An Overview of VLSI 1

VLSI is an acronym that stands for **very-large-scale integration**. This somewhat nebulous term is used to collectively refer to the many fields of electrical and computer engineering that deal with the analysis and design of very dense electronic integrated circuits. Although a strict definition is difficult to come by, one commonly used metric is to say that a VLSI contains more than a million (10^6) or so switching devices or logic gates. Early in the first decade of the 21st century, the actual number of **transistors** (the switching devices) has exceeded 100 million (10^8) for the more complex designs on a piece of silicon (a **chip**), which is typically about 1 centimeter on a side.

This book has been written to provide an understanding of the basics of **digital VLSI chip design**. Emphasis is placed on presenting the details of translating a system specification to a small piece of silicon. The treatment is very technical with many details. Some statements and analyses will appear immediately obvious, while others may not make sense until later chapters. This occurs because the field of VLSI engineering encompasses several distinct "areas of specialization" that mesh together in a unique manner. The most difficult aspect of learning VLSI is seeing the common theme that links the areas together. Once this is accomplished, you are on your way to understanding one of the most fascinating fields of modern times.

1.1 Complexity and Design

Engineering a VLSI chip is an extremely complex task. When attempting to describe the field to a non-technical group, the idea of the "VLSI design funnel" shown in Figure 1.1 helps break the ice. This views the process as one where we provide the basic necessities such as money, an idea, and

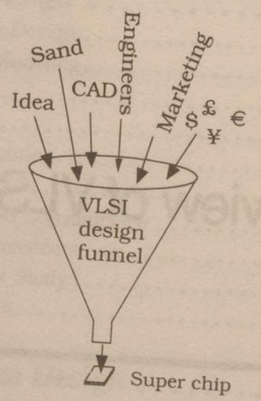


Figure 1.1 The VLSI design funnel

marketing information and dump them all into a "magic technology fu nel." Adding a pile of sand as a raw material produces the super chip the bottom that will sell millions of units and hopefully revolutionize th world. And maybe make someone rich. Of course, engineers and scien tists are needed somewhere in the process, but they just put the thing together. Unfortunately, the process is slightly more complicated tha portrayed in this example.

Any system that is composed of millions of elements is inherently diffi cult to understand. One human mind cannot process information of th complexity that is required for the design and implementation. Creating **design team** provides a realistic approach to approaching a VLSI project as it allows each person to study small sections of the system. In a mod ern design, hundreds of engineers, scientists, and technicians may be working different parts of the design. However, since the team is working on a single project, it is important that each team member have some understanding of where their work falls within the overall scheme. This is accomplished by means of the **design hierarchy**, where the chip is viewed at many different "levels" from the abstract to the physical implementa tion. Every level is important, and each has subdivisions that can evolve into a lifetime career.

In our treatment of VLSI, we will continually stress the fact that the field is inherently multidisciplinary in nature. Specialists in an uncount able number of areas are needed to produce a working functional design. Computer architects must interact with code writers and logic designers, and they must be able to comprehend some of the problems of circuit design and silicon processing. Electronics experts must move beyond cir cuits to see how their units will affect the system. And everyone depends upon the computer-aided design tools and the support groups that per form the 10,000 or so other tasks not described here. If this description

makes the field sound complicated, that's because it is. VLSI is not a simple discipline to understand. But it is possible to learn the basics in a reasonable amount of time. Persons who end up working in the area usually gravitate there because one or more aspects catch their interest and fall within their background.

Now that we have an appreciation of what is involved, let us move to a better description of the design process. An overview with the major steps in the sequence is shown in Figure 1.2. The starting point of a VLSI design is the system specification. At this point, the product is defined in both general and specific terms that provide design targets such as functions, speed, size, etc., for the entire project. This is the "Top" level of the design hierarchy. The system specifications are used to create an abstract, high-level model. Digital design is usually based on some type of **hardware description language** (HDL) that allows abstract modeling of the operation. VHDL and Verilog™ are the most common HDLs in practice, but several others (including C and C++) are used. The abstract model contains information on the behavior of each block and the interaction among the blocks in the system. The model is subjected to extensive verification steps where the design is checked and rechecked to ensure that it is correct.

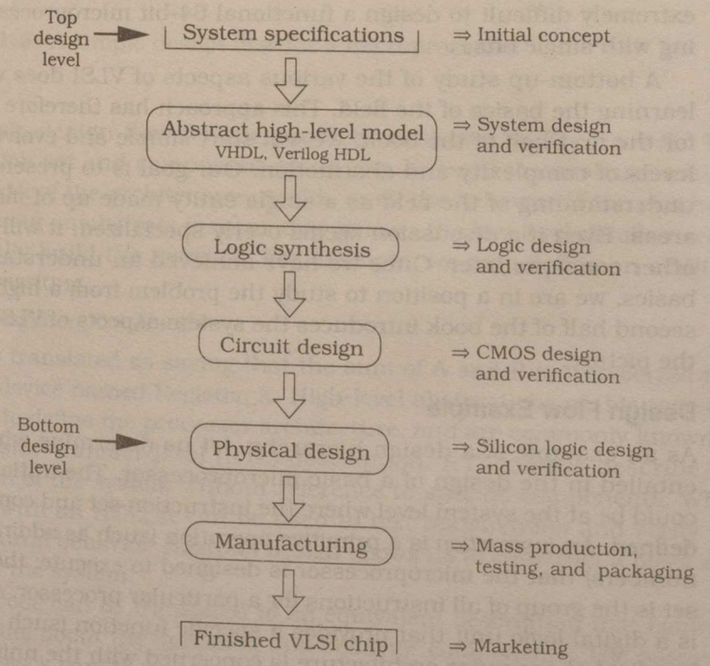


Figure 1.2 General overview of the design hierarchy

The next step in the process is called **synthesis**. The abstra model is used to provide the **logical design** of the network by sp the primitive gates and units needed to build each unit. This the the basis for transferring the design to the **electronic circui** where transistors are used as switches and Boolean variables are as varying voltage signals. To create a transistor, we move down level to that of **physical design**. At this level, the network is bu tiny area on a slice of silicon using a complex mapping schem translated transistors and wires into extremely fine-line patterns als and other materials. The physical design level constitutes the of the design hierarchy. After the design process is completed, the moves on to the manufacturing line. The final result is a finishe tronic VLSI chip.

When we start at the system level specification, the design proc called the **top-down** approach. The initial work is quite abstract an oretical and there is no direct connection to silicon until many step been completed. The reverse approach starts at the silicon or circu and builds primitive units such as logic gates, adders, and regist the first steps. These are combined to obtain larger and more co logic blocks, which are then used as building blocks in even designs. This **bottom-up** approach is acceptable for small projects the complexity of modern VLSI designs makes it impractical extremely difficult to design a functional 64-bit microprocessor by ing with single bits.

A bottom-up study of the various aspects of VLSI does work w learning the basics of the field. This approach has therefore been ch for the first half of the book. We will start simple and evolve into h levels of complexity and abstraction. Our goal is to present a cohe understanding of the field as a single entity made up of many diff areas. Even if a discussion seems overly specialized, it will be link other concepts later. Once we have achieved an understanding basics, we are in a position to study the problem from a higher leve second half of the book introduces the system aspects of VLSI to com the picture.

1.1.1 Design Flow Example

As an example of a design hierarchy, let us determine what woul entailed in the design of a basic microprocessor. The initial concep could be at the system level where the instruction set and componen defined. An *instruction* is a primitive operation (such as adding two b numbers) that the microprocessor is designed to execute; the instru set is the group of all instructions for a particular processor. A compo is a digital logic unit that provides a specific function (such as addi The field of computer architecture is concerned with the units that a up the computer and how they are connected together.

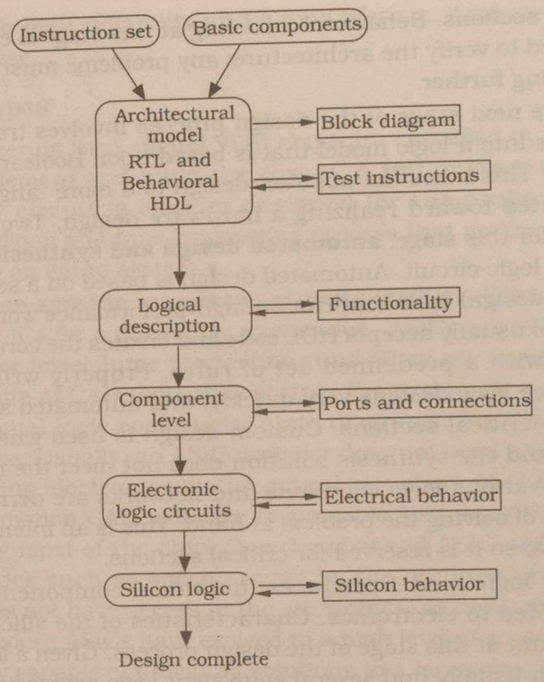


Figure 1.3 A simple design flow for a microprocessor

A basic design flow for the problem is shown in Figure 1.3. The instruction set and component group can be used to construct a high-level model of the architecture. At this level, the behavior of the system is described in an abstract manner that ignores the low-level details needed to actually build the network. For example, we may define an addition event by writing

Register_X ←A + B

which is translated as saying that the sum of A and B is transferred to a storage device named Register_X. High-level abstractions of this type can be used to define the processor architecture, and are commonly known as the register-transfer level (RTL) description. RTL models describe the operation of the system without reference to specific components. When written with an HDL, it can be used to test instructions and verify the architectural behavior Abstract design allows us to construct a block diagram for the system.

RTL code can be translated to an equivalent description that contains more detail about the operation and behavior components. The operation of each block can be summarized at the HDL **behavioral level**, where the emphasis is on the large-scale behavior of the blocks as they interact with

other sections. Behavioral modeling at this stage is extremely critic
is used to verify the architecture; any problems must be solved befo
gressing further.

The next stage of the design process involves translating the s
blocks into a logic model that is based upon Boolean equations and
gates. This takes the abstract design to a more tangible level, and
first step toward realizing a hardware design. Two approaches c
used for this stage: **automated design and synthesis**, or **custom d**
of the logic circuit. Automated design is based on a set of CAD (comp
aided design) tools that run on high-performance workstations. A sy
sis tool usually accepts HDL code and creates the corresponding logi
work with a predefined set of rules. Properly written HDL code
produce logic designs very quickly, and automated synthesis is use
all noncritical sections. Custom design is used when special probl
arise and the synthesis solution does not meet the necessary specif
tions. Various logic equations and networks are derived and tested
means of solving the problem at hand. This is an intense, time-consum
process, so it is reserved for critical sections.

The logic model produces functional components, which are t
translated to electronics. Characteristics of the silicon circuits beco
important at this stage of the design process. Given a large-scale funct
one can usually find several equivalent logic expressions; all produce
same output, but will use different equations and gates. Recall, for exa
ple, how a Karnaugh map is used to simplify logic equations. Silicon V
is complicated by the fact that each type of logic gate or circuit has d
tinct characteristics, and we must often search for circuits that are fas
or smaller than what can be obtained using an obvious solution.
sophisticated synthesis tool can take HDL code and provide suggest
designs for both the logic gates and the silicon circuits. However, t
toolsets have not yet reached the level where they are powerful enough
produce the "best" design, whatever that may mean.

After the logic network and circuits have been designed, the next ste
is to use the information to produce an integrated circuit at the physic
design level. This is accomplished in a series of steps where transistor
are defined as 3-dimensional structures on a chip of silicon, and are the
placed and wired using another set of graphical CAD tools. Once this
accomplished, the designs are tested and verified, and then used to creat
a database that allows the manufacturing line to actually build the elec
tronic chip. Fabricating a VLSI chip is itself a complex specialized field
Once started, it may take several weeks to produce the final circuits.

The specifics of the procedures are much more complicated than wha
is portrayed in the simple flowchart. It does, however, illustrate the
essence of a top-down design flow. VLSI design is concerned with filling in
the details needed to produce a manufactured chip that functions as

designed with high reliability and a long lifetime. And can be sold at a profit, of course.

1.1.2 VLSI Chip Types

At the engineering level, digital VLSI chips are classified by the approach used to implement and build the circuit. A **full-custom** design is one where every circuit is custom designed for the project. This is an extremely tedious and time-consuming process that makes it impractical for designing an entire system.

Application-specific integrated circuits (ASICs) allow digital designers to create ICs for a particular application. ASICs are very popular for prototyping or low-volume production runs. They are designed using an extensive suite of CAD tools that portray the system design in terms of standard digital logic constructs: state diagrams, function tables, and logic diagrams. Usually, an ASIC designer does not need any knowledge of the underlying electronics or the actual structure of the silicon chip. Design automation CAD tools are responsible for taking the logic design and building most of the chip. One drawback of ASIC design is that all characteristics such as speed are set by the architectural design; the designer does not have access to the electronics, so delay times cannot be changed. Modern ASICs have evolved to a high level of sophistication, and are generally capable of providing solutions to a large class of problems.

A **semi-custom** design is in between that of a full-custom and an ASIC-type circuit. The majority of the chip is designed using a group of primitive predefined cells as building blocks. Each cell provides a basic function, such as a logic operation or a storage circuit, and the master design resides in a database collection called a library. A cell entry contains all of the information needed to create the circuit on silicon. If it is not possible to meet the system specifications using the cell library, then the semi-custom approach permits the designer to engineer a solution by creating alternate silicon circuits that have the desired characteristics. These are used only in small sections where the problems occur. For example, floating point circuits in microprocessors can be extremely complex, so that some sections may require custom design to meet the clocking budget. Variations of semi-custom design are used for most high-performance chips.

1.2 Basic Concepts

The objective of this book is to present the field of VLSI in its entirety. Overall, VLSI design is a system design discipline. Many aspects can be taught without any reference to the underlying silicon circuits. System solutions can be generated using the CAD tools, and the necessary data turned over to the manufacturing group for production. While this

approach produces functional solutions, it makes many of the de
invisible to the designer. Simplifying the design process is import
However, many of the most powerful techniques and ideas of VLSI re
at lower levels and are therefore lost. Circuits work, but they are no
fast or as small as they could have been.

VLSI should be thought of as a single discipline that deals with
conception, design, and manufacture of complex integrated circu
Many system-level concepts are based on the characteristics of electr
circuits that are made at the silicon level. When Carver Mead of Cal
pioneered the field in the 1970's, one of the most important foundati
for VLSI arose from his observation that digital electronic integrated
cuits could be viewed as a set of geometrical patterns on the surface
silicon chip. Groups of patterns represented different logic functions
were repeated many times in the system. Complexity could thus be de
with using the concept of repeated patterns that were fitted together
structured manner. Signal flow and data movement could be followed
tracing the paths of the metallic "lines" that carried electricity. It was p
sible to write Boolean expressions that could be directly translated to g
metrical patterns on silicon in a well-defined manner. T
microphotograph of a CMOS chip in Figure 1.4 shows many of these fe
tures in a finished device. Note in particular the repeated patterns a
ordered placement of rectangular lines, polygons, and groups of geomet
patterns.[1] Mead's observation (and a huge amount of work) has stru
tured VLSI into the important field it is today. The importance of gainin
an overall unified view of VLSI becomes clear.

VLSI design encompasses many practical aspects of digital syste
design. One is the fact that even the most powerful **system on a chi**
(SOC) must be interfaced to other components to create an operation
unit. This is achieved by placing the silicon circuitry in the center of
rectangular piece of material, and then providing some type of scheme
that allows external wires to contact it. Figure 1.5 shows the use of **bon**
ing pads, which are square metal sections where wires can be bonde
and connected to the package in which the chip is mounted. A mor
advanced technique developed by IBM is called the C4 technology;
allows metal "bumps" to be located across the surface area. Contact
the package is established by "flipping" the chip so that the bumps are
the bottom and can be aligned to a wiring grid. Regardless of th
approach, there is a limit on the actual size of the chip.

In an ideal world, we could make the chip as large as desired. Thi
would allow increasingly complex systems to be designed without an
bounds. Unfortunately, it is not possible to manufacture a function

[1] This is a section of a binary adder network designed at Georgia Tech. Each group of patterns adds two bi
nd produces sum and carry outputs.

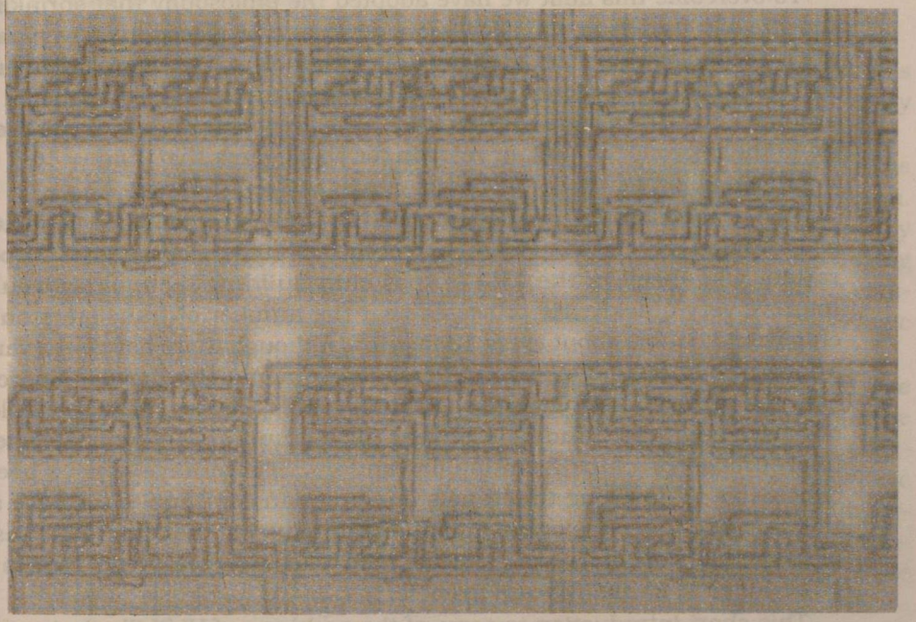


Figure 1.4 Micrograph of a section of a digital CMOS integrated circuit

design because of defects in the silicon crystal structure that cannot be avoided. The larger the area of the circuit, the higher the probability that a defect will occur. Even a single bad transistor or connection renders the circuit nonfunctional, so we attempt to keep the overall size of the chip small. We note in passing that other problems in manufacturing also limit the size of the chip.

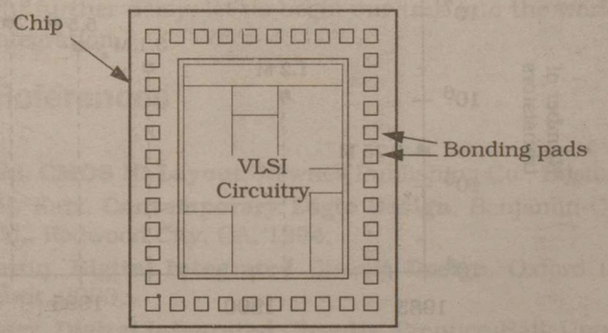


Figure 1.5 Bonding pad frame for interfacing

To overcome this limit we have adopted the philosophy that shrin
the size of a transistor will allow more devices to be placed in a given
area. Technologically, this is a very difficult problem. Precision desig
manufacturing techniques give devices where the smallest dimensi
around 1.3×10^{-7} meters. At this level, we change our measurement
ric to the micrometer (μm), or **micron** for short, such that 1 μm = 10
$= 10^{-4}$ cm, and refer to the technology as a 0.13-μm process.

One of the classical predictions in VLSI transistor densities is kn
as **Moore's Law**. Gordon Moore, one of the cofounders of the Intel Co
ration, visualized in the 1970's that chip building technology wo
improve very quickly. He projected that the number of transistors o
chip would double about every 18 months. Although there have been
ations due to technological problems or economic slowdowns, Moo
Law has proved amazingly close to actual trends. Figure 1.6 shows a
of device count as a function of year for a group of randomly selec
microprocessor chips from major vendors. There have always be
debates as to how long the transistor count can continue to increas
this rate due to technological limitations in reducing the size. Regardl
of the actual slope, however, it seems clear that VLSI design will remai
powerful force for many years to come.

This short introduction to some of the problems of VLSI illustrates t
vast nature of the field itself. The role of a VLSI design group is to creat
large, complex system on a tiny piece of silicon. The group faces co
straints at every level, from the abstract modeling and timing down
building a chip with millions of transistors. Project status presentation
engineering summaries, and critical deadlines are always present.

Welcome to the exciting world of VLSI!

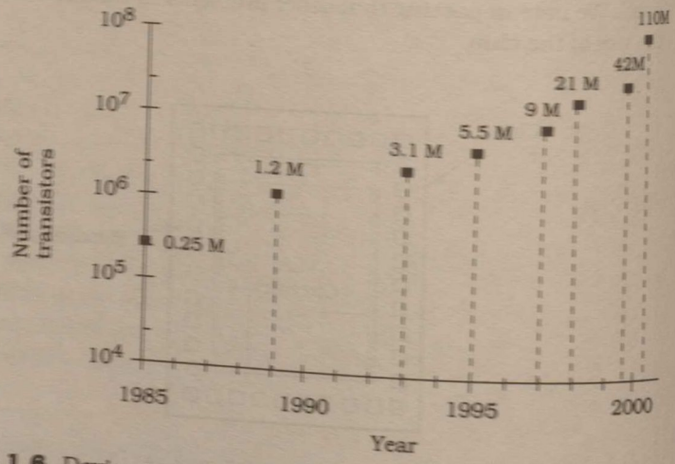


Figure 1.6 Device count by year

1.3 Plan of the Book

The book has been divided into three main sections. For self-study, it is best to follow them sequentially, although it is not necessary to read every section in a first reading.

Part 1 is entitled *Silicon Logic*, and includes Chapters 2 through 5. The material examines the techniques for designing logic networks in silicon. It concentrates on introducing transistor logic circuits and how they translate to patterns on a silicon chip. Details of the CMOS processing sequence are presented, and applied to realistic chip design. After completing Part 1, the reader will be able to design a myriad of CMOS logic gates at both the circuit and silicon levels.

The electronics of VLSI are covered in Part 2, which is entitled *The Logic-Electronics Interface*. Part 2 includes Chapters 6 through 9, with the more advanced concepts in Chapters 8 and 9. Transistor switching characteristics are presented and then used to analyze digital electronic logic gates. The treatment is quite detailed, but it concentrates on the important basics that affect system performance and switching speeds. Completing this section of the book will provide a solid understanding of the relationship between logic design and electrical characteristics.

System-level problems are addressed in Part 3, *The Design of VLSI Systems*, which includes Chapters 10 through 16. The basics of Verilog® HDL are presented as the vehicle for system-level modeling. Many VLSI logic components such a multiplexors, adders, and memories are studied in Chapters 11 through 13. Large-scale chip design issues are addressed in Chapters 14 and 15. The book concludes with an introduction to digital testing in the final chapter.

An effort has been made to present the material in a readable, coherent manner that concentrates on explaining the details. This is particularly true in Part 1, where the reader is exposed to subject matter that is not typically found in other courses.

So, without further delay, let us begin our trek into the world of very-large scale integration.

1.4 General References

[1] Dan Clein, **CMOS IC Layout**, Newnes Publishing Co., Boston, 2000.

[2] Randy H. Katz, **Contemporary Logic Design**, Benjamin-Cummings Publishing Co., Redwood City, CA, 1994.

[3] Ken Martin, **Digital Integrated Circuit Design**, Oxford University Press, New York, 2000.

[4] Jan Rabaey, **Digital Integrated Circuits**, Prentice-Hall, Upper Saddle River, NJ, 1996.

[5] Michael John Sebastian Smith, **Application-Specific Integrate cuits**, Addison-Wesley Longman Inc., Reading, MA, 1997.

[6] John P. Uyemura, **A First Course in Digital Systems De** Brooks-Cole Publishers, Pacific Grove, CA, 2000.

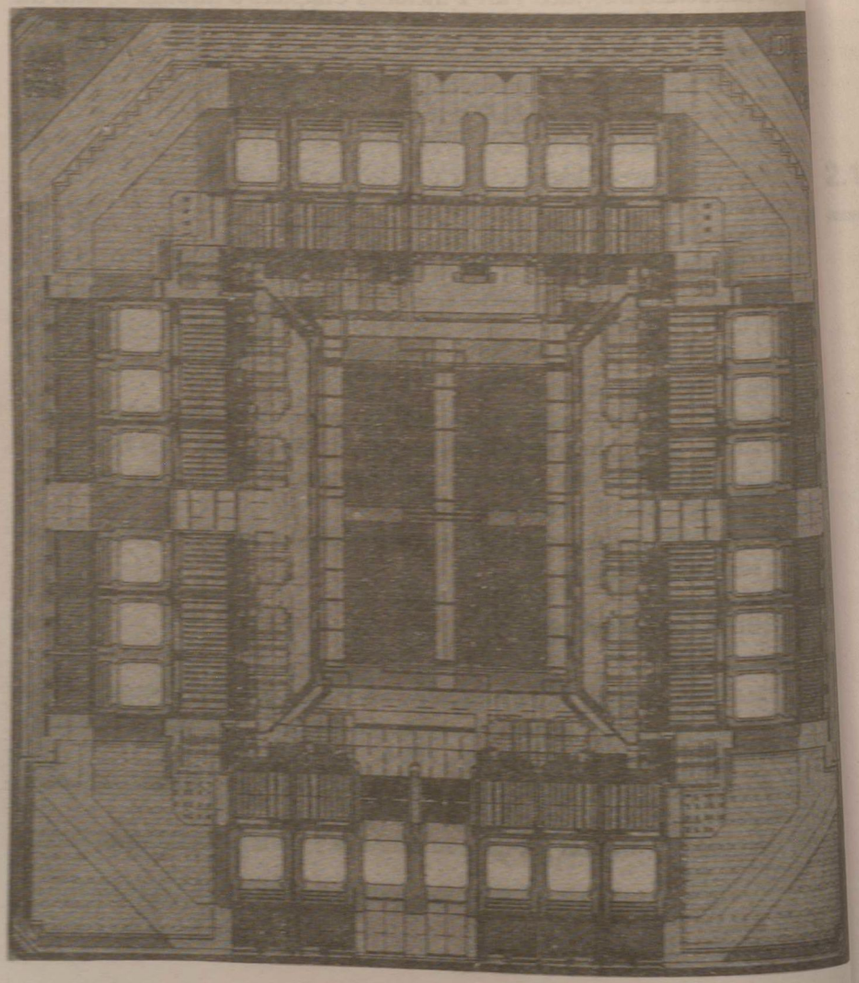
[7] John P. Uyemura, **CMOS Logic Circuit Design**, Kluwer Aca Press, Norwell, MA, 1999.

[8] John P. Uyemura, **Physical Design of CMOS Integrated Ci Using L-Edit®**, PWS /Brooks-Cole Publishers, Pacific Grove, CA, 199

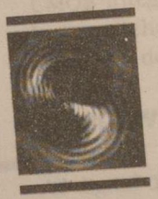
[9] M. Michael Vai, **VLSI Design**, CRC Press. Boca Raton, FL, 2001.

[10] Neil H.E. Weste and Kamran Eshraghian, **Principles of CMOS Design**, 2nd ed., Addison-Wesley Publishing Co., Reading, MA, 1993

[11] Wayne Wolf, **Modern VLSI Design**, 2nd ed., Prentice-Hall Upper Saddle River, NJ, 1998.



Part 1



Silicon Logic

commercial IC products have shown this trend quite clearly as shown in Figure 1.3 and, simultaneously, the effectiveness of the circuits produced has increased with scaling down. A common measure of effectiveness is the speed power product of the basic logic gate circuit of the technology (for nMOS, the *Nor* gate, with *Nand* and *Nor* gates for CMOS). Speed power product is measured in picojoules (pJ) and is the product of the gate switching delay in nanoseconds and the gate power dissipation in milliwatts. Typical figures are implied in Figure 1.2.

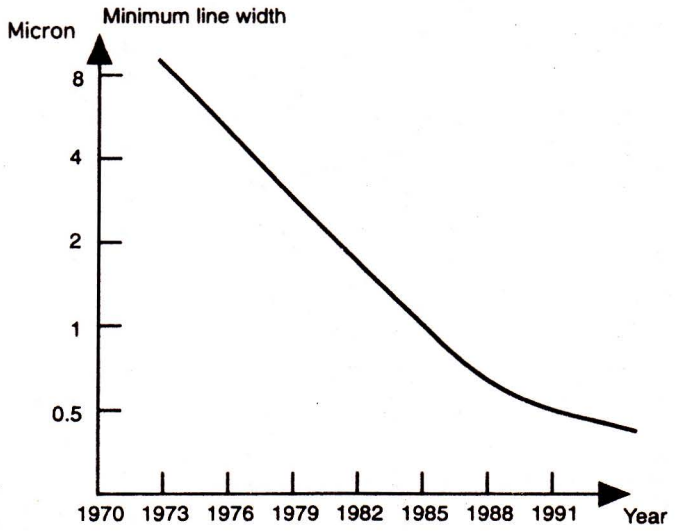


FIGURE 1.3 Approximate minimum line width of commercial products versus year.

1.4 BASIC MOS TRANSISTORS

Having now established some background, let us turn our attention to basic MOS processes and devices. In particular, let us examine the basic nMOS enhancement and depletion mode transistors as shown in Figures 1.4(a) and (b).

nMOS devices are formed in a p-type substrate of moderate doping level. The source and drain regions are formed by diffusing n-type impurities through suitable masks into these areas to give the desired n-impurity concentration and give rise to depletion regions which extend mainly in the more lightly doped p-region as shown. Thus, source and drain are isolated from one another by two diodes. Connections to the source and drain are made by a deposited metal layer. In order to make a useful device, there must be the capability for establishing and controlling a current between source and drain, and this is commonly achieved in one of two ways, giving rise to the enhancement mode and depletion mode transistors.

Consider the enhancement mode device first, shown in Figure 1.4(a). A polysilicon gate is deposited on a layer of insulation over the region between source and drain. Figure 1.4(a) shows a basic enhancement mode device in which the channel is not established and the device is in a non-conducting condition, $V_D = V_S = V_{gs} = 0$. If this gate is connected to a

suitable positive voltage with respect to the source, then the electric field established between the gate and the substrate gives rise to a charge inversion region in the substrate under the gate insulation and a conducting path or channel is formed between source and drain.

The channel may also be established so that it is present under the condition $V_{gs} = 0$ by implanting suitable impurities in the region between source and drain during manufacture and prior to depositing the insulation and the gate. This arrangement is shown in Figure 1.4(b). Under these circumstances, source and drain are connected by a conducting channel, but the channel may now be closed by applying a suitable negative voltage to the gate.

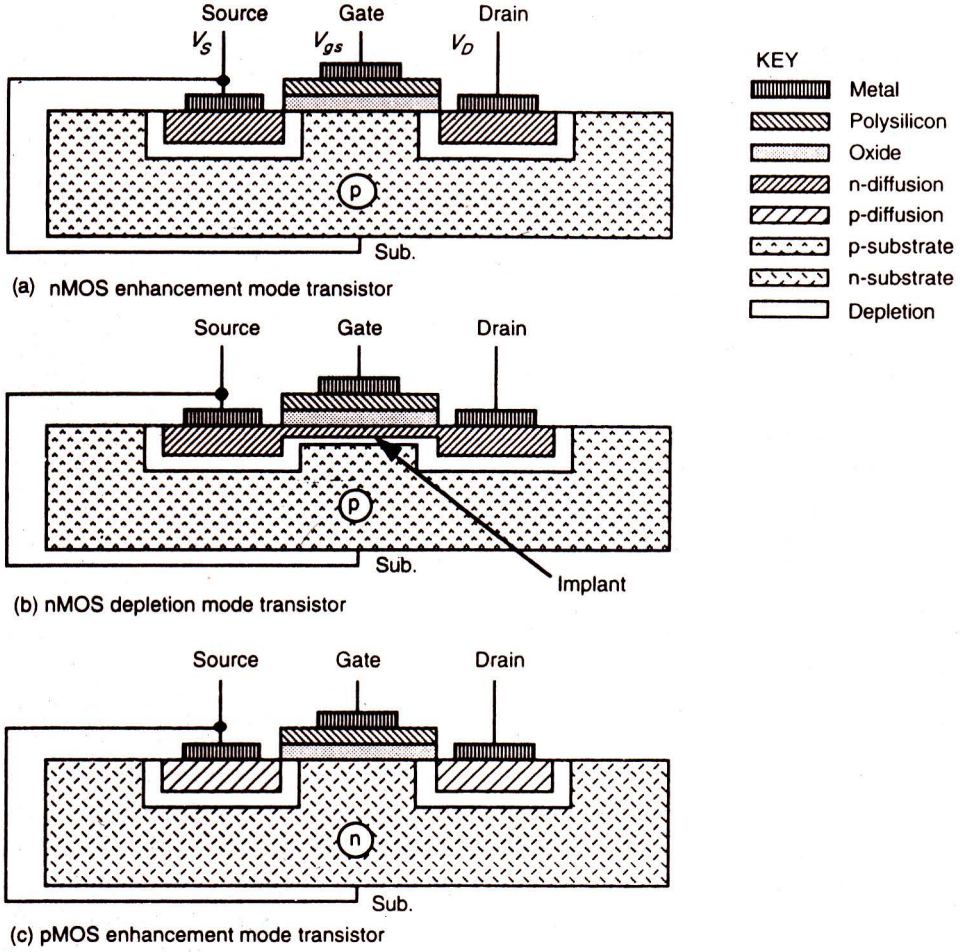


FIGURE 1.4 MOS transistors (V_D = 0 V. Source gate and substrate to 0 V).

In both cases, variations of the gate voltage allow control of any current flow between source and drain.

Figure 1.4(c) shows the basic pMOS transistor structure for an enhancement mode device. In this case the substrate is of n-type material and the source and drain diffusions are

consequently p-type. In the figure, the conditions shown are those for an unbiased device; however, the application of a *negative* voltage of suitable magnitude ($> |V_t|$) between gate and source will give rise to the formation of a channel (p-type) between the source and drain and current may then flow if the drain is made negative with respect to the source. In this case the current is carried by holes as opposed to electrons (as is the case for nMOS devices). In consequence, pMOS transistors are inherently slower than nMOS, since hole mobility μ_p is less, by a factor of approximately 2.5, than electron mobility μ_n. However, bearing these differences in mind, the discussions of nMOS transistors which follow relate equally well to pMOS transistors.

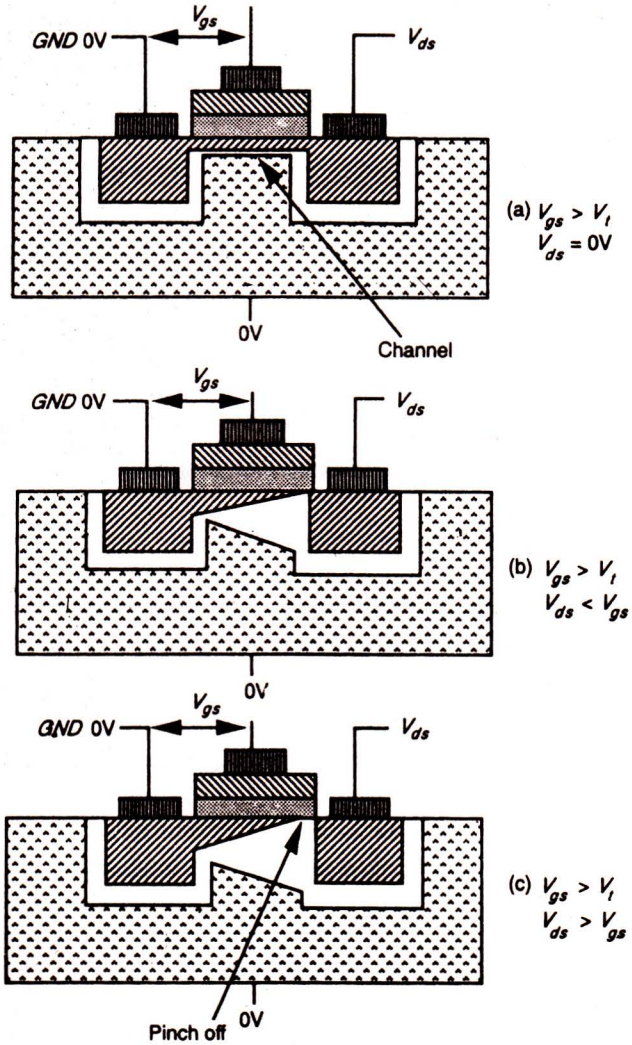
1.5 ENHANCEMENT MODE TRANSISTOR ACTION

To gain some understanding of this mechanism, let us further consider the enhancement mode device, as in Figure 1.5, under three sets of conditions. It must first be recognized that in order to establish the channel in the first place a minimum voltage level of *threshold voltage* V_t must be established between gate and source (and of course between gate and substrate as a result). Figure 1.5(a) then indicates the conditions prevailing with the channel established but no current flowing between source and drain ($V_{ds} = 0$). Now consider the conditions prevailing when current flows in the channel by applying a voltage V_{ds} between drain and source. There must, of course, be a corresponding IR drop = V_{ds} along the channel. This results in the voltage between gate and channel varying with distance along the channel with the voltage being a maximum of V_{gs} at the source end. Since the effective gate voltage is $V_g = V_{gs} - V_t$, (no current flows when $V_{gs} < V_t$) there will be voltage available to invert the channel at the drain end so long as $V_{gs} - V_t \geq V_{ds}$. The limiting condition comes when $V_{ds} = V_{gs} - V_t$. For all voltages $V_{ds} < V_{gs} - V_t$, the device is in the non-saturated region of operation which is the condition shown in Figure 1.5(b).

Consider now what happens when V_{ds} is increased to a level greater than $V_{gs} - V_t$. In this case, an IR drop = $V_{gs} - V_t$ takes place over less than the whole length of the channel so that over part of the channel, near the drain, there is insufficient electric field available to give rise to an inversion layer to create the channel. The channel is, therefore, 'pinched off' as indicated in Figure 1.5(c). Diffusion current completes the path from source to drain in this case, causing the channel to exhibit a high resistance and behave as a constant current source. This region, known as *saturation,* is characterized by almost constant current for increase of V_{ds} above $V_{ds} = V_{gs} - V_t$. In all cases, the channel will cease to exist and no current will flow when $V_{gs} < V_t$. Typically, for enhancement mode devices, $V_t = 1$ volt for $V_{DD} = 5$ V or, in general terms, $V_t = 0.2\ V_{DD}$.

1.6 DEPLETION MODE TRANSISTOR ACTION

For depletion mode devices the channel is established, due to the implant, even when $V_{gs} = 0$, and to cause the channel to cease to exist a negative voltage V_{td} must be applied between gate and source.



Note: V_{ds} is the drain-to-source voltage. Substrate assumed connected to 0 V.

FIGURE 1.5 Enhancement mode transistor for particular values of V_{ds} with ($V_{gs} > V_t$).

V_{td} is typically $< -0.8\ V_{DD}$, depending on the implant and substrate bias, but, threshold voltage differences apart, the action is similar to that of the enhancement mode transistor.

Commonly used symbols for nMOS and pMOS transistors are set out in Figure 1.6.

1.7 nMOS FABRICATION

A brief introduction to the general aspects of the polysilicon gate self-aligning nMOS fabrication process will now be given. As well as being relevant in their own right, the fabrication processes used for nMOS are relevant to CMOS and BiCMOS which may be viewed as

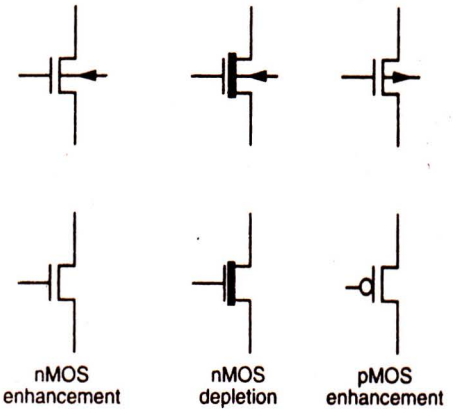


FIGURE 1.6 Transistor circuit symbols.

involving additional fabrication steps. Also, it is clear that an appreciation of the fabrication processes will give an insight into the way in which design information must be presented and into the reasons for certain performance characteristics and limitations. An nMOS process is illustrated in Figure 1.7 and may be outlined as follows:

1. Processing is carried out on a thin wafer cut from a single crystal of silicon of high purity into which the required p-impurities are introduced as the crystal is grown. Such wafers are typically 75 to 150 mm in diameter and 0.4 mm thick and are doped with, say, boron to impurity concentrations of $10^{15}/cm^3$ to $10^{16}/cm^3$, giving resistivity in the approximate range 25 ohm cm to 2 ohm cm.
2. A layer of silicon dioxide (SiO_2), typically 1 μm thick, is grown all over the surface of the wafer to protect the surface, act as a barrier to dopants during processing, and provide a generally insulating substrate on to which other layers may be deposited and patterned.
3. The surface is now covered with a photoresist which is deposited onto the wafer and spun to achieve an even distribution of the required thickness.
4. The photoresist layer is then exposed to ultraviolet light through a mask which defines those regions into which diffusion is to take place together with transistor channels. Assume, for example, that those areas exposed to ultraviolet radiation are polymerized (hardened), but that the areas required for diffusion are shielded by the mask and remain unaffected.
5. These areas are subsequently readily etched away together with the underlying silicon dioxide so that the wafer surface is exposed in the window defined by the mask.
6. The remaining photoresist is removed and a thin layer of SiO_2 (0.1 μm typical) is grown over the entire chip surface and then polysilicon is deposited on top of this to form the gate structure. The polysilicon layer consists of heavily doped polysilicon deposited by chemical vapor deposition (CVD). In the fabrication of fine pattern devices, precise control of thickness, impurity concentration, and resistivity is necessary.
7. Further photoresist coating and masking allows the polysilicon to be patterned (as shown in Step 6) and then the thin oxide is removed to expose areas into which

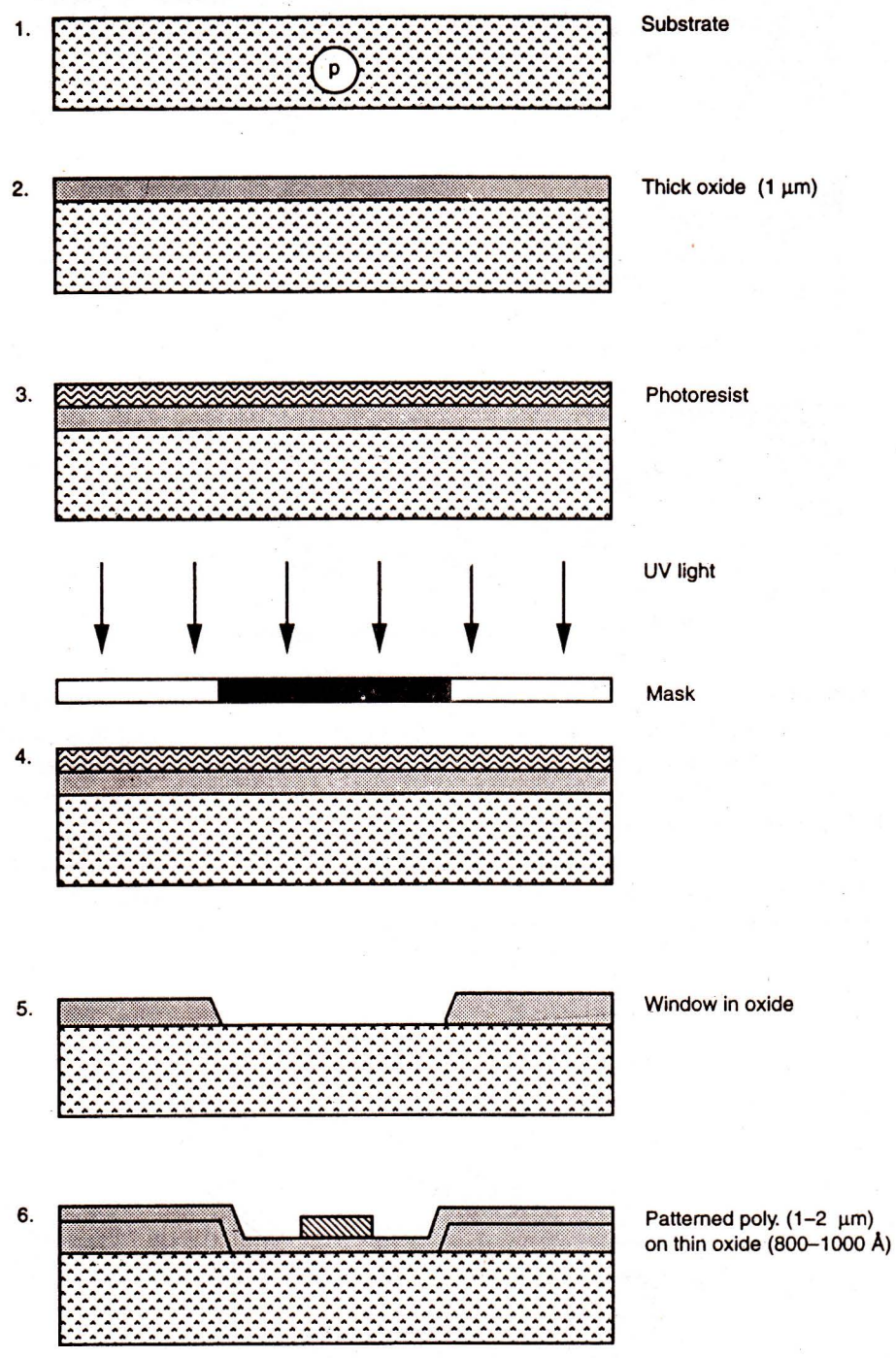


FIGURE 1.7 Continued

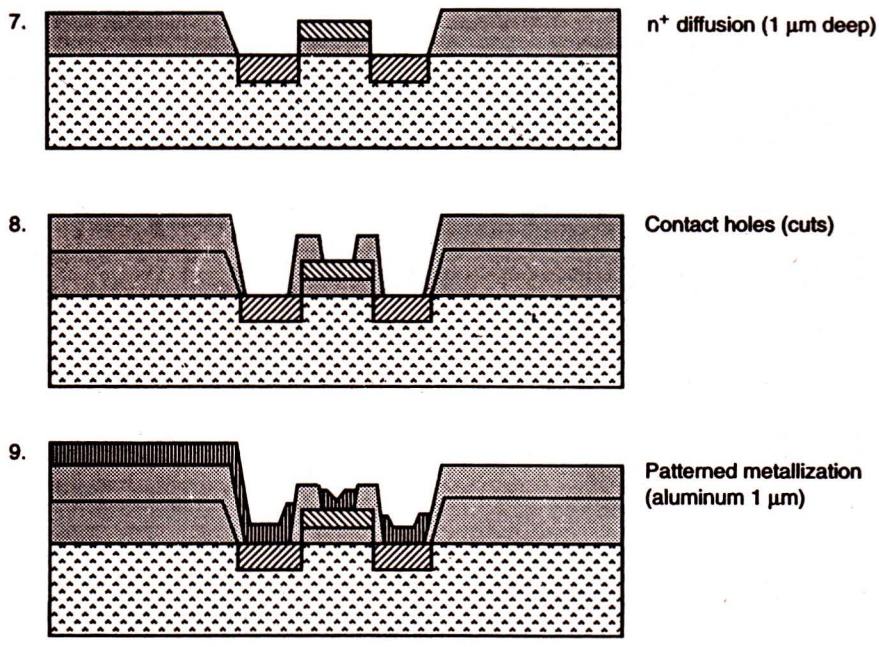


FIGURE 1.7 nMOS fabrication process.

n-type impurities are to be diffused to form the source and drain as shown. Diffusion is achieved by heating the wafer to a high temperature and passing a gas containing the desired n-type impurity (for example, phosphorus) over the surface as indicated in Figure 1.8. Note that the polysilicon with underlying thin oxide act as masks during diffusion—the process is self-aligning.

8. Thick oxide (SiO_2) is grown over all again and is then masked with photoresist and etched to expose selected areas of the polysilicon gate and the drain and source areas where connections (i.e. contact cuts) are to be made.

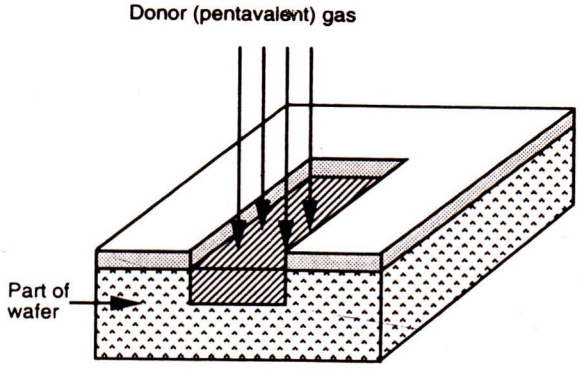


FIGURE 1.8 Diffusion process.

9. The whole chip then has metal (aluminum) deposited over its surface to a thickness typically of 1 μm. This metal layer is then masked and etched to form the required interconnection pattern.

It will be seen that the process revolves around the formation or deposition and patterning of three layers, separated by silicon dioxide insulation. The layers are diffusion within the substrate, polysilicon on oxide on the substrate, and metal insulated again by oxide.

To form depletion mode devices it is only necessary to introduce a masked ion implantation step between steps 5 and 6 or 6 and 7 in Figure 1.7. Again, the thick oxide acts as a mask and this process stage is also self-aligning.

Consideration of the processing steps will reveal that relatively few masks are needed and the self-aligning aspects of the masking processes greatly ease the problems of mask registration. In practice, some extra process steps are necessary, including the overglassing of the whole wafer, except where contacts to the outside world are required. However, the process is basically straightforward to envisage and circuit design eventually comes down to the business of delineating the masks for each stage of the process. The essence of the process may be reiterated as follows.

1.7.1 Summary of An nMOS Process

- Processing takes place on a p-doped silicon crystal wafer on which is grown a 'thick' layer of SiO_2.
- *Mask* 1—Pattern SiO_2 to expose the silicon surface in areas where paths in the diffusion layer or gate areas of transistors are required. Deposit thin oxide over all. For this reason, this mask is often known as the *'thinox' mask* but some texts refer to it as the *diffusion mask.*
- *Mask* 2—Pattern the ion implantation within the thinox region where depletion mode devices are to be produced—*self-aligning.*
- *Mask* 3—Deposit polysilicon over all (1.5 μm thick typically), then pattern using Mask 3. Using the same mask, remove thin oxide layer where it is not covered by polysilicon.
- Diffuse n^+ regions into areas where thin oxide has been removed. Transistor drains and sources are thus self-aligning with respect to the gate structures.
- *Mask* 4—Grow thick oxide over all and then etch for contact cuts.
- *Mask* 5—Deposit metal and pattern with Mask 5.
- *Mask* 6—Would be required for the overglassing process step.

1.8 CMOS FABRICATION

There are a number of approaches to CMOS fabrication, including the p-well, the n-well, the twin-tub, and the silicon-on-insulator processes. In order to introduce the reader to CMOS design we will be concerned mainly with well-based circuits. The p-well process is widely used in practice and the n-well process is also popular, particularly as it is an easy retrofit to existing nMOS lines, so we will also discuss it briefly.

For the lambda-based rules set out later, we will assume a p-well process.

1.8.1 The p-well Process

A brief overview of the fabrication steps may be obtained with reference to Figure 1.9, noting that the basic processing steps are of the same nature as those used for nMOS.

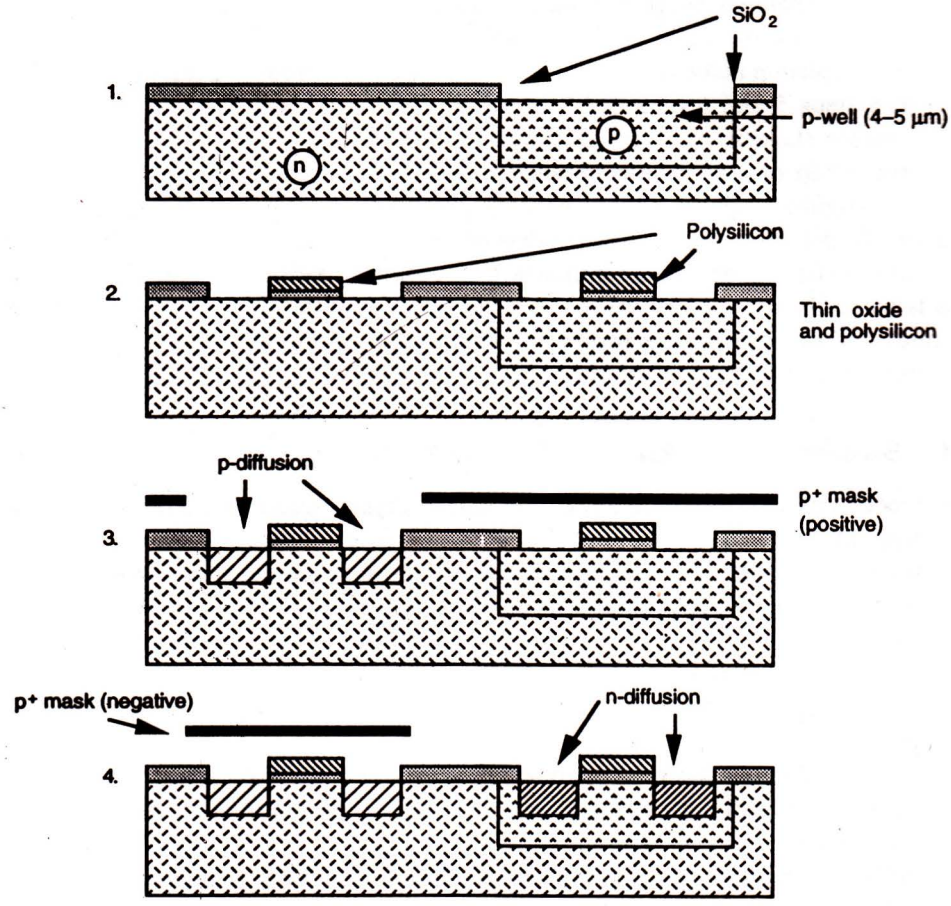


FIGURE 1.9 CMOS p-well process steps.

In primitive terms, the structure consists of an n-type substrate in which p-devices may be formed by suitable masking and diffusion and, in order to accommodate n-type devices, a deep p-well is diffused into the n-type substrate as shown.

This diffusion must be carried out with special care since the p-well doping concentration and depth will affect the threshold voltages as well as the breakdown voltages of the n-transistors. To achieve low threshold voltages (0.6 to 1.0 V) we need either deep-well diffusion or high-well resistivity. However, deep wells require larger spacing between the n- and p-type transistors and wires due to lateral diffusion and therefore a larger chip area.

The p-wells act as substrates for the n-devices within the parent n-substrate, and, provided that voltage polarity restrictions are observed, the two areas are electrically isolated. However,

since there are now in effect two substrates, two substrate connections (V_{DD} and V_{SS}) are required, as shown in Figure 1.10.

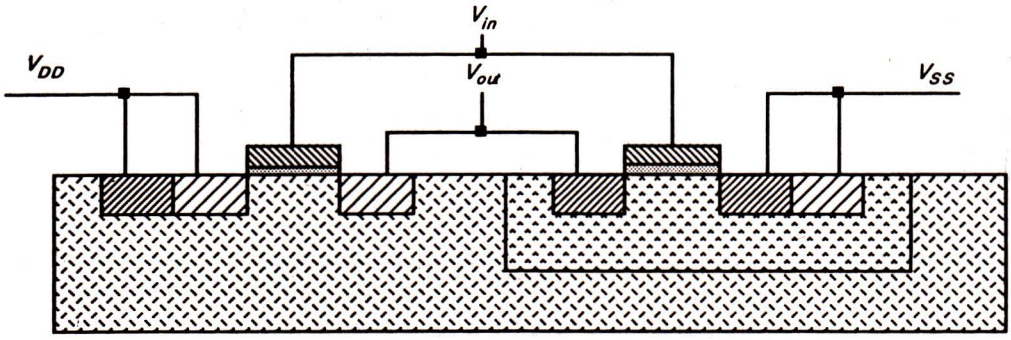


FIGURE 1.10 CMOS p-well inverter showing V_{DD} and V_{SS} substrate connections.

In all other respects—masking, patterning, and diffusion—the process is similar to nMOS fabrication. In summary, typical processing steps are:

- *Mask* 1 — defines the areas in which the deep p-well diffusions are to take place.
- *Mask 2* — defines the thinox regions, namely those areas where the thick oxide is to be stripped and thin oxide grown to accommodate p- and n-transistors and wires.
- *Mask* 3 — used to pattern the polysilicon layer which is deposited after the thin oxide.
- *Mask* 4 — A p-plus mask is now used (to be in effect "Anded" with Mask 2) to define all areas where p-diffusion is to take place.
- *Mask* 5 — This is usually performed using the negative form of the p-plus mask and defines those areas where n-type diffusion is to take place.
- *Mask* 6 — Contact cuts are now defined.
- *Mask* 7 — The metal layer pattern is defined by this mask.
- *Mask* 8 — An overall passivation (overglass) layer is now applied and Mask 8 is needed to define the openings for access to bonding pads.

1.8.2 The n-well Process

As indicated earlier, although the p-well process is widely used, n-well fabrication has also gained wide acceptance, initially as a retrofit to nMOS lines.

N-well CMOS circuits are also superior to p-well because of the lower substrate bias effects on transistor threshold voltage and inherently lower parasitic capacitances associated with source and drain regions.

Typical n-well fabrication steps are illustrated in Figure 1.11. The first mask defines the n-well regions. This is followed by a low dose phosphorus implant driven in by a high temperature diffusion step to form the n-wells. The well depth is optimized to ensure against p-substrate to p^+ diffusion breakdown without compromising the n-well to n^+ mask separation. The next steps are to define the devices and diffusion paths, grow field oxide, deposit and

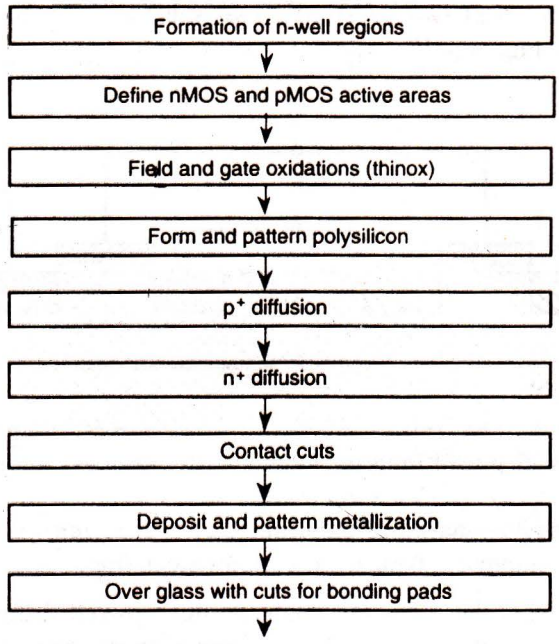


FIGURE 1.11 Main steps in a typical n-well process.

pattern the polysilicon, carry out the diffusions, make contact cuts, and finally metalize as before.

It will be seen that an n^+ mask and its complement may be used to define the n- and p-diffusion regions respectively. These same masks also include the V_{DD} and V_{SS} contacts (respectively). It should be noted that, alternatively, we could have used a p^+ mask and its complement, since the n^+ and p^+ masks are generally complementary.

By way of illustration, Figure 1.12 shows an inverter circuit fabricated by the n-well process, and this may be directly compared with Figure 1.10.

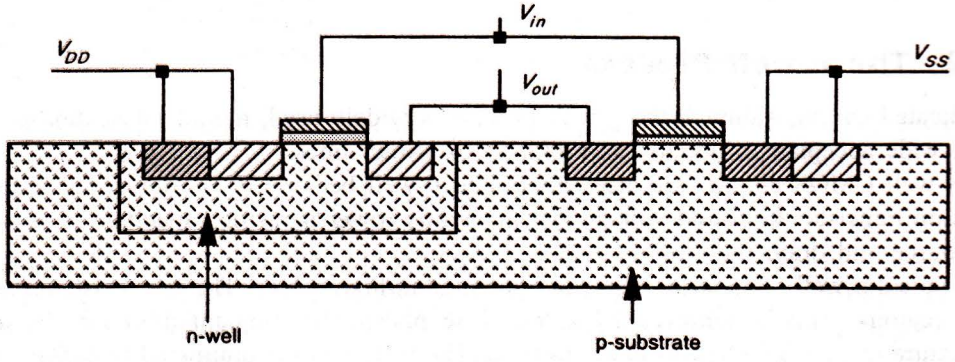


FIGURE 1.12 Cross-sectional view of n-well CMOS inverter.

Owing to differences in charge carrier mobilities, the n-well process creates non-optimum p-channel characteristics. However, in many CMOS designs (such as domino-logic and dynamic-logic structures), this is relatively unimportant since they contain a preponderance of n-channel devices. Thus the n-channel transistors are mainly those used to form logic elements, providing speed and high density of elements.

Latch-up problems can be considerably reduced by using a low-resistivity epitaxial p-type substrate as the starting material, which can subsequently act as a very low resistance ground-plane to collect substrate currents.

However, a factor of the n-well process is that the performance of the already poorly performing p-transistor is even further degraded. Modern process lines have come to grips with these problems, and good device performance may be achieved for both p-well and n-well fabrication.

The design rules which are presented for 1.2 μm and 2 μm technologies in this text are for Orbit™ n-well processes.

1.8.2.1 The Berkeley n-well process

There are a number of p-well and n-well fabrication processes and, in order to look more closely at typical fabrication steps, we will use the Berkeley n-well process as an example. This process is illustrated in Figure 1.13.

1.8.3 The Twin-Tub Process

A logical extension of the p-well and n-well approaches is the twin-tub fabrication process.

Here we start with a substrate of high resistivity n-type material and then create both n-well and p-well regions. Through this process it is possible to preserve the performance of n-transistors without compromising the p-transistors. Doping control is more readily achieved and some relaxation in manufacturing tolerances results. This is particularly important as far as latch-up is concerned.

In general, the twin-tub process allows separate optimization of the n- and p-transistors. The arrangement of an inverter is illustrated in Figure 1.14, which may, in turn, be compared with Figures 1.10 and 1.12.

1.9 THERMAL ASPECTS OF PROCESSING

The processes involved in making nMOS and CMOS devices have differing high temperature sequences as indicated in Figure 1.15.

The CMOS p-well process, for example, has a high temperature p-well diffusion process (1100 to 1250°C), the nMOS process having no such requirement. Because of the simplicity, ease of fabrication, and high density per unit area of nMOS circuits, many of the earlier IC designs, still in current use, have been fabricated using nMOS technology and it is likely that nMOS and CMOS system designs will continue to co-exist for some time to come.

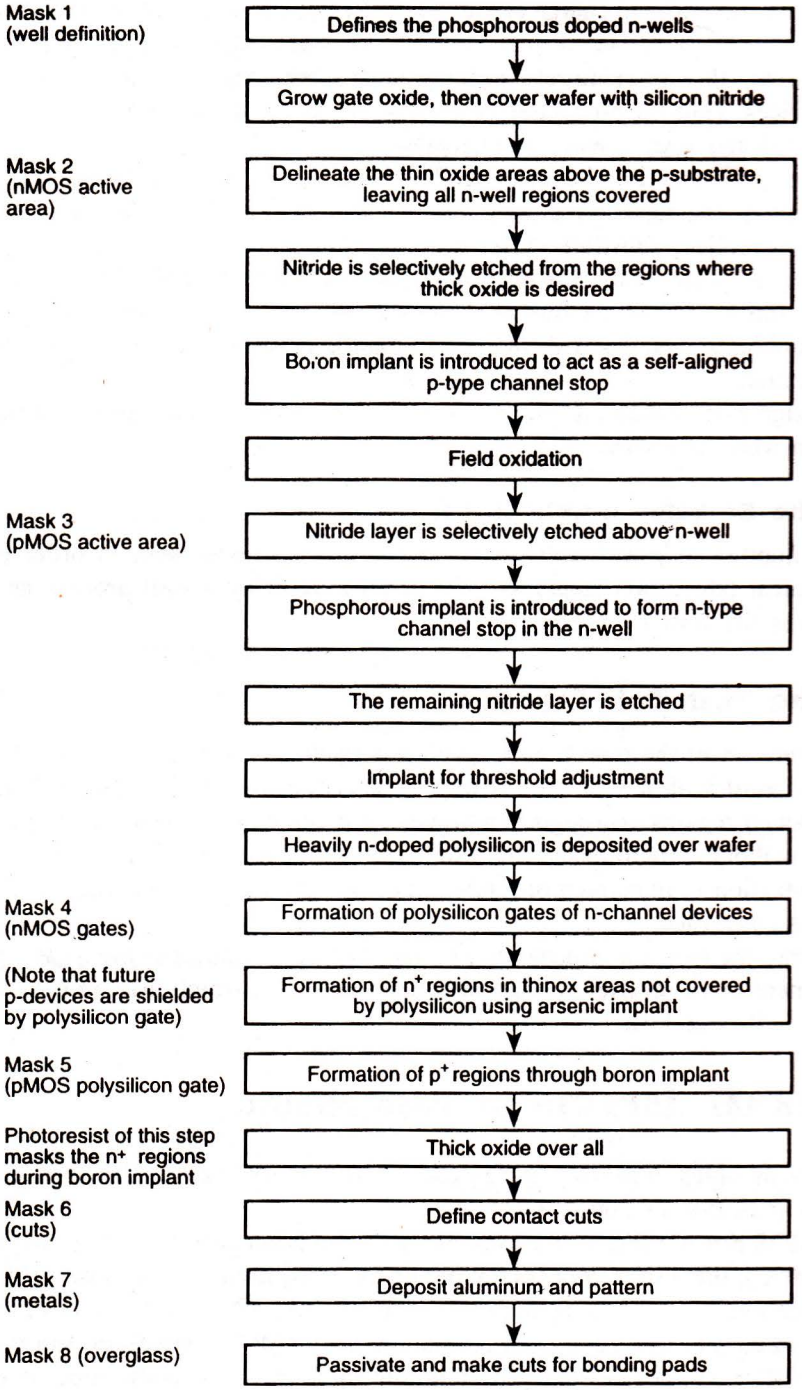


FIGURE 1.13 Flow diagram of Berkeley n-well fabrication.

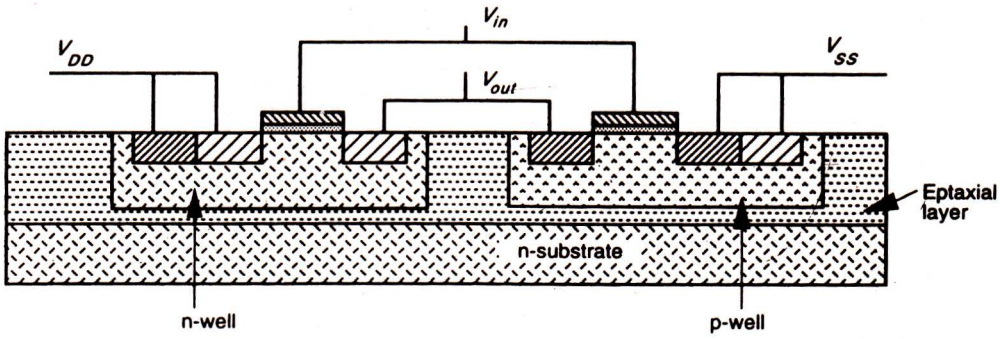


FIGURE 1.14 Twin-tub structure.

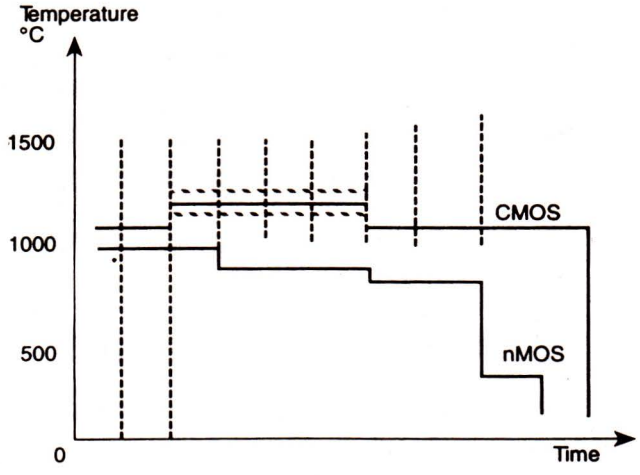


FIGURE 1.15 Thermal sequence difference between nMOS and CMOS processes.

1.10 BiCMOS TECHNOLOGY

A known deficiency of MOS technology lies in the limited load driving capabilities of MOS transistors. This is due to the limited current sourcing and current sinking abilities associated with both p- and n-transistors and although it is possible, for example, to design so called super-buffers using MOS transistors alone, such arrangements do not always compare well with the capabilities of bipolar transistors. Bipolar transistors also provide higher gain and have generally better noise and high frequency characteristics than MOS transistors and it may be seen (Figure 1.2) that BiCMOS gates could be an effective way of speeding up VLSI circuits. However, the application of BiCMOS in sub-systems such as ALU, ROM, a register-file, or, for that matter, a barrel shifter, is not always an effective way of improving speed. This is because most gates in such structures do not have to drive large capacitive loads so that the BiCMOS arrangements give no speed advantage. To take advantage of BiCMOS, the whole functional entity, not just the logic gates, must be considered. A comparison between the characteristics of CMOS and bipolar circuits is set out in Table 1.2 and the

TABLE 1.2 Comparison between CMOS and bipolar technologies

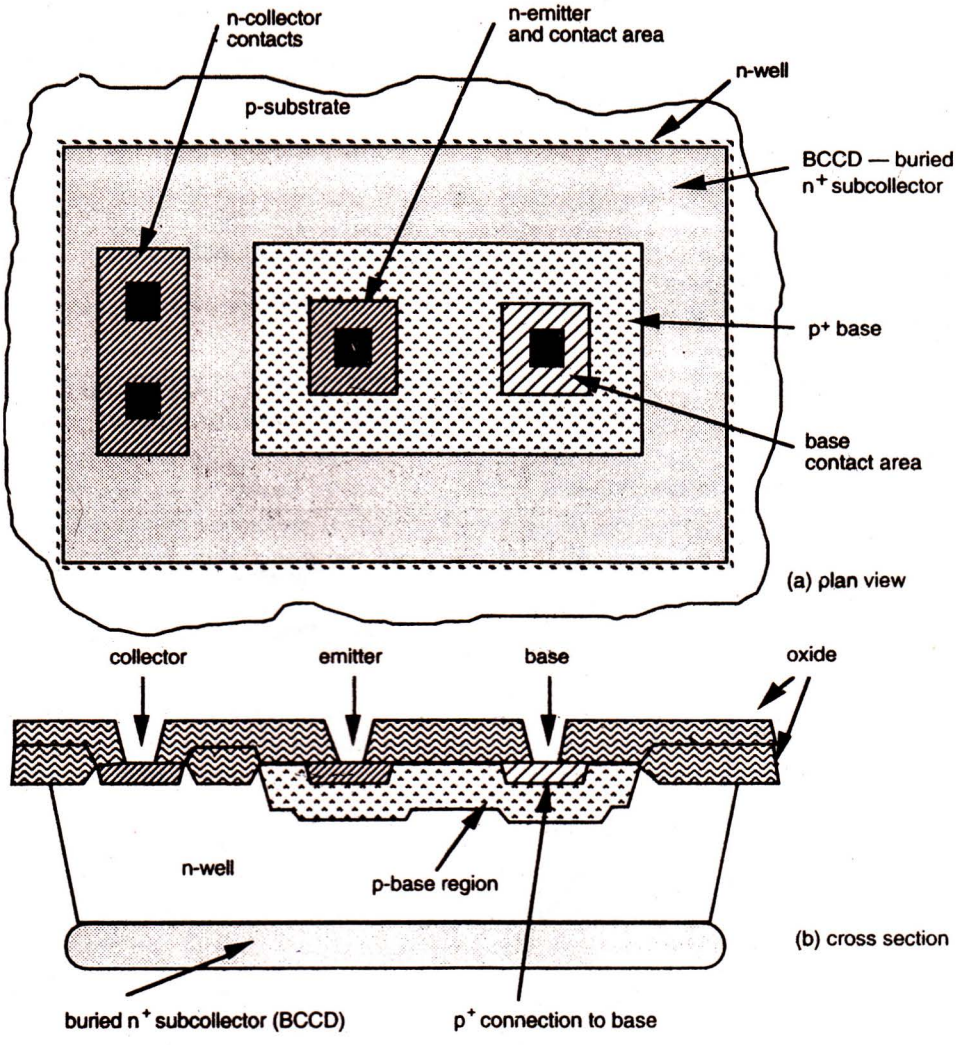
CMOS technology	*Bipolar technology*
• Low static power dissipation	• High power dissipation
• High input impedance (low drive current)	• Low input impedance (high drive current)
• Scalable threshold voltage	
• High noise margin	• Low voltage swing logic
• High packing density	• Low packing density
• High delay sensitivity to load (fan-out limitations)	• Low delay sensitivity to load
• Low output drive current	• High output drive current
• Low g_m ($g_m \alpha V_{in}$)	• High g_m ($g_m \alpha e^{V_{in}}$)
	• High f_t at low currents
• Bidirectional capability (drain and source are interchangeable)	• Essentially unidirectional
• A near ideal switching device	

differences are self-evident. BiCMOS technology goes some way towards combining the virtues of both technologies.

When considering CMOS technology, it becomes apparent that theoretically there should be little difficulty in extending the fabrication processes to include bipolar as well as MOS transistors. Indeed, a problem of p-well and n-well CMOS processing is that parasitic bipolar transistors are inadvertantly formed as part of the outcome of fabrication. The production of npn bipolar transistors with good performance characteristics can be achieved, for example, by extending the standard n-well CMOS processing to include further masks to add two additional layers—the n^+ subcollector and p^+ base layers. The npn transistor is formed in an n-well and the additional p^+ base region is located in the well to form the p-base region of the transistor. The second additional layer—the buried n^+ subcollector (BCCD) is added to reduce the n-well (collector) resistance and thus improve the quality of the bipolar transistor. The simplified general arrangement of such a bipolar npn transistor may be appreciated with regard to Figure 1.16. Bipolar transistor characteristics will follow in Chapter 2 and the relevant design rules are dealt with in Chapter 3. A quick appraisal of Figures 3.13(f) will serve to further illustrate the actual geometry of a BiCMOS bipolar transistor in n-well technology. Since extra design and processing steps are involved, there is an inevitable increase in cost and this is reflected in Figure 1.17; which also includes ECL and GaAs gates for cost comparison.

1.10.1 BiCMOS Fabrication in an n-well Process

The basic process steps used are those already outlined for CMOS but with additional process steps and additional masks defining: (i) the p^+ base region; (ii) n^+ collector area; and (iii) the buried subcollector (BCCD).



Note: For clarity, the layers have not been drawn transparent but BCCD underlies the entire area and the p^+ base underlies all within its boundary.

FIGURE 1.16 Arrangement of BiCMOS npn transistor (Orbit 2 μm CMOS).

Table 1.3 sets out the process steps for a single poly, single Metal CMOS n-well process, showing the additional process steps for the bipolar devices.

1.10.2 Some Aspects of Bipolar and CMOS Devices

Clearly there are relative advantages and disadvantages when comparing bipolar technology with CMOS technology. A readily assimilated comparison of some key features was set out as Table 1.2.

TABLE 1.3 n-well BiCMOS fabrication process steps

Single poly. single metal CMOS	*Additional steps for bipolar devices*
• Form n-well	
	• Form buried n^+ layer (BCCD)
• Delineate active areas	
• Channel stop	
	• Form deep n^+ collector
• Threshold V_t adjustment	
• Delineate poly./gate areas	
• Form n^+ active areas	
• Form p^+ active areas	
	• Form p^+ base for bipolars
• Define contacts	
• Delineate the metal areas	

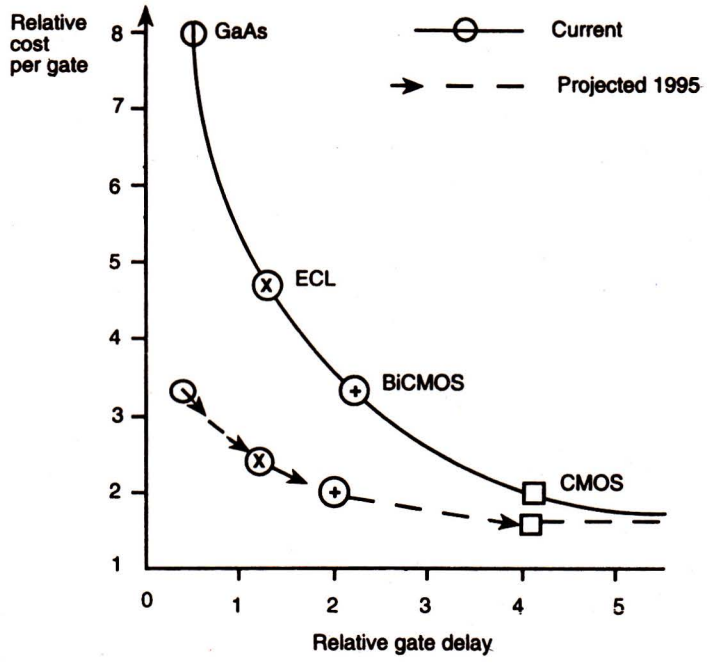


FIGURE 1.17 Cost versus delay for logic gate.

It will be seen that there are several advantages if the properties of CMOS and bipolar technologies could be combined. This is achieved to a significant extent in the BiCMOS technology. As in all things, there is a penalty which, in this case, arises from the additional process steps, some loss of packing density and thus higher cost.

A cost comparison of all current high speed technologies may be assessed from Figure 1.17.

A further advantage which arises from BiCMOS technology is that analog amplifier design is facilitated and improved. High impedance CMOS transistors may be used for the input circuitry while the remaining stages and ouput drivers are realised using Bipolar transistors.

To take maximum advantage of available Silicon technologies one might envisage the following mix of technologies in a silicon system.

CMOS	for logic
BiCMOS	for I/O and driver circuits
ECL	for Critical high speed parts of the system

However, in this text we will not be dealing with the ECL technology.

1.11 PRODUCTION OF E-BEAM MASKS

All the processes discussed have made use of masks at various stages of fabrication. In many processes, the masks are produced by standard optical techniques and much has been written on the photolithographic processes involved. However, as geometric dimensions shrink and also to allow for the processing of a number of different chip designs on a single wafer, other techniques are evolving. One popular process used for this purpose uses an E-beam machine. A rough outline of this type of mask making follows:

1. The starting material consists of chrome-plated glass plates which are coated with an E-beam sensitive resist.
2. The E-beam machine is loaded with the mask description data (MEBES).
3. Plates are loaded into the E-beam machine, where they are exposed with the patterns specified by the customer's mask data.
4. After exposure to the E-beam, the plates are introduced into a developer to bring out the patterns left by the E-beam in the resist coating.
5. The cycle is followed by a bake cycle and a plasma de-summing, which removes the resist residue.
6. The chrome is then etched and the plate is stripped of the remaining E-beam resist.

The advantages of E-beam masks are:

- tighter layer to layer registration;
- smaller feature sizes.

There are two approaches to the design of E-beam machines:

- raster scanning;
- vector scanning.

In the first case, the electron beam scans all possible locations (in a similar fashion to a television display) and a bit map is used to turn the E-beam on and off depending on whether the particular location being scanned is to be exposed or not.

For vector scanning, the beam is directed only to those locations which are to be exposed. Although this is inherently faster, the data handling involved is more complex.

1.12 OBSERVATIONS

This chapter has set the scene by introducing the basically simple MOS transistor structures and the relatively straightforward fabrication processes used in the manufacture of nMOS, CMOS and BiCMOS circuits. We have also attempted to emphasize the revolutionary spread of semiconductor technology which has, in the short space of 30 years, advanced to a point where we now see complex systems completely integrated onto a single chip.

Although this text concentrates on digital circuits and systems, similar techniques can be applied to the design and fabrication of analog devices. Indeed, the trends are toward systems of VLSI (and beyond) complexity which will in future include, on single chips, significant analog interfaces and other appropriate circuitry. This higher level of integration will lead to fewer packages and interconections and to more complex systems than today. There will be a marked beneficial effect on cost and reliability of the systems that will be available to all professions and disciplines and in most aspects of everyday life.

Our discussions of fabrication have in some instances simplfied the processes used in order to reveal or emphasize the essential features. Indeed, the fabrication of similar devices by different fabricators may vary considerably in detail. This is also the case with the design rules (Chapter 3) which are specified by the fabricator. Design rules will be introduced via the concept of "lambda-based" rules, which are a result of the work of Mead and Conway and, although not producing the tightest layouts, these rules are acceptable to many fabricators. A study of lambda-based rules also provide a good way of absorbing the essential concepts underlying any set of design rules. However, the text also gives an up-to-date set of real world "micron-based" rules for 2 μm and for 1.2 μm n-well CMOS technologies which may be used when the designer reaches an acceptable level of competence. The 2 μm rule set is for a BiCMOS process and thus also provides for bipolar npn transistors. It must be noted here that "2 μm technology", for example, means that the minimum line width (and, consequently, the typical feature size of the geometry) of the chip layout will be 2 μm.

In order to understand the basic features MOS and BiCMOS IC technologies, we must now look into the basic electrical properties.

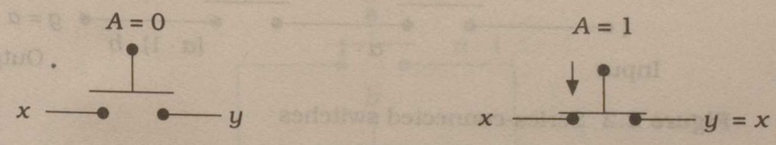
Logic Design with MOSFETs 2

CMOS integrated circuits use bi-directional devices called MOSFETs as logic switches. This chapter examines the logical characteristics of MOSFETs and develops techniques for building digital networks.

2.1 Ideal Switches and Boolean Operations

All digital designs are based on primitive logic operations. The first task in our study of VLSI will be to create electronic logic gates that can be used as building blocks in complex switching networks.

Logic gates are created by using sets of **controlled switches**. The characteristics of an **assert-high** controlled switch are illustrated by the drawings in Figure 2.1. In this idealized situation, the state of the switch (**open** or **closed**) is determined by the value of the control variable *A*. In Figure 2.1(a), the control bit has the value of $A = 0$ which is defined to give an open switch. This means that there is no relationship between the two variables x and y as represented by the gap between the left and right sides. The opposite case is a closed switch where we visualize the top portion of the switch being "pushed down" as shown in Figure 2.1(b). This condition occurs when $A = 1$ and connects the two sides of the switches



(a) Open (b) Closed

Figure 2.1 Behavior of an assert-high switch

so that

$$y = x$$

is valid. If we interpret the left side variable x as the input and the
side as the output, then we can say that the condition $A = 1$ allow
input variable to flow through the switch and establish the value
output. This is called an "assert-high" switch because a high control
$A = 1$ is required to close the circuit.

A different approach to characterizing the behavior of the switch
write the logic equation[1]

$$y = x \cdot A \quad \text{iff} \quad A = 1$$

By itself, the relationship between x and y is undefined if $A = 0$. Alth
this appears to be a serious deficiency, in practice we will avoid the p
lem by using additional switches to define the value of y for this case.

Let us now proceed to create a logic network by combining the con
of an ideal switch with a voltage source. Suppose that we take
switches that are controlled by the independent variables a and b
connect them as shown by the diagram in Figure 2.2. The two switch
are said to be **in series** with each other. As we trace the signal p
through the first switch, equation (2.2) shows that the output (dire
after the switch) is given by $a \cdot 1$ as indicated on the drawing. This act
the input to the second switch, so that applying equation (2.2) ag
yields

$$g = (a \cdot 1) \cdot b = a \cdot b$$

for the output. This is easy to interpret using a qualitative analysis
switches must be closed with $a = 1$ AND $b = 1$ to allow the input
reach the output and result in $g = 1$. The circuit appears to provide
AND2 operation.[2] However, note that equation (2.2) is valid only if
control bit has a value of 1; it if is 0, then there is no direct relation
between the left and right sides of the switch. There are three other po

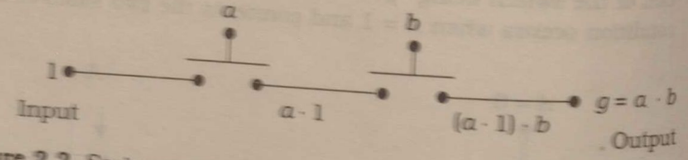


Figure 2.2 Series-connected switches

[1] We use the shorthand mathematical notation "iff" to mean "if, and only if".

[2] We denote a 2-input AND operation as an AND2. This type of notation will be used for all gates. For e
ple, an OR2 operation implies a 2-input OR.

bilities for the two inputs:

$$(a, b) = (1, 0), (0, 1), (0, 0) \qquad (2.4)$$

Any of these input combinations should result in a logical output of $g = 0$ but the logic equations say that g is undefined.

Before proceeding any further, let us clarify our approach to portraying logic networks. In general, the switch drawings will be called **schematic diagrams** since they show the "scheme" used in the wiring. We extend this terminology to include diagrams that contain electronic devices. Since we want to keep the drawings relatively compact and neat looking, wiring lines will often cross one another in the drawing. When this occurs, we will adopt the convention shown in Figure 2.3. In Figure 2.3(a), Wire 1 and Wire 2 are assumed to be totally separate. The signal a on Wire 1 has no relationship to the signal b on Wire 2. If we wish to create a connection, we will use a "dot" as in Figure 2.3(b). In this case, the two wires are connected so that placing a signal a on one of the lines results in the same value on all points of both lines.

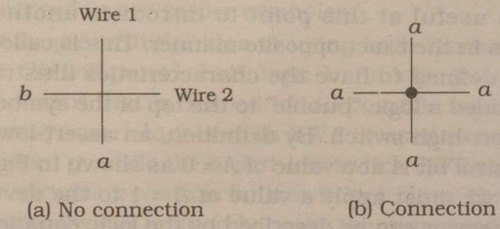


Figure 2.3 Connection convention used in schematic diagrams

Let us examine another circuit that has the same problem. Figure 2.4 shows two switches that are controlled by the independent variables a and b, but the two are wired **in parallel** with one another; this means that the left (input) sides are connected together and the right (output) sides are connected together. The output f can be constructed by recog-

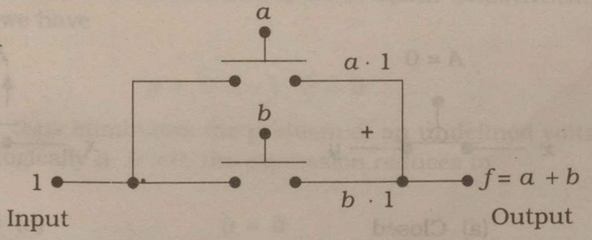


Figure 2.4 Parallel-connected switches

nizing that according to equation (2.2), the top switch produces an o
of $a \cdot 1$ iff $a = 1$, while the lower switch produces an output of $b \cdot 1$
1. Both expressions are shown at the appropriate points in the diag
We conclude that if either $a = 1$ OR $b = 1$ (or both), then the outp
described by the single expression

$$g = a + b$$

which appears to be the OR2 operation at this point in the analysis.
allel-connected switches can thus be used to OR variables together; th
indicated on the diagram by including the "+" between the switches.
however, that if $a = 0$ and $b = 0$ at the same time, then the output g o
switching network is undefined. It thus fails to provide the entire
function.

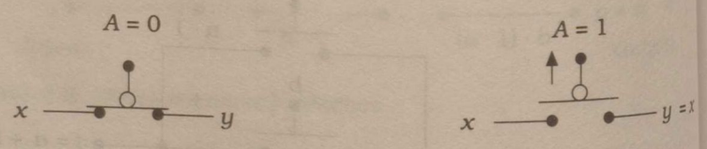
The preceding examples illustrate that switches have characteris
that can be used as a basis for implementing logic operations. Howe
since the logic equation (2.2) is valid only if the switch is closed, we
not able to obtain complete AND and OR gates as neither network co
produce a logic 0 output.

It is useful at this point to introduce another type of switch
behaves in the exact opposite manner. This is called an **assert-low** sw
and is defined to have the characteristics illustrated in Figure 2.5
have added a logic "bubble" to the top of the symbol to distinguish it
an assert-high switch. By definition, an assert-low switch is closed w
the control bit is at a value of $A = 0$ as shown in Figure 2.5(a). To open
switch we must apply a value of $A = 1$ to the device as in Figure 2.5
This behavior can be described by the logic equation

$$y = x \cdot \bar{A} \qquad \text{iff } A = 0$$

In this case, the value of y is not defined if $A = 1$. Comparing the two ty
of switches we see that they behave in a complementary manner.

As an example of how this type of switch can be used, consider
series-connected pair in Figure 2.6. Tracing the signal path from
input through the first switch gives an output of $\bar{a} \cdot 1$ which is valid iff
0. This acts as the input to the second switch so that the output of



(a) Closed (b) Open

Figure 2.5 An assert-low switch

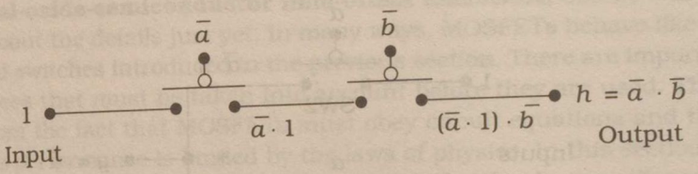


Figure 2.6 Series-connected complementary switches

series chain is given by

$$h = (\bar{a} \cdot 1) \cdot \bar{b} = \overline{a + b} \tag{2.7}$$

where we have used the DeMorgan relation to write the second line. This looks like the NOR2 operation. However, since the second switch must be closed with $b = 0$, this result is correct only if both $a = 0$ and $b = 0$. If either a or b is a 1, then g is undefined. We thus have the same type of problem experienced in our earlier examples.

Let us now progress to the idea of using both types of switches in a single network. We will provide both logic 1 and logic 0 inputs in an effort to produce an output that is defined for all possible input combinations. In Figure 2.7, the assert-high switch SW1 is used to connect a logic 0 input to the output y, while the assert-low switch SW2 connects a logic 1 input to y. The variable a controls both switches. Since the two are in parallel, we may write the OR relation between the upper and lower branches to give the output in the form

$$y = \bar{a} \cdot 1 + a \cdot 0 \tag{2.8}$$

The operation of the circuit can be understood by specifying a value for a. If $a = 0$, then SW1 is open and SW2 is closed which gives

$$y = \bar{0} \cdot 1 + 0 \cdot 0 = 1 \tag{2.9}$$

If $a = 1$, then SW1 is closed and SW2 is open. Substituting into the expression we have

$$y = \bar{1} \cdot 1 + 1 \cdot 0 = 0 \tag{2.10}$$

This circuit thus eliminates the problem of an undefined voltage. Moreover, since logically $a \cdot 0 = 0$, the expression reduces to

$$y = \bar{a} \tag{2.11}$$

In other words, this circuit implements the NOT operation.

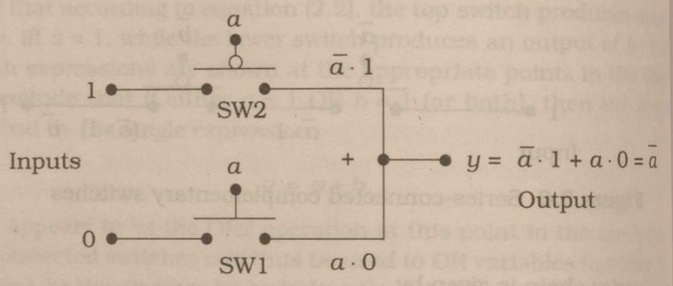


Figure 2.7 A switch-based NOT gate

$$y = \text{NOT}(a) = \bar{a} \qquad (2.$$

This demonstrates that using two switches with opposite characteristi allows us to build a network with well-defined results.

The NOT circuit in Figure 2.7 is based on the behavior of a 2:1 mu plexor as shown in Figure 2.8. The MUX uses the input a to sel between input 0 (that has a "1" applied to it) when $a = 0$, or input 1 (t has a "0" applied to it) when $a = 1$. The output is given by the expressio

$$y = \bar{a} \cdot 1 + a \cdot 0 \qquad (2.$$

which reduces to $y = \bar{a}$. A close examination of the switching circuit Figure 2.7 verifies that there is a one-to-one correspondence with the multiplexor.

2.2 MOSFETs as Switches

MOSFETs are electronic devices that are used to direct and control lo signals in high-density digital IC design.[3] The acronym "MOSFET" sta

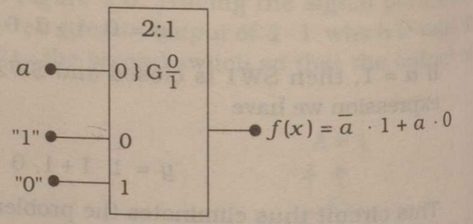


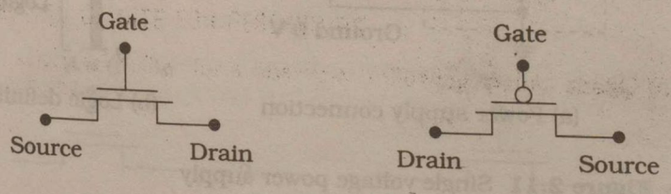
Figure 2.8 A MUX-based NOT gate

[3] "MOSFET" is pronounced as *moss-fet*.

for **metal-oxide-semiconductor field-effect transistor**, but we will not worry about the details just yet. In many ways, MOSFETs behave like the idealized switches introduced in the previous section. There are important differences that must be taken into account before they are used. These arise from the fact that MOSFETs must obey circuit equations and their ultimate performance is limited by the laws of physics. In this section we will concentrate on creating switching models for the devices. The more complicated aspects of current flow will be discussed in later chapters.

Complementary MOS (**CMOS**) uses two types of MOSFETs to create logic networks. One type is called an n-channel MOSFET (or nFET for short), and uses negatively charged electrons for electrical current flow. The circuit symbol for an nFET is shown in Figure 2.9(a). The **gate** terminal acts as the control electrode for the device. Applying a voltage on the gate electrode determines the current flow between the **drain** and **source** terminals. The other type of transistor is called a p-channel MOSFET or pFET. It uses positive charges for current flow and has the circuit symbol drawn in Figure 2.9(b). The only graphical difference between the nFET and pFET symbols is the inversion bubble at the gate. As with the nFET, the voltage applied to the gate determines the current flow between the source and drain terminals. Do not confuse the **gate terminal** of a MOSFET with a **logic gate**, as the two "gates" are not related. The context of the discussion always helps clarify the usage.

MOSFETs are intrinsically electronic devices. To use them as logic-controlled switches, we must first define how to translate between Boolean values and electrical parameters. This is accomplished by using voltages that exist on the chip when we apply an external power supply. In the most general case, two power supply voltages V_{DD} and V_{SS} are defined as shown in Figure 2.10. The reference terminal is taken to be the **ground** connection (which is at 0 volts) between the two sources so that the chip receives both a positive power voltage V_{DD} and a negative power supply voltage V_{SS}. Early generations of silicon MOS logic circuits used both positive and negative supply voltages. However, modern designs require only a single positive voltage V_{DD} and the ground connection; common values are $V_{DD} = 5$ V and 3.3 V or lower. The remaining source is set to a value of



(a) nFET symbol (b) pFET symbol

Figure 2.9 Symbols used for nFETs and pFETs

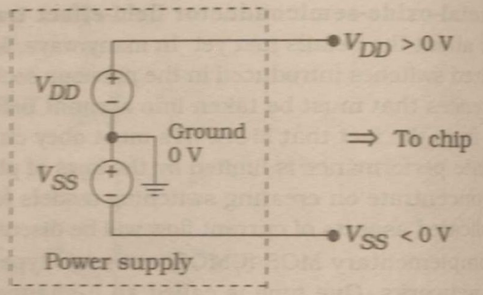


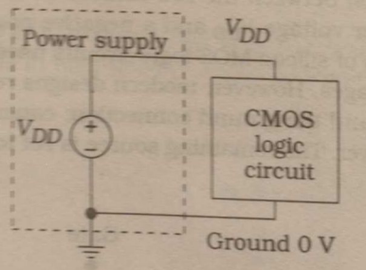
Figure 2.10 Dual power supply voltages

$V_{SS} = 0$ V, which results in the power supply network portrayed in F 2.11(a).[4] We will assume that all of our circuits use only a single po voltage source V_{DD}. In practice it is still common to use V_{SS} to denot lowest voltage in the circuit such that V_{SS} has an implied value of 0 V.

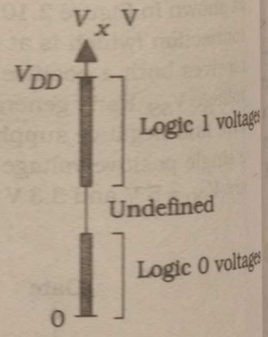
We can now define the relationship between logic variables and ages. Recall that Boolean variables are discrete; a binary variable have the value of $x = 0$ or $x = 1$ only. At the circuit level we represe variable x using a voltage V_x such that

$$0 \le V_x \le V_{DD} \qquad (2$$

gives the normal range of values with a power supply providing 0 V V_{DD} directly to the circuit. The **positive logic convention** then de the **ideal** logic 0 and logic 1 voltages as



(a) Power supply connection



(b) Logic definitions

Figure 2.11 Single voltage power supply

[4] The unit of volt is denoted by V in the text.

$$x = 0 \text{ means that } V_x = 0 \text{ V}$$
$$x = 1 \text{ means that } V_x = V_{DD} \qquad (2.15)$$

Realistic circuits are more lenient and allow us to use a *range* of voltages for both logic 0s and logic 1s as portrayed in Figure 2.11(b). In general,

- Low voltages correspond to logic 0 values
- High voltages correspond to logic 1 values

The transition region between the highest logic 0 voltage and the lowest logic 1 voltage is undefined in that it does not represent either a 0 or a 1. The actual extent of both voltage ranges is determined by the characteristics of the logic circuits and will be dealt with later.

With the logic-voltage conversion defined, let us now examine the switching characteristics of MOSFETs. Ideally, an nFET behaves like an assert-high switch. This is shown in Figure 2.12 where A is the logic variable applied to the gate. If $A = 0$ corresponding to a low voltage, then the nFET acts like an open switch and there is no relationship between the left and right sides; this is illustrated in Figure 2.12(a). Increasing the gate voltage to a high value is the same as changing to $A = 1$. This results in a closed switch as shown in Figure 2.12(b). As with the assert-high switch, this can be described by the logic equation

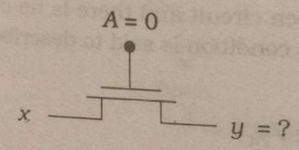
$$y = x \cdot A \qquad (2.16)$$

which is valid iff $A = 1$.

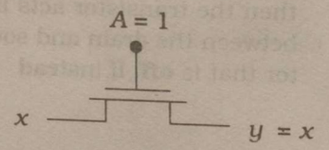
The pFET is exactly opposite in that it behaves like an assert-low switch. In Figure 2.13(a), the signal applied to the gate has a logical value of $A = 1$ corresponding to a high voltage. This gives an open circuit and there is no direct relationship between x and y. If the gate voltage is reduced to give $A = 0$, then the pFET acts as a closed switch. This allows us to write the ideal relationship

$$y = x \cdot \bar{A} \qquad (2.17)$$

which is valid so long as $A = 0$ is true; this condition is shown in Figure 2.13(b).

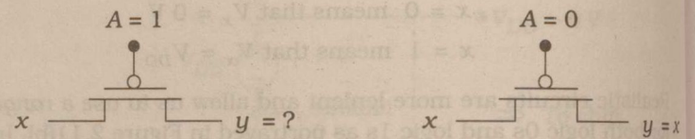


(a) Open



(b) Closed

Figure 2.12 nFET switching characteristics



(a) Open (b) Closed

Figure 2.13 pFET switching characteristics

MOSFETs allow us to design logic circuits using the techniqu assert-high and assert-low switching networks. However, FETs are p cal devices that do not behave exactly like the ideal switch models ab This is not a severe problem so long as we understand the differences learn the limitations.

FET Threshold Voltages

The switching equations assume that the binary variable A applied to gate of a FET is either a 0 or a 1. The corresponding voltage V_A is a p cal quantity and does not behave in such a discrete manner. Moreover want to define a range of voltages for both cases of $A = 0$ and $A = 1$ to in the design of working circuits. Every MOSFET has a characteri parameter called the **threshold voltage** V_T that helps us define important gate voltage ranges. The specific value of V_T is established d ing the manufacturing process, and is thus taken to be a given valu the VLSI designer. One complicating factor is that nFETs and pFETs h different threshold voltages.

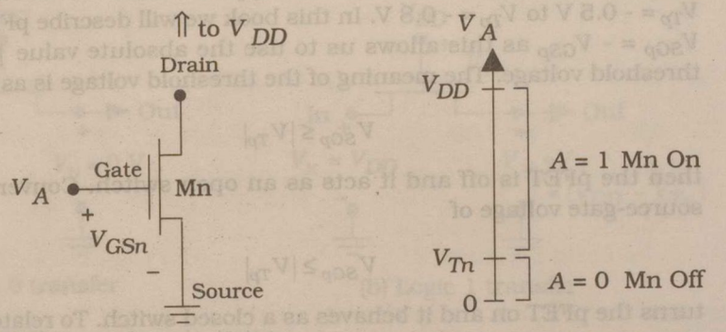
An nFET is characterized by a threshold voltage V_{Tn} that is a pos number with values around $V_{Tn} = 0.5$ V to 0.7 V being typical. The m ing of V_{Tn} can be understood by referring to the parameters shown in ure 2.14(a). First note that the drain terminal has been identified as one closest to the power supply V_{DD}, while the source terminal has b connected to ground (0 V). The gate-source voltage V_{GSn} shown in drawing is the important parameter that determines whether the n acts as an open or closed switch. In particular, if

$$V_{GSn} \le V_{Tn} \qquad (2$$

then the transistor acts like an open circuit and there is no current between the drain and source; this condition is said to describe a tra tor that is **off**. If instead

$$V_{GSn} \ge V_{Tn} \qquad (2$$

then the nFET drain and source are connected and the equivalent s is closed. A transistor that conducts current is said to be **on**. This be



(a) Gate-source voltage (b) Logic translation

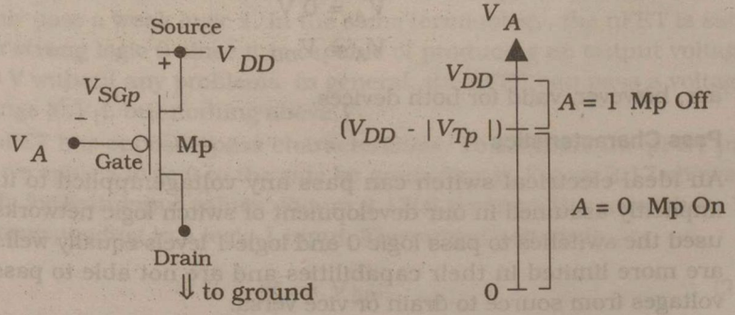
Figure 2.14 Threshold voltage of an nFET

ior allows us to create the voltage diagram shown in Figure 2.14(b) to define the voltage V_A that is associated with the binary variable A. In particular, we note that

$$V_A = V_{GSn} \qquad (2.20)$$

This shows that $A = 0$ corresponds to values of $V_A \le V_{Tn}$, while $A = 1$ implies that $V_A \ge V_{Tn}$. These relations establish the voltage ranges needed to control the nFET.

A pFET behaves in a **complementary** manner. Consider the transistor shown in Figure 2.15(a). For the pFET, the source terminal has been connected to the power supply V_{DD} while the drain is the side closest to ground; this is opposite to that used for the nFET. In this device, the source-gate voltage V_{SGp} is the important applied voltage. By convention, the pFET threshold voltage V_{Tp} is referenced to the gate-source voltage



(a) Source-gate voltage (b) Logic translation

Figure 2.15 pFET threshold voltage

V_{GSp} and is a negative number with typical values in the range of ab $V_{Tp} = -0.5$ V to $V_{Tp} = -0.8$ V. In this book we will describe pFETs by us $V_{SGp} = -V_{GSp}$ as this allows us to use the absolute value $|V_{Tp}|$ for threshold voltage. The meaning of the threshold voltage is as follows.

$$V_{SGp} \leq |V_{Tp}| \quad (2.$$

then the pFET is off and it acts as an open switch. Conversely, a la source-gate voltage of

$$V_{SGp} \geq |V_{Tp}| \quad (2.$$

turns the pFET on and it behaves as a closed switch. To relate this beh ior to the applied voltage V_A we first sum voltages to write

$$V_A + V_{SGp} = V_{DD} \quad (2.$$

Thus,

$$V_A = V_{DD} - V_{SGp} \quad (2.$$

shows that a low value of V_A implies a large V_{SGp} and the pFET is on. Si ilarly, if V_A is large then V_{SGp} is small and the pFET is off. This gives the logic 0 and logic 1 ranges summarized in Figure 2.15(b). Note that t transition between a logic 0 and a logic 1 is at

$$V_{DD} - |V_{Tp}| \quad (2.$$

since this corresponds to the source-gate voltage where the device tur on.

It is important to note that the logic 0 and logic 1 voltage ranges of are different for the two types of FETs. One way around this problem is note that there are regions of overlap for both $A = 0$ and $A = 1$ values th can be used if a uniform definition is needed. The ideal values of

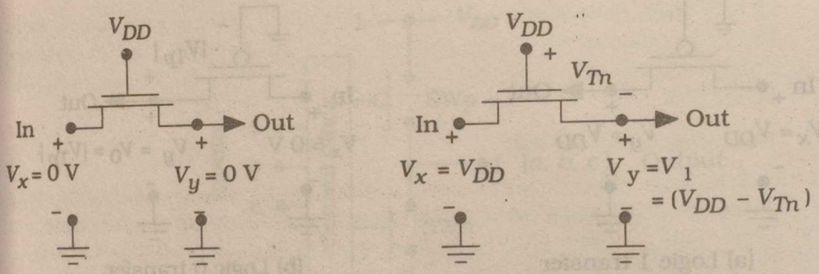
$$V_A = 0 \text{ V} \quad (2.$$
$$V_A = V_{DD}$$

are, however, valid for both devices.

Pass Characteristics

An ideal electrical switch can pass any voltage applied to it. This w implicitly assumed in our development of switch logic networks where used the switches to pass logic 0 and logic 1 levels equally well. MOSFE are more limited in their capabilities and are not able to pass arbitra voltages from source to drain or vice versa.

Let us examine the pass characteristics of nFETs first. Figure 2. summarizes the behavior of the device when we attempt to use it to pass



(a) Logic 0 transfer (b) Logic 1 transfer

Figure 2.16 nFET pass characteristics

voltage from left to right. Applying V_{DD} to the gate insures that the nFET is on and the device acts like a closed switch. In Figure 2.16(a) a logic 0 voltage of $V_x = 0$ V is applied to the left side. This results in an output voltage of $V_y = 0$ V as desired. Increasing the input voltage results in that value being transmitted to the output side. However, a problem occurs if we apply an ideal logic 1 input voltage of $V_x = V_{DD}$ as shown in Figure 2.16(b). In this case, the output voltage V_y is reduced to a value

$$V_1 = V_{DD} - V_{Tn} \quad (2.27)$$

which is less than the input voltage V_{DD}. This is referred to as a **threshold voltage loss**. It arises from the fact that the minimum value of the gate-source voltage need to maintain an on state is

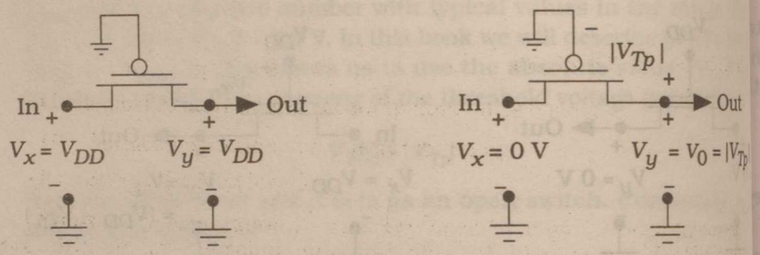
$$V_{GSn} = V_{Tn} \quad (2.28)$$

Using the Kirchhoff voltage law, this subtracts from the voltage V_{DD} that is applied to the *gate* as shown in the drawing.[5] Since the transmitted voltage V_y is less than the ideal logic 1 value of V_{DD}, we say that the nFET can only pass a **weak** logic 1. In the same terminology, the nFET is said to pass a **strong** logic 0 since it is capable of producing an output voltage of $V_y = 0$ V without any problems. In general, the nFET can pass a voltage in the range $[0, V_1]$, but nothing above V_1.

A pFET has opposite pass characteristics. To examine the pFET properties we apply a logic 0 to the gate by grounding it. Figure 2.17 shows the circuits for both input values. Figure 2.17(a) portrays the case where $V_x = V_{DD}$ corresponding to a logic 1 input. The output voltage is

$$V_y = V_{DD} \quad (2.29)$$

[5] Kirchhoff's voltage law (KVL) says that the algebraic sum of voltages around a closed loop is 0.



(a) Logic 1 transfer (b) Logic 0 transfer

Figure 2.17 pFET pass characteristics

which is an ideal logic 1 level. The pFET is therefore capable of pass strong logic 1 voltage. The problem arises when we attempt to pa ideal logic 0 voltage of $V_y = 0$ V and is presented in Figure 2.17(b). I case, the transmitted voltage can only drop to a minimum value of

$$V_y = |V_{Tp}|$$

This is also due to a threshold effect. In order to keep the pFE requires a minimum source-gate voltage of

$$V_{SGp} = |V_{Tp}|$$

as shown in the drawing. Since the gate is at 0 V, this represents a n a voltage of $|V_{Tp}|$, which is turn affects the output. Obviously, the transmits a weak logic 0 voltage. In summary, a pFET can pass a vo in the range $[V_{DD}, V_0]$, but nothing below V_0.

Let us restate the results of the above discussion:

- nFETs pass strong logic 0 voltages, but weak logic 1 values
- pFETs pass strong logic 1 voltages, but weak logic 0 levels

Complementary MOS (CMOS) circuits are designed to account fo transmission levels. In particular, we can write down the following as a basis for our design:

1. Use pFETs to pass logic 1 voltages of V_{DD}
2. Use nFETs to pass logic 0 voltages of $V_{SS} = 0$ V

These allow us to build circuits that can pass the ideal logic voltage and V_{DD} to the output terminal. We will find, however, that ideal level not always needed in practice.

2.3 Basic Logic Gates in CMOS

The concept of a general CMOS digital logic gate can be understo referring to the drawing in Figure 2.18. In this example, *a*, *b*, and *c* a

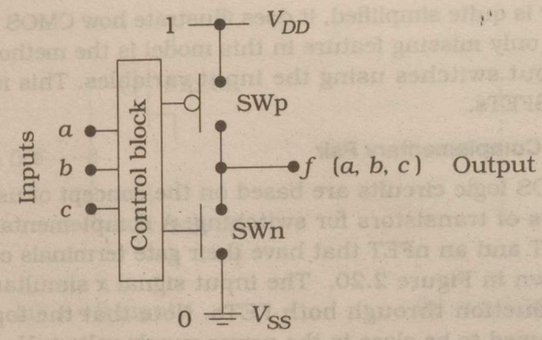
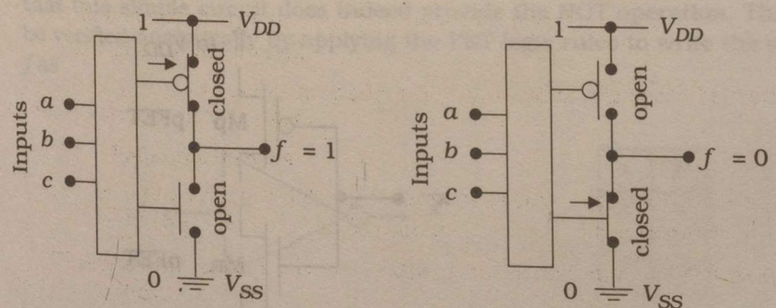


Figure 2.18 General CMOS logic gate

input bits that combine to give the output function bit $f(a, b, c)$. Since this is by definition a digital circuit, all of the quantities are restricted to values of 0 or 1. Digital logic circuits are nonlinear networks that use transistors as electronic switches to divert *one* of the supply voltages V_{DD} or 0 V to the output. This corresponds to a logical result of $f = 1$ or $f = 0$. Internally, we may view the output network of the gate as consisting of two switches SWp (an assert-low device) and SWn (an assert-high device) as shown. These are wired in to insure that one switch is closed while the other switch is open.

The operation of the general logic gate is shown in Figure 2.19 for both output possibilities. In Figure 2.19(a), the upper switch is closed while the lower switch is open. This connects the output to the power supply and yields a value of $f = 1$. The opposite situation is shown in Figure 2.19(b): the upper switch is open and the lower switch is closed. Because the output is now connected to $V_{SS} = 0$ V, the logical result is $f = 0$. Although this



(a) $f = 1$ output (b) $f = 0$ output

Figure 2.19 Operation of a CMOS logic gate

view is quite simplified, it does illustrate how CMOS logic circuits
The only missing feature in this model is the method used to co
output switches using the input variables. This is accomplish
MOSFETs.

The Complementary Pair

CMOS logic circuits are based on the concept of using **comple**
pairs of transistors for switching. A complementary pair consi
pFET and an nFET that have their gate terminals connected tog
shown in Figure 2.20. The input signal x simultaneously contr
conduction through both FETs. Note that the top of the pFE
assumed to be close to the power supply voltage V_{DD}, while the n
is close to the ground (V_{SS}). The behavior of the complementary
easily understood by observing the state of each FET for the two p
input values as in Figure 2.21. An input of $x = 0$ turns Mp on wh
nFET Mn is off and acts like an open switch; this is shown in F
2.21(a). The opposite case shown in Figure 2.21(b) is where $x = 1$. N
pFET MP is off while Mn is on. The name "complementary" is derive
this operation: when one FET is on, the other is off. The important
of this behavior is that the nFET and pFET are electrical opposites
translates directly into a coherent switching scheme.

Now that we have seen the overall structure of CMOS logic gat
the idea of a complementary pair, we have all of the concepts nee
create and analyze basic logic gate circuits.

2.3.1 The NOT Gate

The NOT or INVERT function is often considered the simplest B
operation. It has an input x and produces an output $f(x)$ of

$$f(x) = \text{NOT}(x) = \bar{x}$$

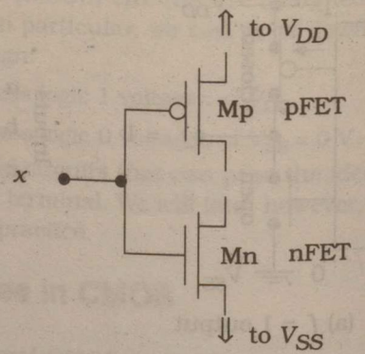
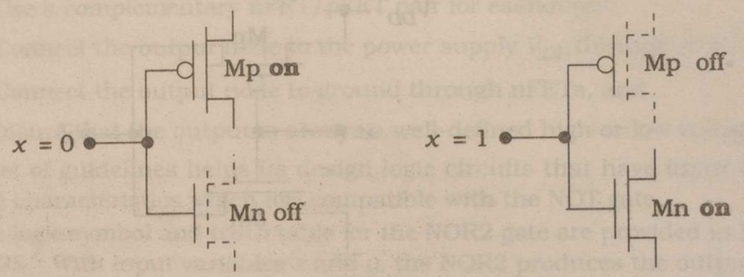


Figure 2.20 A CMOS complementary pair



(a) $x = 0$ input (b) $x = 1$ input

Figure 2.21 Operation of the complementary pair

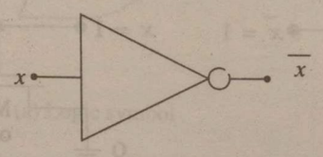
such that

$$\begin{aligned} &\text{If } x = 0 \quad \text{then } \bar{x} = 1 \\ &\text{If } x = 1 \quad \text{then } \bar{x} = 0 \end{aligned} \tag{2.33}$$

defines the notation. The logic symbol and truth table are provided in Figure 2.22 for future reference.

A CMOS NOT gate is shown in Figure 2.23. This has been constructed using the same idea as for the switch-based circuit discussed earlier in the context of Figure 2.7. The circuit uses a complementary pair of MOSFETs such that the input variable x controls both transistors.

The operation follows directly from the properties of the complementary pair. If the input x has a value of 0, then pFET MP is on and the nFET Mn is off. As shown in Figure 2.24(a), this connects the output node to the power supply voltage V_{DD}, giving an output of $\bar{x} = 1$. Conversely, if $x = 1$ then Mp is off and Mn is on. The output is then connected to the ground node and gives $\bar{x} = 0$ as verified by the circuit in Figure 2.24(b). It is clear that this simple circuit does indeed provide the NOT operation. This can be verified analytically by applying the FET logic rules to write the output f as



x	$\bar{x}$
0	1
1	0

(a) Logic symbol (b) Truth table

Figure 2.22 NOT gate

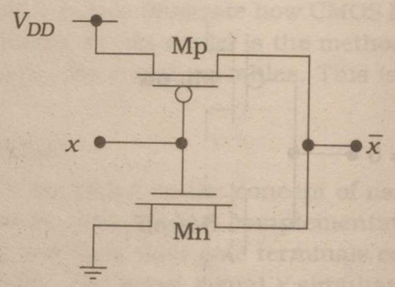


Figure 2.23 CMOS NOT gate

$$f = \bar{x} \cdot 1 + x \cdot 0 \qquad (2.$$

where the first term describes Mp and the second term is due to Mn. Simplifying gives

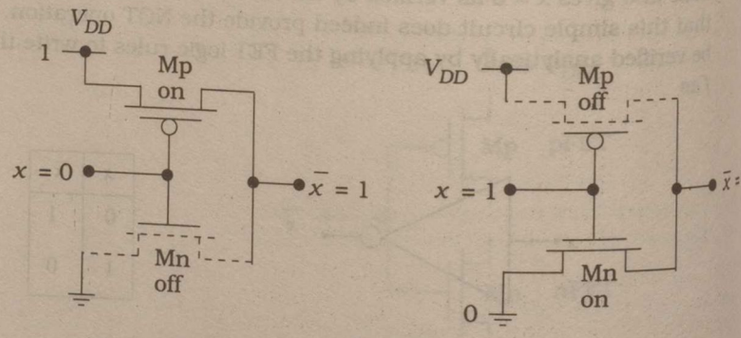
$$f = \bar{x} \qquad (2.$$

as expected.

One of the most important characteristics of the CMOS NOT gate is the manner in which the complementary FET pair insures that, for a given input logic state of $x = 0$ or 1, the output is connected to either V_{DD} or ground and gives a well-defined value. This circuit specifically avoids the possibilities where (i) both FETs are off at the same time, or (ii) both FETs are on at the same time; either situation would give an ill-defined output.

2.3.2 The CMOS NOR Gate

Now that we have seen the basic NOT gate, let us extend the concepts to create a 2-input NOR gate using the same principles. These are



(a) $x = 0$ input (b) $x = 1$ input

Figure 2.24 Operation of the CMOS NOT gate

- Use a complementary nFET/pFET pair for each input
- Connect the output node to the power supply V_{DD} through pFETs
- Connect the output node to ground through nFETs, and
- Insure that the output is always a well-defined high or low voltage

This set of guidelines helps us design logic circuits that have input and output characteristics which are compatible with the NOT gate.

The logic symbol and truth table for the NOR2 gate are provided in Figure 2.25.[6] With input variables x and y, the NOR2 produces the output

$$g(x, y) = \overline{x + y} \qquad (2.36)$$

such that a 1 at either input gives $g = 0$. Only the input combination $(x, y) = (1,1)$ yields an output of $g = 1$.

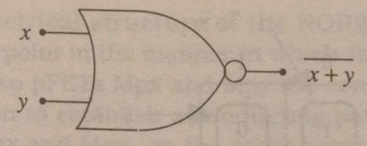
One way to synthesize the NOR2 operation at the logic design level is to use a 4:1 MUX as shown in Figure 2.26(a). Path selection is obtained using the input pair (x, y) such that every combination gives either a 1 or a 0 to the output. The Boolean expression for the output of the MUX is

$$g(x, y) = \bar{x} \cdot \bar{y} \cdot 1 + \bar{x} \cdot y \cdot 0 + x \cdot \bar{y} \cdot 0 + x \cdot y \cdot 0 \qquad (2.37)$$

which reduces to the desired form

$$g(x, y) = \overline{x + y} \qquad (2.38)$$

using the DeMorgan theorem. A voltage-equivalent circuit is obtained by replacing the binary quantities with voltages, and results in the circuit shown in Figure 2.26(b). In the notation of the drawing, V_x and V_y respectively represent the Boolean variables x and y. This information provides the basis for constructing the CMOS NOR2 circuit.



(a) Logic symbol

x	y	$\overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

(b) Truth table

Figure 2.25 NOR logic gate

[6] The terminology "NOR2" means a 2-input NOR gate.

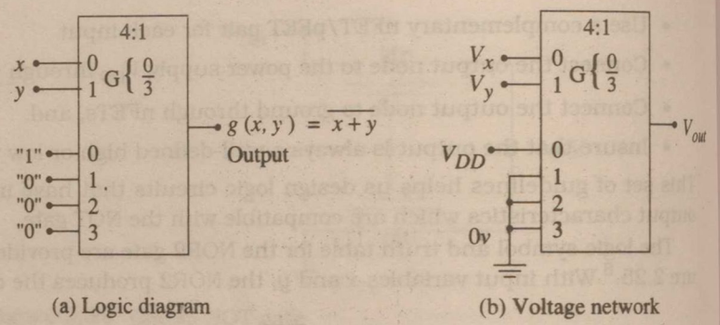


Figure 2.26 NOR2 operation using a 4:1 multiplexor

One approach to building the logic gate is to use the Karnaugh ma drawn in Figure 2.27. CMOS generally produces **inverting** logic becaus our gates are constructed using the NOT circuit as a basis. This create the situation where we are generally interested in the occurrence of bot 1's and 0's when dealing with K-maps. In particular, note that we hav created two 0-groupings in the drawing. The map allows us to write th logic expression in the form

$$g(x, y) = \bar{x} \cdot \bar{y} \cdot 1 + x \cdot 0 + y \cdot 0 \qquad (2.3$$

and work backward to construct the circuit. Each term represents a FE path to the output. The first term connects the output to 1 (the powe supply V_{DD}) and is controlled by complements of the input variables in a series-connected AND arrangement. The second and third terms repre sent two independent nFET paths between the output and 0 (ground Combining these statements results in the CMOS NOR2 circuit shown i

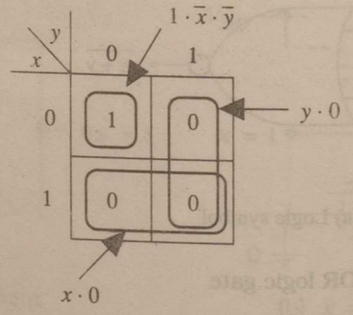


Figure 2.27 NOR2 gate Karnaugh map

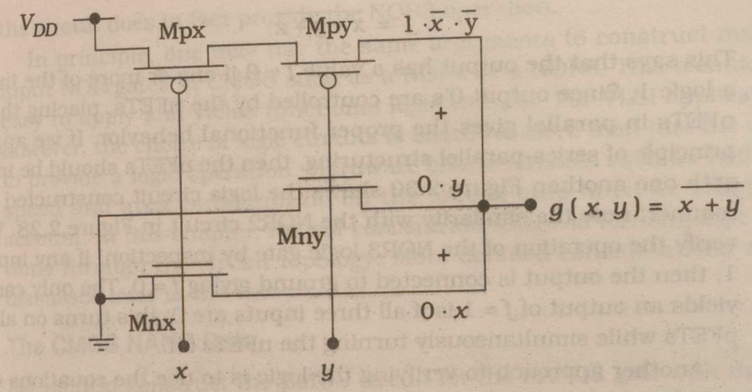


Figure 2.28 CMOS NOR2 gate

Figure 2.28; the one-to-one correspondence between each line in the equation and the circuit is obvious.

To verify that this circuit does have the proper electrical behavior, we may construct the table shown in Figure 2.29. This shows the state (on or off) of every FET for each of the four input possibilities. Tracing the output connections for each possibility easily shows that the switching circuit is consistent with the truth table.

x	y	Mpx	Mpy	Mnx	Mny	g
0	0	on	on	off	off	1
0	1	on	off	off	on	0
1	0	off	on	on	off	0
1	1	off	off	on	on	0

Figure 2.29 Operational summary of the NOR2 gate

The electrical structure of the NOR2 gate also illustrates another important point in the manner in which the FETs are wired together. Note that the two pFETs Mpx and Mpy are connected in series such that both must be on to establish a conducting path from V_{DD} to the output. The nFETs Mnx and Mny, on the other hand, are wired in parallel so that a connection between the output and ground is created if either nFET is on. This is called a **series-parallel** transistor arrangement; the principle allows us to design more complex gates.

As an example, let us construct a 3-input NOR (NOR3) gate using the NOR2 topology as a guideline. Let us label the inputs as x, y, and z. Each input is connected to the gate of a complementary nFET/pFET pair. The logical output expression for the gate is given by

$$f = \overline{x+y+z} \qquad (2.$$

This says that the output has a value $f = 0$ if one or more of the inputs a logic 1. Since output 0's are controlled by the nFETs, placing the nFETs in parallel gives the proper functional behavior. If we apply principle of series-parallel structuring, then the pFETs should be in se with one another. Figure 2.30 shows the logic circuit constructed in manner; note the similarity with the NOR2 circuit in Figure 2.28. We verify the operation of the NOR3 logic gate by inspection: if any input 1, then the output is connected to ground giving $f = 0$. The only case yields an output of $f = 1$ is if all three inputs are 0; this turns on all th pFETs while simultaneously turning the nFETs off.

Another approach to verifying the logic is to use the equations of F switches and derive the MUX equations. The top branch in Figure 2.3 through a series group of three pFETs as described by the term

$$1 \cdot \bar{x} \cdot \bar{y} \cdot \bar{z} \qquad (2.$$

where we recognize that the power supply voltage V_{DD} is equivalent t logic 1. Each of the three nFET branches consists of a single FET pass the ground to the output. Since a ground is a logic 0, we can OR the f branches together to give a complete output expression of

$$f = 1 \cdot \bar{x} \cdot \bar{y} \cdot \bar{z} + 0 \cdot x + 0 \cdot y + 0 \cdot z \qquad (2.$$

The nFET terms insure that the output voltage of the circuit is 0 V whe ever one or more of the inputs is 1. Logically, however, they evaluate t leaving the final form

$$f = 1 \cdot \bar{x} \cdot \bar{y} \cdot \bar{z} = \overline{x+y+z} \qquad (2.$$

where we have use a DeMorgan relation in the reduction. This shows th

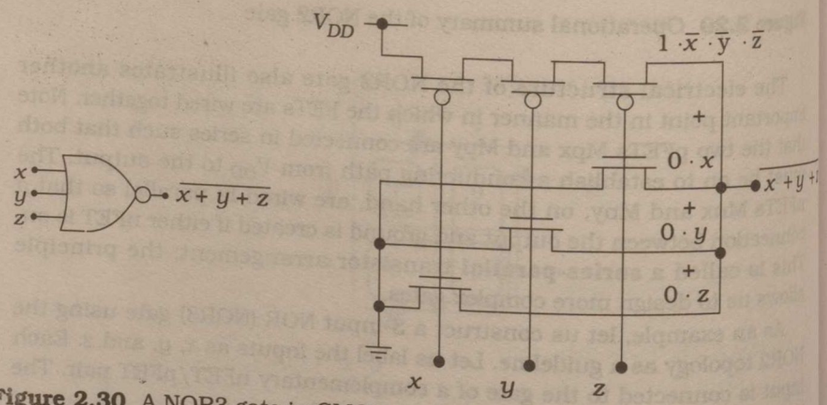


Figure 2.30 A NOR3 gate in CMOS

the circuit does in fact provide the NOR3 operation.

In principle, one may use the same arguments to construct multiple input NOR gates in CMOS such as a NOR4 or a NOR6. This technique is easy to apply and yields functional logic circuits. For VLSI applications, however, the choice of logic circuits is based on more than just the ability to provide a logic operation. Hardware characteristics such as switching speed and area consumption on the silicon chip must be taken into account. In this chapter, we will concentrate solely on forming logic functions through the circuit topology. More detailed considerations will be discussed later in the text.

2.3.3 The CMOS NAND Gate

Let us next construct the CMOS circuit for the NAND2 gate with the logical symbol and behavior summarized in Figure 2.31. This gate is characterized by an output that is 0 unless both of the inputs are 1. The truth table can be used to build the 4:1 MUX implementation drawn in Figure 2.32(a) such that the output is described by

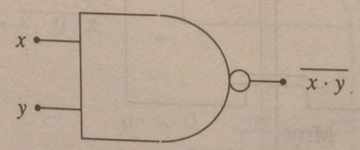
$$h(x, y) = \bar{x} \cdot \bar{y} \cdot 1 + \bar{x} \cdot y \cdot 1 + x \cdot \bar{y} \cdot 1 + x \cdot y \cdot 0 \qquad (2.44)$$

The voltage-equivalent network in Figure 2.32(b) is now somewhat obvious.

As with the NOR2 gate, it is useful to examine the Karnaugh map for the NAND2 function. Figure 2.33 shows the map along with two groupings that simplify the cases where $h = 1$. Using these reductions, our expression can be rewritten to read

$$h(x, y) = \bar{x} \cdot 1 + \bar{y} \cdot 1 + x \cdot y \cdot 0 \qquad (2.45)$$

Translating each term to FET groups yields the CMOS circuit shown in Figure 2.34. This gives the NAND2 function as can be verified by the oper-



(a) Logic symbol

x	y	$\overline{x \cdot y}$
0	0	1
0	1	1
1	0	1
1	1	0

(b) Truth table

Figure 2.31 NAND2 logic gate

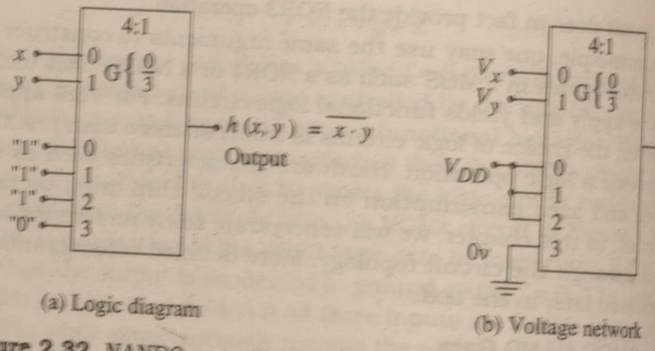


Figure 2.32 NAND2 operation using a 4:1 multiplexor

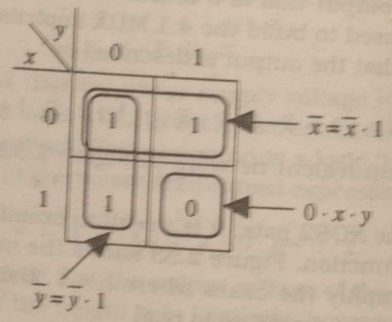


Figure 2.33 NAND2 K-map

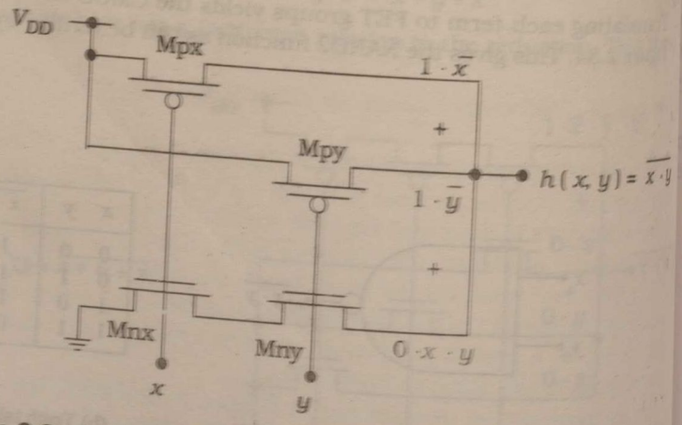


Figure 2.34 CMOS NAND2 logic circuit

x	y	Mpx	Mpy	Mnx	Mny	h
0	0	on	on	off	off	1
0	1	on	off	off	on	1
1	0	off	on	on	off	1
1	1	off	off	on	on	0

Figure 2.35 Operational summary of the NAND2 circuit

ation summarized in the table of Figure 2.35. An important characteristic of the NAND2 gate is that it uses two parallel-connected pFETs, while the nFETs are in series. This is exactly opposite to the structure of the NOR2 gate.

A NAND3 gate can be created using the same topology. It requires three sets of complementary pairs, each driven by a separate input. The nFETs are placed in series, while the pFETs are wired in parallel. This gives the gate shown in Figure 2.36. To verify the operation of the circuit, note that all three inputs must be 1's to provide a conduction path between the output and ground. If any one (or more) of the inputs is a 0, then the corresponding nFET is off while the pFET it drives acts like a closed switch; this gives a logic 1 voltage of V_{DD} at the output.

Switch logic analysis can also be applied by treating the circuit as a multiplexor. The series-connected nFET chain at the bottom of the circuit is described by the logic term

$$0 \cdot x \cdot y \cdot z \tag{2.46}$$

Each pFET branch consists of a single transistor that acts like a closed switch when a 0 is applied. Performing the OR operation among the four

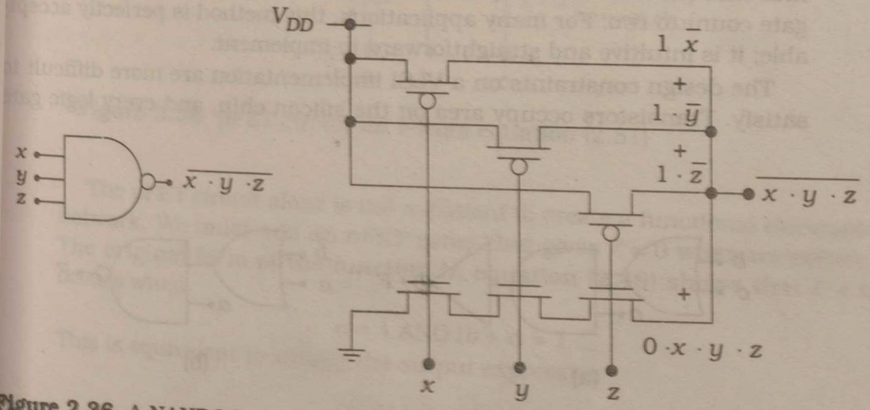


Figure 2.36 A NAND3 logic gate in CMOS

branches gives

$$0 \cdot x \cdot y \cdot z + 1 \cdot \bar{x} + 1 \cdot \bar{y} + 1 \cdot \bar{z}$$

Eliminating the 0 terms and using DeMorgan reduction gives the function as

$$\overline{x \cdot y \cdot z}$$

which is the NAND3 function. This technique can be extended to the CMOS circuitry for NAND gates with more inputs.

2.4 Complex Logic Gates in CMOS

One of the most powerful aspects of building logic circuits in CMOS ability to create a single circuit that provides several primitive oper (NOT, AND, and OR) in an integrated manner. These will be called **plex** or **combinational logic gates** in our discussion. Complex logic are very useful in VLSI system-level design.

To illustrate the main idea of a complex logic gate, consider the ean expression

$$F(a, b, c) = \overline{a \cdot (b + c)}$$

The simplest way to construct the logic network for this function is one OR-gate, one AND-gate, and one NOT-gate as shown in F 2.37(a). Alternately, we might simplify the network to that in F 2.37(b) if a NAND2 gate is available. If we build the electronic equi of either implementation, then the traditional approach would be to one-to-one map: each gate requires one electronic logic circuit. Fo first case (a), this would require three separate gates, while (b) reduce gate count to two. For many applications, this method is perfectly ac able; it is intuitive and straightforward to implement.

The design constraints on a VLSI implementation are more diffic satisfy. Transistors occupy area on the silicon chip, and every logic

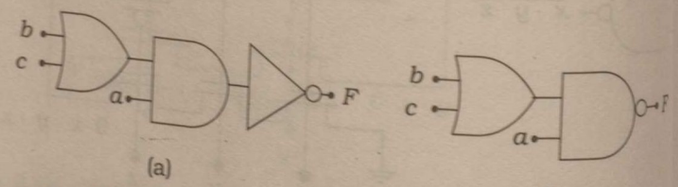


Figure 2.37 Logic function example

uses transistors. Since the gate count on a VLSI chip can easily exceed several hundred thousand, we often look for techniques that reduce the number of gates and/or FETs while still performing the required logic. In the present discussion, we will achieve this objective by building a single logic gate that implements the entire function.

Let us investigate the characteristics of the function F in more detail by applying DeMorgan expansions to the function to write

$$\begin{aligned} F &= \overline{a \cdot (b + c)} \\ &= \bar{a} + \overline{(b + c)} \\ &= [\bar{a} + (\bar{b} \cdot \bar{c})] \cdot 1 \end{aligned} \tag{2.50}$$

The last step is simply ANDing the result with a logical 1. Expanding gives

$$F = \bar{a} \cdot 1 + (\bar{b} \cdot \bar{c}) \cdot 1 \tag{2.51}$$

which is in a form that can be used to build the pFET switching circuit shown in Figure 2.38. The correspondence can be verified by checking each term. The first term implies a pFET connected between the power supply (V_{DD}) and the output that is controlled by the input a. The second term is identical in form to that encountered for the NOR2 gate. It represents two series-connected pFETs (with control variables b and c) that connect the power supply to the output.

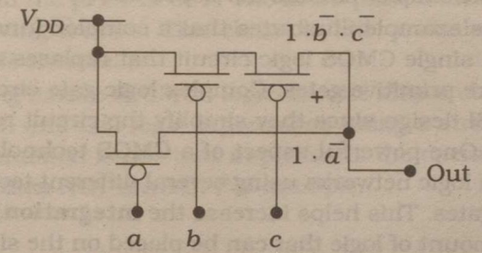


Figure 2.38 pFET circuit for F from equation (2.51)

The pFET circuit alone is not sufficient to create a functional electronic network. We must add an nFET array that gives $F = 0$ when necessary. The original form of the function in equation (2.49) shows that $F = 0$ occurs when

$$a = 1 \text{ AND } (b + c) = 1$$

This is equivalent to writing the output expression

$$0 \cdot [a \cdot (b + c)] \tag{2.52}$$

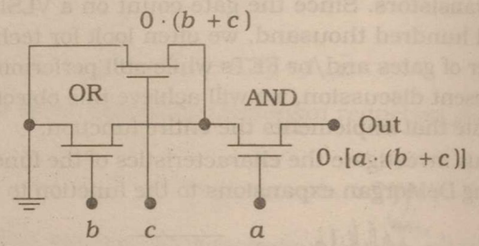


Figure 2.39 nFET logic circuit for *F*

which can in turn be used to describe the nFET array shown in F
2.39. Two parallel-connected nFETs that are controlled by *b* and
the OR operation. This group is in series with the *a*-input nFET t
duce the AND. The logic can be verified by the Karnaugh map grou
shown in Figure 2.40. Simplification to using a single *a*-input
occurs because of the common term encompassed by the two group

The completed CMOS logic gate is built by combining the nFE
pFET circuits, and results in the circuit of Figure 2.41. We have r
the orientation of the FETs by 90 degrees to arrive at the finished
matic. This is the most common way to draw CMOS logic circuits s
makes series and parallel-connected FETs more obvious. The equiv
of the circuit can be verified by tracing out each branch and compa
with the simpler circuits developed above.

This example illustrates that a complex function can be implem
with a single CMOS logic circuit that replaces a cascade made up
or more primitive gates. Complex logic gate circuits can be more ef
in VLSI design since they simplify the circuit requirements and th
flows. One powerful aspect of a CMOS technology is that it allows
design logic networks using several different techniques, such as co
logic gates. This helps increase the **integration density**, which mea
the amount of logic that can be placed on the silicon chip.

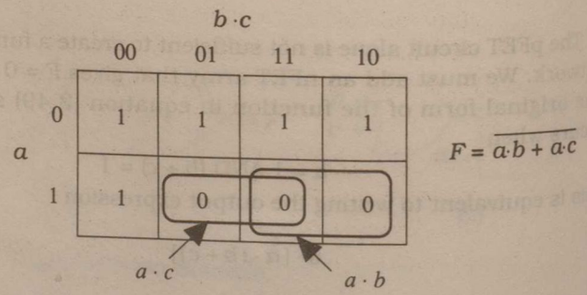


Figure 2.40 Karnaugh map grouping for the nFET circuit

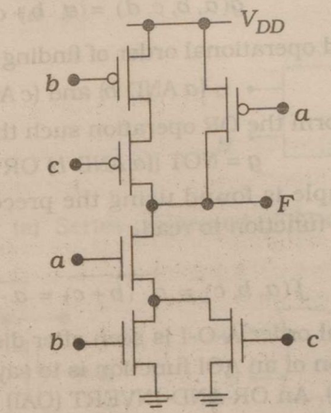


Figure 2.41 Finished complex CMOS logic gate circuit

2.4.1 Structured Logic Design

A structured approach to designing complex logic gates can be developed by focusing on the circuit characteristics. CMOS logic gates are intrinsically **inverting**; this means that the output always produces a NOT operation acting on the input variables. The simple inverter in Figure 2.42 illustrates the origin of this property. If the input *a* is a logic 1, then the nFET is ON and the pFET is OFF. The nFET passes the logic 0 (ground) to the output, giving $\overline{a}$ there. This characteristic was also observed in the NAND and NOR circuits.

The inverting nature of CMOS logic circuits allows us to construct logic circuits for AOI and OAI logic expressions using a structured approach. An AOI logic function is one that implements the operations in the order AND then OR then NOT (invert). For example,

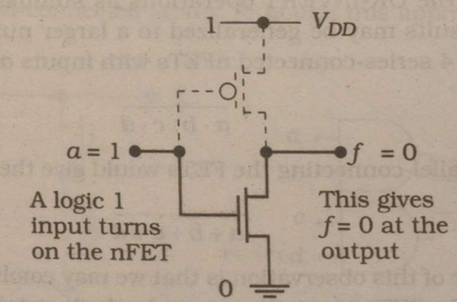


Figure 2.42 Origin of the inverting characteristic of CMOS gates

$$g(a, b, c, d) = \overline{a \cdot b + c \cdot d}$$

has an implied operational order of finding

(a AND b) and (c AND d)

and then perform the OR operation such that the final result is

g = NOT [(a AND b) OR (c AND d)]

Another example is found using the preceding CMOS gate exampl expanding the function to read

$$f(a, b, c) = \overline{a \cdot (b + c)} = \overline{a \cdot b + a \cdot c}$$

The operational order A-O-I is seen after distributing the terms. An a nate description of an AOI function is to say that it is an inverted sum products (SOP). An OR-AND-INVERT (OAI) function reverses the orde the AND and OR operations. An example of an OAI form is

$$h(x, y, z, w) = \overline{(x + y) \cdot (z + w)}$$

since this implies that we first calculate

(x OR y) along with (w OR z)

and then

h = NOT [(x OR y) AND (w OR z)]

to evaluate the value of h. An OAI form is equivalent to an inverted pr uct-of-sums (POS) expression.

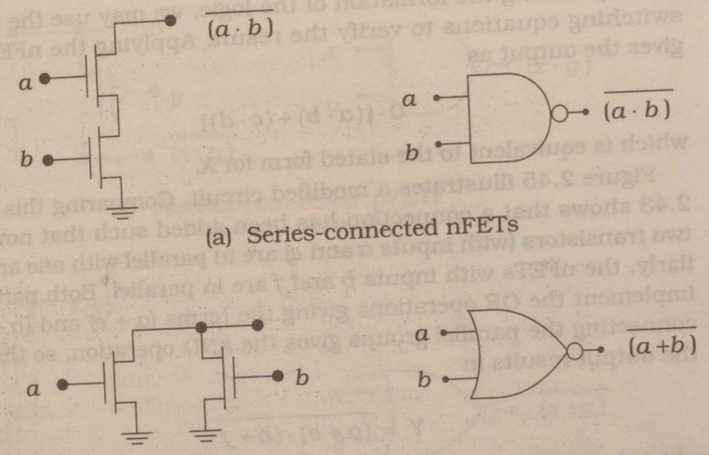
CMOS switching characteristics provide a natural means for imp menting inverting logic forms such as AOI and OAI. The techniqu based on using nFET and pFET arrays in a consistent manner. Comp logic gates of this type allow the designer to compress three or more pr itive operations into a single logic gate. Consider first the logic formati properties of nFETs. From the NAND analysis, we learned that nFETs series provide AND-INVERT logic; this is shown in Figure 2.43(a). Sim larly, the NOR gate analysis showed us that parallel connected nFE produce the OR-INVERT operations as summarized in Figure 2.43 These results may be generalized to a larger number of transistors. F example, 4 series-connected nFETs with inputs a, b, c, d would produc

$$\overline{a \cdot b \cdot c \cdot d}$$

while parallel-connecting the FETs would give the OR-INVERT operation

$$\overline{a + b + c + d}$$

The power of this observation is that we may combine series- and paral connected nFETs to produce complex logic gates. An example of this shown in Figure 2.44. This array consists of parallel-connected group



(a) Series-connected nFETs

(b) Parallel-connected nFETs

Figure 2.43 nFET logic formation

with each group made of 2 series-connected nFETs. The transistors on the left side form the AND operation ($a \cdot b$) while the right group of nFETs yields ($c \cdot d$); the parallel connection of the two groups gives the OR operation, while the final output from the gate yields the NOT. We thus see that the function is described by

$$X = \overline{(a \cdot b) + (c \cdot d)} \tag{2.58}$$

which is an AOI expression that is represented by the logic circuit shown next to the circuit. It is important to note that the NOT operation is viewed at the exit point of the logic (i.e., only for the function X). The AND operation is provided by series-connected nFETs, while the OR is accomplished by using a parallel-connected group. Although this approach is based on

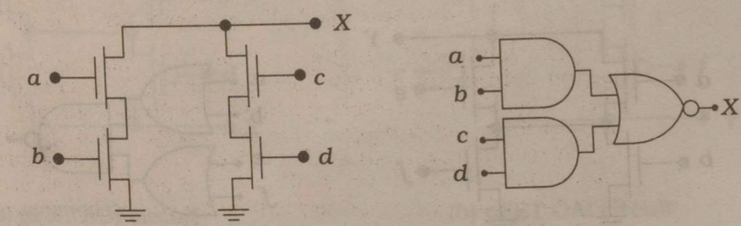


Figure 2.44 nFET AOI circuit

visually tracing the formation of the logic, we may use the formalism switching equations to verify the result. Applying the nFET equati gives the output as

$$0 \cdot [(a \cdot b) + (c \cdot d)] \qquad (2.$$

which is equivalent to the stated form for X.

Figure 2.45 illustrates a modified circuit. Comparing this with Fig 2.43 shows that a connection has been added such that now the up two transistors (with inputs a and e) are in parallel with one another. Si ilarly, the nFETs with inputs b and f are in parallel. Both parallel grou implement the OR operations giving the terms $(a + e)$ and $(b + f)$. Seri connecting the parallel groups gives the AND operation, so that invert the output results in

$$Y = \overline{(a+e) \cdot (b+f)} \qquad (2.$$

which has OAI form. To verify this result, use the switch-level equati to write

$$0 \cdot [(a+e) \cdot (b+f)] \qquad (2.$$

This is equivalent to the expression in equation (2.60) for Y.

Now recall that a CMOS logic gate uses nFETs to pass a 0 to the ou put, and pFETs to pass a logic 1. Since pFETs complement nFETs, we c construct the logic formation characteristics summarized in Figure 2.4 The parallel-connected pFETs shown in Figure 2.46(a) are described the logic equation

$$1 \cdot (\bar{x} + \bar{y}) = 1 \cdot \overline{(x \cdot y)} \qquad (2.$$

which is the AND-NOT operation sequence. To obtain the OR-NOT oper tion, we must use series-connected pFETs as in Figure 2.46(b). In th case, the logic is formed from switching equations as

$$1 \cdot \bar{x} \cdot \bar{y} = 1 \cdot \overline{(x+y)} \qquad (2.$$

which verifies the statement.

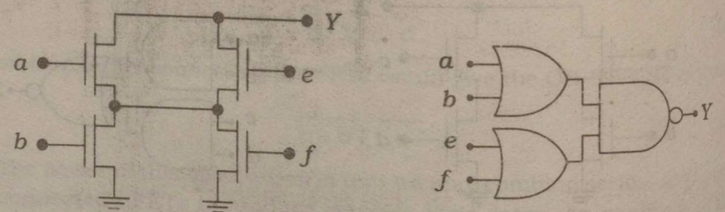
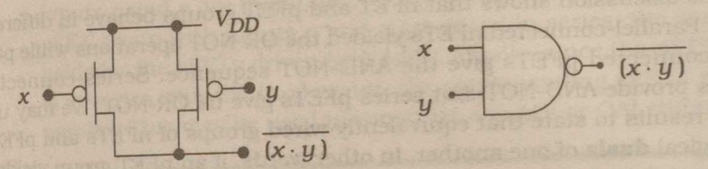
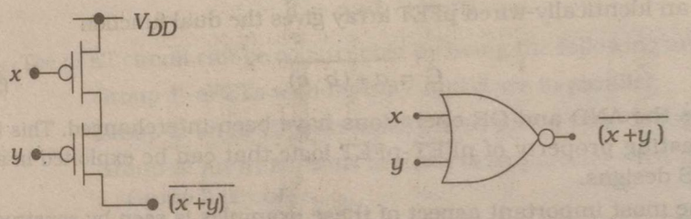


Figure 2.45 nFET OAI network



(a) Parallel-connected pFETs



(b) Series-connected pFETs

Figure 2.46 pFET logic formation

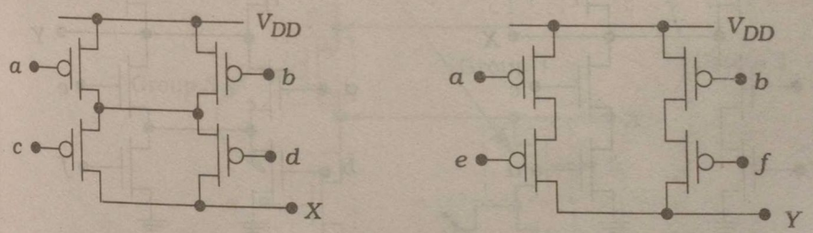
Let us examine the pFET array needed for the AOI function

$$X = \overline{(a \cdot b) + (c \cdot d)} \qquad (2.64)$$

discussed earlier for the nFET circuit shown in Figure 2.44. Using the pFET rules results in the network illustrated in Figure 2.47(a). Similarly, the OAI function

$$Y = \overline{(a+e) \cdot (b+f)} \qquad (2.65)$$

yields the pFET array in Figure 2.47(b).



(a) pFET AOI circuit

(b) pFET OAI circuit

Figure 2.47 pFET arrays for AOI and OAI gates

This discussion shows that nFET and pFET groups behave in di ways. Parallel-connected nFETs yielded the OR-NOT operations whi allel-connected pFETs give the AND-NOT sequence. Series-conn nFETs provide AND-NOT, but series pFETs give us OR-NOT. We m these results to state that equivalently wired groups of nFETs and p are logical **duals** of one another. In other words, if an nFET group y function of the form

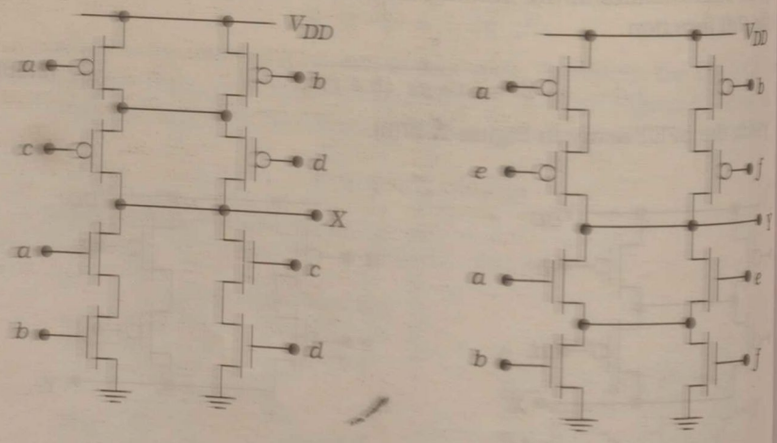
$$g = \overline{a \cdot (b+c)}$$

then an identically-wired pFET array gives the dual function

$$G = \overline{a+(b \cdot c)}$$

where the AND and OR operations have been interchanged. This interesting property of nFET-pFET logic that can be exploited in CMOS designs.

The most important aspect of these examples is seen by constru the complete CMOS circuit for each; both are shown in Figure 2.48 sider first the AOI circuit in Figure 2.48(a). The nFETs with inputs a are in series, while the corresponding pFETs are wired in parallel scheme is also applied to the FETs with input variables c and d. Fi the nFET group with inputs (a, b) is in parallel with the input group so the corresponding pFET groups are in series. This is another exa of series-parallel structuring of the nFET-pFET arrays. The OAI circu Figure 2.48(b) exhibits the same features. In this case, the nFETs



(a) AOI circuit

(b) OAI circuit

Figure 2.48 Complete CMOS AOI and OAI circuits

inputs a and e are in parallel, as are the nFETs with inputs b and f. The pFET group with inputs a and e are wired in series; the same comment holds for the pFETs driven by b and f. Finally, since the nFET (a, e) group is in series with the (b, f) group, the corresponding pFETs groups are in parallel. This may be used to construct any AOI or OAI circuit in CMOS.

Example 2.1

Consider the complex function

$$X = \overline{a+b \cdot (c+d)} \tag{2.68}$$

The nFET circuit can be constructed by using the following arrangements:

Group 1: nFETs with inputs c and d are in parallel;

Group 2: an nFET with input b is in series with Group 1;

Group 3: an nFET with input a is in parallel with the Group 1-Group 2 circuit.

The circuit in Figure 2.49 shows each group explicitly. The pFETs are arranged using series-parallel structuring. Each group of pFETs can be associated with the nFET group that has the same inputs such that

Group 1: pFETs with inputs c and d are in series;

Group 2: a pFET with input b is in parallel with Group 1 pFETs;

Group 3: a pFET with input a is in series with the Group 1-Group 2 pFETs.

The equivalent logic diagram for the circuit is shown in Figure 2.50.

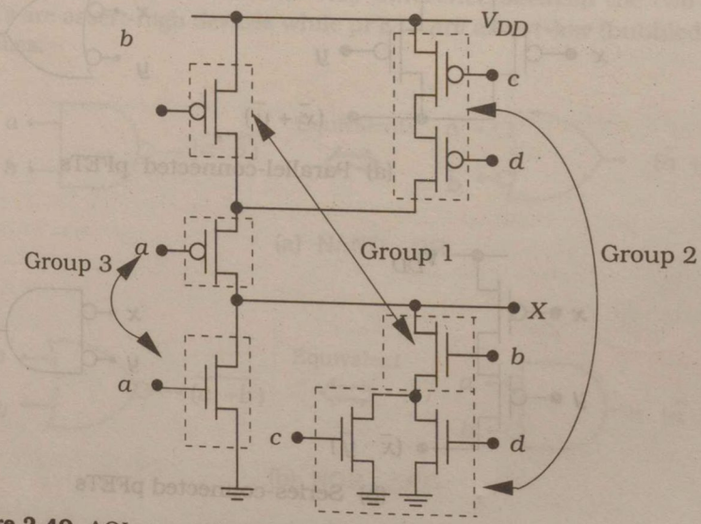


Figure 2.49 AOI circuit for Example 2.1

Tracing the data flow from the inputs to the output shows that t
has OAOI structuring. This is just an AOI circuit with an additio
input.

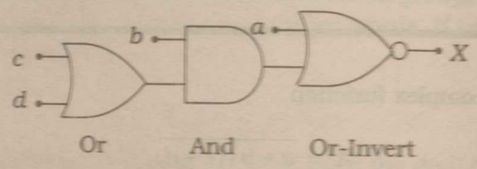
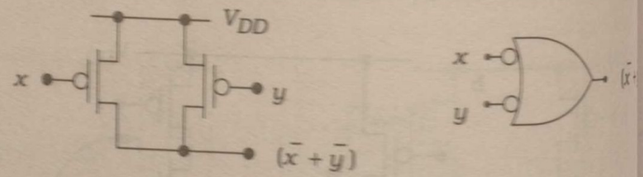


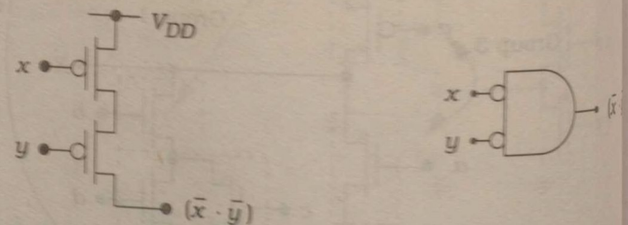
Figure 2.50 Equivalent logic diagram for Example 2.1

Bubble Pushing

Series-parallel wiring of complex CMOS logic circuits can be de
using an approach that is based on logic diagrams. The proced
obtained by applying the DeMorgan rules to the pFET relations illus
in Figure 2.46. Recall that pFETs can be modeled as assert-low sw
Let us therefore model pFET groups as logic gates that have asse
inputs. This leads to the modified logic associations shown in Figur
In Figure 2.51(a), we apply the DeMorgan rule to write



(a) Parallel-connected pFETs



(b) Series-connected pFETs

Figure 2.51 Assert-low models for pFETs

$$1 \cdot (\overline{x \cdot y}) = 1 \cdot (\bar{x} + \bar{y}) \tag{2.69}$$

so that parallel-connected pFETs may be viewed as an OR operation with assert-low (bubbled) inputs. In the same manner, the series-connected pFETs in Figure 2.51(b) provide the AND operation with assert-low inputs as verified by the identity

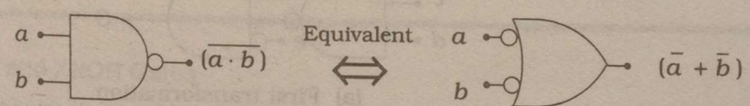
$$1 \cdot (\overline{x + y}) = 1 \cdot (\bar{x} \cdot \bar{y}) \tag{2.70}$$

Both operations can be represented graphically by the operations shown in Figure 2.52 where we visualize pushing the bubble backward through the gate to the inputs to create the dual operation with assert-low input ports.

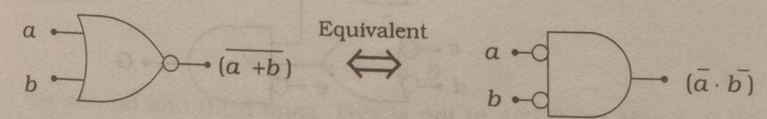
The procedure for designing the transistor circuitry for a CMOS logic gate can be summarized by the following steps.

- Construct the logic diagram using basic AOI or OAI structuring. Deeper nesting, such a OAOI and AOAI, is allowed.
- Use the gate-nFET relations summarized in Figure 2.43 to construct the nFET logic circuit between the output and ground.
- To obtain the topology of the pFET array, start with the original logic diagram and push the bubble back toward the inputs using the DeMorgan rules. Continue the backward pushing until every input is bubbled. The pFET circuitry between the output and VDD is then obtained using the rules in Figure 2.51.

Note that both the nFETs and the pFETs are wired such that parallel-connected transistors give the OR operation, while series-connected FETs provide the AND operation. The only difference between the two is that nFETs are assert-high devices while pFETs are assert-low (bubbled-input) switches.



(a) NAND - OR



(b) NOR - AND

Figure 2.52 Bubble pushing using DeMorgan rules

Example 2.2

Consider the logic diagram shown in Figure 2.53. This provides us w map for building the nFET logic array. We see that the nFETs with in *a* and *b* are in series (due to the AND gate), as are the nFETs with in *c* and *d*. These series-connected groups are in parallel with an nFET has the input *e* since they are OR'ed at the output. The NOT operation the NOR gate) is automatic in the nFET array.

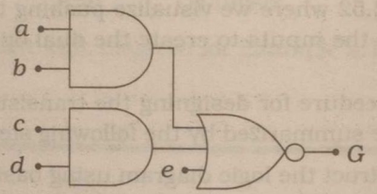
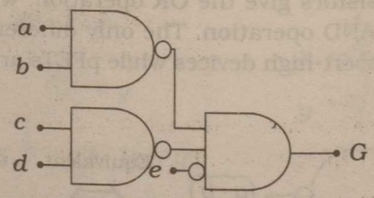
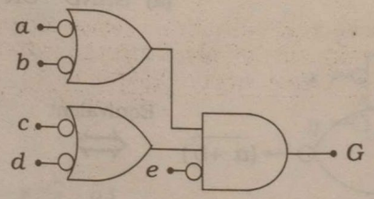


Figure 2.53 AOI logic diagram for bubble-pushing example

To obtain the wiring of the pFETs, we push the bubble back as sh in Figure 2.54. The first step is to transform the output NOR gate int AND gate with assert-low inputs; this results in the intermediate diag drawn in Figure 2.54(a). Pushing the bubbles back through the gates gives assert-low OR gates as in Figure 2.54(b). This shows tha



(a) First transformation



(b) Final form

Figure 2.54 Bubble pushing to obtain the topology of the pFET arra

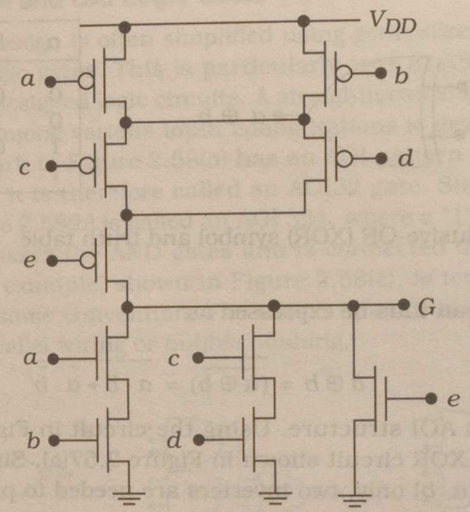


Figure 2.55 Final circuit for the bubble-pushing example

pFET array consists of

- Two pFETs with inputs *a* and *b* wired in parallel
- Two pFETs with inputs *c* and *d* wired in parallel
- One pFET with an input *e* that is in series with the two groups above.

The final circuit is drawn in Figure 2.55. It is worth the effort to trace through the construction procedure. And, it is important to remember that the CMOS logic gate implements the entire function G portrayed in the logic diagram. It is not possible to break the circuit down into more primitive logic.

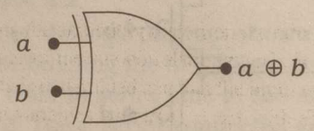
2.4.2 XOR and XNOR Gates

An important example of using an AOI circuit is constructing Exclusive-OR (XOR) and Exclusive-NOR circuits. These often-used gates are constructed from logic primitives. Figure 2.56 gives the circuit symbol and truth table for the XOR. Reading the logic 1 outputs gives the standard SOP equation

$$a \oplus b = \bar{a} \cdot b + a \cdot \bar{b} \tag{2.71}$$

from the second and third lines. This is not in AOI form. However, if we read the 0 output lines, then the XNOR expression is

$$\overline{a \oplus b} = a \cdot b + \bar{a} \cdot \bar{b} \tag{2.72}$$



a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Figure 2.56 Exclusive-OR (XOR) symbol and truth table

The XOR can thus be expressed as

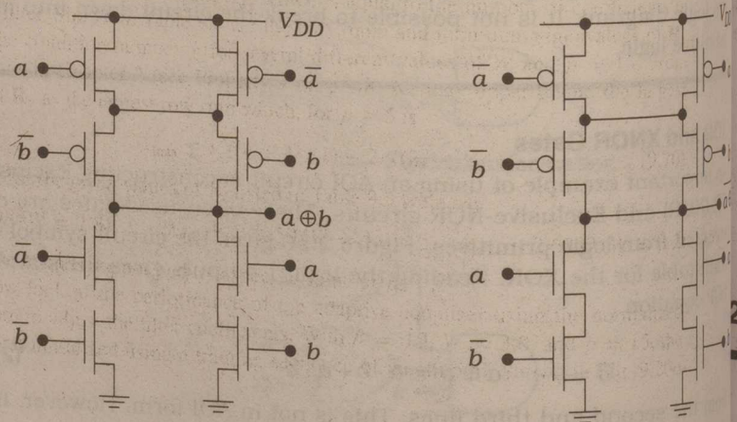
$$a \oplus b = \overline{(\overline{a \oplus b})} = \overline{a \cdot b + \bar{a} \cdot \bar{b}}$$

which has AOI structure. Using the circuit in Figure 2.48(a) gi basic AOI XOR circuit shown in Figure 2.57(a). Since the XOR g inputs of (a, b) only, two inverters are needed to provide the 4-in (a, b, $\bar{a}$, $\bar{b}$) in this circuit.

To obtain an XNOR circuit, we just complement the XOR SOP e to write

$$\overline{a \oplus b} = \overline{\bar{a} \cdot b + a \cdot \bar{b}}$$

Interchanging a and $\bar{a}$ in the XOR circuit thus gives the XNOR gate ure 2.57(b). Switching the b and $\bar{b}$ variables would have given th result.

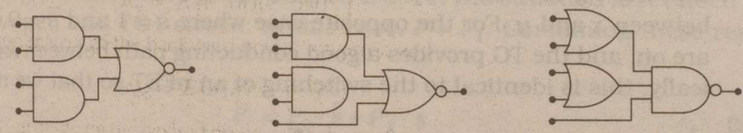


(a) Exclusive-OR (b) Exclusive-NOR

Figure 2.57 AOI XOR and XNOR gates

2.4.3 Generalized AOI and OAI Logic Gates

Standard logic design is often simplified using generalized multiple-input AOI and OAI logic gates. This is particularly true in ASIC-type circuits that rely on predesigned logic circuits. A straightforward nomenclature for distinguishing among various input configurations is developed in Figure 2.58. The network in Figure 2.58(a) has an AOI pattern with 2 inputs to each AND gate; it is therefore called an AOI22 gate. Similarly, the logic pattern in Figure 2.58(b) is called an AOI 321, where a "1" label implies an input that bypasses the AND gates and is connected directly to an OR gate. The third example, shown in Figure 2.58(c), is termed an OAI221 gate using the same convention. The CMOS circuits are easily designed using series-parallel wiring or bubble pushing.



(a) AOI22 gate (b) AOI321 gate (c) OAI221 gate

Figure 2.58 General naming convention

Generalized complex logic gates provide a uniform basis for creating different logic operations using a generic gate. As a simple example, consider the AOI22 gate shown in Figure 2.59(a). This provides an output of

$$\text{AOI22}(a, b, c, d) = \overline{a \cdot b + c \cdot d} \tag{2.75}$$

To create an XOR circuit, we can define the inputs as shown in Figure 2.59(b), which allows us to write

$$a \oplus b = \text{AOI22}(a, b, \bar{a}, \bar{b}) \tag{2.76}$$

Using the same reasoning, the XNOR function can be obtained using

$$\overline{a \oplus b} = \text{AOI22}(a, \bar{b}, \bar{a}, b) \tag{2.77}$$

This illustrates how generic logic gates can be used in random logic design.

2.5 Transmission Gate Circuits

A CMOS **transmission gate** is created by connecting an nFET and pFET in parallel as shown in Figure 2.60(a). The nFET Mn is controlled by the signal s, while the pFET Mp is controlled by the complement $\bar{s}$. When wired in this manner, the pair acts as a good electrical switch between the

2.5.1 Logic Design

Transmission gate logic design has been used extensively in CMOS design for many years. The simplicity of the switching and the ability to transmit the entire range of voltages made it attractive for many applications. TG circuits are found in many ASIC structures, making them worth studying in more detail.

Multiplexors

The ideal-switch characteristics of TGs make them useful for creating some rather unique circuits. An example is the 2-to-1 MUX shown in Figure 2.61. The operation of the circuit is summarized in the table. When the selector signal has a value $s = 0$, TG0 is closed and TG1 is open, so that P_0 is transmitted to the output. If $s = 1$, the situation is reversed with TG0 open and TG1 closed; in this case, $F = P_1$. Combining these results gives

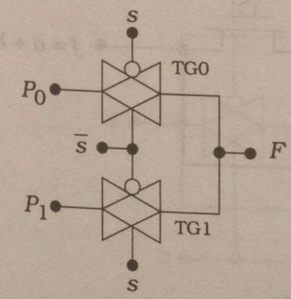
$$F = P_0 \cdot \bar{s} + P_1 \cdot s \tag{2.79}$$

which is the required equation. Note that the use of a pair of TGs eliminates the possibility of having a floating (disconnected) output since one TG is always closed while the other will be open. The 2-to-1 architecture can be extended to a 4:1 network by using the 2-bit selector word (s_1 s_0) that has values of (0 0), (0 1), (1 0), and (1 1). Each input line (P_0, P_1, P_2, P_3) will have two TGs in its path such that the output is

$$F = P_0 \cdot \overline{s_1} \cdot \overline{s_0} + P_1 \cdot \overline{s_1} \cdot s_0 + P_2 \cdot s_1 \cdot \overline{s_0} + P_3 \cdot s_1 \cdot s_0 \tag{2.80}$$

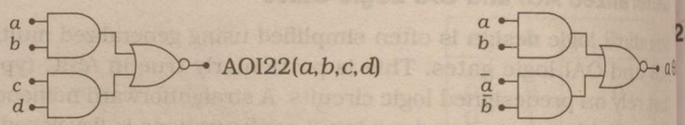
For example, the P_0 path will have TGs that are closed with (s_1 s_0) = (0 0). The construction of the network is left as an exercise for the reader.

The 2:1 MUX can be modified to produce other useful functions. One is illustrated in Figure 2.62(a). The input to the top TG is a; this is inverted so that $\bar{a}$ enters the lower TG. Variable b and its complement are used to control the TGs. When $b = 0$, the upper TG is closed and a is passed to the



s	TG0	TG1	F
0	Closed	Open	P_0
1	Open	Closed	P_1

Figure 2.61 A TG-based 2-to-1 multiplexor



(a) General AOI22 gate (b) XOR

Figure 2.59 Application of an AOI22 gate

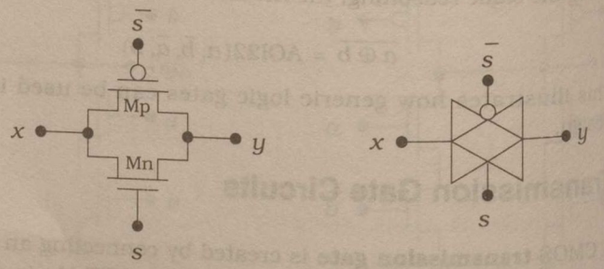
input and the output variables x and y, respectively.

The operation of the switch can be understood by analyzing the cases for s. If $s = 0$, the nFET is OFF; since $\bar{s} = 1$, the pFET is also OF that the TG acts as an open switch. In this case, there is no relation between x and y. For the opposite case where $s = 1$ and $\bar{s} = 0$, both are on, and the TG provides a good conducting path between x and y. ically, this is identical to the switching of an nFET so that we may w

$$y = x \cdot s \quad \text{iff} \quad s = 1$$

This assumes that x is the input and y is the output. However, the classified as a **bi-directional** switch. The TG symbol in Figure 2.60 based on this observation. It is created using two back-to-back a indicating that the data can flow in either direction. Control is achiev s and $\bar{s}$; the bubble indicates the connection to the pFET gate.

Transmission gates are useful because they can transmit the e voltage range [0, V_{DD}] from left to right (or vice versa). This is due t parallel connection of the transistors. Zero voltage levels are transm by the nFET, while the pFET is responsible for transmitting the p supply voltage V_{DD}. The main drawback of using TGs in modern VL that they require two FETs and an implied inverter that takes s and duces $\bar{s}$.



(a) Transistor circuit (b) Symbol

Figure 2.60 Transmission gate (TG)

2.5.1 Logic Design

Transmission gate logic design has been used extensively in CMOS design for many years. The simplicity of the switching and the ability to transmit the entire range of voltages made it attractive for many applications. TG circuits are found in many ASIC structures, making them worth studying in more detail.

Multiplexors

The ideal-switch characteristics of TGs make them useful for creating some rather unique circuits. An example is the 2-to-1 MUX shown in Figure 2.61. The operation of the circuit is summarized in the table. When the selector signal has a value $s = 0$, TG0 is closed and TG1 is open, so that P_0 is transmitted to the output. If $s = 1$, the situation is reversed with TG0 open and TG1 closed; in this case, $F = P_1$. Combining these results gives

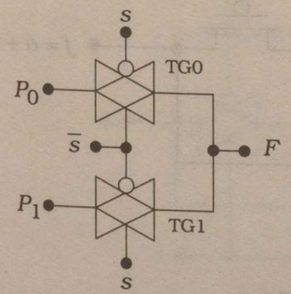
$$F = P_0 \cdot \bar{s} + P_1 \cdot s \tag{2.79}$$

which is the required equation. Note that the use of a pair of TGs eliminates the possibility of having a floating (disconnected) output since one TG is always closed while the other will be open. The 2-to-1 architecture can be extended to a 4:1 network by using the 2-bit selector word (s_1 s_0) that has values of (0 0), (0 1), (1 0), and (1 1). Each input line (P_0, P_1, P_2, P_3) will have two TGs in its path such that the output is

$$F = P_0 \cdot \overline{s_1} \cdot \overline{s_0} + P_1 \cdot \overline{s_1} \cdot s_0 + P_2 \cdot s_1 \cdot \overline{s_0} + P_3 \cdot s_1 \cdot s_0 \tag{2.80}$$

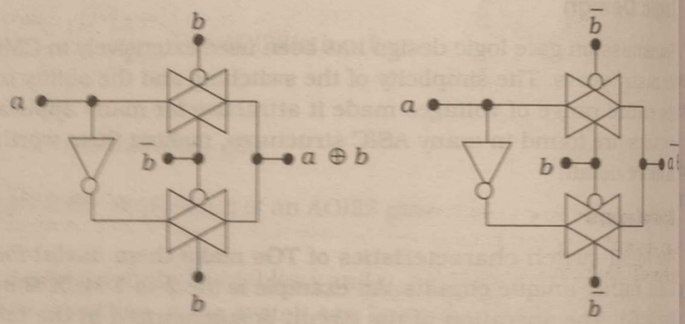
For example, the P_0 path will have TGs that are closed with (s_1 s_0) = (0 0). The construction of the network is left as an exercise for the reader.

The 2:1 MUX can be modified to produce other useful functions. One is illustrated in Figure 2.62(a). The input to the top TG is a; this is inverted so that $\bar{a}$ enters the lower TG. Variable b and its complement are used to control the TGs. When $b = 0$, the upper TG is closed and a is passed to the



s	TG0	TG1	F
0	Closed	Open	P_0
1	Open	Closed	P_1

Figure 2.61 A TG-based 2-to-1 multiplexor



(a) XOR circuit (b) XNOR circuit

Figure 2.62 TG-based exclusive-OR and exclusive-NOR circuits

output, while $b = 1$ closes the lower TG and steers $\bar{a}$ to the outp gives

$$a \cdot \bar{b} + \bar{a} \cdot b = a \oplus b$$

i.e., the circuit provides the XOR (exclusive-OR) function. The exp can be verified using the 2:1 MUX result. An XNOR function

$$\overline{a \oplus b} = a \cdot b + \bar{a} \cdot \bar{b}$$

is obtained if we interchange b and $\bar{b}$. The circuit for this simple m tion is shown in Figure 2.62(b).

OR Gate

Transmission gate characteristics can be used to create the sim circuit shown in Figure 2.63; this is useful since complementary gates can only provide the NOR operation. The operation of the cir

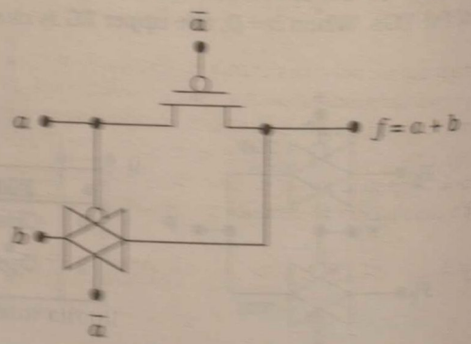


Figure 2.63 A TG-based OR gate

be understood by examining the effect that a has on the switches. If $a = 0$, then the pFET is OFF (since $\bar{a} = 1$ drives it into cutoff) while the TG acts as a closed switch. This gives an output of $f = b$. If $a = 1$, then the pFET is ON and the value of $f = a = 1$ is transmitted to the output. Thus, the output is $f = 1$ if either input is a 1, which establishes the OR operation. We can alternately use logic equations for the TG and the pFET to write the output as

$$\begin{aligned} f &= a \cdot (\overline{\bar{a}}) + \bar{a} \cdot b \\ &= a + \bar{a} \cdot b \\ &= a + b \end{aligned} \tag{2.83}$$

where the last step follows by absorption. This verifies the simpler bit-by-bit analysis.

Alternate XOR/XNOR Circuits

Mixing TGs and FETs as in the OR gate circuit gives rise to many variations for the design of basic logic gates. Many of these designs are for exclusive-OR and equivalence (XNOR) functions due to their importance in adders and error detection/correction algorithms.

An example of this type of circuit is the XNOR network in Figure 2.64. This uses the input pair $(b, \bar{b})$ to control the transmission gate. To understand the operation, remember that the output of an XNOR gate is 1 if and only if the inputs are equal. Suppose that $b = 1$; the TG acts as a closed switch and a is transmitted to the output to give $g = a$. For this case, the output is a 1 iff $a = 1$. The circuit operates differently if $b = 0$. Now, the TG is off and a is directed toward the gates of the Mp/Mn pair. Since $b = 0$ is applied to the source of the nFET Mn and $\bar{b} = 1$ is connected to the source (upper side) of the pFET, the $(b, \bar{b}) = (0, V_{DD})$ pair provides power to the FETs, resulting in an inverter! For this case, the output is $g = \bar{a}$, so that g is 1 iff $a = 0$. This establishes the circuit as an XNOR gate as stated. Interchanging b and $\bar{b}$ gives an XOR gate.

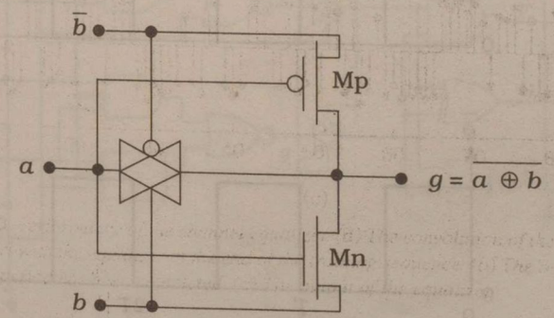


Figure 2.64 An XNOR gate that uses both TGs and FETs

[4] John P. Uyemura, **CMOS Logic Circuit Design**, Kluwer Acade Publishers, Norwell, MA, 1999.

[5] M. Michael Vai, **VLSI Design**, CRC Press, Boca Raton, FL, 2001.

[6] Neil H. E. Weste and Kamran Eshraghian, **Principles of CMOS V Design**, 2nd ed., Addison-Wesley, Reading, MA, 1993.

[7] Wayne Wolf, **Modern VLSI Design**, 2nd ed., Prentice-Hall PTR, Up Saddle River, NJ, 1998.

2.8 Problems

[2.1] Suppose that $V_{DD} = 5$ V and $V_{Tn} = 0.7$ V. Find the output volt V_{out} of the nFET in Figure P2.1 for the following input voltage values $V_{in} = 2$ V; (b) $V_{in} = 4.5$ V; (c) $V_{in} = 3.5$ V; (d) $V_{in} = 0.7$ V.

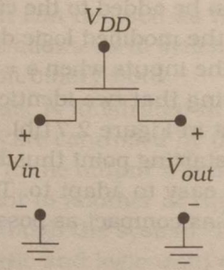


Figure P2.1

[2.2] Consider the two-FET chain in Figure P2.2. The power supply is to a value of $V_{DD} = 3.3$ V and the nFET threshold voltage is $V_{Tn} = 0.5$ Find the output voltage V_{DD} at the right side of the chain for the follow values of V_{in}: (a) $V_{in} = 2.9$ V; (b) $V_{in} = 3.0$ V; (c) $V_{in} = 1.4$ V; (d) $V_{in} = 3.1$

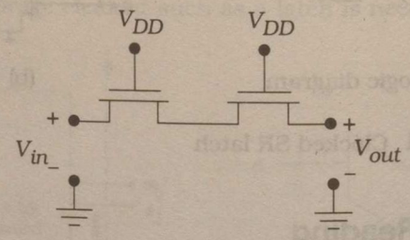


Figure P2.2

[2.3] The output of an nFET is used to drive the gate of another nFE shown in Figure P2.3. Assume that $V_{DD} = 3.3$ V and $V_{Tn} = 0.60$ V. Find output voltage V_{out} when the input voltages are at the following values (a) $V_a = 3.3$ V and $V_b = 3.3$ V; (b) $V_a = 0.5$ V and $V_b = 3.0$ V; (c) $V_a = 2$ and $V_b = 2.5$ V; (d) $V_a = 3.3$ V and $V_b = 1.8$ V.

[2.4] Design a NAND3 gate using an 8:1 MUX.

[2.5] Design a NOR3 gate using an 8:1 MUX as a basis.

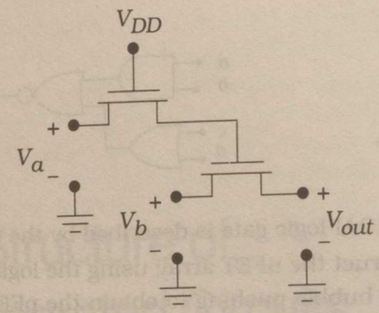


Figure P2.3

[2.6] Consider the 2-input XOR function $a \oplus b$.

(a) Design an XOR gate using a 4:1 MUX.

(b) Modify the circuit in (a) to produce a 2-input XNOR.

(c) A full adder accepts inputs a, b, and c and calculates the sum bit

$$s = a \oplus b \oplus c \tag{2.85}$$

Use your MUX-based gates to design a circuit with this output.

[2.7] Design a CMOS logic gate for the function

$$f = \overline{a \cdot b + a \cdot c + b \cdot d} \tag{2.86}$$

using the smallest number of transistors.

[2.8] Design a CMOS circuit for the OAI expression

$$h = \overline{(a+b) \cdot (a+c) \cdot (b+d)} \tag{2.87}$$

Use the smallest number of transistors in your design.

[2.9] Construct the CMOS logic gate for the function

$$g = \overline{x \cdot (y+z) + y} \tag{2.88}$$

Start with the minimum-transistor nFET network, and then apply bubble pushing to find the pFET wiring.

[2.10] Design a CMOS logic gate circuit that implements

$$F = \overline{a + b \cdot c + a \cdot b \cdot c} \tag{2.89}$$

using series-parallel logic. The objective is to minimize the transistor count.

[2.11] Consider the logic described by the diagram in Figure P2.4. A single, complex logic CMOS gate is to be designed for F.

(a) Construct the nFET array using the logic diagram.

(b) Apply bubble pushing to obtain the pFET logic. Then construct the pFET array using the rules.

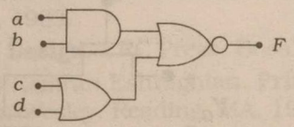


Figure P2.4

[2.12] An AOAI logic gate is described by the schematic in Figure P2.5

(a) Construct the nFET array using the logic diagram.

(b) Apply bubble pushing to obtain the pFET logic. Use the diagram construct the pFET array using the pFET rules.

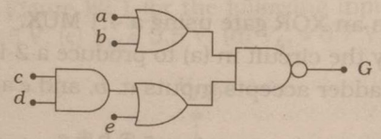


Figure P2.5

[2.13] A pFET logic array is shown in Figure P2.6. Construct the lo diagram using the pFET logic equations. Then construct the nFET circu

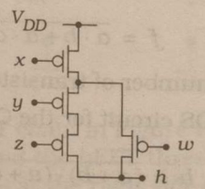


Figure P2.6

[2.14] Design the 4:1 multiplexor circuit that implements the function equation (2.80) by using TG switches.

[2.15] Use an AOI22 gate to design a 2:1 MUX. Inverters are permitted your design.

[2.16] Design a 4:1 MUX using three 2:1 TG multiplexors.

[2.17] A CPU clock ϕ has a frequency 2.1 GHz. What is the period T?

[2.18] Suppose that the hold time for a TG is given as t_{hold} = 120 millise onds (ms). What is the smallest clock frequency that can be used to clo the data flow using a scheme such as that shown in Figure 2.67?

Physical Structure of CMOS Integrated Circuits

3

CMOS integrated circuits are electronic switching networks that are created on small area of a silicon wafer using a complex set of physical and chemical processes. A primary task of the VLSI designer is to translate circuit schematics into silicon form. This process is called **physical design** and is one aspect that separates the field of VLSI from general digital engineering. In this chapter we will examine the structure of a CMOS integrated circuit as seen at the microscopic silicon level in the design hierarchy.

3.1 Integrated Circuit Layers

A silicon integrated circuit can be viewed as a collection of patterned material layers, with each layer having specific conduction properties. The layers may be **metals** that conduct current very well, or they may be **insulators** that block the flow of current. Another material used to create layers is the element **silicon**. It is classified as a **semiconductor**, which means that it is a "partial" conductor. We sometimes refer to both metals and silicon as "conductors," but it is important to distinguish between the two.

An integrated circuit is made by stacking different layers of materials in a specific order to form three-dimensional structures that collectively act as an electronic switching network. Each layer has a predefined pattern that is specified in the system design process. The idea can be understood by referring to Figure 3.1, which portrays two separated layers. The bottom layer is a "sheet" of insulator on a base material called the "substrate." Above this is a patterned material layer of metal that is labeled "Layer M1." The pattern consists of two parallel **lines** of material which are to be placed on top of the insulator in the positions indicated by the