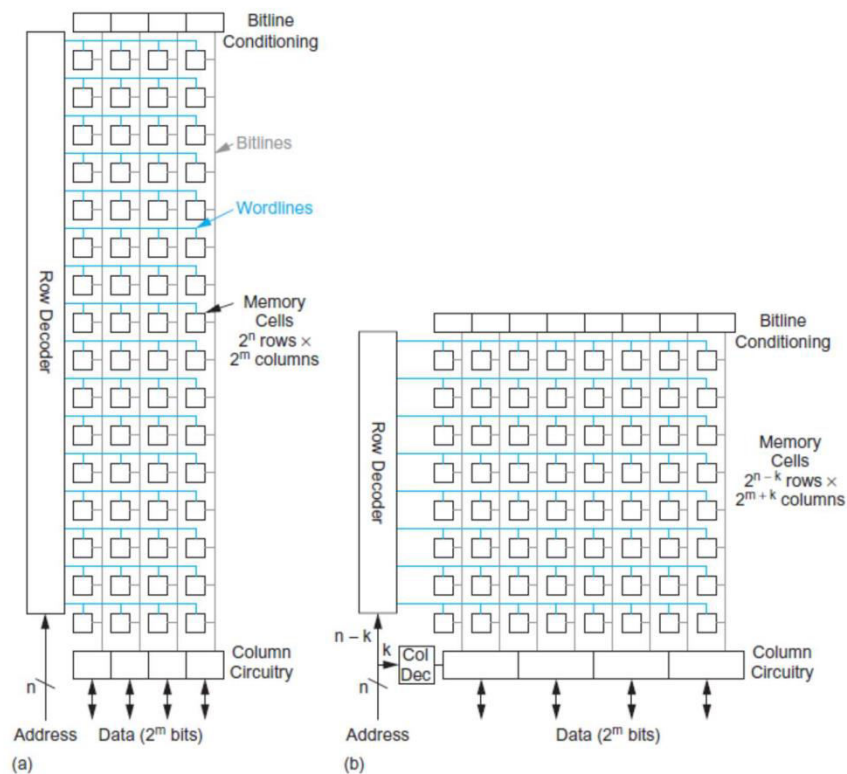
Memory arrays often account for the majority of transistors in a CMOS system-on-chip. Arrays may be divided into categories as shown in Figure. Programmable Logic Arrays (PLAs) perform logic rather than storage functions. Random access memory is accessed with an address and has a latency independent 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 2^n words of 2^m 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 2^{n-k} words, so the array is physically organized as 2^{n-k} rows of 2^{m+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 2^m 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

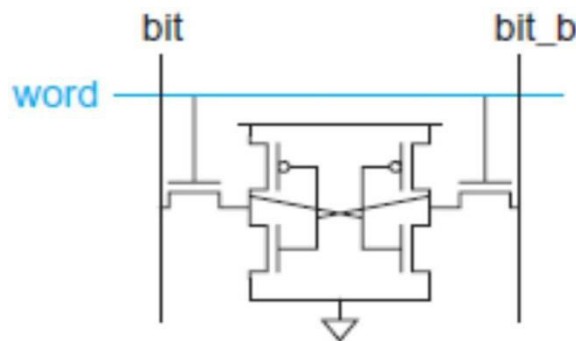
Easier to use than DRAM

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.

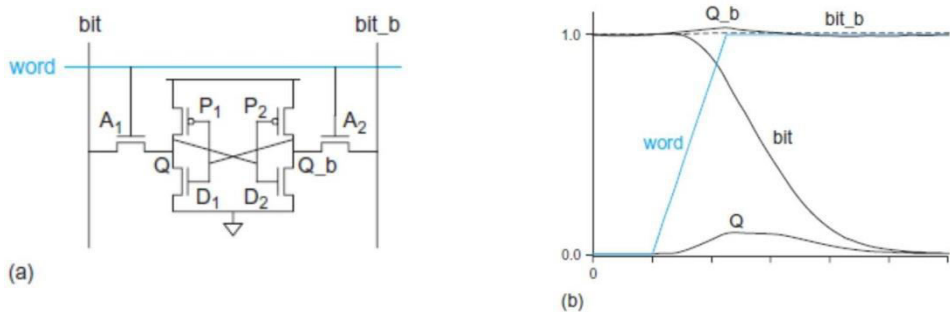
The 6T SRAM cell contains a pair of weak cross-coupled inverters holding the state and a pair of access transistors to read or write the state. The positive feedback corrects disturbances caused by leakage or noise. The cell is written by driving the desired value and its complement onto the bitlines, 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 ϕ_1 and ϕ_2 , 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

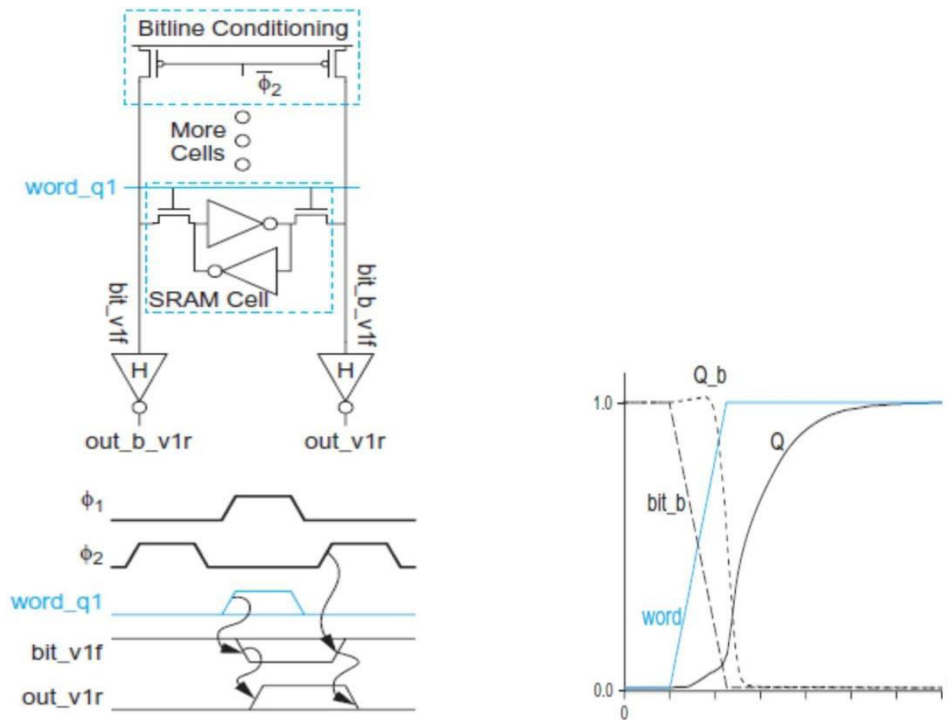


momentarily rises, but does not glitch badly enough to flip the cell.

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.

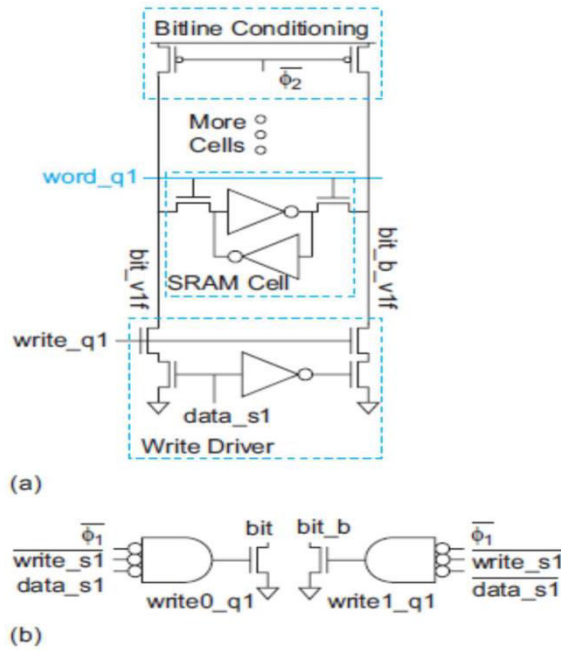
Figure (a) again shows the cell in the context of a full column from the SRAM. During phase 2, the bitlines are precharged high. Write drivers pull the bitline or its complement low during phase 1 to write the cell. The write drivers can consist of a pair of transistors on each bitline for the data and 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

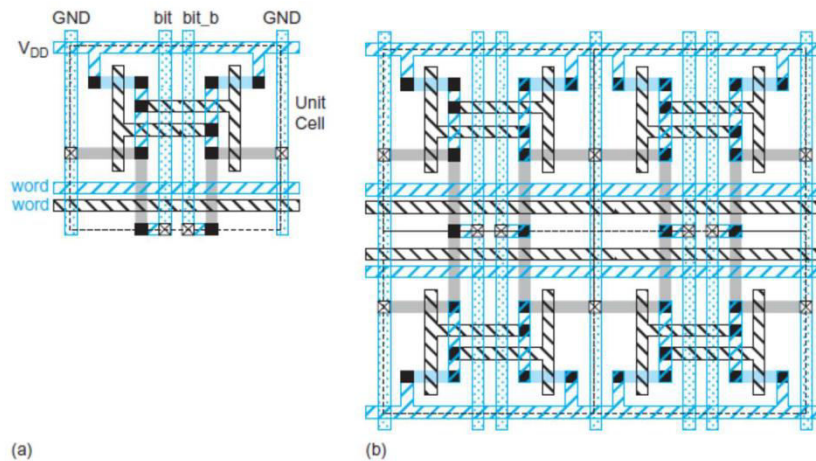
could be $8/2$, the access transistors $4/2$, and the pullups $3/3$. The SRAM cells must operate correctly at all voltages and temperatures despite process variation.

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 metal and polysilicon; the two layers must occasionally be strapped (e.g., every four or eight cells).

3. CONTROL LOGIC IMPLEMENTATION USING PLA'S

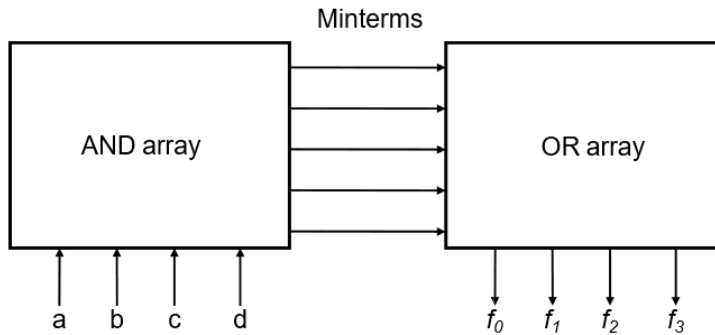
Understand how digital systems may be divided into a data path and control logic. Appreciate the different ways of implementing control logic. FSM design procedure. Draw the state-transition diagram. Check the state diagram. Write state equations. Control logic is a key part of a software program that 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 PLA represents an expression of sum-of-product (SOP)

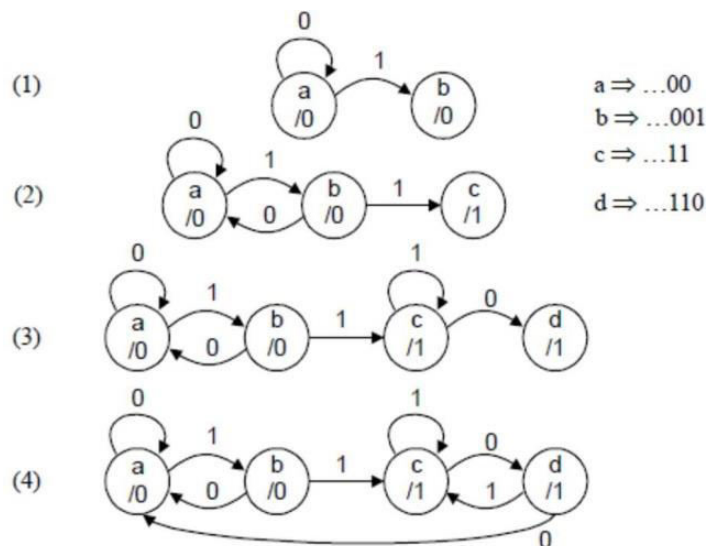
$$f_i = \sum_i m_i(a, b, c, d)$$

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

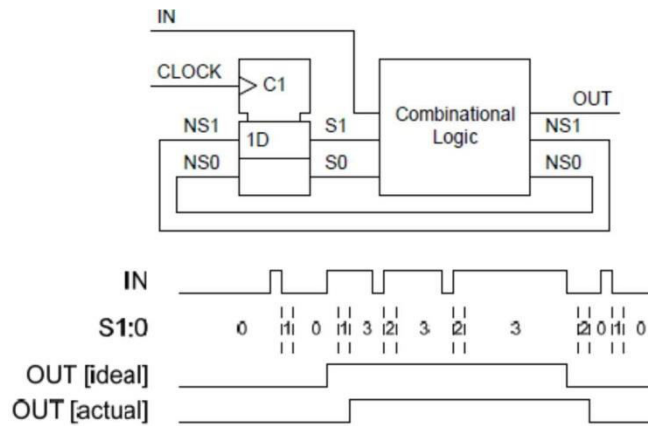
General Design Procedure

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 can forget 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

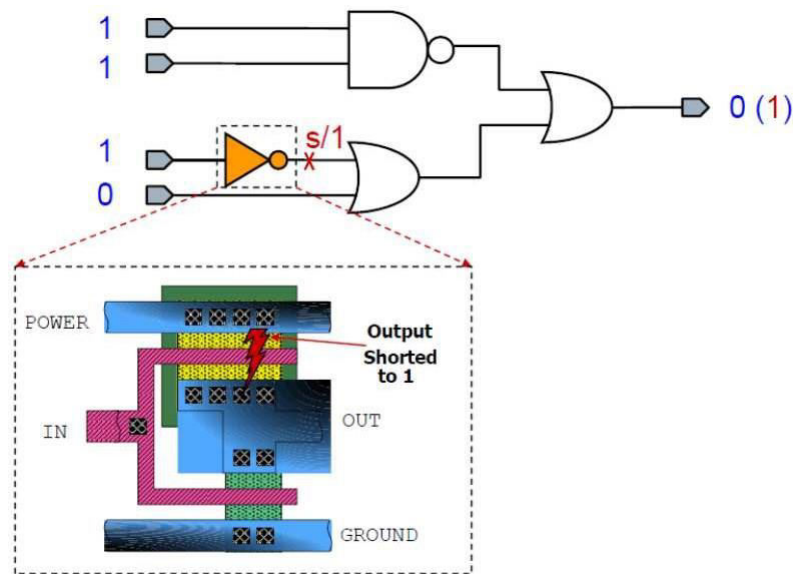
Physical defects are likely in manufacturing

- 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

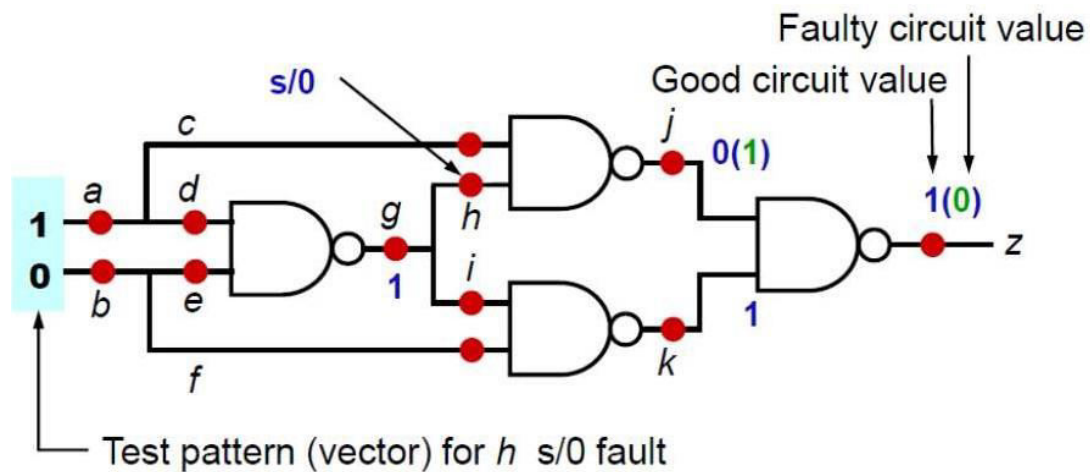
Fault Models

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-1, S-A-1). 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 include stuck-open or shorted 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

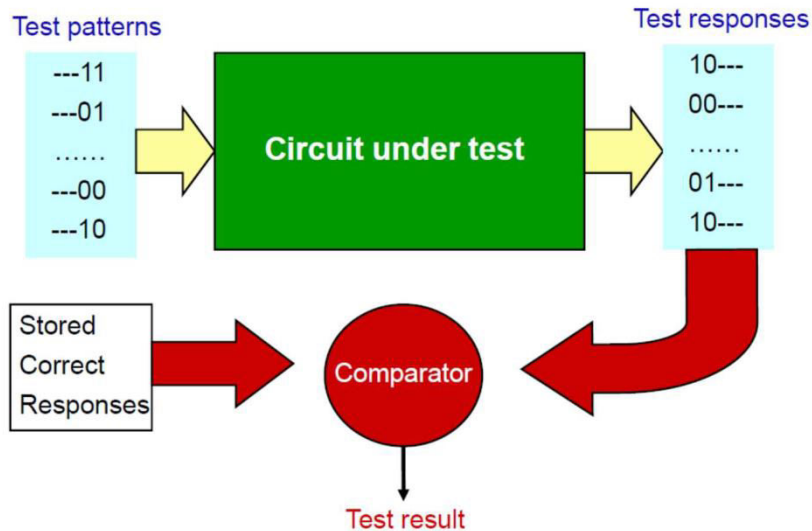
The survivability of a system is the ability to continue function after a fault. For example, error-correcting codes provide survivability in the event of soft errors. Redundant rows and columns in memories and spare cores provide survivability in the event of manufacturing defects. Adaptive techniques 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.

How to Test Chips?



Iddq testing is a method for testing CMOS integrated circuits for the presence of manufacturing faults. It relies on measuring the supply current (I_{dd}) 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 I_{ddq} for I_{dd} (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)

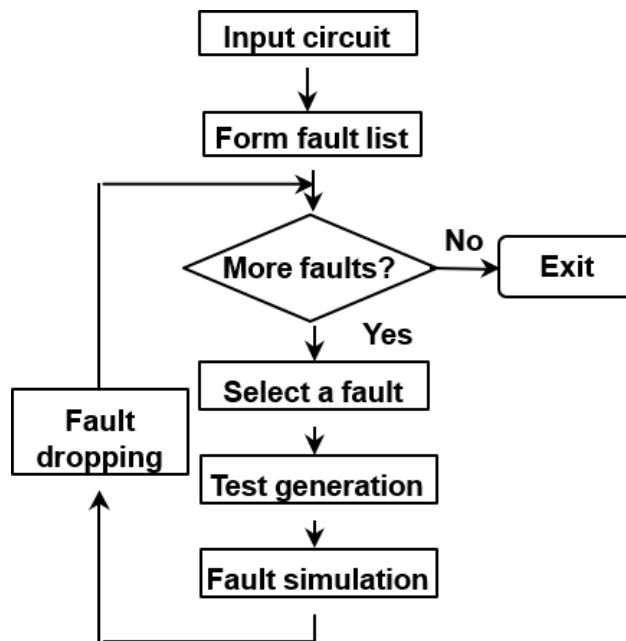
Historically, in the IC industry, logic and circuit designers implemented the functions at the RTL or schematic level, mask designers completed the layout, and test engineers wrote the tests. In many ways, the test engineers were the Sherlock Holmes of the industry, reverse engineering circuits and devising 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 2^n (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

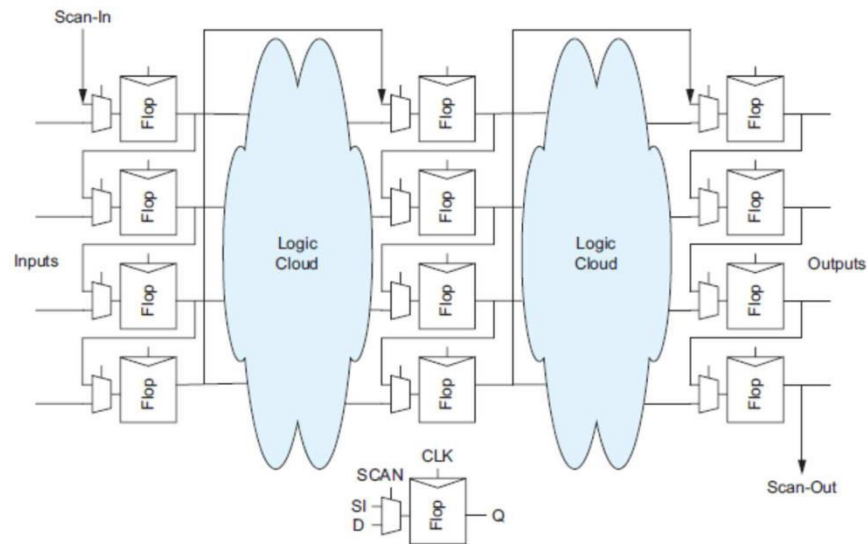
A technique classified in this category is the use of the bus in a bus-oriented system for test purposes. Each register has been made loadable from the bus and capable of being driven onto the bus. Here, the internal logic values that exist on a data bus are enabled onto the bus for testing purposes. 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.



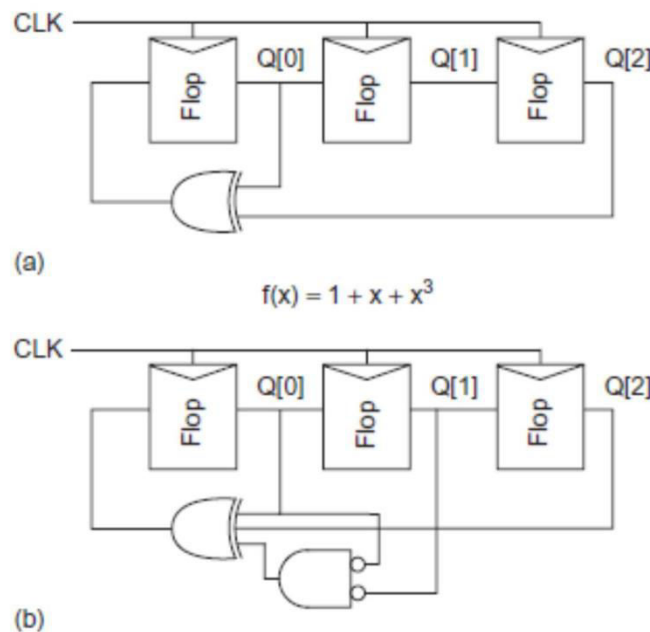
Modern scan is based on the use of scan registers, as shown in Figure. The scan register is a D flip-flop preceded by a multiplexer. When the SCAN signal is deasserted, the register behaves as a conventional register, storing data on the D input. When SCAN is asserted, the data is loaded from the SI 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.

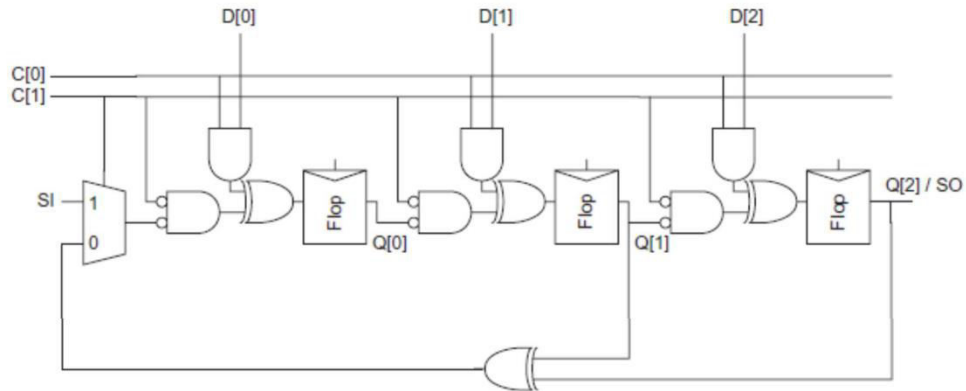
SELF-TEST APPROACHES: BUILT-IN SELF-TEST (BIST)

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.

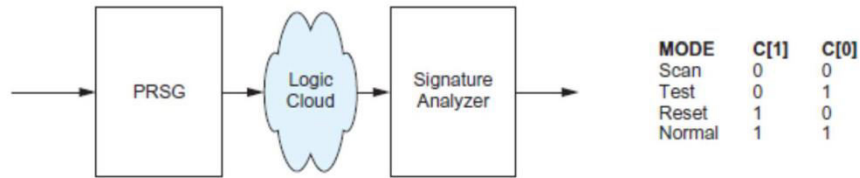


One method of testing a module is to use signature analysis or cyclic redundancy checking. This involves using a pseudo-random sequence generator (PRSG) to produce the input signals for a section of combinational circuitry and a signature analyzer to observe the output signals. A PRSG of length n is 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 $2^n - 1$ states before repeating the sequence. LFSRs are discussed further in Section

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 $0\dots01$, the next state is $0\dots00$. When in state $0\dots00$, the next state is $10\dots0$. 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



(a)

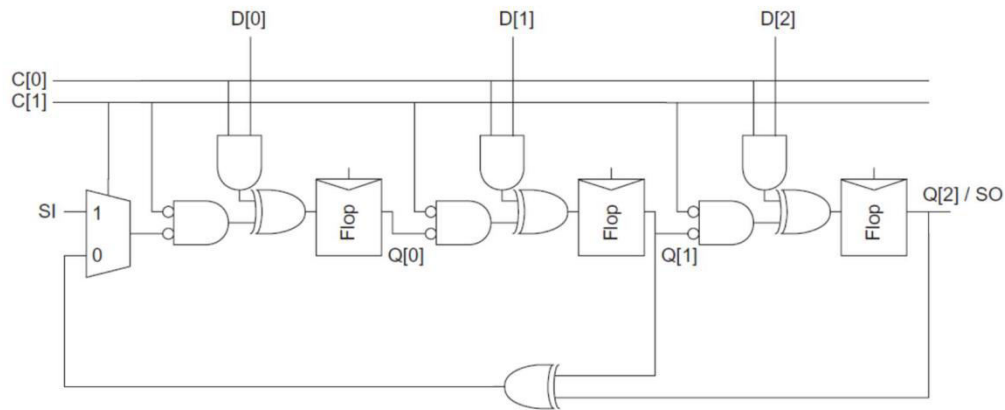


(b)

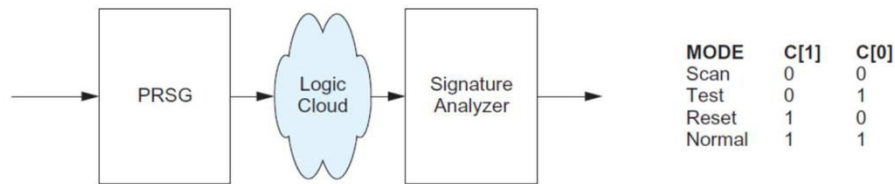
gate.

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.



(a)



(b)

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.

POST MCQ:

1. Design for testability is considered in production for chips because:
 - a) Manufactured chips are faulty and are required to be tested
 - b) The design of chips are required to be tested
 - c) Many chips are required to be tested within short interval of time which yields timely delivery for the customers**
 - d) All of the mentioned
2. For carry skip adder, the minimum total propagation delay can be obtained when m is
 - a) $\sqrt{nk1/k2}$
 - b) $\sqrt{2nk1/k2}$**
 - c) $\sqrt{2k1/nk2}$
 - d) $\sqrt{nk1k2/2}$
3. Latches chosen are
 - a) static shift registers
 - b) any flipflop
 - c) dynamic shift register**
 - d) multiplexers
4. The impedance of pull down transistor in nMOS can be given as
 - a) $2R_s$
 - b) $4R_s$
 - c) $1/2 R_s$**
 - d) $1/4 R_s$
5. Divide and Conquer approach to large and complex circuits for testing is found in:
 - a) Partition and Mux Technique**
 - b) Simplified automatic test pattern generation technique
 - c) Scan based technique
 - d) All of the mentioned

UNIT V

LOGIC FAMILIES AND PROGRAMMABLE LOGIC DEVICES

POST MCQ:

1. Which clock is preferred in storage devices?
 - a) single phase overlapping clock signal
 - b) single phase non overlapping clock signal
 - c) two phase overlapping clock signal
 - d) two phase non overlapping clock signal**
2. Reading a cell is a _____ operation.
 - a) constructive
 - b) destructive**
 - c) semi constructive
 - d) semi destructive
3. Which method is used to determine structural defects?
 - a) deterministic test pattern**
 - b) algorithmic test pattern
 - c) random test pattern
 - d) exhaustive test pattern
4. Which method is used for external functional testing?
 - a) exhaustive test pattern method
 - b) pseudo-exhaustive test pattern method
 - c) random test pattern method**
 - d) pseudo-random test pattern method
5. The electrical behaviour of a circuit is given using
 - a) design rules
 - b) floor plan
 - c) structures and layouts
 - d) mathematical modelling**

THEORY

TYPES OF PROGRAMMABLE LOGIC DEVICES (PLD):

Programmable Arrays

- OR Array
- AND Array

Classifications of Simple Programmable Logic Devices (SPLD)

- Read-Only Memory (ROM)
- Programmable Array Logic (PAL)

- Programmable Logic Array (PLA)
- Programmable Logic Sequencer (PLS)

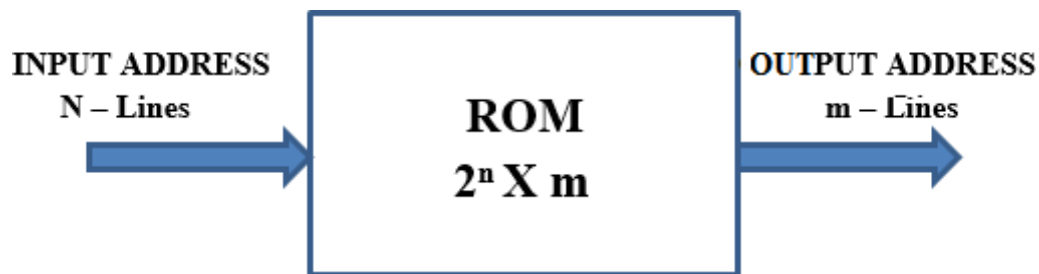
More complex

- FPGA (Field Programmable Gate Arrays)
- CPLD (Complex Programmable Logic Devices)

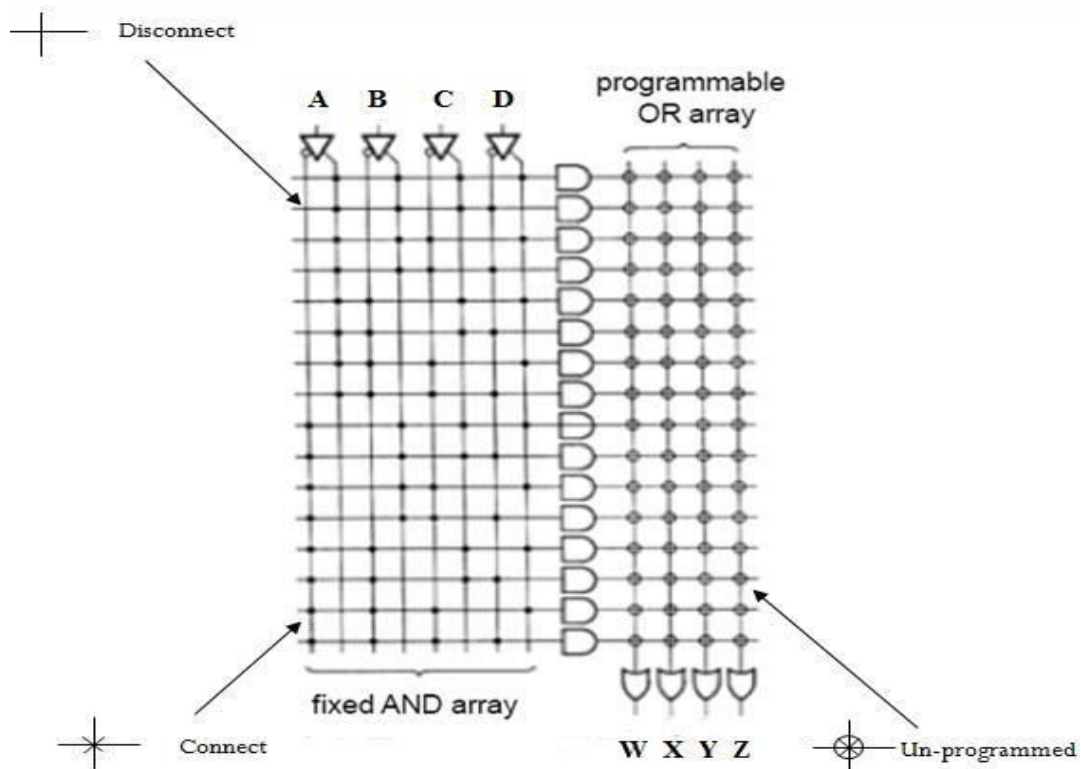
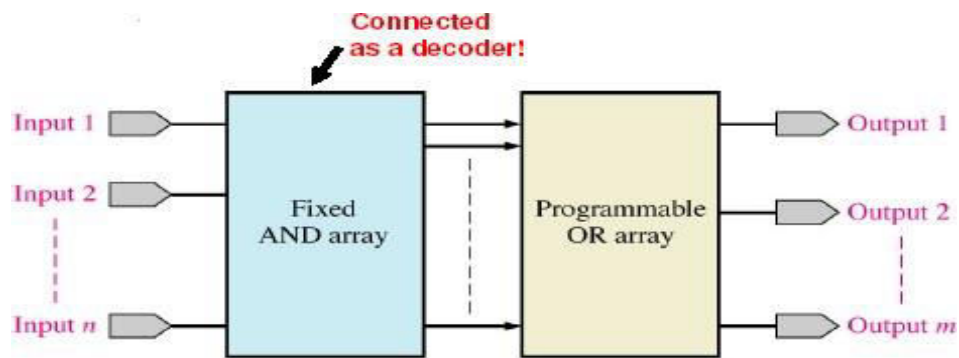
Classifications of Simple Programmable Logic Devices (SPLD)

Read Only Memory (ROM / PROM):

- N-input address lines, m-output data lines.
- Address lines point locations within ROM that store words of m bits.
- ROM size is defined by the no. of locations & the word size.
- A 256 X 4 ROM indicates that the device has 256 storage locations each holding a 4-bit word.
- This ROM would require eight address lines to access 256 locations.
- ROM of 32 X 8 has 32 memory locations each of 8-bit word and requires 5 address lines.
- Structure of ROM is as shown below;



- So No. of address lines (Input Lines) = n.
- No. of data lines (Output Lines) = m.
- ROM size = 2ⁿ X m.
- Size of decode which is to be used = n X 2ⁿ.
- ROM Consists of an array of semiconductor devices interconnected to store an array of binary data.
- Can't be changed once burned in.



- Conceptually, consist of a decoder and a memory array.

Circuit diagram of ROM

Advantages:

- Design become extremely easy.
- It is possible to change or modify the design quickly.
- Reduced cost.
- Modification takes less time than SSI/MSI circuits.

Disadvantages:

- Increase in power requirement.
- Complete circuit is not utilizes

- Increase in size with increase in number of input variables.

EXAMPLE 1: Tabulate the truth for an 8 X 4 ROM / PROM that implements the following four Boolean functions:

$$A(X,Y,Z) = \sum m(1,3,4,6)$$

$$B(X,Y,Z) = \sum m(2,4,5,7)$$

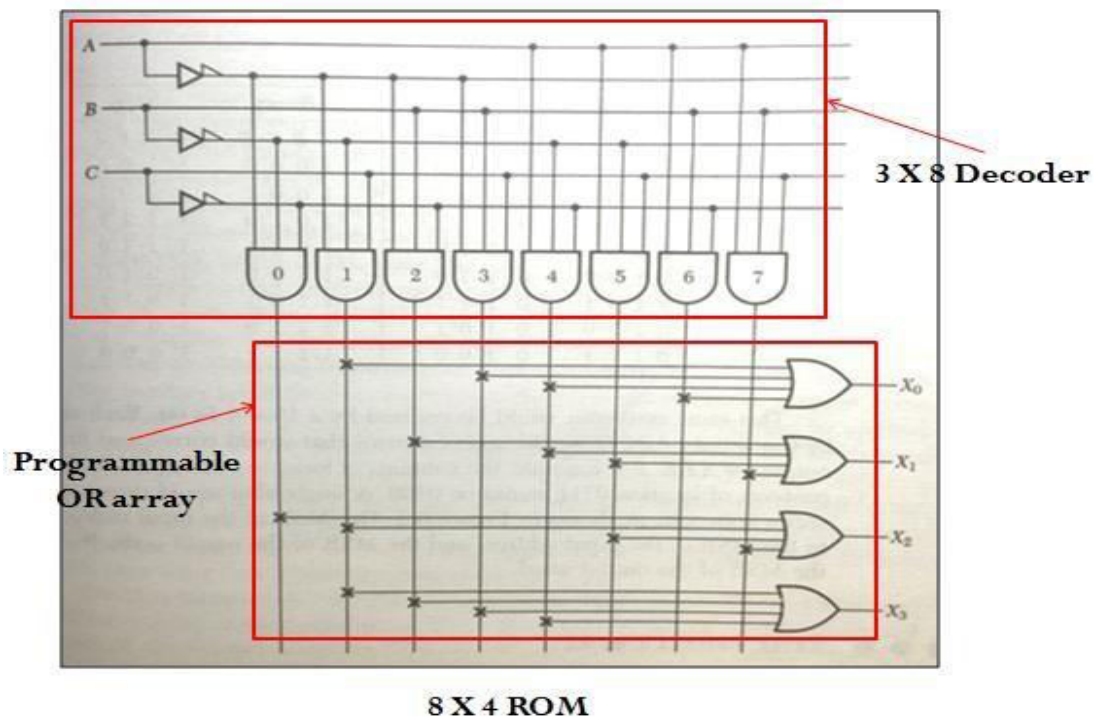
$$C(X,Y,Z) = \sum m(0,1,5,7)$$

$$D(X,Y,Z) = \sum m(1,2,3,4)$$

ANS:

Here, total No. of inputs are three □ A,B,C Total No. of outputs are four □ A,B,C,D ROM size is = 8 X 4 So, according to $2^n \times m$ □ Inputs = $n = 3$, Output = $m = 4$, Size of Decoder = $n \times 2^n = 3 \times 8$

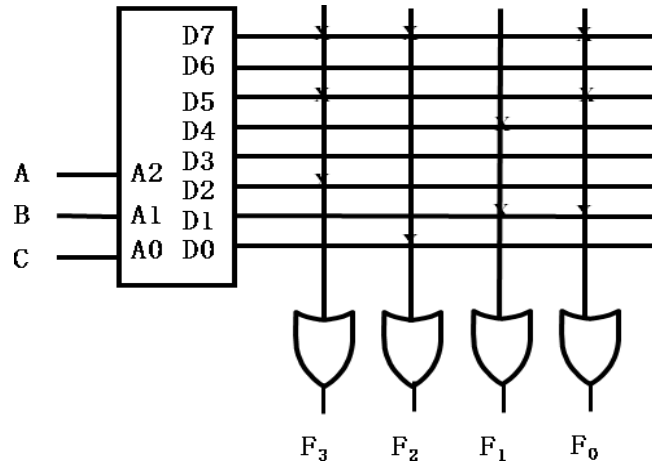
INPUT	OUTPUT			
	A	B	C	D
X'Y'Z'(000)			1	0
X'Y'Z (001)			1	1
X'YZ' (010)			0	1
X'YZ (011)			0	1
XY'Z' (100)			0	1
XY'Z (101)			1	0
XYZ' (110)			0	0
XYZ (111)			1	0



Here, * = Programmed Connection

● = Fixed Connection

EXAMPLE 2: What are functions F_3, F_2, F_1 and F_0 in terms of (A_2, A_1, A_0) ?



ANS:

- $F_3 = D_7 + D_5 + D_2 = A_2A_0 + A_2'A_1A_0'$
- $F_2 = D_7 + D_0 = A_2A_1A_0 + A_2'A_1'A_0'$
- $F_1 = D_4 + D_1 = A_2 A_1'A_0' + A_2'A_1'A_0$
- $F_0 = D_7 + D_5 + D_1 = A_2A_0 + A_1'A_0$

EXAMPLE 3: Tabulate the truth for an 8 X 4 ROM / PROM that implements the following four Boolean functions:

$$A(X,Y,Z) = \sum m(3,6,7);$$

$$B(X,Y,Z) = \sum m(0,1,4,5,6)$$

$$C(X,Y,Z) = \sum m(2,3,4);$$

$$D(X,Y,Z) = \sum m(2,3,4,7)$$

ANS:

ROM size is = 8 X 4 So, according $2^n \times m$

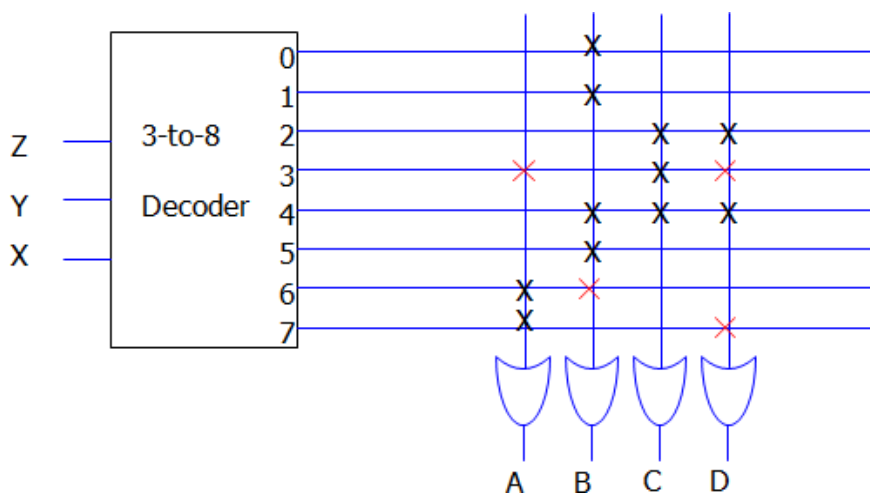
Inputs = $n = 3$, Output = $m = 4$,

Size of Decoder = $n \times 2^n = 3 \times 8$

Inputs are -> X,Y,Z

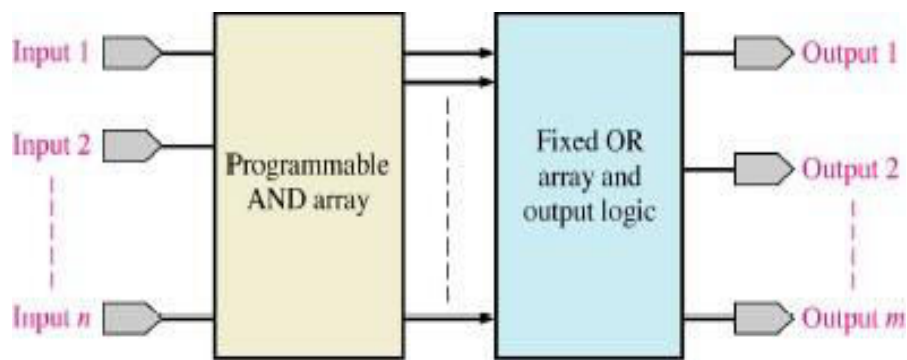
Outputs are -> A,B,C,D

Inputs			Outputs			
X	Y	Z	A	B	C	D
0	0	0	0	1	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	0	0
1	1	0	1	1	0	0
1	1	1	1	0	0	1

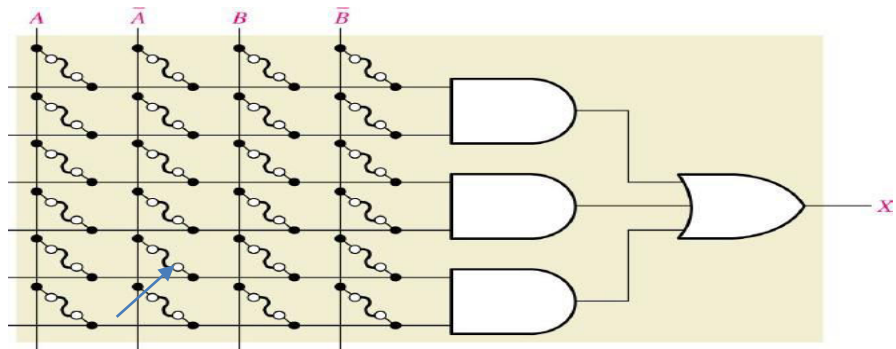


Programmable Array Logic (PAL):

- PAL is most commonly used type of PLD. It is a programmable array of logic gates.
- The array of logic gates is on single chip and it is in the AND-OR configuration.
- The special feature of PLA is that a programmable AND array and fixed OR array.
- Also note that in each OR gate in the OR array gets input from some of the AND gates. That means output of all AND gates are not applied to any of the OR gates



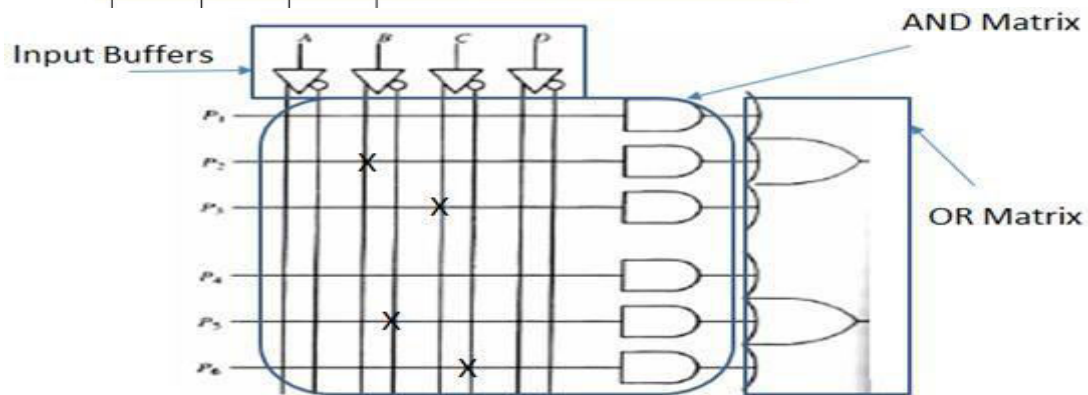
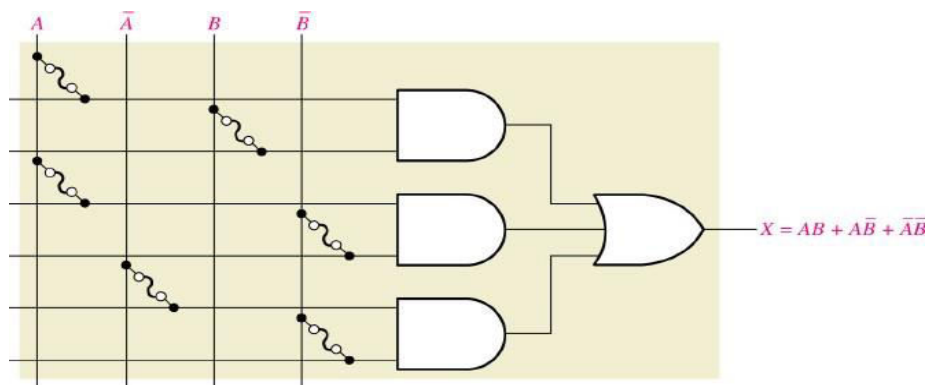
- Un-programmed and programmed PAL are shown in below figure.



Un-programmed PAL

Programmed PAL

- In Un-programmed PAL all the links are connected with Fusible Link as shown in below figure.
- As per required output function one needs to burn the fusible link and this kind of PAL is known as a programmed PAL.
- Simplified representation of PAL is shown in below figure.



Input Buffers:

- Input buffer in a PAL is used for avoiding the loading of sources connected at the inputs.
- The buffer produce inverted and non-inverted versions of their corresponding inputs.

- One such buffer is used for each of the input lines as shown in above figure.

AND Matrix:

- AND matrix is shown as above figure.
- The (X) mark indicate that a connection is present. Each AND gate has 2^M input which are shown only by a single line (e.g. A,B,C, etc....). Where M is the No. of inputs.
- When a logic function is to be implemented, we have to program the array. In programming the desired connections are left with the (X) marks and such mark is not used when connection is not required.

OR Matrix:

- OR matrix is shown as above figure. In PAL fixed OR array is used so there is no need to do programming to the OR array.
- No. of OR arrays are equal to the required No. of functions at the output.

Input and Output Circuit:

- The input and output circuit of PAL are similar to those PLAs.
- The No. of fusible link in PAL is equal to $2^M \times n$. where M = No. of available inputs and n = Corresponds to No. of product terms.

Advantages:

- For given internal complexity, a PAL can have larger N and M.
- Some PALs have outputs that can be complemented, adding POS functions.
- No multilevel circuit implementations in ROM (without external connections from output to input). PAL has outputs from OR terms as internal inputs to all AND terms, making implementation of multi-level circuits easier.

Disadvantages:

- $n \times m$ ROM guaranteed to implement any m functions of n inputs. PAL may have too few inputs to the OR gates.

Designing steps of Combinational Circuits using PAL

- STEP 1: Prepare the Truth Table.
- STEP 2: Write a Boolean expression in SOP form.
- STEP 3: Reduce / Find the Boolean expressions using K-map or reducing Boolean expression method.
- STEP 4: List of product terms for each of the function and decide total No. of AND & OR gates required.
- STEP 5: Decide connections of AND and OR matrix & draw logic diagram.

NOTE: Out of first three steps, all three steps may not require in all examples

EXAMPLE 1: A combinational circuit is defined by given truth Table for that Implement the circuit

Truth Table

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

using PAL

ANS:

- STEP 1: Prepare the Truth Table.
This step is not required in this example because it is given.
- STEP 2: Write a Boolean expression in SOP form.
This step is not required in this example
- STEP 3: Find the Boolean expressions using K-map or reducing Boolean expression method
- STEP 4: List of product terms for each of the function and decide total No. of AND & OR gates required.

$$W = A + BD + BC$$

$$X = BC'$$

$$Y = B + C$$

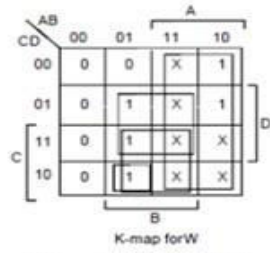
$$Z = A'B'C'D + BCD + AD' + B'CD'$$

Total No. of AND gate = 16 (B'cz maximum No. of product terms are in function Z (4 product terms) and total No. of functions are 4 so $4 \times 4 = 16$).

Total No. of OR gates = Total No. of required functions at the output side (W,X,Y,Z) = 4.

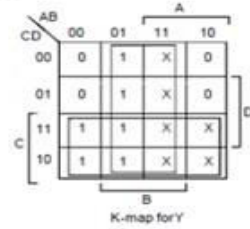
- STEP 5: Decide connections of AND and OR matrix & draw logic diagram

For W:



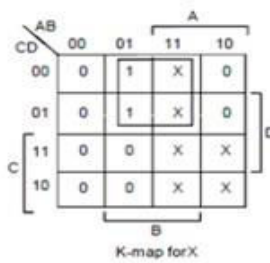
$$W = A + BD + BC$$

For Y:



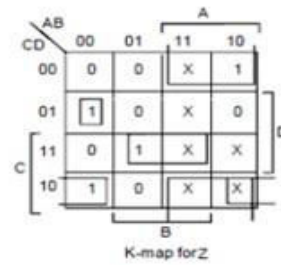
$$Y = B + C$$

For X:

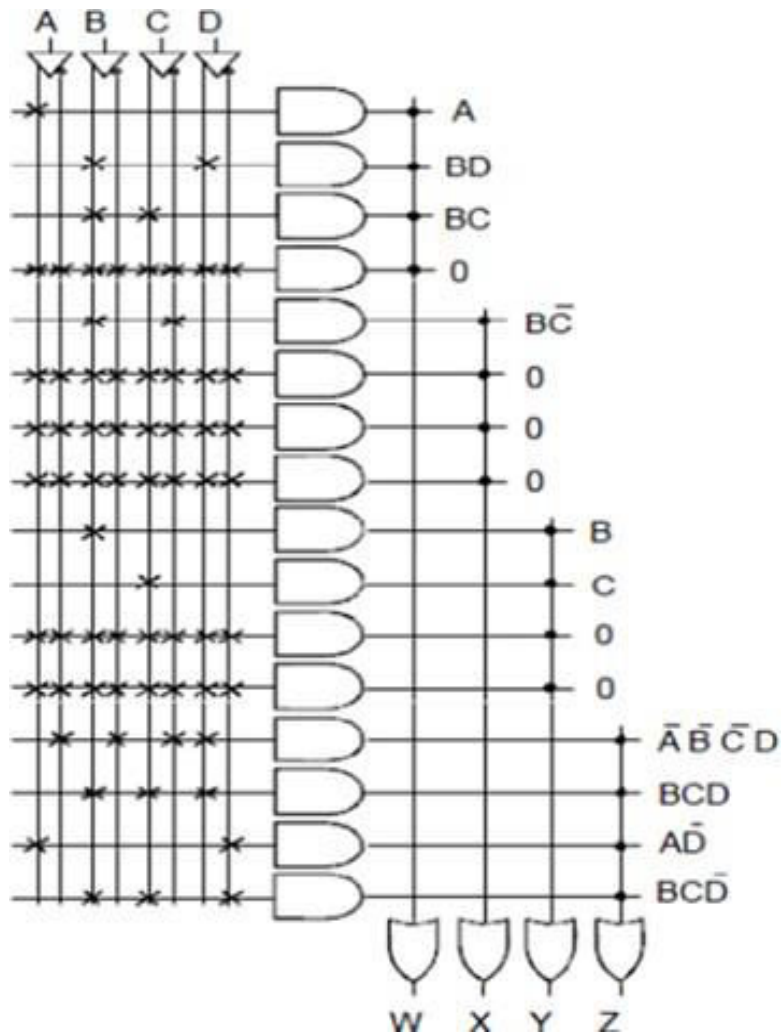


$$X = BC'$$

For Z:

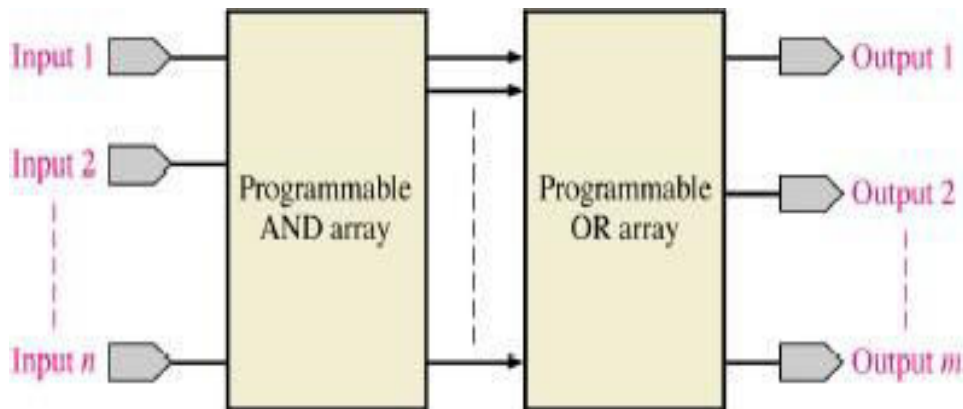


$$Z = A'B'C'D + BCD + AD' + B'CD'$$



Programmable Logic Array (PLA):

- A PLD generally consist of programmable array of logic gates. Interconnections are made with the array inputs.
- PLA consist two levels of logic, an AND-plane and an OR-plane, where both levels are programmable.
- The outputs are connected to the device pins through inverting or non-inverting buffers and flip flops.
- The basic block diagram of a PLA is shown in below figure.
- Here programmable AND matrix can be used to implement the product terms in the SOP form and the programmable OR array can be used for implementing the sum of the product terms.
- Logic gates used can be two level AND-OR, NAND-NAND or NOR-NOR configuration. Sometimes AND-OR-EXOR configuration is also used. But generally AND-OR is most preferable configuration.
- Simplified representation of PLA is shown in below figure.



Simplified block diagram of PLA

Simplified representation of PLA

In PLA No. of outputs = No. of OR gates.

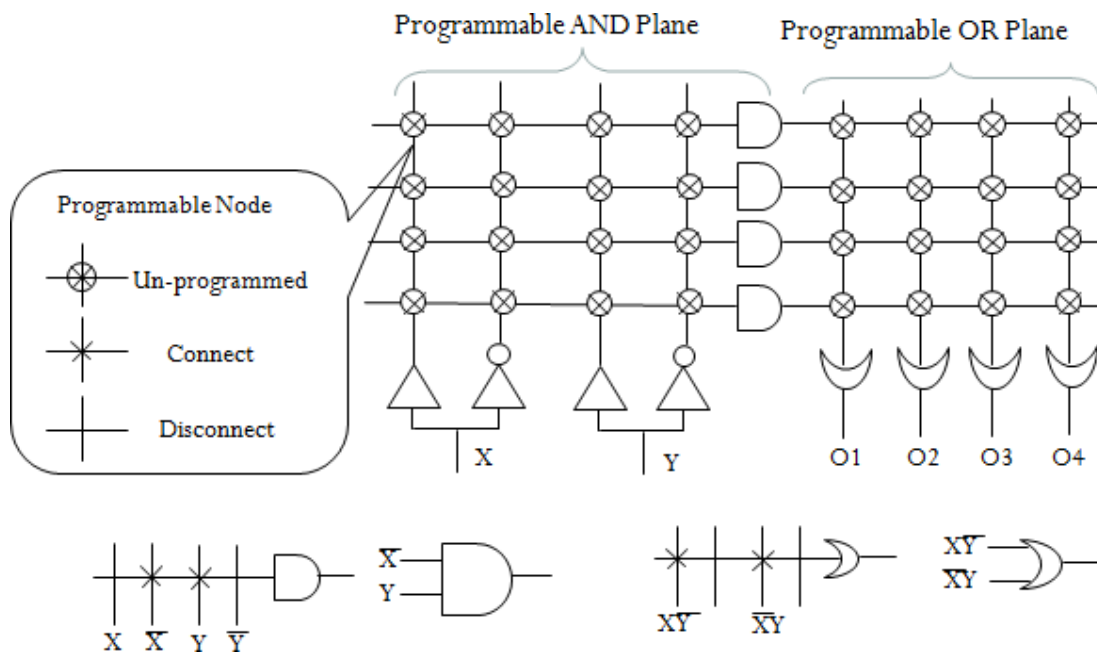
And No. of Product terms = No. of AND gates

Size of PLA is defines as $M \times P \times N$

Where $M = \text{No. of inputs}$, $P = \text{No. of product terms}$, $N = \text{No. of outputs}$

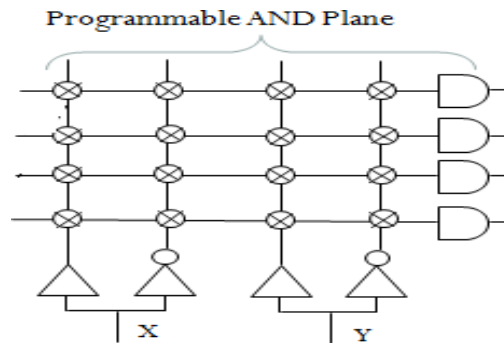
Input Buffers:

- Input buffer in PLA is used for avoiding the loading of sources connected at the inputs.



- Buffer of two types namely, inverted buffers and non-inverted buffers as shown in below figure.
- One such buffer is used in each of the M input lines.



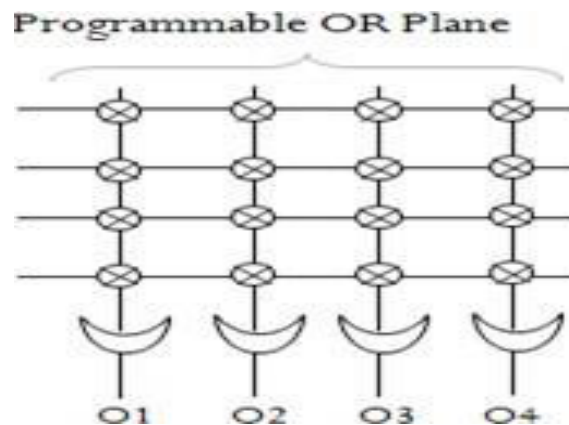


AND Matrix:

AND matrix

- The X indicates that a connection is present. Each AND gate has 2M inputs which are shown only by single line where, M is No. of inputs (e.g. A,B,C, etc....).
- When a logic function is to be implemented, we have to program the array. In programming the desired connections are left with the (X) marks and such mark is not used when connection is not required.

OR Matrix:



OR Matrix:

- Above figure shows simplified representation of the OR matrix.
- It is possible to program the OR matrix, by open circuiting the unwanted fusible links. The open fusible links are equivalent to a '0' at the input of corresponding OR gate.

Applications of PLA:

1. We can implement combinational circuit using PLA. For this only output buffers are used.
2. We can also implement sequential circuit using PLA. For implement this flip flops and buffers are included in output stage.

Designing steps of Combinational Circuits using PLA

- STEP 1: Prepare the Truth Table.

- STEP 2: Write a Boolean expression in SOP form.
- STEP 3: Reduce / Find the Boolean expressions using K-map or reducing Boolean expression method.
- STEP 4: List of unique product terms.
- STEP 5: PLA table implementation.
- STEP 6: Decide connections of AND and OR matrix & draw logic diagram.

NOTE: Out of first three steps, all three steps may not require in all examples

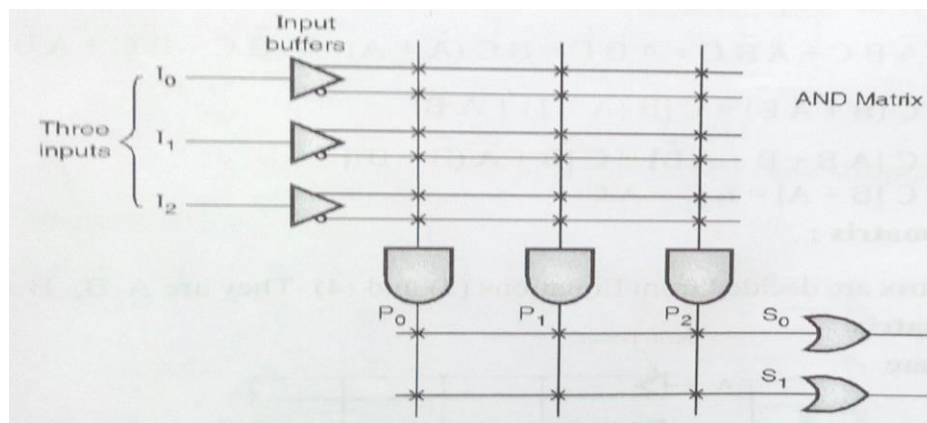
EXAMPLE 1: Draw combinational circuit for a PLA with three inputs, three product terms and two outputs.

ANS:

Given, No. of inputs = 3 = I_0, I_1, I_2

No. of Outputs = 2 = No. of OR gates.

NO. of product terms = No. of AND gates



Difference between ROM, PAL and PLA.

ROM/PROM	PAL	PLA
Consist of fixed AND gate array and programmable OR array.	Consist of programmable AND gate array and fixed OR array.	Consist of programmable AND gate array and Programmable OR array.
Medium speed	High speed (only one programmable gates arrays)	Slow (two programmable gates arrays)

Cheap component (High-volume)	Intermediate cost (less than PLA)	Most expensive (Most complex in design)
Not flexible	Not flexible	Offering maximum programming flexibility
It is possible to decode any minterms	We can get any desired minterms by programming the AND matrix	We can get any desired minterms by programming the AND & OR matrix
SOP function in the standard form only can be implemented	Any SOP function can be implemented	Any SOP function can be implemented

Traffic Light Controller

A simple traffic light controller can be implemented by a state machine that has a state diagram such as the one shown in Figure 1.

The circuit has control over a North-South road and an East-West road. The North-South lights are controlled by outputs called nsr, nsy, and nsg (North-South red, yellow, green). The East-West road is controlled by similar outputs called ewr, ewy, ewg. The cycle is controlled by an input called TIMER which controls the length of the two greenlight cycles (s0 represents the EW green; s2 represents the NS green.) When TIMER = 1, a transition from s0 to s1 or from s2 to s3 is possible. This accompanies a change from green to yellow on the active road. The light on the other road stays red. An unconditional transition follows, changing the yellow light to red on one road and the red light to green on the other.

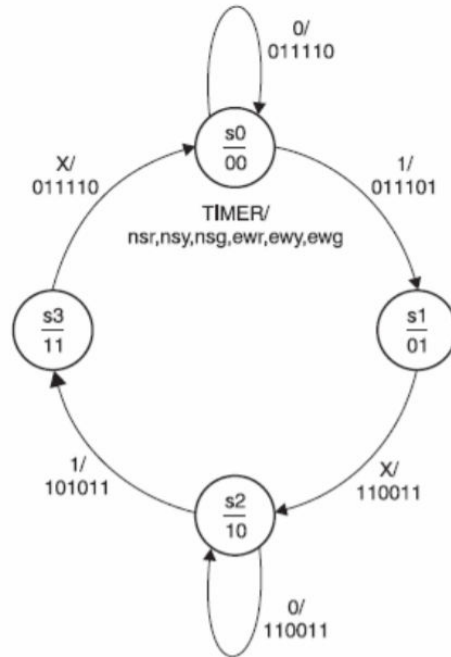


Figure 1. State Diagram for a Traffic Light Controller

The outputs in the state diagram of Figure 30.1 are indicated as active-LOW. For the Altera DE-1 board, you must use active- HIGH outputs instead.

The cycle can be set to any length by changing the signal on the TIMER input. (The yellow light will always be on for one clock pulse.) For ease of observation, we will use a cycle of ten clock pulses for any one road: 4 clocks GREEN, 1 clock YELLOW, 5 clocks RED. This can be generated by a mod-5 counter, as shown in Figure 2.

The clock divider brings the on-board oscillator frequency down to the range of visible observation for our CPLD board. A 22-bit counter is suitable for the Altera DE-1.

Calculate the frequency of the state machine clock for your CPLD board. $f = \underline{\hspace{2cm}}$

Creating a Traffic Light Controller in VHDL

1. Create a VHDL file to implement the traffic controller state diagram shown in Figure 1, combined with the mod-5 cycle timer, but not the clock divider. The individual modules and the top-level of the hierarchy must all be done in VHDL.
2. Create a simulation that shows the combined operation of the output controller and cycle timer.

3. Add a clock divider to the VHDL file for the traffic controller, as shown on the logic diagram of Figure 30.2. Set the clock divider width to 22. Assign pins to the design so that the North-South lights correspond to LED1—LED3 on the CPLD board and the East-West lights correspond to LED9—LED11.
4. Assign a pin to the state machine clock (so that it can be observed directly) and make it correspond to LED16. Disable all other LEDs. Make sure that the controller outputs are at the correct active level for the LEDs on your CPLD board.
5. Download the file to the CPLD board.

Traffic Controller with Walk Signal

1. Modify the VHDL files of the previous section to implement a traffic light controller with a walk signal, as shown in the logic diagram of Figure 3. The walk signal goes on for one green cycle of the direction related to the switch. For example, when you press the NS switch, the North-South walk signal goes on for the next North-South green cycle. On the next NS green cycle, the walk signal is off unless the NS switch has been pressed again. The switches can be pressed at any time, as their states are stored in a pair of flipflops. The status of the flip-flop determines the transition to a state for which the walk signal is (or is not) active. The flip-flop is asynchronously reset by the state machine at the end of its active cycle.

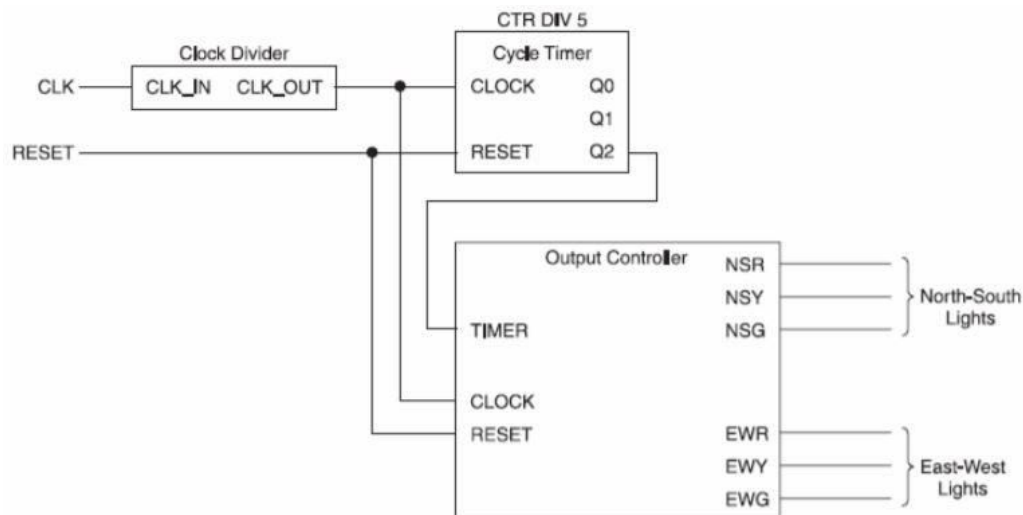


Figure.2 Logic Diagram for a Traffic Light Controller

The state diagram of Figure.1 forms the core of the new state diagram. The machine operates as follows:

- The machine resets to North-South red and East-West green (s0) and remains in this state until the TIMER input goes HIGH. Both walk signals are off.
 - The machine transitions to s1 when TIMER = 1. East-West goes yellow.
 - If the machine is in s1 and the North-South walk switch has not been pressed, the machine goes to s2. North-South is green for four clocks, until Timer = 1. EastWest is red and both walk signals are off. The machine goes to s3 when TIMER = 1. Outputs are NS yellow and EW red. Walk signals are off.
 - If the machine is in s1 and the North-South walk switch has been pressed, the machine goes to a new state, s4, which behaves exactly the same as s2, except that the NS walk signal is now on. As long as TIMER = 0, the machine remains in s4, with North-South light green and East-West light red. When TIMER = 1, s4 makes a transition to s3. In this transition, the walk signal turns off and the latch reset output (NS_WALK_RESET) goes LOW for one clock pulse. Outputs are NS yellow and EW red.
 - If the machine is in s3 and the EW walk switch has not been pressed, the machine makes a transition to s0. Outputs are NS red and EW green. Walk signals are off.
 - If the machine is in s3 and the EW walk switch has been pressed, this is stored as the HIGH state of a latch output. This is sensed at input EW_WALK_SW on the state machine and there is a transition to s5. This behaves the same as s0, except that the EW walk signal is on. As long as TIMER = 0, the machine stays in s5, with outputs North-South red and East-West green. When TIMER = 1, the machine makes a transition to s1. The walk signal turns off and a LOW pulse on EW_WALK_RESET resets the EW latch. The outputs are now NS red and EW yellow.
2. Draw the modified state diagram for the controller, as previously described, in the space provided on this page.
 3. Implement the modified design entirely in VHDL. For the D flip-flops, use the DFF component in Quartus II primitives library. In order to use the DFF primitives, the file requires the following two lines at the beginning: LIBRARY altera; USE altera.maxplus2.ALL; A VHDL component declaration for the latch can be found in the Quartus II Help menu under Primitives.
 4. Assign additional pins for the walk signals: NS walk on LED4 and EW walk on LED12.

Compile and download the file to the CPLD test board.

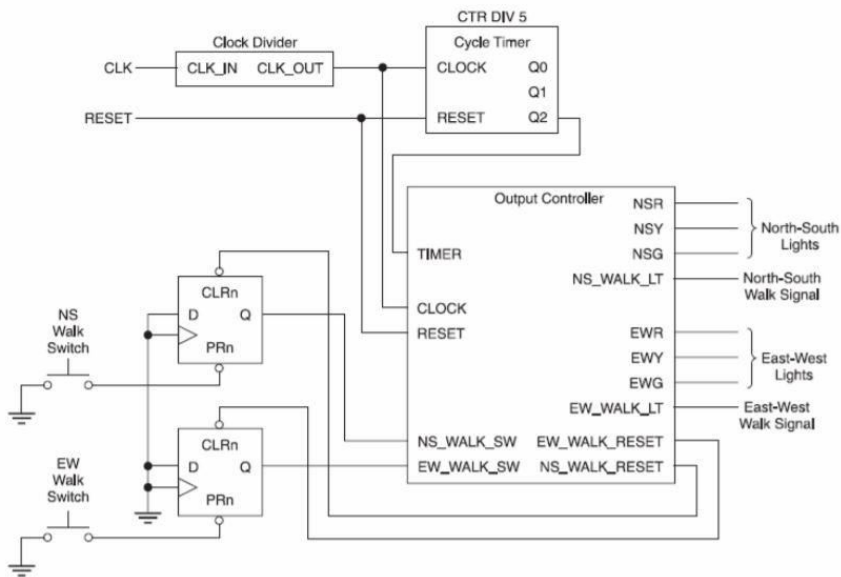
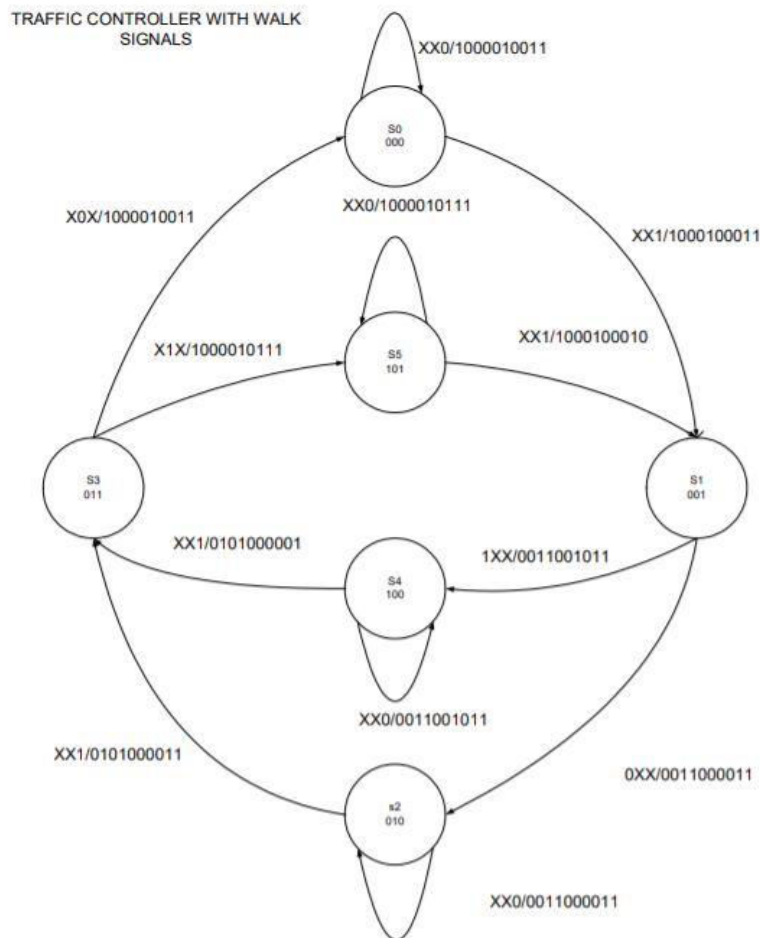
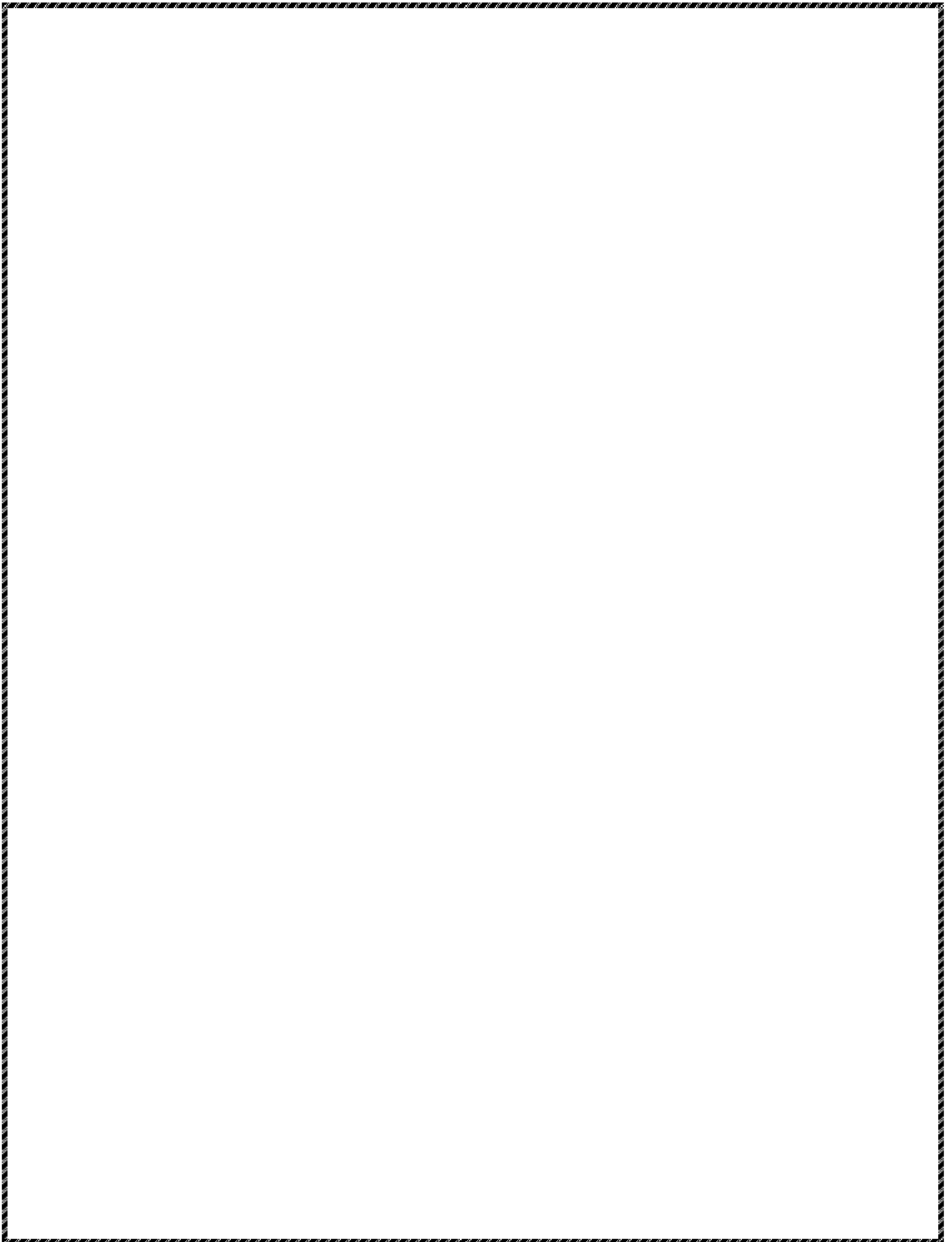


Figure.3 Logic Diagram for a Traffic Light Controller with a Walk Signal



NS_SW,EW_SW,TIMER/NSR,NSY,NSG,EWR,EWY,EWG,NS_WALK,EW_WALK,NS_RESET,EW_RESET

Figure 4. State Diagram for Walk Signal



Introduction to High-Capacity FPDs

Prompted by the development of new types of sophisticated field-programmable devices (FPDs), the process of designing digital hardware has changed dramatically over the past few years. Unlike previous generations of technology, in which board-level designs included large numbers of SSI chips containing basic gates, virtually every digital design produced today consists mostly of high-density devices. This applies not only to custom devices like processors and memory, but also for logic circuits such as state machine controllers, counters, registers, and decoders. When such circuits are destined for high-volume systems they have been integrated into high-density gate arrays. However, gate array NRE costs often are too expensive and gate arrays take too long to manufacture to be viable for prototyping or other low-volume scenarios. For these reasons, most prototypes, and also many production designs are now built using FPDs. The most compelling advantages of FPDs are instant manufacturing turnaround, low start-up costs, low financial risk and (since programming is done by the end user) ease of design changes.

The market for FPDs has grown dramatically over the past decade to the point where there is now a wide assortment of devices to choose from. A designer today faces a daunting task to research the different types of chips, understand what they can best be used for, choose a particular manufacturers' product, learn the intricacies of vendor-specific software and then design the hardware. Confusion for designers is exacerbated by not only the sheer number of FPDs available, but also by the complexity of the more sophisticated devices. The purpose of this paper is to provide an overview of the architecture of the various types of FPDs. The emphasis is on devices with relatively high logic capacity; all of the most important commercial products are discussed.

Before proceeding, we provide definitions of the terminology in this field. This is necessary because the technical jargon has become somewhat inconsistent over the past few years as companies have attempted to compare and contrast their products in literature.

Definitions of Relevant Terminology

The most important terminology used in this paper is defined below.

- *Field-Programmable Device (FPD)* — a general term that refers to any type of integrated circuit used for implementing digital hardware, where the chip can be configured by the end user to realize different designs. Programming of such a device often involves placing the chip into a special programming unit, but some chips can also be configured “in-system”. Another name for FPDs is *programmable logic devices* (PLDs); although PLDs encompass the same types of chips as FPDs, we prefer the term FPD because historically the word PLD has referred to relatively simple types of devices.
- *PLA* — a Programmable Logic Array (PLA) is a relatively small FPD that contains two levels of logic, an AND-plane and an OR-plane, where both levels are programmable (note: although PLA structures are sometimes embedded into full-custom chips, we refer here only to those PLAs that are provided as separate integrated circuits and are user-programmable).
- *PAL*^{*} — a Programmable Array Logic (PAL) is a relatively small FPD that has a programmable AND-plane followed by a fixed OR-plane
- *SPLD* — refers to any type of Simple PLD, usually either a PLA or PAL
- *CPLD* — a more Complex PLD that consists of an arrangement of multiple SPLD-like blocks on a single chip. Alternative names (that will not be used in this paper) sometimes adopted for this style of chip are Enhanced PLD (EPLD), Super PAL, Mega PAL, and others.
- *FPGA* — a Field-Programmable Gate Array is an FPD featuring a general structure that allows very high logic capacity. Whereas CPLDs feature logic resources with a wide number of inputs (AND planes), FPGAs offer more narrow logic resources. FPGAs also offer a higher ratio of flip-flops to logic resources than do CPLDs.
- *HCPLDs* — high-capacity PLDs: a single acronym that refers to both CPLDs and FPGAs. This term has been coined in trade literature for providing an easy way to refer to both types of devices. We do not use this term in the paper.
- *Interconnect* — the wiring resources in an FPD.
- *Programmable Switch* — a user-programmable switch that can connect a logic element to an interconnect wire, or one interconnect wire to another
- *Logic Block* — a relatively small circuit block that is replicated in an array in an FPD. When a

circuit is implemented in an FPD, it is first decomposed into smaller sub-circuits that can each be mapped into a logic block. The term logic block is mostly used in the context of FPGAs, but it could also refer to a block of circuitry in a CPLD.

- *Logic Capacity* — the amount of digital logic that can be mapped into a single FPD. This is usually measured in units of “equivalent number of gates in a traditional gate array”. In other words, the capacity of an FPD is measured by the size of gate array that it is comparable to. In simpler terms, logic capacity can be thought of as “number of 2-input NAND gates”.
- *Logic Density* — the amount of logic per unit area in an FPD.
- *Speed-Performance* — measures the maximum operable speed of a circuit when implemented in an FPD. For combinational circuits, it is set by the longest delay through any path, and for sequential circuits it is the maximum clock frequency for which the circuit functions properly.

In the remainder of this section, to provide insight into FPD development the evolution of FPDs over the past two decades is described. Additional background information is also included on the semiconductor technologies used in the manufacture of FPDs.

Evolution of Programmable Logic Devices

The first type of user-programmable chip that could implement logic circuits was the Programmable Read-Only Memory (PROM), in which address lines can be used as logic circuit inputs and data lines as outputs. Logic functions, however, rarely require more than a few product terms, and a PROM contains a full decoder for its address inputs. PROMS are thus an inefficient architecture for realizing logic circuits, and so are rarely used in practice for that purpose. The first device developed later specifically for implementing logic circuits was the Field-Programmable Logic Array (FPLA), or simply PLA for short. A PLA consists of two levels of logic gates: a programmable “wired” AND-plane followed by a programmable “wired” OR-plane. A PLA is structured so that any of its inputs (or their complements) can be AND’ed together in the AND-plane; each AND-plane output can thus correspond to any product term of the inputs. Similarly, each OR- plane output can be configured to produce the logical sum of any of the AND-plane outputs. With this structure, PLAs are well-suited for implementing logic functions in sum-of-products form. They are also quite versatile, since both the AND terms and OR terms can have many inputs (this feature is often referred to as *wide* AND and OR gates).

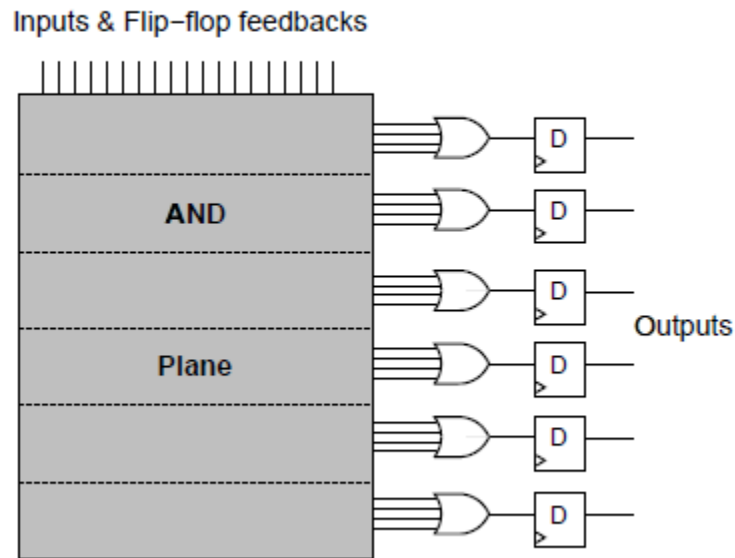


Figure 1 - Structure of a PAL.

When PLAs were introduced in the early 1970s, by Philips, their main drawbacks were that they were expensive to manufacture and offered somewhat poor speed-performance. Both disadvantages were due to the two levels of configurable logic, because programmable logic planes were difficult to manufacture and introduced significant propagation delays. To overcome these weaknesses, Programmable Array Logic (PAL) devices were developed. As Figure 1 illustrates, PALs feature only a single level of programmability, consisting of a programmable “wired” AND-plane that feeds fixed OR-gates. To compensate for lack of generality incurred because the OR-plane is fixed, several variants of PALs are produced, with different numbers of inputs and outputs, and various sizes of OR-gates. PALs usually contain flip-flops connected to the OR-gate outputs so that sequential circuits can be realized. PAL devices are important because when introduced they had a profound effect on digital hardware design, and also they are the basis for some of the newer, more sophisticated architectures that will be described shortly. Variants of the basic PAL architecture are featured in several other products known by different acronyms. All small PLDs, including PLAs, PALs, and PAL-like devices are grouped into a single category called Simple PLDs (SPLDs), whose most important characteristics are low cost and very high pin-to-pin speed-performance.

As technology has advanced, it has become possible to produce devices with higher capacity than SPLDs. The difficulty with increasing capacity of a strict SPLD architecture is that the structure of the programmable logic-planes grow too quickly in size as the number of inputs is increased. The only feasible way to provide large capacity devices based on SPLD architectures is then to integrate multiple SPLDs onto a single chip and provide interconnect to programmably connect the SPLD blocks together. Many commercial FPD products exist on the market today with this basic structure, and are collectively referred to as Complex PLDs (CPLDs).

CPLDs were pioneered by Altera, first in their family of chips called Classic EPLDs, and then in three additional series, called MAX 5000, MAX 7000 and MAX 9000. Because of a rapidly growing market for large FPDs, other manufacturers developed devices in the CPLD category and there are now many choices available. All of the most important commercial products will be described in Section 2. CPLDs provide logic capacity up to the equivalent of about 50 typical SPLD devices, but it is somewhat difficult to extend these architectures to higher densities. To build FPDs with very high logic capacity, a different approach is needed.

The highest capacity general purpose logic chips available today are the traditional gate arrays sometimes referred to as *Mask-Programmable Gate Arrays* (MPGAs). MPGAs consist of an array of pre-fabricated transistors that can be customized into the user's logic circuit by connecting the transistors with custom wires. Customization is performed during chip fabrication by specifying the metal interconnect, and this means that in order for a user to employ an MPGA a large setup cost is involved and manufacturing time is long. Although MPGAs are clearly not FPDs, they are mentioned here because they motivated the design of the user-programmable equivalent: Field-Programmable Gate Arrays (FPGAs). Like MPGAs, FPGAs comprise an array of uncommitted circuit elements, called *logic blocks*, and interconnect resources, but FPGA configuration is performed through programming by the end user. An illustration of a typical FPGA architecture appears in Figure 2. As the only type of FPD that supports very high logic capacity, FPGAs have been responsible for a major shift in the way digital circuits are designed.

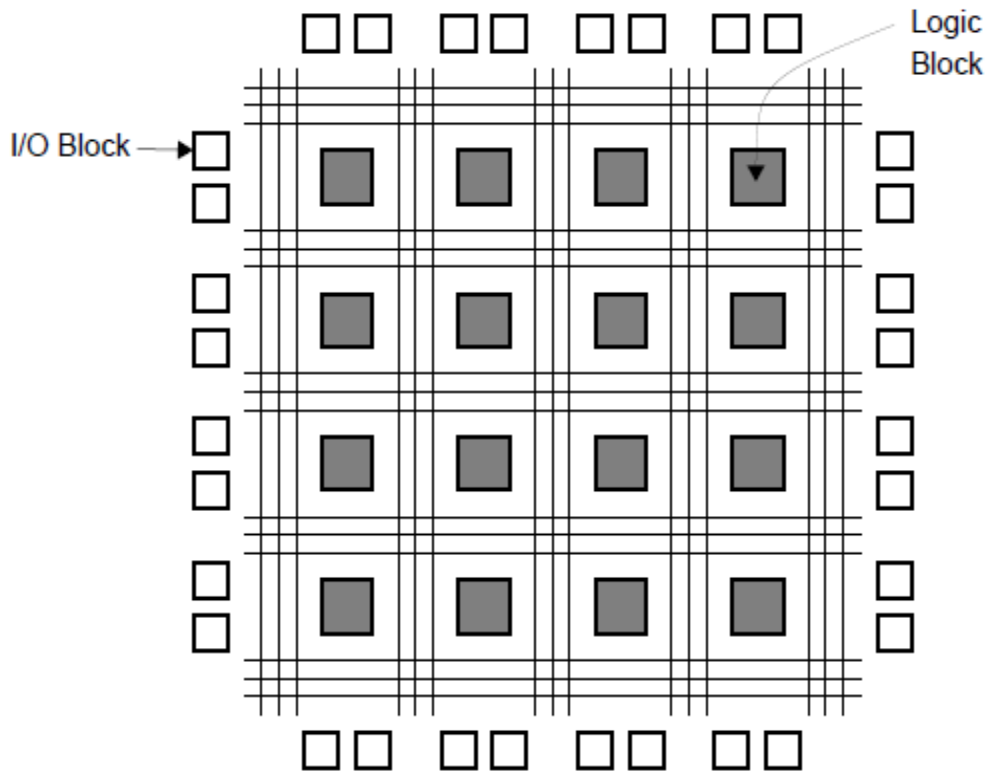


Figure 2 - Structure of an FPGA.

Figure 3 summarizes the categories of FPDs by listing the logic capacities available in each of the three categories. In the figure, “equivalent gates” refers loosely to “number of 2-input NAND gates”. The chart serves as a guide for selecting a specific device for a given application, depending on the logic capacity needed. However, as we will discuss shortly, each type of FPD is inherently better suited for some applications than for others. It should also be mentioned that there exist other special-purpose devices optimized for specific applications (e.g. state machines, analog gate arrays, large interconnection problems). However, since use of such devices is limited they will not be described here. The next sub-section discusses the methods used to implement the user-programmable switches that are the key to the user-customization of FPDs.

User-Programmable Switch Technologies

The first type of user-programmable switch developed was the *fuse* used in PLAs. Although fuses are still used in some smaller devices, we will not discuss them here because they are quickly being replaced by newer technology. For higher density devices, where CMOS

dominates the IC industry, different approaches to implementing programmable switches have been developed. For CPLDs the main switch technologies (in commercial products) are floating gate transistors like those used in EPROM and EEPROM, and for FPGAs they are SRAM and antifuse. Each of these is briefly discussed below.

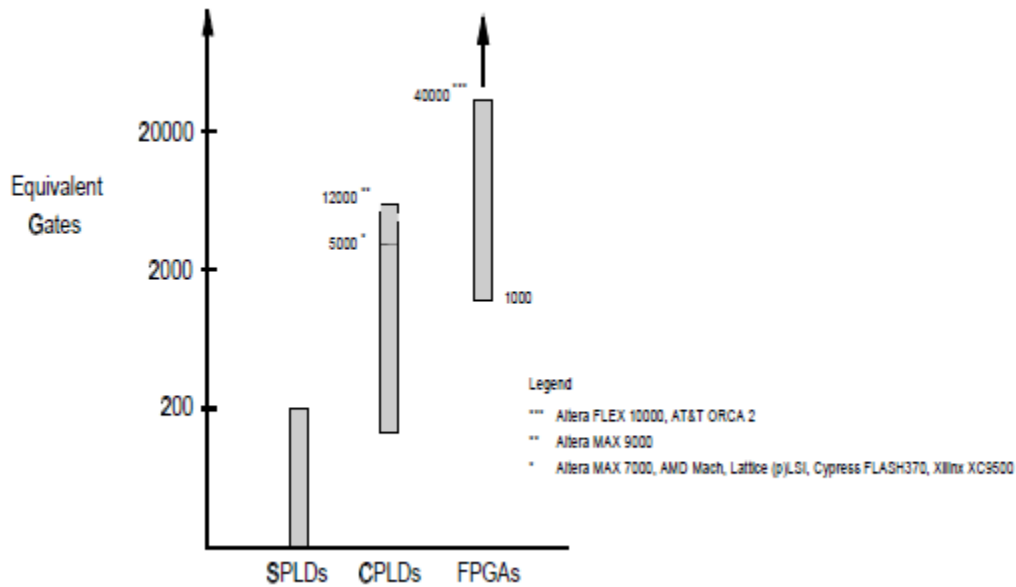


Figure 3 - FPD Categories by Logic Capacity.

An EEPROM or EPROM transistor is used as a programmable switch for CPLDs (and also for many SPLDs) by placing the transistor between two wires in a way that facilitates implementation of wired-AND functions. This is illustrated in Figure 4, which shows EPROM transistors as they might be connected in an AND-plane of a CPLD. An input to the AND-plane can drive a product wire to logic level '0' through an EPROM transistor, if that input is part of the corresponding product term. For inputs that are not involved for a product term, the appropriate EPROM transistors are programmed to be permanently turned off. A diagram for an EEPROM-based device would look similar.

Although there is no technical reason why EPROM or EEPROM could not be applied to FPGAs, current commercial FPGA products are based either on SRAM or antifuse technologies, as discussed below.

An example of usage of SRAM-controlled switches is illustrated in Figure 5, showing two applications of SRAM cells: for controlling the gate nodes of pass-transistor switches and to control the select lines of multiplexers that drive logic block inputs. The figures gives an example of the connection of one logic block (represented by the AND-gate in the upper left corner) to another through two pass-transistor switches, and then a multiplexer, all controlled by SRAM cells. Whether an FPGA uses pass-transistors or multiplexers or both depends on the particular product.

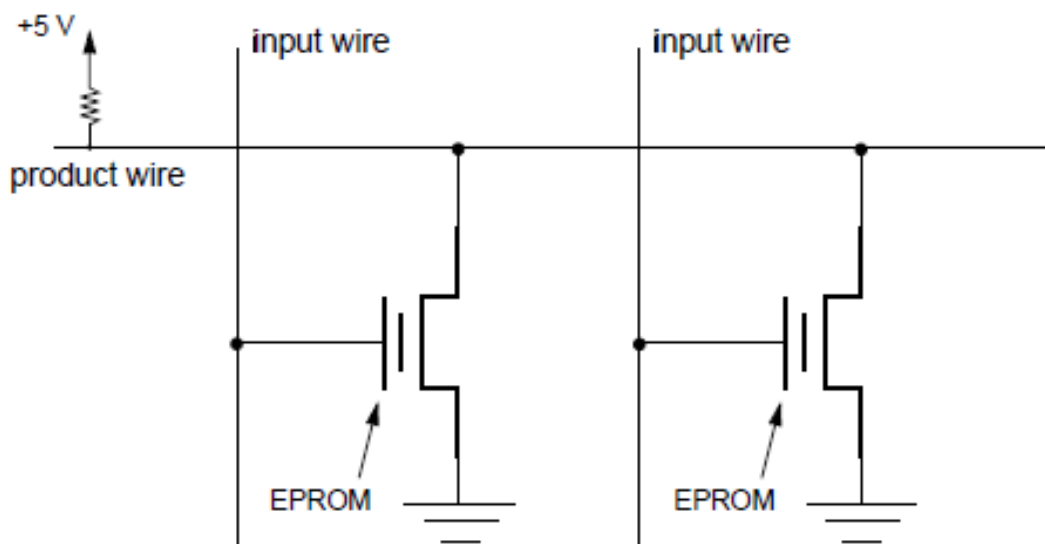


Figure 4 - EPROM Programmable Switches.

The other type of programmable switch used in FPGAs is the antifuse. Antifuses are originally open-circuits and take on low resistance only when programmed. Antifuses are suitable for FPGAs because they can be built using modified CMOS technology. As an example, Actel's antifuse structure, known as PLICE [Ham88], is depicted in Figure 6. The figure shows that an antifuse is positioned between two interconnect wires and physically consists of three sandwiched layers: the top and bottom layers are conductors, and the middle layer is an insulator. When unprogrammed, the insulator isolates the top and bottom layers, but when programmed the insulator changes to become a low-resistance link. PLICE uses Poly-Si and n+ diffusion as conductors and ONO (see [Ham88]) as an insulator, but other antifuses rely on metal for conductors, with

amorphous silicon as the middle layer [Birk92][Marp94].

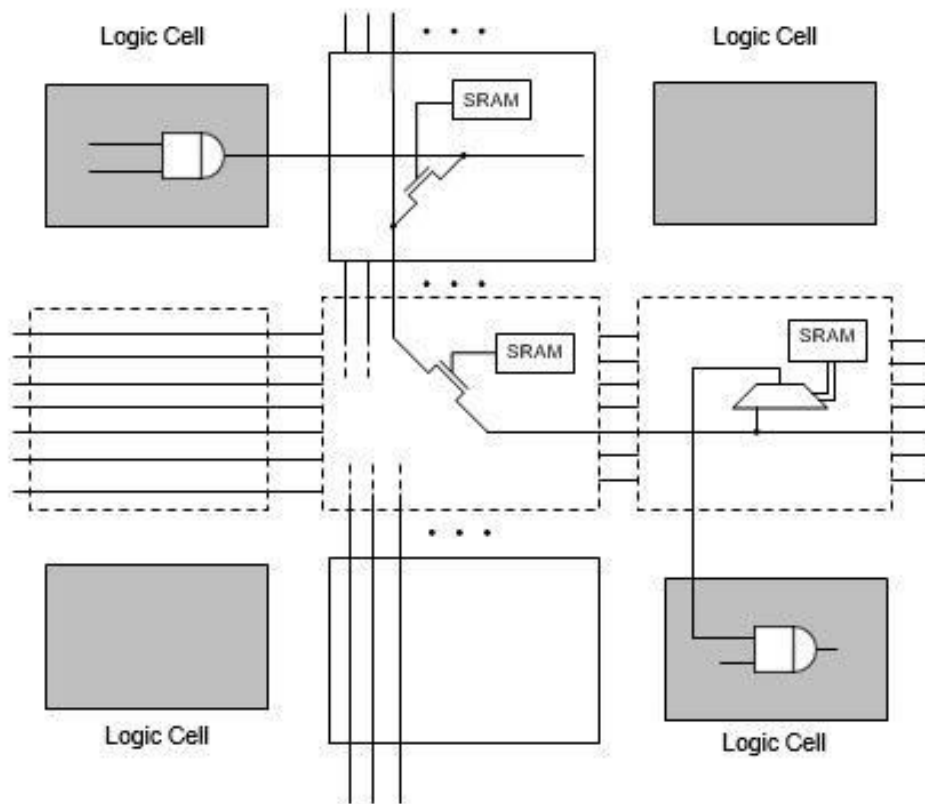


Figure 5 - SRAM-controlled Programmable Switches.

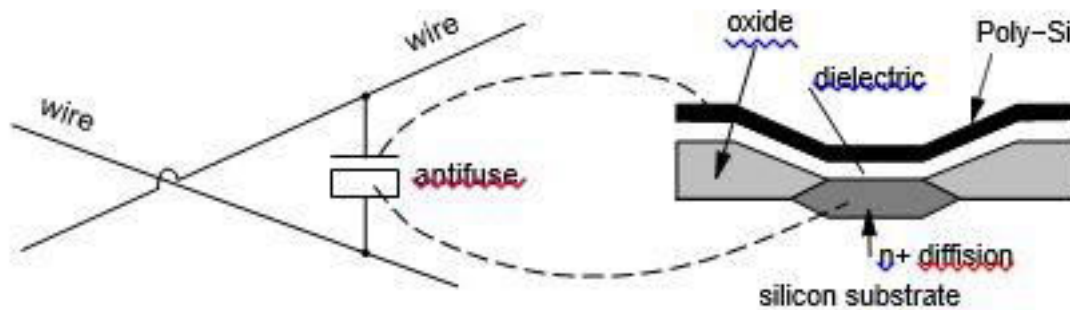


Figure 6 - Actel Antifuse Structure.

Table 1 lists the most important characteristics of the programming technologies discussed in this section. The left-most column of the table indicates whether the programmable switches are one-time programmable (OTP), or can be re-programmed (RP). The next column lists whether the switches are volatile, and the last column names the underlying transistor technology.

Name	Re-programmable	Volatile	Technology
Fuse	no	no	Bipolar
EPROM	yes out of circuit	no	UVCMOS
EEPROM	yes in circuit	no	EECMOS
SRAM	yes in circuit	yes	CMOS
Antifuse	no	no	CMOS+

Table 1 - *Summary of Programming Technologies.*

Computer Aided Design (CAD) Flow for FPDs

When designing circuits for implementation in FPDs, it is essential to employ Computer-Aided Design (CAD) programs. Such software tools are discussed briefly in this section to provide a feel for the design process involved.

CAD tools are important not only for complex devices like CPLDs and FPGAs, but also for SPLDs. A typical CAD system for SPLDs would include software for the following tasks: initial design entry, logic optimization, device fitting, simulation, and configuration. This design flow is illustrated in Figure 7, which also indicates how some stages feedback to others. Design entry may be done either by creating a schematic diagram with a graphical CAD tool, by using a text-based system to describe a design in a simple hardware description language, or with a mixture of design entry methods. Since initial logic entry is not usually in an optimized form, algorithms are employed to optimize the circuits, after which additional algorithms analyse the resulting logic equations and “fit” them into the SPLD. Simulation is used to verify correct operation, and the user would return to the design entry step to fix errors. When a design simulates correctly it can be loaded into a programming unit and used to configure an SPLD. One final detail to note about Figure 7 is that while the original design entry step is performed manually by the designer, all other steps are carried out automatically by most CAD systems.

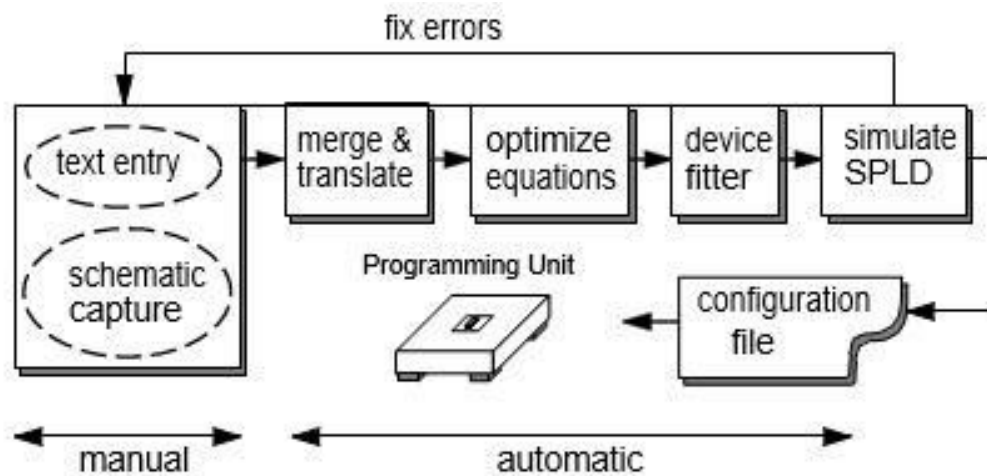


Figure 7 - CAD Design Flow for SPLDs.

The steps involved for implementing circuits in CPLDs are similar to those for SPLDs, but the tools themselves are more sophisticated. Because the devices are complex and can accommodate large designs, it is more common to use a mixture of design entry methods for different modules of a complete circuit. For instance, some modules might be designed with a small hardware description language like ABEL, others drawn using a symbolic schematic capture tool, and still others described via a full-featured hardware description language such as VHDL. Also, for CPLDs the process of “fitting” a design may require steps similar to those described below for FPGAs, depending on how sophisticated the CPLD is. The necessary software for these tasks is supplied either by the CPLD manufacturer or a third party.

The design process for FPGAs is similar to that for CPLDs, but additional tools are needed to support the increased complexity of the chips. The major difference is in the “device fitter” step that comes after logic optimization and before simulation, where FPGAs require at least three steps: a technology mapper to map from basic logic gates into the FPGA’s logic blocks, placement to choose which specific logic blocks to use in the FPGA, and a router to allocate the wire segments in the FPGA to interconnect the logic blocks. With this added complexity, the CAD tools might require a fairly long period of time (often more than an hour or even several hours) to complete their tasks.

Overview of Commercially Available FPDs

This section provides many examples of commercial FPD products. SPLDs are first discussed briefly, and then details are given for all of the most important CPLDs and FPGAs. The reader who is interested in more details on the commercial products is encouraged to contact the manufacturers, or their distributors, for the latest data sheets* .

Commercially Available SPLDs

As the staple for digital hardware designers for the past two decades, SPLDs are very important devices. SPLDs represent the highest speed-performance FPDs available, and are inexpensive. However, they are also fairly straight-forward and well understood, so this paper will discuss them only briefly.

Two of the most popular SPLDs are the PALs produced by Advanced Micro Devices (AMD) known as the 16R8 and 22V10. Both of these devices are industry standards and are widely second-sourced by various companies. The name “16R8” means that the PAL has a maximum of 16 inputs (there are 8 dedicated inputs and 8 input/outputs), and a maximum of 8 outputs. The “R” refers to the type of outputs provided by the PAL and means that each output is “registered” by a D flip-flop. Similarly, the “22V10” has a maximum of 22 inputs and 10 outputs. Here, the “V” means each output is “versatile” and can be configured in various ways, some configurations registered and some not.

Another widely used and second sourced SPLD is the Altera Classic EP610. This device is similar in complexity to PALs, but it offers more flexibility in the way that outputs are produced and has larger AND- and OR- planes. In the EP610, outputs can be registered and the flip-flops are configurable as any of D, T, JK, or SR.

In addition to the SPLDs mentioned above many other products are available from a wide array of companies. All SPLDs share common characteristics, like some sort of logic planes (AND, OR, NOR, or NAND), but each specific product offers unique features that may be particularly attractive for some applications. A partial list of companies that offer SPLDs includes: AMD, Altera, ICT, Lattice, Cypress, and Philips-Signetics. Since some of these SPLDs have complexity approaching that found in CPLDs, the paper will now move on to more sophisticated devices.

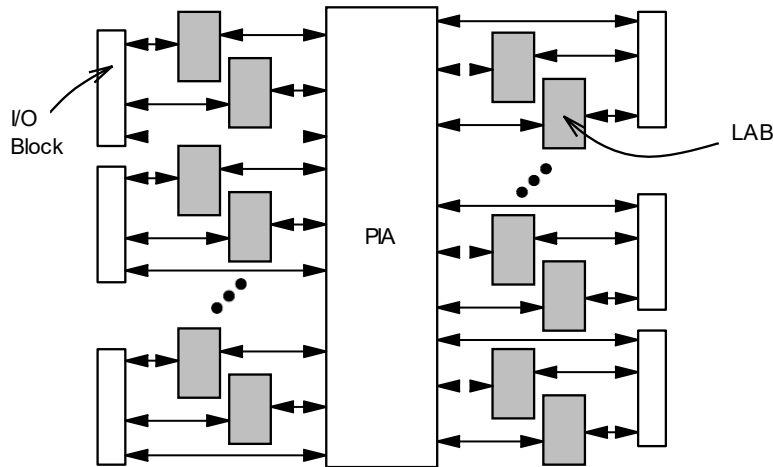


Figure 8 - Altera MAX 7000 Series.

Commercially Available CPLDs

As stated earlier, CPLDs consist of multiple SPLD-like blocks on a single chip. However, CPLD products are much more sophisticated than SPLDs, even at the level of their basic SPLD-like blocks. In this section, CPLDs are discussed in detail, first by surveying the available commercial products and then by discussing the types of applications for which CPLDs are best suited. Sufficient details are presented to allow a comparison between the various competing products, with more attention being paid to devices that we believe are in more widespread use than others.

Altera CPLDs

Altera has developed three families of chips that fit within the CPLD category: MAX 5000, MAX 7000, and MAX 9000. Here, the discussion will focus on the MAX 7000 series, because it is widely used and offers state-of-the-art logic capacity and speed-performance. MAX 5000 represents an older technology that offers a cost effective solution, and MAX 9000 is similar to MAX 7000, except that MAX 9000 offers higher logic capacity (the industry's highest for CPLDs).

The general architecture of the Altera MAX 7000 series is depicted in Figure 8. It comprises an array of blocks called Logic Array Blocks (LABs), and interconnect wires called a Programmable Interconnect Array (PIA). The PIA is capable of connecting any LAB input or

output to any other LAB. Also, the inputs and outputs of the chip connect directly to the PIA and to LABs. A LAB can be thought of as a complex SPLD-like structure, and so the entire chip can be considered to be an array of SPLDs. MAX 7000 devices are available both based in EPROM and EEPROM technology. Until recently, even with EEPROM, MAX 7000 chips could be programmable only “out-of-circuit” in a special-purpose programming unit; however, in 1996 Altera released the 7000S series, which is reprogrammable “in-circuit”.

The structure of a LAB is shown in Figure 9. Each LAB consists of two sets of eight *macrocells* (shown in Figure 10), where a macrocell comprises a set of programmable product terms (part of an AND-plane) that feeds an OR-gate and a flip-flop. The flip-flops can be configured as D type, JK, T, SR, or can be transparent. As illustrated in Figure 10, the number of inputs to the

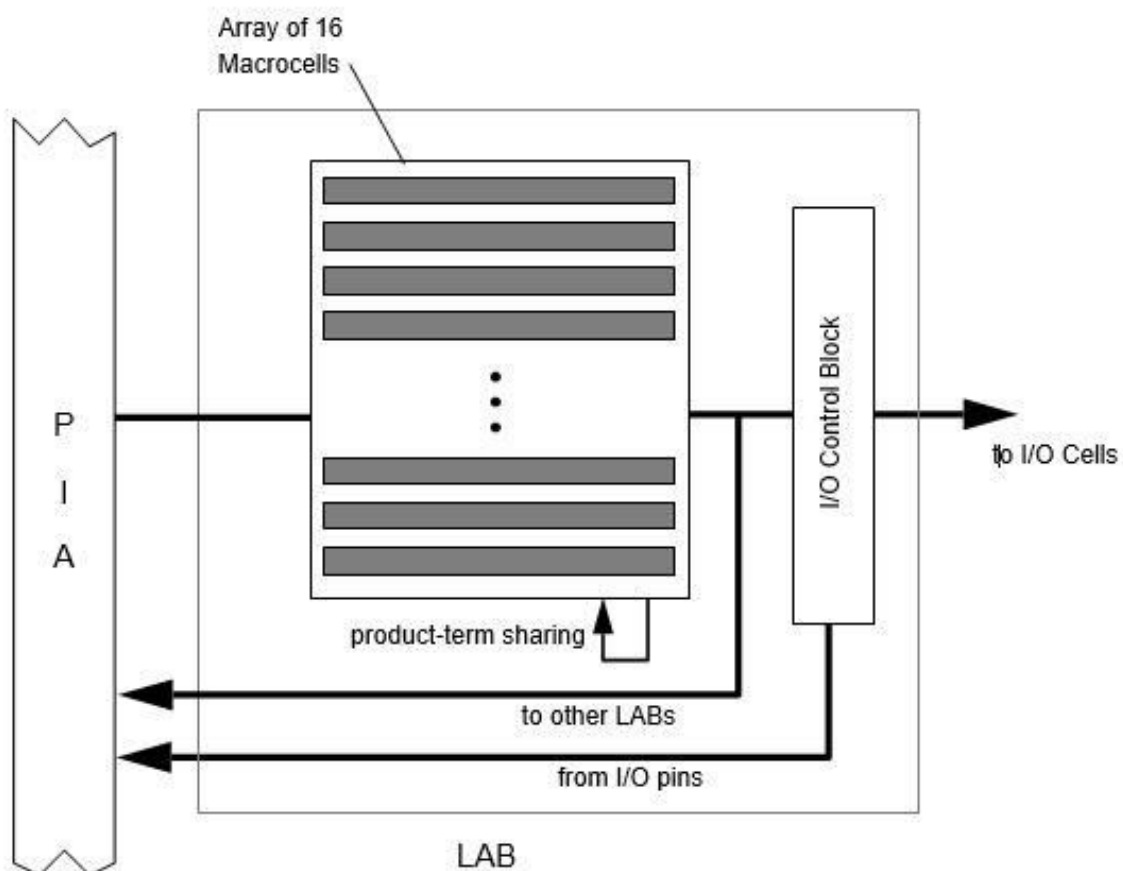


Figure 9 - Altera MAX 7000 Logic Array Block (LAB).

OR-gate in a macrocell is variable; the OR-gate can be fed from any or all of the five product terms within the macrocell, and in addition can have up to 15 extra product terms from macrocells in the same LAB. This product term flexibility makes the MAX 7000 series LAB more efficient in terms of chip area because typical logic functions do not need more than five product terms, and the architecture supports wider functions when they are needed. It is interesting to note that variable sized OR-gates of this sort are not available in basic SPLDs (see Figure 1). Similar features of this kind are found in other CPLD architectures discussed shortly.

Besides Altera, several other companies produce devices that can be categorized as CPLDs. For example, AMD manufactures the Mach family, Lattice has the (i)pLSI series, Xilinx produces a CPLD series that they call XC7000 (unrelated to the Altera MAX 7000 series) and has announced a new family called XC9500, and ICT has the PEEL array. These devices are discussed in the following sub-sections.

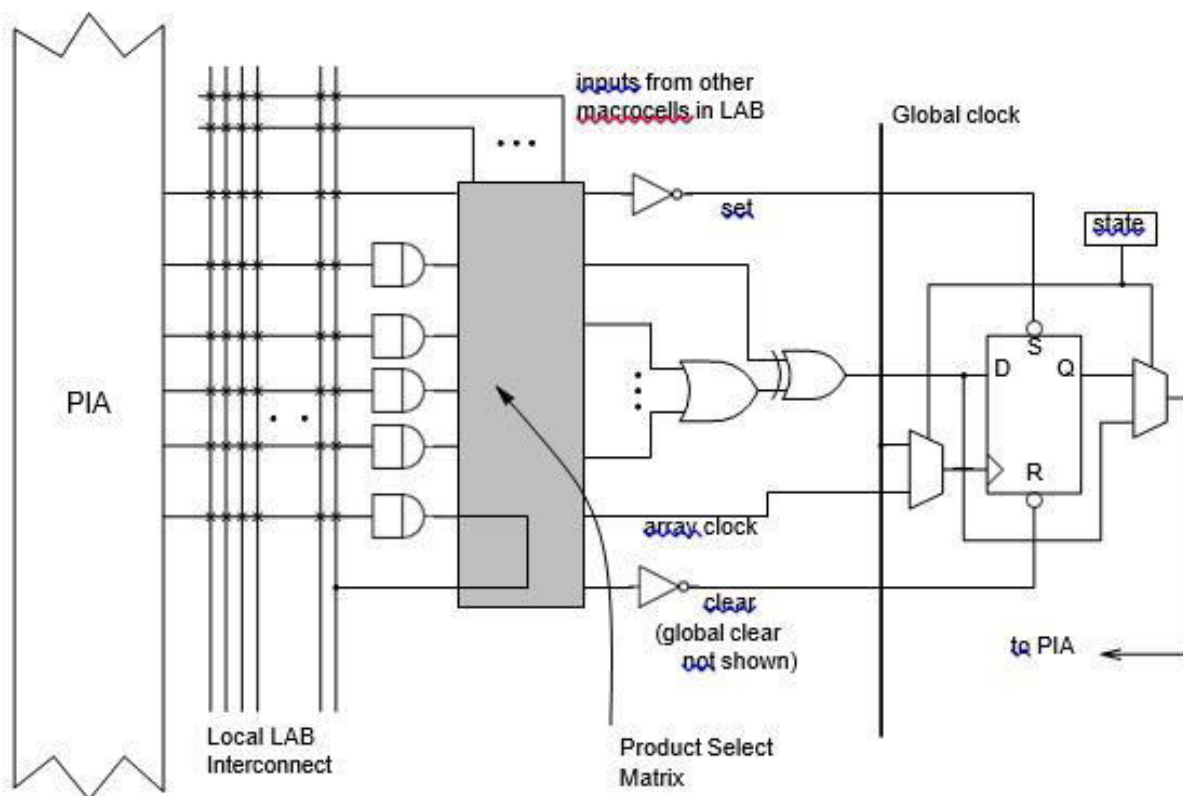


Figure 10 - MAX 7000 Macrocell.

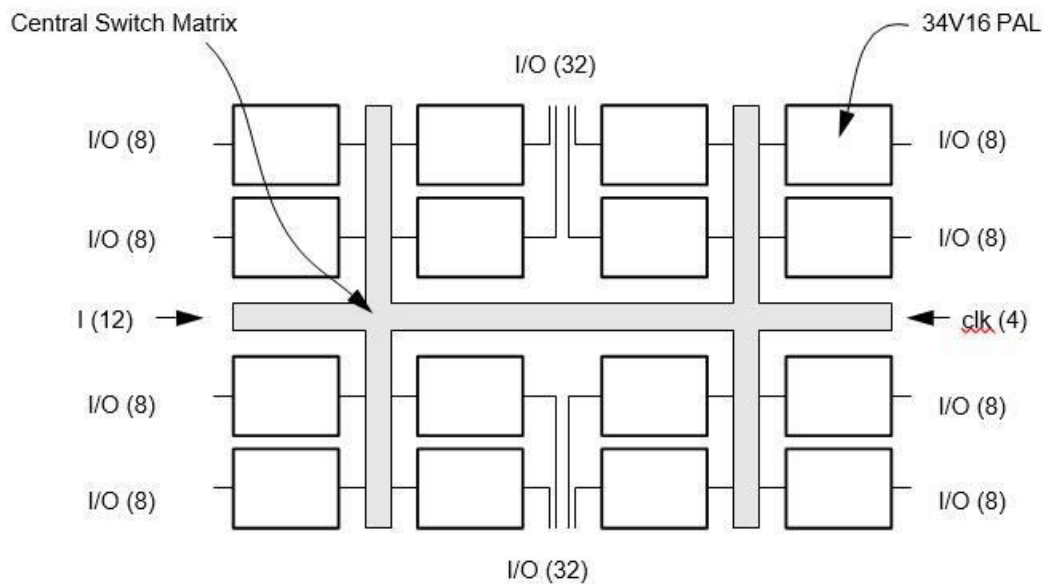


Figure 11 - Structure of AMD Mach 4 CPLDs.

Advanced Micro Devices (AMD) CPLDs

AMD offers a CPLD family with five sub-families called Mach 1 to Mach 5. Each Mach device comprises multiple PAL-like blocks: Mach 1 and 2 consist of optimized 22V16 PALs, and Mach 3 and 4 comprise several optimized 34V16 PALs, and Mach 5 is similar but offers enhanced speed-performance. All Mach chips are based on EEPROM technology, and together the five sub-families provide a wide range of selection, from small, inexpensive chips to larger state-of-the-art ones. This discussion will focus on Mach 4, because it represents the most advanced currently available parts in the Mach family.

Figure 11 depicts a Mach 4 chip, showing the multiple 34V16 PAL-like blocks, and the interconnect, called Central Switch Matrix, for connecting the blocks together. Chips range in size from 6 to 16 PAL blocks, which corresponds roughly to 2000 to 5000 equivalent gates and are in-circuit programmable. All connections in Mach 4 between one PAL block and another (even from a PAL block to itself) are routed through the Central Switch Matrix. The device can thus be viewed not only as a collection of PALs, but also as a single large device. Since all connections travel through the same path, timing delays of circuits implemented in Mach 4 are

predictable.

A Mach 4 PAL-like block is depicted in Figure 12. It has 16 outputs and a total of 34 inputs (16 of which are the outputs fed-back), so it corresponds to a 34V16 PAL. However, there are two key differences between this block and a normal PAL: 1. there is a *product term allocator* between the AND-plane and the macrocells (the macrocells comprise an OR-gate, an EX-OR gate and a flip-flop), and 2. there is an *output switch matrix* between the OR-gates and the I/O pins. These two features make a Mach 4 chip easier to use, because they “decouple” sections of the PAL block. More specifically, the product term allocator distributes and shares product terms from the AND-plane to whichever OR-gates require them. This is much more flexible than the fixed-size OR-gates in regular PALs. The output switch matrix makes it possible for any macro-cell output (OR-gate or flip-flop) to drive any of the I/O pins connected to the PAL block. Again, flexibility is enhanced over a PAL, where each macrocell can drive only one specific I/O pin. Mach 4’s combination of in-system programmability and high flexibility promote easy hardware design changes.

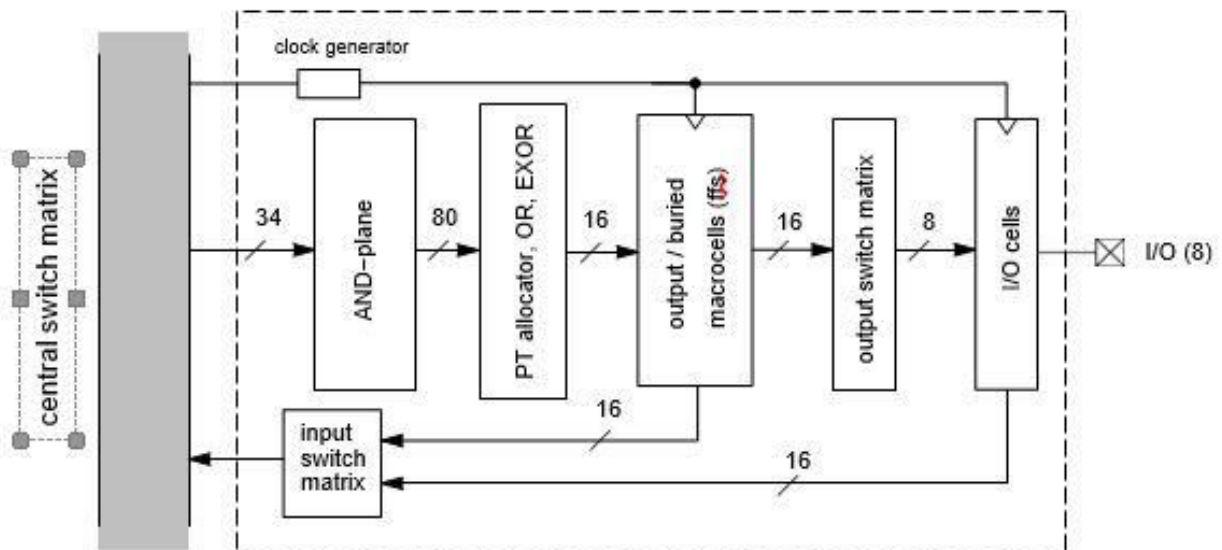


Figure 12 - AMD Mach 4 PAL-like (34V16) Block.

Lattice CPLDs

Lattice offers a complete range of CPLDs, with two main product lines: the Lattice pLSI consist of three families of EEPROM CPLDs, and the ispLSI are the same as the pLSI devices, except that they are in-system programmable. For both the pLSI and ispLSI products, Lattice offers three families that have different logic capacities and speed-performance.

Lattice's earliest generation of CPLDs is the pLSI and ispLSI 1000 series. Each chip consists of a collection of SPLD-like blocks, described in more detail later, and a global routing pool to connect blocks together. Logic capacity ranges from about 1200 to 4000 gates. Pin-to-pin delays are 10 nsec. Lattice also offers a CPLD family called the 2000 series, which are relatively small CPLDs, with between 600 and 2000 gates that offer a higher ratio of macrocells to I/O pins and higher speed-performance than the 1000 series. At 5.5 nsec pin-to-pin delays, the 2000 series offers state-of-the-art speed.

Lattice's 3000 series represents their largest CPLDs, with up to 5000 gates. Pin-to-pin delays for this device are about 10-15 nsec. In terms of other chips discussed so far, the 3000 series functionality is most similar to AMD's Mach 4. The 3000 series offers some enhancements over the other Lattice parts to support more recent design styles, such as JTAG boundary scan.

The general structure of a Lattice pLSI or ispLSI device is indicated in Figure 13. Around the outside edges of the chip are the bi-directional I/Os, which are connected both to the Generic Logic Blocks (GLBs) and the Global Routing Pool (GRP). As the fly-out on the right side of the figure shows, the *GLBs* are small PAL-like blocks that consist of an AND-plane, product term allocator, and macrocells. The *GRP* is a set of wires that span the entire chip and are available to connect the GLB inputs and outputs together. All interconnections pass through the GRP, so timing between levels of logic in the Lattice chips is fully predictable, much as it is for the AMD Mach devices.

Cypress FLASH370 CPLDs

Cypress has recently developed a family of CPLD products that are similar to both the AMD and Lattice devices in several ways. The Cypress CPLDs, called FLASH370 are based on FLASH EEPROM technology, and offer speed-performance of 8.5 to 15 nsec pin-to-pin delays. The FLASH370 parts are not in-system programmable. Recognizing that larger chips need more I/Os, FLASH370 provides more I/Os than competing products, featuring a linear relationship between the number of macrocells and number of bi-directional I/O pins. The smallest parts have 32 macrocells and 32 I/Os and the largest 256 macrocells and 256 I/Os.

Figure14 shows that FLASH370 has a typical CPLD architecture with multiple PAL-like blocks and a programmable interconnect matrix (PIM) to connect them. Within each PAL-like block, there is an AND-plane that feeds a product term allocator that directs from 0 to 16 product terms to each of 32 OR-gates. Note that in the feed-back path from the macrocell outputs to the

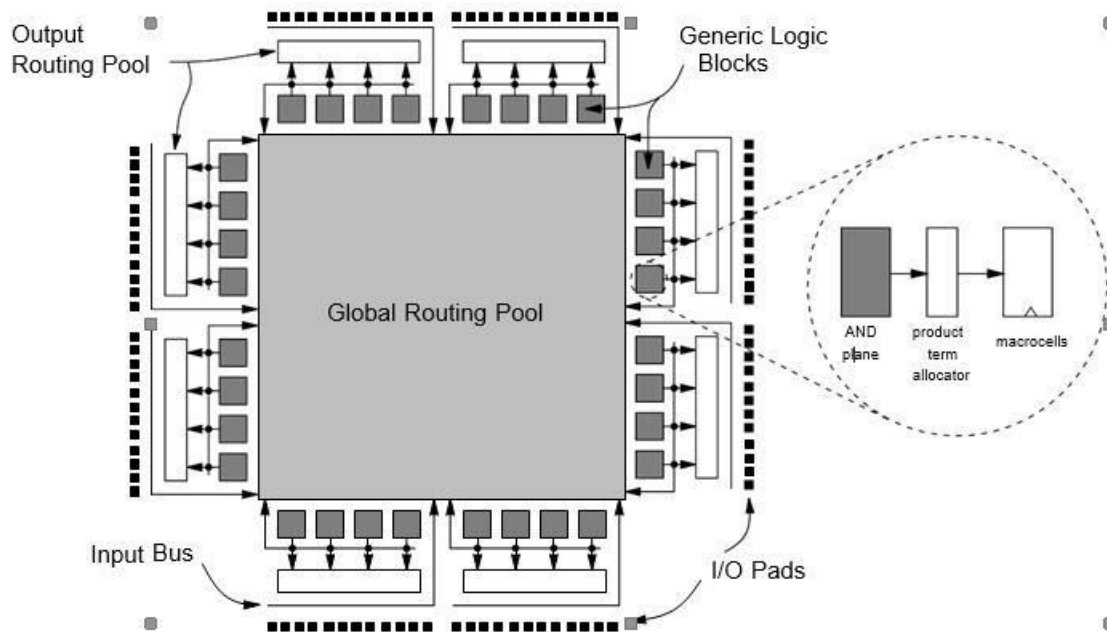


Figure 13 - Lattice (i)PLSI Architecture.

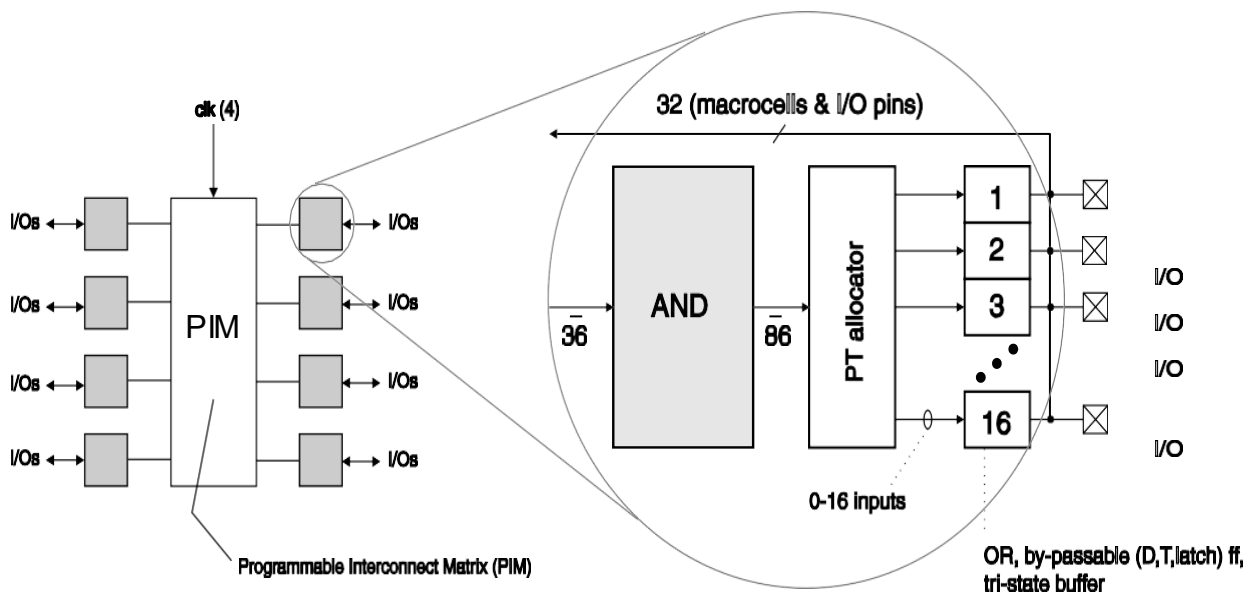


Figure 14 - Architecture of Cypress FLASH370 CPLDs.

PIM, there are 32 wires; this means that a macrocell can be buried (not drive an I/O pin) and yet the I/O pin that could be driven by the macrocell can still be used as an input. This illustrates another type of flexibility available in PAL-like blocks in CPLDs, but not present in normal PALs.

Xilinx XC7000 CPLDs

Although Xilinx is mostly a manufacturer of FPGAs, they also offer a selection of CPLDs, called XC7000, and have announced a new CPLD family called XC9500. There are two main families in the XC7000 offering: the 7200 series, originally marketed by Plus Logic as the Hiper EPLDs, and the 7300 series, developed by Xilinx. The 7200 series are moderately small devices, with about 600 to 1500 gates capacity, and they offer speed-performance of about 25 nsec pin-to-pin delays. Each chip consists of a collection of SPLD-like blocks that each have 9 macrocells. The macrocells in the 7200 series are different from those in other CPLDs in that each macrocell includes two OR-gates and each of these OR-gates is input to a two-bit Arithmetic Logic Unit (ALU). The ALU can produce any functions of its two inputs, and its output feeds a configurable flip-flop. The Xilinx 7300 series is an enhanced version of the 7200, offering more capacity (up to 3000 gates when the entire family becomes available) and higher speed-performance. Finally, the new XC9500, when available, will offer in-circuit programmability with 5 nsec pin-to-pin delays and up to 6200 logic gates.

Altera FLASHlogic CPLDs

Altera's FLASHlogic, previously known as Intel's FLEXlogic, features in-system programmability and provides on-chip SRAM blocks, a unique feature among CPLD products. The upper part of Figure 15 illustrates the architecture of FLASHlogic devices; it comprises a collection of PAL-like blocks, called Configurable Function Blocks (CFBs), that each represents an optimized 24V10 PAL.

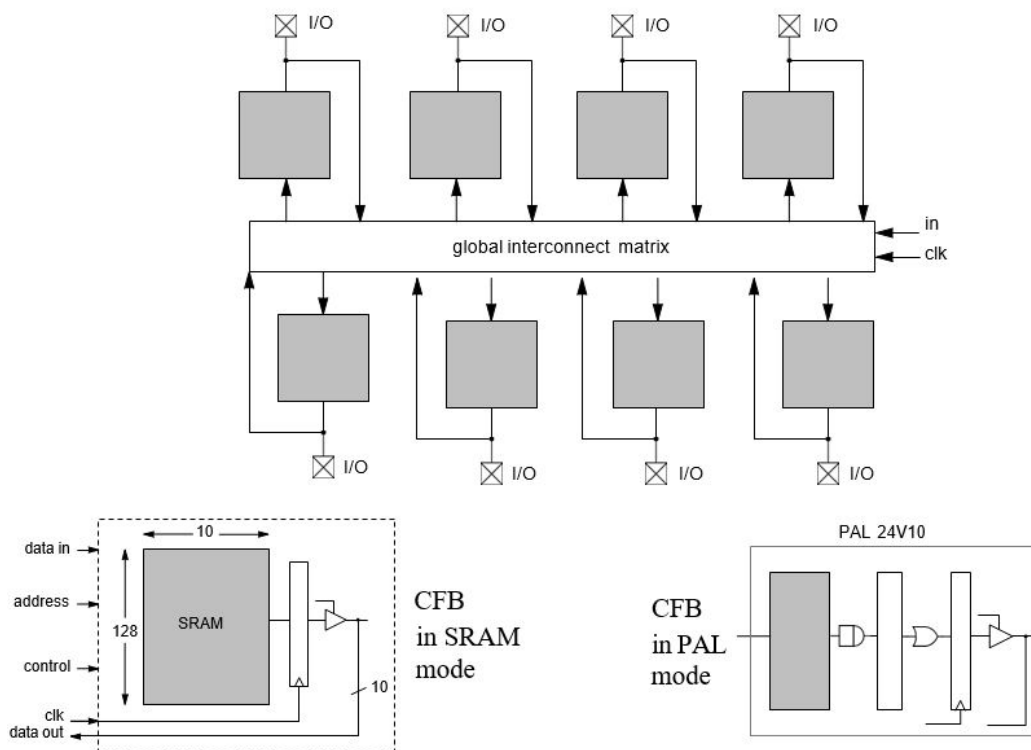


Figure 15 - Altera FLASHlogic CPLD.

In terms of basic structure, FLASHlogic is similar to other products already discussed. However, they have one unique feature that stands them apart from all other CPLDs: each PAL-like block, instead of being used for AND-OR logic, can be configured as a block of 10 nsec Static RAM. This concept is illustrated in the lower part of Figure 15, which shows one CFB being used as a PAL and another configured as an SRAM. In the SRAM configuration, the PAL block becomes a 128 word by 10 bit read/write memory. Inputs that would normally feed the AND-plane in the PAL in this case become address lines, data in, and control signals for the memory. Notice that the flip-flops and tri-state buffers are still available when the PAL block is configured as memory.

In the FLASHlogic device, the AND-OR logic plane's configuration bits are SRAM cells that are "shadowed" by EPROM or EEPROM cells. The SRAM cells are loaded with a copy of the non-volatile EPROM or EEPROM memory when power is applied, but it is the SRAM cells that control the configuration of the chip. It is possible to re-configure the chips in-system by downloading new information into the SRAM cells. The SRAM cells' contents can be written back to the EEPROM, so that non-volatile re-programming (in-circuit) is available.

ICT PEEL Arrays

The ICT PEEL Arrays are basically large PLAs that include logic macrocells with flop-flops and feedback to the logic planes. This structure is illustrated by Figure 16, which shows a programmable AND-plane that feeds a programmable OR-plane. The outputs of the OR-plane are divided into groups of four, and each group can be input to any of the logic cells. The logic cells provide registers for the sum terms and can feed-back the sum terms to the AND-plane. Also, the logic cells connect sum terms to I/O pins.

Because they have a PLA-like structure, logic capacity of PEEL Arrays is somewhat difficult to measure compared to the CPLDs discussed so far; an estimate is 1600 to 2800 equivalent gates.

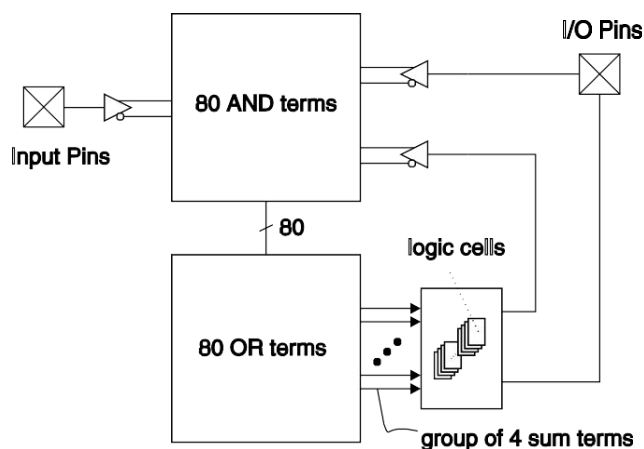


Figure 16 - Architecture of ICT PEEL Arrays.

PEEL Arrays offer relatively few I/O pins, with the largest part being offered in a 40 pin package. Since they do not comprise SPLD-like blocks, PEEL Arrays do not fit well into the CPLD category, however they are included here because they represent an example of PLA-based, rather than PAL-based devices, and they offer larger capacity than a typical SPLD.

The logic cell in the PEEL Arrays, depicted in Figure 17, includes a flip-flop, configurable as D, T, or JK, and two multiplexers. The multiplexers each produce an output of the logic cell and can provide either a registered or combinational output. One of the logic cell outputs can connect to an I/O pin and the other output is buried. One of the interesting features of the logic cell is that the flip-flop clock, as well as preset and clear, are full sum-of-product logic functions. This differs from all other CPLDs, which simply provide product terms for these signals and is attractive for some applications. Because of their PLA-like OR-plane, the ICT PEEL Arrays are especially well-suited to applications that require very wide sum terms.

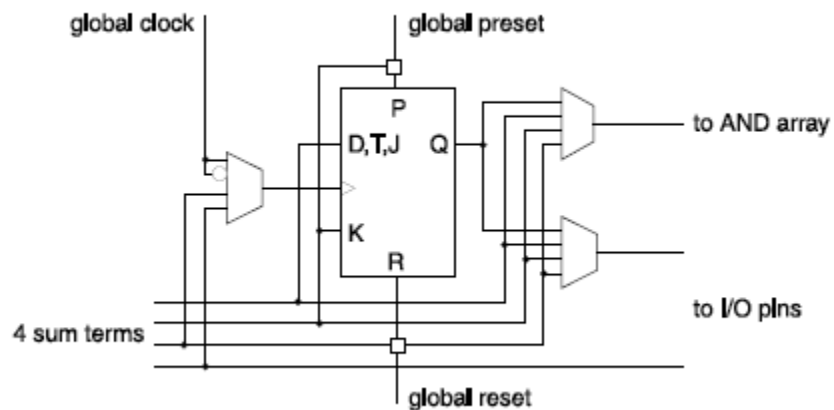


Figure 17 - Structure of ICT PEEL Array Logic Cell.

Applications of CPLDs

We will now briefly examine the types of applications which best suit CPLD architectures. Because they offer high speeds and a range of capacities, CPLDs are useful for a very wide assortment of applications, from implementing random glue logic to prototyping small gate arrays. One of the most common uses in industry at this time, and a strong reason for the large growth of the CPLD market, is the conversion of designs that consist of multiple SPLDs into a smaller number of CPLDs.

CPLDs can realize reasonably complex designs, such as graphics controller, LAN controllers, UARTs, cache control, and many others. As a general rule-of-thumb, circuits that can exploit wide AND/OR gates, and do not need a very large number of flip-flops are good candidates for

implementation in CPLDs. A significant advantage of CPLDs is that they provide simple design changes through re-programming (all commercial CPLD products are re-programmable). With in-system programmable CPLDs it is even possible to re-configure hardware (an example might be to change a protocol for a communications circuit) without power-down.

Designs often partition naturally into the SPLD-like blocks in a CPLD. The result is more predictable speed-performance than would be the case if a design were split into many small pieces and then those pieces were mapped into different areas of the chip. Predictability of circuit implementation is one of the strongest advantages of CPLD architectures.

Commercially Available FPGAs

As one of the largest growing segments of the semiconductor industry, the FPGA market-place is volatile. As such, the pool of companies involved changes rapidly and it is somewhat difficult to say which products will be the most significant when the industry reaches a stable state. For this reason, and to provide a more focused discussion, we will not mention all of the FPGA manufacturers that currently exist, but will instead focus on those companies whose products are in wide-spread use at this time. In describing each device we will list its capacity, nominally in 2-input NAND gates as given by the vendor. Gate count is an especially contentious issue in the FPGA industry, and so the numbers given in this paper for all manufacturers should not be taken too seriously. Wags have taken to calling them “dog” gates, in reference to the traditional ratio between human and dog years.

There are two basic categories of FPGAs on the market today: 1. SRAM-based FPGAs and 2. antifuse-based FPGAs. In the first category, Xilinx and Altera are the leading manufacturers in terms of number of users, with the major competitor being AT&T. For antifuse-based products, Actel, Quicklogic and Cypress, and Xilinx offer competing products.

Xilinx SRAM-based FPGAs

The basic structure of Xilinx FPGAs is *array-based*, meaning that each chip comprises a two-dimensional array of logic blocks that can be interconnected via horizontal and vertical routing channels. An illustration of this type of architecture was shown in Figure 2. Xilinx introduced the first FPGA family, called the XC2000 series, in about 1985 and now offers three more

generations: XC3000, XC4000, and XC5000. Although the XC3000 devices are still widely used, we will focus on the more recent and more popular XC4000 family. We note that XC5000 is similar to XC4000, but has been engineered to offer similar features at a more attractive price, with some penalty in speed. We should also note that Xilinx has recently introduced an FPGA family based on anti-fuses, called the XC8100. The XC8100 has many interesting features, but since it is not yet in widespread use, we will not discuss it here. The Xilinx 4000 family devices range in capacity from about 2000 to more than 15,000 equivalent gates.

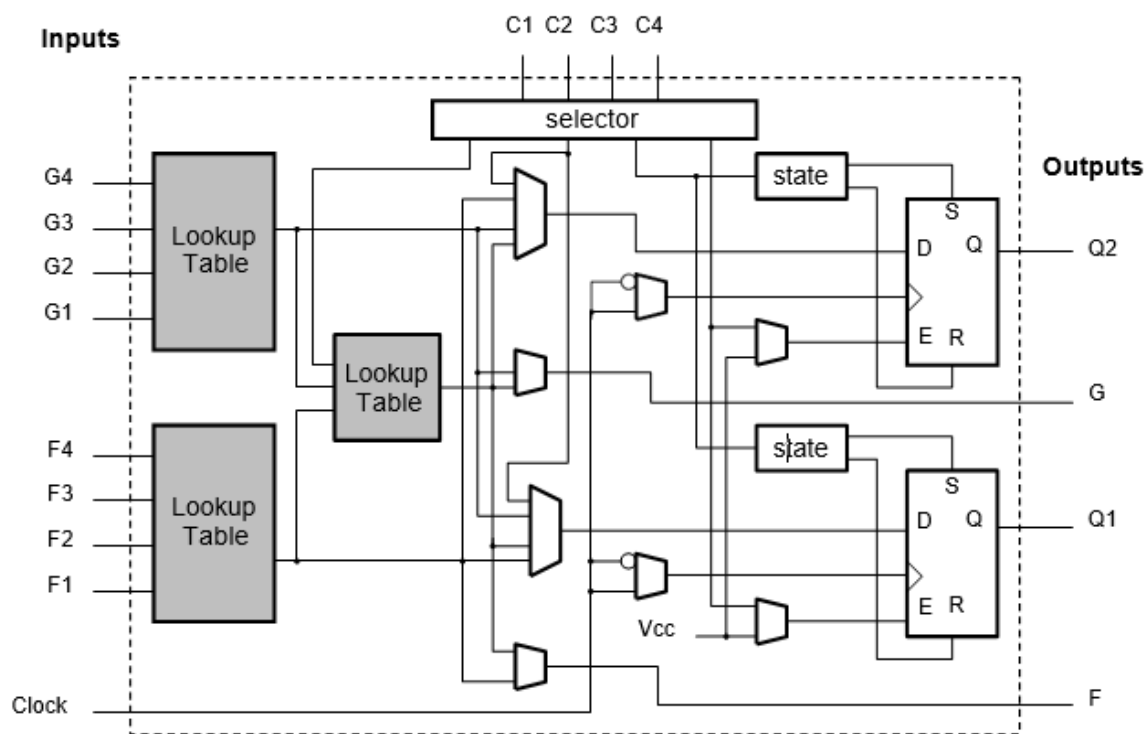


Figure 18 - Xilinx XC4000 Configurable Logic Block (CLB).

The XC4000 features a logic block (called a *Configurable Logic Block (CLB)* by Xilinx) that is based on look-up tables (LUTs). A LUT is a small one bit wide memory array, where the address lines for the memory are inputs of the logic block and the one bit output from the memory is the LUT output. A LUT with K inputs would then correspond to a $2^K \times 1$ bit memory, and can realize any logic function of its K inputs by programming the logic function's truth table directly into the memory. The XC4000 CLB contains three separate LUTs, in the configuration shown in Figure 18. There are two 4-input LUTs that are fed by CLB inputs, and the third LUT can be used

in combination with the other two. This arrangement allows the CLB to implement a wide range of logic functions of up to nine inputs, two separate functions of four inputs or other possibilities. Each CLB also contains two flip-flops.

Toward the goal of providing high density devices that support the integration of entire systems, the XC4000 chips have “system oriented” features. For instance, each CLB contains circuitry that allows it to efficiently perform arithmetic (i.e., a circuit that can implement a fast carry operation for adder-like circuits) and also the LUTs in a CLB can be configured as read/write RAM cells. A new version of this family, the 4000E, has the additional feature that the RAM can be configured as a dual port RAM with a single write and two read ports. In the 4000E, RAM blocks can be synchronous RAM. Also, each XC4000 chip includes very wide AND-planes around the periphery of the logic block array to facilitate implementing circuit blocks such as wide decoders.

Besides logic, the other key feature that characterizes an FPGA is its interconnect structure. The XC4000 interconnect is arranged in horizontal and vertical channels. Each channel contains some number of short wire segments that span a single CLB (the number of segments in each channel depends on the specific part number), longer segments that span two CLBs, and very long segments that span the entire length or width of the chip. Programmable switches are available (see Figure 5) to connect the inputs and outputs of the CLBs to the wire segments, or to connect one wire segment to another. A small section of a routing channel representative of an XC4000 device appears in Figure 19. The figure shows only the wire segments in a horizontal channel, and does not show the vertical routing channels, the CLB inputs and outputs, or the routing switches. An important point worth noting about the Xilinx interconnect is that signals must pass through switches to reach one CLB from another, and the total number of switches traversed depends on the particular set of wire segments used. Thus, speed-performance of an implemented circuit depends in part on how the wire segments are allocated to individual signals by CAD tools.

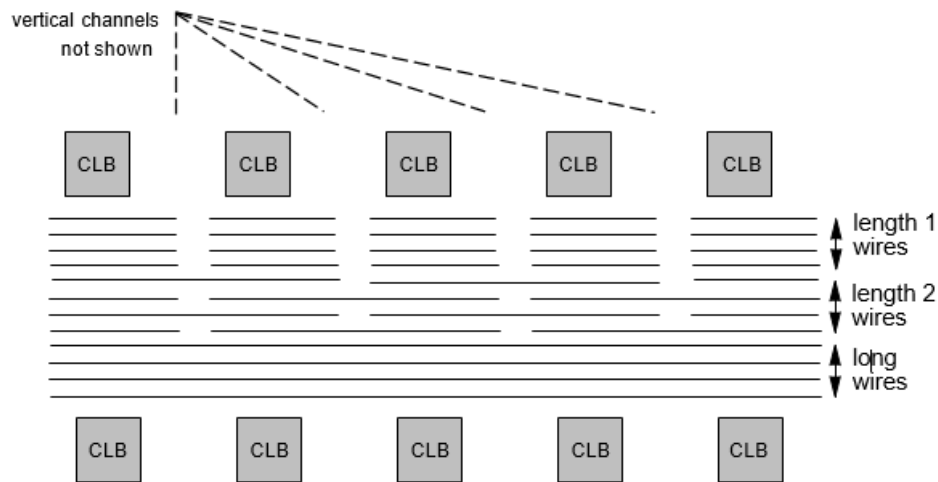


Figure 19 - Xilinx XC4000 Wire Segments.

Altera FLEX 8000 and FLEX 10000 FPGAs

Altera's FLEX 8000 series consists of a three-level hierarchy much like that found in CPLDs. However, the lowest level of the hierarchy consists of a set of lookup tables, rather than an SPLD-like block, and so the FLEX 8000 is categorized here as an FPGA. It should be noted, however, that FLEX 8000 is a combination of FPGA and CPLD technologies. FLEX 8000 is SRAM-based and features a four-input LUT as its basic logic block. Logic capacity ranges from about 4000 gates to more than 15,000 for the 8000 series.

The overall architecture of FLEX 8000 is illustrated in Figure 20. The basic logic block, called a Logic Element (LE) contains a four-input LUT, a flip-flop, and special-purpose carry circuitry for arithmetic circuits (similar to Xilinx XC4000). The LE also includes cascade circuitry that allows for efficient implementation of wide AND functions. Details of the LE are illustrated in Figure 21.

In the FLEX 8000, LEs are grouped into sets of 8, called Logic Array Blocks (LABs, a term borrowed from Altera's CPLDs). As shown in Figure 22, each LAB contains local interconnect and each local wire can connect any LE to any other LE within the same LAB. Local interconnect

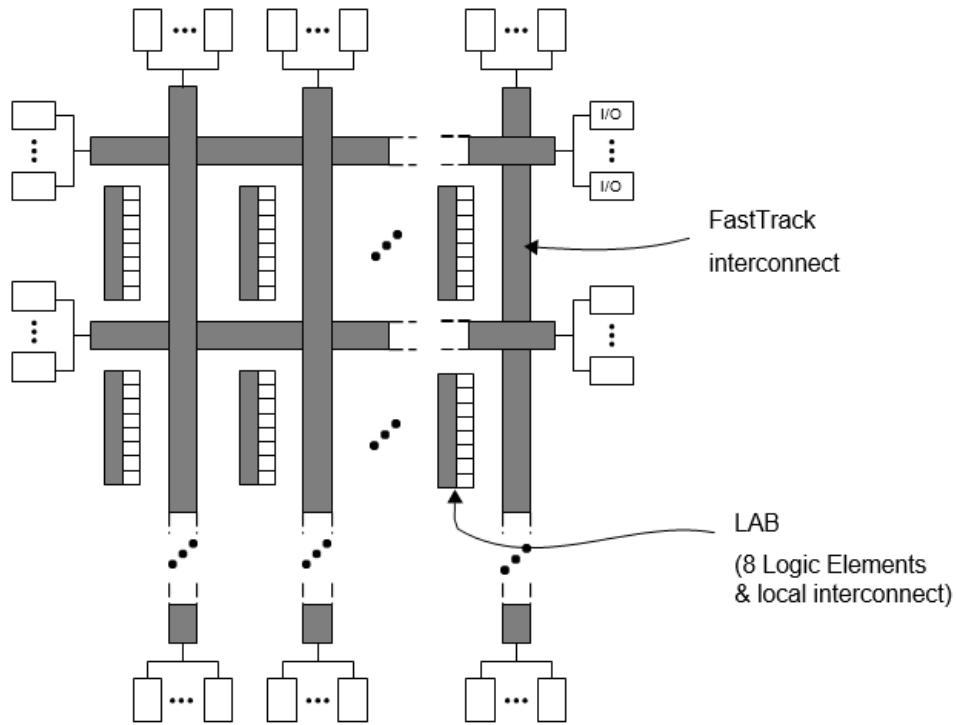


Figure 20 - Architecture of Altera FLEX 8000 FPGAs.

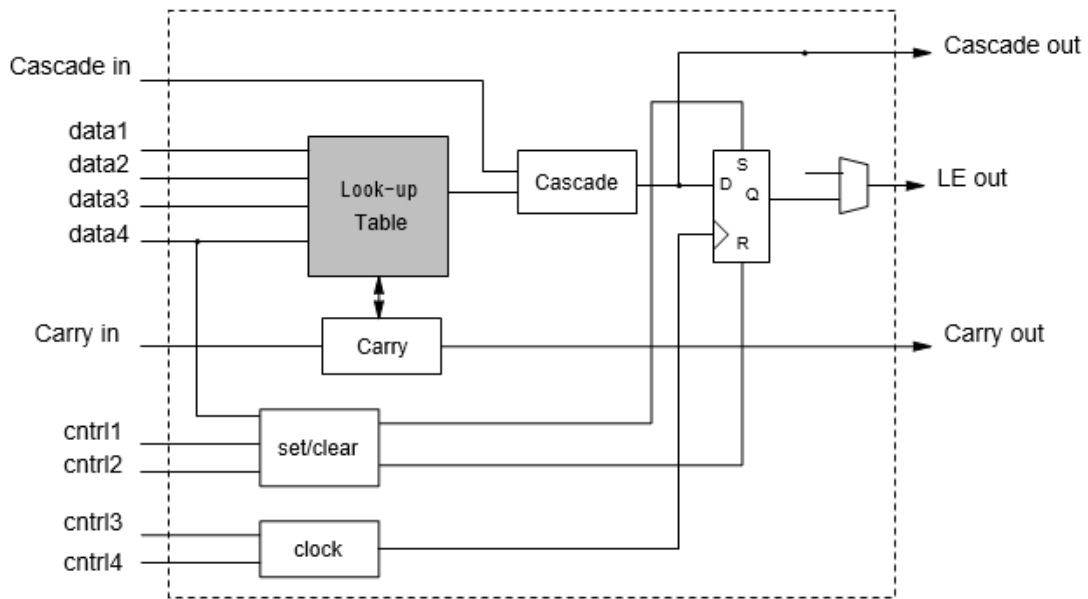


Figure 21 - Altera FLEX 8000 Logic Element (LE).

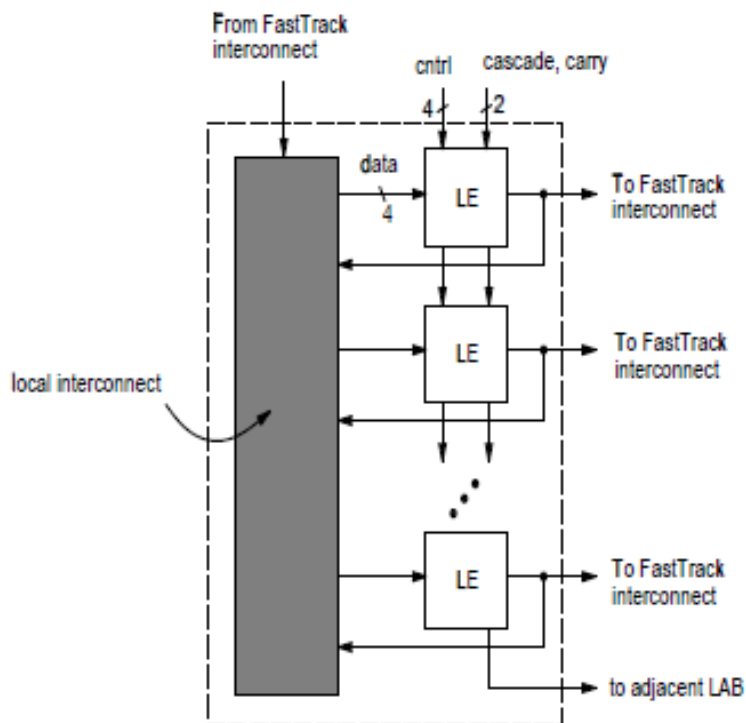


Figure 22 - Altera FLEX 8000 Logic Array Block (LAB).

also connects to the FLEX 8000's *global interconnect*, called FastTrack. FastTrack is similar to Xilinx long lines in that each FastTrack wire extends the full width or height of the device. However, a major difference between FLEX 8000 and Xilinx chips is that FastTrack consists of *only* long lines. This makes the FLEX 8000 easy for CAD tools to automatically configure. All FastTrack wires horizontal wires are identical, and so interconnect delays in the FLEX 8000 are more predictable than FPGAs that employ many smaller length segments because there are fewer programmable switches in the longer paths. Predictability is furthered aided by the fact that connections between horizontal and vertical lines pass through active buffers.

The FLEX 8000 architecture has been extended in the state-of-the-art FLEX 10000 family. FLEX 10000 offers all of the features of FLEX 8000, with the addition of variable-sized blocks of SRAM, called Embedded Array Blocks (EABs). This idea is illustrated in Figure 23, which shows that each row in a FLEX 10000 chip has an EAB on one end. Each EAB is configurable to serve as an SRAM block with a variable aspect ratio: 256 x 8, 512 x 4, 1K x 2, or 2K x 1. In addition, an EAB can alternatively be configured to implement a complex logic circuit, such as a multiplier, by

employing it as a large multi-output lookup table. Altera provides, as part of their CAD tools, several macro-functions that implement useful logic circuits in EABs. Counting the EABs as logic gates, FLEX 10000 offers the highest logic capacity of any FPGA, although it is hard to provide an accurate number.

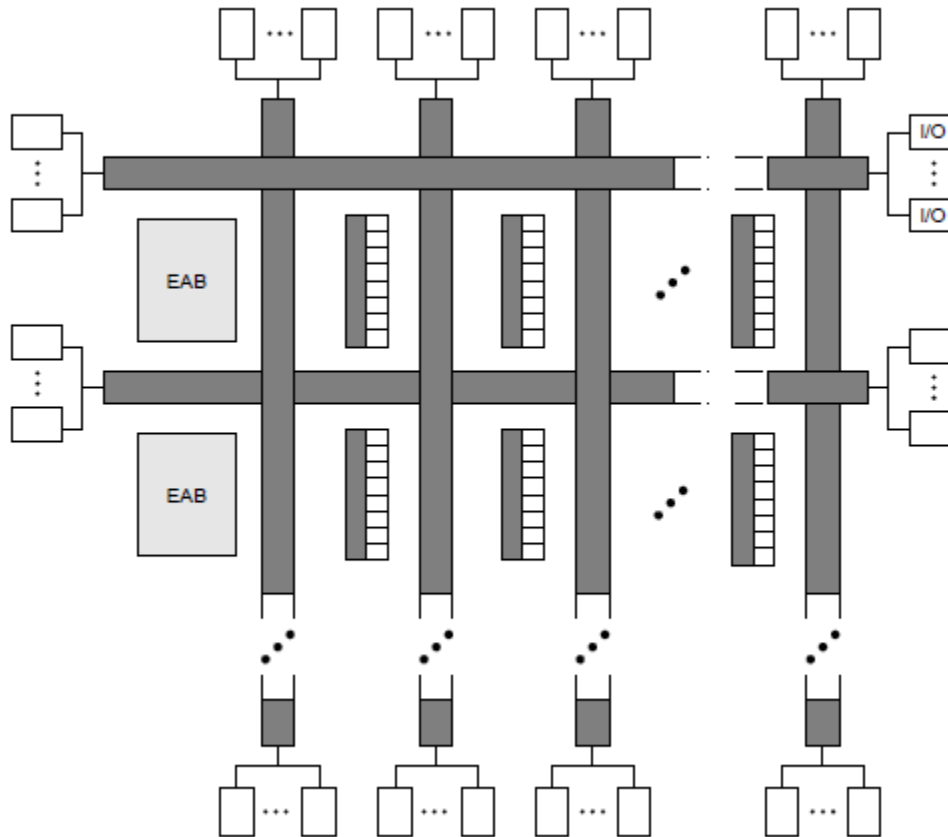


Figure 23 - Architecture of Altera FLEX 10K FPGAs.

AT&TORCA FPGAs

AT&T's SRAM-based FPGAs feature an overall structure similar to that in Xilinx FPGAs, and is called Optimized Reconfigurable Cell Array (ORCA). The ORCA logic block is based on LUTs, containing an array of *Programmable Function Units* (PFUs). The structure of a PFU is shown in Figure 24. A PFU possesses a unique kind of configurability among LUT-based logic blocks, in that it can be configured in the following ways: as four 4-input LUTs, as two 5-input LUTs, and as one 6-input LUT. A key element of this architecture is that when used as four 4- input LUTs, several of the LUTs' inputs must come from the same PFU input. While this reduces the

apparent functionality of the PFU, it also significantly reduces the cost of the wiring associated with the chip. The PFU also includes arithmetic circuitry, like Xilinx XC4000 and Altera FLEX 8000, and like Xilinx XC4000 a PFU can be configured as a RAM block. A recently announced version of the ORCA chip also allows dual-port and synchronous RAM.

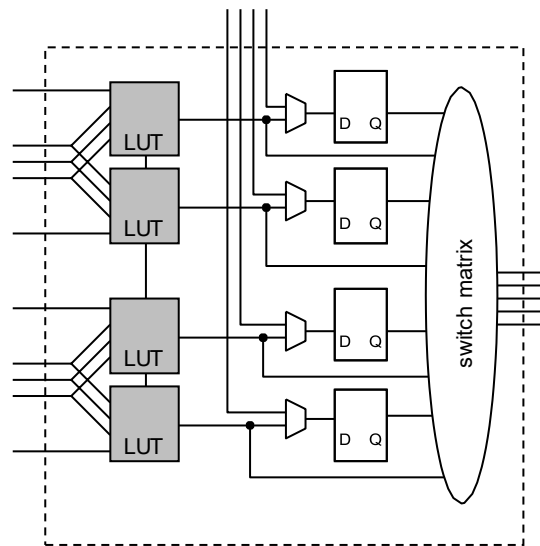


Figure 24 - AT&T Programmable Function Unit (PFU).

ORCA's interconnect structure is also different from those in other SRAM-based FPGAs. Each PFU connects to interconnect that is configured in four-bit buses. This provides for more efficient support for "system-level" designs, since buses are common in such applications. The ORCA family has been extended in the ORCA 2 series, and offers very high logic capacity up to 40,000 logic gates. ORCA 2 features a two-level hierarchy of PFUs based on the original ORCA architecture.

Actel FPGAs

In contrast to FPGAs described above, the devices manufactured by Actel are based on anti-fuse technology. Actel offers three main families: Act 1, Act 2, and Act 3. Although all three generations have similar features, this paper will focus on the most recent devices, since they are apt to be more widely used in the longer term. Unlike the FPGAs described above, Actel devices are based on a structure similar to traditional gate arrays; the logic blocks are arranged in rows and there are horizontal routing channels between adjacent rows. This architecture is illustrated in Figure 25. The logic blocks in the Actel devices are relatively small in comparison to the LUT- based ones described above, and are based on multiplexers. Figure 26 illustrates the logic block in the Act 3 and shows that it comprises an AND and OR gate that are connected to a multiplexer- based circuit block. The multiplexer circuit is arranged such that, in combination with the two logic gates, a very wide range of functions can be realized in a single logic block. About half of the logic blocks in an Act 3 device also contain a flip-flop.

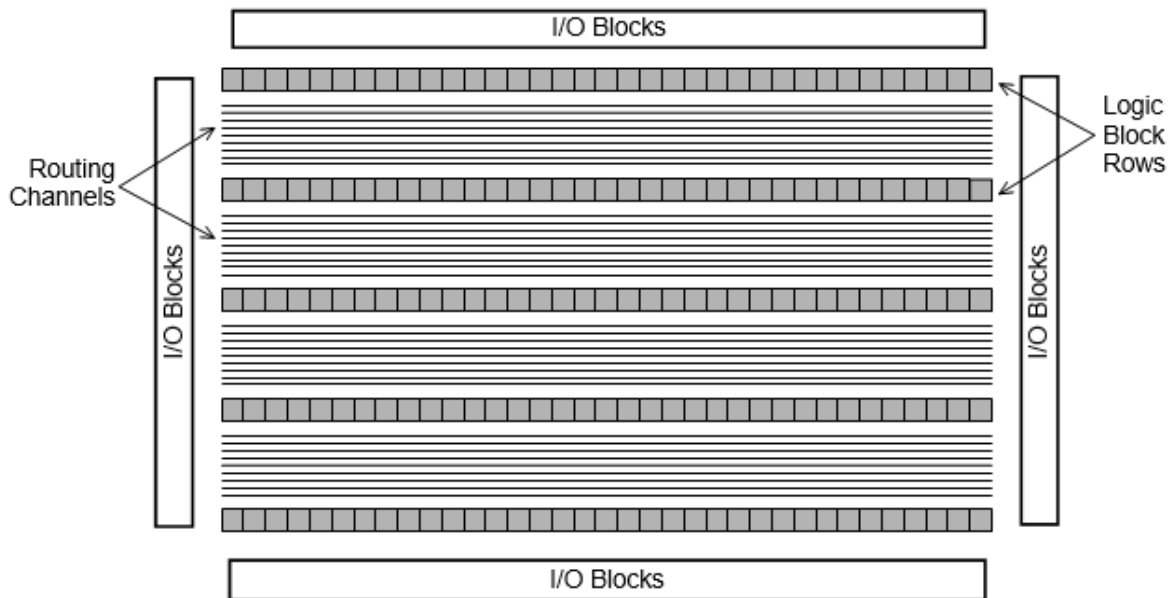


Figure 25 - Structure of Actel FPGAs.

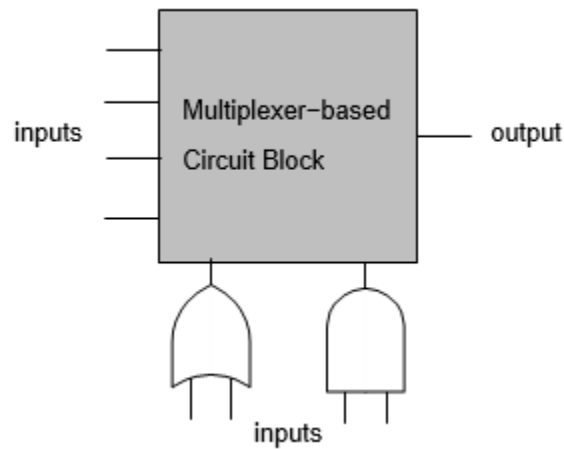


Figure 26 - Actel Act 3 Logic Module.

As stated above, Actel's interconnect is organized in horizontal routing channels. The channels consist of wire segments of various lengths with antifuses to connect logic blocks to wire segments or one wire to another. Also, although not shown in Figure 25, Actel chips have vertical wires that overlay the logic blocks, for signal paths that span multiple rows. In terms of speed-performance, it would seem probable that Actel chips are not fully predictable, because the number of antifuses traversed by a signal depends on how the wire segments are allocated during circuit implementation by CAD tools. However, Actel provides a rich selection of wire segments of different length in each channel and has developed algorithms that guarantee strict limits on the number of antifuses traversed by any two-point connection in a circuit which improves speed-performance significantly.

Quicklogic pASIC FPGAs

The main competitor for Actel in antifuse-based FPGAs is Quicklogic, whose has two families of devices, called pASIC and pASIC-2. The pASIC-2 is an enhanced version that has only recently been introduced, and will not be discussed here. The pASIC, as illustrated in Figure 27, has similarities to several other FPGAs: the overall structure is array-based like Xilinx FPGAs, its logic blocks use multiplexers similar to Actel FPGAs, and the interconnect consists of only long

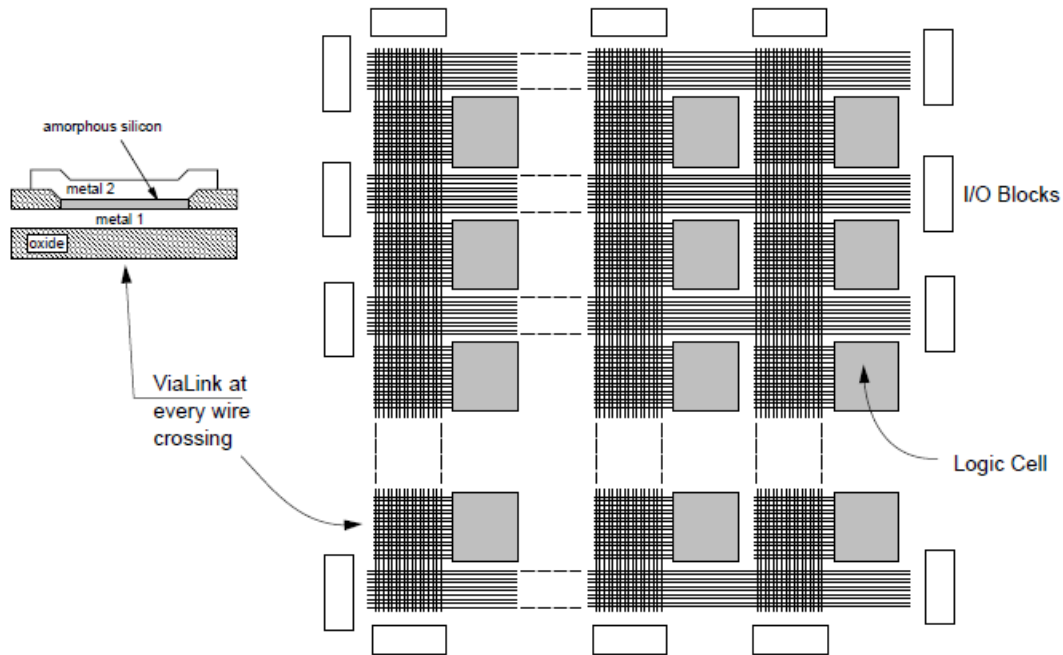


Figure 27 - Structure of Quicklogic pASIC FPGA.

Quicklogic's antifuse structure, called ViaLink, is illustrated on the left-hand side of Figure 27. It consists of a top layer of metal, an insulating layer of amorphous silicon, and a bottom layer of metal. When compared to Actel's PLICE antifuse, ViaLink offers a very low on-resistance of about 50 ohms (PLICE is about 300 ohms) and a low parasitic capacitance. Figure 27 shows that ViaLink antifuses are present at every crossing of logic block pins and interconnect wires, providing generous connectivity. pASIC's multiplexer-based logic block is depicted in Fig 28. It is more complex than Actel's Logic Module, with more inputs and wide (6-input) AND-gates on the multiplexer select lines. Every logic block also contains a flip-flops.

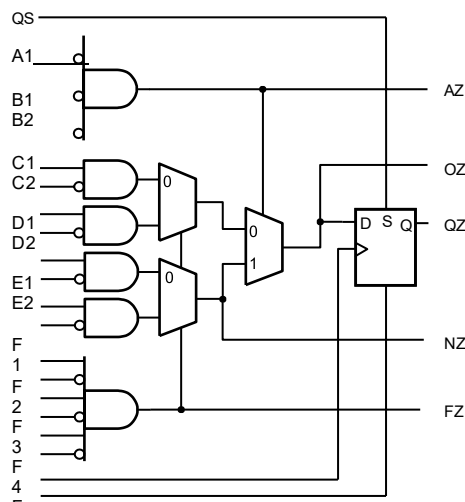


Figure 28 - Quicklogic (Cypress) Logic Cell.

Applications of FPGAs

FPGAs have gained rapid acceptance and growth over the past decade because they can be applied to a very wide range of applications. A list of typical applications includes: random logic, integrating multiple SPLDs, device controllers, communication encoding and filtering, small to medium sized systems with SRAM blocks, and many more.

Other interesting applications of FPGAs are prototyping of designs later to be implemented in gate arrays, and also emulation of entire large hardware systems. The former of these applications might be possible using only a single large FPGA (which corresponds to a small Gate Array in terms of capacity), and the latter would entail many FPGAs connected by some sort of interconnect; for emulation of hardware, Quick Turn [Wolff90] (and others) has developed products that comprise many FPGAs and the necessary software to partition and map circuits.

Another promising area for FPGA application, which is only beginning to be developed, is the usage of FPGAs as custom computing machines. This involves using the programmable parts to “execute” software, rather than compiling the software for execution on a regular CPU.

POST MCQ:

1. Which clock is preferred in storage devices?
 - a) single phase overlapping clock signal
 - b) single phase non overlapping clock signal
 - c) two phase overlapping clock signal
 - d) two phase non overlapping clock signal**
2. Reading a cell is a _____ operation.
 - a) constructive
 - b) destructive**
 - c) semi constructive
 - d) semi destructive
3. Which method is used to determine structural defects?
 - a) deterministic test pattern**
 - b) algorithmic test pattern
 - c) random test pattern
 - d) exhaustive test pattern
4. Which method is used for external functional testing?
 - a) exhaustive test pattern method
 - b) pseudo-exhaustive test pattern method

c) random test pattern method

d) pseudo-random test pattern method

5. The electrical behaviour of a circuit is given using
- a) design rules
 - b) floor plan
 - c) structures and layouts

d) mathematical modelling

6. Which among the following is a process of transforming design entry information of the circuit into a set of logic equations?
- a. Simulation
 - b. Optimization
 - c. Synthesis**
 - d. Verification

7. Which among the following is an output generated by synthesis process?
- a. Attributes & Library
 - b. RTL VHDL description
 - c. Circuit constraints

d. Gate-level net list

8. In fusible link technologies, the undesired fuses are removed by the pulse application of _____ voltage & current to device input.
- a. Low
 - b. Moderate
 - c. High**
 - d. All of the above
9. An Antifuse programming technology is predominantly associated with _____.
- a. SPLDs
 - b. FPGAs**
 - c. CPLDs
 - d. All of the above

TEXT 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.

REFERENCE BOOKS

1. VLSI Design Principles- John P. Uyemura, John Wiley,2002
2. E. Fabricious , Introduction to VLSI design, McGraw-Hill 1990
3. Wayne Wolf, Modern VLSI Design, Pearson Education 2003A. Anand Kumar, "Switching Theory and Logic Design" – PHI, 2nd Edition

