VLSI ENGINEERING (3-1-0)

Module-I (10 Hours)
Issues and Challenges in VLSI Design, VLSI Design Methodology, VLSI Design Flow, VLSI Design Hierarchy, VLSI Design Styles, CAD Technology. VLSI Fabrication Technology: Basic Steps of Fabrication, CMOS p-Well and n-Well Processes, Layout Design, Design Rules, Stick Diagram, Bi-CMOS Fabrication Process.

Module-II (10 Hours)
CMOS Inverter: MOS Device Model with Sub-micron Effects, VTC Parameters (DC Characteristics), CMOS Propagation Delay, Parasitic Capacitance Estimation, Layout of an Inverter, Switching, Short-Circuit and Leakage Components of Energy and Power; Interconnects: Resistance, Capacitance Estimation, delays, Buffer Chains, Low Swing Drivers, Power Dissipation, and Performance Optimization of Digital Circuits by Logical Effort and Transistor Sizing.

Module-III (10 Hours)
Combinational Logic Circuits: Static CMOS Logic Circuits: Complementary CMOS, Ratioed Logic, Pass Transistor Logic, Transmission Gate Logic, DCVS Logic, Dynamic CMOS Logic Circuits; Sequential Logic Circuits: Static Latches and Registers, Dynamic Latches and Registers, Pulse Based Registers; Sense Amplifier Based Registers, Semiconductor Memories: Non-Volatile and Volatile Memory Devices, Flash Memories, SRAM, DRAM.

Module-IV (10 Hours)
Design Capture Tools, Hardware Description Language: VHDL, Testing and Verification: Defects, Fault Models, Design Strategies for Testing, Chip Level and System Level Test Techniques, Packaging Technology.

Text Books:

1. J.M. Rabaey, A. Chandrakasan and B. Nikolic, Digital Integrated Circuits- A Design Perspective, 2/e, Prentice Hall of India, 2003.

2. N. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 3/e, Pearson Education India, 2007
3. Kang and Leblebici, CMOS Digital Integrated Circuits Analysis and Design, 3/e, McGraw Hill, 2003.

Reference Books:

1. D. A. Hodges, H. G. Jackson, R. Saleh, Analysis and Design of Digital Integrated Circuits in Deep submicron Technology, 3/e, McGraw Hill, 2004.
2. Douglas A.Pucknell, Kamran Eshiraghian, Basic VLSI Design, 3/e P H I, 1993.
3. J. P. Uyemura, Introduction to VLSI Circuits and Systems, John Wiley & Sons (Asia), 2002
4. W. Wolf, Modern VLSI Design - System on Chip design, 3/e, Pearson Education, 2004.
5. VHDL Programming by example- Perry, T M H

VLSI ENGINEERING (3-1-0)

MODULE 1:

Issues and Challenges in VLSI Design:

Integration density and performance of integrated circuits have gone through an astounding revolution in the last couple of decades. In the 1960s, Gordon Moore, then with Fairchild Corporation and later cofounder of Intel, predicted that the number of transistors that can be integrated on a single die would grow exponentially with time. This prediction, later called *Moore's law*, has proven to be amazingly visionary. Its validity is best illustrated with the aid of a set of graphs. Figure 1 plots the integration density of both logic IC's and memory as a function of time. As can be observed, integration complexity doubles approximately every 1 to 2 years. As a result, memory density has increased by more than a thousand fold since 1970.

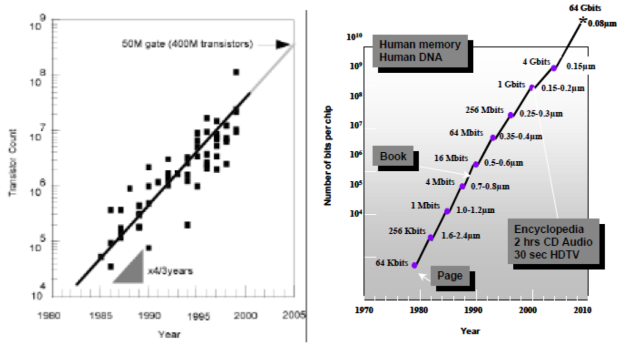


Figure 1: Evolution of integration complexity of logic ICs and memories as a function of time.

Typically used abstraction levels in digital circuit design are, in order of increasing abstraction, the device, circuit, gate, functional module (e.g., adder) and system levels (e.g., processor), as illustrated in Figure 2. A semiconductor device is an entity with a very complex behavior.

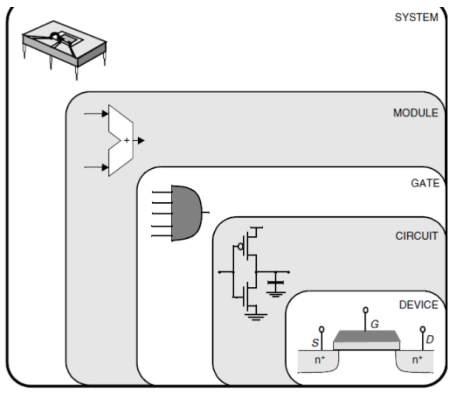
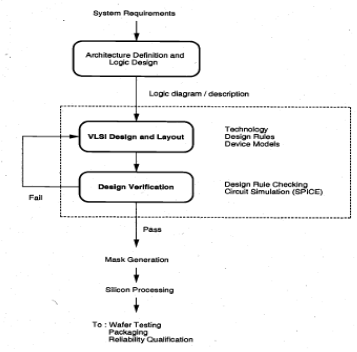


Figure 2 Design abstraction levels in digital circuits.

VLSI Design Flow:



VLSI Fabrication Technology:

The CMOS process requires a large number of steps, each of which consists of a sequence of basic operations. A number of these steps and/or operations are executed very repetitively in the course of the manufacturing process.

The Silicon Wafer

The base material for the manufacturing process comes in the form of a single-crystalline, lightly doped *wafer*. These wafers have typical diameters between 4 and 12 inches (10 and 30 cm, respectively) and a thickness of at most 1 mm, and are obtained by cutting a single crystal ingot into thin slices.

Photolithography:

In each processing step, a certain area on the chip is masked out using the appropriate optical mask so that a desired processing step can be selectively applied to the remaining regions. The processing step can be any of a wide range of tasks including oxidation, etching, metal and polysilicon deposition, and ion implantation. The technique to accomplish this selective masking, called *photolithography*, is applied throughout the manufacturing process. Figure 3 gives a graphical overview of the different operations involved in a typical photolithographic process. The following steps can be identified:

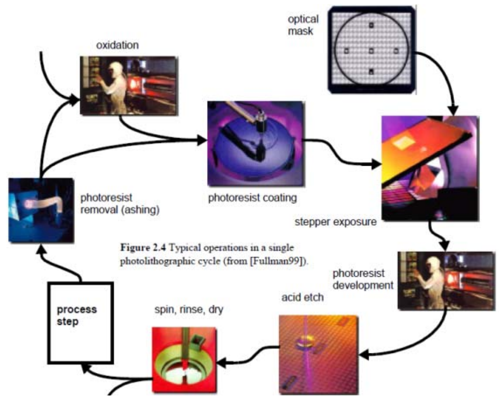


Figure 3: Graphical Overview

1. Oxidation layering — this optional step deposits a thin layer of SiO2 over the complete wafer by exposing it to a mixture of high-purity oxygen and hydrogen at approximately 1000°C. The oxide is used as an insulation layer and also forms transistor gates.
2. Photoresist coating — a light-sensitive polymer (similar to latex) is evenly applied while spinning the wafer to a thickness of approximately 1 mm. This material is originally soluble in an organic solvent, but has the property that the polymers crossoxidation link when exposed to light, making the affected regions insoluble. A photoresist of this type is called negative. A positive photoresist has the opposite properties;

originally insoluble, but soluble after exposure. By using both positive and negative resists, a single mask can sometimes be used for two steps, making complementary regions available for processing. Since the cost of a mask is increasing quite rapidly with the scaling of technology, a reduction of the number of masks is surely of high priority.

3. Stepper exposure — a glass mask (or reticle), containing the patterns that we want to transfer to the silicon, is brought in close proximity to the wafer. The mask is opaque in the regions that we want to process, and transparent in the others (assuming a negative photo resist). The glass mask can be thought of as the negative of one layer of the microcircuit. The combination of mask and wafer is now exposed to ultra-violet light. Where the mask is transparent, the photo resist becomes insoluble.
4. Photoresist development and bake — the wafers are developed in either an acid or base solution to remove the non-exposed areas of photoresist. Once the exposed photoresist is removed, the wafer is "soft-baked" at a low temperature to harden the remaining photoresist.
5. Acid Etching — material is selectively removed from areas of the wafer that are not covered by photoresist. This is accomplished through the use of many different types of acid, base and caustic solutions as a function of the material that is to be removed. Much of the work with chemicals takes place at large wet benches where special solutions are prepared for specific tasks. Because of the dangerous nature of some of these solvents, safety and environmental impact is a primary concern.
6. Spin, rinse, and dry — a special tool (called SRD) cleans the wafer with deionized water and dries it with nitrogen. The microscopic scale of modern semiconductor devices means that even the smallest particle of dust or dirt can destroy the circuitry. To prevent this from happening, the processing steps are performed in ultra-clean rooms where the number of dust particles per cubic foot of air ranges between 1 and 10. Automatic wafer handling and robotics are used whenever possible. This explains why the cost of a state-of-the-art fabrication facility easily ranges in the multiple billions of dollars. Even then, the wafers must be constantly cleaned to avoid contamination, and to remove the left-over of the previous process steps.
7. Photoresist removal (or ashing) — a high-temperature plasma is used to selectively remove the remaining photoresist without damaging device layers.

<u>CMOS Process Flow:</u>

The process starts with the definition of the active regions; this is the regions where transistors will be constructed. All other areas of the die will be covered with a thick layer of silicon dioxide (SiO2), called the field oxide. This oxide acts as the insulator between neighboring devices, and is either grown (as in the process of Figure 2.1), or deposited in etched trenches (Figure 2.2) — hence the name trench insulation. Further insulation is provided by the addition of a reverse-biased np-diode, formed by adding an extra p+ region, called the channel-stop implant (or field implant) underneath the field oxide. Next, lightly doped p- and n-wells are formed through ion implantation.
To construct an NMOS transistor in a p-well, heavily doped n-type source and drain regions are implanted (or diffused) into the lightly doped p-type substrate. A thin layer of SiO2, called the gate oxide, separates the region between the source and drain, and is itself covered by conductive polycrystalline silicon (or polysilicon, for short).

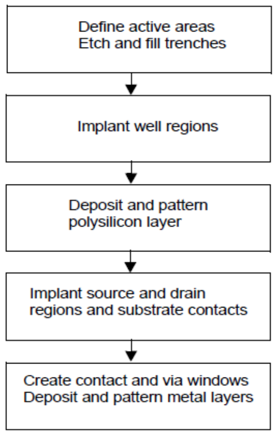
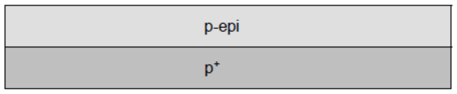
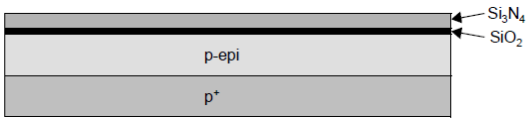


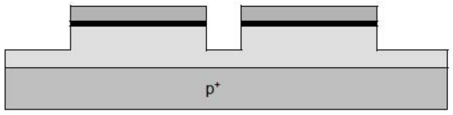
Figure 4: Simplified process sequence for the manufacturing of a ndual-well CMOS circuit.



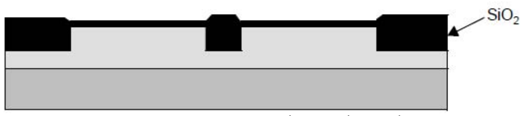
(a) Base material: p+ substrate with p-epi layer



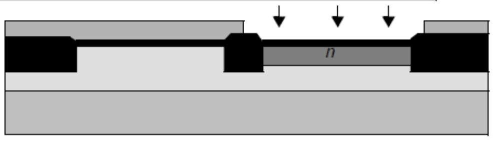
(b) After deposition of gate-oxide sacrificial nitride (acts as a buffer layer)



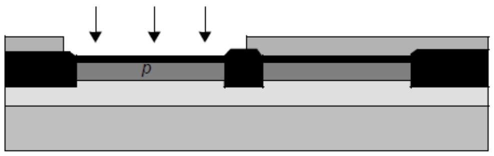
(c) After plasma etch of insulating trenches using the inverse of the active area mask



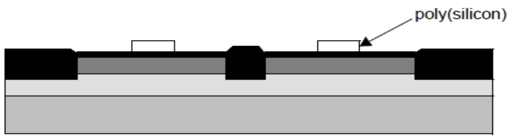
(d) After trench filling, CMP planarization, and removal of sacrificial nitride



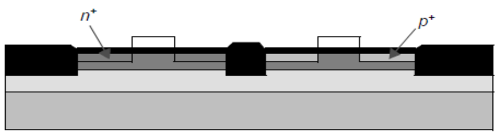
(e) After *n*-well and V_{Tp} adjust implants



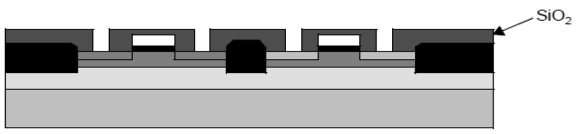
(f) After *p*-well and V_{Tn} adjust implants



(g) After polysilicon deposition and etch



(h) After n^+ source/drain and p^+ source/drain implants. These steps also dope the polysilicon.



(i) After deposition of SiO_2 insulator and contact hole etch.

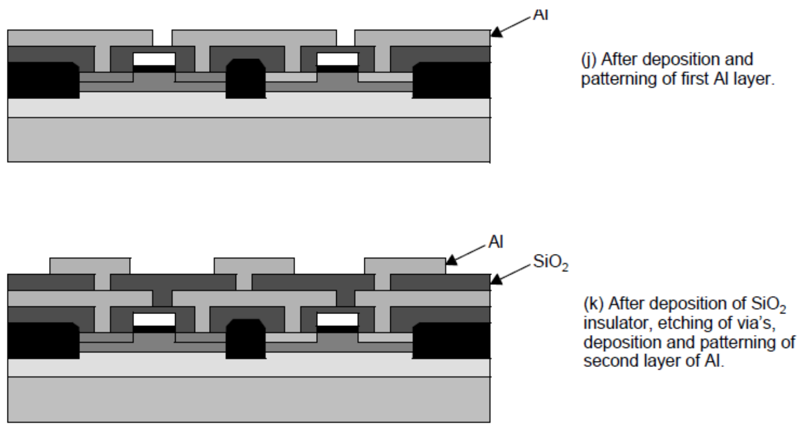

Al
(j) After deposition and patterning of first Al layer.
Al
SiO$_2$
(k) After deposition of SiO$_2$ insulator, etching of via's, deposition and patterning of second layer of Al.

Figure 5: Process flow for the fabrication of an NMOS and a PMOS transistor in a dual-well CMOS process.

Design Rules:

The goal of defining a set of design rules is to allow for a ready translation of a circuit concept into an actual geometry in silicon. The design rules act as the interface or even the contract between the circuit designer and the process engineer. Circuit designers in general want tighter, smaller designs, which lead to higher performance and higher circuit density. The process engineer, on the other hand, wants a reproducible and high-yield process. Design rules are, consequently, a compromise that attempts to satisfy both sides. The design rules provide a set of guidelines for constructing the various masks needed in the patterning process. They consist of minimum-width and minimum-spacing constraints and requirements between objects on the same or on different layers.

The fundamental unity in the definition of a set of design rules is the *minimum line width*. It stands for the minimum mask dimension that can be safely transferred to the semiconductor material. In general, the minimum line width is set by the resolution of the patterning process, which is most commonly based on optical lithography.

Mead and Conway [Mead80], defines all rules as a function of a single parameter, most often called **λ based rule**. The rules are chosen so that a design is easily ported over a cross section of industrial processes. Scaling of the minimum dimension is accomplished by simply changing the value of λ. This results in a *linear scaling* of all dimensions. For a given process, λ is set to a specific value, and all design dimensions are consequently translated into absolute numbers.
This approach, while attractive, suffers from some disadvantages

- Linear scaling is only possible over a limited range of dimensions (for instance, between 0.25 mm and 0.18 mm). When scaling over larger ranges, the relations between the different layers tend to vary in a nonlinear way that cannot be adequately covered by the linear scaling rules.

As circuit density is a prime goal in industrial designs, most semiconductor companies tend to use ***micron rules*****,** which express the design rules in absolute dimensions and can therefore exploit the features of a given process to a maximum degree. Scaling and porting designs between technologies under these rules is more demanding and has to be performed either manually or using advanced CAD tools.

Layout Design:

The layer concept translates the intractable set of masks currently used in CMOS into a simple set of conceptual layout levels that are easier to visualize by the circuit designer. From a designer's viewpoint, all CMOS designs are based on the following entities:

1. Substrates and/or wells, being p-type (for NMOS devices) and n-type (for PMOS)
2. Diffusion regions (n+ and p+) defining the areas where transistors can be formed.These regions are often called the active areas. Diffusions of an inverse type are needed to implement contacts to the wells or to the substrate. These are called select regions.

3. One or more polysilicon layers, which are used to form the gate electrodes of the transistors (but serve as interconnect layers as well).
4. A number of metal interconnect layers.
5. Contact and via layers to provide interlayer connections.

A layout consists of a combination of polygons, each of which is attached to a certain layer. The functionality of the circuit is determined by the choice of the layers, as well as the interplay between objects on different layers. For instance, an MOS transistor is formed by the cross section of the diffusion layer and the polysilicon layer. An interconnection between two metal layers is formed by a cross section between the two metal layers and an additional contact layer. To visualize these relations, each layer is assigned a standard color (or stipple pattern for a black-and-white representation).

Layout Exmple:

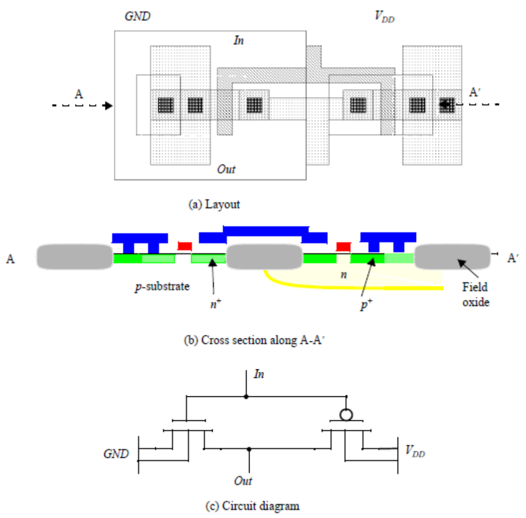


Figure 6 A detailed layout examples, including vertical process cross section and circuit diagram.

Module-II

CMOS 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.

The Static CMOS Inverter — An Intuitive Perspective

Figure 7 shows the circuit diagram of a static CMOS inverter. Its operation is readily understood with the aid of the simple switch model of the MOS transistor, the transistor is nothing more than a switch with an infinite off resistance (for $|VGS| < |VT|$), and a finite on-resistance (for $|VGS| > |VT|$). This leads to the following interpretation of the inverter. When V_{in} is high and equal to V_{DD}, the NMOS transistor is on, while the PMOS is off. This yields the equivalent circuit of Figure 8. A direct path exists between V_{out} and the ground node, resulting in a steady-state value of 0 V. On the other hand, when the input voltage is low (0 V), NMOS and PMOS transistors are off and on, respectively. The equivalent circuit of Figure 9 shows that a path exists between VDD and V_{out}, yielding a high output voltage. The gate clearly functions as an inverter.

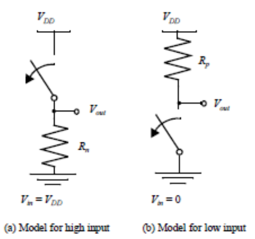


Figure 7.Switch models of CMOS inverter.

A number of other important properties of static CMOS can be derived from this switch level view:

- The high and low output levels equal *VDD* and *GND*, respectively; in other words, the voltage swing is equal to the supply voltage. This results in high noise margins.
- The logic levels are not dependent upon the relative device sizes, so that the transistors can be minimum size. Gates with this property are called *ratioless.* This is in contrast with *ratioed logic*, where logic levels are determined by the relative dimensions of the composing transistors.
- In steady state, there always exists a path with finite resistance between the output and either *VDD* or *GND*. A well-designed CMOS inverter, therefore, has a *low output impedance*, which makes it less sensitive to noise and disturbances. Typical values of the output resistance are in kW range.
- The *input resistance* of the CMOS inverter is extremely high, as the gate of an MOS transistor is a virtually perfect insulator and draws no dc input current. Since the input node of the inverter only connects to transistor gates, the steady-state input current is nearly zero. A single inverter can theoretically drive an infinite number of gates (or have an infinite fan-out) and still be functionally operational; however, increasing the fan-out also increases the propagation delay, as will become clear below. So, although fan-out does not have any effect on the steady-state behavior, it degrades the transient response.
- No direct path exists between the supply and ground rails under steady-state operating conditions (this is, when the input and outputs remain constant). The absence of current flow (ignoring leakage currents) means that the gate does not consume anystatic power.

MOS Device Model with Sub-micron Effects:

The complexity of the behavior of the short-channel MOS transistor and its many parasitic effects has led to the development of a wealth of models for varying degrees of accuracy and computing efficiency. In general, more accuracy also means more complexity and, hence, an increased run time. In this section, we briefly discuss the characteristics of the more popular MOSFET models, and describe how to instantiate a MOS transistor in a circuit description.

SPICE Models:

SPICE has three built-in MOSFET models, selected by the LEVEL parameter in the model card. Unfortunately, all these models have been rendered obsolete by the progression to short-channel devices. They should only be used for first-order analysis, and we therefore limit ourselves to a short discussion of their main properties.

- The LEVEL 1 SPICE model implements the *Shichman-Hodges model*, which is based on the square law long-channel expressions, derived earlier in this chapter. It does not handle short-channel effects.
- The LEVEL 2 model is a geometry-based model, which uses detailed device physics to define its equations. It handles effects such as velocity saturation, mobility degradation, and drain-induced barrier lowering. Unfortunately, including all 3D-effects of an advanced submicron process in a pure physics-based model becomes complex and inaccurate.
- LEVEL 3 is a semi-empirical model. It relies on a mixture of analytical and empirical expressions, and uses measured device data to determine its main parameters. It works quite well for channel lengths down to 1 μm.

In response to the inadequacy of the built-in models, SPICE vendors and semi-conductor manufacturers have introduced a wide range of accurate, but proprietary models.

Parameter Name	Symbol	SPICE Name	Units	Default Value
Drawn Length	*L*	L	m	–
Effective Width	*W*	W	m	–
Source Area	*AREA*	AS	m^2	0
Drain Area	*AREA*	AD	m^2	0
Source Perimeter	*PERIM*	PS	m	0
Drain Perimeter	*PERIM*	PD	m	0
Squares of Source Diffusion		NRS	–	1
Squares of Drain Diffusion		NRD	–	1

Table 1: Some SPICE transistor parameters

VTC Parameters (DC Characteristics):

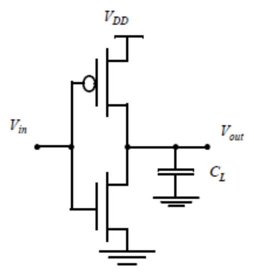


Figure 8: Static CMOS inverter

The Voltage-Transfer Characteristic:

Assume now that a logical variable *in* serves as the input to an inverting gate that produces the variable *out*. The electrical function of a gate is best expressed by its *voltage-transfer characteristic* (VTC) (sometimes called the *DC transfer characteristic*), which plots the output voltage as a function of the input voltage $V_{out} = f(V_{in})$. An example of an inverter VTC is shown in Figure 9. The high and low nominal voltages, V_{OH} and V_{OL}, can readily be identified—$V_{OH} = f(V_{OL})$ and $V_{OL} = f(V_{OH})$. Another point of interest of the VTC is the *gate or switching threshold voltage VM* (not to be confused with the threshold voltage of a transistor), that is defined as $V_M = f(V_M)$. V_M can also be found graphically at the intersection of the VTC curve and the line given by $V_{out} = V_{in}$. The gate threshold voltage presents the midpoint of the switching characteristics, which is obtained when the output of a gate is short-circuited to the input. This point will prove to be of particular interest when studying circuits with feedback (also called *sequential circuits*).

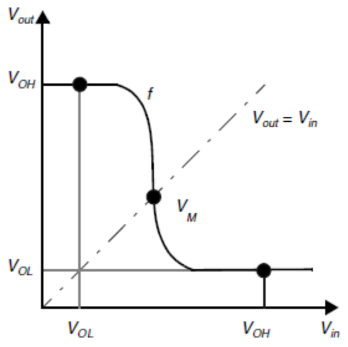


Figure 9:Inverter Voltage Transfer Characteristics(VTC)

Even if an ideal nominal value is applied at the input of a gate, the output signal often deviates from the expected nominal value. These deviations can be caused by noise or by the loading on the output of the gate (i.e., by the number of gates connected to the output signal). Figure 10(a) illustrates how a logic level is represented in reality by a range of acceptable voltages, separated by a region of uncertainty, rather than by nominal levels alone. The regions of acceptable high and low voltages are delimited by the V_{IH} and V_{IL} voltage levels, respectively. These represent by definition the points where the gain (= dV_{out} / dV_{in}) of the VTC equals −1 as shown in Figure 10(b). The region between V_{IH} and V_{IL} is called the *undefined region* (sometimes also referred to as *transition width,* or *TW*). Steady-state signals should avoid this region if proper circuit operation is to be ensured.

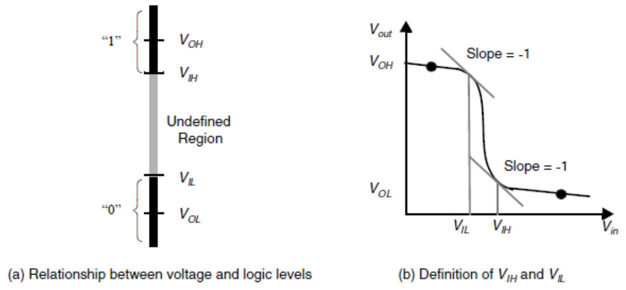


Figure 10: Mapping logic levels to the voltage domain.

Noise Margins:

For a gate to be robust and insensitive to noise disturbances, it is essential that the "0" and "1" intervals be as large as possible. A measure of the sensitivity of a gate to noise is given by the noise margins NM_L (*noise margin low*) and NM_H (*noise margin high*), which quantize the size of the legal "0" and "1", respectively, and set a fixed maximum threshold on the noise value:

$$NM_L = V_{IL} - V_{OL}$$
$$NM_H = V_{OH} - V_{IH}$$

The noise margins represent the levels of noise that can be sustained when gates are cascaded as illustrated in Figure 11. It is obvious that the margins should be larger than 0 for a digital circuit to be functional and by preference should be as large as possible.

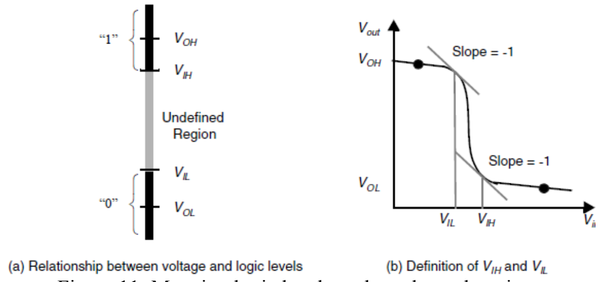


Figure 11: Mapping logic levels to the voltage domain.

CMOS Propagation Delay:

The input and output voltage waveforms of a typical inverter circuit are shown in Figure 12. The propagation delay times γ_{PHL} and γ_{pLH} determine the input-to-output signal delay during the high-to-low and low-to-high transitions of the output, respectively. By definition, γ_{PHL} is the time delay between the $V_{50\%}$-transition of the *rising* input voltage and the $V_{50\%}$ -transition of *the falling* output voltage. Similarly, γ_{pLH} is defined as the time delay between the $V_{50\%}$ transition of *the falling* input voltage and the $V_{50\%}$ transition of the *rising* output voltage.

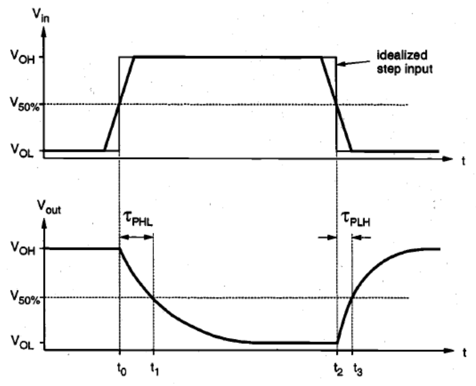


Figure 12: Input and output voltage waveforms of a typical inverter

Thus, the propagation delay times γ_{PHL} and γ_{PLH} are

$$\gamma_{PHL} = t3 - to$$
$$\gamma_{PLH} = t3 - t2$$

Calculation of parasitic capacitances:

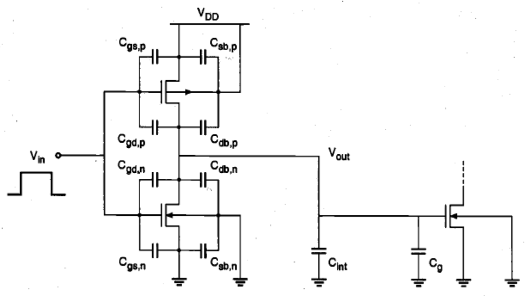


Figure 13: **Cascaded CMOS inverter stages.**

The problem of analyzing the output voltage waveform is fairly complicated, even for this relatively simple circuit, because a number of nonlinear, voltage-dependent capacitances are involved. To simplify the problem, we first combine the capacitancesseen in Fig. 6.1 into an equivalent *lumped* linear capacitance, connected between theoutput node of the inverter and the ground. This combined capacitance at the output node will be called the load capacitance, $\boldsymbol{C_{load}}$.

$$C_{load} = C_{gd},. + C_{gdp} + C_{db},,n + C_{dbp} + C_{int} + C_g$$

Layout of an Inverter:

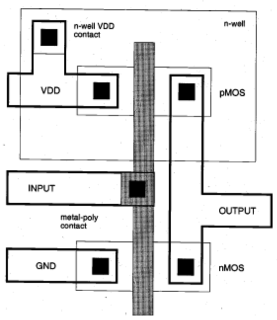


Figure 14: Complete mask layout of the CMOS inverter.

Switching Power Dissipation of CMOS Inverters:

The static power dissipation of the CMOS inverter is quite negligible. During switching events where the output load capacitance is alternatingly charged up and charged down, on the other hand, the CMOS inverter inevitably dissipates power. In the following section, we will derive the expressions for the dynamic power consumption of the CMOS inverter.Consider the simple CMOS inverter circuit shown in Fig. 15. We will assume that the input voltage is an ideal step waveform with negligible rise and fall times.

Typical input and output voltage waveforms and the expected load capacitor current waveform are shown in Fig. 16. When the input voltage switches from low to high, the pMOS transistor in the circuit is turned off, and the nMOS transistor starts conducting. During this phase, the output load capacitance C_{load} is being discharged through the nMOS transistor.

Thus, the capacitor current equals the instantaneous drain current of the nMOS transistor. When the input voltage switches from high to low, the nMOS transistor in the circuit is turned off, and the pMOS transistor starts conducting. During this phase, the output load capacitance C_{load} is being charged up through the pMOS transistor; therefore, the capacitor current equals the instantaneous drain current of the pMOS transistor.

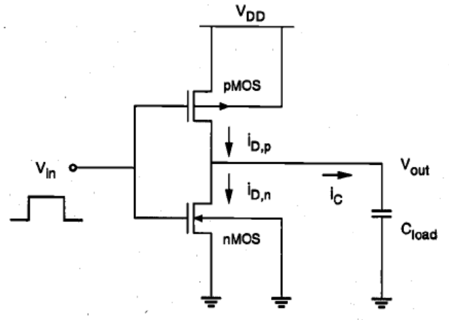


Figure 15: CMOS inverter used in the dynamic power-dissipation analysis.

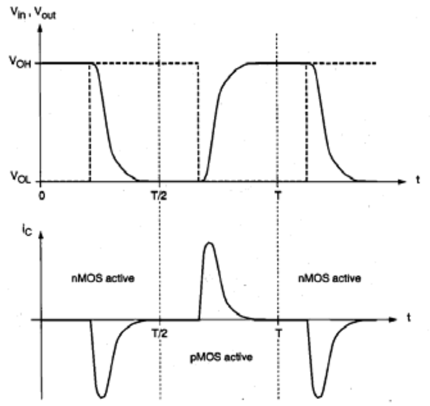


Figure 16: Typical input and output voltage waveforms and the capacitor current waveform during switching

Assuming periodic input and output waveforms, the average power dissipated by any device over one period can be found as follows:

$$P_{avg} = \frac{1}{T}\int_0^T V(t)i(t)dt$$

Since during switching, the nMOS transistor and the pMOS transistor in a CMOS inverter conduct current for one-half period each, the average power dissipation of the CMOS inverter can be calculated as the power required to charge up and charge down the output load capacitance.
By solving we get $P_{avg}=C_{load}\,V_{DD}^2 f$

Estimation of Interconnect Parasitics:

The classical approach for determining the switching speed of a logic gate is based on the assumption that the loads are mainly capacitive and lumped. We have examined the relatively simple delay models for inverters with purely capacitive load at the output node, which can be used to estimate the transient behavior of the circuit once the load is determined. The conventional delay estimation approaches seek to classify three main components of the output load, all of which are assumed to be purely capacitive, as: (i) internal parasitic capacitances of the transistors, (ii) interconnect (line) capacitances, and (iii) input capacitances of the fan-out gates. Of these three components, the load conditions imposed by the interconnection lines present serious problems, especially in submicron circuits.

Figure 17 shows a simple situation where an inverter is driving three other inverters, linked by interconnection lines of different length and geometry. If the load from each interconnection line can be approximated by a lumped capacitance, then the total load seen by the primary inverter is simply the sum of all capacitive components described above. In most cases, however, the load conditions imposed by the interconnection line are far from being simple. The line, itself a three-dimensional structure in metal and/or polysilicon, usually has a non-negligible resistance in addition to its capacitance. The (length/width) ratio of the wire usually dictates that the parameters are distributed, making the interconnect a true transmission line. Also, an interconnect is rarely isolated from other influences. In realistic conditions, the interconnection line is in very close proximity to a number of other lines, either on the same level or on different levels. The capacitive/inductive coupling and the signal interference between neighboring lines should also be taken into consideration for an accurate estimation of delay.

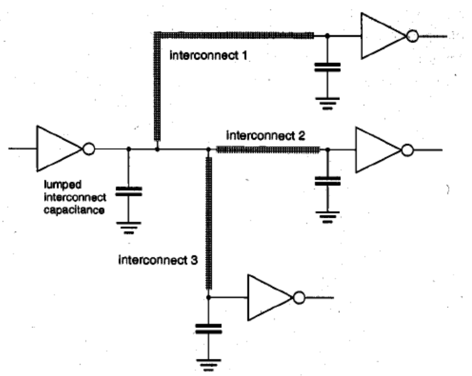


Figure 17: An inverter driving three other inverters over interconnection lines.

In general, if the time of flight across the interconnection line (as determined by the speed of light) is much shorter than the signal rise/fall times, then the wire can be modeled as a capacitive load, or as a lumped or distributed RC network. If the interconnection lines are sufficiently long and the rise times of the signal waveforms are comparable to the time of flight across the line, then the inductance also becomes important, and the interconnection lines must be modeled as transmission lines. The following is a simple rule of thumb which can be used to determine when to use transmission-line models.

$\gamma_{rise}(\gamma_{fall})<2.5\text{x}(L/V)$ => Transmission line modeling

$2.5\text{x}(L/V) < \gamma_{rise}(\gamma_{fall}) < 5\text{x}(L/V)$ Either transmission line or lumped modeling

$\gamma_{rise}(\gamma_{fall})> 5\text{x}(L/V)$

Here, L is the interconnect line length, and V is the propagation speed. Note that transmission line analysis always gives the correct result irrespective of the rise/fall time and the interconnect length; yet the same result can be obtained with the same accuracy using lumped approximation when rise/fall times are sufficiently large.

The transmission-line effects have not been a serious concern in CMOS VLSI chips until recently, since the gate delays due to capacitive load components dominated the line delay in most cases. But as the fabrication technologies move to finer submicron design rules, the intrinsic gate delays tend to decrease significantly. By contrast, the overall chip size and the worst-case line length on a chip tend to increase mainly due to increasing chip complexity, thus, the importance of interconnect delay increases in submicron technologies. In addition, as the widths of metal lines shrink, the transmission line effects and signal coupling between neighboring lines become even more pronounced.

This fact is illustrated in Figure 18, where typical intrinsic gate delay and interconnect delay are plotted qualitatively, for different technologies. It can be seen that for submicron technologies, the interconnect delay starts to dominate the gate delay. In order to deal with the implications and to optimize a system for speed, chip designers must have reliable and efficient means for (i) estimating the interconnect parasitics in a large chip, and (ii) simulating the transient effects.

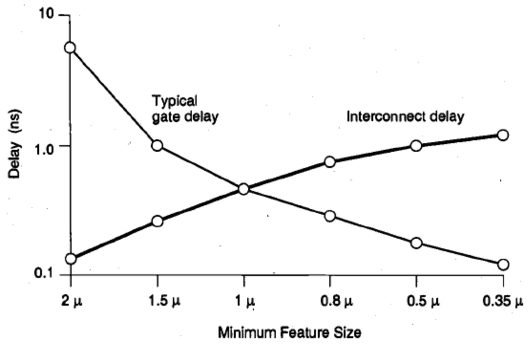


Figure 18: Interconnect delay dominates gate delay in submicron CMOS technologies.

Interconnect Capacitance Estimation:

In a large-scale integrated circuit, 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 19 shows a simplified view of six interconnections on three different levels, running 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 a complicated task.

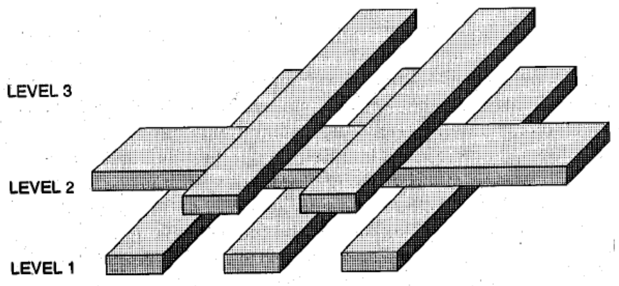


Figure 19: An example of six interconnect lines running on three different levels.

First, consider the section of a single interconnect which is shown in Figure 20. It is assumed that this wire segment has a length of *(1)* 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 Figure 20, 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 (figure 21).

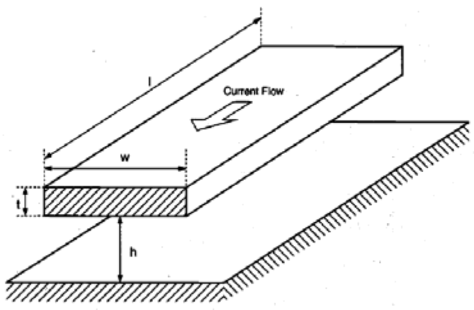


Figure 20: Interconnect segment running parallel to the surface, which is used for parasitic resistance and capacitance estimations.

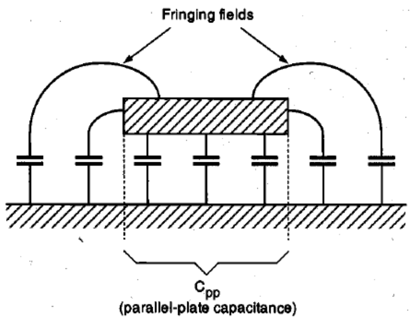


Figure 21: Influence of fringing electric fields upon the parasitic wire capacitance.

Field oxide thickness	0.52	µm	
Gate oxide thickness	16.0	nm	(= 0.016 µm)
Polysilicon thickness	0.35	µm	(minimum width 0.8 µm)
Poly - metal oxide thickness	0.65	µm	
Metal - 1 thickness	0.60	µm	(minimum width 1.4 µm)
Via oxide thickness	1.00	µm	
Metal - 2 thickness	1.00	µm	(minimum width 1.4 µm)
n + junction depth	0.40	µm	
p + junction depth	0.40	µm	
n - well junction depth	3.50	µm	

Table 2: Thickness values of different layers in a typical 0.8 micron CMOS process.

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 excessive 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 capacitance) and the perimeter capacitance (fringing-field capacitance) figures for each layer, which are backed up by measurement of capacitance test structures. These figures can be used to extract the parasitic capacitance 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 or coupling of a specific line, accurate 3-D simulation of interconnect parasitic effects is the only reliable solution.

Poly over field oxide	C_{pf}	Area	0.066	$fF/\mu m^2$
		Perimeter	0.046	$fF/\mu m$
Metal - 1 over field oxide	C_{m1f}	Area	0.030	$fF/\mu m^2$
		Perimeter	0.044	$fF/\mu m$
Metal - 2 over field oxide	C_{m2f}	Area	0.016	$fF/\mu m^2$
		Perimeter	0.042	$fF/\mu m$
Metal - 1 over Poly	C_{m1p}	Area	0.053	$fF/\mu m^2$
		Perimeter	0.051	$fF/\mu m$
Metal - 2 over Poly	C_{m2p}	Area	0.021	$fF/\mu m^2$
		Perimeter	0.045	$fF/\mu m$
Metal - 2 over Metal - 1	C_{m2m1}	Area	0.035	$fF/\mu m^2$
		Perimeter	0.051	$fF/\mu m$

Table 3: Parasitic capacitance values between various layers, for atypical double-metal 0.8 micron CMOS technology.

Interconnect Resistance Estimation:

The parasitic resistance of a metal or polysilicon line can also have a significant influence on the signal propagation delay over that line. The resistance of a line depends on the type of material used (e.g., polysilicon, aluminum, or 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 Figure 20. The total resistance in the indicated current direction can be found as

$$R_{wire}=\rho x(L/W.t)=R_{sheet}(L/W)$$

where (ρ) represents the characteristic resistivity of the interconnect material, and R_{sheet} represents the sheet resistivity of the line, in (Ω/square).

$$R_{sheet}=\rho/t$$

Calculation of Interconnect Delay:

Two models are introduced here for calculation of interconnected delay.

- RC Delay Models
- The Elmore Delay

RC Delay Models

As we have already discussed in the previous Section, an interconnect line can be modeled as a lumped RC network if the time of flight across the interconnection line is significantly shorter than the signal rise/fall times. This is usually the case in most on-chip interconnects, thus, we will mainly concentrate on the calculation of delay in RC networks, in the following. The simplest model which can be used to represent the resistive and capacitive parasitics of the interconnect line consists of one lumped resistance and one lumped capacitance (Figure 22). Assuming that the capacitance is discharged initially, and assuming that the input signal is a rising step pulse at time t = 0, the output voltage waveform of this simple RC circuit is found as

$$V_{out}(t)=V_{DD}\left(1-e^{-\frac{t}{RC}}\right)$$

And the propagation delay for the simple lumped RC network is found as $\tau = 0.69\,RC$

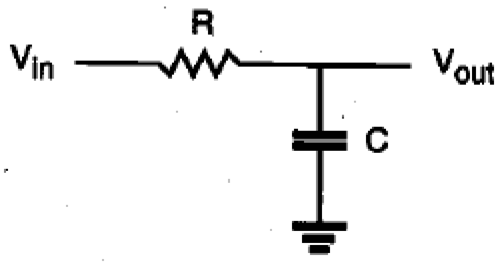


Figure 22: Simple lumped RC model of an interconnect line, where R and C represent the total line resistance and capacitance, respectively

The Elmore Delay:

Consider a general RC tree network, as shown in Figure 23. Note that (i) there are no resistor loops in this circuit, (ii) all of the capacitors in an RC tree are connected between a node and the ground, and (iii) there is one input node in the circuit. Also notice that there is a unique resistive path, from the input node to any other node in the circuit. Inspecting the general topology of this RC tree network, we can make the following path definitions:

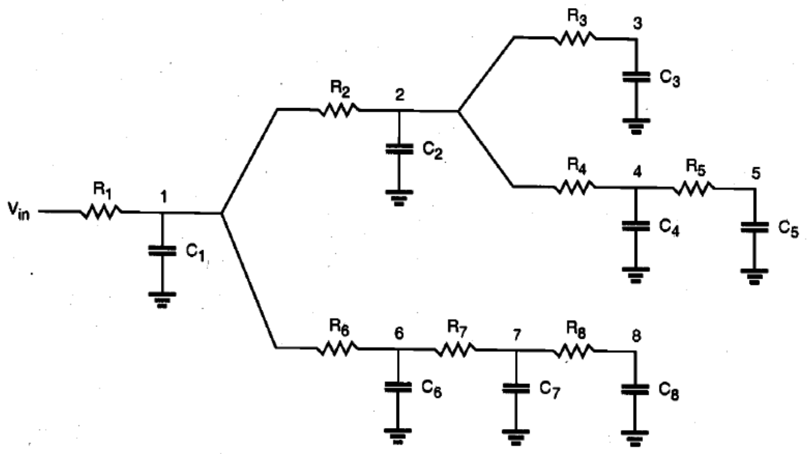


Figure 23. A general RC tree network consisting of several branches.

Assuming that the input signal is a step pulse at time *t* = 0, the Elmore delay at node i of this RC tree is given by the following expression.

$$\tau_{Di} = \sum_{j=1}^{N} C_j \sum_{\substack{\text{for all} \\ k \in P_{ij}}} R_k$$

Calculation of the Elmore delay is equivalent to deriving the first-order time constant (first moment of the impulse response) of this circuit. Note that although this delay is still an *approximation* for the actual signal propagation delay from the input node to node i, it provides a fairly simple and accurate means of predicting the behavior of the RC line. The procedure to calculate the delay at any node in the circuit is very straightforward. For example, the Elmore delay at node 7 can be found according to the above equation.

$$\tau_{D7} = R_1 C_1 + R_1 C_2 + R_1 C_3 + R_1 C_4 + R_1 C_5 + (R_1 + R_6) C_6 + (R_1 + R_6 + R_7) C_7 + (R_1 + R_6 + R_7) C_8$$

as in Elmore Delay model is τ = 0.50 RC

Module 3

Combinational Logic Circuits

Combinational logic circuits, or gates, which perform Boolean operations on multiple input variables and determine the outputs as Boolean functions of the inputs, are the basic building blocks of all digital systems.In this chapter, we will examine the static and dynamic characteristics of various combinational MOS logic circuits.

The first major class of combinational logic circuits to be presented in this chapter is the nMOS depletion-load gates. Our purpose for including nMOS depletion-load circuits here is mainly pedagogical, to emphasize the load concept, which is still being widely used in many areas in digital circuit design. We will examine simple circuit configurations such as two-input NAND and NOR gates and then expand our analysis to more general cases of multiple-input circuit structures. Next, the CMOS logic circuits will be presented in a similar fashion. We will stress the similarities and differences between the nMOS depletion-load logic and CMOS logic circuits and point out the advantages of CMOS gates with examples. The design of complex logic gates, which allows the realization of complex Boolean functions of multiple variables, will be examined in detail. Finally, we will devote the last section to CMOS transmission gates and to transmission gate (TG) logic circuits.

In its most general form, a combinational logic circuit, or gate, performing a Boolean function can be represented as a multiple-input single-output system.

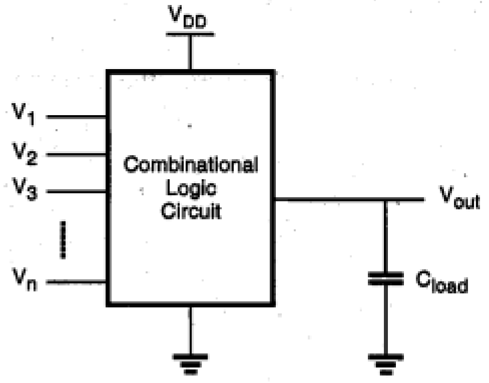


Figure 24: Generic combinational logic circuit (gate).

All input variables are represented by node voltages, referenced to the ground potential.Using positive logic convention, the Boolean(or logic) value of "1" can be represented by a high voltage of VDD, and the Boolean(or logic) value of "0" can be represented by a low voltage of 0. The output node is loaded with capacitance CL, which represents the combined parasitic device capacitances in the circuit and the interconnect capacitance components seen by the output node. This output load capacitance certainly plays a very significant role in the dynamic operation of the logic gate.

As in the simple inverter case, the voltage transfer characteristic (VTC) of a combinational logic gate provides valuable information on the DC operating performance of the circuit. Critical voltage points such as VOL or Vth are considered to be important design parameters for combinational logic circuits. Other design parameters and concerns include the dynamic (transient) response characteristics of the circuit, the silicon area occupied by the circuit, and the amount of static and dynamic power dissipation.

Static CMOS Logic Circuits:

Complementary CMOS

Introduction- Complementary Static CMOS design offers low noise sensitivity, speed and low power consumption. Static CMOS design means that at any time, the output of the gate is directly connected to VSS or VDD. In comparison, Dynamic CMOS gates depend upon the device capacitance to temporarily hold charge at the output.

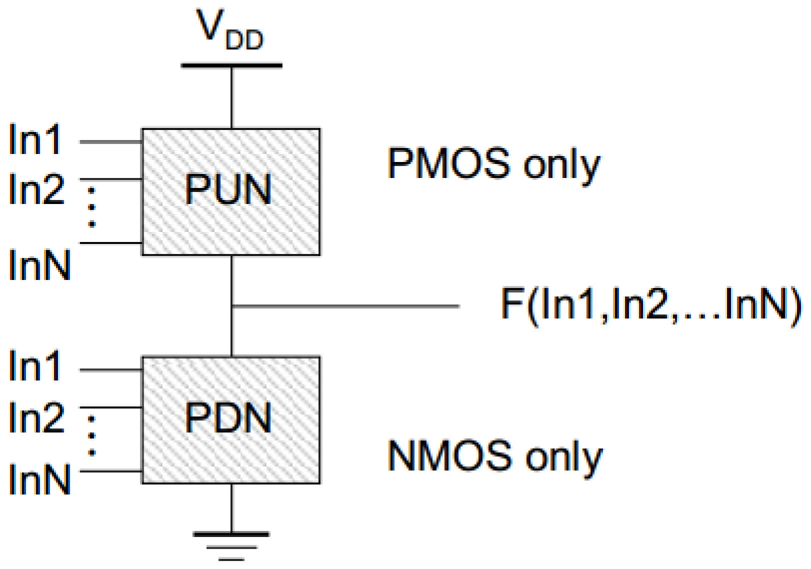


Figure 25: Pull up and Pull down networks

Properties of Complementary Static CMOS

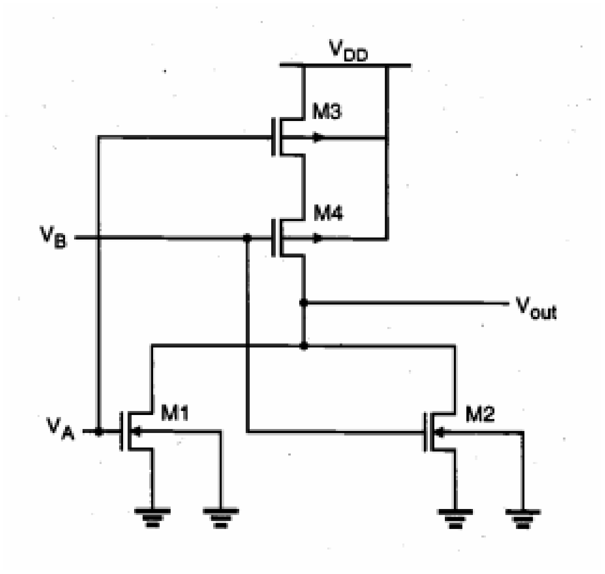
1) Contains a pull up network (PUP) and pull down network (PDN)
2) PUP networks consist of PMOS transistors
3) PDN networks consist of NMOS transistors
4) Each network is the dual of the other network
5) The output of the complementary gate is inverted.

PDN Design - Any function is first designed by the Pull Down network. The Pull Up network can be designed by simply taking the dual of the Pull Down network, once it is found.An NMOS gate will pass current between source and drain when 5 volts is presented at the gate.

1) All functions are composed of either and'ed or or'ed sub-functions.
2) The And function is composed of NMOS transistors in series.
3) The Or function is composed of NMOS transistors in parallel.

CMOS Logic Circuits

CMOS NOR2 (Two-Input NOR) Gate



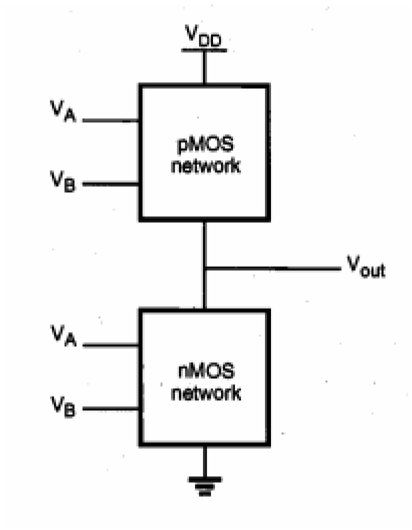


Figure 26: A CMOS NOR2 gate and its complementary operation: Either the nMOS network is on and the pMOS network is off, or the pMOS network is on and the nMOS network is off.

The circuit consists of a parallel-connected n-net and a series-connected complementary pnet. The input voltages VA and VB are applied to the gates of one nMOS and one pMOS transistor.

The complementary nature of the operation can be summarized as follows: When either one or both inputs are high, i.e., when the n-net creates a conducting path between the output node and the ground, the p-net is cut-off. On the other hand, if both input voltages are low, i.e., the n-net is cut-off, then the p-net creates a conducting path between the output node and the supply voltage VDD. Thus, the dual or complementary circuit structure allows that, for any given input combination, the output is connected either to VDD or to ground via a low-resistance path. A DC current path between the VDD and ground is not established for any of the input combinations.

The output voltage of the CMOS NOR2 gate will attain a logic-low voltage of VOL= 0 and a logic-high voltage of VOH = VDD. For circuit design purposes, the switching threshold voltage Vth of the CMOS gate emerges as an important design criterion. We start our analysis of the switching threshold by assuming that both input voltages switch simultaneously, i.e.VA =VB. Furthermore, it is assumed that the device sizes in each block are identical, (W/L)n,A = (W/L)n,B and (W/L)p,A = (W/L)p,B and the substrate-bias effect for the pMOS transistors is neglected for simplicity.

By definition, the output voltage is equal to the input voltage at the switching threshold.

$$V_A = V_B = V_{out} = V_{th}$$

It is obvious that the two parallel nMOS transistors are saturated at this point, because VGS=VDS. The combined drain current of the two nMOS transistors is

$$I_D = k_n\left(V_{th} - V_{T,n}\right)^2$$

Thus, we obtain the first equation for the switching threshold Vth

$$V_{th} = V_{T,n} + \sqrt{\frac{I_D}{k_n}}$$

Examination of the p-net shows that the pMOS transistor M3 operates in the linear region, while the other pMOS transistor, M4, is in saturation for Vth=Vout.Thus,

$$I_{D3} = \frac{k_p}{2}\left[2\left(V_{DD} - V_{th} - \left|V_{T,p}\right|\right)V_{SD3} - V_{SD3}^2\right]$$

$$I_{D4} = \frac{k_p}{2}\left(V_{DD} - V_{th} - \left|V_{T,p}\right| - V_{SD3}\right)^2$$

The drain currents of bothpMOS transistors are identical, i.e., ID3=ID4=ID. Thus,

$$V_{DD} - V_{th} - \left|V_{T,p}\right| = 2\sqrt{\frac{I_D}{k_p}}$$

This yields the second equation of the switching threshold voltage Vth

$$V_{th}(\mathrm{NOR2}) = \frac{V_{T,n} + \frac{1}{2}\sqrt{\frac{k_p}{k_n}}\left(V_{DD} - \left|V_{T,p}\right|\right)}{1 + \frac{1}{2}\sqrt{\frac{k_p}{k_n}}}$$

If kn = kp and V T,n = |VT,p|, the switching threshold of the CMOS inverter is equal to VDD/2. Using the same parameters, the switching threshold of the NOR2 gate is

$$V_{th}(\text{NOR2}) = \frac{V_{DD} + V_{T,n}}{3}$$

which is not equal to VDD/2. For example, when VDD = 5 V and VT,n =|VT,p| = 1 V, the switching threshold voltages of the NOR2 gate and the inverter are

$$V_{th}(\text{NOR2}) = 2\text{ V}$$

$$V_{th}(\text{INR}) = 2.5\text{ V}$$

The switching threshold voltage of the NOR2 gate can also be obtained by using the equivalent-inverter approach. When both inputs are identical, the parallel-connected nMOS transistors can be represented by a single nMOS transistor with 2kb. Similarly, the series-connected pMOS transistors are represented by a single pMOS transistor with Kp/2.

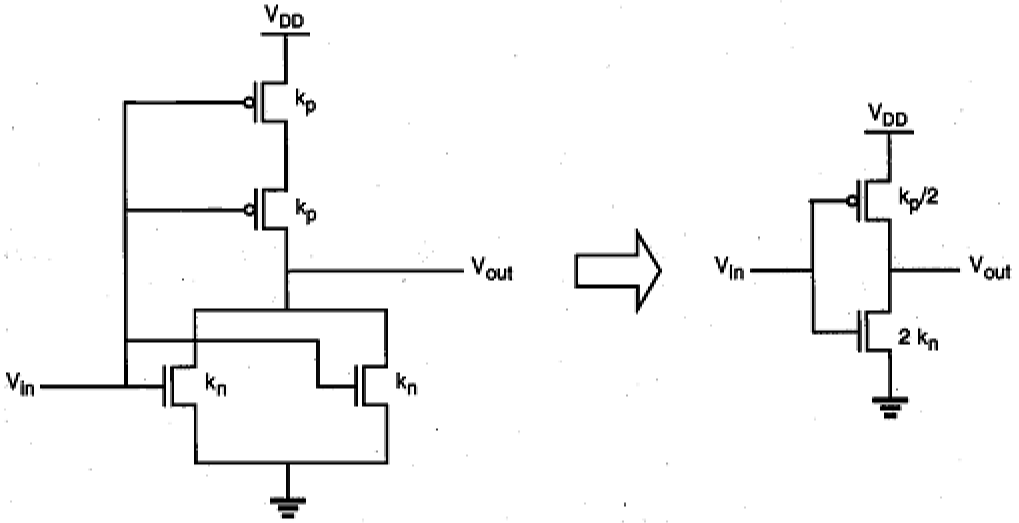


Figure 27: A CMOS NOR2 gate and its inverter equivalent.

CMOS NAND2 (Two-Input NAND) Gate

The operating principle of this circuit is the exact dual of the CMOS NOR2 operation examined earlier. The n-net consisting of two series-connected nMOS transistors creates a conducting path between the output node and the ground only if both input voltages are logic-high, i.e., are equal to VOH In this case, both of the parallel-connected pMOS transistors in the p-net will be off. For all other input combinations, either one or both of the pMOS transistors will be turned on, while the n-net is cut-off, thus creating a current path between the output node and the power supply voltage.

Using an analysis similar to the one developed for the NOR2 gate, we can easily calculate the switching threshold for the CMOS NAND2 gate. Again, we will assume that the device sizes in each block are identical, with (W/L)n,A = (W/L)n,B and (W/L)p,A = (W/L)p,B. The switching threshold for this gate is then found as

$$V_{th}(\mathrm{NAND2}) = \frac{V_{T,n} + 2\sqrt{\frac{k_p}{k_n}}\left(V_{DD} - |V_{T,p}|\right)}{1 + 2\sqrt{\frac{k_p}{k_n}}}$$

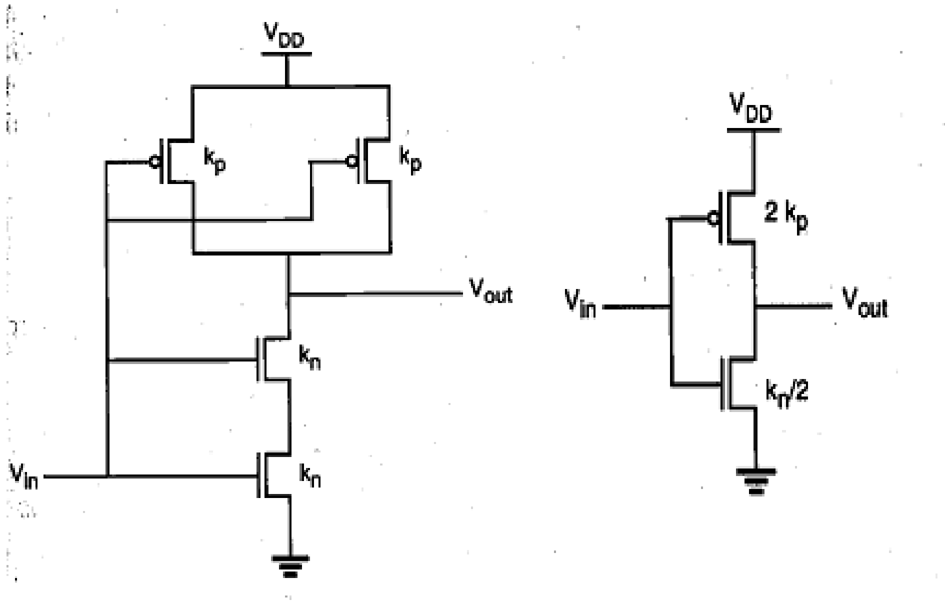


Figure 28: A CMOS NAND2 gate and its inverter equivalent.

Ratioed Logic

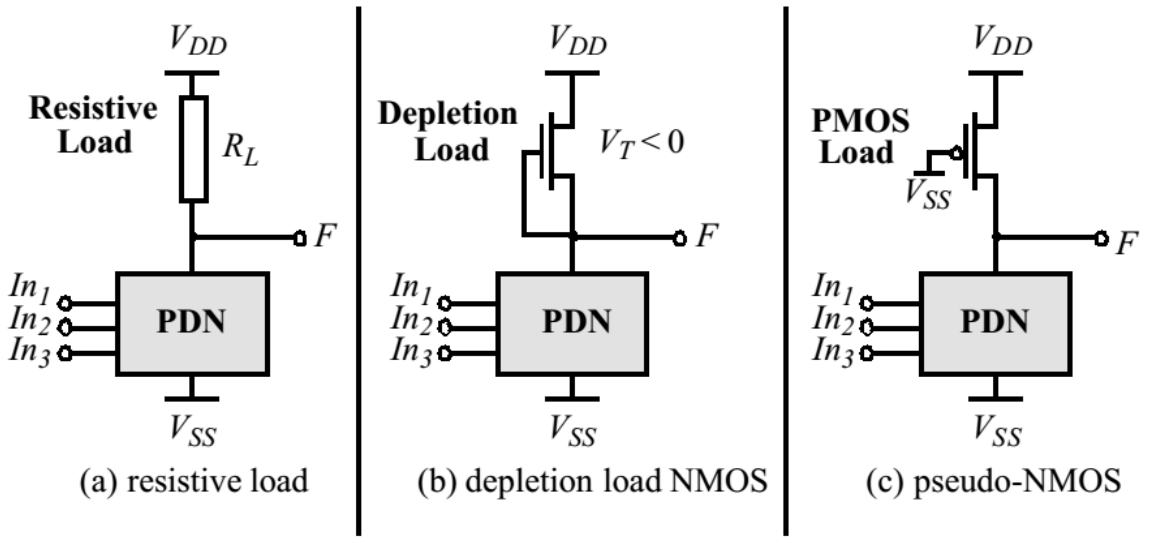
It is one method to reduce the circuit complexity of static CMOS. Here, the logic function is built in the PDN and used in combination with a simple load device

Comparison of Logic Families

Complementary CMOS	Ratioed Logic
Full Swing	Reduced Swing (smaller noise margin)
No Static Power	Static Power

tpHL= tpLH
if sized correctly
	tpLH>tpHL

Large PMOS parasitic capactiances	much smaller parasitic capactiances



DC characteristics:

ƒ VOH= VDD

ƒ VOL depends on PMOS to NMOS ratio

Let's assume the load can be represented as linearized resistors. When the PDN is on, the output voltage is determined by:

$$V_{OL} = \frac{R_{PDN}}{R_L + R_{PDN}} V_{DD}$$

This logic style is called ratioed because care must be taken in scaling the impedances properly. Note that full complementary CMOS is ratioless, since the output signals do not depend on the size of the transistors. In order to keep the noise margins high, RL >> RPDN

However, RL must be able to provide as much current as possible to minimize delay.

$$t_{pLH} = 0.69 R_L C_L$$

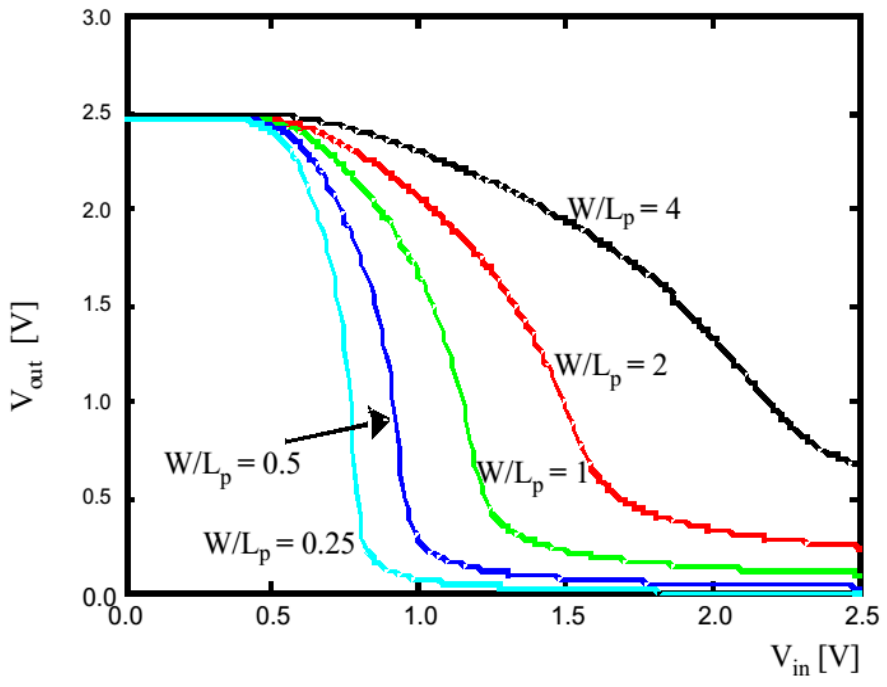
$$t_{pHL} = 0.69 (R_L // R_{PDN}) C_L$$

These are conflicting requirements:

RL large: Noise margins.

RL small: performance and power dissipation.

Pseudo-NMOS VTC



Pass-Transistor Logic

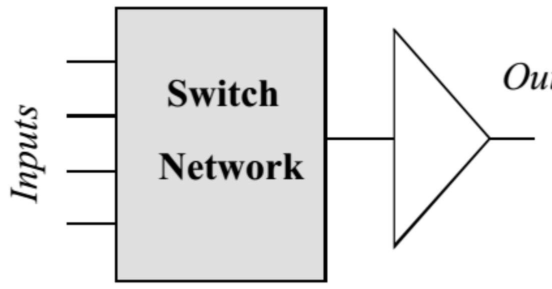
Introduction

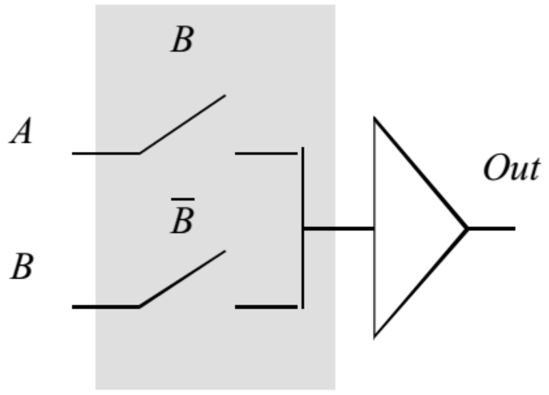
Realizing boolean functions using transistors as a switch, this is typically known as pass transistor logic circuits.

Characteristics of an ideal switch

- It has zero on resistance (Vin=Vout)
- Infinite(very very high) OFF resistance (Vout=0 v), previous value retained in case of capacitive load.

- Relay is an ideal switch. In relay logic, presence of high voltage level is considered to be “1”. Absence of high voltage level is considered to be “0”.
- Pass transitor logic is different from relay logic.



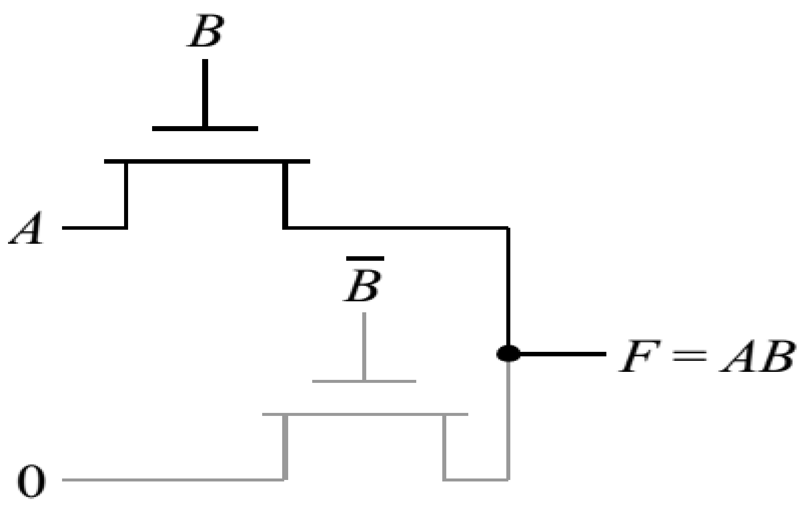


- N inputs, N transistors
- No static power consumption (ideally)

Pass transistor logic Design rules

- One must not drive the gate of a pass transistor by the output of another pass transistor.
- It is essential to provide both charging and discharging path for the load capacitance.
- When a signal is steered through several stages of pass transistors, the delay can be considerable.

Example: AND Gate



When B is “1”, top device turns on and copies the input A to output F. When B is low, bottom device turns on and passes a “0”.

The presence of the switch driven by B is essential to ensure that the gate is static – a low-impedance path must exist to supply rails.

Advantage:

Fewer devices to implement some functions.

Example: AND2 requires 4 devices (including inverter to invert B) vs. 6 for complementary CMOS (lower total capacitance).

NMOS is effective at passing a 0, but poor at pulling a node to Vdd. When the pass transistor a node high, the output only charges up to Vdd- Vtn. This becomes worse due to the body effect. The node will be charged up to Vdd– Vtn(Vs)

Complementary Pass Transistor Logic

The complexity of full-CMOS pass-gate logic circuits can be reduced dramatically by adopting another circuit concept, called Complementary Pass-transistor Logic (CPL). The main idea behind CPL is to use a purely nMOS pass-transistor network for the logic operations, instead of a CMOS TG network. All inputs are applied in complementary form, i.e., every input signal and its inverse must be provided; the circuit also produces complementary outputs, to be used by subsequent CPL stages. Thus, the CPL circuit essentially consists of complementary inputs, an nMOS pass transistor logic network to generate complementary outputs, and CMOS output inverters to restore the output signals.

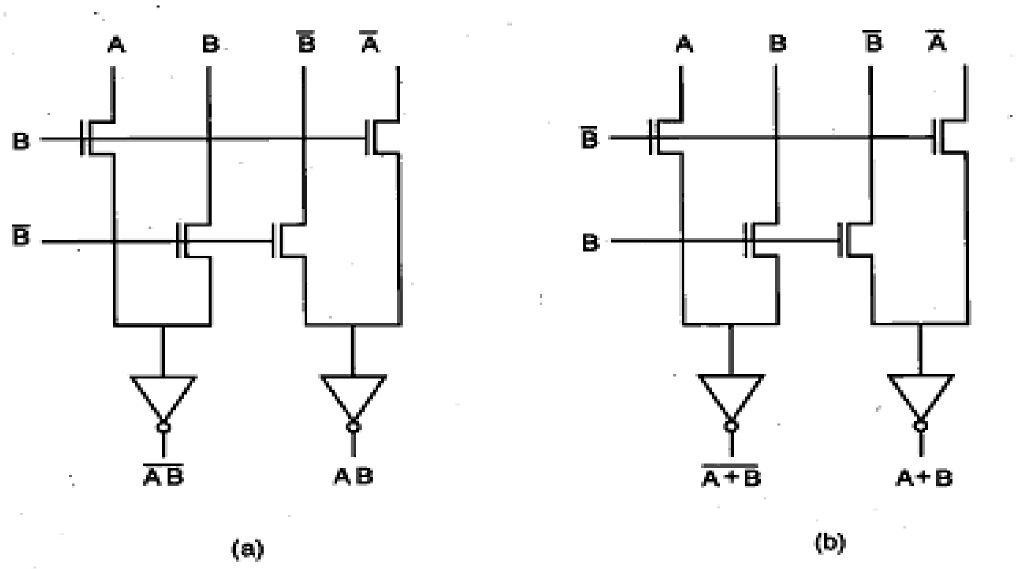


Figure 29: Circuit diagram of (a) CPL NAND2 gate and (b) CPL NOR2 gate

The elimination of pMOStransistors from the pass-gate network significantly reduces the parasitic capacitances associated with each node in the circuit, thus, the operation speed is typically higher compared to a full-CMOS counterpart. But the improvement in transient characteristics comes at a price of increased process complexity. In CPL circuits, the threshold voltages of the nMOStransistors in the pass-gate network must be reduced to about 0 V through threshold-adjustment implants, in order to eliminate the threshold-voltage drop. This, on the other hand, reduces the overall noise immunity and makes the transistors more susceptible to subthreshold conduction in the off-mode. Also note that the CPL design style is highly modular, a wide range of functions can be realized by using the same basic pass-transistor structures.

Transmission Gate Logic

The CMOS transmission gate consists of one nMOS and one pMOS transistor, connected in parallel. The gate voltages applied to these two transistors are also set to be complementary signals. As such, the CMOS TG operates as a bidirectional switch between the nodes A and B which is controlled by signal C.

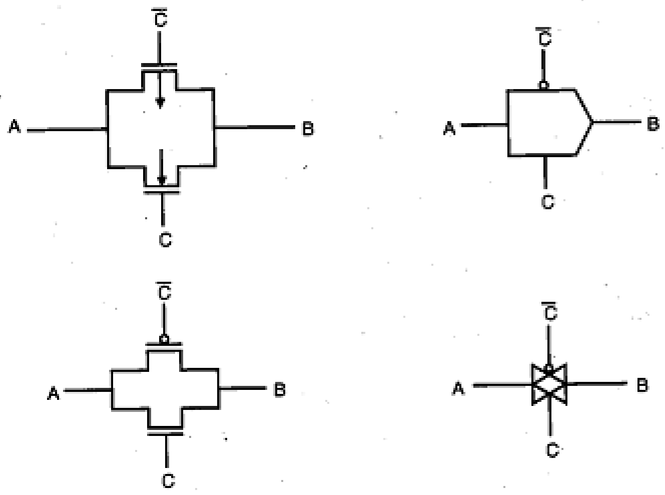
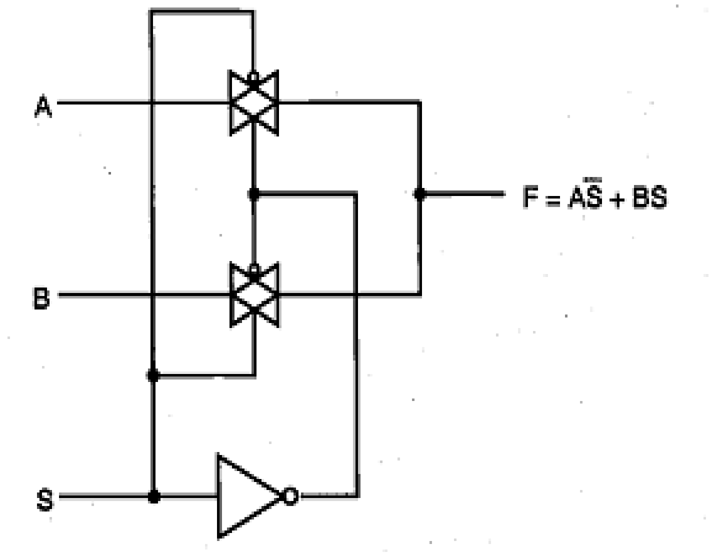


Figure 30: Four different representation of CMOS transmission gate(TG)

If the control signal C is logic-high, i.e., equal to VDD, then both transistors are turned on and provide a low-resistance current path between the nodes A and B. If, on the other hand, the control signal C is low, then both transistors will be off, and the path between the nodes A and B will be an open circuit. This condition is also called the high-impedance state.

Charecteristics

- NMOS passes a strong “0”
- PMOS passes a strong “1”
- Transmission gates enable rail-to-rail swing
- These gates are particularly efficient in implementing MUXs



Two-input multiplexor circuit implemented using two CMOS TGs.

DCVS (Differential Cascode Voltage Switch Logic) Logic

It requires mainly N-channel MOSFET transistors to implement the logic using true and complementary input signals, and also needs two P-channel transistors at the top to pull one of the outputs high.

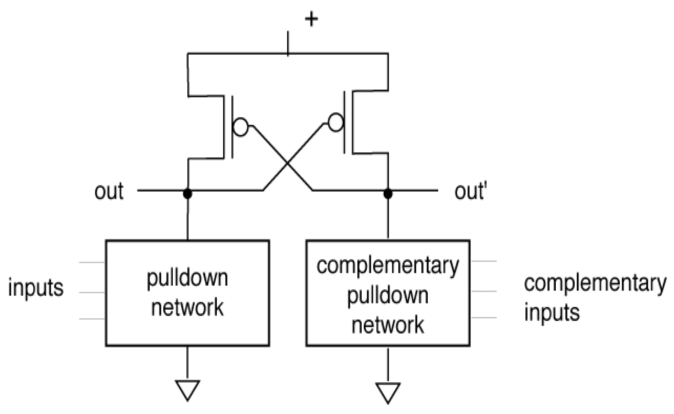
Static logic—consumes no dynamic or static power.

Uses latch to compute output quickly.

Requires true/complement inputs, produces true/complement outputs.

The cascode (sometimes verbified to cascoding) is a universal technique for improving analog circuit performance, applicable to both vacuum tubes and transistors. The word was first used in an article by F.V. Hunt and R.W. Hickman in 1939, in a discussion for application in low-voltage stabilizers. They proposed a cascade of two triodes (first one with common cathode, the second one with common grid) as a replacement of a pentode.

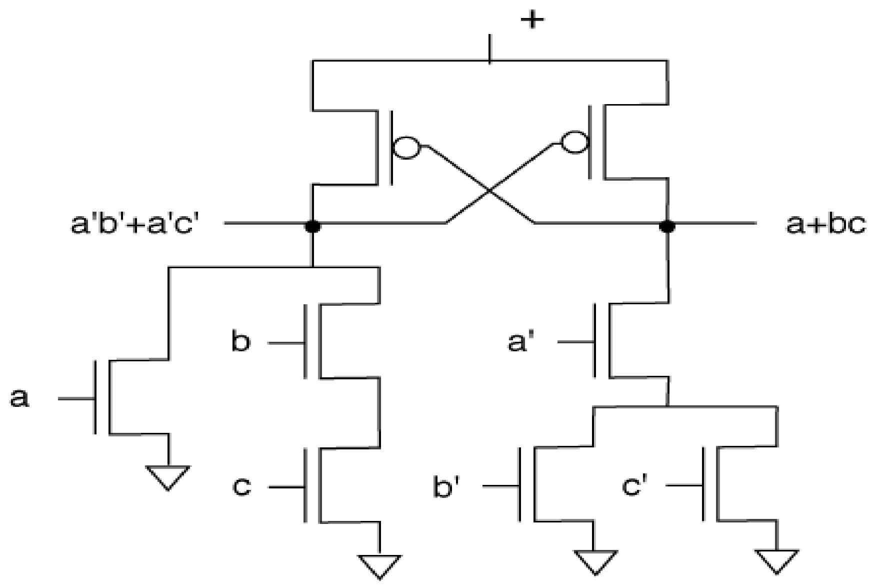
DCVS structure



DCVS operation

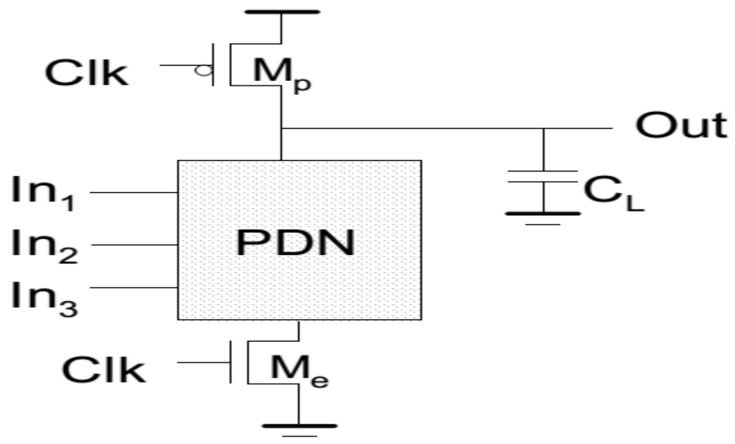
- ➢ Exactly one of true/complement pulldown networks will complete a path to the power supply.
- ➢ Pulldown network will lower output voltage, turning on other p-type, which also turns off p-type for node which is going down.

DCVS example



<u>Dynamic CMOS Logic Circuits</u>

- Output not permanently connected to Vdd or Vss
- Output value partly relies on storage of signal values on the capacitance of high impedance circuit nodes.
- Input only active when clockis active
- Requires N+2 transistors for N inputs



Two phase operation

Precharge (CLK = 0) – Me is off => no static power

Evaluate (CLK = 1) –Mp is off

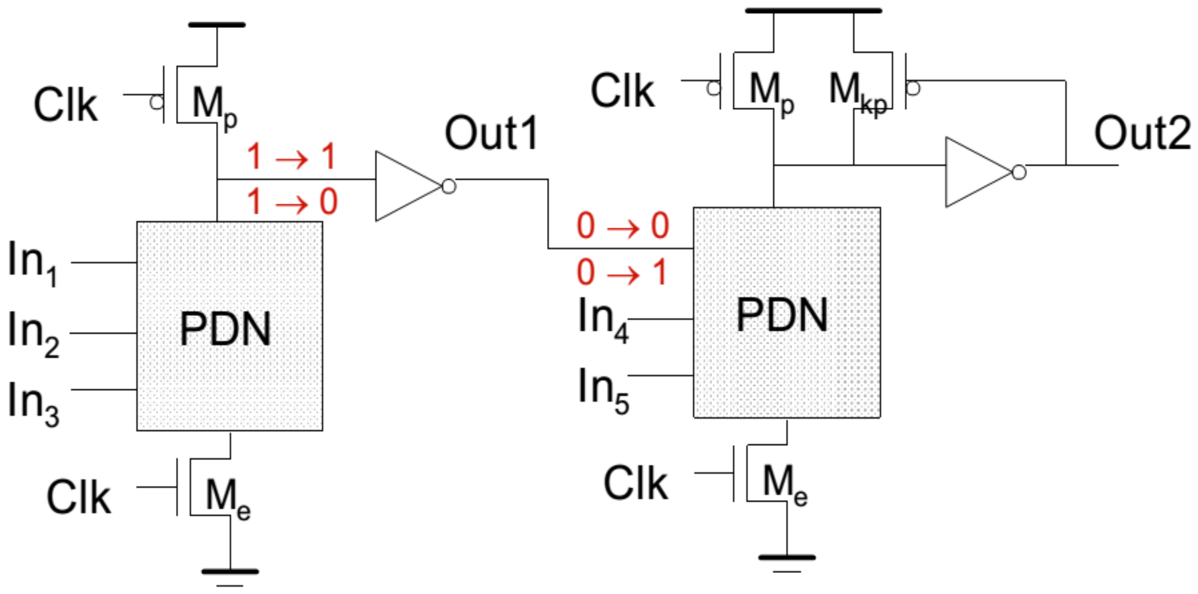
Conditions on Output

- Once the output of a dynamic gate is discharged, it cannot be charged again until the next precharge operation**.**
- Inputs to the gate can make at most one transition during evaluation.
- Output can be in the high impedance state during and after evaluation (PDN off), state is stored on CL (fundamentally different than static gates)

Properties of Dynamic Gates

- Logic function is implemented by the PDN only(number of transistors is N + 2 (versus 2N for static complementary CMOS)
- Full swing outputs (VOL= GND and VOH= VDD)
- Non-ratioed - sizing of the devices does not affect the logic levels-Sizing the PMOS doesn't impact correct functionality. Sizing it up improves the low- to-high transition time (not too critical), but trades- off increase in clock -power dissipation.
- Faster switching speeds
    - reduced load capacitance due to lower input capacitance (Cin)
    - reduced load capacitance due to smaller output loading (Cout)
    - no Isc, so all the current provided by PDN goes into discharging CL
- Overall power dissipation usually higher than static CMOS
    - no static current path ever exists between VDD and GND (including Psc)
    - higher transition probabilities
    - extra load on Clock -> higher dynamic power consumption
- Needs a precharge /evaluate clock

Domino Logic



The inverter:

(a) ensures that all inputs are set to "0" at the end of the precharge phase

(b) has a low impedance output, which increases the noise immunity.

(c) Can be used to drive a keeper device to combat leakage and charge redistribution.

Why Domino?

- Since each dynamic gate has a static inverter, only non -inverting logic can be implemented (there are ways to deal with this)
- Very high speed
    - Input capacitance reduced

Sequential Logic Circuits

Introduction:

Combinational logic circuits that were described earlier have the property that the output of a logic block is only a function of the current input values, assuming that enough time has elapsed for the logic gates to settle. Yet virtually all useful systems require storage of state information, leading to another class of circuits called sequential logic circuits. In these circuits, the output not only depends upon the current values of the inputs, but also upon preceding input values. In other words, a sequential circuit remembers some of the past history of the system—it has memory.

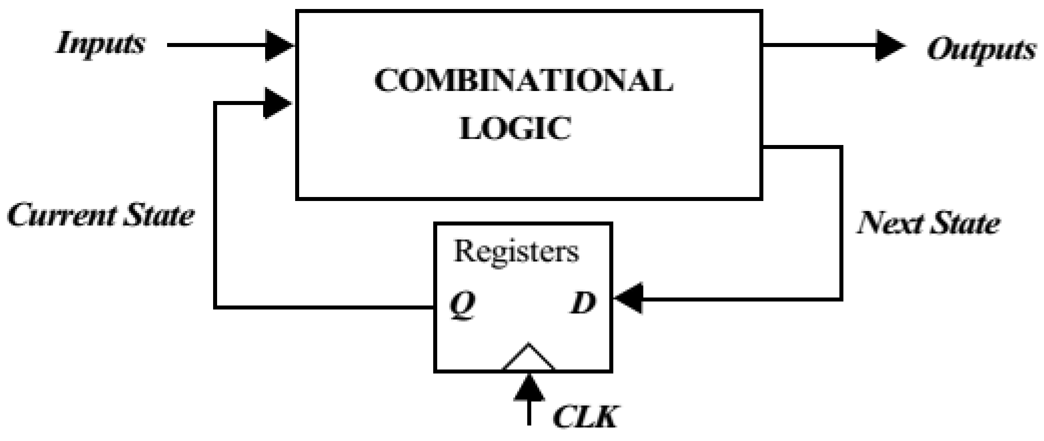


Figure 31:Block diagram of a finite state machine using positive edge-triggered registers

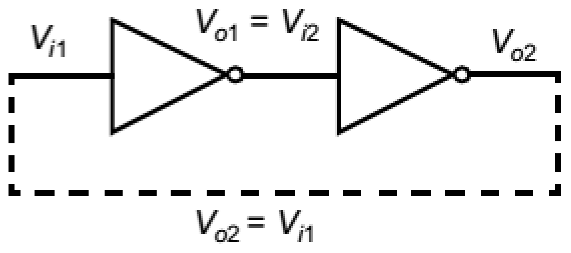
Above figure shows a block diagram of a generic finite state machine(FSM) that consists of combinational logic and registers that hold the system state. The system depicted here belongs to the class of synchronous sequential systems, in which all registers are under control of a single global clock. The outputs of the FSM are a function of the current Input sand the Current State. The Next State is determined based on the Current State and the current Input sand is fed to the inputs of registers. On the rising edge of the clock, the Next State bits are copied to the outputs of the registers (after some propagation delay), and a new cycle begins. The register then ignores changes in the input signals until the next rising edge. In general, registers can be positive edge-triggered(where the input data is copied on the positive edge) or negative edge triggered (where the input data is copied on the negative edge of the clock, as is indicated by a small circle at the clock input).

This chapter discusses the CMOS implementation of the most important sequential building blocks. A variety of choices in sequential primitives and clocking methodologies exist; making the correct selection is getting increasingly important in modern digital circuits, and can have a great impact on performance, power, and/or design complexity. Before embarking on a detailed discussion on the various design options, a revision of the design metrics, and a classification of the sequential elements is necessary.

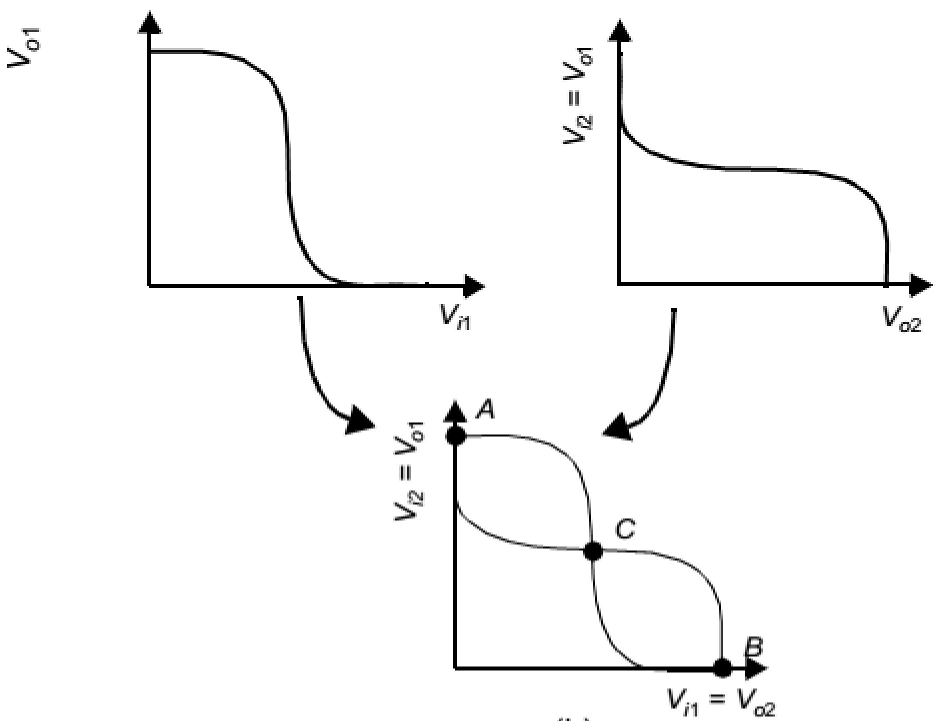
Static Latches and Registers

The Bistability Principle

Static memories use positive feedback to create abistable circuit —a circuit having two stable states that represent 0 and 1. The basic idea is two inverters connected in cascade along with a voltage-transfer characteristic typical of such a circuit.



(a)

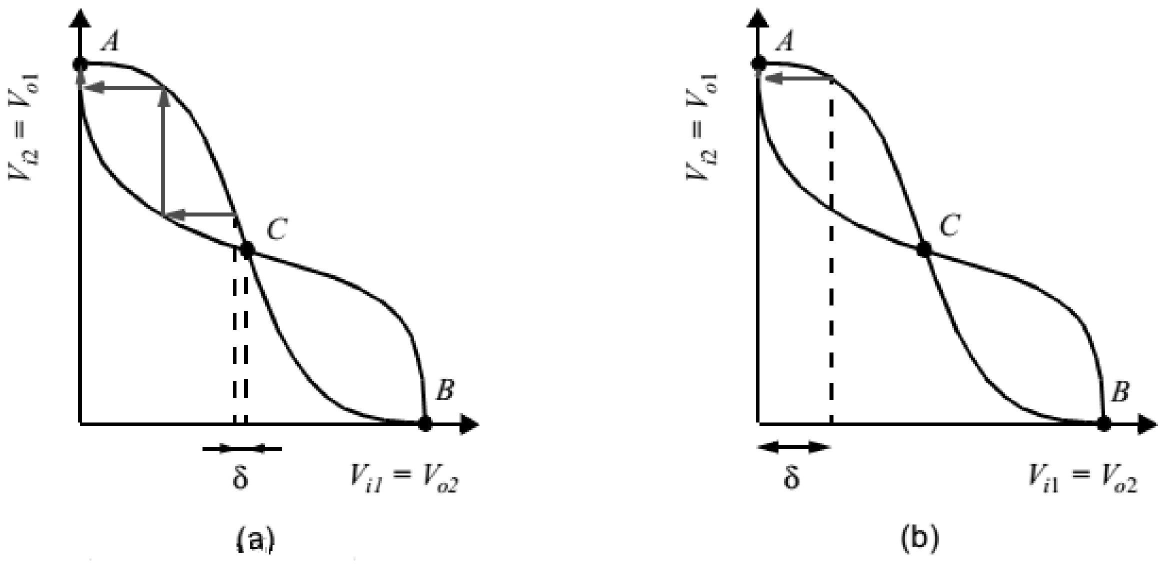


(b)

Also plotted are the VTCs of the first inverter, that is Vo1 versusVi1and the second inverter (Vo2 versusVo1). The latter plot is rotated to accentuate that Vi2 =Vo1 Assume now that the output of the second inverterVo2 is connected to the input of the first Vi1. The resulting circuit has only three possible operation points (A, B and C), as demonstrated on the combined VTC. The following important conjecture is easily proven to be valid:

Under the condition that the gain of the inverter in the transient region is larger than 1, only A and B are stable operation points, and C is a metastable operation point.

Suppose that the cross-coupled inverter pair is biased at point C. A small deviation from this bias point, possibly caused by noise, is amplified and regenerated around the circuit loop. This is a consequence of the gain around the loop being larger than 1.A small deviation δ is applied toVi1(biased in C). This deviation is amplified by the gain of the inverter. The enlarged divergence is applied to the second inverter and amplified once more. The bias point moves away from C until one of the operation points A or B is reached. In conclusion, C is an unstable operation point. Every deviation (even the smallest one) causes the operation point to run away from its original bias. The chance is indeed very small that the cross-coupled inverter pair is biased at C and stays there. Operation points with this property are termed metastable.

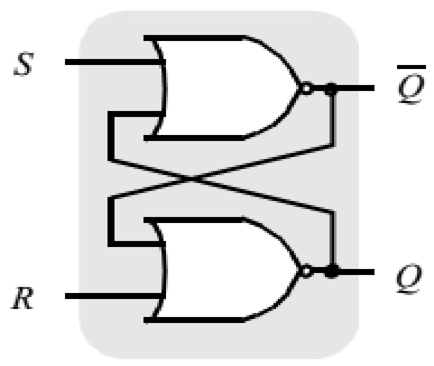


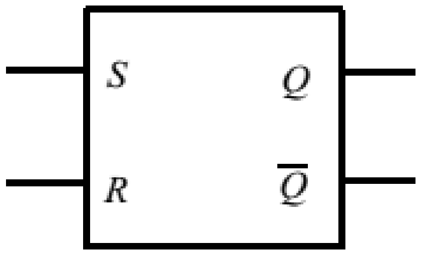
On the other hand, A and B are stable operation points. In these points, the loop gain is much smaller than unity. Even a rather large deviation from the operation point is reduced in size and disappears.Hence the cross-coupling of two inverters results in a bistable circuit, that is, a circuit with two stable states, each corresponding to a logic state. The circuit serves as a memory, storing either a 1 or a 0 (corresponding to positions A and B).

In order to change the stored value, we must be able to bring the circuit from state A to Band vice-versa. Since the precondition for stability is that the loop gain G is smaller than unity, we can achieve this by making A (or B) temporarily unstable by increasing G to a value larger than 1. This is generally done by applying a trigger pulse atVi1 or Vi2. For instance, assume that the system is in position A(Vi1 =0,Vi2 =1).ForcingVi1 to 1 causes both inverters to be on simultaneously for a short time and the loop gain G to be larger than 1. The positive feedback regenerates the effect of the trigger pulse, and the circuit moves to the other state (Bin this case). The width of the trigger pulse need be only a little larger than the total propagation delay around the circuit loop, which is twice the average propagation delay of the inverters.

SR Flip-Flops

The cross-coupled inverter pair shown in the previous section provides an approach to store a binary variable in a stable way. However, extra circuitry must be added to enable control of the memory states. The simplest incarnation accomplishing this is the well knowSR —or set-reset— flip-flop.
This circuit is similar to the cross-coupled inverter pair with NOR gates replacing the inverters. The second input of the NOR gates is connected to the trigger inputs (S and R), that make it possible to force the outputs Q and Qbarto a given state. These outputs are complimentary (except for theSR= 11 state). When both S andRare 0, the flip-flop is in a quiescent state and both outputs retain their value (a NOR gate with one of its input being 0 looks like an inverter, and the structure looks like a cross coupled inverter). If a positive (or 1) pulse is applied to theSinput, theQoutput is forced into the 1 state (withQbar going to 0). Vice versa, a 1 pulse onRresets the flip-flop and theQoutput goes to 0.





(a) Schematic diagram (b) Logic symbol

S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
1	1	0	0

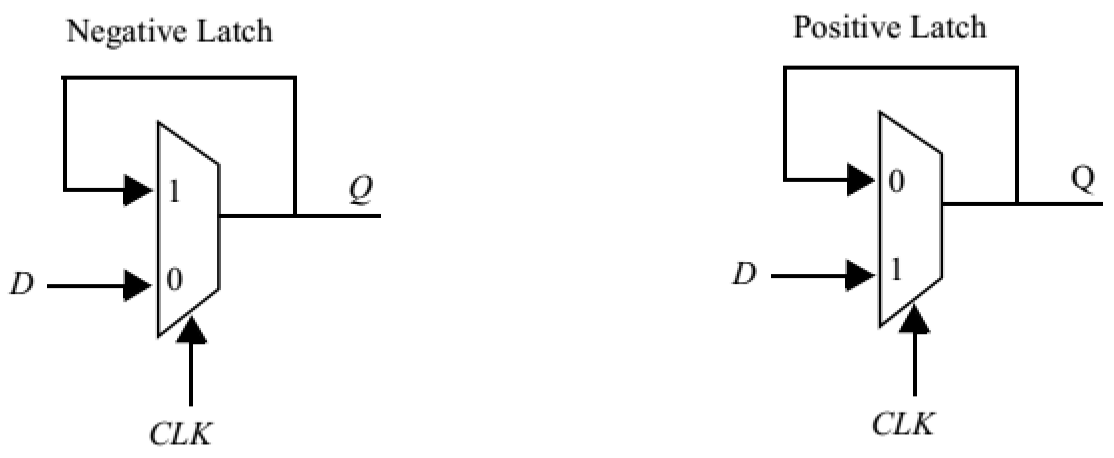
Forbidden State

(c) Characteristic table

These results are summarized in the characteristic table of the flip-flop. The characteristic table is the truth table of the gate and lists the output states as functions of all possible input conditions. When both S and R are high, both Q and Qbar are forced to zero. Since this does not correspond with our constraint that Q and Qbar must be complementary, this input mode is considered to be forbidden. An additional problem with this condition is that when the input triggers return to their zero levels, the resulting state of the latch is unpredictable and depends on whatever input is last to go low.

Multiplexer Based Latches

There are many approaches for constructing latches. One very common technique involves the use of transmission gate multiplexers. Multiplexer based latches can provide similar functionality to the SR latch, but has the important added advantage that the sizing of devices only affects performance and is not critical to the functionality**.**



Negative and positive latches based on multiplexers.

For a negative latch, when the clock signal is low, the input 0 of the multiplexer is selected, and the D input is passed to the output. When the clock signal is high, the input 1 of the multiplexer, which connects to the output of the latch, is selected. The feedback holds the output stable while the clock signal is high. Similarly in the positive latch, the D input is selected when clock is high, and the output is held (using feedback) when clock is low.

A transistor level implementation of a positive latch based on multiplexers is shown below.When CLK is high, the bottom transmission gate is on and the latch is transparent-that is, the D input is copied to the Q output. During this phase, the feedback loop is open since the top transmission gate is off. Unlike the SRFF, the feedback does not have to be overridden to write the memory and hence sizing of transistors is not critical for realizing correct functionality. The number of transistors that the clock touches is important since it has an activity factor of 1. This particular latch implementation is not particularly efficient from this metric as it presents a load of 4 transistors to the CLK signal.

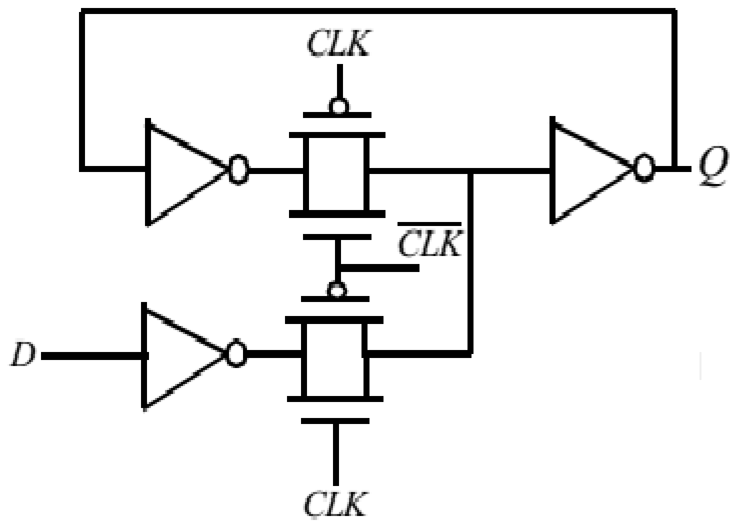
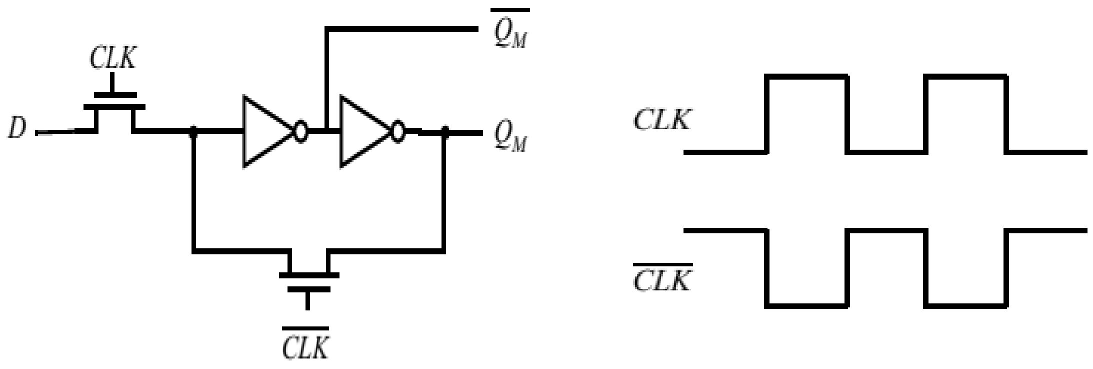


Figure 32: Transistor level implementation ofa positive latch built using transmission gates.

It is possible to reduce the clock load to two transistors by using implement multiplexers using NMOS only pass transistor as shown in Figure 7.13. The advantage of this approach is the reduced clock load of only two NMOS devices. When CLK is high, the latch samples the D input, while a low clock-signal enables the feedback-loop, and puts the latch in the hold mode. While attractive for its simplicity, the use of NMOS only pass transistors results in the passing of a degraded high voltage of VDD-VTn to the input of the first inverter. This impacts both noise margin and the switching performance, especially in the case of low values of VDD and high values of VTn. It also causes static power dissipation in first inverter, as already pointed out in Chapter 6. Since the maximum input-voltage to the inverter equals VDD-VTn, the PMOS device of the inverter is never turned off, resulting is a static current flow.



.

(a) Schematic diagram (b) Non overlapping clocks

Fig: Multiplexer based NMOS latch using NMOS only pass transistors for multiplexers

Master-Slave Based Edge Triggered Register

The most common approach for constructing an edge-triggered register is to use a master-slave configuration. The register consists of cascading a negative latch (master stage) with a positive latch (slave stage). A multiplexer based latch is used in this particular implementation, though any latch can be used to realize the master and slave stages. On the low phase of the clock, the master stage is transparent and the D input is passed to the master stage output, QM. During this period, the slave stage is in the hold mode, keeping its previous value using feedback. On the rising edge of the clock, the master slave stops sampling the input, and the slave stage starts sampling. During the high phase of the clock, the slave stage samples the output of the master stage (QM), while the master stage remains in a hold mode. Since QM is constant during the high phase of the clock, the output Q makes only one transition per cycle. The value of Q is the value of D right before the rising edge of the clock, achieving the positive edge-triggered effect. A negative edge-triggered register can be constructed using the same principle by simply switching the order of the positive and negative latch (i.e., placing the positive latch first).

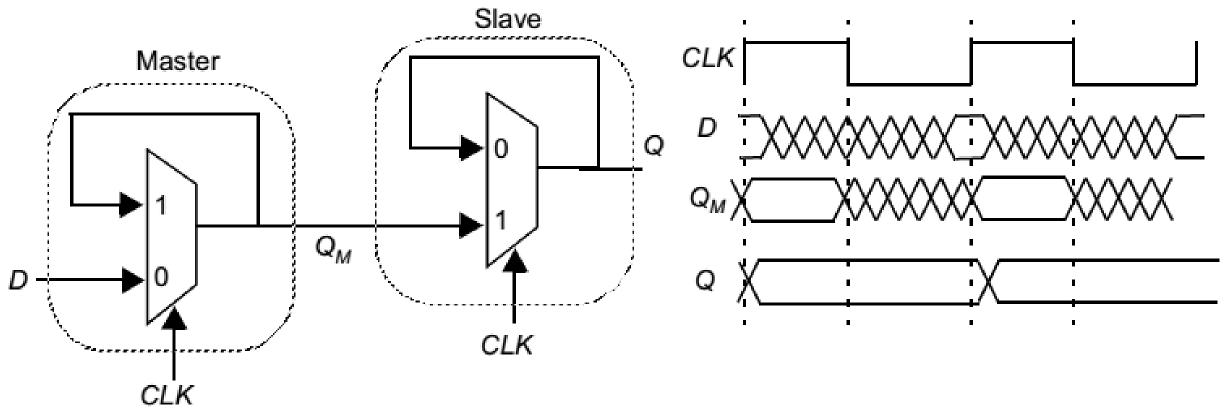


Fig: Positive edge-triggered register based on a master-slave configuration.

A complete transistor level implementation of a the master-slave positive edge-triggered register is shown. The multiplexer is implemented using transmission gates as discussed in the previous section. When clock is low (CLK=0).T1 is on and T2 is off, and the D input is sampled onto node QM. During this period, T3 is off and T4 is on and the cross-coupled inverters (I5, I6) holds the state of the slave latch. When the clock goes high, the master stage stops sampling the input and goes into aholdmode.T1 is off and T2 is on, and the cross coupled inverters I3 and I4 holds the state of QM. Also, T3 is on and T4 is off, and QM is copied to the output Q.

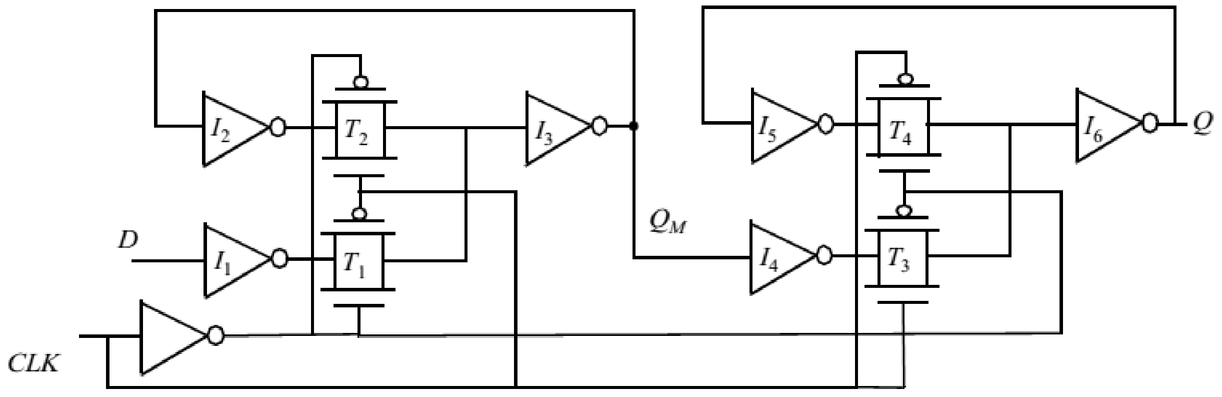


Figure 33: Transistor-level implementation of a master-slave postive edge-triggered register using multiplexers.

DYNAMIC LATCHES AND REGISTERS

Storage in a static sequential circuit relies on the concept that a cross-coupled inverter pair produces a bistable element and can thus be used to memorize binary values. This approach has the useful property that a stored value remains valid as long as the supply voltage is applied to the circuit, hence the name static. The major disadvantage of the static gate, however, is its complexity. When registers are used in computational structures that are constantly clocked such as pipelined data path, the requirement that the memory should hold state for extended periods of time can be significantly relaxed.

This results in a class of circuits based on temporary storage of charge on parasitic capacitors. The principle is exactly identical to the one used in dynamic logic — charge stored on a capacitor can be used to represent a logic signal. The absence of charge denotes a 0, while its presence stands for a stored 1. No capacitor is ideal, unfortunately, and some charge leakage is always present. A stored value can hence only be kept for a limited amount of time, typically in the range of milliseconds. If one wants to preserve signal integrity, a periodic refresh of its value is necessary. Hence the name dynamic storage. Reading the value of the stored signal from a capacitor without disrupting the charge requires the availability of a device with a high input impedance.

Dynamic Transmission-Gate Based Edge-triggered Registers

A fully dynamic positive edge-triggered register based on the master-slave concept is shown.When CLK= 0, the input data is sampled on storage node 1, which has an equivalent capacitance of C1 consisting of the gate capacitance of I1 , the junction capacitance of T1, and the overlap gate capacitance of T1. During this period, the slave stage is in a hold mode, with node 2 in a high-impedance (floating) state. On the rising edge of clock, the transmission gate T2 turns on, and the value sampled on node 1 right before the rising edge propagates to the output Q(note that node 1 is stable during the high phase of the clock since the first transmission gate is turned off). Node 2 now stores the inverted version of node 1. This implementation of an edge-triggered register is very efficient as it requires only 8 transistors. The sampling switches can be implemented using NMOS-only pass transistors, resulting in an even-simpler 6 transistor implementation. The reduced transistor count is attractive for high-performance and low-power systems.

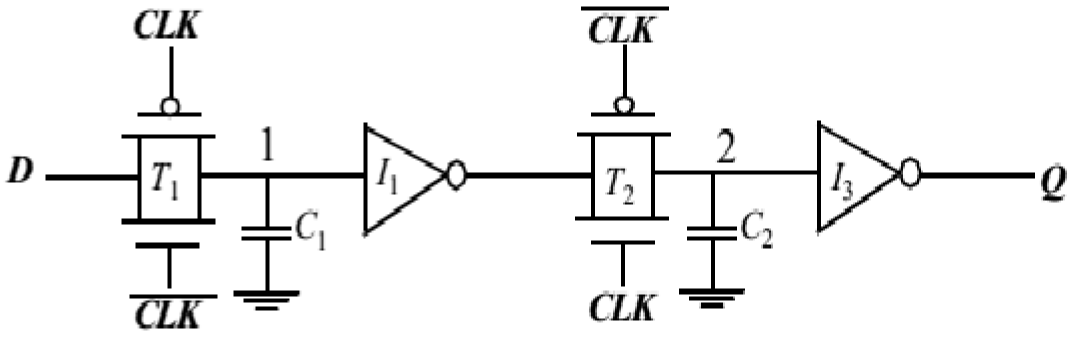


Figure 33: Dynamic edge-triggered register.

The set-up time of this circuit is simply the delay of the transmission gate, and corresponds to the time it takes node 1 to sample the D input. The hold time is approximately zero, since the transmission gate is turned off on the clock edge and further inputs changes are ignored. The propagation delay(tc-q) is equal to two inverter delays plus the delay of the transmission gate T2.

One important consideration for such a dynamic register is that the storage nodes (i.e., the state) has to be refreshed at periodic intervals to prevent a loss due to charge leakage, due to diode leakage as well as sub-threshold currents. In data path circuits, the refresh rate is not an issue since the registers are periodically clocked, and the storage nodes are constantly updated.

Clock overlap is an important concern for this register. Consider the clock waveforms shown.

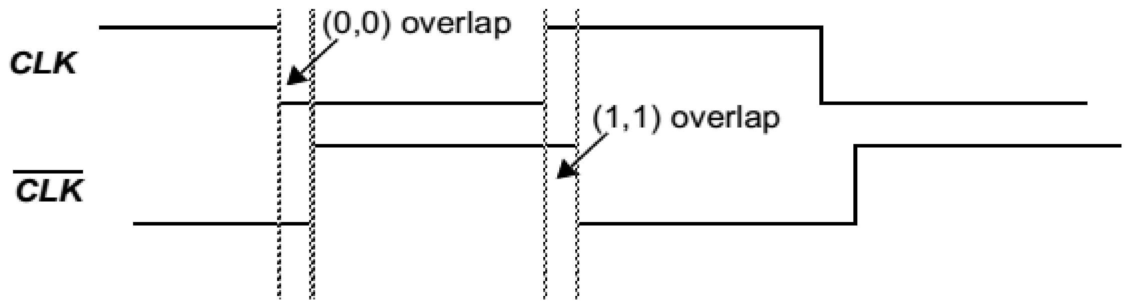


Figure 35: Impact of non-overlapping clocks.

During the 0-0 overlap period, the NMOS of T1 and the PMOS ofT2 are simultaneously on, creating a direct path for data to flow from the D input of the register to the Q output. This is known as a race condition. The output Q can change on the falling edge if the overlap period is large — obviously an undesirable effect for a positive edge-triggered register. The same is true for the 1-1 overlap region, where an input-output path exists through the PMOS of T1 and the NMOS ofT2. The latter case is taken care off by enforcing a hold time constraint. That is, the data must be stable during the high-high overlap period. The former situation (0-0 overlap) can be addressed by making sure that there is enough delay between the D input and node 2 ensuring that new data sampled by the master stage does not propagate through to the slave stage. Generally the built in single inverter delay should be sufficient and the overlap period constraint is given as:

$$t_{overlap0-0} < t_{T1} + t_{I1} + t_{T2}$$

Similarly, the constraint for the 1-1 overlap is given as:

$$t_{hold} > t_{overlap1-1}$$

C^{2}MOS Dynamic Register: A Clock Skew Insensitive Approach

The C^{2}MOS Register

The register operates in two phases.

1. CLK =0: The first tri-state driver is turned on, and the master stage acts as an inverter sampling the inverted version of D on the internal node X. The master stage is in the evaluation mode. Meanwhile, the slave section is in a high-impedance mode, or in ahold mode. Both transistors M7 andM8 are off, decoupling the output from the input. The output Q retains its previous value stored on the output capacitor $C_{L2.}$

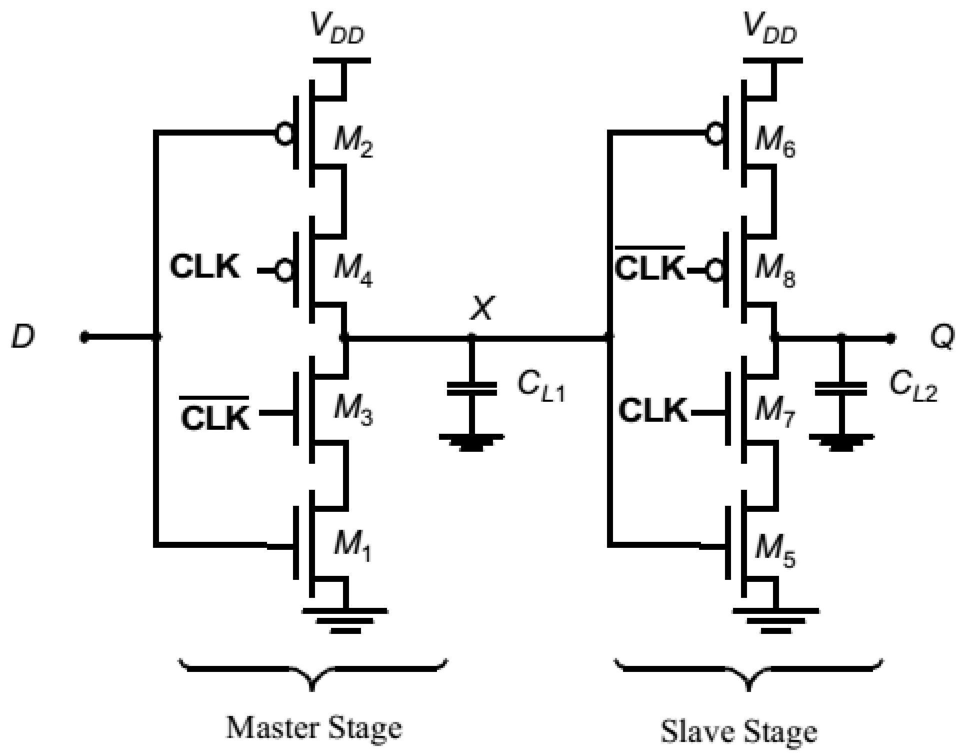


Figure 36: C^2MOS master-slave positive edge-triggered register.

2.The roles are reversed when CLK = 1: The master stage section is in hold mode (M3-M4 off), while the second section evaluates (M7-M8 on). The value stored onCL1 ropagates to the output node through the slave stage which acts as an inverter.

The overall circuit operates as a positive edge-triggered master-slave register — very similar to the transmission-gate based register presented earlier. However, there is an important difference:

A C^2MOS register with CLK-~~CLK~~ clocking is insensitive to overlap, as long as the rise and fall times of the clock edges are sufficiently small.

Dual-edge Triggered Registers

It is also possible to design sequential circuits that sample the input on both edges. The advantage of this scheme is that a lower frequency clock (half of the original rate) is distributed for the same functional throughput, resulting in power savings in the clock distribution network.

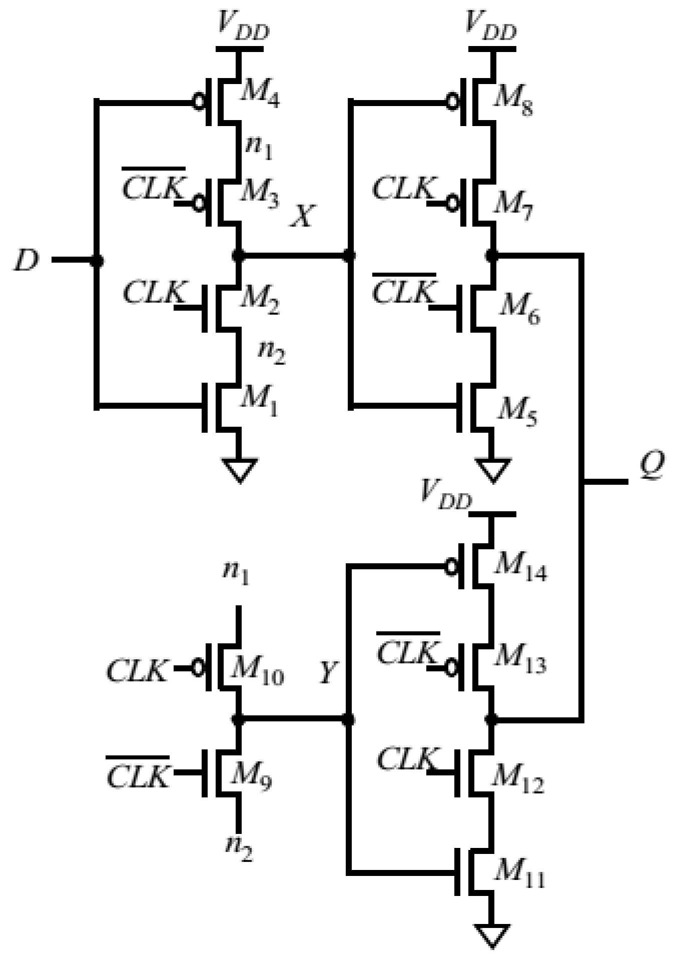


Figure 37: C^{2}MOS based dual-edge triggered register

This is a modification of the C^{2}MOS register to enable sampling on both edges. It consists of two parallel master-slavebased edge-triggered registers, whose outputs are multiplexed using the tri-state drivers.

When clock is high, the positive latch composed of transistorsM1-M4 is sampling the inverted D input on node X. Node Y is held stable, since devices M9 and M10 are turned off. On the falling edge of the clock, the top slave latch M5-M8 turns on, and drives the inverted value of X to the Q output. During the low phase, the bottom master latch (M1,M4, M9, M10) is turned on, sampling the inverted D input on node Y. Note that the devices M1 and M4 are reused, reducing the load on the D input. On the rising edge, the bottom slave latch conducts, and drives the inverted version of Y on node Q. Data hence changes on both edges. Note that the slave latches operate in a complementary fashion — this is, only one of them is turned on during each phase of the clock.

True Single-Phase Clocked Register (TSPCR)

In the two-phase clocking schemes described above, care must be taken in routing the two clock signals to ensure that overlap is minimized. While the C2MOS provides a skew-tolerant solution, it is possible to design registers that only use a single phase clock. The True Single-Phase Clocked Register (TSPCR) proposed by Yuan and Svensson uses a single clock(without an inverse clock).

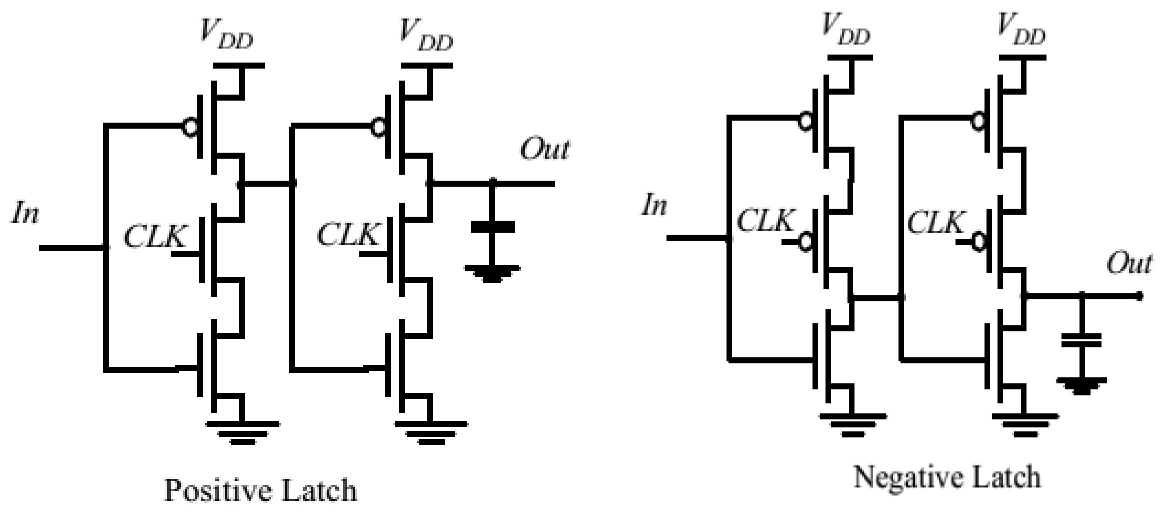


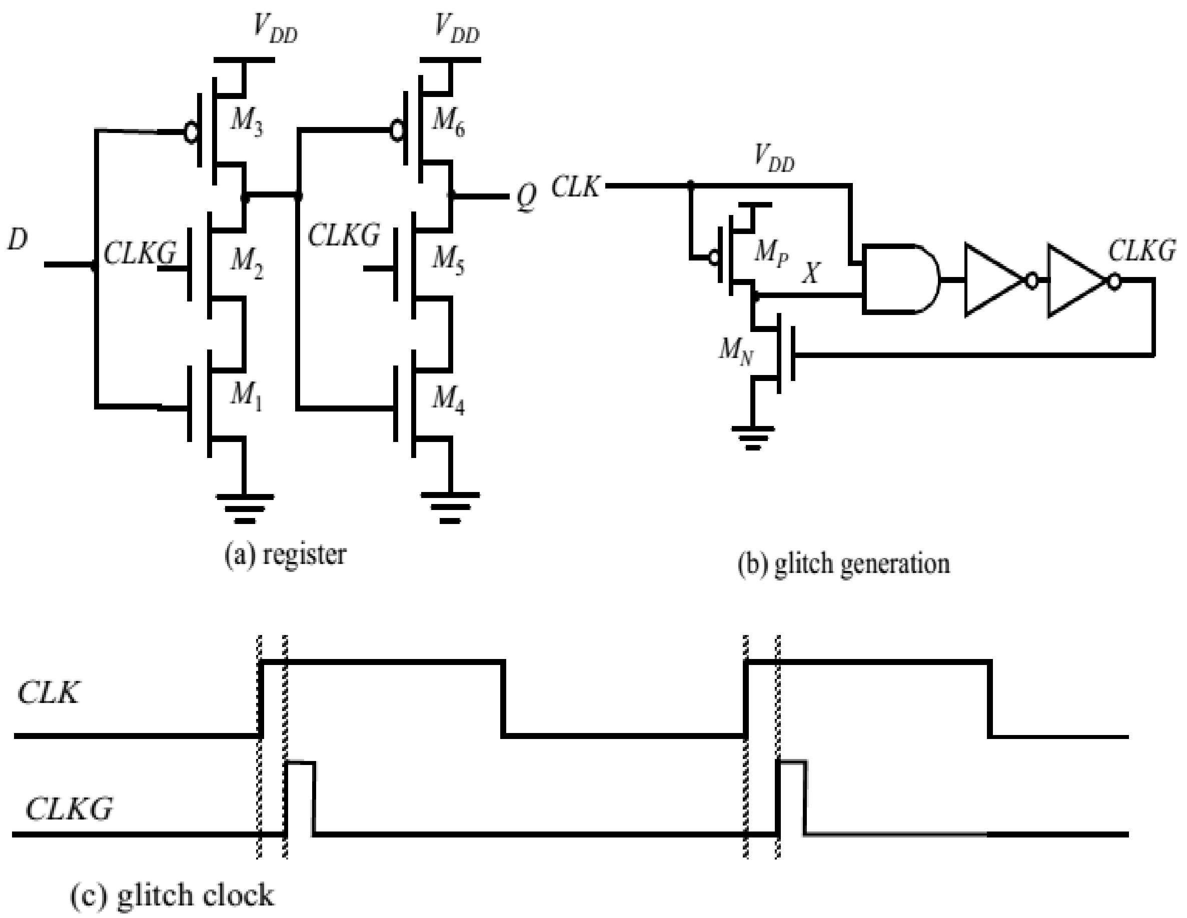
Figure 38: True Single Phase Latches.

For the positive latch, when CLK is high, the latch is in the transparent mode and corresponds to two cascaded inverters; the latch is non-inverting, and propagates the input to the output. On the other hand, when CLK= 0, both inverters are disabled, and the latch is in hold-mode. Only the pull-up networks are still active, while the pull-down circuits are deactivated. As a result of the dual-stage approach, no signal can ever propagate from the input of the latch to the output in this mode. A register can be constructed by cascading positive and negative latches. The clock load is similar to a conventional transmission gate register, or C^2MOS register. The main advantage is the use of a single clock phase. The disadvantage is the slight increase in the number of transistors — 12 transistors are required.

TSPC offers an additional advantage: the possibility of embedding logic functionality into the latches. This reduces the delay overhead associated with the latches. In addition to performing the latching function. While the set-up time of this latch has increased over the one, the overall performance of the digital circuit (that is, the clock period of a sequential circuit) has improved: the increase inset-up time is typically smaller than the delay of an AND gate. This approach of embedding logic into latches has been used extensively in the design of the EV4 DEC Alpha microprocessor and many other high performance processors.

PULSE BASED REGISTERS

A fundamentally different approach for constructing a register uses pulse signals. The idea is to construct a short pulse around the rising (or falling) edge of the clock. This pulse acts as the clock input to a latch (e.g., a TSPC flavor), sampling the input only in a short window. Race conditions are thus avoided by keeping the opening time (i.e, the transparent period) of the latch very short. The combination of the glitch generation circuitry and the latch results in a positive edge-triggered register.



.

Figure 39: Glitch latch - timing generation and register.

When CLK= 0, node X is charged up to VDD(MN is off since CLKG is low). On the rising edge of the clock, there is a short period of time when both inputs of the AND gate are high, causing CLKG to go high. This in turn activates MN, pulling X and eventually CLKG low.The length of the pulse is controlled by the delay of the AND gate and the two inverters. Note that there exists also a delay between the rising edges of the input clock (CLK) and the glitch clock (CLKG)— alsoequaltothedelayoftheANDgateandthetwoinverters.Ifeveryregisteronthechip uses the same clock generation mechanism, this sampling delay does not matter. However, process variations and load variations may cause the delays through the glitch clock circuitry to be different.

If set-up time and hold time are measured in reference to the rising edge of the glitch clock, the set-up time is essentially zero, the hold time is equal to the length of the pulse (if the contamination delay is zero for the gates), and the propagation delay(tc-q) equals two gate delays. The advantage of the approach is the reduced clock load and the small number of transistors required. The glitch-generation circuitry can be amortized over multiple register bits. The disadvantage is a substantial increase in verification complexity. This has prevented a wide-spread use. They do however provide an alternate approach to conventional schemes, and have been adopted in some high performance processors

SENSE AMPLIFIER BASED REGISTER

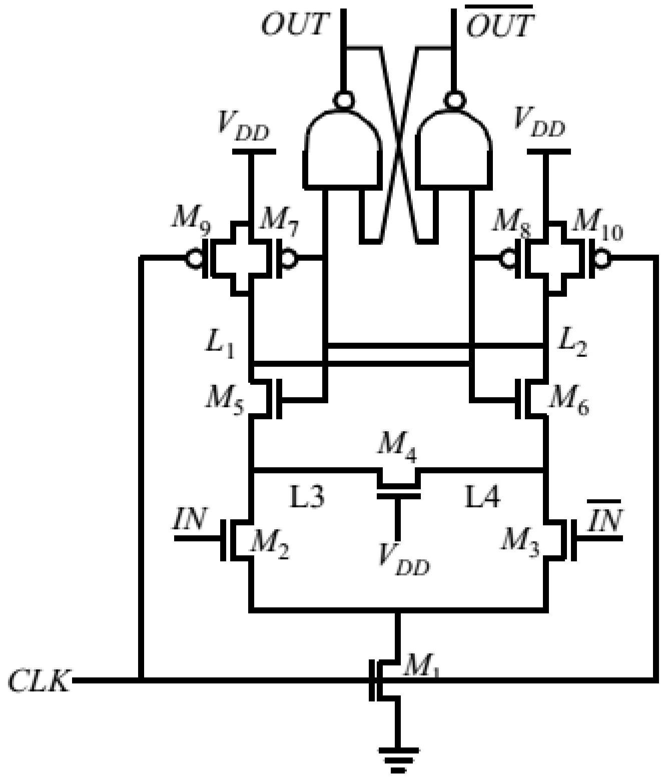


Figure 40: Positive edge-triggered register based on sense-amplifier.

Sense amplifier circuits accept small input signals and amplify them to generate rail-to-rail swings. As we will see, sense amplifier circuits are used extensively in memory cores and in low swing bus drivers to amplify small voltage swings present in heavily loaded wires. There are many techniques to construct these amplifiers,with the use of feedback (e.g., cross-coupled inverters) being one common approach. The circuit shown uses a precharged front-end amplifier that samples the differential input signal on the rising edge of the clock signal. The outputs of front-end are fed into a NAND cross-coupled SR FF that holds the data and gurantees that the differential outputs switch only once per clock cycle. The differential inputs in this implementation don't have to have rail-to-rail swing and hence this register can be used as a receiver for a reduced swing differential bus.

The core of the front-end consists of a cross-coupled inverter (M5-M8) whose outputs (L1andL2) are precharged using devices M9andM10 during the low phase of the clock. As a result, PMOS transistorsM7 andM8 to be turned off and the NAND FF is holding its previous state. Transistor M1is similar to an evaluate switch in dynamic circuits and is turned off ensuring that the differential inputs don't affect the output during the low phase of the clock. On the rising edge of the clock, the evaluate transistor turns on and the differential input pair (M2andM3) is enabled, and the difference between the input signals is amplified on the output nodes onL1 andL2. The cross-coupled inverter pair flips to one of its the stable states based on the value of the inputs. For example, if IN is 1, L1 is pulledto0,andL2 remains at VDD. Due to the amplifying properties of the input stage, it is not necessary for the input to swing all the way up to VDD and enables the use of low swing signaling on the input wires.

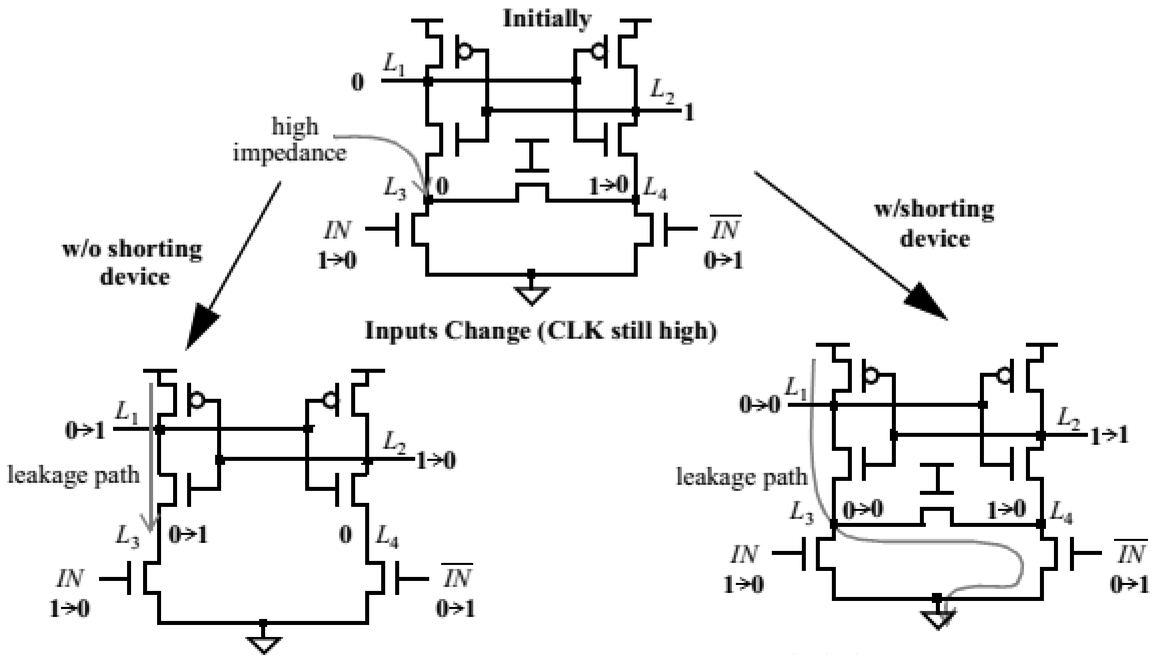


Figure 41: The need for the shorting transistor M4

The shorting transistor,M4, is used to provide a DC leakage path from either node L3,orL4, to ground. This is necessary to accommodate the case where the inputs change their value after the positive edge of CLK has occurred, resulting in eitherL3orL4 being left in a high-impedance state with a logical low voltage level stored on the node. Without the leakage path that node would be susceptible to charging by leakage currents. The latch could then actually change state prior to the next rising edge of CLK!

NORA-CMOS—A Logic Style for Pipelined Structures

The latch-based pipeline circuit can also be implemented using C^2MOS latches. The operation is similar to the one discussed above. This topology has one additional, important property:

A C^2MOS-based pipelined circuit is race-free as long as all the logic functions F(implemented using static logic) between the latches are noninverting.

The reasoning for the above argument is similar to the argument made in the construction of a C^2MOS register. During a (0-0) overlap between CLK and CLK, al l C^2MOS latches, simplify to pure pull-up networks. The only way a signal can race from stage to stage under this condition is when the logic function F is inverting, where F is replaced by a single, static CMOS inverter. Similar considerations are valid for the (1-1) overlap.

Based on this concept, a logic circuit style called NORA-CMOS was conceived It combines C^2MOS pipeline registers and NOR A dynamic logic function blocks. Each module consists of a block of combinational logic that can be a mixture of static and dynamic logic, followed by a C^2MOS latch. Logic and latch are clocked in such a way that both are simultaneously in either evaluation, or hold (precharge) mode. A block that is in evaluation during CLK= 1 is called a CLK-module, while the inverse is called a ~~CLK~~-module.

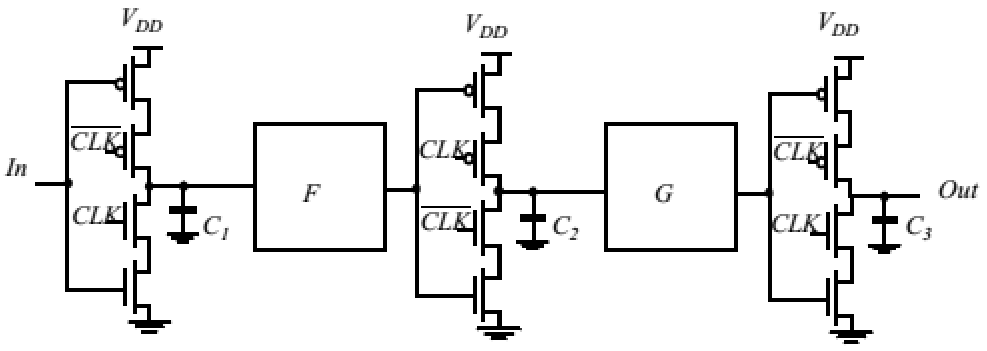


Figure 7.42 Pipelined datapath using C^{2}MOS latches.

A NORA data path consists of a chain of alternating CLK and ~~CLK~~ modules. While one class of modules is precharging with its output latch in hold mode, preserving the previous output value, the other class is evaluating. Data is passed in a pipelined fashion from module to module. NORA offers designers a wide range of design choices. Dynamic and static logic can be mixed freely, and both CLKp and CLKn dynamic blocks can be used in cascaded or in pipelined form. With this freedom of design, extra inverter stages, as required in DOMINO-CMOS, are most often avoided.

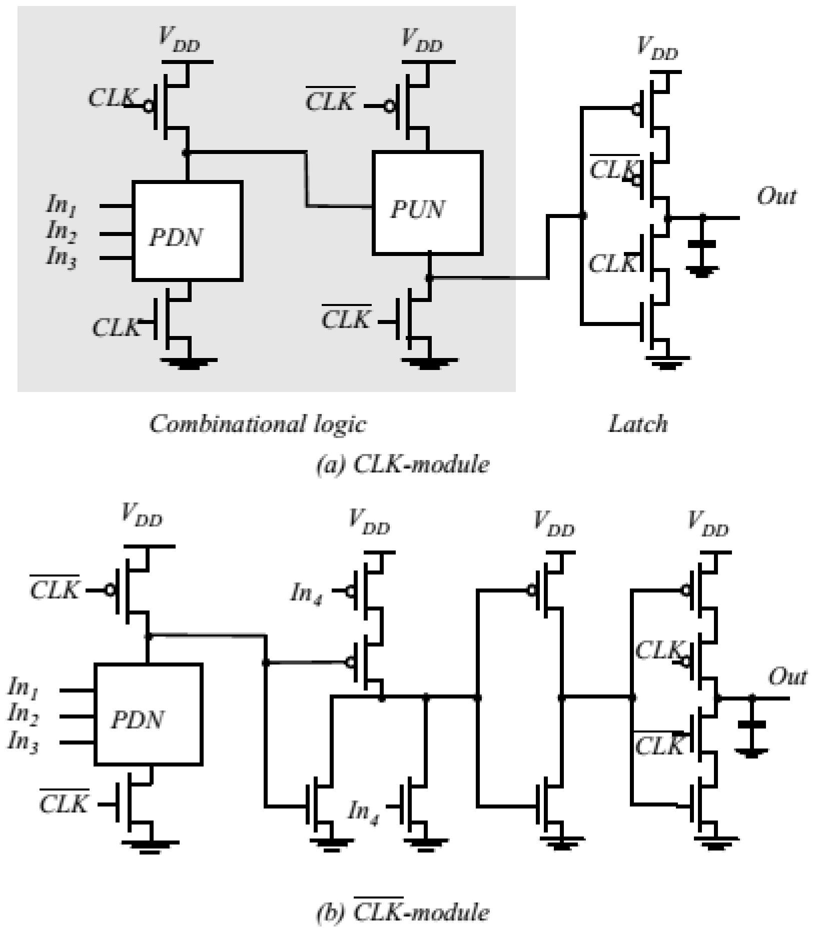


Figure 42: Examples of NORA CMOS Modules

Semiconductor Memories:

Semiconductor based electronics is the foundation to the information technology society we live in today. Ever since the first transistor was invented way back in 1948, the semiconductor industry has been growing at a tremendous pace. Semiconductor memories and microprocessors are two major fields, which are benefited by the growth in semiconductor technology.

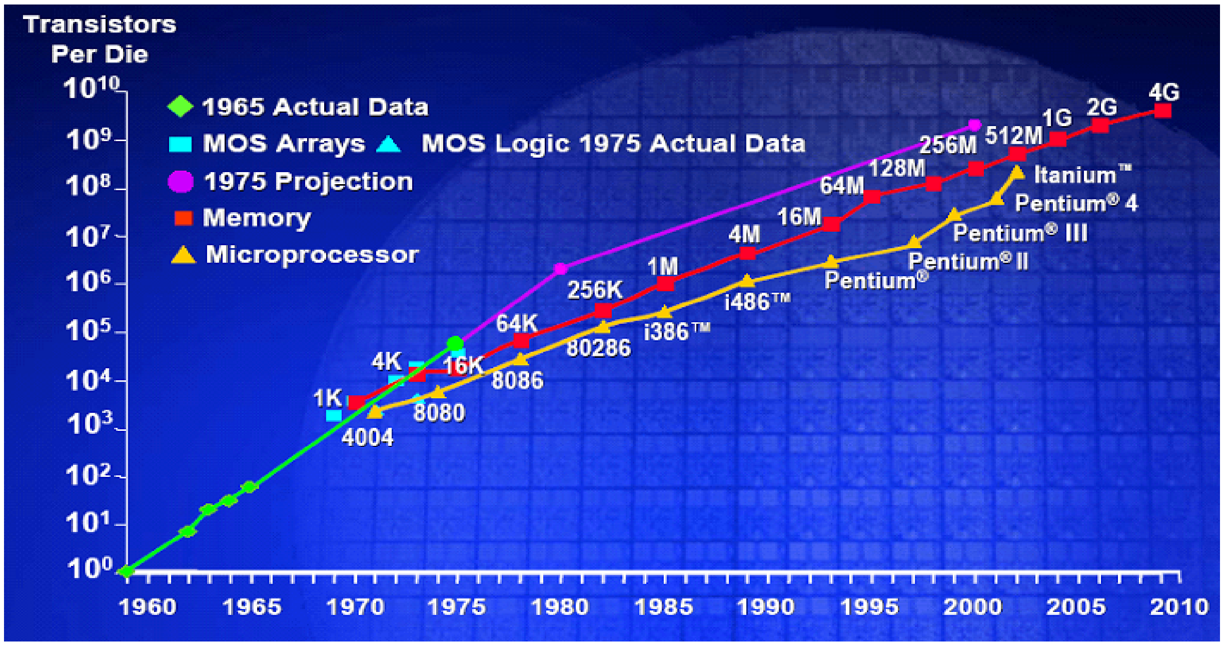


Fig: Increasing memory capacity over the years

The technological advancement has improved performance as well as packing density of these devices over the years Gordon Moore made his famous observation in 1965, just four years after the first planar integrated circuit was discovered. He observed an exponential growth in the number of transistors per integrated circuit in which the number of transistors nearly doubled every couple of years. This observation, popularly known as Moore's Law, has been maintained and still holds true today. Keeping up with this law, the semiconductor memory capacity also increases by a factor of two every year.

Memory Classification

Size: Depending upon the level of abstraction, different means are used to express the size of the memory unit. A circuit designer usually expresses memory in terms of bits, which are equivalent to the number of individual cells need to store the data. Going up one level in the hierarchy to the chip design level, it is common to express memory in terms of bytes, which is a group of 8 bits. And on a system level, it can be expressed in terms of words or pages, which are in turn collection of bytes.

Function: Semiconductor memories are most often classified on the basis of access patterns, memory functionality and the nature of the storage mechanism. Based on the access patterns, they can be classified into random access and serial access memories. A random access memory can be accessed for read/write in a random fashion. On the other hand, in serial access memories, the data can be accessed only in a serial fashion. FIFO (First In First Out) and LIFO (Last In Last Out) are examples of serial memories. Most of the memories fall under the random access types.

Based on their functionalities, memory can be broadly classified into Read/Write memories and Read-only memories. As the name suggests, Read/Write memory offers both read and write operations and hence is more flexible. **SRAM (Static RAM)** and **DRAM (Dynamic RAM)** come under this category. A Read-only memory on the other hand encodes the information into the circuit topology. Since the topology is hardwired, the data cannot be modified; it can only be read. However **ROM** structures belong to the class of the **nonvolatile memories**. Removal of the supply voltage does not result in a loss of the stored data. Examples of such structures include **PROMs, ROMs and PLDs**. The most recent entry in the filed are memory modules that can be classified as nonvolatile, yet offer both read and write functionality. Typically, their write operation takes substantially longer time than the read operation. An **EPROM, EEPROM** and **Flash memory** fall under this category.

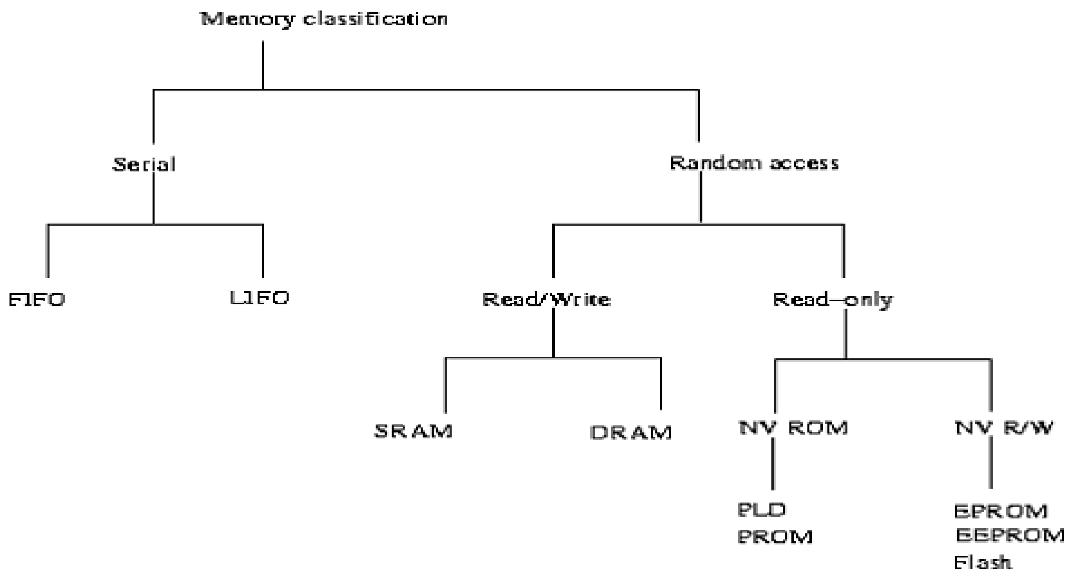
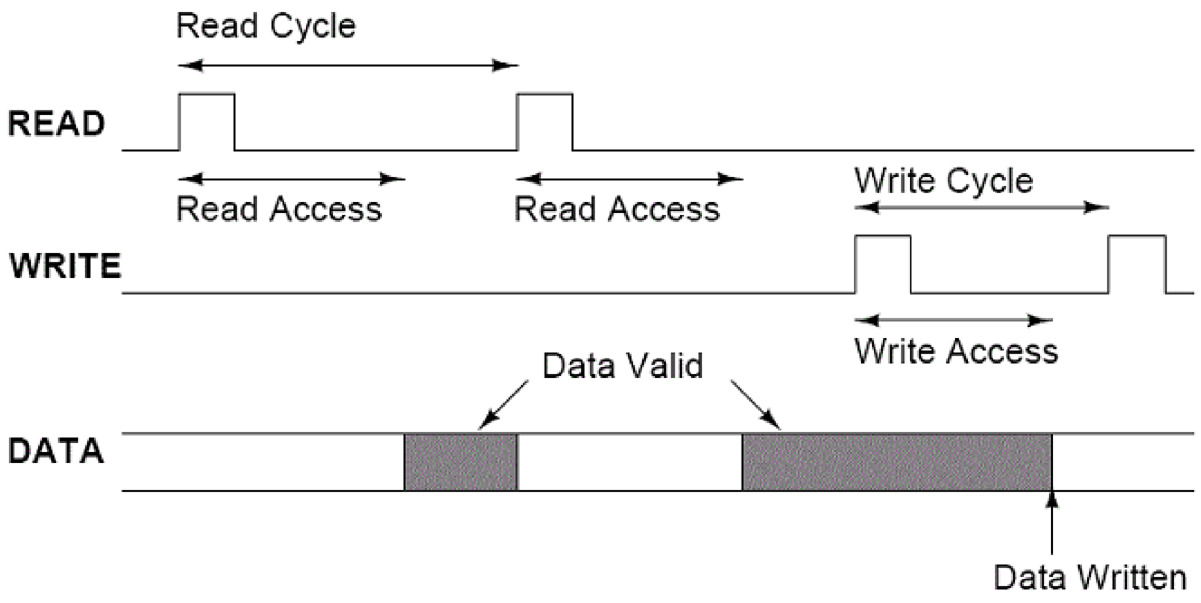


Figure 44: Classification of memories

Timing Parameters: The time it takes to retrieve data from the memory is called the readaccess time. This is equal to the delay between the read request and the moment the data is available at the output. Similarly, write-access time is the time elapsed between a write request and the final writing of the input data into the memory. Finally, there is another important parameter, which is the cycle time (read or write), which is the minimum time required between two successive read or write cycles. This time is normally greater than the access time.



Memory Architecture and Building Blocks

The straightforward way of implementing a N-word memory is to stack the words in a linear fashion and select one word at a time for reading or writing operation by means of a select bit. Only one such select signal can be high at a time. Though this approach is quite simple, one runs into a number of problems when trying to use it for larger memories. The number of interface pins in the memory module varies linearly with the size of the memory and this can easily run into huge values.

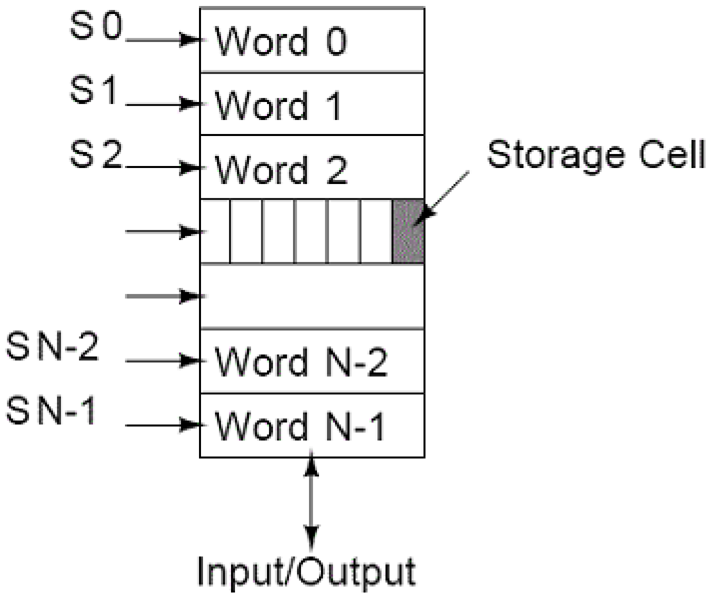


Figure 45: Basic Memory Organization

To overcome this problem, the address provided to the memory module is generally encoded. A decoder is used internally to decode this address and make the appropriate select line high. With 'k' address pins, 2 K number of select pins can be driven and hence the number of interface pins will get reduced by a factor of log2N.

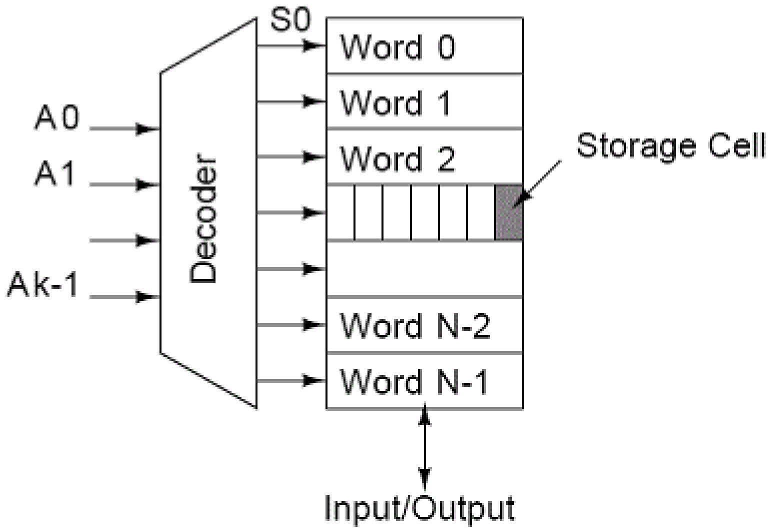


Figure 46: Memory with decoder logic

Though this approach resolves the select problem, it does not address the issues of the memory aspect ratio. For an N-word memory, with a word length of M, the aspect ratio will be nearly N:M, which is very difficult to implement for large values of N. Also such sort of a design slows down the circuit very much. This is because,the vertical wires connecting the storage cells to the inputs/outputs become excessively long. To address this problem, memory arrays are organized so that the vertical and horizontal dimensions are of the same order of magnitude, making the aspect ratio close to unity. To route the correct word to the input/output terminals, an extra circuit called column decoder is needed. The address word is partitioned into column address (A0 to AK-1) and row address (AK-1 to AL-1). The row address enables one row of the memory for read/write, while the column address picks one particular word from the selected row.

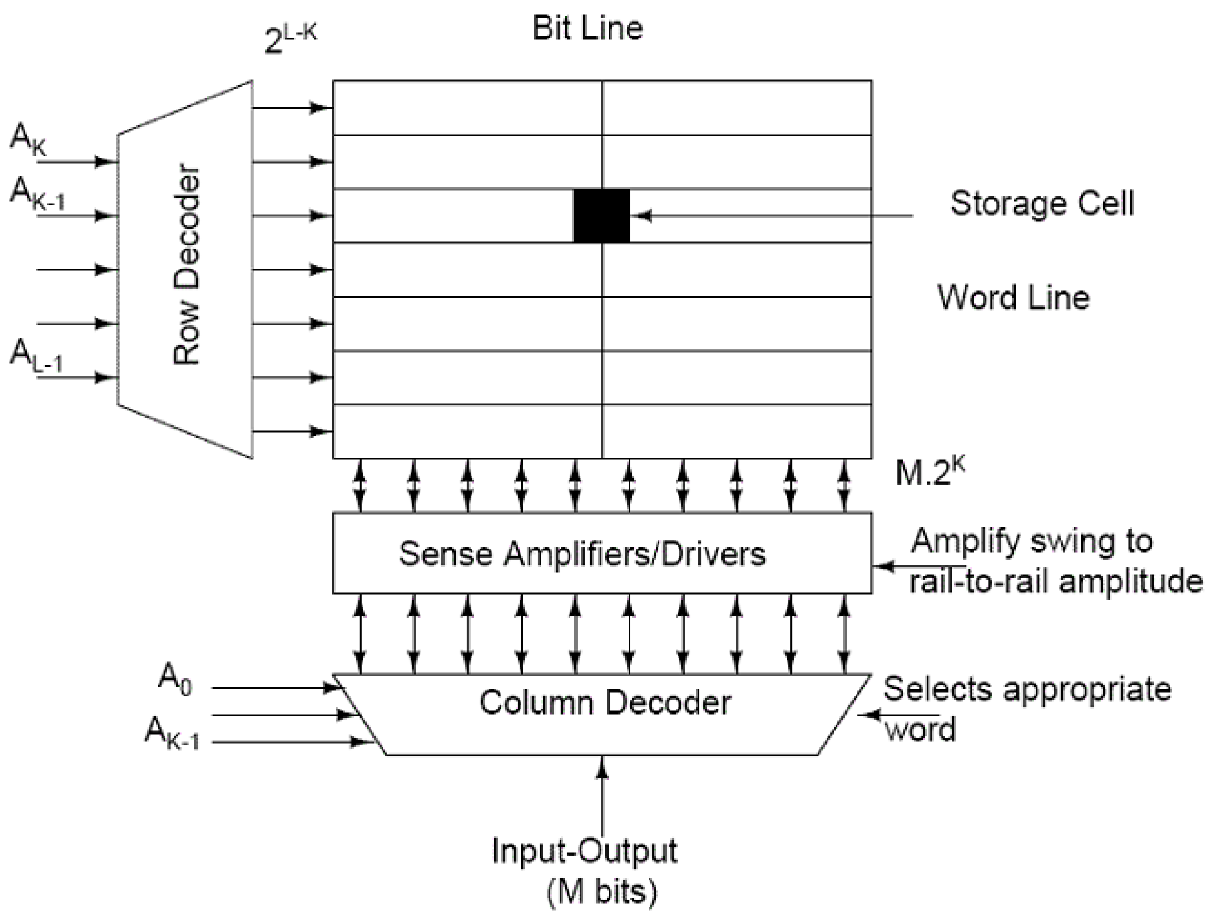


Figure 47: Memory with row and column decoders

Static and Dynamic RAMs

RAMs are of two types, static and dynamic. Circuits similar to basic D flip-flop are used to construct static RAMs (SRAMs) internally. A typical SRAM cell consists of six transistors which are connected in such a way as to form a regenerative feedback. In contrast to DRAM, the information stored is stable and does not require clocking or refresh cycles to sustain it. Compared to DRAMs, SRAMs are much faster having typical access times in the order of a few nanoseconds. Hence SRAMs are used as level 2 cache memory.

Dynamic RAMs do not use flip-flops, but instead are an array of cells, each containing a transistor and a tiny capacitor. '0's and '1's can be stored by charging or discharging the capacitors. The electric charge tends to leak out and hence each bit in a DRAM must be refreshed every few milliseconds to prevent loss of data. This requires external logic to take care of refreshing which makes interfacing of DRAMs more complex than SRAMs. This disadvantage is compensated by their larger capacities. A high packing density is achieved since DRAMs require only one transistor and one capacitor per bit. This makes them ideal to build main memories. But DRAMs are slower having delays in the order tens of nanoseconds. Thus the combination of static RAM cache and a dynamic RAM main memory attempts to combine the good properties of each.

Static Random Access Memory (SRAM)

SRAM Basics

The memory circuit is said to be static if the stored data can be retained indefinitely, as long as the power supply is on, without any need for periodic refresh operation. The data storage cell, i.e., the one-bit memory cell in the static RAM arrays, invariably consists of a simple latch circuit with two stable operating points. Depending on the preserved state of the two inverter latch circuit, the data being held in the memory cell will be interpreted either as logic '0' or as logic '1'. To access the data contained in the memory cell via a bit line, we need atleast one switch, which is controlled by the corresponding word line.

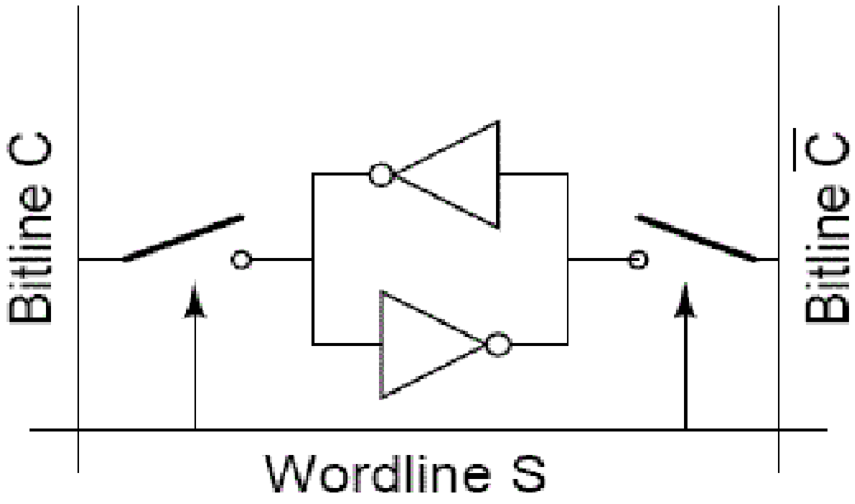


Figure 48: SRAM Cell

CMOS SRAM Cell

A low power SRAM cell may be designed by using cross-coupled CMOS inverters. The most important advantage of this circuit topology is that the static power dissipation is very small; essentially, it is limited by small leakage current. Other advantages of this design are high noise immunity due to larger noise margins, and the ability to operate at lower power supply voltage. The major disadvantage of this topology is larger cell size. The memory cell consists of simple CMOS inverters connected back to back, and two access transistors. The access transistors are turned on whenever a word line is activated for read or write operation, connecting the cell to the complementary bit line columns.

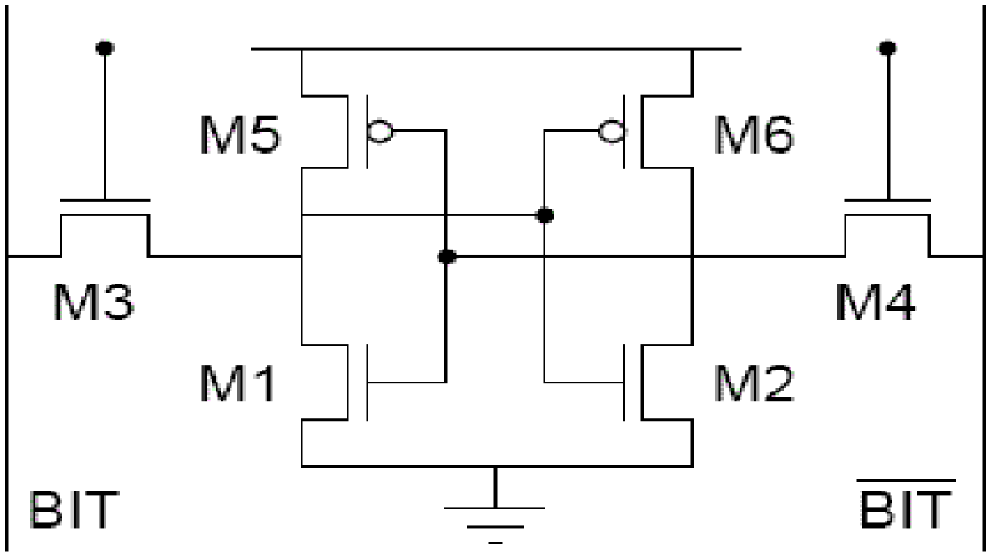


Figure 49: Full CMOS SRAM cell

CMOS SRAM Cell Design

To determine W/L ratios of the transistors, a number of design criteria must be taken into consideration. The two basic requirements, which dictate W/L ratios, are that the data read operation should not destroy the stored information in the cell. The cell should allow stored information modification during write operation. In order to consider operations of SRAM, we have to take into account, the relatively large parasitic column capacitance $C_{bit}$and $C_{\overline{bit}}$column pull-up transistors

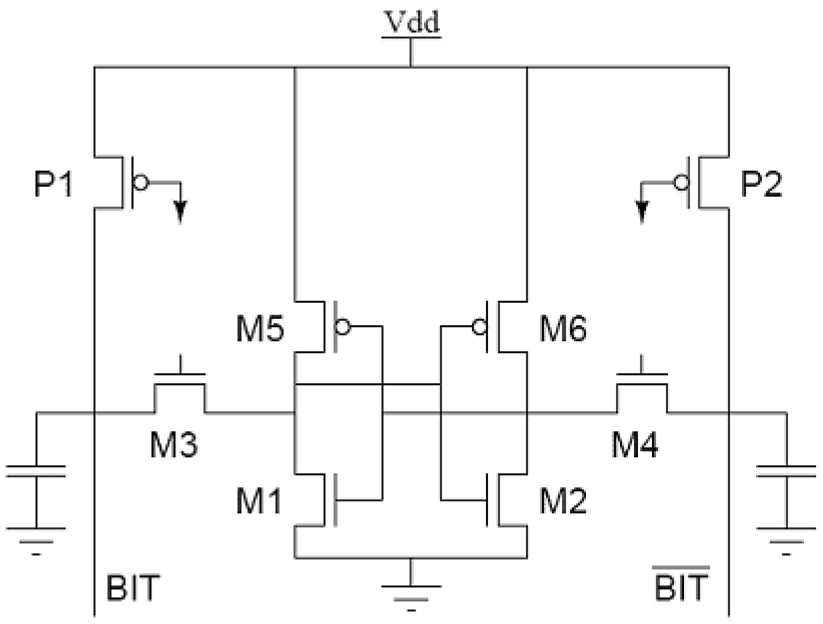
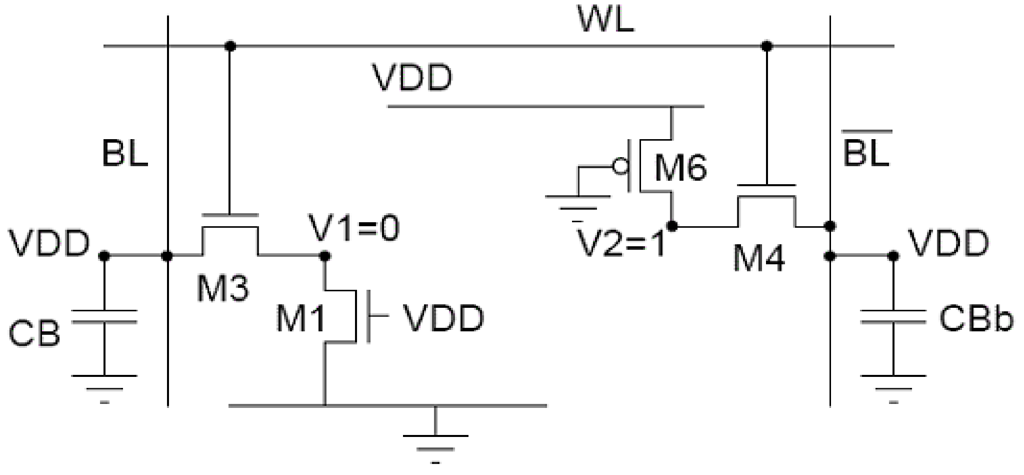


Figure 50: CMOS SRAM cell with precharge transistors

When none of the word lines is selected, the pass transistors M3 and M4 are turned off and the data is retained in all memory cells. The column capacitances are charged by the pull-up transistors P1 and P2. The voltages across the column capacitors reach VDD - VT.

READ Operation

Consider a data read operation, shown in Figure 28.41, assuming that logic '0' is stored in the cell. The transistors M2 and M5 are turned off, while the transistors M1 and M6 operate in linear mode. Thus internal node voltages are V1 = 0 and V2 = VDD before the cell access transistors are turned on.



Read Operation

After the pass transistors M3 and M4 are turned on by the row selection circuitry, the voltageC_{Bb} of will not change any significant variation since no current flows through M4. On the other hand M1 and M3 will conduct a nonzero current and the voltage level ofC_B will begin to drop slightly. The node voltage V1 will increase from its initial value of '0'V. The node voltage V1 may exceed the threshold voltage of M2 during this process, forcing an unintended change of the stored state. Therefore voltage must not exceed the threshold voltage of M2, so the transistor M2 remains turned off during read phase.

WRITE Operation

Consider the write '0' operation assuming that logic '1' is stored in the SRAM cell initially. Figure 28.51 shows the voltage levels in the CMOS SRAM cell at the beginning of the data write operation. The transistors M1 and M6 are turned off, while M2 and M5 are operating in the linear mode. Thus the internal node voltage V1 = VDD and V2 = 0 before the access transistors are turned on. The column voltage V_b is forced to '0' by the write circuitry. Once M3 and M4 are turned on, we expect the nodal voltage V2 to remain below the threshold voltage of M1, since M2 and M4 are designed.

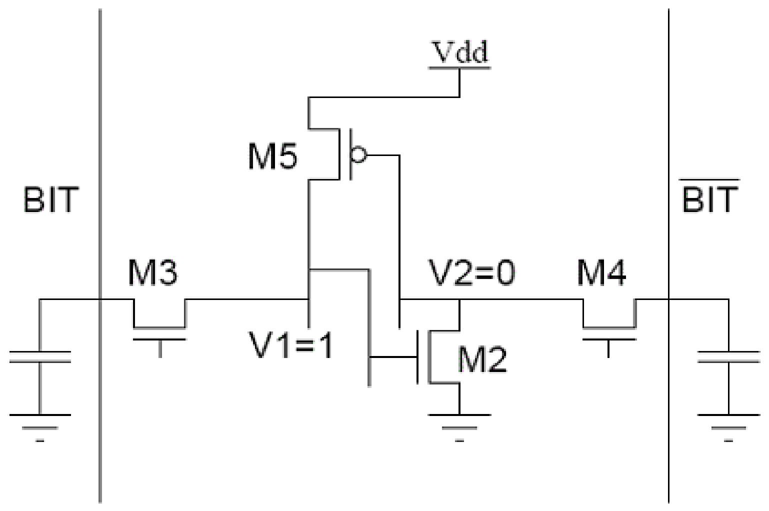


Figure 50: SRAM start of write '0'

The voltage at node 2 would not be sufficient to turn on M1. To change the stored information, i.e., to force V1 = 0 and V2 = VDD, the node voltage V1 must be reduced below the threshold voltage of M2, so that M2 turns off. When $V_1=V_{T,n}$ the transistor M3 operates in linear region while M5 operates in saturation region.

WRITE Circuit

The principle of write circuit is to assert voltage of one of the columns to a low level. This can be achieved by connecting either BIT or BIT' to ground through transistor M3 and either of M2 or M1. The transistor M3 is driven by the column decoder selecting the specified column. The transistor M1 is on only in the presence of the write enable signal and when the data bit to be written is '0'. The transistor M2 is on only in the presence of the write signal W'=0 and when the data bit to be written is '1'.

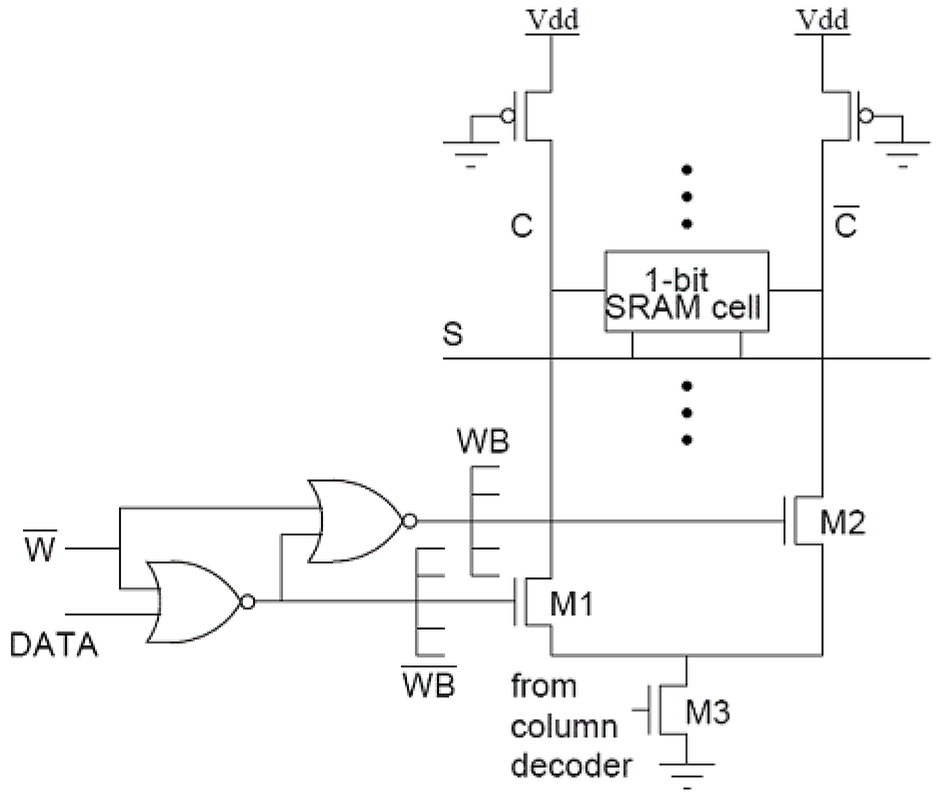


Figure 51: Circuit for write operation

DYNAMIC RANDOM ACCESS MEMORY (DRAM)

DRAM Basics

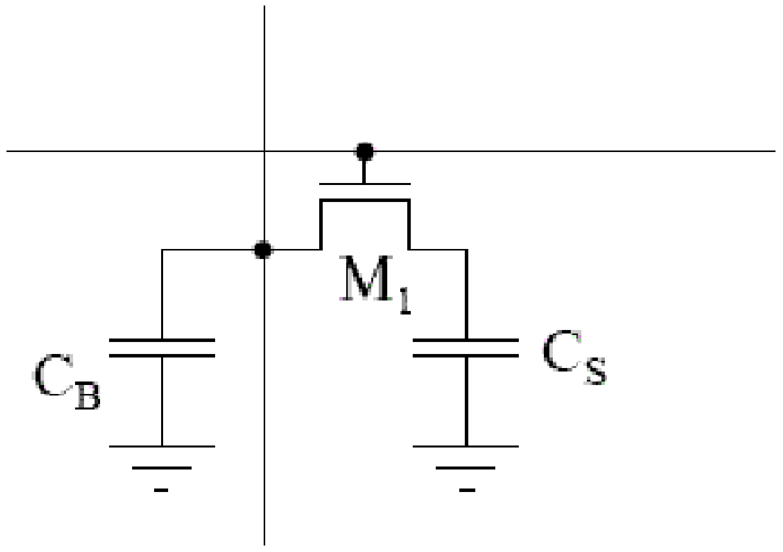


Fig: DRAM Cell

The C_S capacitor stores the charge for the cell. Transistor M1 gives the R/W access to the cell. C_B is the capacitance of the bit line per unit length.

Memory cells are etched onto a silicon wafer in an array of columns (bit lines) and rows (word lines). The intersection of a bit line and word line constitutes the address of the memory cell.

DRAM works by sending a charge through the appropriate column (CAS) to activate the transistor at each bit in the column. When writing, the row lines contain the state the capacitor should take on. When reading, the sense amplifier determines the level of charge in the capacitor. If it is more than 50%, it reads it as "1"; otherwise it reads it as "0". The counter tracks the refresh sequence based on which rows have been accessed in what order. The length of time necessary to do all this is so short that it is expressed in nanoseconds (billionths of a second). e.g. a memory chip rating of 70ns means that it takes 70 nanoseconds to completely read and recharge each cell.

The capacitor in a dynamic RAM memory cell is like a leaky bucket. Dynamic RAM has to be dynamically refreshed all of the time or it forgets what it is holding. This refreshing takes time and slows down the memory.

Semiconductor ROMs

Introduction

Read only memories are used to store constants, control information and program instructions in digital systems. They may also be thought of as components that provide a fixed, specified binary output for every binary input.

The read only memory can also be seen as a simple combinational Boolean network, which produces a specified output value for each input combination, i.e. for each address. Thus storing binary information at a particular address location can be achieved by the presence or absence of a data path from the selected row (word line) to the selected column (bit line), which is equivalent to the presence or absence of a device at that particular location.

The two different types of implementations of ROM array are:

- NOR-based ROM array
- NAND-based ROM array

NOR-based ROM Array

There are two different ways to implement MOS ROM arrays. Consider the first 4-bit X 4-bit memory array as shown in Figure 31.21. Here, each column consists of a pseudo nMOS NOR gate driven by some of the row signals, i.e., the word line.

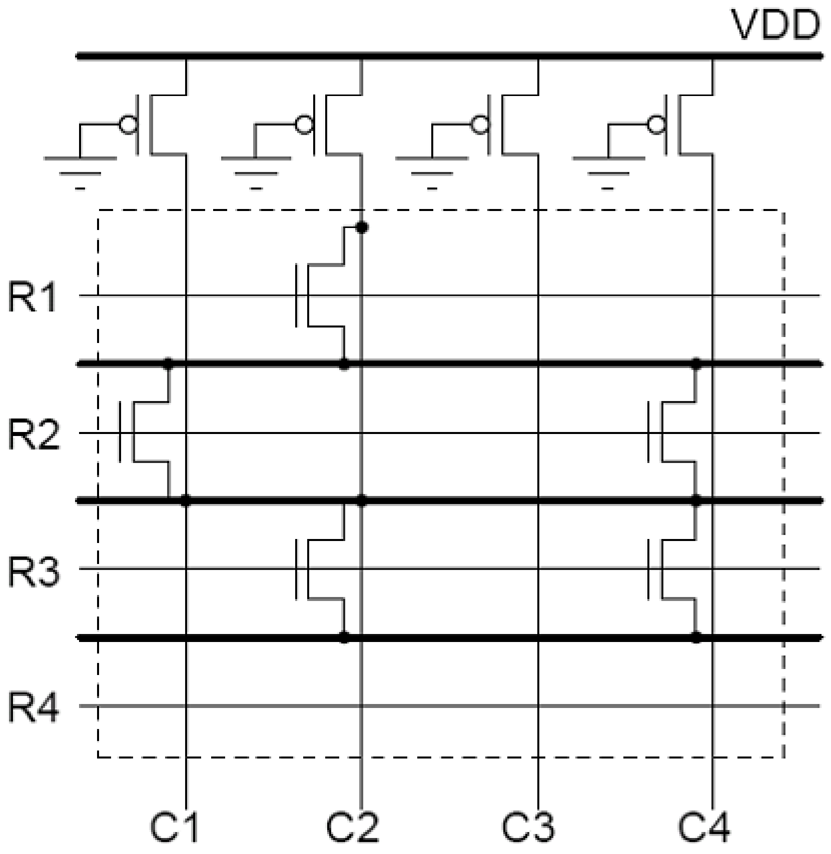


Fig: NOR-based ROM array

As we know, only one word line is activated at a time by raising tis voltage to VDD, while all other rows are held at a low votlage level. If an active transistor exists at the cross point of a column and the selected row, the column voltage is pulled down to the logic LOW level by that transistor. If no active transistor exists at the cross point, the column voltage is pulled HIGH by the pMOS load device. Thus, a logic "1"-bit is stored as the absence of an active transistor, while a logic "0"-bit is stored as the presence of an active transistor at the cross point.

NAND-based ROM Array

In this types of ROM array which is shown in Figure 31.31, each bit line consists of a depletion-load NAND gate, driven by some of the row signals, i.e. the word lines. In normal operation, all word lines are held at the logic HIGH voltage level except for the selected line, which is pulled down to logic LOW level. If a transistor exists at the cross point of a column and the selected row, that transistor is turned off and column voltage is pulled HIGH by the load device. On the other hand, if no transistor exists (shorted) at that particular cross point, the column voltage is pulled LOW by the other nMOS transistors in the multi-input NAND structure. Thus, a logic "1"-bit is stored by the presence of a transistor that can be deactivated, while a logic "0"-bit is stored by a shorted or normally ON transistor at the cross point.

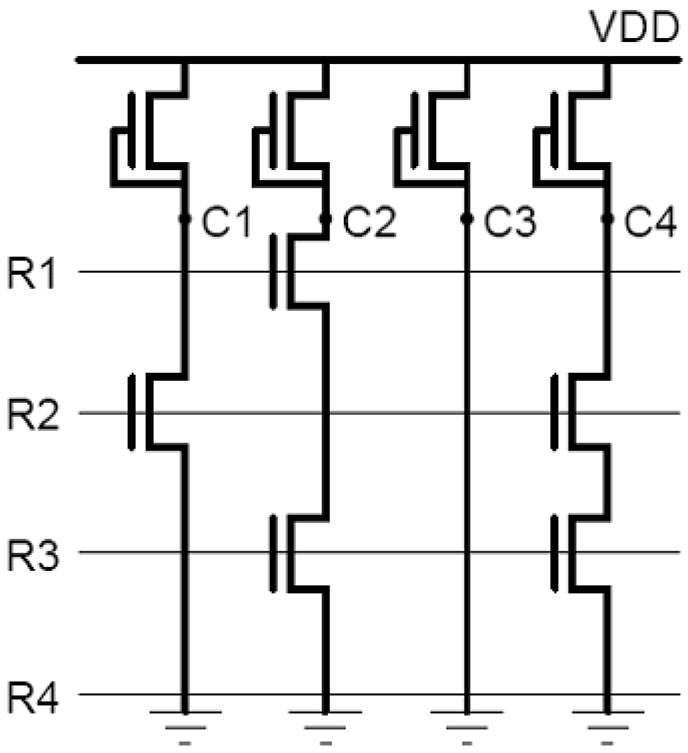


Figure 51: NAND-based ROM

Few special Examples of Memories

Erasable Programmable Read Only Memory (EPROM)

An EPROM is erased by shining ultraviolet light on the cells through a transparent window in the package. The UV radiation renders the oxide slightly conductive by direct generation of electron-hole pair in the material. The erasure process is slow and can take from seconds to several minutes, depending on the intensity of the UV source. Programming takes several (5-10) microseconds/word. Another problem with the process is limited endurance, that is, the number of erase/program cycles is generally limited to maximum of 1000, mainly as a result of UV erase procedure. Reliability is also an issue. The device threshold might vary with repeated programming cycles. Most EPROM memories therefore contain on-chip circuitry to control the value of thresholds to within a specified range during programming. Finally, the injection always entails a large channel current, as high as 0.5mA at a control gate voltage of 12.5V. This causes high power dissipation during programming. The EPROM cell is extremely simple and dense, making it possible to fabricate large memories at a low cost. EPROMs were therefore attractive in applications that do not require regular programming. Due to cost and reliability issues, EPROMs have fallen out of favor and have been replaced by Flash Memories.

Electrically Erasable Programmable Read Only Memory (EEPROM)

The major disadvantage of the EPROM approach is that erasure procedure has to occur "off system". This means the memory must be removed from the board and placed in the EPROM programmer for programming. The EEPROM approach avoids this labor intensive and annoying procedure by using another mechanism to inject or remove charges from the floating gate viz. tunneling. procedure.

Module 4

Design Capture Tool

Computer aided design (CAD) tools are essential for timely development of integrated circuits.Although CAD tools cannot replace the creative and inventive parts of the design activities,the majority of time consuming and computation intensive mechanistic parts of the design can be executed by using CAD tools.The CAD technology for VLSI chip design can be categorized into the following areas:

a)High level synthesis

b)Logic synthesis

c)circuit optimization

d)Layout

e)simulation

f)Design rule checking

g)Formal verification

Synthesis tools

The high level synthesis tools using hardware description languages(HDLs) such as VHDL or Verilog address the automation of the design phase in the top level of the design hierarchy,With an accurate estimation of lower level design features such as chip area and signal delay it can very effectively determine the types and quantities of modules to be included in the chip design.

Layout tools

The tools for circuit optimization are concerned with transistor sizing for minimization of delays and with process variation ,noise and reliability hazards.The layout CAD tools include floorplanning,place and route and module generation.

Simulation and verification tools

The simulation category ,which is the most mature area of VLSI CAD,includes many tools ranging from circuit level simulation,timimg level simulation,logic level simulation and behavirol simulation.

Hardware Description Language: VHDL

What is VHDL?

• A hardware description language that can be used to model a digital system.

• VHDL = VHSIC (Very High Speed Integrated Circuit) Hardware Description Language

• Can describe:

− behavior

− structure, and

− timing of a logic circuit.

Hardware Modelling in VHDL

VHDL is NOT a programming language like C or Java. It is used to model the physical hardware used in digital systems.Therefore you must always think about the hardware you wish to implement when designing systems using VHDL.

Data Objects

A data object holds a value of a specified type. The data objects that can be synthesized directly in hardware are:

1. **Signal**: Represents a physical wire in a circuit. It holds a list of values which includes its current value and a set of possible future values.

2. **Variable**: Used to hold results of computations. It does not necessarily represent a wire in a circuit.

3. **Constant**: Contains a value that cannot be changed. It is set before the beginning of a simulation.

Predefined Data Types

• Standard logic type:	STD_LOGIC, STD_LOGIC_VECTOR (Can hold 0, 1, Z, and —.)
• Bit type:	BIT, BIT_VECTOR
• Integer type:	INTEGER
• Floating−point type:	REAL
• Physical type:	TIME
• Enumeration type:	BOOLEAN, CHARACTER

- To use the STD_LOGIC and STD_LOGIC_VECTOR types, the std_logic_1164 package must be included in the VHDL design file.

- We can define our own data types. This is especially useful when designing finite−state machines (FSMs).

- An object declaration is used to declare an object, it type, its class, and optionally to assign it a value

Object Declaration Examples

- Signal declarations:
  SIGNAL sresetn : STD_LOGIC;
  SIGNAL address : STD_LOGIC_VECTOR(7 downto 0);
- Variable declarations:
  VARIABLE index : INTEGER range 0 to 99 :=20;
  VARIABLE memory : BIT_MATRIX(0 to 7, 0 to 1023);
- Constant declarations:
  CONSTANT cycle_time : TIME := 100 ns;
  CONSTANT cst : UNSIGNED(3 downto 0);

Operators

Operator Class	Operator
Miscellaneous	**, ABS, NOT
Multiplication	*, /, MOD, REM
Unary Arithmetic (Sign)	+, −
Addition	+, −, &
Shift/Rotate	sll, srl, sla, sra, rol, ror
Relational	=, /=, <, <=, >, >=
Logical	and, or, nand, nor, xor, xnor

- The individual operators in each class have the same precedence.

VHDL Design Entity

Entity Declaration

− Specifies the interface of entity to the outside world.

− Has a name.

− Includes PORT statement which specifies the entity’s input and output signals (ports)

− Ports can have different modes: IN, OUT, INOUT, BUFFER

Architecture

− Provides circuit details for an entity

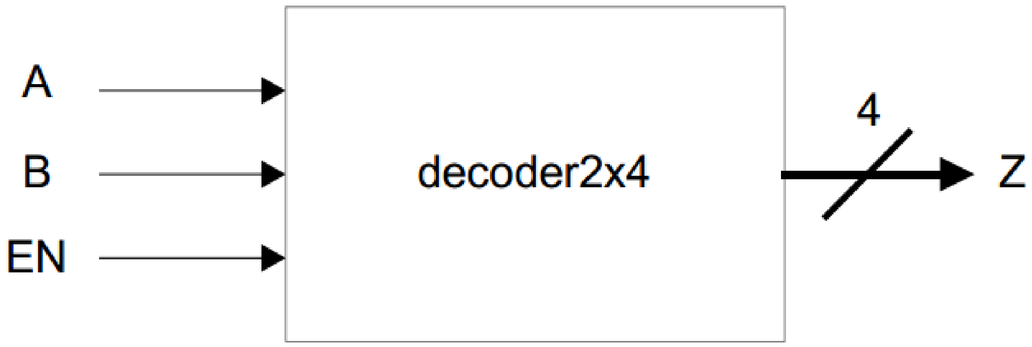
General form:
ARCHITECTURE arch_name OF entity_name IS
    Signal declarations

```
        Constant declarations
        Type declarations
        Component declarations
BEGIN
        Component instantiations
        Concurrent assignment statements
        Process statements
END arch_name
```

Concurrent Assignment Statements

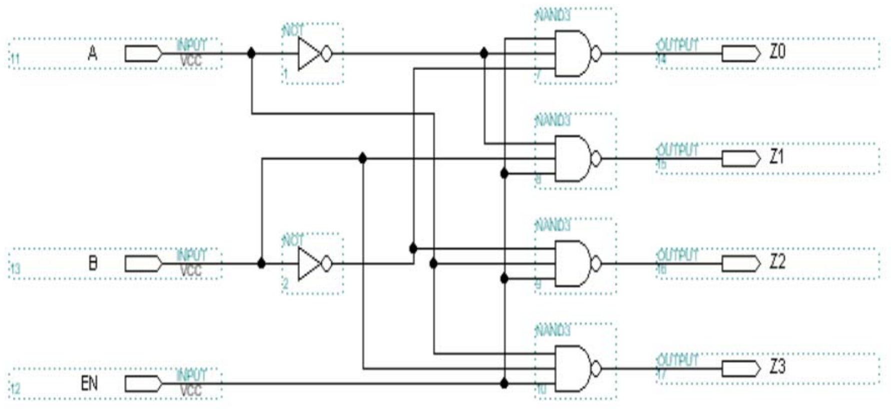
- A concurrent assignment statement is used to assign a value to a signal in an architecture body.
- Used to model combinational circuits.
- The order in which these statements occur does not affect the meaning of the code.

2x4 Decoder Example



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY decoder2x4 IS
PORT (A, B, EN : IN STD_LOGIC;
Z : OUT STD_LOGIC_VECTOR(3 downto 0));
END decoder2x4;
––this is a comment
```

Architecture Body of 2x4 Decoder



```
ARCHITECTURE dec_df OF decoder2x4 IS
SIGNAL ABAR, BBAR : STD_LOGIC;
BEGIN
––Order of concurrent signal assignment
statements is not important.
Z(3) <= not (A and B and EN);
Z(0) <= not (ABAR and BBAR and EN);
BBAR <= not B;
Z(2) <= not (A and BBAR and EN);
ABAR <= not A;
Z(1) <= not (ABAR and B and EN);
END dec_df;
```

Sequential Assignment Statements

- Sequential assignment statements assign values to signals and variables. The order in which these statements appear can affect the meaning of the code.
- Can be used to model combinational circuits and sequential circuits.
- Require use of the PROCESS statement.
- Include three variants: IF statements, CASE statements, and LOOP statements. 2x4 Decoder Revisited

IF and CASE Statements

- IF and CASE statements are used to model multiplexers, decoders, encoders, and comparators.
- Can only be used in a PROCESS.

Modelling a 4−1 Multiplexer

Using an IF statement:

```
PROCESS (Sel, A, B, C, D)
```

```
BEGIN
IF (Sel = "00") THEN
Y <= A;
ELSIF (Sel = "01")
THEN
Y <= B;
ELSIF (Sel = "10")
THEN
Y <= C;
ELSE
Y <= D;
END IF;
END PROCESS;
```

Using a CASE statement:

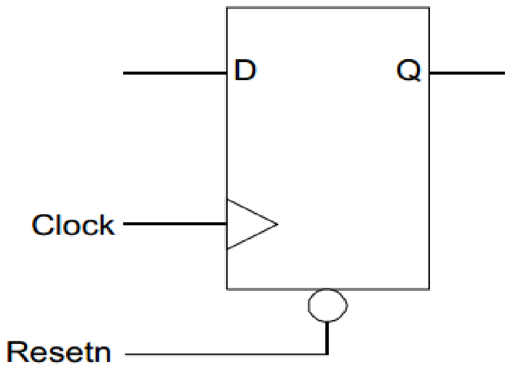
```
PROCESS (Sel, A, B, C, D)
BEGIN
CASE Sel IS
WHEN "00" => Y <= A;
WHEN "01" => Y <= B;
WHEN "10" => Y <= C;
WHEN "11" => Y <= D;
WHEN OTHERS =>Y
<=A;
END CASE;
END PROCESS;
```

- Can also model multiplexers with WHEN/ELSE clause and WITH/SELECT clause. These can only be used outside of a PROCESS.

Behavioural vs. Structural Modelling

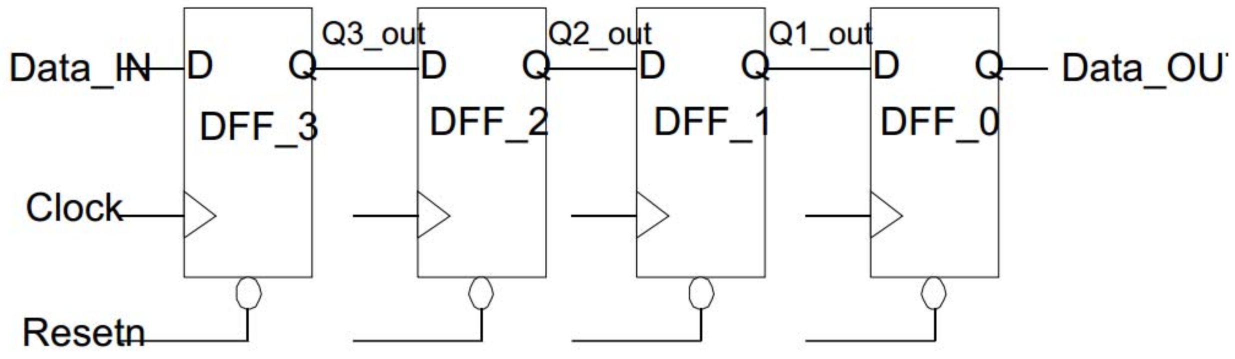
- With VHDL, we can describe the behaviour of simple circuit building blocks and then use these to build up the structure of a more complex circuit.
- Behavioural modelling is useful because it allows the designer to build a logic circuit without having to worry about the low−level details.
- Structural modelling is useful because it tells the synthesis tools exactly how to construct a desired circuit.

Behavioural Model of a D Flip−Flop



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY my_dff IS
PORT ( D : IN  STD_LOGIC;
Q : OUT STD_LOGIC;
Clock : IN  STD_LOGIC;
Resetn: IN STD_LOGIC);
END my_dff;
ARCHITECTURE Behaviour OF my_dff IS
BEGIN
PROCESS(Clock, Resetn)
BEGIN
IF Resetn='0' THEN
Q <= '0';
ELSIF (Clock'EVENT AND Clock='1') THEN
Q <= D;
END IF;
END PROCESS;
END Behaviour;
```

Structural Model of a 4−Bit Shift Register



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

```
ENTITY shift_reg_struct IS
PORT (Data_IN : IN STD_LOGIC;
Data_OUT: OUT STD_LOGIC;
CLK, RESN : IN STD_LOGIC);
END shift_reg_struct;
ARCHITECTURE Structure OF shift_reg_struct IS
COMPONENT my_dff
PORT ( D : IN  STD_LOGIC;
Q : OUT STD_LOGIC;
Clock : IN  STD_LOGIC;
Resetn: IN STD_LOGIC);
END COMPONENT;
SIGNAL Q3_out, Q2_out, Q1_out : STD_LOGIC;
BEGIN
DFF_3 : my_dff PORT MAP (Data_IN, Q3_out, CLK, RESN);
DFF_2 : my_dff PORT MAP (Q3_out, Q2_out, CLK, RESN);
DFF_1 : my_dff PORT MAP (D=>Q2_out, Q=>Q1_out,
Clock=>CLK, Resetn=>RESN);
DFF_0 : my_dff PORT MAP (D=>Q1_out, Q=>Data_OUT,
Clock=>CLK, Resetn=>RESN);
END Structure;
```

Testing and Verification:

Defects, Errors and Faults

Models bridge the gap between *physical reality* and a *mathematical abstraction* and allow for development of analytical tools.

Definitions:
- **Defect**: An untended difference between the implemented hardware and its intended function.
- **Error**: A wrong *output signal* produced by a defective system.
- **Fault** is a *logic level abstraction* of a **physical defect**. Used to describe the change in the logic function caused by the defect. *Fault abstractions* reduce the number of conditions that must be considered in deriving tests.

A *collection of faults*, all of which are based on the same set of assumptions concerning the nature of defects, is called a **fault model**.
For example:
- **Defect**: *A* shorted to GND.
- **Fault**: *A* Stuck-at logic 0.
- **Error**: *A=1*, *B=1 => C=1*

Correct output is *C=0*.
Note that the **error** is *not* permanent
For example, no error occurs if at least one input is 0.

Functional Vs. Structural Testing

Design verification may need to use exhaustive functional tests. Fortunately, for hardware testing, we **can assume** the function is correct. The focus on **structure** makes it possible to develop algorithms that are independent of the design. The algorithms are based on **fault models**.

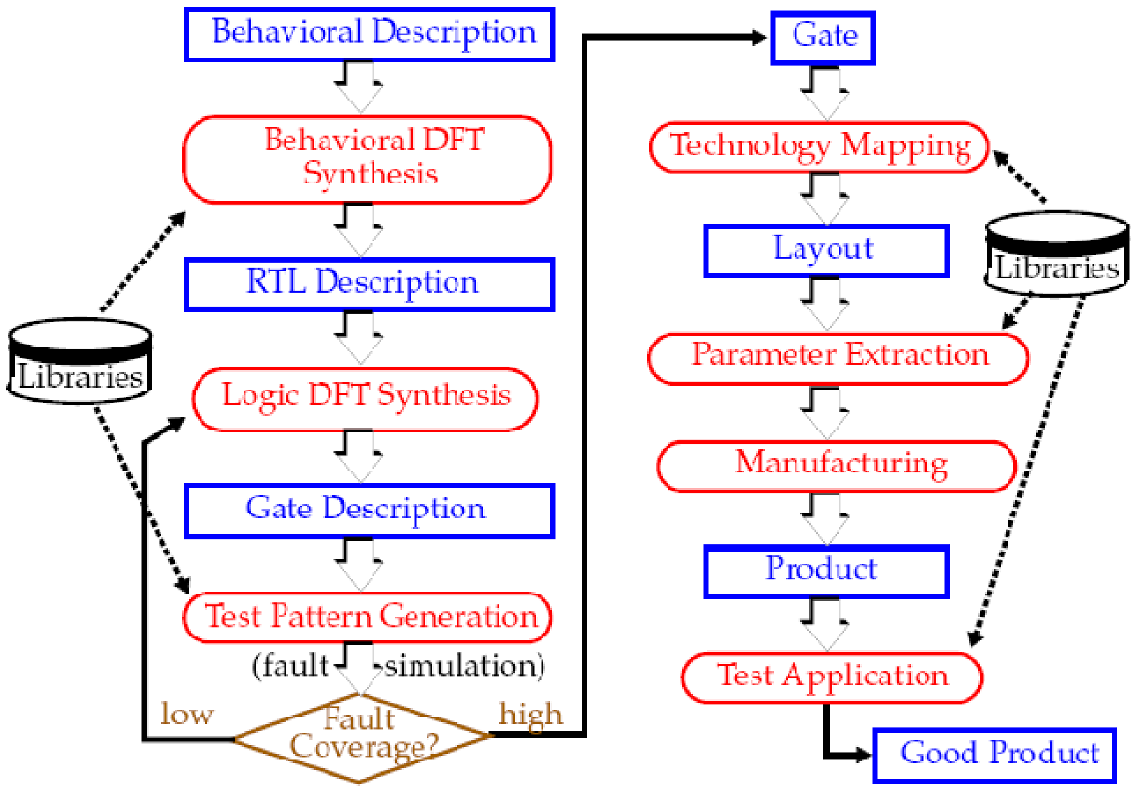
Fault models can be formulated at the various levels of design abstraction.

• *Behavioral level*: Faults may not have any obvious correlation to defects.

• *RTL and Logic level*: Stuck-at faults most popular, followed by bridging and delay fault models.

• *Transistor (switch) level*: Technology-dependent faults.

Design Levels from Testability Perspective:



Fault Models

The text describes these and other fault models:

• Bridging fault

• Defect-oriented fault (e.g. bridging, stuck-open, IDDQ).

• Delay fault (transition, gate-delay, line-delay, segment-delay, path-delay).

• Intermittent fault

• Logical fault (often stuck-at)

• Memory fault (single cell SA0/1, pattern sensitive, cell coupling faults).

• Non-classical fault (stuck-open or stuck-on, transistor faults for CMOS).

• Pattern sensitive fault

• Pin fault (SA faults on the signal pins of all modules in the circuit).

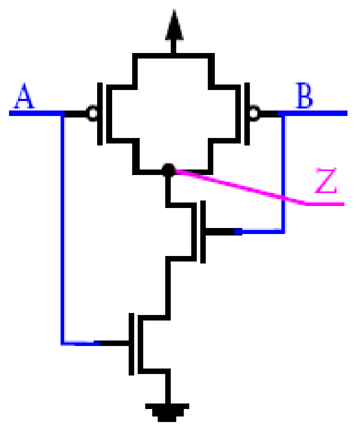
• Redundant fault

• Stuck-at fault

Single stuck-at faults (SSF)

Assumes defects cause the signal net or line to remain at a fixed voltage level. Model includes **stuck-at-0** (SA0) or **stuck-at-1** (SA1) faults and assumes only one fault exists.

For example, how many *SSF* faults can occur on an *n-input* NAND gate?



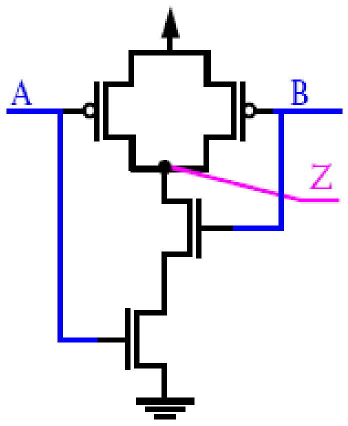
Inputs	Fault-Free	Faulty Response					
AB	Response	A/0	B/0	Z/0	A/1	B/1	Z/1
00	1	1	1	0	1	1	1
01	1	1	1	0	0	1	1
10	1	1	1	0	1	0	1
11	0	1	1	0	0	0	1

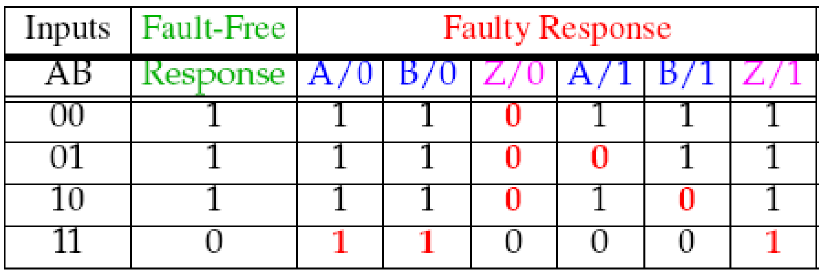
What fault(s) does the pattern *AB* = *01* detect?

What is the *minimum* number of tests needed to "detect" all of them?

What are the tests?

A **dominant** input value is defined as the value that *determines* the state of the output independent of the other values of the inputs.





Inputs	Fault-Free	Faulty Response					
AB	Response	A/0	B/0	Z/0	A/1	B/1	Z/1
00	1	1	1	0	1	1	1
01	1	1	1	0	0	1	1
10	1	1	1	0	1	0	1
11	0	1	1	0	0	0	1

What is the dominant input value for the NAND?

How many tests do you need to diagnosis the fault (hint: deduction can be used to distinguish)?

Can you distinguish between all of the faults? An *n*-line circuit can have at most *2n* SSF faults.

This number can be further reduced through **fault collapsing**.

Fault detection requires:

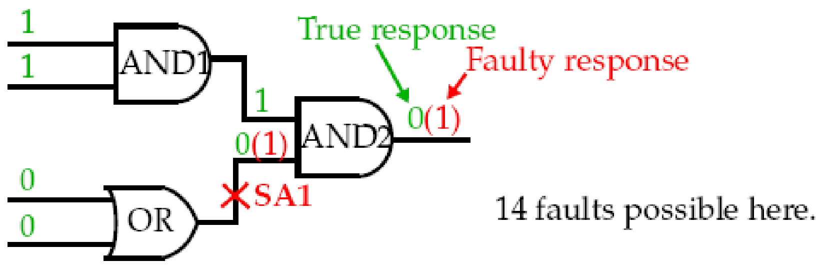
• A test *t* activates or provokes the fault *f*.

• *t* propagates the error to *observation point* (primary output (PO)/scan latch).

A line that changes with *f* is said to be **sensitized** to the fault site.

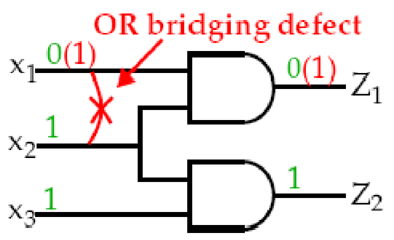
Fault propagation requires *off-path* inputs be set to *non-dominant* values.

01, *10*, and *11*do **not** provoke the fault



Let *Z(t)* represent the response of a circuit *N* under input vector *t*.

Fault *f* transforms the circuit to *Nf* and its response to *Zf(t)*.



$Z_1 = x_1x_2 \longrightarrow Z_{1f} = x_1 + x_2$

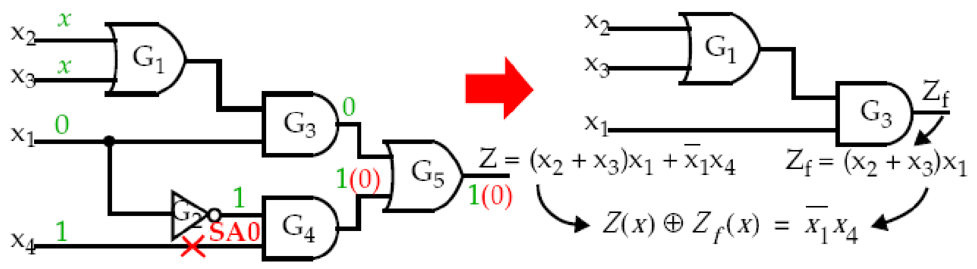
$Z_2 = x_2x_3 \longrightarrow Z_{2f} = (x_1 + x_2)x_3$

011 defects *f* because $Z(011) = 01$ and $Z_f(011) = 11$.

The set of all tests *x* that detect *f* is given by

$$Z(x) \oplus Z_f(x) = 1$$

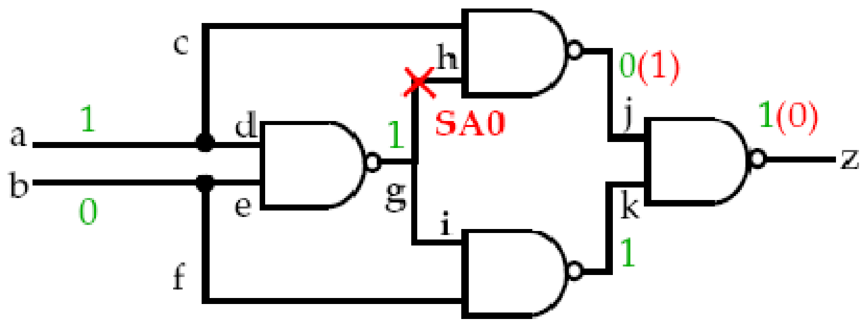
In this example, any test in which *x1* = 0 and *x4* = 1 is a test for *f*.



The expression (x1compliment)x4 represents 4 tests (*0001*, *0011*, *0101*, *0111*).

Note that faults on *fanout stems* and *fanout branches* are **not** the same. This will be important later when we develop a method to *collapse* faults that are equivalent.

In this example, we assume line *g*, *h* and *i* carry the same signal value.

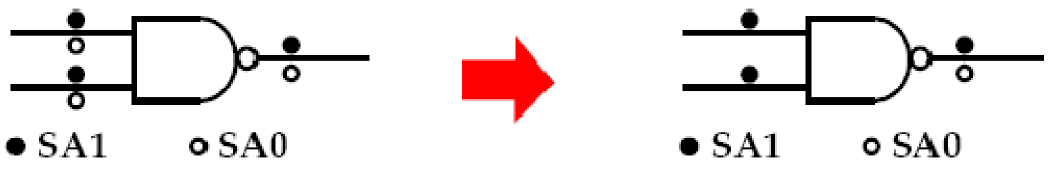


Input *ab* = (*10*) activates *h SA0* and is detectable at *z*. This input also activates *SA0* faults on *g* and *i* .However, *i SA0* is **not** detectable since it is blocked by *f* = 0.

We consider faults *h* and *i* as different because the tests that detect *h* and *i* are not the same -- fault *g* is also different because both vectors detect it.

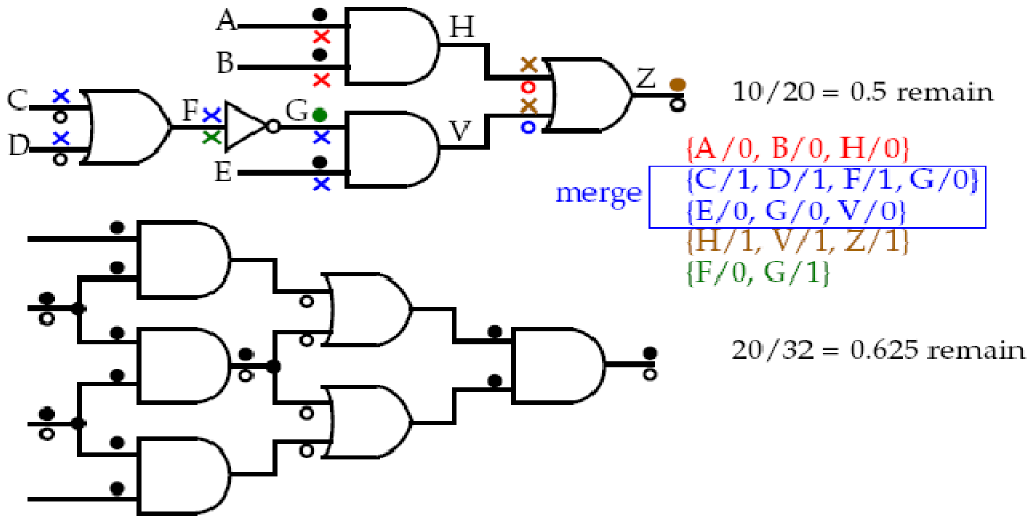
Fault Equivalence

Two faults *f* and *g* are considered **functionally equivalent** if Zf(x) = Zg(x).
There is no test that can distinguish between *f* and *g*. i.e., *all tests* that detect *f* also detect *g*. For large circuits, determining this is computationally intensive. However, equivalence can be determined for simple gates and applied to large circuits.
Any *n*-input gate has 2(n+1) SA faults. For the NAND gate, the *SA0* on the inputs are equivalent to *SA1* on the output and all three are detected by the same test pattern *AB*=(*11*). For any *n*-input primitive gate with *n*>1, only ***n+2*** single SA faults need to be considered.



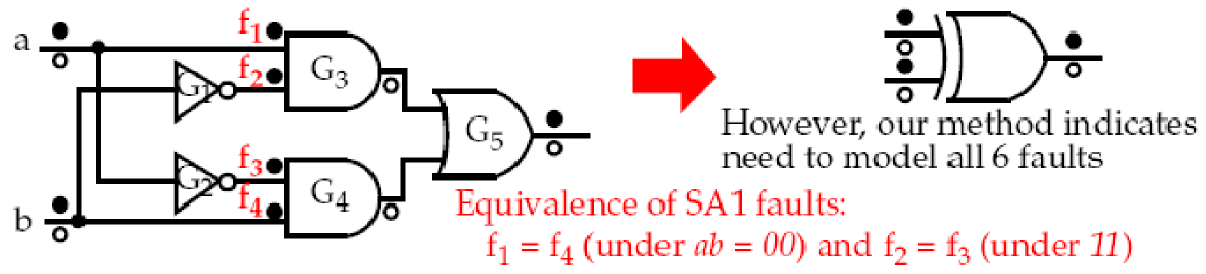
Fault Equivalence

Equivalence fault collapsing is performed from inputs to output.



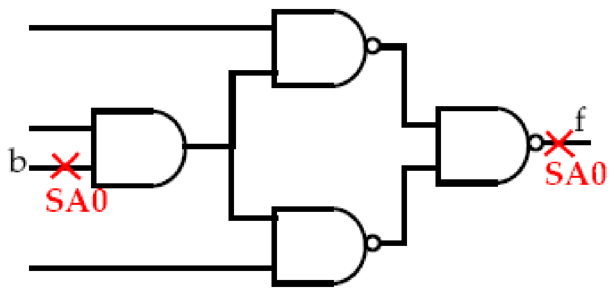
Reduction is between 50-60% and is larger, in general, for fanout free circuits.

Functional equivalence partitions the set of all faults into *functional equivalence classes*, from each of which only one fault needs to be considered. This property is useful for test generation programs. Fault equivalence reduces the size of the fault list. Fault equivalence is important for fault location analysis as well. A *complete location test* set can diagnose a fault to within a functional equivalence class. This represents the *maximal diagnostic* resolution achievable. Fault collapsing using this simple method cannot find all equivalencies.


a
b
f1
f2
f3
f4
G1
G2
G3
G4
G5
However, our method indicates need to model all 6 faults
Equivalence of SA1 faults:
f1 = f4 (under ab = 00) and f2 = f3 (under 11)

Therefore, the equivalence classes derived are not maximal.

Our method will not identify faults *b* SA0 and *f* SA0 as equivalent

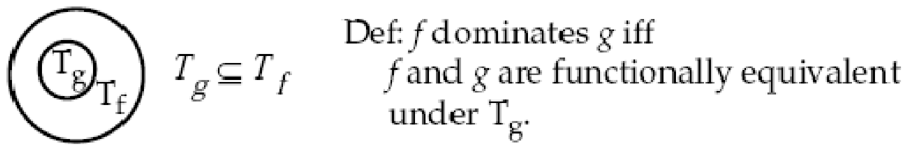


Algorithms are available to solve the more general case of **functional equivalence** based on: **SA0** *Z f* (*x*) Å *Zg*(*x*) = 0

The benefit of identifying these additional fault equivalences is usually not worth the effort, however.

See text for ISCAS'85 benchmark circuit results. If fault detection is the objective (not diagnosis), then **fault dominance** can be used to further reduce the fault list.

A fault *f* dominates another fault *g* if the set of all tests that detect *g*, Tg, is a subset of the test set of Tf.

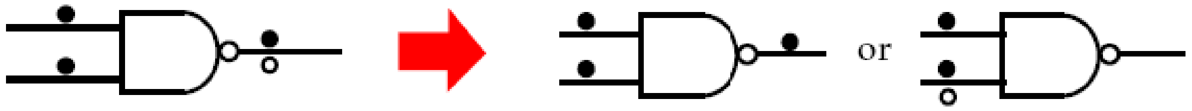


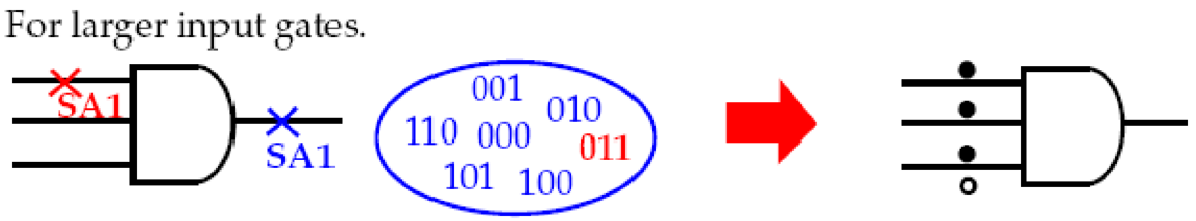
Therefore, any test that detects *g* will also detect *f*.

Since *g* implies *f*, it is sufficient to include *g* in the fault list.

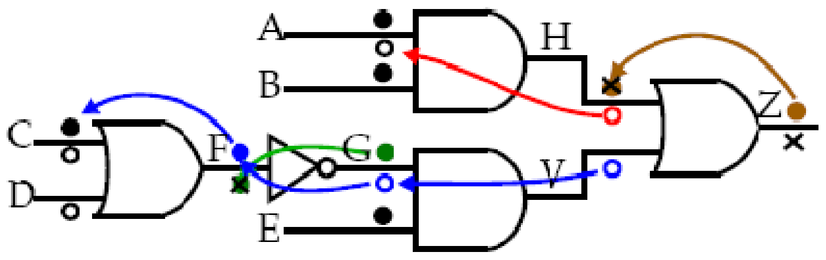
For example, the *SA1* test (*g*) for input *A* of the NAND gate also detects *SA0* (*f*) on the output (the same is true for input *B*.)

Therefore, *Z-SA0* can be dropped from the list.





Dominance fault collapsing is performed from outputs to inputs.



7/20 = 0.3 remain

Here, the *x* indicates the collapsing of the fault via dominance.

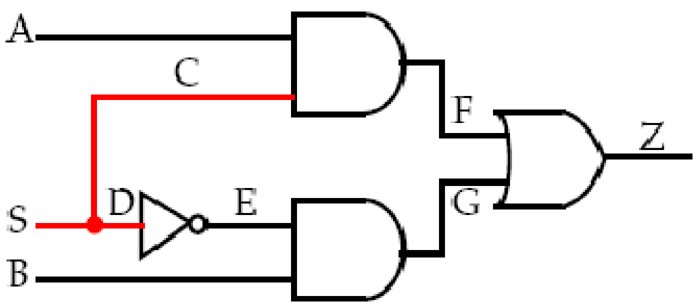
We can also, optionally, choose to move the output fault preserved in the equivalence fault collapsing to an arbitrary input.

One such fault list may be: {*A/0*, *A/1*, *B/1*, *C/0*, *C/1*, *D/0*, *E/1*}

Or another may be: {*B/0*, *A/1*, *B/1*, *C/0*, *D/1*, *D/0*, *E/1*}

Note that these lists contains **only** faults on the PIs.Any test set that detects all SSFs on the PIs of a **fanout free** combinational circuit C detects all SSFs in C.

What happens in the presence of fanout?



Eliminating S, reduces the circuit to a fanout free circuit.

SAB = (*010*) detects C SA1, *SAB* = (*001*) detects D SA1 => both detect S SA1.

SAB = (*110*) detects C SA0, *SAB* = (*101*) detects D SA0 => both detect S SA0.

Therefore, SA faults on *stem* **dominate** SA faults on the *fanout branches*.

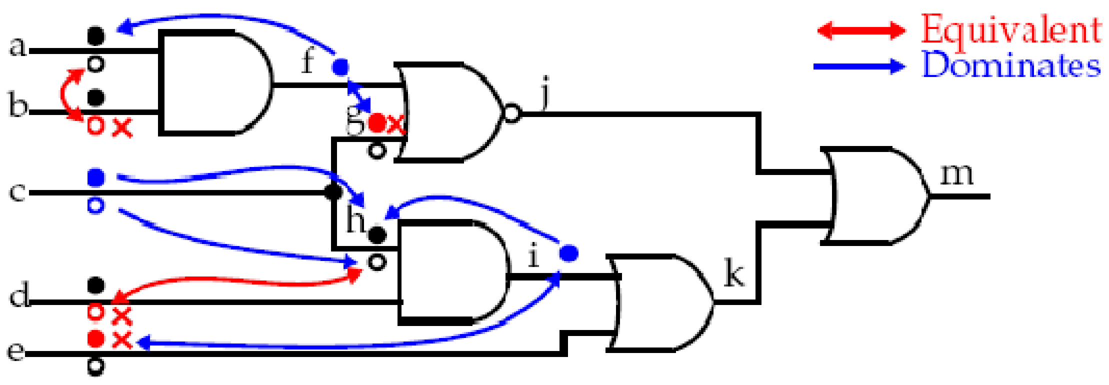
More generally, any test set that detects all SSF on the PIs and fanout branches of C detects all SSF in C.

The PIs and fanout branches are called **checkpoints**. Therefore, it is sufficient to target faults only at the *checkpoints*.

Equivalence and dominance relations can then be used to further collapse the list of faults.

For example, this circuit has 24 SSFs.

But it only has 14 checkpoint faults (the 5 PIs) + *g* and *h*.



This leaves 8 faults from the original list of 14 checkpoint faults.

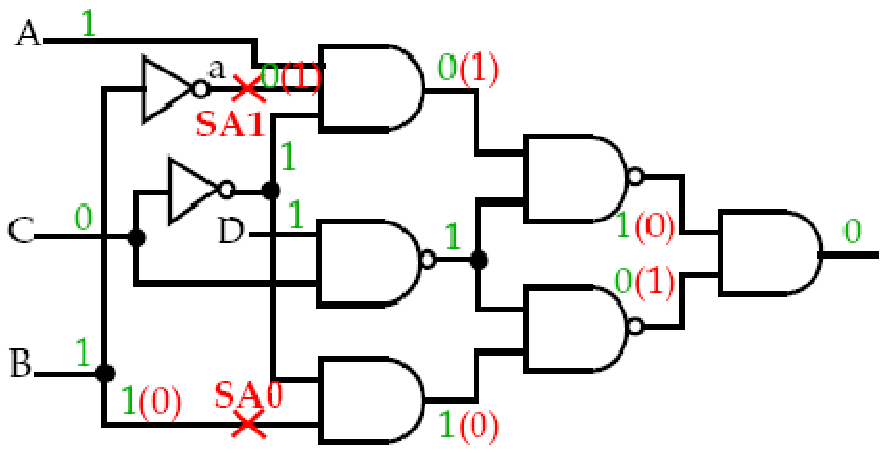
Undetectable Faults

A fault is **detectable** if there exists a test *t* that defects *f*. It is **undetectable**, if **no** test simultaneously activates *f* and creates a sensitized path to a PO.

Undetectable faults may appear to be harmless.

However, a *complete* test set may not be sufficient if one is present in a chip with *multiple faults*.

The fault *b* SA0 is **no longer detectable** by the test *t* = (*1101*) if the **undetectable fault** *a* SA1 is present.

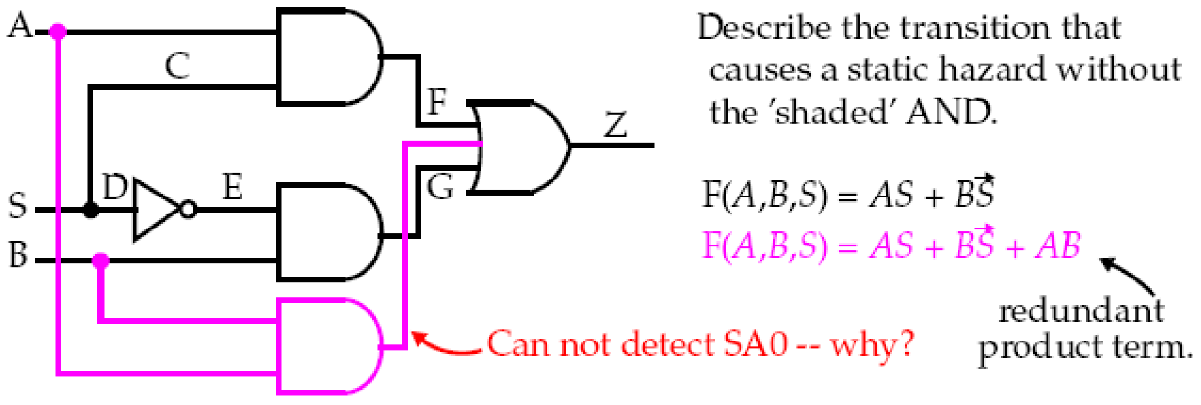


If test *t* is the only test in the complete test set *T* that detects *b* SA0, then *T* is no longer complete in the presence of *a* SA1.

Redundancy: A combinational circuit that contains an *undetectable fault* is said to be redundant.

Redundant faults cause ATPG algorithms to exhibit worst-case behavior.

Redundancy is not always undesirable.


A
C
F
Z
S
D
E
G
B
Can not detect SA0 -- why?
Describe the transition that causes a static hazard without the 'shaded' AND.
$F(A,B,S) = AS + B\overline{S}$
$F(A,B,S) = AS + B\overline{S} + AB$
redundant product term.

Design stratergies for testing

Design for Testability Techniques Design for testability (DFT) refers to those design techniques that make the task of subsequent testing easier. There is definitely no single methodology that solves all embedded system-testing problems. There also is no single DFT technique, which is effective for all kinds of circuits. DFT techniques can largely be divided into two categories, i.e., ad hoc techniques and structured (systematic) techniques. DFT methods for digital circuits:

1) Ad-hoc methods

2) Structured methods:

• Scan

• Partial Scan

• Built-in self-test

• Boundary scan Ad-hoc DFT methods

Ad Hoc Testable Design Techniques

One way to increase the testability is to make nodes more accessible at some cost by physically inserting more access circuits to the original design. Listed below are some of the ad hoc testable design techniques.

Partition-and-Mux Technique

Since the sequence of many serial gates, functional blocks, or large circuits are difficult to test, such circuits can be partitioned and multiplexors (muxes) can be inserted such that some of the primary inputs can be fed to partitioned parts through multiplexers with accessible control signals. With this design technique, the number of accessible nodes can be increased and the number of test patterns can be reduced. A case in point would be the 32-bit counter. Dividing this counter into two 16-bit parts would reduce the testing time in principle by a factor of 215. However, circuit partitioning and addition of multiplexers may increase the chip area and circuit delay. This practice is not unique and is similar to the divide-and-conquer approach to large, complex problems.

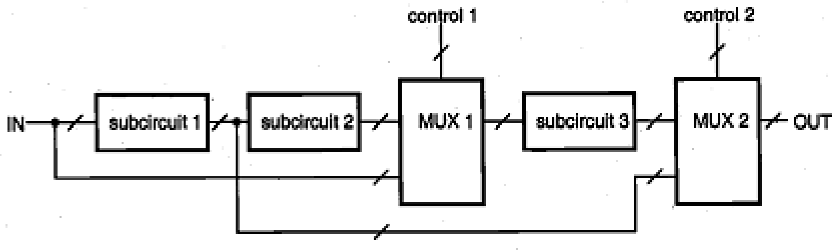


Figure 51: Partition-and-mux method for large circuits.

Initialize Sequential Circuit

When the sequential circuit is powered up, its initial state can be a random, unknown state. In this case, it is not possible to start the test sequence correctly. The state of a sequential circuit can be brought to a known state through initialization. In many designs, the initialization can be easily done by connecting asynchronous preset or clear-input signals from primary or controllable inputs to flip-flops or latches.

Disable Internal Oscillators and Clocks

To avoid synchronization problems during testing, internal oscillators and clocks should be disabled. For example, rather than connecting the circuit directly to the on-chip oscillator, the clock signal can be ORed with a disabling signal followed by an insertion of a testing signal.

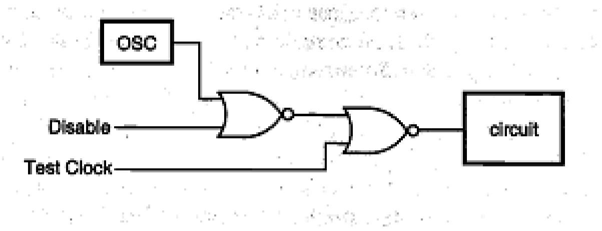


Figure 51: Avoid synchronization problems-via disabling of the oscillator.

Avoid Asynchronous Logic and Redundant Logic

The enhancement of testability requires serious tradeoffs. The speed of an asynchronous logic circuit can be faster than that of the synchronous logic circuit counterpart. However, the design and test of an asynchronous logic circuit are more difficult than for a synchronous logic circuit, and its state transition times are difficult to predict. Also, the operation of an asynchronous logic circuit is sensitive to input test patterns, often causing race problems and hazards of having momentary signal values opposite to the expected values. Sometimes, designed-in logic redundancy is used to mask a static hazard condition for reliability. However, the redundant node cannot be observed since the primary output value cannot be made dependent on the value of the redundant node. Hence, certain faults on the redundant node cannot be tested or detected. The bottom NAND2 gate is redundant and the stuck-at- fault on its output line cannot be detected. If a fault is undetectable, the associated line or gate can be removed without changing the logic function.

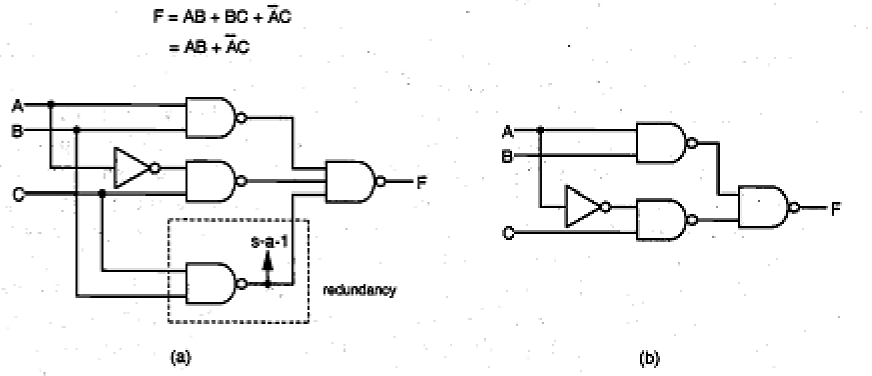


Figure 52: (a) A redundant logic gate example. (b) Equivalent gate with redundancy removed.

Avoid Delay-Dependent Logic

Chains of inverters can be used to design in delay times and use AND operation of their outputs along with inputs to generate pulses.

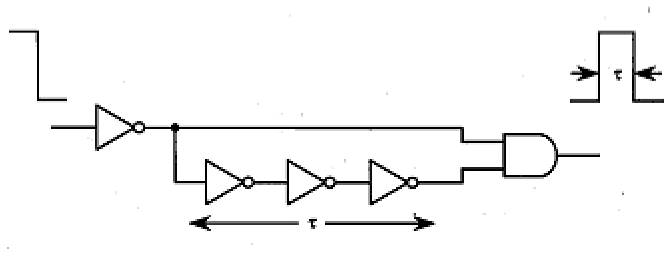


Figure 53: A pulse-generation circuit using a delay chain of three inverters.

Most automatic test pattern generation (ATPG) programs do not include logic delays to minimize the complexity of the program. As a result, such delay-dependent logic is viewed as redundant combinational logic, and the output of the reconvergent gate is always set to logic 0, which is not correct. Thus, the use of delay-dependent logic should be avoided in design for testability

Scan-Based Technique

As discussed earlier, the controllability and observability can be enhanced by providing more accessible logic nodes with use of additional primary input lines and multiplexors. However, the use of additional I/O pins can be costly not only for chip fabrication but also for packaging. A popular alternative is to use scan registers with both shift and parallel load capabilities. The scan design technique is a structured approach to design sequential circuits for testability. The storage cells in registers are used as observation points, control points, or both. By using the scan design techniques, the testing of a sequential circuit is reduced to the problem of testing a combinational circuit.

In general, a sequential circuit consists of a combinational circuit and some storage elements. In the scan-based design, the storage elements are connected to form a long serial shift register, the so-called scan path, by using multiplexors and a mode (test/ normal) control signal.

In the test mode, the scan-in signal is clocked into the scan path, and the output of the last stage latch is scanned out. In the normal mode, the scan-in path is disabled and the circuit functions as a sequential circuit. The testing sequence is as follows:

Step 1: Set the mode to test and, let latches accept data from scan-in input,

Step 2: Verify the scan path by shifting in and out the test data.

Step 3: Scan in (shift in) the desired state vector into the shift register.

Step 4: Apply the test pattern to the prim ary input pins. I : ;

Step 5: Set the mode to normal and observe the primary outputs of the circuit after

sufficient time for propagation.,

Step 6: Assert the circuit clock, for one machine cycle to capture the outputs of the

combinational logic into the registers.

Step 7: Return to test mode; scan out the contents of the registers, and at the same time scan in the next pattern.

Step 8: Repeat steps 3-7 until all test patterns are applied.

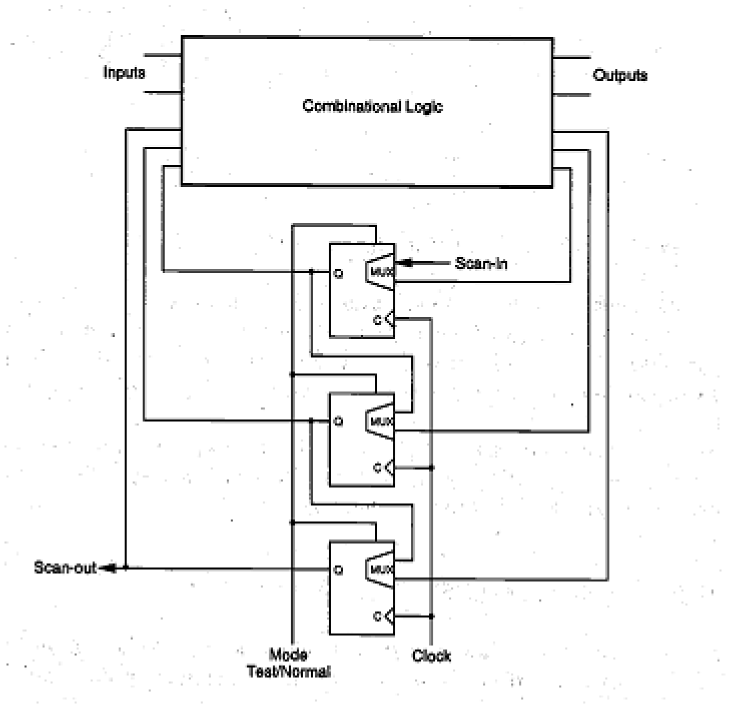


Figure 54: The general structure of scan-based design

The storage cells in scan design can be implemented using edge-triggered D flipflops, master-slave flip-flops, or level-sensitive latches controlled by complementary clock signals to ensure race-free operation. D flip-flop. In large high-speed circuits, optimizing a single clock signal for skews, etc., both for normal operation and for shift operation, is difficult. To overcome this difficulty, two separate clocks, one for normal operation and one for shift operation, are used. Since the shift operation does not have to be performed at the target speed, its clock is much less constrained.

An important approach among scan-based designs is the level sensitive scan design (LSSD), which incorporates both the level sensitivity and the scan path approach using shift registers. The level sensitivity is to ensure that the sequential circuit response is independent of the transient characteristics of the circuit, such as the component and wire delays. Thus, LSSD removes hazards and races. Its ATPG is also simplified since tests have to be generated only for the combinational part of the circuit.

The boundary scan test method is also used for testing printed circuit boards (PCBs) and multichip modules (MCMs) carrying multiple chips. Shift registers are placed in each chip close to I/O pins in order to form a chain around the board for testing. With successful implementation of the boundary scan method, a simpler tester can be used for PCB testing.

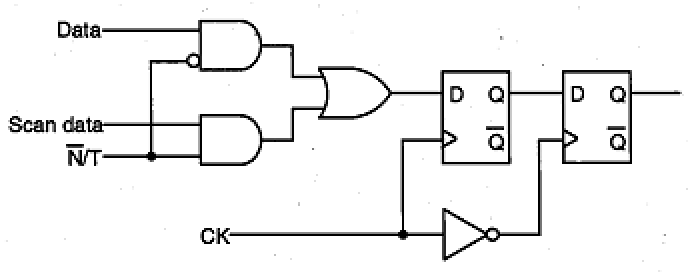


Figure 55: Scan-based design of an edge-triggered D flip-flop

On the negative side, scan design uses more complex latches, flip-flops, I/O pins, and interconnect wires and, thus, requires more chip area. The testing time per test pattern is also increased due to shift time in long registers.

Built-in Self-Test (BIST)

• Capability of a circuit to test itself

• On-line: – Concurrent : simultaneous with normal operation – Nonconcurrent : idle during normal operation

• Off-line: – Functional : diagnostic S/W or F/W – Structural : LFSR-based

• We deal primarily with structural off-line testing

Basic Architecture of BIST

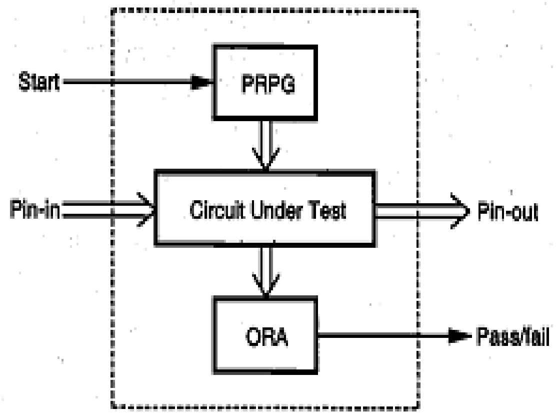


Fig: A procedure for BIST.

• PRPG: Pseudo random pattern generator
• ORA :Output response analyzer

Built-in Self Testing

• Test pattern generation
  – Exhaustive
  – Pseudoexhaustive
  – Pseudorandom
• Test response compression
  – One’s count

– Transition count
– Parity checking
– Syndrome checking
– Signature analysis

]
Test Pattern Generation for BIST
• Exhaustive testing
• Pseudorandom testing
  – Weighted and Adaptive TG
• Pseudoexhaustive testing
  – Syndrome driver counter
  – Constant-weight counter
  – Combined LFSR and shift register
  – Combined LFSR and XOR
  – Cyclic LFSR

Pseudo Random Pattern Generator

To test the circuit, test patterns first have to be generated either by using a pseudo random pattern generator, a weighted test generator, an adaptive test generator, or other means.
A pseudo random test generator circuit can use an LFSR.

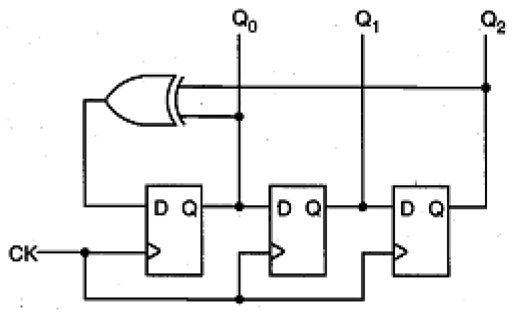


Figure 56: A pseudo-random sequence generator using LFSR

Linear Feedback Shift Register (LFSR) as an ORA

To reduce the chip area penalty, data compression schemes are used to compare the compacted test responses instead of the entire raw test data. One of the popular data compression schemes is the signature analysis, which is based on the concept of cyclic redundancy checking. It uses polynomial division, which divides the polynomial representation of the test output data by a characteristic polynomial and then finds the remainder as the signature. The signature is then compared with the expected signature to determine whether the device under test is faulty. It is known that compression can cause some loss of fault coverage. It is possible that the output of a faulty circuit can match the output of the fault-free circuit; thus, the fault can go undetected in the signature analysis. Such a phenomenon is called aliasing.

In its simplest form, the signature generator consists of a single-input linear feedback shift register (LFSR), all the latches are edge-triggered. In this case, the signature is the content of this register after the last input bit has been sampled. The input sequence $\{a_n\}$ is represented by polynomial G(x) and the output sequence by Q(x). It can be shown that G(x) = Q(x) P(x) + R(x), where P(x) is the characteristic polynomial of LFSR and R(x) is the remainder, the degree of which is lower than that of P(x).

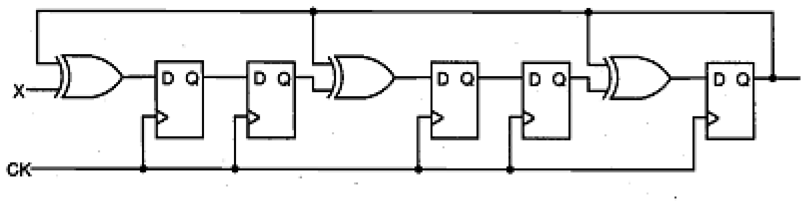


Figure 57: Polynomial division using LFSR for signature analysis.

The characteristic polynomial is

$$P(x)=1+x^2+x^4+x^5$$

For the 8-bit input sequence { 1 1 1 1 0 1 0 1, the corresponding input polynomial is

$$G(x)=x^7+x^6+x^5+x^4+x^2+1$$

and the remainder term becomes $R(x) = x^4 + x^2$, which corresponds to the register contents of {0 0 1 0 11}

Built-In Logic Block Observer

The built-in logic block observer (BILBO) register is a form of ORA which can be used in each cluster of partitioned registers. A basic BILBO circuit allows four different modes controlled by $C_0$and C_1 signals.

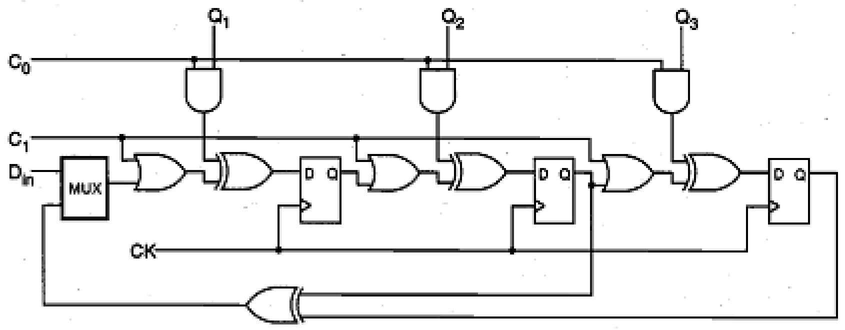


Figure 58: 3-bit built-in logic observer (BILBO) example.

C_0	C_1	Mode
0	0	linear shift
1	0	signature analysis

1	1	data(complemented) latch
0	1	reset

The BILBO operation allows monitoring of circuit operation through exclusive ORing into LFSR at multiple points, which corresponds to the signature analyzer with multiple inputs.

Packaging Technology

Proper packaging technology is critical to the success of the chip development.Package issues have to be taken into consideration in early stages of chip development.Ensure sufficient design margins to accommodate the parasitics of the package

Important packaging concerns:

-Hermetic seals to prevent the penetration of moisture

-Thermal conductivity

-Thermal expansion coefficient

-Pin density

-Parasitic inductance and capacitance

-particle protection

-Cost

Types of packaging technology

Classified by the method used to solder the package on the printed PCB

-Pin-through-hole (PTH)

-Surface-mounted technology (SMT)

Dual in-line packages (DIP)

Advantage of low cost.Not applicable for high-speed operations due to the inductance of the bond wires-Maximum pin count is typically limited to 64

Pin grid array (PGA) packages

-Offers a higher pin count (several houndreds)

-High thermal conductivity especially with a passive or active heat sink

-Requires large PCB area

-Cost is higher than DIP

Chip carrier packages (CCP)

-Leadless chip carrier:

-Chip mounted on PCB directly

-Supports higher pin count

-Problem with difference in thermal coefficient

-Leaded chip carrier:

Quad flat packages (QFP)

-Similar to leaded chip carrier with leads extending outward

Multi-chip modules (MCM)

-Used for very high performance in special applications

-Multiple chips are assembled on a common substrate in a single package

-A large number of critical interconnects among the chips are made within the package

Important features:

-Significant reduction in the overall system size

-Reduced package lead counts

-Faster operation allowed

-Higher implementation cos