CHAPTER

1

INTRODUCTION

The evolution of digital circuit design

Compelling issues in digital circuit design

How to measure the quality of digital design

Valuable references

1.1 A Historical Perspective

1.2 Issues in Digital Integrated Circuit Design

1.3 Quality Metrics of A Digital Design

1.4 Summary

1.5 To Probe Further
Chapter 1

1.1 A Historical Perspective

The concept of digital data manipulation has made a dramatic impact on our society. One has long grown accustomed to the idea of digital computers. Evolving steadily from mainframe and minicomputers, personal and laptop computers have proliferated into daily life. More significant, however, is a continuous trend towards digital solutions in all other areas of electronics. Instrumentation was one of the first noncomputing domains where the potential benefits of digital data manipulation over analog processing were recognized. Other areas such as control were soon to follow. Only recently have we witnessed the conversion of telecommunications and consumer electronics towards the digital format. Increasingly, telephone data is transmitted and processed digitally over both wired and wireless networks. The compact disk has revolutionized the audio world, and digital video is following in its footsteps.

The idea of implementing computational engines using an encoded data format is by no means an idea of our times. In the early nineteenth century, Babbage envisioned large-scale mechanical computing devices, called *Difference Engines* [Swade93]. Although these engines use the decimal number system rather than the binary representation now common in modern electronics, the underlying concepts are very similar. The Analytical Engine, developed in 1834, was perceived as a general-purpose computing machine, with features strikingly close to modern computers. Besides executing the basic repertoire of operations (addition, subtraction, multiplication, and division) in arbitrary sequences, the machine operated in a two-cycle sequence, called “store” and “mill” (execute), similar to current computers. It even used pipelining to speed up the execution of the addition operation! Unfortunately, the complexity and the cost of the designs made the concept impractical. For instance, the design of Difference Engine I (part of which is shown in Figure 1.1) required 25,000 mechanical parts at a total cost of £17,470 (in 1834!).
Section 1.1 A Historical Perspective

The electrical solution turned out to be more cost effective. Early digital electronics systems were based on magnetically controlled switches (or relays). They were mainly used in the implementation of very simple logic networks. Examples of such are train safety systems, where they are still being used at present. The age of digital electronic computing only started in full with the introduction of the vacuum tube. While originally used almost exclusively for analog processing, it was realized early on that the vacuum tube was useful for digital computations as well. Soon complete computers were realized. The era of the vacuum tube based computer culminated in the design of machines such as the ENIAC (intended for computing artillery firing tables) and the UNIVAC I (the first successful commercial computer). To get an idea about integration density, the ENIAC was 80 feet long, 8.5 feet high and several feet wide and incorporated 18,000 vacuum tubes. It became rapidly clear, however, that this design technology had reached its limits. Reliability problems and excessive power consumption made the implementation of larger engines economically and practically infeasible.

All changed with the invention of the transistor at Bell Telephone Laboratories in 1947 [Bardeen48], followed by the introduction of the bipolar transistor by Schockley in 1949 [Schockley49]. It took till 1956 before this led to the first bipolar digital logic gate, introduced by Harris [Harris56], and even more time before this translated into a set of integrated-circuit commercial logic gates, called the Fairchild Micrologic family [Norman60]. The first truly successful IC logic family, *TTL* (Transistor-Transistor Logic) was pioneered in 1962 [Beeson62]. Other logic families were devised with higher performance in mind. Examples of these are the current switching circuits that produced the first subnanosecond digital gates and culminated in the *ECL* (Emitter-Coupled Logic) family [Masaki74], which is discussed in more detail in this textbook. TTL had the advantage, however, of offering a higher integration density and was the basis of the first integrated circuit revolution. In fact, the manufacturing of TTL components is what spear-headed the first large semiconductor companies such as Fairchild, National, and Texas Instruments. The family was so successful that it composed the largest fraction of the digital semiconductor market until the 1980s.

Ultimately, bipolar digital logic lost the battle for hegemony in the digital design world for exactly the reasons that haunted the vacuum tube approach: the large power consumption per gate puts an upper limit on the number of gates that can be reliably integrated on a single die, package, housing, or box. Although attempts were made to develop high integration density, low-power bipolar families (such as *IIL*—Integrated Injection Logic [Hart72]), the torch was gradually passed to the MOS digital integrated circuit approach.

The basic principle behind the MOSFET transistor (originally called IGFET) was proposed in a patent by J. Lilienfeld (Canada) as early as 1925, and, independently, by O. Heil in England in 1935. Insufficient knowledge of the materials and gate stability problems, however, delayed the practical usability of the device for a long time. Once these were solved, MOS digital integrated circuits started to take off in full in the early 1970s. Remarkably, the first MOS logic gates introduced were of the CMOS variety [Wanlass63], and this trend continued till the late 1960s. The complexity of the manufac-

---

1 An intriguing overview of the evolution of digital integrated circuits can be found in [Murphy93]. (Most of the data in this overview has been extracted from this reference). It is accompanied by some of the historically ground-breaking publications in the domain of digital IC’s.
turing process delayed the full exploitation of these devices for two more decades. Instead, the first practical MOS integrated circuits were implemented in PMOS-only logic and were used in applications such as calculators. The second age of the digital integrated circuit revolution was inaugurated with the introduction of the first microprocessors by Intel in 1972 (the 4004) and 1974 (the 8080) [Shima74]. These processors were implemented in NMOS-only logic, that has the advantage of higher speed over the PMOS logic. Simultaneously, MOS technology enabled the realization of the first high-density semiconductor memories. For instance, the first 4Kbit MOS memory was introduced in 1970 [Hoff70].

These events were at the start of a truly astounding evolution towards ever higher integration densities and speed performances, a revolution that is still in full swing right now. The road to the current levels of integration has not been without hindrances, however. In the late 1970s, NMOS-only logic started to suffer from the same plague that made high-density bipolar logic unattractive or infeasible: power consumption. This realization, combined with progress in manufacturing technology, finally tilted the balance towards the CMOS technology, and this is where we still are today. Interestingly enough, power consumption concerns are rapidly becoming dominant in CMOS design as well, and this time there does not seem to be a new technology around the corner to alleviate the problem.

Although the large majority of the current integrated circuits are implemented in the MOS technology, other technologies come into play when very high performance is at stake. An example of this is the BiCMOS technology that combines bipolar and MOS devices on the same die. BiCMOS is used in high-speed memories and gate arrays. When even higher performance is necessary, other technologies emerge besides the already mentioned bipolar silicon ECL family—Gallium-Arsenide, Silicon-Germanium and even superconducting technologies. These technologies only play a very small role in the overall digital integrated circuit design scene. With the ever increasing performance of CMOS, this role is bound to be further reduced with time. Hence the focus of this textbook on CMOS only.

1.2 Issues in Digital Integrated Circuit Design

Integration density and performance of integrated circuits have gone through an astounding revolution in the last couple of decades. In the 1960s, Gordon Moore, then with Fairchild Corporation and later cofounder of Intel, predicted that the number of transistors that can be integrated on a single die would grow exponentially with time. This prediction, later called *Moore’s law*, has proven to be amazingly visionary. Its validity is best illustrated with the aid of a set of graphs. Figure 1.2 plots the integration density of both logic IC’s and memory as a function of time. As can be observed, integration complexity doubles approximately every 1 to 2 years. As a result, memory density has increased by more than a thousandfold since 1970.

An intriguing case study is offered by the microprocessor. From its inception in the early seventies, the microprocessor has grown in performance and complexity at a steady and predictable pace. The number of transistors and the clock frequency for a number of landmark designs are collected in Figure 1.3. The million-transistor/chip barrier was crossed in the late eighties. Clock frequencies double every three years and have reached
Section 1.2 Issues in Digital Integrated Circuit Design

into the GHz range. This is illustrated in Figure 1.4, which plots the microprocessor trends in terms of complexity and performance at the beginning of the 21st century. An important observation is that, as of now, these trends have not shown any signs of a slow-down.

It should be no surprise to the reader that this revolution has had a profound impact on how digital circuits are designed. Early designs were truly hand-crafted. Every transistor was laid out and optimized individually and carefully fitted into its environment. This is adequately illustrated in Figure 1.5a, which shows the design of the Intel 4004 microprocessor. This approach is, obviously, not appropriate when more than a million devices have to be created and assembled. With the rapid evolution of the design technology, time-to-market is one of the crucial factors in the ultimate success of a component.

Figure 1.2 Evolution of integration complexity of logic ICs and memories as a function of time.

Figure 1.3 Historical evolution of microprocessor transistor count and clock frequency (from [Sasaki91]).
Designers have, therefore, increasingly adhered to rigid design methodologies and strategies that are more amenable to design automation. The impact of this approach is apparent from the layout of one of the later Intel microprocessors, the Pentium, shown in Figure 1.5b. Instead of the individualized approach of the earlier designs, a circuit is constructed in a hierarchical way: a processor is a collection of modules, each of which consists of a number of cells on its own. Cells are reused as much as possible to reduce the design effort and to enhance the chances for a first-time-right implementation. The fact that this hierarchical approach is at all possible is the key ingredient for the success of digital circuit design and also explains why, for instance, very large scale analog design has never caught on.

The obvious next question is why such an approach is feasible in the digital world and not (or to a lesser degree) in analog designs. The crucial concept here, and the most important one in dealing with the complexity issue, is abstraction. At each design level, the internal details of a complex module can be abstracted away and replaced by a black box view or model. This model contains virtually all the information needed to deal with the block at the next level of hierarchy. For instance, once a designer has implemented a multiplier module, its performance can be defined very accurately and can be captured in a model. The performance of this multiplier is in general only marginally influenced by the way it is utilized in a larger system. For all purposes, it can hence be considered a black box with known characteristics. As there exists no compelling need for the system

**Figure 1.4** Microprocessor trends at the beginning of the 21st century. Observe how the fraction of the transistors is being devoted to memory is increasing over time ([Young99]).
Section 1.2  Issues in Digital Integrated Circuit Design

Figure 1.5  Comparing the design methodologies of the Intel 4004 (1971) and Pentium-II™ (1997) microprocessors (reprinted with permission from Intel).
designer to look inside this box, design complexity is substantially reduced. The impact of this *divide and conquer* approach is dramatic. Instead of having to deal with a myriad of elements, the designer has to consider only a handful of components, each of which are characterized in performance and cost by a small number of parameters.

This is analogous to a software designer using a library of software routines such as input/output drivers. Someone writing a large program does not bother to look inside those library routines. The only thing he cares about is the intended result of calling one of those modules. Imagine what writing software programs would be like if one had to fetch every bit individually from the disk and ensure its correctness instead of relying on handy “file open” and “get string” operators.

Typically used abstraction levels in digital circuit design are, in order of increasing abstraction, the device, circuit, gate, functional module (e.g., adder) and system levels (e.g., processor), as illustrated in Figure 1.6. A semiconductor device is an entity with a very complex behavior. No circuit designer will ever seriously consider the solid-state physics equations governing the behavior of the device when designing a digital gate. Instead he will use a simplified model that adequately describes the input-output behavior of the transistor. For instance, an AND gate is adequately described by its Boolean expres-

![Figure 1.6 Design abstraction levels in digital circuits.](image-url)
Section 1.2 Issues in Digital Integrated Circuit Design

...its bounding box, the position of the input and output terminals, and the delay between the inputs and the output.

This design philosophy has been the enabler for the emergence of elaborate computer-aided design (CAD) frameworks for digital integrated circuits; without it the current design complexity would not have been achievable. Design tools include simulation at the various complexity levels, design verification, layout generation, and design synthesis. An overview of these tools and design methodologies is given in Chapter 11 of this textbook.

Furthermore, to avoid the redesign and reverification of frequently used cells such as basic gates and arithmetic and memory modules, designers most often resort to cell libraries. These libraries contain not only the layouts, but also provide complete documentation and characterization of the behavior of the cells. The use of cell libraries is, for instance, apparent in the layout of the Pentium processor (Figure 1.5b). The integer and floating-point unit, just to name a few, contain large sections designed using the so-called standard cell approach. In this approach, logic gates are placed in rows of cells of equal height and interconnected using routing channels. The layout of such a block can be generated automatically given that a library of cells is available.

The preceding analysis demonstrates that design automation and modular design practices have effectively addressed some of the complexity issues incurred in contemporary digital design. This leads to the following pertinent question. If design automation solves all our design problems, why should we be concerned with digital circuit design at all? Will the next-generation digital designer ever have to worry about transistors or parasitics, or is the smallest design entity he will ever consider the gate and the module?

The truth is that the reality is more complex, and various reasons exist as to why an insight into digital circuits and their intricacies will still be an important asset for a long time to come.

- First of all, someone still has to design and implement the module libraries. Semiconductor technologies continue to advance from year to year. Until one has developed a fool-proof approach towards “porting” a cell from one technology to another, each change in technology—which happens approximately every two years—requires a redesign of the library.

- Creating an adequate model of a cell or module requires an in-depth understanding of its internal operation. For instance, to identify the dominant performance parameters of a given design, one has to recognize the critical timing path first.

- The library-based approach works fine when the design constraints (speed, cost or power) are not stringent. This is the case for a large number of application-specific designs, where the main goal is to provide a more integrated system solution, and performance requirements are easily within the capabilities of the technology. Unfortunately for a large number of other products such as microprocessors, success hinges on high performance, and designers therefore tend to push technology to its limits. At that point, the hierarchical approach tends to become somewhat less attractive. To resort to our previous analogy to software methodologies, a programmer tends to “customize” software routines when execution speed is crucial; compilers—or design tools—are not yet to the level of what human sweat or ingenuity can deliver.
• Even more important is the observation that the abstraction-based approach is only correct to a certain degree. The performance of, for instance, an adder can be substantially influenced by the way it is connected to its environment. The interconnection wires themselves contribute to delay as they introduce parasitic capacitances, resistances and even inductances. The impact of the interconnect parasitics is bound to increase in the years to come with the scaling of the technology.

• Scaling tends to emphasize some other deficiencies of the abstraction-based model. Some design entities tend to be global or external (to resort anew to the software analogy). Examples of global factors are the clock signals, used for synchronization in a digital design, and the supply lines. Increasing the size of a digital design has a profound effect on these global signals. For instance, connecting more cells to a supply line can cause a voltage drop over the wire, which, in its turn, can slow down all the connected cells. Issues such as clock distribution, circuit synchronization, and supply-voltage distribution are becoming more and more critical. Coping with them requires a profound understanding of the intricacies of digital circuit design.

• Another impact of technology evolution is that new design issues and constraints tend to emerge over time. A typical example of this is the periodical reemergence of power dissipation as a constraining factor, as was already illustrated in the historical overview. Another example is the changing ratio between device and interconnect parasitics. To cope with these unforeseen factors, one must at least be able to model and analyze their impact, requiring once again a profound insight into circuit topology and behavior.

• Finally, when things can go wrong, they do. A fabricated circuit does not always exhibit the exact waveforms one might expect from advance simulations. Deviations can be caused by variations in the fabrication process parameters, or by the inductance of the package, or by a badly modeled clock signal. Troubleshooting a design requires circuit expertise.

For all the above reasons, it is my belief that an in-depth knowledge of digital circuit design techniques and approaches is an essential asset for a digital-system designer. Even though she might not have to deal with the details of the circuit on a daily basis, the understanding will help her to cope with unexpected circumstances and to determine the dominant effects when analyzing a design.

Example 1.1 Clocks Defy Hierarchy

To illustrate some of the issues raised above, let us examine the impact of deficiencies in one of the most important global signals in a design, the clock. The function of the clock signal in a digital design is to order the multitude of events happening in the circuit. This task can be compared to the function of a traffic light that determines which cars are allowed to move. It also makes sure that all operations are completed before the next one starts—a traffic light should be green long enough to allow a car or a pedestrian to cross the road. Under ideal circumstances, the clock signal is a periodic step waveform with abrupt transitions between the low and the high values (Figure 1.7a).

Consider, for instance, the circuit configuration of Figure 1.7c. The register module samples the value of the input signal at the rising edge of the clock signal $\phi$. This sampled value is preserved and appears at the output until the clock rises anew and a new input is sam-
Section 1.2  Issues in Digital Integrated Circuit Design

pled. Under normal circuit operating conditions, this is exactly what happens, as demonstrated in the simulated response of Figure 1.7d. On the rising edge of clock $\phi$, the input $In$ is sampled and appears at the output $Out$.

Assume now that, due to added loading on the clock signal (for instance, connecting more latches), the clock signal is degenerated, and the clock slopes become less steep (clock $\phi'$ in Figure 1.6d). When the degeneration is within bounds, the functionality of the latch is not impacted. When these bounds are exceeded the latch suddenly starts to malfunction as shown in Figure 1.6d (signal $Out'$). The output signal makes unexpected transitions at the falling clock edge, and extra spikes can be observed as well. Propagation of these erroneous values can cause the digital system to go into an unforeseen mode and crash. This example clearly shows how global effects, such as adding extra load to a clock, can change the behavior of an individual module. Observe that the effects shown are not universal, but are a property of the register circuit used.

Besides the requirement of steep edges, other constraints must be imposed on clock signals to ensure correct operation. A second requirement related to clock alignment, is illustrated in Figure 1.8. The circuit under analysis consists of two cascaded registers, both operating on the rising edge of the clock $\phi$. Under normal operating conditions, the input $In$ gets sampled into the first register on the rising edge of $\phi$ and appears at the output exactly one clock period later. This is confirmed by the simulations shown in Figure 1.7b (signal $Out$).

![Figure 1.7](image)

**Figure 1.7**  Reduced clock slopes can cause a register circuit to fail.
Due to delays associated with routing the clock wires, it may happen that the clocks become misaligned with respect to each other. As a result, the registers are interpreting time indicated by the clock signal differently. Consider the case that the clock signal for the second register is delayed—or skewed—by a value $\delta$. The rising edge of the delayed clock $\phi'$ will postpone the sampling of the input of the second register. If the time it takes to propagate the output of the first register to the input of the second is smaller than the clock delay, the latter will sample the wrong value. This causes the output to change prematurely, as clearly illustrated in the simulation, where the signal $Out'$ goes high at the first rising edge of $\phi'$ instead of the second one.

Clock misalignment, or clock skew, as it is normally called, is another example of how global signals may influence the functioning of a hierarchically designed system. Clock skew is actually one of the most critical design problems facing the designers of large, high-performance systems.

The purpose of this textbook is to provide a bridge between the abstract vision of digital design and the underlying digital circuit and its peculiarities. While starting from a solid understanding of the operation of electronic devices and an in-depth analysis of the nucleus of digital design—the inverter—we will gradually channel this knowledge into the design of more complex entities, such as complex gates, datapaths, registers, controllers, and memories. The persistent quest for a designer when designing each of the mentioned modules is to identify the dominant design parameters, to locate the section of the design he should focus his optimizations on, and to determine the specific properties that make the module under investigation (e.g., a memory) different from any others.
The text also addresses other compelling (global) issues in modern digital circuit design such as power dissipation, interconnect, timing, and synchronization.

1.3 Quality Metrics of A Digital Design

This section defines a set of basic properties of a digital design. These properties help to quantify the quality of a design from different perspectives: cost, functionality, robustness, performance, and energy consumption. Which one of these metrics is most important depends upon the application. For instance, pure speed is a crucial property in a computer server. On the other hand, energy consumption is a dominant metric for hand-held mobile applications such as cell phones. The introduced properties are relevant at all levels of the design hierarchy, be it system, chip, module, and gate. To ensure consistency in the definitions throughout the design hierarchy stack, we propose a bottom-up approach: we start with defining the basic quality metrics of a simple inverter, and gradually expand these to the more complex functions such as gate, module, and chip.

1.3.1 Cost of an Integrated Circuit

The total cost of any product can be separated into two components: the recurring expenses or the variable cost, and the non-recurring expenses or the fixed cost.

Fixed Cost

The fixed cost is independent of the sales volume, the number of products sold. An important component of the fixed cost of an integrated circuit is the effort in time and manpower it takes to produce the design. This design cost is strongly influenced by the complexity of the design, the aggressiveness of the specifications, and the productivity of the designer. Advanced design methodologies that automate major parts of the design process can help to boost the latter. Bringing down the design cost in the presence of an ever-increasing IC complexity is one of the major challenges that is always facing the semiconductor industry. Additionally, one has to account for the indirect costs, the company overhead that cannot be billed directly to one product. It includes amongst others the company’s research and development (R&D), manufacturing equipment, marketing, sales, and building infrastructure.

Variable Cost

This accounts for the cost that is directly attributable to a manufactured product, and is hence proportional to the product volume. Variable costs include the costs of the parts used in the product, assembly costs, and testing costs. The total cost of an integrated circuit is now

\[
\text{cost per IC} = \text{variable cost per IC} + \left( \frac{\text{fixed cost}}{\text{volume}} \right) \quad (1.1)
\]
The impact of the fixed cost is more pronounced for small-volume products. This also explains why it makes sense to have a large design team working for a number of years on a hugely successful product such as a microprocessor.

While the cost of producing a single transistor has dropped exponentially over the past decades, the basic variable-cost equation has not changed:

$$\text{variable cost} = \frac{\text{cost of die} + \text{cost of die test} + \text{cost of packaging}}{\text{final test yield}} \quad (1.2)$$

As will be elaborated on in Chapter 2, the IC manufacturing process groups a number of identical circuits onto a single wafer (Figure). Upon completion of the fabrication, the wafer is chopped into dies, which are then individually packaged after being tested. We will focus on the cost of the dies in this discussion. The cost of packaging and test is the topic of later chapters.

The die cost depends upon the number of good die on a wafer, and the percentage of those that are functional. The latter factor is called the die yield.

$$\text{cost of die} = \frac{\text{cost of wafer}}{\text{dies per wafer} \times \text{die yield}} \quad (1.3)$$

The number of dies per wafer is, in essence, the area of the wafer divided by the die area. The actual situation is somewhat more complicated as wafers are round, and chips are square. Dies around the perimeter of the wafer are therefore lost. The size of the wafer has been steadily increasing over the years, yielding more dies per fabrication run. Eq. (1.3) also presents the first indication that the cost of a circuit is dependent upon the chip area—increasing the chip area simply means that less dies fit on a wafer.

The actual relation between cost and area is more complex, and depends upon the die yield. Both the substrate material and the manufacturing process introduce faults that can cause a chip to fail. Assuming that the defects are randomly distributed over the wafer, and that the yield is inversely proportional to the complexity of the fabrication process, we obtain the following expression of the die yield:
Section 1.3 Quality Metrics of A Digital Design

\[ \text{die yield} = \left(1 + \frac{\text{defects per unit area} \times \text{die area}}{\alpha}\right)^{-\alpha} \quad (1.4) \]

\(\alpha\) is a parameter that depends upon the complexity of the manufacturing process, and is roughly proportional to the number of masks. \(\alpha = 3\) is a good estimate for today’s complex CMOS processes. The defects per unit area is a measure of the material and process induced faults. A value between 0.5 and 1 defects/cm\(^2\) is typical these days, but depends strongly upon the maturity of the process.

Example 1.2 Die Yield

Assume a wafer size of 12 inch, a die size of 2.5 cm\(^2\), 1 defects/cm\(^2\), and \(\alpha = 3\). Determine the die yield of this CMOS process run.

The number of dies per wafer can be estimated with the following expression, which takes into account the lost dies around the perimeter of the wafer.

\[ \text{dies per wafer} = \frac{\pi \times (\text{wafer diameter}/2)^2}{\text{die area}} - \frac{\pi \times \text{wafer diameter}}{\sqrt{2} \times \text{die area}} \]

This means 252 (= 296 - 44) potentially operational dies for this particular example. The die yield can be computed with the aid of Eq. (1.4), and equals 16%! This means that on the average only 40 of the dies will be fully functional.

The bottom line is that the number of functional of dies per wafer, and hence the cost per die is a strong function of the die area. While the yield tends to be excellent for the smaller designs, it drops rapidly once a certain threshold is exceeded. Bearing in mind the equations derived above and the typical parameter values, we can conclude that die costs are proportional to the fourth power of the area:

\[ \text{cost of die} = f(\text{die area})^4 \quad (1.5) \]

The area is a function that is directly controllable by the designer(s), and is the prime metric for cost. Small area is hence a desirable property for a digital gate. The smaller the gate, the higher the integration density and the smaller the die size. Smaller gates furthermore tend to be faster and consume less energy, as the total gate capacitance—which is one of the dominant performance parameters—often scales with the area.

The number of transistors in a gate is indicative for the expected implementation area. Other parameters may have an impact, though. For instance, a complex interconnect pattern between the transistors can cause the wiring area to dominate. The gate complexity, as expressed by the number of transistors and the regularity of the interconnect structure, also has an impact on the design cost. Complex structures are harder to implement and tend to take more of the designers valuable time. Simplicity and regularity is a precious property in cost-sensitive designs.

1.3.2 Functionality and Robustness

A prime requirement for a digital circuit is, obviously, that it performs the function it is designed for. The measured behavior of a manufactured circuit normally deviates from the
expected response. One reason for this aberration are the variations in the manufacturing process. The dimensions, threshold voltages, and currents of an MOS transistor vary between runs or even on a single wafer or die. The electrical behavior of a circuit can be profoundly affected by those variations. The presence of disturbing noise sources on or off the chip is another source of deviations in circuit response. The word *noise* in the context of digital circuits means "unwanted variations of voltages and currents at the logic nodes." Noise signals can enter a circuit in many ways. Some examples of digital noise sources are depicted in Figure 1.10. For instance, two wires placed side by side in an integrated circuit form a coupling capacitor and a mutual inductance. Hence, a voltage or current change on one of the wires can influence the signals on the neighboring wire. Noise on the power and ground rails of a gate also influences the signal levels in the gate.

Most noise in a digital system is internally generated, and the noise value is proportional to the signal swing. Capacitive and inductive cross talk, and the internally-generated power supply noise are examples of such. Other noise sources such as input power supply noise are external to the system, and their value is not related to the signal levels. For these sources, the noise level is directly expressed in Volt or Ampere. Noise sources that are a function of the signal level are better expressed as a fraction or percentage of the signal level. Noise is a major concern in the engineering of digital circuits. How to cope with all these disturbances is one of the main challenges in the design of high-performance digital circuits and is a recurring topic in this book.

The steady-state parameters (also called the *static behavior*) of a gate measure how robust the circuit is with respect to both variations in the manufacturing process and noise disturbances. The definition and derivation of these parameters requires a prior understanding of how digital signals are represented in the world of electronic circuits.

Digital circuits (DC) perform operations on *logical* (or *Boolean*) variables. A logical variable \( x \) can only assume two discrete values:

\[
x \in \{0,1\}
\]

As an example, the inversion (i.e., the function that an inverter performs) implements the following compositional relationship between two Boolean variables \( x \) and \( y \):

\[
y = \overline{x} \quad \{ x = 0 \Rightarrow y = 1; \ x = 1 \Rightarrow y = 0 \}
\]

Figure 1.10  Noise sources in digital circuits.
A logical variable is, however, a mathematical abstraction. In a physical implementation, such a variable is represented by an electrical quantity. This is most often a node voltage that is not discrete but can adopt a continuous range of values. This electrical voltage is turned into a discrete variable by associating a nominal voltage level with each logic state: $1 \iff V_{OH}$, $0 \iff V_{OL}$, where $V_{OH}$ and $V_{OL}$ represent the high and the low logic levels, respectively. Applying $V_{OH}$ to the input of an inverter yields $V_{OL}$ at the output and vice versa. The difference between the two is called the logic or signal swing $V_{sw}$.

$$
V_{OH} = (V_{OL}) \\
V_{OL} = (V_{OH})
$$

(1.7)

**The Voltage-Transfer Characteristic**

Assume now that a logical variable in serves as the input to an inverting gate that produces the variable out. The electrical function of a gate is best expressed by its voltage-transfer characteristic (VTC) (sometimes called the DC transfer characteristic), which plots the output voltage as a function of the input voltage $V_{out} = f(V_{in})$. An example of an inverter VTC is shown in Figure 1.11. The high and low nominal voltages, $V_{OH}$ and $V_{OL}$, can readily be identified—$V_{OH} = f(V_{OL})$ and $V_{OL} = f(V_{OH})$. Another point of interest of the VTC is the gate or switching threshold voltage $V_M$ (not to be confused with the threshold voltage of a transistor), that is defined as $V_M = f(V_M)$. $V_M$ can also be found graphically at the intersection of the VTC curve and the line given by $V_{out} = V_{in}$. The gate threshold voltage presents the midpoint of the switching characteristics, which is obtained when the output of a gate is short-circuited to the input. This point will prove to be of particular interest when studying circuits with feedback (also called sequential circuits).

![Figure 1.11 Inverter voltage-transfer characteristic.](image-url)
alone. The regions of acceptable high and low voltages are delimited by the \( V_{ih} \) and \( V_{il} \) voltage levels, respectively. These represent by definition the points where the gain \((= dV_{out}/dV_{in})\) of the VTC equals \(-1\) as shown in Figure 1.12b. The region between \( V_{ih} \) and \( V_{il} \) is called the undefined region (sometimes also referred to as transition width, or TW). Steady-state signals should avoid this region if proper circuit operation is to be ensured.

**Noise Margins**

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

\[
NM_L = V_{il} - V_{ol} \\
NM_H = V_{oh} - V_{ih}
\]

The noise margins represent the levels of noise that can be sustained when gates are cascaded as illustrated in Figure 1.13. It is obvious that the margins should be larger than 0 for a digital circuit to be functional and by preference should be as large as possible.

**Regenerative Property**

A large noise margin is a desirable, but not sufficient requirement. Assume that a signal is disturbed by noise and differs from the nominal voltage levels. As long as the signal is within the noise margins, the following gate continues to function correctly, although its output voltage varies from the nominal one. This deviation is added to the noise injected at the output node and passed to the next gate. The effect of different noise sources may accumulate and eventually force a signal level into the undefined region. This, fortunately, does not happen if the gate possesses the regenerative property, which ensures that a dis-

![Figure 1.12](image-url)  
(a) Relationship between voltage and logic levels  
(b) Definition of \( V_{iw} \) and \( V_{il} \)  

**Figure 1.12** Mapping logic levels to the voltage domain.
Section 1.3 Quality Metrics of A Digital Design

19

turbed signal gradually converges back to one of the nominal voltage levels after passing through a number of logical stages. This property can be understood as follows:

An input voltage \( v_{in} \) (\( v_{in} \in \{0\} \)) is applied to a chain of \( N \) inverters (Figure 1.14a). Assuming that the number of inverters in the chain is even, the output voltage \( v_{out}(N \to \infty) \) will equal \( V_{OL} \) if and only if the inverter possesses the regenerative property. Similarly, when an input voltage \( v_{in} \) (\( v_{in} \in \{1\} \)) is applied to the inverter chain, the output voltage will approach the nominal value \( V_{OH} \).

**Example 1.3 Regenerative property**

The concept of regeneration is illustrated in Figure 1.14b, which plots the simulated transient response of a chain of CMOS inverters. The input signal to the chain is a step-waveform with a degraded amplitude, which could be caused by noise. Instead of swinging from rail to rail,
\( v_0 \) only extends between 2.1 and 2.9 V. From the simulation, it can be observed that this deviation rapidly disappears, while progressing through the chain; \( v_1 \), for instance, extends from 0.6 V to 4.45 V. Even further, \( v_2 \) already swings between the nominal \( V_{OL} \) and \( V_{OH} \). The inverter used in this example clearly possesses the regenerative property.

The conditions under which a gate is regenerative can be intuitively derived by analyzing a simple case study. Figure 1.15(a) plots the VTC of an inverter \( V_{out} = f(V_{in}) \) as well as its inverse function \( finv() \), which reverts the function of the \( x \)- and \( y \)-axis and is defined as follows:

\[
in = f(out) \Rightarrow in = finv(out)
\]  

Figure 1.15  Conditions for regeneration.

Assume that a voltage \( v_0 \), deviating from the nominal voltages, is applied to the first inverter in the chain. The output voltage of this inverter equals \( v_1 = f(v_0) \) and is applied to the next inverter. Graphically this corresponds to \( v_1 = finv(v_2) \). The signal voltage gradually converges to the nominal signal after a number of inverter stages, as indicated by the arrows. In Figure 1.15(b) the signal does not converge to any of the nominal voltage levels but to an intermediate voltage level. Hence, the characteristic is nonregenerative. The difference between the two cases is due to the gain characteristics of the gates. To be regenerative, the VTC should have a transient region (or undefined region) with a gain greater than 1 in absolute value, bordered by the two legal zones, where the gain should be smaller than 1. Such a gate has two stable operating points. This clarifies the definition of the \( V_{IH} \) and the \( V_{IL} \) levels that form the boundaries between the legal and the transient zones.

**Noise Immunity**

While the noise margin is a meaningful means for measuring the robustness of a circuit against noise, it is not sufficient. It expresses the capability of a circuit to “overpower” a noise source. **Noise immunity**, on the other hand, expresses the ability of the system to pro-
Section 1.3 Quality Metrics of A Digital Design

cess and transmit information correctly in the presence of noise [Dally98]. Many digital circuits with low noise margins have very good noise immunity because they reject a noise source rather than overpower it.

To study the noise immunity of a gate, we have to construct a noise budget that allocates the power budget to the various power sources. As discussed earlier, the noise sources can be divided into sources that are proportional to the signal swing \( V_{Nm} = g \cdot V_{sw} \), and others that are fixed \( V_{Nf} \). We assume, for the sake of simplicity, that the noise margin equals half the signal swing (for both H and L). To operate correctly, the noise margin has to be larger than the sum of the noise values.

\[
V_{NM} = \frac{V_{sw}}{2} \geq \sum V_{Nf} + \sum g_j V_{sw}
\]

(1.10)

Given a set of noise sources, we can derive the minimum signal swing necessary for the system to be operational,

\[
V_{sw} \geq \frac{2 \sum V_{Nf}}{1 - 2 \sum g_j}
\]

(1.11)

This makes it clear that the signal swing (and the noise margin) has to be large enough to overpower the fixed sources. On the other hand, the impact of the internal sources is strongly dependent upon the noise suppressing capabilities of the gates, i.e. the proportionality or gain factors \( g_j \), which should be as small as possible. In later chapters, we will discuss some differential logic families that suppress most of the internal noise, and hence can get away with very small noise margins and signal swings.

Directivity

The directivity property requires a gate to be unidirectional, that is, changes in an output level should not appear at any unchanging input of the same circuit. If not, an output-signal transition reflects to the gate inputs as a noise signal, affecting the signal integrity.

In real gate implementations, full directivity can never be achieved. Some feedback of changes in output levels to the inputs cannot be avoided. Capacitive coupling between inputs and outputs is a typical example of such a feedback. It is important to minimize these changes so that they do not affect the logic levels of the input signals.

Fan-In and Fan-Out

The fan-out denotes the number of load gates \( N \) that are connected to the output of the driving gate (Figure 1.16). Increasing the fan-out of a gate can affect its logic output levels. From the world of analog amplifiers, we know that this effect is minimized by making the input resistance of the load gates as large as possible (minimizing the input currents) and by keeping the output resistance of the driving gate small (reducing the effects of load currents on the output voltage). When the fan-out is large, the added load can deteriorate the dynamic performance of the driving gate. For these reasons, many generic and library
components define a maximum fan-out to guarantee that the static and dynamic performance of the element meet specification.

The fan-in of a gate is defined as the number of inputs to the gate (Figure 1.16b). Gates with large fan-in tend to be more complex, which often results in inferior static and dynamic properties.

The Ideal Digital Gate

Based on the above observations, we can define the ideal digital gate from a static perspective. The ideal inverter model is important because it gives us a metric by which we can judge the quality of actual implementations.

Its VTC is shown in Figure 1.17 and has the following properties: infinite gain in the transition region, and gate threshold located in the middle of the logic swing, with high and low noise margins equal to half the swing. The input and output impedances of the ideal gate are infinity and zero, respectively (i.e., the gate has unlimited fan-out). While this ideal VTC is unfortunately impossible in real designs, some implementations, such as the static CMOS inverter, come close.
Example 1.4 Voltage-Transfer Characteristic

Figure 1.18 shows an example of a voltage-transfer characteristic of an actual, but outdated gate structure (as produced by SPICE in the DC analysis mode). The values of the dc-parameters are derived from inspection of the graph.

\[
\begin{align*}
V_{OH} &= 3.5 \text{ V} ; \quad V_{OL} = 0.45 \text{ V} \\
V_{IH} &= 2.35 \text{ V} ; \quad V_{IL} = 0.66 \text{ V} \\
V_{M} &= 1.64 \text{ V} \\
NM_{H} &= 1.15 \text{ V} ; \quad NM_{L} = 0.21 \text{ V}
\end{align*}
\]

The observed transfer characteristic, obviously, is far from ideal: it is asymmetrical, has a very low value for \(NM_{L}\), and the voltage swing of 3.05V is substantially below the maximum obtainable value of 5 V (which is the value of the supply voltage for this design).

1.3.3 Performance

From a system designer’s perspective, the performance of a digital circuit expresses the computational load that the circuit can manage. For instance, a microprocessor is often characterized by the number of instructions it can execute per second. This performance metric depends both on the architecture of the processor—for instance, the number of instructions it can execute in parallel—and the actual design of logic circuitry. While the former is crucially important, it is not the focus of this textbook. We refer the reader to the many excellent books on this topic [for instance, Patterson96]. When focusing on the pure design, performance is most often expressed by the duration of the clock period (clock cycle time), or its rate (clock frequency). The minimum value of the clock period for a given technology and design is set by a number of factors such as the time it takes for the signals to propagate through the logic, the time it takes to get the data in and out of the
registers, and the uncertainty of the clock arrival times. Each of these topics will be discussed in detail on the course of this text book. At the core of the whole performance analysis, however, lays the performance of an individual gate.

The propagation delay $t_p$ of a gate defines how quickly it responds to a change at its input(s). It expresses the delay experienced by a signal when passing through a gate. It is measured between the 50% transition points of the input and output waveforms, as shown in Figure 1.19 for an inverting gate. Because a gate displays different response times for rising or falling input waveforms, two definitions of the propagation delay are necessary. The $t_{p_{LH}}$ defines the response time of the gate for a low to high (or positive) output transition, while $t_{p_{HL}}$ refers to a high to low (or negative) transition. The propagation delay $t_p$ is defined as the average of the two.

$$t_p = \frac{t_{p_{LH}} + t_{p_{HL}}}{2} \quad (1.12)$$

The propagation delay is not only a function of the circuit technology and topology, but depends upon other factors as well. Most importantly, the delay is a function of the slopes of the input and output signals of the gate. To quantify these properties, we introduce the rise and fall times $t_r$ and $t_f$, which are metrics that apply to individual signal waveforms rather than gates (Figure 1.19), and express how fast a signal transits between the different levels. The uncertainty over when a transition actually starts or ends is avoided by defining the rise and fall times between the 10% and 90% points of the waveforms.

CAUTION: Observe that the propagation delay $t_p$, in contrast to $t_{p_{LH}}$ and $t_{p_{HL}}$, is an artificial gate quality metric, and has no physical meaning per se. It is mostly used to compare different semiconductor technologies, or logic design styles.

The propagation delay is not only a function of the circuit technology and topology, but depends upon other factors as well. Most importantly, the delay is a function of the slopes of the input and output signals of the gate. To quantify these properties, we introduce the rise and fall times $t_r$ and $t_f$, which are metrics that apply to individual signal waveforms rather than gates (Figure 1.19), and express how fast a signal transits between the different levels. The uncertainty over when a transition actually starts or ends is avoided by defining the rise and fall times between the 10% and 90% points of the waveforms.

---

2 The 50% definition is inspired the assumption that the switching threshold $V_{sw}$ is typically located in the middle of the logic swing.
forms, as shown in the Figure. The rise/fall time of a signal is largely determined by the strength of the driving gate, and the load presented by the node itself, which sums the contributions of the connecting gates (fan-out) and the wiring parasitics.

When comparing the performance of gates implemented in different technologies or circuit styles, it is important not to confuse the picture by including parameters such as load factors, fan-in and fan-out. A uniform way of measuring the rise/fall time of a gate, so that technologies can be judged on an equal footing, is desirable. The de-facto standard circuit for delay measurement is the ring oscillator, which consists of an odd number of inverters connected in a circular chain (Figure 1.20). Due to the odd number of inversions, this circuit does not have a stable operating point and oscillates. The period $T$ of the oscillation is determined by the propagation time of a signal transition through the complete chain, or $T = 2 \times t_p \times N$ with $N$ the number of inverters in the chain. The factor 2 results from the observation that a full cycle requires both a low-to-high and a high-to-low transition. Note that this equation is only valid for $2Nt_p >> t_f + t_r$. If this condition is not met, the circuit might not oscillate—one “wave” of signals propagating through the ring will overlap with a successor and eventually dampen the oscillation. Typically, a ring oscillator needs at least five stages to be operational.

---

**CAUTION:** We must be extremely careful with results obtained from ring oscillator measurements. A $t_p$ of 20 psec by no means implies that a circuit built with those gates will operate at 50 GHz. The oscillator results are primarily useful for quantifying the differences between various manufacturing technologies and gate topologies. The oscillator is an idealized circuit where each gate has a fan-in and fan-out of exactly one and parasitic loads are minimal. In more realistic digital circuits, fan-ins and fan-outs are higher, and interconnect delays are non-negligible. The gate functionality is also substantially more complex than a simple invert operation. As a result, the achievable clock frequency on average is 50 to a 100 times slower than the frequency predicted from ring oscillator mea-
measurements. This is an average observation; carefully optimized designs might approach the ideal frequency more closely.

Example 1.5 Propagation Delay of First-Order RC Network

Digital circuits are often modeled as first-order RC networks of the type shown in Figure 1.21. The propagation delay of such a network is thus of considerable interest.

![First-order RC network](image)

When applying a step input (with $v_{in}$ going from 0 to $V$), the transient response of this circuit is known to be an exponential function, and is given by the following expression (where $\tau = RC$, the time constant of the network):

$$v_{out}(t) = (1 - e^{-t/\tau}) V$$  \hspace{1cm} (1.13)

The time to reach the 50% point is easily computed as $t = \ln(2) \tau = 0.69 \tau$. Similarly, it takes $t = \ln(9) \tau = 2.2 \tau$ to get to the 90% point. It is worth memorizing these numbers, as they are extensively used in the rest of the text.

1.3.4 Power and Energy Consumption

The power consumption of a design determines how much energy is consumed per operation, and much heat the circuit dissipates. These factors influence a great number of critical design decisions, such as the power-supply capacity, the battery lifetime, supply-line sizing, packaging and cooling requirements. Therefore, power dissipation is an important property of a design that affects feasibility, cost, and reliability. In the world of high-performance computing, power consumption limits, dictated by the chip package and the heat removal system, determine the number of circuits that can be integrated onto a single chip, and how fast they are allowed to switch. With the increasing popularity of mobile and distributed computation, energy limitations put a firm restriction on the number of computations that can be performed given a minimum time between battery recharges.

Depending upon the design problem at hand, different dissipation measures have to be considered. For instance, the peak power $P_{peak}$ is important when studying supply-line sizing. When addressing cooling or battery requirements, one is predominantly interested in the average power dissipation $P_{av}$. Both measures are defined in equation Eq. (1.14):

$$P_{peak} = i_{peak} V_{supply} = \max[p(t)]$$
$$P_{av} = \frac{1}{T} \int_0^T p(t) dt = \frac{V_{supply}}{T} \int_0^T i_{supply}(t) dt$$  \hspace{1cm} (1.14)
where \( p(t) \) is the instantaneous power, \( i_{\text{supply}} \) is the current being drawn from the supply voltage \( V_{\text{supply}} \) over the interval \( t \in [0, T] \), and \( i_{\text{peak}} \) is the maximum value of \( i_{\text{supply}} \) over that interval.

The dissipation can further be decomposed into static and dynamic components. The latter occurs only during transients, when the gate is switching. It is attributed to the charging of capacitors and temporary current paths between the supply rails, and is, therefore, proportional to the switching frequency: the higher the number of switching events, the higher the dynamic power consumption. The static component on the other hand is present even when no switching occurs and is caused by static conductive paths between the supply rails or by leakage currents. It is always present, even when the circuit is in stand-by. Minimization of this consumption source is a worthwhile goal.

The propagation delay and the power consumption of a gate are related—the propagation delay is mostly determined by the speed at which a given amount of energy can be stored on the gate capacitors. The faster the energy transfer (or the higher the power consumption), the faster the gate. For a given technology and gate topology, the product of power consumption and propagation delay is generally a constant. This product is called the power-delay product (or PDP) and can be considered as a quality measure for a switching device. The PDP is simply the energy consumed by the gate per switching event. The ring oscillator is again the circuit of choice for measuring the PDP of a logic family.

An ideal gate is one that is fast, and consumes little energy. The energy-delay product (E-D) is a combined metric that brings those two elements together, and is often used as the ultimate quality metric. From the above, it should be clear that the E-D is equivalent to \( \text{power-delay}^2 \).

**Example 1.6 Energy Dissipation of First-Order RC Network**

Let us consider again the first-order RC network shown in Figure 1.21. When applying a step input (with \( V_\text{in} \) going from 0 to \( V \)), an amount of energy is provided by the signal source to the network. The total energy delivered by the source (from the start of the transition to the end) can be readily computed:

\[
E_\text{in} = \int_0^\infty i(t) v_\text{in}(t) \, dt = V \int_0^\infty C \frac{dv_{\text{out}}}{dt} \, dt = (CV) \int_0^V dv_{\text{out}} = CV^2 \quad (1.15)
\]

It is interesting to observe that the energy needed to charge a capacitor from 0 to \( V \) volt with a step input is a function of the size of the voltage step and the capacitance, but is independent of the value of the resistor. We can also compute how much of the delivered energy gets stored on the capacitor at the end of the transition.

\[
E_\text{C} = \int_0^\infty C(t) v_{\text{out}}(t) \, dt = \int_0^\infty C \frac{dv_{\text{out}}}{dt} v_{\text{out}} \, dt = CV_{\text{out}} v_{\text{out}} \int_0^V dv_{\text{out}} = CV_{\text{out}}^2 \quad (1.16)
\]

This is exactly half of the energy delivered by the source. For those who wonder happened with the other half—a simple analysis shows that an equivalent amount gets dissipated as heat in the resistor during the transition. We leave it to the reader to demonstrate that dur-
ing the discharge phase (for a step from $V$ to 0), the energy originally stored on the capacitor gets dissipated in the resistor as well, and turned into heat.

1.4 Summary

In this introductory chapter, we learned about the history and the trends in digital circuit design. We also introduced the important quality metrics, used to evaluate the quality of a design: cost, functionality, robustness, performance, and energy/power dissipation. At the end of the Chapter, you can find an extensive list of reference works that may help you to learn more about some of the topics introduced in the course of the text.

1.5 To Probe Further

The design of digital integrated circuits has been the topic of a multitude of textbooks and monographs. To help the reader find more information on some selected topics, an extensive list of reference works is listed below. The state-of-the-art developments in the area of digital design are generally reported in technical journals or conference proceedings, the most important of which are listed.

JOURNALS AND PROCEEDINGS

*IEEE Journal of Solid-State Circuits*
*IEICE Transactions on Electronics (Japan)*
*Proceedings of The International Solid-State and Circuits Conference (ISSCC)*
*Proceedings of the Integrated Circuits Symposium*
*European Solid-State Circuits Conference (ESSCIRC)*

REFERENCE BOOKS

*MOS*
Section 1.5 To Probe Further


**High-Performance Design**

**Low-Power Design**

**Memory Design**

**Interconnections and Packaging**

**Design Tools and Methodologies**

**Bipolar and BiCMOS**
INTRODUCTION  Chapter 1

General

REFERENCES

Section 1.6 Exercises

1. [E, None, 1.2] Based on the evolutionary trends described in the chapter, predict the integration complexity and the clock speed of a microprocessor in the year 2010. Determine also how much DRAM should be available on a single chip at that point in time, if Moore's law would still hold.

2. [D, None, 1.2] By scanning the literature, find the leading-edge devices at this point in time in the following domains: microprocessor, SRAM, and DRAM. Determine for each of those, the number of integrated devices, the overall area and the maximum clock speed. Evaluate the match with the trends predicted in section 1.2.

3. [D, None, 1.2] Find in the library the latest November issue of the Journal of Solid State Circuits. For each of the papers, determine its application class (such as microprocessor, signal processor, DRAM, SRAM, Gate Array), the type of manufacturing technology used (MOS, bipolar, etc.), the minimum feature size, the number of devices on a single die, and the maximum clock speed. Tabulate the results along the various application classes.

4. [E, None, 1.2] Provide at least three examples for each of the abstraction levels described in Figure 1.6.
Chapter 2

The Manufacturing Process

Overview of manufacturing process

Design rules

IC packaging

Future Trends in Integrated Circuit Technology

2.1 Introduction

2.2 Manufacturing CMOS Integrated Circuits
   2.2.1 The Silicon Wafer
   2.2.2 Photolithography
   2.2.3 Some Recurring Process Steps
   2.2.4 Simplified CMOS Process Flow

2.3 Design Rules — The Contract between Designer and Process Engineer

2.4 Packaging Integrated Circuits
   2.4.1 Package Materials
   2.4.2 Interconnect Levels
   2.4.3 Thermal Considerations in Packaging

2.5 Perspective — Trends in Process Technology
   2.5.1 Short-Term Developments
   2.5.2 In the Longer Term

2.6 Summary
2.1 Introduction

Most digital designers will never be confronted with the details of the manufacturing process that lies at the core of the semiconductor revolution. Yet, some insight in the steps that lead to an operational silicon chip comes in quite handy in understanding the physical constraints that are imposed on a designer of an integrated circuit, as well as the impact of the fabrication process on issues such as cost.

In this chapter, we briefly describe the steps and techniques used in a modern integrated circuit manufacturing process. It is not our aim to present a detailed description of the fabrication technology, which easily deserves a complete course [Plummer00]. Rather we aim at presenting the general outline of the flow and the interaction between the various steps. We learn that a set of optical masks forms the central interface between the intrinsics of the manufacturing process and the design that the user wants to see transferred to the silicon fabric. The masks define the patterns that, when transcribed onto the different layers of the semiconductor material, form the elements of the electronic devices and the interconnecting wires. As such, these patterns have to adhere to some constraints in terms of minimum width and separation if the resulting circuit is to be fully functional. This collection of constraints is called the design rule set, and acts as the contract between the circuit designer and the process engineer. If the designer adheres to these rules, he gets a guarantee that his circuit will be manufacturable. An overview of the common design rules, encountered in modern CMOS processes, will be given. Finally, an overview is given of the IC packaging options. The package forms the interface between the circuit implemented on the silicon die and the outside world, and as such has a major impact on the performance, reliability, longevity, and cost of the integrated circuit.

2.2 Manufacturing CMOS Integrated Circuits

A simplified cross section of a typical CMOS inverter is shown in Figure 2.1. The CMOS process requires that both n-channel (NMOS) and p-channel (PMOS) transistors be built in the same silicon material. To accommodate both types of devices, special regions called wells must be created in which the semiconductor material is opposite to the type of the channel. A PMOS transistor has to be created in either an n-type substrate or an n-well, while an NMOS device resides in either a p-type substrate or a p-well. The cross section
shown in Figure 2.1 features an *n*-well CMOS process, where the NMOS transistors are implemented in the *p*-doped substrate, and the PMOS devices are located in the *n*-well. Increasingly, modern processes are using a *dual-well* approach that uses both *n*- and *p*-wells, grown on top on an epitaxial layer, as shown in Figure 2.2. We will restrict the remainder of this discussion to the latter process (without loss of generality).

![Cross section of modern dual-well CMOS process.](image)

The CMOS process requires a large number of steps, each of which consists of a sequence of basic operations. A number of these steps and/or operations are executed very repetitively in the course of the manufacturing process. Rather than diving directly into a description of the overall process flow, we first discuss the starting material followed by a detailed perspective on some of the most-often recurring operations.

### 2.2.1 The Silicon Wafer

The base material for the manufacturing process comes in the form of a single-crystalline, lightly doped *wafer*. These wafers have typical diameters between 4 and 12 inches (10 and 30 cm, respectively) and a thickness of at most 1 mm, and are obtained by cutting a single-crystal ingot into thin slices (Figure 2.3). A starting wafer of the *p*-type might be doped around the levels of $2 \times 10^{21}$ impurities/m$^3$. Often, the surface of the wafer is doped more heavily, and a single crystal *epitaxial layer* of the opposite type is grown over the surface before the wafers are handed to the processing company. One important metric is the defect density of the base material. High defect densities lead to a larger fraction of non-functional circuits, and consequently an increase in cost of the final product.

![Single-crystal ingot and sliced wafers (from Fullman)](image)
2.2.2 Photolithography

In each processing step, a certain area on the chip is masked out using the appropriate optical mask so that a desired processing step can be selectively applied to the remaining regions. The processing step can be any of a wide range of tasks including oxidation, etching, metal and polysilicon deposition, and ion implantation. The technique to accomplish this selective masking, called photolithography, is applied throughout the manufacturing process. Figure 2.4 gives a graphical overview of the different operations involved in a typical photolithographic process. The following steps can be identified:

1. Oxidation layering — this optional step deposits a thin layer of SiO₂ over the complete wafer by exposing it to a mixture of high-purity oxygen and hydrogen at ± 1000°C. The oxide is used as an insulation layer and also forms transistor gates.

2. Photoresist coating — a light-sensitive polymer (similar to latex) is evenly applied while spinning the wafer to a thickness of approximately 1 µm. This material is originally soluble in an organic solvent, but has the property that the polymers cross-link when exposed to light, making the affected regions insoluble. A photoresist of
this type is called negative. A positive photoresist has the opposite properties; originally insoluble, but soluble after exposure. By using both positive and negative resists, a single mask can sometimes be used for two steps, making complementary regions available for processing. Since the cost of a mask is increasing quite rapidly with the scaling of technology, a reduction of the number of masks is surely of high priority.

3. **Stepper exposure** — a glass mask (or reticle), containing the patterns that we want to transfer to the silicon, is brought in close proximity to the wafer. The mask is opaque in the regions that we want to process, and transparent in the others (assuming a negative photoresist). The glass mask can be thought of as the negative of one layer of the microcircuit. The combination of mask and wafer is now exposed to ultra-violet light. Where the mask is transparent, the photoresist becomes insoluble.

4. **Photoresist development and bake** — the wafers are developed in either an acid or base solution to remove the non-exposed areas of photoresist. Once the exposed photoresist is removed, the wafer is "soft-baked" at a low temperature to harden the remaining photoresist.

5. **Acid Etching** — material is selectively removed from areas of the wafer that are not covered by photoresist. This is accomplished through the use of many different types of acid, base and caustic solutions as a function of the material that is to be removed. Much of the work with chemicals takes place at large wet benches where special solutions are prepared for specific tasks. Because of the dangerous nature of some of these solvents, safety and environmental impact is a primary concern.

6. **Spin, rinse, and dry** — a special tools (called SRD) cleans the wafer with deionized water and dries it with nitrogen. The microscopic scale of modern semiconductor devices means that even the smallest particle of dust or dirt can destroy the circuitry. To prevent this from happening, the processing steps are performed in ultra-clean rooms where the number of dust particles per cubic foot of air ranges between 1 and 10. Automatic wafer handling and robotics are used whenever possible. This explains why the cost of a state-of-the-art fabrication facility easily ranges in the multiple billions of dollars. Even then, the wafers must be constantly cleaned to avoid contamination, and to remove the left-over of the previous process steps.

7. **Various process steps** — the exposed area can now be subjected to a wide range of process steps, such as ion implantation, plasma etching, or metal deposition. These are the subjects of the subsequent section.

8. **Photoresist removal (or ashing)** — a high-temperature plasma is used to selectively remove the remaining photoresist without damaging device layers.

We illustrate the use of the photolithographic process for one specific example, the patterning of a layer of SiO\(_2\), in Figure 2.5. The sequence of process steps shown in the Figure patterns exactly one layer of the semiconductor material, and may seem very complex. Yet, the reader has to bear in mind that that same sequence patterns the layer of the **complete surface of the wafer**. It is hence a very parallel process, transferring hundreds of millions of patterns to the semiconductor surface simultaneously. The concurrent and
scalable nature of the optolithographical process is what makes the cheap manufacturing of complex semiconductor circuits possible, and lies at the core of the economic success of the semiconductor industry.

The continued scaling of the minimum feature sizes in integrated circuits puts an enormous burden on the developer of semiconductor manufacturing equipment. This is especially true for the optolithographical process. The dimensions of the features to be transcribed approach the wavelengths of the optical light sources, so that achieving the necessary resolution and accuracy becomes harder and harder. So far, ingenious engineering has extended the lifetime of this process at least until the 100 nm (or 0.1 µm) process generation. Beyond that point, other solutions that offer a finer resolution such as X-ray or electron-beam may be needed. These techniques, while fully functional, are currently less attractive from an economic viewpoint.
2.2.3 Some Recurring Process Steps

Diffusion and Ion Implantation

Many steps of the integrated circuit manufacturing process require a change in the dopant concentration of some parts of the material. The creation of the source and drain regions, well and substrate contacts, the doping of the polysilicon, and the adjustments of the device threshold are examples of such. There exist two approaches for introducing these dopants—diffusion and ion implantation. In both techniques, the area to doped is exposed, while the rest of the wafer is coated with a layer of buffer material, typically SiO$_2$.

In diffusion implantation, the wafers are placed in a quartz tube embedded in a heated furnace. A gas containing the dopant is introduced in the tube. The high temperatures of the furnace, typically 900 to 1100 °C, cause the dopants to diffuse into the exposed surface both vertically and horizontally. The final dopant concentration is the greatest at the surface and decreases in a gaussian profile deeper in the material.

In ion implantation, dopants are introduced as ions into the material. The ion implantation system directs and sweeps a beam of purified ions over the semiconductor surface. The acceleration of the ions determines how deep they will penetrate the material, while the beam current and the exposure time determine the dosage. The ion implantation method allows for an independent control of depth and dosage. This is the reason that ion implantation has largely displaced diffusion in modern semiconductor manufacturing.

Ion implantation has some unfortunate side effects however, the most important one being lattice damage. Nuclear collisions during the high energy implantation cause the displacement of substrate atoms, leading to material defects. This problem is largely resolved by applying a subsequent annealing step, in which the wafer is heated to around 1000°C for 15 to 30 minutes, and then allowed to cool slowly. The heating step thermally vibrates the atoms, which allows the bonds to reform.

Deposition

Any CMOS process requires the repetitive deposition of layers of a material over the complete wafer, to either act as buffers for a processing step, or as insulating or conducting layers. We have already discussed the oxidation process, which allows a layer of SiO$_2$ to be grown. Other materials require different techniques. For instance, silicon nitride (Si$_3$N$_4$) is used as a sacrificial buffer material during the formation of the field oxide and the introduction of the stopper implants. This silicon nitride is deposited everywhere using a process called chemical vapor deposition or CVD, which uses a gas-phase reaction with energy supplied by heat at around 850°C.

Polysilicon, on the other hand, is deposited using a chemical deposition process, which flows silane gas over the heated wafer coated with SiO$_2$ at a temperature of approximately 650°C. The resulting reaction produces a non-crystalline or amorphous material called polysilicon. To increase to conductivity of the material, the deposition has to be followed by an implantation step.

The Aluminum interconnect layers are typically deployed using a process known as sputtering. The aluminum is evaporated in a vacuum, with the heat for the evaporation...
delivered by electron-beam or ion-beam bombarding. Other metallic interconnect materials such as Copper require different deposition techniques.

**Etching**

Once a material has been deposited, etching is used to selectively form patterns such as wires and contact holes. The *wet etching* process was described earlier, and makes use of acid acid or basic solutions. For instance, hydrofluoric acid buffered with ammonium fluoride is typically used to etch SiO$_2$.

In recent years, *dry* or *plasma etching* has made a lot of inroad. A wafer is placed into the etch tool's processing chamber and given a negative electrical charge. The chamber is heated to 100°C and brought to a vacuum level of 10 millitorrs, then filled with a positively charged plasma (usually a mix of nitrogen, chlorine and boron trichloride). The opposing electrical charges cause the rapidly moving plasma molecules to align themselves in a vertical direction, forming a microscopic chemical and physical "sandblasting" action which removes the exposed material. Plasma etching has the advantage of offering a well-defined directionality to the etching action, creating patterns with sharp vertical contours.

**Planarization**

To reliably deposit a layer of material onto the semiconductor surface, it is essential that the surface is approximately flat. If no special steps were taken, this would definitely not be the case in modern CMOS processes, where multiple patterned metal interconnect layers are superimposed onto each other. Therefore, a *chemical-mechanical planarization* (CMP) step is included before the deposition of an extra metal layer on top of the insulating SiO$_2$ layer. This process uses a slurry compound—a liquid carrier with a suspended abrasive component such as aluminum oxide or silica—to microscopically plane a device layer and to reduce the step heights.

### 2.2.4 Simplified CMOS Process Flow

The gross outline of a potential CMOS process flow is given in Figure 2.6. The process starts with the definition of the *active regions*, this is the regions where transistors will be constructed. All other areas of the die will be covered with a thick layer of silicon dioxide (SiO$_2$), called the *field oxide*. This oxide acts as the insulator between neighboring devices, and is either grown (as in the process of Figure 2.1), or deposited in etched trenches (Figure 2.2)—hence the name *trench insulation*. Further insulation is provided by the addition of a reverse-biased np-diode, formed by adding an extra p' region, called the *channel-stop implant* (or *field implant*) underneath the field oxide. Next, lightly doped p- and n-wells are formed through ion implantation. To construct an NMOS transistor in a p-well, heavily doped n-type source and drain regions are implanted (or diffused) into the lightly doped p-type substrate. A thin layer of SiO$_2$, called the *gate oxide*, separates the region between the source and drain, and is itself covered by conductive polycrystalline silicon (or polysilicon, for short). The conductive material forms the *gate* of the transistor. PMOS transistors are constructed in an n-well in a similar fashion (just reverse n’s and
Section 2.2 Manufacturing CMOS Integrated Circuits

Multiple insulated layers of metallic (most often Aluminum) wires are deposited on top of these devices to provide for the necessary interconnections between the transistors.

A more detailed breakdown of the flow into individual process steps and their impact on the semiconductor material is shown graphically in Figure 2.7. While most of the operations should be self-explanatory in light of the previous descriptions, some comments on individual operations are worthwhile. The process starts with a \( p \) -substrate surfaced with a lightly doped \( p \) -epitaxial layer (a). A thin layer of \( \text{SiO}_2 \) is deposited, which will serve as the gate oxide for the transistors, followed by a deposition of a thicker sacrificial silicon nitride layer (b). A plasma etching step using the complimentary of the active area mask creates the trenches, used for insulating the devices (c). After providing the channel stop implant, the trenches are filled with \( \text{SiO}_2 \) followed by a number of steps to provide a flat surface (including inverse active pattern oxide etching, and chemical-mechanical planarization). At that point, the sacrificial nitride is removed (d). The \( n \) -well mask is used to expose only the \( n \) -well areas (the rest of the wafer is covered by a thick buffer material), after which an implant-annealing sequence is applied to adjust the well-doping. This is followed by a second implant step to adjust the threshold voltages of the PMOS transistors. This implant only impacts the doping in the area just below the gate oxide (e). Similar operations (using other dopants) are performed to create the \( p \) -wells, and to adjust the thresholds of the NMOS transistors (f). A thin layer of polysilicon is chemically deposited, and patterned with the aid of the polysilicon mask. Polysilicon is used both as gate electrode material for the transistors as well as an interconnect medium (g). Consecutive ion implantations are used to dope the source and drain regions of the PMOS (\( p^+ \)) and NMOS (\( n^+ \)) transistors, respectively (h), after which the thin gate oxide not covered by the polysilicon is etched away\(^1\). The same implants are also use to dope the

---

\(^1\) Most modern processes also include extra implants for the creation of the lightly-doped drain regions (LDD), and the creation of gate spacers at this point. We have omitted these for the sake of simplicity.
(a) Base material: p+ substrate with p-epi layer

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

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

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

(e) After n-well and $V_{Tn}$ adjust implants

(f) After p-well and $V_{Tp}$ adjust implants
Section 2.2 Manufacturing CMOS Integrated Circuits

(g) After polysilicon deposition and etch

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

(i) After deposition of SiO$_2$ insulator and contact hole etch.

(j) After deposition and patterning of first Al layer.

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

Figure 2.7 Process flow for the fabrication of an NMOS and a PMOS transistor in a dual-well CMOS process. Be aware that the drawings are stylized for understanding, and that the aspect ratios are not proportioned to reality.
polysilicon on the surface, reducing its resistivity. Undoped polysilicon has a very high resistivity. Note that the polysilicon gate, which is patterned before the doping, actually defines the precise location of the channel region, and hence the location of the source and drain regions. This procedure allows for a very precise positioning of the two regions relative to the gate, and hence is called the self-aligned process. The process continues with the deposition of the metallic interconnect layers. These consists of a repetition of the following steps (i-k): deposition of the insulating material (most often SiO₂), etching of the contact or via holes, deposition of the metal (most often Aluminum, although Tungsten is often used for the lower layers), and patterning of the metal. Intermediate planarization steps ensure that the surface remains reasonable flat, even in the presence of multiple interconnect layers. After the last level of metal is deposited, a final passivation or over-glass is deposited for protection. The layer would be CVD SiO₂, although often an additional layer of nitride is deposited as it is more impervious to moisture. The final processing step is to etch openings to the pads used for bonding.

A cross-section of the final artifact is shown in Figure 2.8. Observe how the transistors occupy only a small fraction of the total height of the structure. The interconnect layers take up the majority of the vertical dimension.

2.3 Design Rules — The Contract between Designer and Process Engineer

As processes become more complex, requiring the designer to understand the intricacies of the fabrication process and interpret the relations between the different masks is a sure road to trouble. The goal of defining a set of design rules is to allow for a ready translation of a circuit concept into an actual geometry in silicon. The design rules act as the interface or even the contract between the circuit designer and the process engineer.

Circuit designers in general want tighter, smaller designs, which lead to higher performance and higher circuit density. The process engineer, on the other hand, wants a reproducible and high-yield process. Design rules are, consequently, a compromise that attempts to satisfy both sides.
The design rules provide a set of guidelines for constructing the various masks needed in the patterning process. They consist of minimum-width and minimum-spacing constraints and requirements between objects on the same or on different layers.

The fundamental unity in the definition of a set of design rules is the minimum line width. It stands for the minimum mask dimension that can be safely transferred to the semiconductor material. In general, the minimum line width is set by the resolution of the patterning process, which is most commonly based on optical lithography. More advanced approaches use electron-beam or X-ray sources that offer a finer resolution, but are less attractive from an economical viewpoint.

Even for the same minimum dimension, design rules tend to differ from company to company, and from process to process. This makes porting an existing design between different processes a time-consuming task. One approach to address this issue is to use advanced CAD techniques, which allow for migration between compatible processes. Another approach is to use scalable design rules. The latter approach, made popular by Mead and Conway [Mead80], defines all rules as a function of a single parameter, most often called \( \lambda \). The rules are chosen so that a design is easily ported over a cross section of industrial processes. Scaling of the minimum dimension is accomplished by simply changing the value of \( \lambda \). This results in a linear scaling of all dimensions. For a given process, \( \lambda \) is set to a specific value, and all design dimensions are consequently translated into absolute numbers. Typically, the minimum line width of a process is set to \( 2\lambda \). For instance, for a 0.25 \( \mu \)m process (i.e., a process with a minimum line width of 0.25 \( \mu \)m), \( \lambda \) equals 0.125 \( \mu \)m.

This approach, while attractive, suffers from some disadvantages:

1. Linear scaling is only possible over a limited range of dimensions (for instance, between 0.25 \( \mu \)m and 0.15 \( \mu \)m). When scaling over larger ranges, the relations between the different layers tend to vary in a nonlinear way that cannot be adequately covered by the linear scaling rules.

2. Scalable design rules are conservative. As they represent a cross section over different technologies, they have to represent the worst-case rules for the whole set. This results in overdimensioned and less-dense designs.

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

For this textbook, we have selected a “vanilla” 0.25 \( \mu \)m CMOS process as our preferred implementation medium. The rest of this section is devoted to a short introduction and overview of the design rules of this process, which fall in the micron-rules class. A complete design-rule set consists of the following entities: a set of layers, relations between objects on the same layer, and relations between objects on different layers. We discuss each of them in sequence.
Layer Representation

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

- **Substrates** and/or **wells**, being $p$-type (for NMOS devices) and $n$-type (for PMOS)
- **Diffusion regions** ($n^+$ and $p^+$) defining the areas where transistors can be formed. These regions are often called the **active areas**. Diffusions of an inverse type are needed to implement contacts to the wells or to the substrate. These are called **select regions**.
- One or more **polysilicon** layers, which are used to form the gate electrodes of the transistors (but serve as interconnect layers as well).
- **A number of metal interconnect layers**.
- **Contact and via layers** to provide interlayer connections.

A layout consists of a combination of polygons, each of which is attached to a certain layer. The functionality of the circuit is determined by the choice of the layers, as well as the interplay between objects on different layers. For instance, an MOS transistor is formed by the cross section of the diffusion layer and the polysilicon layer. An interconnection between two metal layers is formed by a cross section between the two metal layers and an additional contact layer. To visualize these relations, each layer is assigned a standard color (or stipple pattern for a black-and-white representation). The different layers used in our CMOS process are represented in Colorplate 1 (color insert).

Intralayer Constraints

A first set of rules defines the minimum dimensions of objects on each layer, as well as the minimum spacings between objects on the same layer. All distances are expressed in $\mu$m. These constraints are presented in a pictorial fashion in Colorplate 2.

Interlayer Constraints

These rules tend to be more complex. The fact that multiple layers are involved makes it harder to visualize their meaning or functionality. Understanding layout requires the capability of translating the two-dimensional picture of the layout drawing into the three-dimensional reality of the actual device. This takes some practice.

We present these rules in a set of separate groupings.

1. **Transistor Rules** (Colorplate 3). A transistor is formed by the overlap of the active and the polysilicon layers. From the intralayer design rules, it is already clear that the minimum length of a transistor equals 0.24 $\mu$m (the minimum width of polysilicon), while its width is at least 0.3 $\mu$m (the minimum width of diffusion). Extra rules include the spacing between the active area and the well boundary, the gate overlap of the active area, and the active overlap of the gate.
2. Contact and Via Rules (Colorplates 2 and 4). A contact (which forms an interconnection between metal and active or polysilicon) or a via (which connects two metal layers) is formed by overlapping the two interconnecting layers and providing a contact hole, filled with metal, between the two. In our process, the minimum size of the contact hole is 0.3 µm, while the polysilicon and diffusion layers have to extend at least over 0.14 µm beyond the area of the contact hole. This sets the minimum area of a contact to 0.44 µm × 0.44 µm. This is larger than the dimensions of a minimum-size transistor! Excessive changes between interconnect layers are thus to be avoided. The figure, furthermore, points out the minimum spacings between contact and via holes, as well as their relationship with the surrounding layers.

Well and Substrate Contacts (Colorplate 5). For robust digital circuit design, it is important for the well and substrate regions to be adequately connected to the supply voltages. Failing to do so results in a resistive path between the substrate contact of the transistors and the supply rails, and can lead to possibly devastating parasitic effects, such as latchup. It is therefore advisable to provide numerous substrate (well) contacts spread over the complete region. To establish an ohmic contact between a supply rail, implemented in metal1, and a p-type material, a p⁺ diffusion region must be provided. This is enabled by the select layer, which reverses the type of diffusion. A number of rules regarding the use of the select layer are illustrated in Colorplate 5.

Consider an n-well process, which implements the PMOS transistors into an n-type well diffused in a p-type material. The nominal diffusion is p⁺. To invert the polarity of the diffusion, an n-select layer is provided that helps to establish the n⁺ diffusions for the well-contacts in the n-region as well as the n⁺ source and drain regions for the NMOS transistors in the substrate.

Verifying the Layout

Ensuring that none of the design rules is violated is a fundamental requirement of the design process. Failing to do so will almost surely lead to a nonfunctional design. Doing so for a complex design that can contain millions of transistors is no sinecure, especially when taking into account the complexity of some design-rule sets. While design teams used to spend numerous hours staring at room-size layout plots, most of this task is now done by computers. Computer-aided Design-Rule Checking (called DRC) is an integral part of the design cycle for virtually every chip produced today. A number of layout tools even perform on-line DRC and check the design in the background during the time of conception.

Example 2.1 Layout Example

An example of a complete layout containing an inverter is shown in Figure 2.9. To help the visualization process, a vertical cross section of the process along the design center is included as well as a circuit schematic.

It is left as an exercise for the reader to determine the sizes of both the NMOS and the PMOS transistor.
2.4 Packaging Integrated Circuits

The IC package plays a fundamental role in the operation and performance of a component. Besides providing a means of bringing signal and supply wires in and out of the silicon die, it also removes the heat generated by the circuit and provides mechanical support. Finally, it also protects the die against environmental conditions such as humidity.

The packaging technology furthermore has a major impact on the performance and power-dissipation of a microprocessor or signal processor. This influence is getting more pronounced as time progresses by the reduction in internal signal delays and on-chip capacitance as a result of technology scaling. Up to 50% of the delay of a high-performance computer is currently due to packaging delays, and this number is expected to rise. The search for higher-performance packages with fewer inductive or capacitive parasitics has accelerated in recent years.
The increasing complexity of what can be integrated on a single die also translates into a need for ever more input-output pins, as the number of connections going off-chip tends to be roughly proportional to the complexity of the circuitry on the chip. This relationship was first observed by E. Rent of IBM (published in [Landman71]), who translated it into an empirical formula that is appropriately called Rent’s rule. This formula relates the number of input/output pins to the complexity of the circuit, as measured by the number of gates.

\[ P = K \times G^\beta \]  \hspace{1cm} (2.1)

where \( K \) is the average number of I/Os per gate, \( G \) the number of gates, \( \beta \) the Rent exponent, and \( P \) the number of I/O pins to the chip. \( \beta \) varies between 0.1 and 0.7. Its value depends strongly upon the application area, architecture, and organization of the circuit, as demonstrated in Table 2.1. Clearly, microprocessors display a very different input/output behavior compared to memories.

<table>
<thead>
<tr>
<th>Application</th>
<th>( \beta )</th>
<th>( K )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static memory</td>
<td>0.12</td>
<td>6</td>
</tr>
<tr>
<td>Microprocessor</td>
<td>0.45</td>
<td>0.82</td>
</tr>
<tr>
<td>Gate array</td>
<td>0.5</td>
<td>1.9</td>
</tr>
<tr>
<td>High-speed computer (chip)</td>
<td>0.63</td>
<td>1.4</td>
</tr>
<tr>
<td>High-speed computer (board)</td>
<td>0.25</td>
<td>82</td>
</tr>
</tbody>
</table>

The observed rate of pin-count increase for integrated circuits varies between 8% to 11% per year, and it has been projected that packages with more than 2000 pins will be required by the year 2010. For all these reasons, traditional dual-in-line, through-hole mounted packages have been replaced by other approaches such as surface-mount, ball-grid array, and multichip module techniques. It is useful for the circuit designer to be aware of the available options, and their pros and cons.

Due to its multifunctionality, a good package must comply with a large variety of requirements.

- **Electrical requirements**—Pins should exhibit low capacitance (both interwire and to the substrate), resistance, and inductance. A large characteristic impedance should be tuned to optimize transmission line behavior. Observe that intrinsic integrated-circuit impedances are high.

- **Mechanical and thermal properties**—The heat-removal rate should be as high as possible. Mechanical reliability requires a good matching between the thermal properties of the die and the chip carrier. Long-term reliability requires a strong connection from die to package as well as from package to board.

- **Low Cost**—Cost is always one of the more important properties. While ceramics have a superior performance over plastic packages, they are also substantially more expensive. Increasing the heat removal capacity of a package also tends to raise the
50

THE MANUFACTURING PROCESS Chapter 2

package cost. For instance, chips dissipating over 50 W require special heat sink attachments. Even more extreme techniques such as fans and blowers, liquid cooling hardware, or heat pipes, are needed for higher dissipation levels.

Packing density is a major factor in reducing board cost. The increasing pin count either requires an increase in the package size or a reduction in the pitch between the pins. Both have a profound effect on the packaging economics.

Packages can be classified in many different ways — by their main material, the number of interconnection levels, and the means used to remove heat. In this short section, we can only glance briefly at each of those issues.

2.4.1 Package Materials

The most common materials used for the package body are ceramic and polymers (plastics). The latter have the advantage of being substantially cheaper, but suffer from inferior thermal properties. For instance, the ceramic Al$_2$O$_3$ (Alumina) conducts heat better than SiO$_2$ and the Polyimide plastic, by factors of 30 and 100 respectively. Furthermore, its thermal expansion coefficient is substantially closer to the typical interconnect metals. The disadvantage of alumina and other ceramics is their high dielectric constant, which results in large interconnect capacitances.

2.4.2 Interconnect Levels

The traditional packaging approach uses a two-level interconnection strategy. The die is first attached to an individual chip carrier or substrate. The package body contains an internal cavity where the chip is mounted. These cavities provide ample room for many connections to the chip leads (or pins). The leads compose the second interconnect level and connect the chip to the global interconnect medium, which is normally a PC board. Complex systems contain even more interconnect levels, since boards are connected together using backplanes or ribbon cables. The first two layers of the interconnect hierarchy are illustrated in the drawing of Figure 2.10.

![Figure 2.10 Interconnect hierarchy in traditional IC packaging.](image)

This deep hierarchy of interconnect levels is becoming unacceptable in today’s complex designs with their higher levels of integration, large signal counts, and increased performance requirements. The trend is toward reducing the number of levels. In the
future, improved manufacturing, design, and testing capabilities will make it possible to integrate a complex computer system with all its peripherals on a single piece of semiconductor. For the time being, attention is focused on the elimination of the first level in the packaging hierarchy. Instead of housing dies in individual packages, they are mounted directly on the interconnect medium or board. This packaging approach is called the multichip module technique (or MCM), and results in a substantial increase in packing density as well as improved performance. The following sections provide a brief overview of the interconnect techniques used at levels one and two of the interconnect hierarchy, followed by a short discussion of the MCM technology.

**Interconnect Level 1 — Die-to-Package-Substrate**

For a long time, wire bonding was the technique of choice to provide an electrical connection between die and package. In this approach, the backside of the die is attached to the substrate using glue with a good thermal conductance. Next, the chip pads are individually connected to the lead frame with aluminum or gold wires. The wire-bonding machine used for this purpose operates much like a sewing machine. An example of wire bonding is shown in Figure 2.11. Although the wire-bonding process is automated to a large degree, it has some major disadvantages.

1. Wires must be attached serially, one after the other. This leads to longer manufacturing times with increasing pin counts.
2. Larger pin counts make it substantially more challenging to find bonding patterns that avoid shorts between the wires.
3. Bonding wires have inferior electrical properties, such as a high individual inductance (5 nH or more) and mutual inductance with neighboring signals.
4. The exact value of the parasitics is hard to predict because of the manufacturing approach and irregular layout.

New attachment techniques are being explored as a result of these deficiencies. In one approach, called Tape Automated Bonding (or TAB), the die is attached to a metal lead frame that is printed on a polymer film (typically polyimide) (Figure 2.12a). The connection between chip pads and polymer film wires is made using solder bumps (Figure 2.12b). The tape can then be connected to the package body using a number of techniques. One possible approach is to use pressure connectors.
The advantage of the TAB process is that it is highly automated. The sprockets in the film are used for automatic transport. All connections are made simultaneously. The printed approach helps to reduce the wiring pitch, which results in higher lead counts. Elimination of the long bonding wires improves the electrical performance. For instance, for a two-conductor layer, 48 mm TAB Circuit, the following electrical parameters hold: $L \approx 0.3–0.5$ nH, $C \approx 0.2–0.3$ pF, and $R \approx 50–200$ Ω [Doane93, p. 420].

Another approach is to flip the die upside-down and attach it directly to the substrate using solder bumps. This technique, called flip-chip mounting, has the advantage of a superior electrical performance (Figure 2.13). Instead of making all the I/O connections on the die boundary, pads can be placed at any position on the chip. This can help address the power- and clock-distribution problems, since the interconnect materials on the substrate (e.g., Cu or Au) are typically of a better quality than the Al on the chip.

**Figure 2.12** Tape-automated bonding (TAB).

**Figure 2.13** Flip-chip bonding.

**Interconnect Level 2—Package Substrate to Board**

When connecting the package to the PC board, through-hole mounting has been the packaging style of choice. A PC board is manufactured by stacking layers of copper and insulating epoxy glass. In the through-hole mounting approach, holes are drilled through the board and plated with copper. The package pins are inserted and electrical connection is made with solder (Figure 2.14a). The favored package in this class was the dual-in-line package or DIP (Figure 2.15a). The packaging density of the DIP degrades rapidly when the number of pins exceeds 64. This problem can be alleviated by using the pin-grid-array
Section 2.4 Packaging Integrated Circuits

A PGA (Package Grid Array) package that has leads on the entire bottom surface instead of only on the periphery (Figure 2.15b). PGAs can extend to large pin counts (over 400 pins are possible).

The through-hole mounting approach offers a mechanically reliable and sturdy connection. However, this comes at the expense of packaging density. For mechanical reasons, a minimum pitch of 2.54 mm between the through-holes is required. Even under those circumstances, PGAs with large numbers of pins tend to substantially weaken the board. In addition, through-holes limit the board packing density by blocking lines that might otherwise have been routed below them, which results in longer interconnections. PGAs with large pin counts hence require extra routing layers to connect to the multitudes of pins. Finally, while the parasitic capacitance and inductance of the PGA are slightly lower than that of the DIP, their values are still substantial (Table 2.2).

Many of the shortcomings of the through-hole mounting are solved by using the surface-mount technique. A chip is attached to the surface of the board with a solder connection without requiring any through-holes (Figure 2.14b). Packing density is increased for the following reasons: (1) through-holes are eliminated, which provides more wiring space; (2) the lead pitch is reduced; and (3) chips can be mounted on both sides of the board. In addition, the elimination of the through-holes improves the mechanical strength of the board. On the negative side, the on-the-surface connection makes the chip-board connection weaker. Not only is it cumbersome to mount a component on a board, but also more expensive equipment is needed, since a simple soldering iron will not do anymore.
Finally, testing of the board is more complex, because the package pins are no longer accessible at the backside of the board. Signal probing becomes hard or even impossible. A variety of surface-mount packages are currently in use with different pitch and pin-count parameters. Three of these packages are shown in Figure 2.15: the small-outline package with gull wings, the plastic leaded package (PLCC) with J-shaped leads, and the leadless chip carrier. An overview of the most important parameters for a number of packages is given in Table 2.2.

Table 2.2 Parameters of various types of chip carriers.

<table>
<thead>
<tr>
<th>Package type</th>
<th>Lead spacing (Typical)</th>
<th>Lead count (Maximum)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual-in-line</td>
<td>2.54 mm</td>
<td>64</td>
</tr>
<tr>
<td>Pin grid array</td>
<td>2.54 mm</td>
<td>&gt; 300</td>
</tr>
<tr>
<td>Small-outline IC</td>
<td>1.27 mm</td>
<td>28</td>
</tr>
<tr>
<td>Leaded chip carrier (PLCC)</td>
<td>1.27 mm</td>
<td>124</td>
</tr>
<tr>
<td>Leadless chip carrier</td>
<td>0.75 mm</td>
<td>124</td>
</tr>
</tbody>
</table>

Even surface-mount packaging is unable to satisfy the quest for evermore higher pin-counts. When more than 300 I/O connections are needed, solder balls replace pins as the preferred interconnect medium between package and board. An example of such a packaging approach, called ceramic ball grid array (BGA), is shown in . Solder bumps are used to connect both the die to the package substrate, and the package to the board. The area array interconnect of the BGA provides constant input/output density regardless of the number of total package I/O pins. A minimum pitch between solder balls of as low as 0.8 mm can be obtained, and packages with multiple 1000’s of I/O signals are feasible.

Figure 2.16 Ball grid array packaging; (a) cross-section, (b) photo of package bottom

Multi-Chip Modules—Die-to-Board

Eliminating one layer in the packaging hierarchy by mounting the die directly on the wiring backplanes—board or substrate—offers a substantial benefit when performance or density is a major issue. A number of the previously mentioned die-mounting techniques
Section 2.4 Packaging Integrated Circuits

can be adapted to mount dies directly on the substrate. This includes wire bonding, TAB, and flip-chip, although the latter two are preferable. The substrate itself can vary over a wide range of materials, depending upon the required mechanical, electrical, thermal, and economical requirements. Materials of choice are epoxy substrates (similar to PC boards), metal, ceramics, and silicon. Silicon has the advantage of presenting a perfect match in mechanical and thermal properties with respect to the die material.

The main advantages of the MCM approach are the increased packaging density and performance. An example of an MCM module implemented using a silicon substrate (commonly dubbed silicon-on-silicon) is shown in Figure 2.17. The module, which implements an avionics processor module and is fabricated by Rockwell International, contains 53 ICs and 40 discrete devices on a 2.2” × 2.2” substrate with aluminum polyimide interconnect. The interconnect wires are only an order of magnitude wider than what is typical for on-chip wires, since similar patterning approaches are used. The module itself has 180 I/O pins. Performance is improved by the elimination of the chip-carrier layer with its assorted parasitics, and through a reduction of the global wiring lengths on the die, a result of the increased packaging density. For instance, a solder bump has an assorted capacitance and inductance of only 0.1 pF and 0.01 nH respectively. The MCM technology can also reduce power consumption significantly, since large output drivers—and associated dissipation—become superfluous due to the reduced load capacitance of the output pads. The dynamic power associated with the switching of the large load capacitances is simultaneously reduced.

While MCM technology offers some clear benefits, its main disadvantage is economic. This technology requires some advanced manufacturing steps that make the process expensive. The approach is only justifiable when either dense housing or extreme performance is essential. In the near future, this argument might become obsolete as

![Figure 2.17 Avionics processor module. Courtesy of Rockwell International.](image-url)
MCM approaches proliferate; for example, some of the more advanced microprocessors, such as the Intel P6 (Pentium Pro), employ MCM technology.

### 2.4.3 Thermal Considerations in Packaging

As the power consumption of integrated circuits rises, it becomes increasingly important to efficiently remove the heat generated by the chips. A large number of failure mechanisms in ICs are accentuated by increased temperatures. Examples are leakage in reverse-biased diodes, electromigration, and hot-electron trapping. To prevent failure, the temperature of the die must be kept within certain ranges. The supported temperature range for commercial devices during operation equals 0° to 70°C. Military parts are more demanding and require a temperature range varying from –55° to 125°C.

The cooling effectiveness of a package depends upon the thermal conduction of the package material, which consists of the package substrate and body, the package composition, and the effectiveness of the heat transfer between package and cooling medium. Standard packaging approaches use still or circulating air as the cooling medium. The transfer efficiency can be improved by adding finned metal heat sinks to the package. More expensive packaging approaches, such as those used in mainframes or supercomputers, force air, liquids, or inert gases through tiny ducts in the package to achieve even greater cooling efficiencies.

As an example, a 40-pin DIP has a thermal resistance of 38 °C/W and 25 °C/W for natural and forced convection of air. This means that a DIP can dissipate 2 watts (3 watts) of power with natural (forced) air convection, and still keep the temperature difference between the die and the environment below 75°C. For comparison, the thermal resistance of a ceramic PGA ranges from 15 ° to 30 °C/W.

Since packaging approaches with decreased thermal resistance are prohibitively expensive, keeping the power dissipation of an integrated circuit within bounds is an economic necessity. The increasing integration levels and circuit performance make this task nontrivial. An interesting relationship in this context has been derived by Nagata [Nagata92]. It provides a bound on the integration complexity and performance as a function of the thermal parameters

\[ \frac{N_G}{t_p} \leq \frac{\Delta T}{\theta E} \]  

where \( N_G \) is the number of gates on the chip, \( t_p \) the propagation delay, \( \Delta T \) the maximum temperature difference between chip and environment, \( \theta \) the thermal resistance between them, and \( E \) the switching energy of each gate.

**Example 2.2 Thermal Bounds On Integration**

For \( \Delta T = 100 \, ^\circ\mathrm{C} \), \( \theta = 2.5 \, ^\circ\mathrm{C/W} \) and \( E = 0.1 \, \text{pJ} \), this results in \( N_G/t_p \leq 4 \times 10^5 \) (gates/nsec). In other words, the maximum number of gates on a chip, when all gates are operating simultaneously, must be less than 400,000 if the switching speed of each gate is 1 nsec. This is equivalent to a power dissipation of 40 W.
Fortunately, not all gates are operating simultaneously in real systems. The maximum number of gates can be substantially larger, based on the activity in the circuit. For instance, it was experimentally derived that the ratio between the average switching period and the propagation delay ranges from 20 to 200 in mini- and large-scale computers [Masaki92].

Nevertheless, Eq. (2.2) demonstrates that heat dissipation and thermal concerns present an important limitation on circuit integration. Design approaches for low power that reduce either $E$ or the activity factor are rapidly gaining importance.

2.5 Perspective — Trends in Process Technology

Modern CMOS processes pretty much track the flow described in the previous sections although a number of the steps might be reversed, a single well approach might be followed, a grown field oxide instead of the trench approach might be used, or extra steps such as LDD (Lightly Doped Drain) might be introduced. Also, it is quite common to cover the polysilicon interconnections as well as the drain and source regions with a silicide such as TiSi$_2$ to improve the conductivity (see Figure 2.2). This extra operation is inserted between steps $i$ and $j$ of our process. Some important modifications or improvements to the technology are currently under way or are on the horizon, and deserve some attention. Beyond these, it is our belief that no dramatic changes, breaking away from the described CMOS technology, must be expected in the next decade.

2.5.1 Short-Term Developments

Copper and Low-$k$ Dielectrics

A recurring theme in this text book will be the increasing impact of interconnect on the overall design performance. Process engineers are continuously evaluating alternative options for the traditional ‘Aluminum conductor—SiO$_2$ insulator’ combination that has been the norm for the last decades. In 1998, engineers at IBM introduced an approach that finally made the use of Copper as an interconnect material in a CMOS process viable and economical [IEEE Spectrum98]. Copper has the advantage of having a resistivity that is substantially lower than Aluminum. Yet it has the disadvantage of easy diffusion into silicon, which degrades the characteristics of the devices. Coating the copper with a buffer material such as Titanium Nitride, preventing the diffusion, addresses this problem, but requires a special deposition process. The Dual Damascene process, introduced by IBM, (Figure 2.18) uses a metallization approach that fills trenches etched into the insulator, followed by a chemical-mechanical polishing step. This is in contrast with the the traditional approach that first deposits a full metal layer, and removes the redundant material through etching.

In addition to the lower resistivity interconnections, insulator materials with a lower dielectric constant than SiO$_2$ — and hence lower capacitance — have also found their way into the production process starting with the 0.18 µm CMOS process generation.
Silicon-on-Insulator

While having been around for quite a long time, there seems to be a good chance that Silicon-on-Insulator (SOI) CMOS might replace the traditional CMOS process, described in the previous sections (also known as the *bulk CMOS process*). The main difference lies in the start material: the transistors are constructed in a very thin layer of silicon, deposited on top of a thick layer of insulating SiO$_2$ (Figure 2.19). The primary advantages of the SOI process are reduced parasitics and better on-off characteristics. It has, for instance, been demonstrated by researchers at IBM that the simple porting of a design from a bulk CMOS to an SOI process —leaving all other design and process parameters such as channel length and oxide thickness identical— yields a performance improvement of 22% [Allen99]. Preparing a high quality SOI substrate at an economical cost was long the main hindrance against a large-scale introduction of the process. This picture has changed at the end of the nineties, and SOI is steadily moving into the mainstream.

2.5.2 In the Longer Term

Extending the life of CMOS technology beyond the next decade, and deeply below the 100 nm channel length region however will require re-engineering of both the process technology and the device structure. While projecting what approaches will dominate in that era equals resorting to crystal-ball gazing, some interesting developments are worth mentioning.
Vertical Transistors

Even while the addition of many metal layers has turned the integrated circuit into a truly three-dimensional artifact, the transistor itself is still mostly laid out in a horizontal plane. This forces the device designer to jointly optimize packing density and performance parameters. By rotating the device so that the drain ends up on top, and the source at the bottom, these concerns are separated: packing density still is dominated by horizontal dimensions, while performance issues are mostly determined by vertical spacings (Figure 2.20). Operational devices of this type have been fabricated with channel lengths substantially below 0.1 \( \mu \text{m} \). [LucentRef].

Truly Three-Dimensional Integrated Circuits

Getting signals in and out of the computation elements in a timely fashion is one of the main challenges presented by the continued increase in integration density. One way to address this problem is to introduce extra active layers, and to sandwich them in-between the metal interconnect layers (Figure 2.21). This enables us to position high density memory on top of the logic processors implemented in the bulk CMOS, reducing the distance between computation and storage, and hence also the delay [Saraswat00]. In addition,
devices with different voltage, performance, or substrate material requirements can be placed in different layers. For instance, the top active layer can be reserved for the realization of optical transceivers, which may help to address the input/output and the long distance interconnect problems of today's IC's.

While this approach may seem promising, a number of major challenges and hindrances have to be resolved to make it really viable. How to remove the dissipated heat is one of the compelling questions. Ensuring yield is another one. Yet, researchers are demonstrating major progress, and 3D integration might very well be on the horizon. Before the true solution arrives, we might have to rely on some intermediate approaches. One alternative, called 2.5D integration, is to bond two fully processed wafers, on which circuits are fabricated on the surface such that the chips completely overlap. Vias are etched to electrically connect both chips after metallization. The advantages of this technology lie in the similar electrical properties of devices on all active levels and the independence of processing temperature since all chips can be fabricated separately and later bonded. The major limitation of this technique is its lack of precision (best case alignment +/− 2 µm), which restricts the inter-chip communication to global metal lines.

One picture that strongly emerges from these futuristic devices is that the line between chip, substrate, package, and board is blurring, and that designers of these systems-on-a-die will have to consider all these aspects simultaneously.

2.6 Summary

This chapter has presented an a birds-eye view on issues regarding the manufacturing and packaging of CMOS integrated circuits.

- The manufacturing process of integrated circuits require a large number of steps, each of which consists of a sequence of basic operations. A number of these steps and/or operations, such as photolithographical exposure and development, material deposition, and etching, are executed very repetitively in the course of the manufacturing process.
Section 2.7 To Probe Further

- The *optical masks* forms the central interface between the intrinsics of the manufacturing process and the design that the user wants to see transferred to the silicon fabric.

- The *design rules set* define the constraints in terms of minimum width and separation that the IC design has to adhere to if the resulting circuit is to be fully functional. This design rules acts as the contract between the circuit designer and the process engineer.

- The *package* forms the interface between the circuit implemented on the silicon die and the outside world, and as such has a major impact on the performance, reliability, longevity, and cost of the integrated circuit.

2.7 To Probe Further

Many textbooks on semiconductor manufacturing have been published over the last few decades. An excellent overview of the state-of-the-art in CMOS manufacturing can be found in the “Silicon VLSI Technology” book by J. Plummer, M. Deal, and P. Griffin [Plummer2000]. Other sources for information are the IEEE Transactions on Electron Devices, and the Technical Digest of the IEDM conference.

REFERENCES
The increasing complexity of the integrated circuit has made the role of design-automation tools indispensable, and raises the abstractions the designer is working with to ever higher levels. Yet, when performance or design density is of primary importance, the designer has no other choice than to return to handcrafting the circuit topology and physical design. The labor-intensive nature of this approach, called custom design, translates into a high cost and a long time-to-market. Therefore, it can only be justified economically under the following conditions:

- The custom block can be reused many times, for instance as a library cell
- The cost can be amortized over a large volume. Microprocessors and semiconductor memories are examples of applications in this class.
- Cost is not the prime design criterion. This is becoming increasingly rare. Examples are space-applications and scientific instrumentation.

With continuous progress in the design-automation arena, the share of custom design reduces from year to year. Even in high-performance microprocessors, large portions are designed automatically using semicustom design approaches. Only the most performance-critical modules such as the integer and floating-point execution units are handcrafted.
Even though the amount of design automation in the custom design process is minimal, some design tools have proven to be indispensable. Together with circuit simulators, these programs form the core of every design-automation environment, and are the first tools an aspirant circuit designer will encounter.

**Layout Editor**

The layout editor is the premier working tool of the designer and exists primarily for the generation of a physical representation of a design, given a circuit topology. Virtually every design-automation vendor offers an entry in this field. Most well-known is the MAGIC tool developed at the University of California at Berkeley [Ousterhout84], which has been widely distributed. Even though MAGIC did not withstand the evolution of software technology and user interface, some of its offspring did. Throughout this textbook, we will be using a layout tool called max, a MAGIC descendant developed by a company called MicroMagic [mmi00]. A typical max display is shown in Figure 3.1 and illustrates the basic function of the layout editor—placing polygons on different mask layers so that a functional physical design is obtained (scathingly called *polygon pushing*).

![Figure 3.1 View of a max display window. It plots the layout of two stacked NMOS transistor. The menu on the left side allows for the selection of the layer a particular polygon will be placed on.](image)

Since physical design occupies a major fraction of the design time for a new cell or component, techniques to expedite this process have been in continual demand. The *symbolic-layout* approach has gained popularity over the years. In this design methodology, the designer only draws a shorthand notation for the layout structure. This notation indicates only the *relative* positioning of the various design components (transistors, contacts,
wires). The *absolute* coordinates of these elements are determined automatically by the editor using a *compactor* [Hsueh79, Weste93]. The compactor translates the design rules into a set of constraints on the component positions, and solves a constrained optimization problem that attempts to minimize the area or another cost function.

An example of a symbolic notation for a circuit topology, called a *sticks diagram*, is shown in Figure 3.2. The different layout entities are dimensionless, since only positioning is important. The advantage of this approach is that the designer does not have to worry about design rules, because the compactor ensures that the final layout is physically correct. Thus, she can avoid cumbersome polygon manipulations. Another plus of the symbolic approach is that cells can adjust themselves automatically to the environment. For example, automatic pitch-matching of cells is an attractive feature in module generators. Consider the case of Figure 3.3 (from [Croes88]), in which the original cells have different heights, and the terminal positions do not match. Connecting the cells would require extra wiring. The symbolic approach allows the cells to adjust themselves and connect without any overhead.

The disadvantage of the symbolic approach is that the outcome of the compaction phase is often unpredictable. The resulting layout can be less dense than what is obtained with the manual approach. Notwithstanding, symbolic layout tools have improved considerably over the years and are currently a part of the mainstream design process.
Design-Rule Checking

Design rules were introduced in Chapter 2 as a set of layout restrictions that ensure the manufactured design will operate as desired with no short or open circuits. A prime requirement of the physical layout of a design is that it adhere to these rules. This can be verified with the aid of a design-rule checker (DRC), which uses as inputs the physical layout of a design and a description of the design rules presented in the form of a technology file. Since a complex circuit can contain millions of polygons that must be checked against each other, efficiency is the most important property of a good DRC tool. The verification of a large chip can take hours or days of computation time. One way of expediting the process is to preserve the design hierarchy at the physical level. For instance, if a cell is used multiple times in a design, it should be checked only once. Besides speeding up the process, the use of hierarchy can make error messages more informative by retaining knowledge of the circuit structure.

DRC tools come in two formats: (1) The on-line DRC runs concurrent with the layout editor and flags design violations during the cell layout. For instance, max has a built-in design-rule checking facility. An example of on-line DRC is shown in Figure 3.4. (2) Batch DRC is used as a post-design verifier, and is run on a complete chip prior to shipping the mask descriptions to the manufacturer.

Circuit Extraction

Another important tool in the custom-design methodology is the circuit extractor, which derives a circuit schematic from a physical layout. By scanning the various layers and their interactions, the extractor reconstructs the transistor network, including the sizes of the devices and the interconnections. The schematic produced can be used to verify that the artwork implements the intended function. Furthermore, the resulting circuit diagram contains precise information on the parasitics, such as the diffusion and wiring capacitances and resistances. This allows for a more accurate simulation and analysis. The complexity of the extraction depends greatly upon the desired information. Most extractors extract the transistor network and the capacitances of the interconnect with respect to GND or other network nodes. Extraction of the wiring resistances already comes at a
greater cost, yet has become a necessity for virtually all high-performance circuits. Clever algorithms have helped to reduce the complexity of the resulting circuit diagrams. For very high speed circuits, extraction of the inductance would be desirable as well. Unfortunately, this requires a three-dimensional analysis and is only feasible for small-sized circuits at present.

To Probe Further

More detailed information regarding the MAGIC and max layout editors can be found on the web-site of this book. In-depth textbooks on layout generation and verification have been published, and can be of great help to the novice designer. Just to mention a number of them, [Clein00], [Uyemura95], [Wolf94] offer some comprehensive and well-illustrated treatment and discussion.

REFERENCES


Exercises

1.
4.1 Introduction
4.2 A First Glance
4.3 Interconnect Parameters — Capacitance, Resistance, and Inductance
  4.3.1 Capacitance
  4.3.2 Resistance
  4.3.3 Inductance
4.4 Electrical Wire Models
  4.4.1 The Ideal Wire
  4.4.2 The Lumped Model
  4.4.3 The Lumped RC model
  4.4.4 The Distributed rc Line
4.4.5 The Transmission Line
4.5 SPICE Wire Models
  4.5.1 Distributed rc Lines in SPICE
  4.5.2 Transmission Line Models in SPICE
4.6 Perspective: A Look into the Future
4.1 Introduction

Throughout most of the past history of integrated circuits, on-chip interconnect wires were considered to be second class citizens that had only to be considered in special cases or when performing high-precision analysis. With the introduction of deep-submicron semiconductor technologies, this picture is undergoing rapid changes. The parasitics effects introduced by the wires display a scaling behavior that differs from the active devices such as transistors, and tend to gain in importance as device dimensions are reduced and circuit speed is increased. In fact, they start to dominate some of the relevant metrics of digital integrated circuits such as speed, energy-consumption, and reliability. This situation is aggravated by the fact that improvements in technology make the production of ever-larger die sizes economically feasible, which results in an increase in the average length of an interconnect wire and in the associated parasitic effects. A careful and in-depth analysis of the role and the behavior of the interconnect wire in a semiconductor technology is therefore not only desirable, but even essential.

4.2 A First Glance

The designer of an electronic circuit has multiple choices in realizing the interconnections between the various devices that make up the circuit. State-of-the-art processes offer multiple layers of Aluminum, and at least one layer of polysilicon. Even the heavily doped n⁺ or p⁺ layers, typically used for the realization of source and drain regions, can be employed for wiring purposes. These wires appear in the schematic diagrams of electronic circuits as simple lines with no apparent impact on the circuit performance. In our discussion on the integrated-circuit manufacturing process, it became clear that this picture is overly simplistic, and that the wiring of today’s integrated circuits forms a complex geometry that introduces capacitive, resistive, and inductive parasitics. All three have multiple effects on the circuit behavior.

1. An increase in propagation delay, or, equivalently, a drop in performance.
2. An impact on the energy dissipation and the power distribution.
3. An introduction of extra noise sources, which affects the reliability of the circuit.

A designer can decide to play it safe and include all these parasitic effects in her analysis and design optimization process. This conservative approach is non-constructive and even unfeasible. First of all, a “complete” model is dauntingly complex and is only applicable to very small topologies. It is hence totally useless for today’s integrated circuits with their millions of circuit nodes. Furthermore, this approach has the disadvantage that the “forest gets lost between the trees”. The circuit behavior at a given circuit node is only determined by a few dominant parameters. Bringing all possible effects to bear, only obscures the picture and turns the optimization and design process a “trial-and-error” operation rather than an enlightened and focused search.

To achieve the latter, it is important that the designer has a clear insight in the parasitic wiring effects, their relative importance, and their models. This is best illustrated with the simple example, shown in Figure 4.1. Each wire in a bus network connects a transmit-
Section 4.2 A First Glance

A full-fledged circuit model, taking into account the parasitic capacitance, resistance, and the inductance of the interconnections, is shown in Figure 4.2a. Observe that these extra circuit elements are not located in a single physical point, but are distributed over the length of the wire. This is a necessity when the length of the wire becomes significantly larger than its width. Notice also that some parasitics are inter-wire, hence creating coupling effects between the different bus-signals that were not present in the original schematics.

Analyzing the behavior of this schematic, which only models a small part of the circuit, is slow and cumbersome. Fortunately, substantial simplifications can often be made, some of which are enumerated below.

- Inductive effects can be ignored if the resistance of the wire is substantial — this is for instance the case for long Aluminum wires with a small cross-section — or if the rise and fall times of the applied signals are slow.

- When the wires are short, the cross-section of the wire is large, or the interconnect material used has a low resistivity, a capacitance-only model can be used (Figure 4.2b).

- Finally, when the separation between neighboring wires is large, or when the wires only run together for a short distance, inter-wire capacitance can be ignored, and all the parasitic capacitance can be modeled as capacitance to ground.

Obviously, the latter problems are the easiest to model, analyze, and optimize. The experienced designer knows to differentiate between dominant and secondary effects. The goal of this chapter is to present the reader the basic techniques to estimate the values of
the various interconnect parameters, simple models to evaluate their impact, and a set of rules-of-thumb to decide when and where a particular model or effect should be considered.

4.3 Interconnect Parameters — Capacitance, Resistance, and Inductance

4.3.1 Capacitance

An accurate modeling of the wire capacitance(s) in a state-of-the-art integrated circuit is a non-trivial task and is even today the subject of advanced research. The task is complicated by the fact that the interconnect structure of contemporary integrated circuits is three-dimensional, as was clearly demonstrated in the process cross-section of FIGURE (CHAPTER 2). The capacitance of such a wire is a function of its shape, its environment, its distance to the substrate, and the distance to surrounding wires. Rather than getting lost in complex equations and models, a designer typically will use an advanced extraction tool to get precise values of the interconnect capacitances of a completed layout. Most semiconductor manufacturers also provide empirical data for the various capacitance contributions, as measured from a number of test dies. Yet, some simple first-order models come in handy to provide a basic understanding of the nature of interconnect capacitance and its parameters, and of how wire capacitance will evolve with future technologies.

Consider first a simple rectangular wire placed above the semiconductor substrate, as shown in Figure 4.3. If the width of the wire is substantially larger than the thickness of the insulating material, it may be assumed that the electrical-field lines are orthogonal to the capacitor plates, and that its capacitance can be modeled by the parallel-plate capaci-
tor model (also called area capacitance). Under those circumstances, the total capacitance of the wire can be approximated as

\[ C_{\text{tot}} = \frac{\varepsilon_{di} WL}{t_{di}} \]  

(4.1)

where \( W \) and \( L \) are respectively the width and length of the wire, and \( t_{di} \) and \( \varepsilon_{di} \) represent the thickness of the dielectric layer and its permittivity. \( \text{SiO}_2 \) is the dielectric material of choice in integrated circuits, although some materials with lower permittivity, and hence lower capacitance, are coming in use. Examples of the latter are organic polyimides and aerogels. \( \varepsilon \) is typically expressed as the product of two terms, or \( \varepsilon = \varepsilon_r \varepsilon_0 \). \( \varepsilon_0 = 8.854 \times 10^{-12} \) F/m is the permittivity of free space, and \( \varepsilon_r \) the relative permittivity of the insulating material. Table 4.1 presents the relative permittivity of several dielectrics used in integrated circuits. In summary, the important message from Eq. (4.1) is that the capacitance is proportional to the overlap between the conductors and inversely proportional to their separation.

Table 4.1 Relative permittivity of some typical dielectric materials.

<table>
<thead>
<tr>
<th>Material</th>
<th>( \varepsilon_r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free space</td>
<td>1</td>
</tr>
<tr>
<td>Aerogels</td>
<td>-1.5</td>
</tr>
<tr>
<td>Polyimides (organic)</td>
<td>3-4</td>
</tr>
<tr>
<td>Silicon dioxide</td>
<td>3.9</td>
</tr>
<tr>
<td>Glass-epoxy (PC board)</td>
<td>5</td>
</tr>
<tr>
<td>Silicon Nitride (Si(_3)N(_4))</td>
<td>7.5</td>
</tr>
<tr>
<td>Alumina (package)</td>
<td>9.5</td>
</tr>
<tr>
<td>Silicon</td>
<td>11.7</td>
</tr>
</tbody>
</table>

\[ ^1 \] To differentiate between distributed (per unit length) wire parameters versus total lumped values, we will use lowercase to denote the former and uppercase for the latter.
In actuality, this model is too simplistic. To minimize the resistance of the wires while scaling technology, it is desirable to keep the cross-section of the wire \((W \times H)\) as large as possible — as will become apparent in a later section. On the other hand, small values of \(W\) lead to denser wiring and less area overhead. As a result, we have over the years witnessed a steady reduction in the \(W/H\)-ratio, such that it has even dropped below unity in advanced processes. This is clearly visible on the process cross-section of FIGURE. Under those circumstances, the parallel-plate model assumed above becomes inaccurate. The capacitance between the side-walls of the wires and the substrate, called the \textit{fringing capacitance}, can no longer be ignored and contributes to the overall capacitance. This effect is illustrated in Figure 4.4a. Presenting an exact model for this difficult geometry is hard. So, as good engineering practice dictates, we will use a simplified model that approximates the capacitance as the sum of two components (Figure 4.4b): a parallel-plate capacitance determined by the orthogonal field between a wire of width \(w\) and the ground plane, in parallel with the fringing capacitance modeled by a cylindrical wire with a dimension equal to the interconnect thickness \(H\). The resulting approximation is simple and works fairly well in practice.

\[
c_{\text{wire}} = c_{\text{pp}} + c_{\text{fringe}} = \frac{w e_d}{t_d} + \frac{2 \pi e_d}{\log(t_d/H)}
\]

with \(w = W - H/2\) a good approximation for the width of the parallel-plate capacitor. Numerous more accurate models (e.g. [Vdm84]) have been developed over time, but these tend to be substantially more complex, and defeat our goal of developing a conceptual understanding.

To illustrate the importance of the fringing-field component, Figure 4.5 plots the value of the wiring capacitance as a function of \((W/H)\). For larger values of \((W/H)\) the total capacitance approaches the parallel-plate model. For \((W/H)\) smaller than 1.5, the fringing
component actually becomes the dominant component. The fringing capacitance can increase the overall capacitance by a factor of more than 10 for small line widths. It is interesting to observe that the total capacitance levels off to a constant value of approximately 1 pF/cm for line widths smaller than the insulator thickness. In other words, the capacitance is no longer a function of the width.

Figure 4.5: Capacitance of interconnect wire as a function of \( \frac{W}{H} \), including fringing-field effects (from [Schaper83]). Two values of \( T/H \) are considered.

So far, we have restricted our analysis to the case of a single rectangular conductor placed over a ground plane. This structure, called a microstripline, used to be a good model for semiconductor interconnections when the number of interconnect layers was restricted to 1 or 2. Today's processes offer many more layers of interconnect, which are packed quite densely in addition. In this scenario, the assumption that a wire is completely isolated from its surrounding structures and is only capacitively coupled to ground, becomes untenable. This is illustrated in Figure 4.6, where the capacitance components of a wire embedded in an interconnect hierarchy are identified. Each wire is not only coupled to the grounded substrate, but also to the neighboring wires on the same layer and on adjacent layers. To a first order, this does not change the total capacitance connected to a given wire. The main difference is that not all its capacitive components do terminate at the grounded substrate, but that a large number of them connect to other wires, which have dynamically varying voltage levels. We will later see that these floating capacitors form not only a source of noise (crosstalk), but also can have a negative impact on the performance of the circuit.

In summary, interwire capacitances become a dominant factor in multi-layer interconnect structures. This effect is more outspoken for wires in the higher interconnect layers, as these wires are farther away from the substrate. The increasing contribution of the interwire capacitance to the total capacitance with decreasing feature sizes is best illustrated by Figure 4.7. In this graph, which plots the capacitive components of a set of parallel wires routed above a ground plane, it is assumed that dielectric and wire thickness are...
held constant while scaling all other dimensions. When $W$ becomes smaller than $1.75 \ H$, the interwire capacitance starts to dominate.

![Figure 4.6](image1.png) Capacitive coupling between wires in interconnect hierarchy.

![Figure 4.7](image2.png) Interconnect capacitance as a function of design rules. It consists of a capacitance to ground and an inter-wire capacitance (from [Schaper83]).

**Interconnect Capacitance Design Data**

A set of typical interconnect capacitances for a standard 0.25 $\mu$m CMOS process are given in Table 4.2. The process supports 1 layer of polysilicon and 5 layers of Aluminum. Metal layers 1 to 4 have the same thickness and use a similar dielectric, while the wires at metal layer 5 are almost twice as thick and are embedded in a dielectric with a higher permittivity. When placing the wires over the thick field oxide that is used to isolate different transistors, use the “Field” column in the table, while wires routed over the active area see a higher capacitance as seen in the “Active” column. Be aware that the presented values are only indicative. To obtain more
Section 4.3  Interconnect Parameters — Capacitance, Resistance, and Inductance

accurate results for actual structures, complex 3-dimensional models should be used that take
the environment of the wire into account.

Table 4.2  Wire area and fringe capacitance values for typical 0.25 \( \mu \)m CMOS process. The table rows represent the top plate of the capacitor, the columns the bottom plate. The area capacitances are expressed in aF/\( \mu \)m\(^2\), while the fringe capacitances (given in the shaded rows) are in aF/\( \mu \)m.

<table>
<thead>
<tr>
<th>Field</th>
<th>Active</th>
<th>Poly</th>
<th>Al1</th>
<th>Al2</th>
<th>Al3</th>
<th>Al4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poly</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Al1</td>
<td>30</td>
<td>41</td>
<td>57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>47</td>
<td>54</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Al2</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>27</td>
<td>29</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Al3</td>
<td>8.9</td>
<td>9.4</td>
<td>10</td>
<td>15</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>27</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>Al4</td>
<td>6.5</td>
<td>6.8</td>
<td>7</td>
<td>8.9</td>
<td>15</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>15</td>
<td>15</td>
<td>18</td>
<td>27</td>
<td>45</td>
</tr>
<tr>
<td>Al5</td>
<td>5.2</td>
<td>5.4</td>
<td>5.4</td>
<td>6.6</td>
<td>9.1</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td>27</td>
</tr>
</tbody>
</table>

Table 4.3  Interwire capacitance per unit wire length for different interconnect layers of typical 0.25 \( \mu \)m CMOS process. The capacitances are expressed in aF/\( \mu \)m, and are for minimally-spaced wires.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Poly</th>
<th>Al1</th>
<th>Al2</th>
<th>Al3</th>
<th>Al4</th>
<th>Al5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacitance</td>
<td>40</td>
<td>95</td>
<td>85</td>
<td>85</td>
<td>85</td>
<td>115</td>
</tr>
</tbody>
</table>
Example 4.1 Capacitance of Metal Wire

Some global signals, such as clocks, are distributed all over the chip. The length of those wires can be substantial. For die sizes between 1 and 2 cm, wires can reach a length of 10 cm and have associated wire capacitances of substantial value. Consider an aluminum wire of 10 cm long and 1 µm wide, routed on the first Aluminum layer. We can compute the value of the total capacitance using the data presented in Table 4.2.

Area (parallel-plate) capacitance: 
\[
(0.1 \times 10^6 \text{ µm}^2) \times 30 \text{ aF/µm}^2 = 3 \text{ pF}
\]

Fringing capacitance: 
\[
2 \times (0.1 \times 10^6 \text{ µm} \times 40 \text{ aF/µm} = 8 \text{ pF}
\]

Total capacitance: 11 pF

Notice the factor 2 in the computation of the fringing capacitance, which takes the two sides of the wire into account.

Suppose now that a second wire is routed alongside the first one, separated by only the minimum allowed distance. From Table 4.3, we can determine that this wire will couple to the first with a capacitance equal to

\[
C_{\text{int}} = (0.1 \times 10^6 \text{ µm}) \times 95 \text{ aF/µm} = 9.5 \text{ pF}
\]

which is almost as large as the total capacitance to ground!

A similar exercise shows that moving the wire to Al4 would reduce the capacitance to ground to 3.45 pF (0.65 pF area and 2.8 pF fringe), while the interwire capacitance would remain approximately the same at 8.5 pF.

4.3.2 Resistance

The resistance of a wire is proportional to its length \(L\) and inversely proportional to its cross-section \(A\). The resistance of a rectangular conductor in the style of Figure 4.3 can be expressed as

\[
R = \frac{\rho L}{A} = \frac{\rho L}{HW}
\]  

where the constant \(\rho\) is the resistivity of the material (in \(\Omega\cdot\text{m}\)). The resistivities of some commonly-used conductive materials are tabulated in Table 4.4. Aluminum is the interconnect material most often used in integrated circuits because of its low cost and its compatibility with the standard integrated-circuit fabrication process. Unfortunately, it has a large resistivity compared to materials such as Copper. With ever-increasing performance targets, this is rapidly becoming a liability and top-of-the-line processes are now increasingly using Copper as the conductor of choice.

<table>
<thead>
<tr>
<th>Material</th>
<th>(\rho) (Ω·m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Silver (Ag)</td>
<td>(1.6 \times 10^{-4})</td>
</tr>
</tbody>
</table>
Since \( H \) is a constant for a given technology, Eq. (4.3) can be rewritten as follows,

\[
R = \frac{R_d}{\sqrt{\frac{L}{W}}}
\]

with

\[
R_d = \frac{\rho}{H}
\]

the sheet resistance of the material, having units of \( \Omega/\text{sq} \) (pronounced as Ohm-per-square). This expresses that the resistance of a square conductor is independent of its absolute size, as is apparent from Eq. (4.4). To obtain the resistance of a wire, simply multiply the sheet resistance by its ratio \((L/W)\).

**Table 4.4** Resistivity of commonly-used conductors (at 20 C).

<table>
<thead>
<tr>
<th>Material</th>
<th>( \rho ) (( \Omega \cdot \text{m} ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copper (Cu)</td>
<td>( 1.7 \times 10^{-8} )</td>
</tr>
<tr>
<td>Gold (Au)</td>
<td>( 2.2 \times 10^{-8} )</td>
</tr>
<tr>
<td>Aluminum (Al)</td>
<td>( 2.7 \times 10^{-8} )</td>
</tr>
<tr>
<td>Tungsten (W)</td>
<td>( 5.5 \times 10^{-8} )</td>
</tr>
</tbody>
</table>

Typical values of the sheet resistance of various interconnect materials are given in Table 4.5.

**Table 4.5** Sheet resistance values for a typical 0.25 \( \mu \text{m} \) CMOS process.

<table>
<thead>
<tr>
<th>Material</th>
<th>Sheet Resistance (( \Omega/\text{sq} ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n ) - or ( p )-well diffusion</td>
<td>1000 – 1500</td>
</tr>
<tr>
<td>( n' ), ( p' ) diffusion</td>
<td>50 – 150</td>
</tr>
<tr>
<td>( n' ), ( p' ) diffusion with silicide</td>
<td>3 – 5</td>
</tr>
<tr>
<td>( n' ), ( p' ) polysilicon</td>
<td>150 – 200</td>
</tr>
<tr>
<td>( n' ), ( p' ) polysilicon with silicide</td>
<td>4 – 5</td>
</tr>
<tr>
<td>Aluminum</td>
<td>0.05 – 0.1</td>
</tr>
</tbody>
</table>

From this table, we conclude that Aluminum is the preferred material for the wiring of long interconnections. Polysilicon should only be used for local interconnect. Although the sheet resistance of the diffusion layer \((n', p')\) is comparable to that of polysilicon, the use of diffusion wires should be avoided due to its large capacitance and the associated RC delay.
Advanced processes also offer silicided polysilicon and diffusion layers. A silicide is a compound material formed using silicon and a refractory metal. This creates a highly conductive material that can withstand high-temperature process steps without melting. Examples of silicides are $\text{WSi}_2$, $\text{TiSi}_2$, $\text{PtSi}_2$, and $\text{TaSi}$. $\text{WSi}_2$, for instance, has a resistivity $\rho$ of 130 $\mu\Omega\cdot\text{cm}$, which is approximately eight times lower than polysilicon. The silicides are most often used in a configuration called a polycide, which is a simple layered combination of polysilicon and a silicide. A typical polycide consists of a lower level of polysilicon with an upper coating of silicide and combines the best properties of both materials—good adherence and coverage (from the poly) and high conductance (from the silicide). A MOSFET fabricated with a polycide gate is shown in Figure 4.8. The advantage of the silicided gate is a reduced gate resistance. Similarly, silicided source and drain regions reduce the source and drain resistance of the device.

Transitions between routing layers add extra resistance to a wire, called the contact resistance. The preferred routing strategy is thus to keep signal wires on a single layer whenever possible and to avoid excess contacts or vias. It is possible to reduce the contact resistance by making the contact holes larger. Unfortunately, current tends to concentrate around the perimeter in a larger contact hole. This effect, called current crowding, puts a practical upper limit on the size of the contact. The following contact resistances (for minimum-size contacts) are typical for a 0.25 $\mu\text{m}$ process: 5-20 $\Omega$ for metal or polysilicon to $n^+$, $p^+$, and metal to polysilicon; 1-5 $\Omega$ for via’s (metal-to-metal contacts).

**Example 4.2 Resistance of a Metal Wire**

Consider again the aluminum wire of Example 4.2, which is 10 cm long and 1 $\mu\text{m}$ wide, and is routed on the first Aluminum layer. Assuming a sheet resistance for Al1 of 0.075 $\Omega/\square$, we can compute the total resistance of the wire

$$R_{\text{wire}} = 0.075 \frac{\Omega}{\square} \times (0.1 \times 10^6 \mu\text{m}) / (1 \mu\text{m}) = 7.5 \text{k}\Omega$$

Implementing the wire in polysilicon with a sheet resistance of 175 $\Omega/\square$ raises the overall resistance to 17.5 M$\Omega$, which is clearly unacceptable. Silicided polysilicon with a sheet resistance of 4 $\Omega/\square$ offers a better alternative, but still translates into a wire with a 400 k$\Omega$ resistance.

So far, we have considered the resistance of a semiconductor wire to be linear and constant. This is definitely the case for most semiconductor circuits. At very high frequencies however, an additional phenomenon — called the skin effect — comes into play such that the resistance becomes frequency-dependent. High-frequency currents tend to flow primarily on the surface of a conductor with the current density falling off exponentially...
Section 4.3 Interconnect Parameters — Capacitance, Resistance, and Inductance

with depth into the conductor. The **skin depth** $\delta$ is defined as the depth where the current falls off to a value of $e^{-1}$ of its nominal value, and is given by

$$\delta = \frac{\rho}{\sqrt{\pi f \mu}}$$  \hspace{1cm} (4.6)

with $f$ the frequency of the signal and $\mu$ the permeability of the surrounding dielectric (typically equal to the permeability of free space, or $\mu = 4\pi \times 10^{-7}$ H/m). For Aluminum at 1 GHz, the skin depth is equal to 2.6 $\mu$m. The obvious question is now if this is something we should be concerned about when designing state-of-the-art digital circuits?

The effect can be approximated by assuming that the current flows uniformly in an outer shell of the conductor with thickness $\delta$, as is illustrated in Figure 4.9 for a rectangular wire. Assuming that the overall cross-section of the wire is now limited to approximately $2(W+H)\delta$, we obtain the following expression for the resistance (per unit length) at high frequencies ($f \gg f_s$):

$$r(f) = \frac{\sqrt{\pi f / \mu \rho}}{2(H+W)}$$  \hspace{1cm} (4.7)

The increased resistance at higher frequencies may cause an extra attenuation — and hence distortion — of the signal being transmitted over the wire. To determine the on-set of the skin-effect, we can find the frequency $f_s$ where the skin depth is equal to half the largest dimension (W or H) of the conductor. Below $f_s$ the whole wire is conducting current, and the resistance is equal to (constant) low-frequency resistance of the wire. From Eq. (4.6), we find the value of $f_s$:

$$f_s = \frac{4\rho}{\pi \mu (\max(W,H))^2}$$  \hspace{1cm} (4.8)

**Example 4.3 Skin-effect and Aluminum wires**

We determine the impact of the skin-effect on contemporary integrated circuits by analyzing an Aluminum wire with a resistivity of $2.7 \times 10^{-8}$ $\Omega$-m, embedded in a SiO$_2$ dielectric with a permeability of $4\pi \times 10^{-7}$ H/m. From Eq. (4.8), we find that the largest dimension of wire should be at least 5.2 $\mu$m for the effect to be noticeable at 1 GHz. This is confirmed by the more accurate simulation results of Figure 4.10, which plots the increase in resistance due to skin effects for different width Aluminum conductors. A 30% increase in resistance...
can be observed at 1 GHz for a 20 µm wire, while the increase for a 1 µm wire is less than 1%.

In summary, the skin-effect is only an issue for wider wires. Since clocks tend to carry the highest-frequency signals on a chip and also are fairly wide to limit resistance, the skin effect is likely to have its first impact on these lines. This is a real concern for GHz-range design, as clocks determine the overall performance of the chip (cycle time, instructions per second, etc.). Another major design concern is that the adoption of better conductors such as Copper may move the on-set of skin-effects to lower frequencies.

4.3.3 Inductance

Integrated-circuit designers tend to dismiss inductance as something they heard about in their physics classes, but that has no impact on their field. This was definitely the case in the first decades of integrated digital circuit design. Yet with the adoption of low-resistive interconnect materials and the increase of switching frequencies to the super GHz range, inductance starts to play a role even on a chip. Consequences of on-chip inductance include ringing and overshoot effects, reflections of signals due to impedance mismatch, inductive coupling between lines, and switching noise due to $L\frac{di}{dt}$ voltage drops.

The inductance of a section of a circuit can always be evaluated with the aid of its definition, which states that a changing current passing through an inductor generates a voltage drop $\Delta V$

$$\Delta V = L\frac{di}{dt} \quad (4.9)$$
Section 4.3 Interconnect Parameters — Capacitance, Resistance, and Inductance

It is possible to compute the inductance of a wire directly from its geometry and its environment. A simpler approach relies on the fact that the capacitance $c$ and the inductance $l$ (per unit length) of a wire are related by the following expression

$$cl = \varepsilon\mu$$  

(4.10)

with $\varepsilon$ and $\mu$ respectively the permittivity and permeability of the surrounding dielectric. The caveat is that for this expression to be valid the conductor must be completely surrounded by a uniform dielectric medium. This is most often not the case. Yet even when the wire is embedded in different dielectric materials, it is possible to adopt “average” dielectric constants such that Eq. (4.10) still can be used to get an approximative value of the inductance.

Some other interesting relations, obtained from Maxwell’s laws, can be pointed out. The constant product of permeability and permittivity also defines the speed $\nu$ at which an electromagnetic wave can propagate through the medium

$$\nu = \frac{1}{\sqrt{\varepsilon\mu}} = \frac{c_0}{\sqrt{\varepsilon_r\mu_r}}$$  

(4.11)

c_0 equals the speed of light (30 cm/nsec) in a vacuum. The propagation speeds for a number of materials used in the fabrication of electronic circuits are tabulated in Table 4.6. The propagation speed for SiO$_2$ is two times slower than in a vacuum.

<table>
<thead>
<tr>
<th>Dielectric</th>
<th>$\varepsilon_r$</th>
<th>Propagation speed (cm/nsec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vacuum</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>SiO$_2$</td>
<td>3.9</td>
<td>15</td>
</tr>
<tr>
<td>PC board (epoxy glass)</td>
<td>5.0</td>
<td>13</td>
</tr>
<tr>
<td>Alumina (ceramic package)</td>
<td>9.5</td>
<td>10</td>
</tr>
</tbody>
</table>

**Example 4.4 Inductance of a Semiconductor Wire**

Consider an Al wire implemented in the 0.25 micron CMOS technology and routed on top of the field oxide. From Table 4.2, we can derive the capacitance of the wire per unit length:

$$c = (W \times 30 + 2 \times 40) \text{ aF/\mu m}$$

From Eq. (4.10), we can derive the inductance per unit length of the wire, assuming SiO$_2$ as the dielectric and assuming a uniform dielectric (make sure to use the correct units!)

$$l = (3.9 \times 8.854 \times 10^{-12}) \times (4 \pi 10^{-7}) / C$$

For wire widths of 0.4 $\mu$m, 1$\mu$m and 10$\mu$m, this leads to the following numbers:
\[ W = 0.4 \ \mu m; \ c = 92 \ aF/\mu m; \ l = 0.47 \ \mu H/\mu m \]
\[ W = 1 \ \mu m; \ c = 110 \ aF/\mu m; \ l = 0.39 \ \mu H/\mu m \]
\[ W = 10 \ \mu m; \ c = 380 \ aF/\mu m; \ l = 0.11 \ \mu H/\mu m \]

Assuming a sheet resistance of 0.075 \( \Omega/\mu m \), we can also determine the resistance of the wire,

\[ r = \frac{0.075}{W} \ \Omega/\mu m \]

It is interesting to observe that the inductive part of the wire impedance becomes equal in value to the resistive component at a frequency of 27.5 GHz (for a 1 \( \mu m \) wide wire), as can be obtained from solving the following expression:

\[ \omega l = \frac{2\pi f l}{r} \]

For extra wide wires, this frequency reduces to approximately 11 GHz. For wires with a smaller capacitance and resistance (such as the thicker wires located at the upper interconnect layers), this frequency can become as low as 500 MHz, especially when better interconnect materials such as Copper are being used. Yet, these numbers indicate that inductance only becomes an issue in integrated circuits for frequencies that are well above 1 GHz.

### 4.4 Electrical Wire Models

In previous sections, we have introduced the electrical properties of the interconnect wire — capacitance, resistance, and inductance — and presented some simple relations and techniques to derive their values from the interconnect geometries and topologies. These parasitic elements have an impact on the electrical behavior of the circuit and influence its delay, power dissipation, and reliability. To study these effects requires the introduction of electrical models that estimate and approximate the real behavior of the wire as a function of its parameters. These models vary from very simple to very complex depending upon the effects that are being studied and the required accuracy. In this section, we first derive models for manual analysis, while how to cope with interconnect wires in the SPICE circuit simulator is the topic follows next.

#### 4.4.1 The Ideal Wire

In schematics, wires occur as simple lines with no attached parameters or parasitics. These wires have no impact on the electrical behavior of the circuit. A voltage change at one end of the wire propagates immediately to its other ends, even if those are some distance away. Hence, it may be assumed that the same voltage is present at every segment of the wire at the every point in time, and that the whole wire is an equipotential region. While this ideal-wire model is simplistic, it has its value, especially in the early phases of the design process when the designer wants to concentrate on the properties and the behavior of the transistors that are being connected. Also, when studying small circuit components such as gates, the wires tend to be very short and their parasites ignorable. Taking these into account would just make the analysis unnecessarily complex. More often though, wire parasites play a role and more complex models should be considered.
4.4.2 The Lumped Model

The circuit parasitics of a wire are distributed along its length and are not lumped into a single position. Yet, when only a single parasitic component is dominant, when the interaction between the components is small, or when looking at only one aspect of the circuit behavior, it is often useful to lump the different fractions into a single circuit element. The advantage of this approach is that the effects of the parasitic then can be described by an ordinary differential equation. As we will see later, the description of a distributed element requires partial differential equations.

As long as the resistive component of the wire is small and the switching frequencies are in the low to medium range, it is meaningful to consider only the capacitive component of the wire, and to lump the distributed capacitance into a single capacitor as shown in Figure 4.11. Observe that in this model the wire still represents an equipotential region, and that the wire itself does not introduce any delay. The only impact on performance is introduced by the loading effect of the capacitor on the driving gate. This capacitive lumped model is simple, yet effective, and is the model of choice for the analysis of most interconnect wires in digital integrated circuits.

Example 4.5 Lumped capacitance model of wire

Example 4.5 Lumped capacitance model of wire

For the circuit of Figure 4.11, assume that a driver with a source resistance of 10 kΩ is used to drive a 10 cm long, 1 µm wide Al1 wire. In Example 4.1, we have found that the total lumped capacitance for this wire equals 11 pF.

The operation of this simple RC network is described by the following ordinary differential equation:

$$C_{lumped} \frac{dV_{out}}{dt} + \frac{V_{out} - V_{in}}{R_{driver}} = 0$$

When applying a step input (with $V_{in}$ going from 0 to $V$), the transient response of this circuit is known to be an exponential function, and is given by the following expression (where $\tau = R_{driver} \times C_{lumped}$ the time constant of the network):

$$V_{out}(t) = (1 - e^{-\frac{t}{\tau}}) V$$

The time to reach the 50% point is easily computed as $t = ln(2) \tau = 0.69\tau$. Similarly, it takes $t = ln(9)\tau = 2.2\tau$ to get to the 90% point. It is worth memorizing these numbers, as they are extensively used in the rest of the text. Plugging in the numbers for this specific example yields...
\[ t_{50\%} = 0.69 \times 10 \, \text{K}\Omega \times 11 \, \text{pF} = 76 \, \text{nsec} \]
\[ t_{90\%} = 2.2 \times 10 \, \text{K}\Omega \times 11 \, \text{pF} = 242 \, \text{nsec} \]

These numbers are not even acceptable for the lowest performance digital circuits. Techniques to deal with this bottleneck, such as reducing the source resistance of the driver, will be introduced in Chapter \textit{ZZZ}.

While the lumped capacitor model is the most popular, sometimes it is also useful to present lumped models of a wire with respect to either resistance and inductance. This is often the case when studying the supply distribution network. Both the resistance and inductance of the supply wires can be interpreted as parasitic noise sources that introduce voltage drops and bounces on the supply rails.

\subsection*{4.4.3 The Lumped \textit{RC} model}

On-chip metal wires of over a few mm length have a significant resistance. The equipotential assumption, presented in the lumped-capacitor model, is no longer adequate, and a resistive-capacitive model has to be adopted.

A first approach lumps the total wire resistance of each wire segment into one single \( R \) and similarly combines the global capacitance into a single capacitor \( C \). This simple model, called the \textit{lumped RC model} is pessimistic and inaccurate for long interconnect wires, which are more adequately represented by a \textit{distributed rc-model}. Yet, before analyzing the distributed model, its is worthwhile to spend some time on the analysis and the modeling of lumped \textit{RC} networks for the following reasons:

- The distributed \textit{rc}-model is complex and no closed form solutions exist. The behavior of the distributed \textit{rc}-line can be adequately modeled by a simple \textit{RC} network.
- A common practice in the study of the transient behavior of complex transistor-wire networks is to reduce the circuit to an \textit{RC} network. Having a means to analyze such a network effectively and to predict its first-order response would add a great asset to the designer's tool box.

In Example 4.5, we analyzed a single resistor-single capacitor network. The behavior of such a network is fully described by a single differential equation, and its transient waveform is modeled by an exponential with a single time-constant (or network pole). Unfortunately, deriving the correct waveforms for a network with a larger number of capacitors and resistors rapidly becomes hopelessly complex: describing its behavior requires a set of ordinary differential equations, and the network now contains many time-constants (or poles and zeros). Short of running a full-fledged SPICE simulation, delay calculation methods such as the \textit{Elmore delay formula} come to the rescue [Elmore48].

Consider the resistor-capacitor network of Figure 4.12. This circuit is called an \textit{RC-tree} and has the following properties:

- the network has a single input node (called \textit{s} in Figure 4.12)
- all the capacitors are between a node and the ground
- the network does not contain any resistive loops (which makes it a tree)
An interesting result of this particular circuit topology is that there exists a unique resistive path between the source node $s$ and any node $i$ of the network. The total resistance along this path is called the path resistance $R_{ii}$. For example, the path resistance between the source node $s$ and node 4 in the example of Figure 4.12 equals

$$R_{44} = R_1 + R_3 + R_4$$

The definition of the path resistance can be extended to address the shared path resistance $R_{ik}$, which represents the resistance shared among the paths from the root node $s$ to nodes $k$ and $i$:

$$R_{ik} = \sum R_j \Rightarrow (R_j \in [\text{path}(s \rightarrow i) \cap \text{path}(s \rightarrow k)])$$

(4.12)

For the circuit of Figure 4.12, $R_{44} = R_1 + R_3$ while $R_{12} = R_1$.

Assume now that each of the $N$ nodes of the network is initially discharged to GND, and that a step input is applied at node $s$ at time $t = 0$. The Elmore delay at node $i$ is then given by the following expression:

$$\tau_{Di} = \sum_{k=1}^{N} C_k R_{ik}$$

(4.13)

The Elmore delay is equivalent to the first-order time constant of the network (or the first moment of the impulse response). The designer should be aware that this time-constant represents a simple approximation of the actual delay between source node and node $i$. Yet in most cases this approximation has proven to be quite reasonable and acceptable. It offers the designer a powerful mechanism for providing a quick estimate of the delay of a complex network.

**Example 4.6  RC delay of a tree-structured network**

Using Eq. (4.13), we can compute the Elmore delay for node $i$ in the network of Figure 4.12.

$$\tau_{Di} = R_1 C_i + R_1 C_2 + (R_1 + R_3) C_3 + (R_1 + R_3) C_4 + (R_1 + R_3 + R_i) C_i$$
As a special case of the RC tree network, let us consider the simple, non-branched RC chain (or ladder) shown in Figure 4.13. This network is worth analyzing because it is a structure that is often encountered in digital circuits, and also because it represents an approximative model of a resistive-capacitive wire. The Elmore delay of this chain network can be derived with the aid of Eq. (4.13):

\[ \tau_i = C_1 R_1 + C_2 (R_1 + R_2) + \ldots + C_i (R_1 + R_2 + \ldots + R_i) \]

or the shared-path resistance is replaced by simply the path resistance. As an example, consider node 2 in the RC chain of Figure 4.13. Its time-constant consists of two components contributed by nodes 1 and 2. The component of node 1 consists of \( C_1 R_1 \) with \( R_1 \) the total resistance between the node and the source, while the contribution of node 2 equals \( C_2 (R_1 + R_2) \). The equivalent time constant at node 2 equals \( C_1 R_1 + C_2 (R_1 + R_2) \). \( \tau_i \) of node \( i \) can be derived in a similar way.

\[ \tau_{DN} = \sum_{i=1}^{N} C_i \sum_{j=1}^{i} R_j = \sum_{i=1}^{N} C_i R_{i+1} \]

or the shared-path resistance is replaced by simply the path resistance.

**Example 4.7 Time-Constant of Resistive-Capacitive Wire**

The model presented in Figure 4.13 can be used as an approximation of a resistive-capacitive wire. The wire with a total length of \( L \) is partitioned into \( N \) identical segments, each with a length of \( L/N \). The resistance and capacitance of each segment are hence given by \( rL/N \) and \( cL/N \), respectively. Using the Elmore formula, we can compute the dominant time-constant of the wire:

\[ \tau_{DN} = \left( \frac{L}{N} \right)^2 (rc + 2rc + \ldots + Nrc) = (rcL^2) \frac{N(N+1)}{2N^2} = RC \left( \frac{N+1}{2N} \right) \]

with \( R (= rL) \) and \( C (= cL) \) the total lumped resistance and capacitance of the wire. For very large values of \( N \), this model asymptotically approaches the distributed \( rc \) line. Eq. (4.15) then simplifies to the following expression:

\[ \tau_{DN} = \frac{RC}{2} = \frac{rcL^2}{2} \]

Eq. (4.16) leads to two important conclusions:

- The delay of a wire is a **quadratic function of its length**! This means that doubling the length of the wire quadruples its delay.
• The delay of the distributed \textit{rc}-line is \textbf{one half of the delay} that would have been predicted by the lumped \textit{RC} model. The latter combines the total resistance and capacitance into single elements, and has a time-constant equal to \textit{RC} (as is also obtained by setting \( N = 1 \) in Eq. (4.15)). This confirms the observation made earlier that the lumped model presents a pessimistic view on the delay of resistive wire.

\textbf{WARNING:} Be aware that an \textit{RC}-chain is characterized by a number of time-constants. The Elmore expression determines the value of only the dominant one, and presents thus a first-order approximation.

The Elmore delay formula has proven to be extremely useful. Besides making it possible to analyze wires, the formula can also be used to approximate the propagation delay of complex transistor networks. In the switch model, transistors are replaced by their equivalent, linearized on-resistance. The evaluation of the propagation delay is then reduced to the analysis of the resulting \textit{RC} network. More precise minimum and maximum bounds on the voltage waveforms in an \textit{RC} tree have further been established [Rubinstein83]. These bounds have formed the basis for most computer-aided timing analyzers at the switch and functional level [Horowitz83]. An interesting result [Ref] is that the exponential voltage waveform with the Elmore delay as time constant is always situated between these min and max bounds, which demonstrates the validity of the Elmore approximation.

### 4.4.4 The Distributed \textit{rc} Line

In the previous paragraphs, we have shown that the lumped \textit{RC} model is a pessimistic model for a resistive-capacitive wire, and that a distributed \textit{rc} model (Figure 4.14a) is more appropriate. As before, \( L \) represents the total length of the wire, while \( r \) and \( c \) stand for the resistance and capacitance per unit length. A schematic representation of the distributed \textit{rc} line is given in Figure 4.14b.

The voltage at node \( i \) of this network can be determined by solving the following set of partial differential equations:

\[
\frac{c \Delta L}{\Delta t} \frac{\partial V_i}{\partial t} = \frac{(V_{i+1} - V_i) + (V_i - V_{i-1})}{r \Delta L}
\]  

(4.17)

The correct behavior of the distributed \textit{rc} line is then obtained by reducing \( \Delta L \) asymptotically to 0. For \( \Delta L \to 0 \), Eq. (4.17) becomes the well-known \textit{diffusion equation}:

\[
rc \frac{\partial^2 V}{\partial x^2} = \frac{\partial V}{\partial t}
\]  

(4.18)

where \( V \) is the voltage at a particular point in the wire, and \( x \) is the distance between this point and the signal source. No closed-form solution exists for this equation, but approximative expressions such as the formula presented in Eq. (4.19) can be derived [Bakoglu90]. These equations are difficult to use for ordinary circuit analysis. It is known however that the distributed \textit{rc} line can be approximated by a lumped \textit{RC} ladder network,
which can be easily used in computer-aided analysis. Some of these models will be presented in a later section, discussing SPICE wire models.

\[
V_{\text{out}}(t) = 2 \text{erfc} \left( \frac{R_C}{\sqrt{4t}} \right) 
\]

\[
= 1.0 - 1.366e^{-2.5359 \frac{L}{R_C}} + 0.366e^{-9.4641 \frac{L}{R_C}} 
\]

\( t \ll RC \)

\( t \gg RC \)

Figure 4.15 shows the response of a wire to a step input, plotting the waveforms at different points in the wire as a function of time. Observe how the step waveform “diffuses” from the start to the end of the wire, and the waveform rapidly degrades, resulting in a considerable delay for long wires. Driving these \( rc \) lines and minimizing the delay and
signal degradation is one of the trickiest problems in modern digital integrated circuit design. It hence will receive considerable attention in later chapters.

Some of the important reference points in the step response of the lumped and the distributed RC model of the wire are tabulated in Table 4.7. For instance, the propagation delay (defined at 50% of the final value) of the lumped network not surprisingly equals $0.69 \, RC$. The distributed network, on the other hand, has a delay of only $0.38 \, RC$, with $R$ and $C$ the total resistance and capacitance of the wire. This confirms the result of Eq. (4.16).

**Example 4.8 RC delay of Aluminum Wire**

Let us consider again the 10 cm long, 1 µm wide Al wire of Example 4.1. In Example 4.4, we derived the following values for $r$ and $c$:

$$c = 110 \, \text{aF/µm}; \quad r = 0.075 \, \Omega/\text{µm};$$

Using the entry of Table 4.7, we derive the propagation delay of the wire:

$$t_p = 0.38 \, RC = 0.38 \times (0.075 \, \Omega/\text{µm}) \times (110 \, \text{aF/µm}) \times (10^5 \, \text{µm})^2 = 31.4 \, \text{nsec}$$

We can also deduce the propagation delays of an identical wire implemented in polysilicon and Al5. The values of the capacitances are obtained from Table 4.2, while the resistances are assumed to be respectively $150 \, \Omega/\mu m$ and $0.0375 \, \Omega/\mu m$ for Poly and Al5.
THE WIRE  Chapter 4

Poly: \( t_p = 0.38 \times (150 \, \Omega/\mu m) \times (88 + 2 \times 54 \, aF/\mu m) \times (10^5 \, \mu m)^2 = 112 \, \mu sec \)

Al5: \( t_p = 0.38 \times (0.0375 \, \Omega/\mu m) \times (5.2 + 2 \times 12 \, aF/\mu m) \times (10^5 \, \mu m)^2 = 4.2 \, nsec \)

Obviously, the choice of the interconnect material and layer has a dramatic impact on the delay of the wire.

An important question for a designer to answer when analyzing an interconnect network whether the effects of \( RC \) delays should be considered, or whether she can get away with a simpler lumped capacitive model. A simple rule of thumb proves to be very useful here.

### Design Rules of Thumb

- **\( rc \) delays should only be considered when \( t_{pRC} \gg t_{pgate} \) of the driving gate.**

  This translates into Eq. (4.20), which determines the critical length \( L \) of the interconnect wire where \( RC \) delays become dominant.

\[
L_{crit} \gg \sqrt{\frac{t_{pgate}}{0.38 rc}}
\]  

(4.20)

The actual value of \( L_{crit} \) depends upon the sizing of the driving gate and the chosen interconnect material.

- **\( rc \) delays should only be considered when the rise (fall) time at the line input is smaller than \( RC \), the rise (fall) time of the line.**

\[
t_{rise} < RC
\]  

(4.21)

with \( R \) and \( C \) the total resistance and capacitance of the wire. When this condition is not met, the change in signal is slower than the propagation delay of the wire, and a lumped capacitive model suffices.

### Example 4.9  RC versus Lumped C

The presented rule can be illustrated with the aid of the simple circuit shown in Figure 4.16. It is assumed here that the driving gate can be modeled as voltage source with a finite source resistance \( R_s \). The total propagation delay of the network can be approximated by the following expression, obtained by applying the Elmore formula:\(^2\)

\[
\tau_D = R_s C_w + \frac{R_w C_w}{2} = R_s C_w + 0.5 R_w C_w L^2
\]

and

\(^2\) Hint: replace the wire by the lumped RC network of Figure 4.13 and apply the Elmore equation on the resulting network.
Section 4.4 Electrical Wire Models

with \( R_w = rL \) and \( C_w = cL \). The delay introduced by the wire resistance becomes dominant when \( (R_wC_w)/2 \geq R_sC_w \) or \( L \geq 2R_s/r \). Assume now a driver with a source resistance of 1 kΩ, driving an Al1 wire of 1 µm wide \( (r = 0.075 \, \Omega/\mu m) \). This leads to a critical length of 2.67 cm.

4.4.5 The Transmission Line

When the switching speeds of the circuits become sufficiently fast, and the quality of the interconnect material become high enough so that the resistance of the wire is kept within bounds, the inductance of the wire starts to dominate the delay behavior, and transmission line effects must be considered. This is more precisely the case when the rise and fall times of the signal become comparable to the time of flight of the signal waveform across the line as determined by the speed of light. With the advent of Copper interconnect and the high switching speeds enabled by the deep-submicron technologies, transmission line effects are soon to be considered in the fastest CMOS designs.

In this section, we first analyze the transmission line model. Next, we apply it to the current semiconductor technology and determine when those effects should be actively considered in the design process.

Transmission Line Model

Similar to the resistance and capacitance of an interconnect line, the inductance is distributed over the wire. A distributed \( rlc \) model of a wire, known as the transmission line model, becomes the most accurate approximation of the actual behavior. The transmission line has the prime property that a signal propagates over the interconnection medium as a wave. This is in contrast to the distributed \( rc \) model, where the signal diffuses from the source to the destination governed by the diffusion equation, Eq. (4.18). In the wave mode, a signal propagates by alternatively transferring energy from the electric to the magnetic fields, or equivalently from the capacitive to the inductive modes.

Consider the point \( x \) along the transmission line of Figure 4.17 at time \( t \). The following set of equations holds:
Assuming that the leakage conductance $g$ equals 0, which is true for most insulating materials, and eliminating the current $i$ yields the wave propagation equation, Eq. (4.23).

$$\frac{\partial v}{\partial x} = -ri - \frac{\partial i}{\partial t}$$

(4.22)

$$\frac{\partial i}{\partial x} = -gv - c\frac{\partial v}{\partial t}$$

Assuming that the leakage conductance $g$ equals 0, which is true for most insulating materials, and eliminating the current $i$ yields the wave propagation equation, Eq. (4.23).

$$\frac{\partial^2 v}{\partial x^2} = -ri - \frac{\partial i}{\partial t}$$

(4.23)

where $r$, $c$, and $l$ are the resistance, capacitance, and inductance per unit length, respectively.

To understand the behavior of the transmission line, we will first assume that the resistance of the line is small. In this case, a simplified capacitive/inductive model, called the lossless transmission line, is appropriate. This model is applicable for wires at the printed-circuit board level. Due to the high conductivity of the Copper interconnect material used there, the resistance of the transmission line can be ignored. On the other hand, resistance plays an important role in integrated circuits, and a more complex model, called the lossy transmission line should be considered. The lossy model is only discussed briefly at the end.

**The Lossless Transmission Line**

For the lossless line, Eq. (4.23) simplifies to the ideal wave equation:

$$\frac{\partial^2 v}{\partial x^2} = -ri - \frac{\partial i}{\partial t}$$

(4.24)

A step input applied to a lossless transmission line propagates along the line with a speed $v$, given by Eq. (4.11) and repeated below.

$$v = \frac{1}{\sqrt{LC}} = \frac{1}{\sqrt{l/c}} = \frac{c_0}{\sqrt{\mu_0 \varepsilon_0}}$$

(4.25)

Even though the values of both $l$ and $c$ depend on the geometric shape of the wire, their product is a constant and is only a function of the surrounding media. The propagation delay per unit wire length ($t_p$) of a transmission line is the inverse of the speed:
Section 4.4 Electrical Wire Models

Let us now analyze how a wave propagates along a lossless transmission line. Suppose that a voltage step $V$ has been applied at the input and has propagated to point $x$ of the line (Figure 4.18). All currents are equal to 0 at the right side of $x$, while the voltage over the line equals $V$ at the left side. An additional capacitance $c dx$ must be charged for the wave to propagate over an additional distance $dx$. This requires the following current:

$$I = \frac{dQ}{dt} = \frac{dc}{dt} V = c \nu V = \frac{c}{\eta} V$$  \hspace{1cm} (4.27)

since the propagation speed of the signal $dx/dt$ equals $\nu$. This means that the signal sees the remainder of the line as areal impedance,

$$Z_0 = \frac{V}{I} = \frac{\int I}{\eta c} = \frac{\sqrt{\varepsilon / \mu}}{c} = \frac{1}{\nu}.$$  \hspace{1cm} (4.28)

This impedance, called the *characteristic impedance* of the line, is a function of the dielectric medium and the geometry of the conducting wire and isolator (Eq. (4.28)), and is independent of the length of the wire and the frequency. That a line of arbitrary length has a constant, real impedance is a great feature as it simplifies the design of the driver circuitry. Typical values of the characteristic impedance of wires in semiconductor circuits range from 10 to 200 $\Omega$.

**Example 4.10 Propagation Speeds of Signal Waveforms**

The information of Table 4.6 shows that it takes 1.5 nsec for a signal wave to propagate from source-to-destination on a 20 cm wire deposited on an epoxy printed-circuit board. If transmission line effects were an issue on silicon integrated circuits, it would take 0.67 nsec for the signal to reach the end of a 10 cm wire.

**WARNING:** The characteristic impedance of a wire is a function of the overall interconnect topology. The electro-magnetic fields in complex interconnect structures tend to be irregular, and are strongly influenced by issues such as the current return path. Providing a general answer to the latter problem has so far proven to be illusive, and no closed-formed
analytical solutions are typically available. Hence, accurate inductance and characteristic impedance extraction is still an active research topic. For some simplified structures, approximative expressions have been derived. For instance, the characteristic impedances of a triplate strip-line (a wire embedded in between two ground planes) and a semiconductor micro strip-line (wire above a semiconductor substrate) are approximated by Eq. (4.29) and Eq. (4.30), respectively.

\[
Z_0(\text{triplate}) \approx 94\Omega \frac{\mu_r}{\sqrt{\varepsilon_r}} \ln\left(\frac{2t + W}{H + W}\right)
\]  

(4.29)

and

\[
Z_0(\text{microstrip}) \approx 60\Omega \frac{\mu_r}{\sqrt{0.475\varepsilon_r + 0.67}} \ln\left(\frac{4t}{0.536W + 0.67H}\right)
\]  

(4.30)

Termination

The behavior of the transmission line is strongly influenced by the termination of the line. The termination determines how much of the wave is reflected upon arrival at the wire end. This is expressed by the reflection coefficient \( \rho \) that determines the relationship between the voltages and currents of the incident and reflected waveforms.

\[
\rho = \frac{V_{\text{refl}}}{V_{\text{inc}}} = \frac{I_{\text{refl}}}{I_{\text{inc}}} = \frac{R - Z_0}{R + Z_0}
\]  

(4.31)

where \( R \) is the value of the termination resistance. The total voltages and currents at the termination end are the sum of incident and reflected waveforms.

\[
V = V_{\text{inc}}(1 + \rho)
\]

\[
I = I_{\text{inc}}(1 - \rho)
\]  

(4.32)

Three interesting cases can be distinguished, as illustrated in Figure 4.19. In case (a) the terminating resistance is equal to the characteristic impedance of the line. The termination appears as an infinite extension of the line, and no waveform is reflected. This is also demonstrated by the value of \( \rho \), which equals 0. In case (b), the line termination is an open circuit \( (R = \infty) \), and \( \rho = 1 \). The total voltage waveform after reflection is twice the incident one as predicted by Eq. (4.32). Finally, in case (c) where the line termination is a short circuit, \( R = 0 \), and \( \rho = -1 \). The total voltage waveform after reflection equals zero.

The transient behavior of a complete transmission line can now be examined. It is influenced by the characteristic impedance of the line, the series impedance of the source \( Z_S \), and the loading impedance \( Z_L \) at the destination end, as shown in Figure 4.20.

Consider first the case where the wire is open at the destination end, or \( Z_L = \infty \), and \( \rho_L = 1 \). An incoming wave is completely reflected without phase reversal. Under the assumption that the source impedance is resistive, three possible scenarios are sketched in Figure 4.21: \( R_S = 5Z_0 \), \( R_S = Z_0 \), and \( R_S = 1/5Z_0 \).
Section 4.4 Electrical Wire Models

1. Large source resistance — $R_s = 5Z_0$ (Figure 4.21a)

Only a small fraction of the incoming signal $V_{in}$ is injected into the transmission line. The amount injected is determined by the resistive divider formed by the source resistance and the characteristic impedance $Z_0$:

$$V_{source} = \frac{Z_0}{Z_0 + R_s} V_{in} = \frac{1}{6} \times 5 \text{ V} = 0.83 \text{ V} \quad (4.33)$$

This signal reaches the end of the line after $L/v$ sec, where $L$ stands for the length of the wire and is fully reflected, which effectively doubles the amplitude of the wave ($V_{dist} = 1.67$ V). The time it takes for the wave to propagate from one end of the wire to the other is called the time-of-flight, $t_{flight} = L/v$. Approximately the same happens when the wave reaches the source node again. The incident waveform is reflected with an amplitude determined by the source reflection coefficient, which equals $2/3$ for this particular case.

$$\rho_s = \frac{5Z_0 - Z_0}{5Z_0 + Z_0} = \frac{2}{3} \quad (4.34)$$

The voltage amplitude at source and destination nodes gradually reaches its final value of $V_{in}$. The overall rise time is, however, many times $L/v$. 

---

**Figure 4.19** Behavior of various transmission line terminations.

**Figure 4.20** Transmission line with terminating impedances.
When multiple reflections are present, as in the above case, keeping track of waves on the line and total voltage levels rapidly becomes cumbersome. Therefore a graphical construction called the lattice diagram is often used to keep track of the data (Figure 4.22). The diagram contains the values of the voltages at the source and destination ends, as well as the values of the incident and reflected wave forms. The line voltage at a termination point equals the sum of the previous voltage, the incident, and reflected waves.

2. **Small source resistance** \( R_S = \frac{Z_0}{5} \) (Figure 4.21c)

A large portion of the input is injected in the line. Its value is doubled at the destination end, which causes a severe overshoot. At the source end, the phase of the signal is reversed \( \rho_S = -\frac{2}{3} \). The signal bounces back and forth and exhibits severe ringing. It takes multiple \( L/\nu \) before it settles.

3. **Matched source resistance** \( R_S = Z_0 \) (Figure 4.21b)

Half of the input signal is injected at the source. The reflection at the destination end doubles the signal, so that the final value is reached immediately. It is obvious that this is
Section 4.4 Electrical Wire Models

the most effective case. Matching the line impedance at the source end is called series termination.

Note that the above analysis is an ideal one, as it is assumed that the input signal has a zero rise time. In real conditions the signals are substantially smoother, as demonstrated in the simulated response of Figure 4.23 (for \( R_S = \frac{Z_0}{5} \) and \( t_r = t_{\text{flight}} \)).

Problem 4.1 Transmission Line Response

Derive the lattice diagram of the above transmission line for \( R_S = \frac{Z_0}{5}, R_L = \infty, V_{\text{step}} = 5 \) V. Also try the reverse picture—assume that the series resistance of the source equals zero, and consider different load impedances.

Similar considerations are valid when the termination is provided at the destination end, called parallel termination. Matching the load impedance to the characteristic impedance of the line once again results in the fastest response. This leads to the following conclusion.
To avoid potentially disastrous transmission line effects such as ringing or slow propagation delays, the transmission line should be terminated, either at the source (series termination), or at the destination (parallel termination) with a resistance equal to the characteristic impedance $Z_0$ of the transmission line.

### Example 4.11 Capacitive Termination

Loads in MOS digital circuits tend to be of a capacitive nature. One might wonder how this influences the transmission line behavior and when the load capacitance should be taken into account.

The characteristic impedance of the transmission line determines the current that can be supplied to charge capacitive load $C_L$. From the load’s point of view, the line behaves as a resistance with value $Z_0$. The transient response at the capacitor node, therefore, displays a time constant $Z_0 C_L$. This is illustrated in Figure 4.24, which shows the simulated transient response of a series-terminated transmission line with a characteristic impedance of 50 $\Omega$ loaded by a capacitance of 2 pF. The response shows how the output rises to its final value with a time-constant of 100 psec ($= 50 \Omega \times 2$ pF) after a delay equal to the time-of-flight of the line.

This asymptotic response causes some interesting artifacts. After $2 t_{\text{flight}}$, an unexpected voltage dip occurs at the source node that can be explained as follows. Upon reaching the destination node, the incident wave is reflected. This reflected wave also approaches its final value asymptotically. Since $V_{\text{dest}}$ equals 0 initially instead of the expected jump to 5 V, the reflection equals $-2.5$ V rather than the expected 2.5 V. This forces the transmission line temporarily to 0 V, as shown in the simulation. This effect gradually disappears as the output node converges to its final value.

The propagation delay of the line equals the sum of the time-of-flight of the line ($= 50$ psec) and the time it takes to charge the capacitance ($= 0.69 Z_0 C_L = 69$ psec). This is exactly what the simulation yields. In general, we can say that the capacitive load should only be considered in the analysis when its value is comparable to or larger than the total capacitance of the transmission line [Bakoglu90].

![Figure 4.24 Capacitively terminated transmission line: $R_S = 50 \Omega$, $R_L = \infty$, $C_L = 2$ pF, $Z_0 = 50 \Omega$, $t_{\text{flight}} = 50$ psec.](image)
Lossy Transmission Line

While board and module wires are thick and wide enough to be treated as lossless transmission lines, the same is not entirely true for on-chip interconnect where the resistance of the wire is an important factor. The lossy transmission-line model should be applied instead. Going into detail about the behavior of a lossy line would lead us to far astray. We therefore only discuss the effects of resistive loss on the transmission line behavior in a qualitative fashion.

The response of a lossy RLC line to a unit step combines wave propagation with a diffusive component. This is demonstrated in Figure 4.25, which plots the response of the RLC transmission line as a function of distance from the source. The step input still propagates as a wave through the line. However, the amplitude of this traveling wave is attenuated along the line:

\[
\frac{V_{\text{step}}(x)}{V_{\text{step}}(0)} = e^{-\frac{r}{2Z_0}x}
\]

The arrival of the wave is followed by a diffusive relaxation to the steady-state value at point \(x\). The farther it is from the source, the more the response resembles the behavior of a distributed RC line. In fact, the resistive effect becomes dominant, and the line behaves as a distributed RC line when \(R = rL, \text{the total resistance of the line} \gg 2Z_0\).

When \(R = 5Z_0\), only 8% of the original step reaches the end of the line. At that point, the line is more appropriately modeled as a distributed \(\text{RC}\) line.

Be aware that the actual wires on chips, boards, or substrates behave in a far more complex way than predicted by the above analysis. For instance, branches on wires, often called transmission line taps, cause extra reflections and can affect both signal shape and delay. Since the analysis of these effects is very involved, the only meaningful approach is to use computer analysis and simulation techniques. For a more extensive discussion of these effects, we would like to refer the reader to [Bakoglu90] and [Dally98].
Once again, we have to ask ourselves the question when it is appropriate to consider transmission line effects. From the above discussion, we can derive two important constraints:

- Transmission line effects should be considered when the rise or fall time of the input signal \((t_r, t_f)\) is smaller than the time-of-flight of the transmission line \((t_{\text{flight}})\).

This leads to the following rule of thumb, which determines when transmission line effects should be considered:

\[
t_r(t_f) < 2.5t_{\text{flight}} = 2.5 \frac{L}{V}
\]  

(4.36)

For on-chip wires with a maximum length of 1 cm, one should only worry about transmission line effects when \(t_r < 150\) psec. At the board level, where wires can reach a length of up to 50 cm, we should account for the delay of the transmission line when \(t_r < 8\) nsec. This condition is easily achieved with state-of-the-art processes and packaging technologies. Ignoring the inductive component of the propagation delay can easily result in overly optimistic delay predictions.

- Transmission line effects should only be considered when the total resistance of the wire is limited:

\[
R < 5Z_0
\]

(4.37)

If this is not the case, the distributed RC model is more appropriate. Both constraints can be summarized in the following set of bounds on the wire length:

\[
\frac{t_r}{2.5} \sqrt{\frac{1}{c}} < L < \frac{5}{\left(\frac{L}{\nu:\nu:\nu}\right)}
\]

(4.38)

- The transmission line is considered lossless when the total resistance is substantially smaller than the characteristic impedance, or

\[
R < \frac{Z_0}{2}
\]

(4.39)

**Example 4.12 When to Consider Transmission Line Effects**

Consider again our Al1 wire. Using the data from Example 4.4 and Eq. (4.28), we can approximate the value of \(Z_0\) for various wire widths:

- \(W = 0.1\ \mu\text{m}: c = 92\ \text{aF/\mu m}; Z_0 = 74\ \Omega\)
- \(W = 1.0\ \mu\text{m}: c = 110\ \text{aF/\mu m}; Z_0 = 60\ \Omega\)
- \(W = 10\ \mu\text{m}: c = 380\ \text{aF/\mu m}; Z_0 = 17\ \Omega\)
Using a wire with a width of $1\mu m$, we can derive the maximum length of the wire for which we should consider transmission line effects using Eq. (4.37):

$$L_{max} = \frac{5Z_0}{r} \approx \frac{5 \times 60\Omega}{0.075\Omega/\mu m} = 4000\mu m$$

From Eq. (4.36), we find a corresponding maximum rise (or fall) time of the input signal equal to

$$t_{r\max} = 2.5 \times (4000\mu m)/(15\text{ cm}/\text{sec}) = 67\text{ psec}$$

This is hard to accomplish in current technologies. For these wires, a lumped capacitance model is more appropriate. Transmission line effects are more plausible in wider wires. For a $10\mu m$ wide wire, we find a maximum length of $11.3\text{ mm}$, which corresponds to a maximum rise time of $188\text{ psec}$.

Assume now a Copper wire, implemented on level 5, with a characteristic impedance of $200\Omega$ and a resistance of $0.025\Omega/\mu m$. The resulting maximum wire length equals $40\text{ mm}$. Rise times smaller than $670\text{ psec}$ will cause transmission line effects to occur.

Be aware however that the values for $Z_0$, derived in this example, are only approximations. In actual designs, more complex expressions or empirical data should be used.

---

**Example 4.13 Simulation of Transmission Line Effects**

Show SPICE simulation

---

### 4.5 SPICE Wire Models

In previous sections, we have discussed the

#### 4.5.1 Distributed $rc$ Lines in SPICE

Because of the importance of the distributed $rc$-line in today's design, most circuit simulators have built-in distributed $rc$-models of high accuracy. For instance, the Berkeley SPICE3 simulator supports a uniform-distributed $rc$-line model (URC). This model approximates the $rc$-line as a network of lumped RC segments with internally generated nodes. Parameters include the length of the wire $L$ and (optionally) the number of segments used in the model.

**Example 4.14 SPICE3 URC Model**

A typical example of a SPICE3 instantiation of a distributed $rc$-line is shown below. $N1$ and $N2$ represent the terminal nodes of the line, while $N3$ is the node the capacitances are connected to. RPERL and CPERL stand for the resistance and capacitance per meter.

```
U1 N1=1 N2=2 N3=0 URCMOD L=50m N=6
.MODEL URCMOD URC(RPERL=75K CPERL=100pF)
```
If your simulator does not support a distributed \( rc \)-model, or if the computational complexity of these models slows down your simulation too much, you can construct a simple yet accurate model yourself by approximating the distributed \( rc \) by a lumped \( RC \) network with a limited number of elements. Figure 4.26 shows some of these approximations ordered along increasing precision and complexity. The accuracy of the model is determined by the number of stages. For instance, the error of the \( \pi 3 \) model is less than 3\%, which is generally sufficient.

![Simulation models for distributed RC line.](image)

**Figure 4.26** Simulation models for distributed \( RC \) line.

### 4.5.2 Transmission Line Models in SPICE

SPICE supports a lossless transmission line model. The line characteristics are defined by the characteristic impedance \( Z_0 \), while the length of the line can be defined in either of two forms. A first approach is to directly define the transmission delay \( TD \), which is equivalent to the time-of-flight. Alternatively, a frequency \( F \) may be given together with \( NL \), the dimensionless, normalized electrical length of the transmission line, which is measured with respect to the wavelength in the line at the frequency \( F \). The following relation is valid.

\[
NL = F \cdot TD
\]  
(4.40)
Section 4.6 Perspective: A Look into the Future

No lossy transmission line model is currently provided. When necessary, loss can be added by breaking up a long transmission line into shorter sections and adding a small series resistance in each section to model the transmission line loss. Be careful when using this approximation. First of all, the accuracy is still limited. Secondly, the simulation speed might be severely affected, since SPICE chooses a time step that is less than or equal to half of the value of $TD$. For small transmission lines, this time step might be much smaller than what is needed for transistor analysis.

4.6 Perspective: A Look into the Future

Similar to the approach we followed for the MOS transistor, it is worthwhile to explore how the wire parameters will evolve with further scaling of the technology. As transistor dimensions are reduced, the interconnect dimensions must also be reduced to take full advantage of the scaling process.

A straightforward approach is to scale all dimensions of the wire by the same factor $S$ as the transistors (ideal scaling). This might not be possible for at least one dimension of the wire, being the length. It can be surmised that the length of local interconnections — wires that connect closely grouped transistors — scales in the same way as these transistors. On the other hand, global interconnections, that provide the connectivity between large modules and the input-output circuitry, display a different scaling behavior. Examples of such wires are clock signals, and data and instruction buses. Figure 4.27 contains a histogram showing the distribution of the wire lengths in an actual microprocessor design, containing approximately 90,000 gates [Davis98]. While most of the wires tend to be only a couple of gate pitches long, a substantial number of them are much longer and can reach lengths up to 500 gate pitches.

The average length of these long wires is proportional to the die size (or complexity) of the circuit. An interesting trend is that while transistor dimensions have continued to shrink over the past decades, the chip sizes have gradually increased. In fact, the size of the typical die (which is the square root of the die area) is increasing by 6% per year, doubling about every decade. Chips have scaled from 2 mm × 2 mm in the early 1960s to approximately 2 cm × 2 cm in 2000. They are projected to reach 4 cm on the side by 2010!

Figure 4.27 Distribution of wire lengths in an advanced microprocessor as a function of the gate pitch.
This argues that when studying the scaling behavior of the wire length, we have to differentiate between local and global wires. In our subsequent analysis, we will therefore consider three models: local wires ($S_L > 1$), constant length wires ($S_L = 1$), and global wires ($S_L = S_C < 1$).

Assume now that all other wire dimensions of the interconnect structure ($W, H, t$) scale with the technology factor $S$. This leads to the scaling behavior illustrated in Table 4.8. Be aware that this is only a first-order analysis, intended to look at overall trends. Effects such as fringing capacitance are ignored, and breakthroughs in semiconductor technology such as new interconnect and dielectric materials are also not considered.

The eye-catching conclusion of this exercise is that scaling of the technology does not reduce wire delay (as personified by the $RC$ time-constant). A constant delay is predicted for local wires, while the delay of the global wires goes up with 50% per year (for $S = 1.15$ and $S_C = 0.94$). This is in great contrast with the gate delay, which reduces from year to year. This explains why wire delays are starting to play a predominant role in today's digital integrated circuit design.

The ideal scaling approach clearly has problems, as it causes a rapid increase in wire resistance. This explains why other interconnect scaling techniques are attractive. One option is to scale the wire thickness at a different rate. The “constant resistance” model of Table 4.9 explores the impact of not scaling the wire thickness at all. While this approach seemingly has a positive impact on the performance, it causes the fringing and interwire capacitance components to come to the foreground. We therefore introduce an extra capacitance scaling factor $\varepsilon_c (> 1)$, that captures the increasingly horizontal nature of the capacitance when wire widths and pitches are shrunk while the height is kept constant.

### Table 4.8  Ideal Scaling of Wire Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Relation</th>
<th>Local Wire</th>
<th>Constant Length</th>
<th>Global Wire</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W, H, t$</td>
<td>$1/S$</td>
<td>$1/S$</td>
<td>$1/S$</td>
<td></td>
</tr>
<tr>
<td>$L$</td>
<td>$1/S$</td>
<td>$1/S$</td>
<td>$1/S$</td>
<td></td>
</tr>
<tr>
<td>$C$</td>
<td>$LW/t$</td>
<td>$1/S$</td>
<td>$1$</td>
<td>$1/S_C$</td>
</tr>
<tr>
<td>$R$</td>
<td>$L/WH$</td>
<td>$S$</td>
<td>$S^2$</td>
<td>$S^2/S_C$</td>
</tr>
<tr>
<td>$CR$</td>
<td>$L^2/Ht$</td>
<td>$1$</td>
<td>$S^2$</td>
<td>$S^2/S_C^2$</td>
</tr>
</tbody>
</table>

### Table 4.9  “Constant Resistance” Scaling of Wire Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Relation</th>
<th>Local Wire</th>
<th>Constant Length</th>
<th>Global Wire</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W, t$</td>
<td>$1/S$</td>
<td>$1/S$</td>
<td>$1/S$</td>
<td></td>
</tr>
<tr>
<td>$H$</td>
<td>$1$</td>
<td>$1$</td>
<td>$1$</td>
<td></td>
</tr>
<tr>
<td>$L$</td>
<td>$1/S$</td>
<td>$1$</td>
<td>$1/S_C$</td>
<td></td>
</tr>
<tr>
<td>$C$</td>
<td>$\varepsilon_LW/t$</td>
<td>$\varepsilon_c$</td>
<td>$\varepsilon_c$</td>
<td>$\varepsilon_c/S_C$</td>
</tr>
</tbody>
</table>
Section 4.7 Summary

This scaling scenario offers a slightly more optimistic perspective, assuming of course that $\varepsilon_c < S$. Yet, delay is bound to increase substantially for intermediate and long wires, independent of the scaling scenario. To keep these delays from becoming excessive, interconnect technology has to be drastically improved. One option is to use better interconnect (Cu) and insulation materials (polymers and air). The other option is to differentiate between local and global wires. In the former, density and low-capacitance are crucial, while keeping the resistance under control is crucial in the latter. To address these conflicting demands, modern interconnect topologies combine a dense and thin wiring grid at the lower metal layers with fat, widely spaced wires at the higher levels, as is illustrated in Figure 4.28. Even with these advances, it is obvious that interconnect will play a dominant role in both high-performance and low-energy circuits for years to come.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Relation</th>
<th>Local Wire</th>
<th>Constant Length</th>
<th>Global Wire</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>$L/WH$</td>
<td>1</td>
<td>$S$</td>
<td>$S/S_c$</td>
</tr>
<tr>
<td>$CR$</td>
<td>$L^2/Ht$</td>
<td>$\varepsilon_c/S$</td>
<td>$\varepsilon_c S$</td>
<td>$\varepsilon_c S/S_c^2$</td>
</tr>
</tbody>
</table>

4.7 Summary

This chapter has presented a careful and in-depth analysis of the role and the behavior of the interconnect wire in modern semiconductor technology. The main goal is to identify the dominant parameters that set the values of the wire parasitics (being capacitance, resistance, and inductance), and to present adequate wire models that will aid us in the further analysis and optimization of complex digital circuits.

Figure 4.28 Interconnect hierarchy of 0.25 µm CMOS process, drawn to scale.
4.8 To Probe Further

Interconnect and its modeling is a hotly debated topic, that receives major attention in journals and conferences. A number of textbooks and reprint volumes have been published. [Bakoglu90], [Tewksbury94], and [Dally98] present an in-depth coverage of interconnect issues, and are a valuable resource for further browsing.

REFERENCES

Section 4.9  Exercises and Design Problems


4.9  Exercises and Design Problems
CHAPTER 5

THE CMOS INVERTER

Quantification of integrity, performance, and energy metrics of an inverter
Optimization of an inverter design

5.1 Introduction

5.2 The Static CMOS Inverter — An Intuitive Perspective

5.3 Evaluating the Robustness of the CMOS Inverter: The Static Behavior
  5.3.1 Switching Threshold
  5.3.2 Noise Margins
  5.3.3 Robustness Revisited

5.4 Performance of CMOS Inverter: The Dynamic Behavior
  5.4.1 Computing the Capacitances
  5.4.2 Propagation Delay: First-Order Analysis
  5.4.3 Propagation Delay Revisited

5.5 Power, Energy, and Energy-Delay
  5.5.1 Dynamic Power Consumption
  5.5.2 Static Consumption
  5.5.3 Putting It All Together
  5.5.4 Analyzing Power Consumption Using SPICE

5.6 Perspective: Technology Scaling and its Impact on the Inverter Metrics
Section 5.1 Introduction

5.1 Introduction

The inverter is truly the nucleus of all digital designs. Once its operation and properties are clearly understood, designing more intricate structures such as NAND gates, adders, multipliers, and microprocessors is greatly simplified. The electrical behavior of these complex circuits can be almost completely derived by extrapolating the results obtained for inverters. The analysis of inverters can be extended to explain the behavior of more complex gates such as NAND, NOR, or XOR, which in turn form the building blocks for modules such as multipliers and processors.

In this chapter, we focus on one single incarnation of the inverter gate, being the static CMOS inverter — or the CMOS inverter, in short. This is certainly the most popular at present, and therefore deserves our special attention. We analyze the gate with respect to the different design metrics that were outlined in Chapter 1:

- **cost**, expressed by the complexity and area
- **integrity and robustness**, expressed by the static (or steady-state) behavior
- **performance**, determined by the dynamic (or transient) response
- **energy efficiency**, set by the energy and power consumption

From this analysis arises a model of the gate that will help us to identify the parameters of the gate and to choose their values so that the resulting design meets desired specifications. While each of these parameters can be easily quantified for a given technology, we also discuss how they are affected by **scaling of the technology**.

While this Chapter focuses uniquely on the CMOS inverter, we will see in the following Chapter that the same methodology also applies to other gate topologies.

5.2 The Static CMOS Inverter — An Intuitive Perspective

Figure 5.1 shows the circuit diagram of a static CMOS inverter. Its operation is readily understood with the aid of the simple switch model of the MOS transistor, introduced in Chapter 3 (Figure 3.25): the transistor is nothing more than a switch with an infinite off-resistance (for $|V_{GS}| < |V_T|$), and a finite on-resistance (for $|V_{GS}| > |V_T|$). This leads to the

![Figure 5.1 Static CMOS inverter. $V_{DD}$ stands for the supply voltage.](image)
following interpretation of the inverter. When \( V_{\text{in}} \) is high and equal to \( V_{DD} \), the NMOS transistor is on, while the PMOS is off. This yields the equivalent circuit of Figure 5.2a. A direct path exists between \( V_{\text{out}} \) and the ground node, resulting in a steady-state value of 0 V. On the other hand, when the input voltage is low (0 V), NMOS and PMOS transistors are off and on, respectively. The equivalent circuit of Figure 5.2b shows that a path exists between \( V_{DD} \) and \( V_{\text{out}} \) yielding a high output voltage. The gate clearly functions as an inverter.

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

- The high and low output levels equal \( V_{DD} \) and \( GND \), respectively; in other words, the voltage swing is equal to the supply voltage. This results in high noise margins.
- The logic levels are not dependent upon the relative device sizes, so that the transistors can be minimum size. Gates with this property are called ratioless. This is in contrast with ratioed logic, where logic levels are determined by the relative dimensions of the composing transistors.
- In steady state, there always exists a path with finite resistance between the output and either \( V_{DD} \) or \( GND \). A well-designed CMOS inverter, therefore, has a low output impedance, which makes it less sensitive to noise and disturbances. Typical values of the output resistance are in kΩ range.
- The input resistance of the CMOS inverter is extremely high, as the gate of an MOS transistor is a virtually perfect insulator and draws no dc input current. Since the input node of the inverter only connects to transistor gates, the steady-state input current is nearly zero. A single inverter can theoretically drive an infinite number of gates (or have an infinite fan-out) and still be functionally operational; however, increasing the fan-out also increases the propagation delay, as will become clear below. So, although fan-out does not have any effect on the steady-state behavior, it degrades the transient response.
Section 5.2  The Static CMOS Inverter — An Intuitive Perspective

- No direct path exists between the supply and ground rails under steady-state operating conditions (this is, when the input and outputs remain constant). The absence of current flow (ignoring leakage currents) means that the gate does not consume any static power.

SIDELINE: The above observation, while seemingly obvious, is of crucial importance, and is one of the primary reasons CMOS is the digital technology of choice at present. The situation was very different in the 1970s and early 1980s. All early microprocessors, such as the Intel 4004, were implemented in a pure NMOS technology. The lack of complementary devices (such as the NMOS and PMOS transistor) in such a technology makes the realization of inverters with zero static power non-trivial. The resulting static power consumption puts a firm upper bound on the number of gates that can be integrated on a single die; hence the forced move to CMOS in the 1980s, when scaling of the technology allowed for higher integration densities.

The nature and the form of the voltage-transfer characteristic (VTC) can be graphically deduced by superimposing the current characteristics of the NMOS and the PMOS devices. Such a graphical construction is traditionally called a load-line plot. It requires that the I-V curves of the NMOS and PMOS devices are transformed onto a common coordinate set. We have selected the input voltage \( V_{\text{in}} \), the output voltage \( V_{\text{out}} \) and the NMOS drain current \( I_{\text{DN}} \) as the variables of choice. The PMOS I-V relations can be translated into this variable space by the following relations (the subscripts \( n \) and \( p \) denote the NMOS and PMOS devices, respectively):

\[
\begin{align*}
I_{\text{DP}} &= -I_{\text{DS}} \\
V_{\text{GS}} &= V_{\text{in}}; \quad V_{\text{GSP}} = V_{\text{in}} - V_{\text{DD}} \\
V_{\text{DS}} &= V_{\text{out}}; \quad V_{\text{DSP}} = V_{\text{out}} - V_{\text{DD}}
\end{align*}
\] (5.1)

The load-line curves of the PMOS device are obtained by a mirroring around the x-axis and a horizontal shift over \( V_{\text{DD}} \). This procedure is outlined in Figure 5.3, where the subsequent steps to adjust the original PMOS I-V curves to the common coordinate set \( V_{\text{in}} \), \( V_{\text{out}} \), and \( I_{\text{DN}} \) are illustrated.

![Figure 5.3 Transforming PMOS I-V characteristic to a common coordinate set (assuming VDD = 2.5 V).](image-url)
The resulting load lines are plotted in Figure 5.4. For a dc operating points to be valid, the currents through the NMOS and PMOS devices must be equal. Graphically, this means that the dc points must be located at the intersection of corresponding load lines. A number of those points (for $V_{in}=0, 0.5, 1, 1.5, 2, \text{ and } 2.5$ V) are marked on the graph. As can be observed, all operating points are located either at the high or low output levels. The VTC of the inverter hence exhibits a very narrow transition zone. This results from the high gain during the switching transient, when both NMOS and PMOS are simultaneously on, and in saturation. In that operation region, a small change in the input voltage results in a large output variation. All these observations translate into the VTC of Figure 5.5.

The resulting load lines are plotted in Figure 5.4. For a dc operating points to be valid, the currents through the NMOS and PMOS devices must be equal. Graphically, this means that the dc points must be located at the intersection of corresponding load lines. A number of those points (for $V_{in}=0, 0.5, 1, 1.5, 2, \text{ and } 2.5$ V) are marked on the graph. As can be observed, all operating points are located either at the high or low output levels. The VTC of the inverter hence exhibits a very narrow transition zone. This results from the high gain during the switching transient, when both NMOS and PMOS are simultaneously on, and in saturation. In that operation region, a small change in the input voltage results in a large output variation. All these observations translate into the VTC of Figure 5.5.

Before going into the analytical details of the operation of the CMOS inverter, a qualitative analysis of the transient behavior of the gate is appropriate as well. This response is dominated mainly by the output capacitance of the gate, $C_L$, which is com-
Section 5.3 Evaluating the Robustness of the CMOS Inverter: The Static Behavior

The static behavior of the CMOS inverter is primarily determined by the capacitances involved: the drain diffusion capacitances of the NMOS and PMOS transistors, the capacitance of the connecting wires, and the input capacitance of the fan-out gates. Assuming temporarily that the transistors switch instantaneously, we can get an approximate idea of the transient response by using the simplified switch model again (Figure 5.6). Let us consider the low-to-high transition first (Figure 5.6a). The gate response time is simply determined by the time it takes to charge the capacitor $C_L$ through the resistor $R_p$. In Example 4.5, we learned that the propagation delay of such a network is proportional to the time-constant $R_pC_L$. Hence, a fast gate is built either by keeping the output capacitance small or by decreasing the on-resistance of the transistor. The latter is achieved by increasing the $W/L$ ratio of the device. Similar considerations are valid for the high-to-low transition (Figure 5.6b), which is dominated by the $R_nC_L$ time-constant. The reader should be aware that the on-resistance of the NMOS and PMOS transistor is not constant, but is a nonlinear function of the voltage across the transistor. This complicates the exact determination of the propagation delay. An in-depth analysis of how to analyze and optimize the performance of the static CMOS inverter is offered in Section 5.4.

5.3 Evaluating the Robustness of the CMOS Inverter: The Static Behavior

In the qualitative discussion above, the overall shape of the voltage-transfer characteristic of the static CMOS inverter was derived, as were the values of $V_{OH}$ and $V_{OL}$ ($V_{DD}$ and $GND$, respectively). It remains to determine the precise values of $V_M$, $V_{IH}$, and $V_{IL}$ as well as the noise margins.

5.3.1 Switching Threshold

The switching threshold, $V_M$, is defined as the point where $V_{in} = V_{out}$. Its value can be obtained graphically from the intersection of the VTC with the line given by $V_{in} = V_{out}$ (see Figure 5.5). In this region, both PMOS and NMOS are always saturated, since $V_{DS} = V_{GS}$. An analytical expression for $V_M$ is obtained by equating the currents through the tran-
sistors. We solve the case where the supply voltage is high so that the devices can be assumed to be velocity-saturated (or $V_{\text{DSAT}} < V_M - V_T$). We furthermore ignore the channel-length modulation effects.

\[ k_n V_{\text{DSATn}} \left( V_M - V_{Tn} - \frac{V_{\text{DSATn}}}{2} \right) + k_p V_{\text{DSATp}} \left( V_M - V_{DD} - V_{Tp} - \frac{V_{\text{DSATp}}}{2} \right) = 0 \]  

Solving for $V_M$ yields

\[ V_M = \left( V_{Tn} + \frac{V_{\text{DSATn}}}{2} \right) + r \left( V_{DD} + V_{Tp} + \frac{V_{\text{DSATp}}}{2} \right) \quad \text{with} \quad r = \frac{k_p V_{\text{DSATp}}}{k_n V_{\text{DSATn}}} = \frac{u_{\text{satp}} W_p}{u_{\text{satn}} W_n} \]  

assuming identical oxide thicknesses for PMOS and NMOS transistors. For large values of $V_{DD}$ (compared to threshold and saturation voltages), Eq. (5.3) can be simplified:

\[ V_M \approx \frac{r V_{DD}}{1 + r} \]  

Eq. (5.4) states that the switching threshold is set by the ratio $r$, which compares the relative driving strengths of the PMOS and NMOS transistors. It is generally considered to be desirable for $V_M$ to be located around the middle of the available voltage swing (or at $V_{DD}/2$), since this results in comparable values for the low and high noise margins. This requires $r$ to be approximately 1, which is equivalent to sizing the PMOS device so that $(W/L)_p = (W/L)_n \times (V_{\text{DSATp}}' k_p'/k_n$). To move $V_M$ upwards, a larger value of $r$ is required, which means making the PMOS wider. Increasing the strength of the NMOS, on the other hand, moves the switching threshold closer to GND.

From Eq. (5.2), we can derive the required ratio of PMOS versus NMOS transistor sizes such that the switching threshold is set to a desired value $V_M$. When using this expression, please make sure that the assumption that both devices are velocity-saturated still holds for the chosen operation point.

\[ \frac{W/L}_p = \frac{k_n' V_{\text{DSATn}} (V_M - V_{Tn} - \frac{V_{\text{DSATn}}}{2})}{k_p' V_{\text{DSATp}} (V_{DD} - V_M + V_{Tp} + \frac{V_{\text{DSATp}}}{2})} \]  

Problem 5.1 Inverter switching threshold for long-channel devices, or low supply-voltages.

The above expressions were derived under the assumption that the transistors are velocity-saturated. When the PMOS and NMOS are long-channel devices, or when the supply voltage is low, velocity saturation does not occur ($V_M - V_T < V_{\text{DSAT}}$). Under these circumstances, Eq. (5.6) holds for $V_M$. Derive.

\[ V_M = \frac{V_{Tn} + r(V_{DD} + V_{Tp})}{1 + r} \quad \text{with} \quad r = \sqrt{\frac{k_p}{k_n}} \]
When designing static CMOS circuits, it is advisable to balance the driving strengths of the transistors by making the PMOS section wider than the NMOS section, if one wants to maximize the noise margins and obtain symmetrical characteristics. The required ratio is given by Eq. (5.5).

Example 5.1 Switching threshold of CMOS inverter

We derive the sizes of PMOS and NMOS transistors such that the switching threshold of a CMOS inverter, implemented in our generic 0.25 $\mu$m CMOS process, is located in the middle between the supply rails. We use the process parameters presented in Example 3.7, and assume a supply voltage of 2.5 V. The minimum size device has a width/length ratio of 1.5. With the aid of Eq. (5.5), we find:

$$
\frac{(W/L)_{p}}{(W/L)_{n}} = \frac{115 \times 10^{-6}}{30 \times 10^{-6}} \times \frac{0.63}{1.0} \times \frac{(1.25 - 0.43 - 0.63/2)}{(1.25 - 0.4 - 1.0/2)} = 3.5
$$

Figure 5.7 plots the values of switching threshold as a function of the PMOS/NMOS ratio, as obtained by circuit simulation. The simulated PMOS/NMOS ratio of 3.4 for a 1.25 V switching threshold confirms the value predicted by Eq. (5.5).

An analysis of the curve of Figure 5.7 produces some interesting observations:

1. $V_M$ is relatively insensitive to variations in the device ratio. This means that small variations of the ratio (e.g., making it 3 or 2.5) do not disturb the transfer characteristic that much. It is therefore an accepted practice in industrial designs to set the width of the PMOS transistor to values smaller than those required for exact symmetry. For the above example, setting the ratio to 3, 2.5, and 2 yields switching thresholds of 1.22 V, 1.18 V, and 1.13 V, respectively.

Figure 5.7 Simulated inverter switching threshold versus PMOS/NMOS ratio (0.25 $\mu$m CMOS, $V_{DD} = 2.5$ V)
2. The effect of changing the $W_p/W_n$ ratio is to shift the transient region of the VTC. Increasing the width of the PMOS or the NMOS moves $V_M$ towards $V_{DD}$ or $GND$ respectively. This property can be very useful, as asymmetrical transfer characteristics are actually desirable in some designs. This is demonstrated by the example of Figure 5.8. The incoming signal $V_{in}$ has a very noisy zero value. Passing this signal through a symmetrical inverter would lead to erroneous values (Figure 5.8a). This can be addressed by raising the threshold of the inverter, which results in a correct response (Figure 5.8b). Further in the text, we will see other circuit instances where inverters with asymmetrical switching thresholds are desirable.

Changing the switching threshold by a considerable amount is however not easy, especially when the ratio of supply voltage to transistor threshold is relatively small ($2.5/0.4 = 6$ for our particular example). To move the threshold to 1.5 V requires a transistor ratio of 11, and further increases are prohibitively expensive. Observe that Figure 5.7 is plotted in a semilog format.

5.3.2 Noise Margins

By definition, $V_{IH}$ and $V_{IL}$ are the operational points of the inverter where $\frac{dV_{out}}{dV_{in}} = -1$. In the terminology of the analog circuit designer, these are the points where the gain $g$ of the amplifier, formed by the inverter, is equal to −1. While it is indeed possible to derive analytical expressions for $V_{IH}$ and $V_{IL}$, these tend to be unwieldy and provide little insight into what parameters are instrumental in setting the noise margins.

A simpler approach is to use a piecewise linear approximation for the VTC, as shown in Figure 5.9. The transition region is approximated by a straight line, the gain of which equals the gain $g$ at the switching threshold $V_M$. The crossover with the $V_{OH}$ and the $V_{OL}$ lines is used to define $V_{IH}$ and $V_{IL}$ points. The error introduced is small and well...
within the range of what is required for an initial design. This approach yields the follow-

\[ V_{IH} - V_{IL} = \frac{(V_{OH} - V_{OL})}{g} = -\frac{V_{DD}}{g} \]

\[ V_{IH} = V_{M} - \frac{V_{M}}{g} \quad V_{IL} = V_{M} + \frac{V_{DD} - V_{M}}{g} \]

(5.7)

These expressions make it increasingly clear that a high gain in the transition region is very desirable. In the extreme case of an infinite gain, the noise margins simplify to \( V_{OH} - V_{M} \) and \( V_{M} - V_{OL} \) for \( NM_H \) and \( NM_L \), respectively, and span the complete voltage swing.

Remains us to determine the midpoint gain of the static CMOS inverter. We assume once again that both PMOS and NMOS are velocity-saturated. It is apparent from Figure 5.4 that the gain is a strong function of the slopes of the currents in the saturation region. The channel-length modulation factor hence cannot be ignored in this analysis — doing so would lead to an infinite gain. The gain can now be derived by differentiating the current equation (5.8), valid around the switching threshold, with respect to \( V_{in} \).

\[ k_n V_{DSATn} \left( V_{in} - V_{Tn} - \frac{V_{DSATn}}{2} \right) \left( 1 + \lambda_n V_{out} \right) + k_p V_{DSATp} \left( V_{in} - V_{DD} - V_{Tp} - \frac{V_{DSATp}}{2} \right) \left( 1 + \lambda_p V_{out} - \lambda_p V_{DD} \right) = 0 \]

(5.8)

Differentiation and solving for \( dV_{out}/dV_{in} \) yields

\[ \frac{dV_{out}}{dV_{in}} = \frac{k_n V_{DSATn} \left( 1 + \lambda_n V_{out} \right) + k_p V_{DSATp} \left( 1 + \lambda_p V_{out} - \lambda_p V_{DD} \right)}{\lambda_n k_n V_{DSATn} \left( V_{in} - V_{Tn} - \frac{V_{DSATn}}{2} \right) + \lambda_p k_p V_{DSATp} \left( V_{in} - V_{DD} - V_{Tp} - \frac{V_{DSATp}}{2} \right)} \]

(5.9)

Ignoring some second-order terms, and setting \( V_{in} = V_{M} \) results in the gain expression,
with $I_D(V_M)$ the current flowing through the inverter for $V_{in} = V_M$. The gain is almost purely determined by technology parameters, especially the channel length modulation. It can only in a minor way be influenced by the designer through the choice of supply and switching threshold voltages.

**Example 5.2 Voltage transfer characteristic and noise margins of CMOS Inverter**

Assume an inverter in the generic 0.25 $\mu$m CMOS technology designed with a PMOS/NMOS ratio of 3.4 and with the NMOS transistor minimum size ($W = 0.375$ $\mu$m, $L = 0.25$ $\mu$m, $W/L = 1.5$). We first compute the gain at $V_M (= 1.25$ V),

$$g = \frac{1}{I_D(V_M)} \left( \frac{k_n V_{DSATn}}{\lambda_n} + \frac{k_p V_{DSATp}}{\lambda_p} \right) \approx \frac{(V_M - V_{TN} - V_{DSATn}/2)(\lambda_n - \lambda_p)}{(V_M - V_{TH} - V_{DSATp}/2)(\lambda_n - \lambda_p)}$$

(Eq. 5.10)

This yields the following values for $V_{IL}$, $V_{IH}$, $NM_L$, $NM_H$:

$$V_{IL} = 1.2 \text{ V}, \quad V_{IH} = 1.3 \text{ V}, \quad NM_L = NM_H = 1.2.$$  

Figure 5.10 plots the simulated VTC of the inverter, as well as its derivative, the gain. A close to ideal characteristic is obtained. The actual values of $V_{IL}$ and $V_{IH}$ are 1.03 V and 1.05 V, respectively, which leads to noise margins of 1.03 V and 1.05 V. These values are lower than those predicted for two reasons:

- Eq. (5.10) overestimates the gain. As observed in Figure 5.10b, the maximum gain (at $V_M$) equals only 17. This reduced gain would yield values for $V_{IL}$ and $V_{IH}$ of 1.17 V, and 1.33 V, respectively.

- The most important deviation is due to the piecewise linear approximation of the VTC, which is optimistic with respect to the actual noise margins.

The obtained expressions are however perfectly useful as first-order estimations as well as means of identifying the relevant parameters and their impact.

To conclude this example, we also extracted from simulations the output resistance of the inverter in the low- and high-output states. Low values of 2.4 k$\Omega$ and 3.3 k$\Omega$ were observed, respectively. The output resistance is a good measure of the sensitivity of the gate in respect to noise induced at the output, and is preferably as low as possible.

**SIDELINE:** Surprisingly (or not so surprisingly), the static CMOS inverter can also be used as an analog amplifier, as it has a fairly high gain in its transition region. This region is very narrow however, as is apparent in the graph of Figure 5.10b. It also receives poor marks on other amplifier properties such as supply noise rejection. Yet, this observation can be used to demonstrate one of the major differences between analog and digital design. Where the analog designer would bias the amplifier in the middle of the transient region, so that a maximum linearity is obtained, the digital designer will operate the
Section 5.3 Evaluating the Robustness of the CMOS Inverter: The Static Behavior

Problem 5.2 Inverter noise margins for long-channel devices

Derive expressions for the gain and noise margins assuming that PMOS and NMOS are long-channel devices (or that the supply voltage is low), so that velocity saturation does not occur.

5.3.3 Robustness Revisited

Device Variations

While we design a gate for nominal operation conditions and typical device parameters, we should always be aware that the actual operating temperature might vary over a large range, and that the device parameters after fabrication probably will deviate from the nominal values we used in our design optimization process. Fortunately, the dc-characteristics of the static CMOS inverter turn out to be rather insensitive to these variations, and the gate remains functional over a wide range of operating conditions. This already became apparent in Figure 5.7, which shows that variations in the device sizes have only a minor impact on the switching threshold of the inverter. To further confirm the assumed robustness of the gate, we have re-simulated the voltage transfer characteristic by replacing the nominal devices by their worst- or best-case incarnations. Two corner-cases are plotted in Figure 5.11: a better-than-expected NMOS combined with an inferior PMOS, and the opposite scenario. Comparing the resulting curves with the nominal response shows that the variations mostly cause a shift in the switching threshold, but that the operation of the
gate is by no means affected. This robust behavior that ensures functionality of the gate over a wide range of conditions has contributed in a big way to the popularity of the static CMOS gate.

Scaling the Supply Voltage

In Chapter 3, we observed that continuing technology scaling forces the supply voltages to reduce at rates similar to the device dimensions. At the same time, device threshold voltages are virtually kept constant. The reader probably wonders about the impact of this trend on the integrity parameters of the CMOS inverter. Do inverters keep on working when the voltages are scaled and are there potential limits to the supply scaling?

A first hint on what might happen was offered in Eq. (5.10), which indicates that the gain of the inverter in the transition region actually increases with a reduction of the supply voltage! Note that for a fixed transistor ratio $r$, $V_M$ is approximately proportional to $V_{DD}$. Plotting the (normalized) VTC for different supply voltages not only confirms this conjecture, but even shows that the inverter is well and alive for supply voltages close to the threshold voltage of the composing transistors (Figure 5.12a). At a voltage of 0.5 V — which is just 100 mV above the threshold of the transistors — the width of the transition region measures only 10% of the supply voltage (for a maximum gain of 35), while it widens to 17% for 2.5 V. So, given this improvement in dc characteristics, why do we not choose to operate all our digital circuits at these low supply voltages? Three important arguments come to mind:

- In the following sections, we will learn that reducing the supply voltage indiscriminately has a positive impact on the energy dissipation, but is absolutely detrimental to the performance on the gate.
- The dc-characteristic becomes increasingly sensitive to variations in the device parameters such as the transistor threshold, once supply voltages and intrinsic voltages become comparable.
- Scaling the supply voltage means reducing the signal swing. While this typically helps to reduce the internal noise in the system (such as caused by crosstalk), it makes the design more sensitive to external noise sources that do not scale.

\[ V_{in}(V) \quad V_{out}(V) \]

Figure 5.11 Impact of device variations on static CMOS inverter VTC. The “good” device has a smaller oxide thickness (-3nm), a smaller length (-25 nm), a higher width (+30 nm), and a smaller threshold (-60 mV). The opposite is true for the “bad” transistor.
Section 5.3 Evaluating the Robustness of the CMOS Inverter: The Static Behavior

To provide an insight into the question on potential limits to the voltage scaling, we have plotted in Figure 5.12b the voltage transfer characteristic of the same inverter for the even-lower supply voltages of 200 mV, 100 mV, and 50 mV (while keeping the transistor thresholds at the same level). Amazingly enough, we still obtain an inverter characteristic, this while the supply voltage is not even large enough to turn the transistors on! The explanation can be found in the sub-threshold operation of the transistors. The sub-threshold currents are sufficient to switch the gate between low and high levels, and provide enough gain to produce acceptable VTCs. The very low value of the switching currents ensures a very slow operation but this might be acceptable for some applications (such as watches, for example).

At around 100 mV, we start observing a major deterioration of the gate characteristic. $V_{OL}$ and $V_{OH}$ are no longer at the supply rails and the transition-region gain approaches 1. The latter turns out to be a fundamental show-stopper. To achieving sufficient gain for use in a digital circuit, it is necessary that the supply must be at least a couple times $\Phi_T = kT/q = 25 \text{ mV}$ at room temperature, the thermal voltage introduced in Chapter 3 [Swanson72]. It turns out that below this same voltage, thermal noise becomes an issue as well, potentially resulting in unreliable operation.

$$V_{DDmin} > 2\cdots 4 \frac{kT}{q}$$  \hspace{1cm} (5.11)

Eq. (5.11) presents a true lower bound on supply scaling. It suggests that the only way to get CMOS inverters to operate below 100 mV is to reduce the ambient temperature, or in other words to cool the circuit.

**Problem 5.3 Minimum supply voltage of CMOS inverter**

Once the supply voltage drops below the threshold voltage, the transistors operate the sub-threshold region, and display an exponential current-voltage relationship (as expressed in Eq. (3.40)). Derive an expression for the gain of the inverter under these circumstances.
(assume symmetrical NMOS and PMOS transistors, and a maximum gain at $V_M = V_{DD}/2$).

The resulting expression demonstrates that the minimum voltage is a function of the slope factor $n$ of the transistor.

$$g = -\frac{1}{n} \left( e^{V_{DD}/2\phi_T} - 1 \right)$$

(5.12)

According to this expression, the gain drops to -1 at $V_{DD} = 48$ mV (for $n = 1.5$ and $\phi_T = 25$ mV).

5.4 Performance of CMOS Inverter: The Dynamic Behavior

The qualitative analysis presented earlier concluded that the propagation delay of the CMOS inverter is determined by the time it takes to charge and discharge the load capacitor $C_L$ through the PMOS and NMOS transistors, respectively. This observation suggests that getting $C_L$ as small as possible is crucial to the realization of high-performance CMOS circuits. It is hence worthwhile to first study the major components of the load capacitance before embarking onto an in-depth analysis of the propagation delay of the gate. In addition to this detailed analysis, the section also presents a summary of techniques that a designer might use to optimize the performance of the inverter.

5.4.1 Computing the Capacitances

Manual analysis of MOS circuits where each capacitor is considered individually is virtually impossible and is exacerbated by the many nonlinear capacitances in the MOS transistor model. To make the analysis tractable, we assume that all capacitances are lumped together into one single capacitor $C_L$, located between $V_{out}$ and $GND$. Be aware that this is a considerable simplification of the actual situation, even in the case of a simple inverter.

![Figure 5.13](Parasitic capacitances, influencing the transient behavior of the cascaded inverter pair.)
Section 5.4 Performance of CMOS Inverter: The Dynamic Behavior

Figure 5.13 shows the schematic of a cascaded inverter pair. It includes all the capacitances influencing the transient response of node $V_{out}$. It is initially assumed that the input $V_{in}$ is driven by an ideal voltage source with zero rise and fall times. Accounting only for capacitances connected to the output node, $C_L$ breaks down into the following components.

**Gate-Drain Capacitance $C_{gd12}$**

M1 and M2 are either in cut-off or in the saturation mode during the first half (up to 50% point) of the output transient. Under these circumstances, the only contributions to $C_{gd12}$ are the overlap capacitances of both M1 and M2. The channel capacitance of the MOS transistors does not play a role here, as it is located either completely between gate and bulk (cut-off) or gate and source (saturation) (see Chapter 3).

The lumped capacitor model now requires that this floating gate-drain capacitor be replaced by a capacitance-to-ground. This is accomplished by taking the so-called Miller effect into account. During a low-high or high-low transition, the terminals of the gate-drain capacitor are moving in opposite directions (Figure 5.14). The voltage change over the floating capacitor is hence twice the actual output voltage swing. To present an identical load to the output node, the capacitance-to-ground must have a value that is twice as large as the floating capacitance.

We use the following equation for the gate-drain capacitors: $C_{gd} = 2 C_{GD0} W$ (with $C_{GD0}$ the overlap capacitance per unit width as used in the SPICE model). For an in-depth discussion of the Miller effect, please refer to textbooks such as Sedra and Smith ([Sedra87], p. 57).\(^1\)

The Miller effect discussed in this context is a simplified version of the general analog case. In a digital inverter, the large scale gain between input and output always equals -1.

\[ \Delta V_{out} = \Delta V_{in} \]

\[ V_{out} \rightarrow M1 \rightarrow C_{gd1} \rightarrow V_{in} \]

Figure 5.14 The Miller effect—A capacitor experiencing identical but opposite voltage swings at both its terminals can be replaced by a capacitor to ground, whose value is twice the original value.

**Diffusion Capacitances $C_{db1}$ and $C_{db2}$**

The capacitance between drain and bulk is due to the reverse-biased $pn$-junction. Such a capacitor is, unfortunately, quite nonlinear and depends heavily on the applied voltage. We argued in Chapter 3 that the best approach towards simplifying the analysis is to replace the nonlinear capacitor by a linear one with the same change in charge for the voltage range of interest. A multiplication factor $K_{eq}$ is introduced to relate the linearized capacitor to the value of the junction capacitance under zero-bias conditions.

\[ 1 \quad \text{The Miller effect discussed in this context is a simplified version of the general analog case. In a digital inverter, the large scale gain between input and output always equals -1.} \]
with \( C_{j0} \) the junction capacitance per unit area under zero-bias conditions. An expression for \( K_{eq} \) was derived in Eq. (3.11) and is repeated here for convenience

\[
K_{eq} = \frac{-\phi_0^m}{(V_{high} - V_{low})(1 - m)} [(\phi_0 - V_{high})^{1 - m} - (\phi_0 - V_{low})^{1 - m}]
\]

with \( \phi_0 \) the built-in junction potential and \( m \) the grading coefficient of the junction.

Example 5.3 \( K_{eq} \) for a 2.5 V CMOS Inverter

Consider the inverter of Figure 5.13 designed in the generic 0.25 \( \mu \)m CMOS technology. The relevant capacitance parameters for this process were summarized in Table 3.5.

Let us first analyze the NMOS transistor (\( C_{db1} \) in Figure 5.13). The propagation delay is defined by the time between the 50% transitions of the input and the output. For the CMOS inverter, this is the time-instance where \( V_{out} \) reaches 1.25 V, as the output voltage swing goes from rail to rail or equals 2.5 V. We, therefore, linearize the junction capacitance over the interval \( \{2.5 \text{ V}, 1.25 \text{ V}\} \) for the high-to-low transition, and \( \{0, 1.25 \text{ V}\} \) for the low-to-high transition.

During the high-to-low transition at the output, \( V_{out} \) initially equals 2.5 V. Because the bulk of the NMOS device is connected to GND, this translates into a reverse voltage of 2.5 V over the drain junction or \( V_{high} = -2.5 \text{ V} \). At the 50% point, \( V_{out} = 1.25 \text{ V} \) or \( V_{low} = -1.25 \text{ V} \). Evaluating Eq. (5.14) for the bottom plate and sidewall components of the diffusion capacitance yields

Bottom plate: \( K_{eq} (m = 0.5, \phi_0 = 0.9) = 0.57 \),
Sidewall: \( K_{eqsw} (m = 0.44, \phi_0 = 0.9) = 0.61 \)

During the low-to-high transition, \( V_{low} \) and \( V_{high} \) equal 0 V and –1.25 V, respectively, resulting in higher values for \( K_{eq} \).

Bottom plate: \( K_{eq} (m = 0.5, \phi_0 = 0.9) = 0.79 \),
Sidewall: \( K_{eqsw} (m = 0.44, \phi_0 = 0.9) = 0.81 \)

The PMOS transistor displays a reverse behavior, as its substrate is connected to 2.5 V. Hence, for the high-to-low transition (\( V_{low} = 0, V_{high} = -1.25 \text{ V} \)),

Bottom plate: \( K_{eq} (m = 0.48, \phi_0 = 0.9) = 0.79 \),
Sidewall: \( K_{eqsw} (m = 0.32, \phi_0 = 0.9) = 0.86 \)

and for the low-to-high transition (\( V_{low} = -1.25 \text{ V}, V_{high} = -2.5 \text{ V} \))

Bottom plate: \( K_{eq} (m = 0.48, \phi_0 = 0.9) = 0.59 \),
Sidewall: \( K_{eqsw} (m = 0.32, \phi_0 = 0.9) = 0.7 \)

Using this approach, the junction capacitance can be replaced by a linear component and treated as any other device capacitance. The result of the linearization is a minor distortion of the voltage waveforms. The logic delays are not significantly influenced by this simplification.
Wiring Capacitance $C_w$

The capacitance due to the wiring depends upon the length and width of the connecting wires, and is a function of the distance of the fanout from the driving gate and the number of fanout gates. As argued in Chapter 4, this component is growing in importance with the scaling of the technology.

Gate Capacitance of Fanout $C_{g3}$ and $C_{g4}$

We assume that the fanout capacitance equals the total gate capacitance of the loading gates M3 and M4. Hence,

$$C_{fanout} = C_{gate}(\text{NMOS}) + C_{gate}(\text{PMOS})$$

$$= (C_{GSOn} + C_{GDon} + W_nL_nC_{ox}) + (C_{GSOp} + C_{GDOp} + W_pL_pC_{ox}) \quad (5.15)$$

This expression simplifies the actual situation in two ways:

- It assumes that all components of the gate capacitance are connected between $V_{out}$ and GND (or $V_{DD}$), and ignores the Miller effect on the gate-drain capacitances. This has a relatively minor effect on the accuracy, since we can safely assume that the connecting gate does not switch before the 50% point is reached, and $V_{out2}$ therefore, remains constant in the interval of interest.

- A second approximation is that the channel capacitance of the connecting gate is constant over the interval of interest. This is not exactly the case as we discovered in Chapter 3. The total channel capacitance is a function of the operation mode of the device, and varies from approximately 1/3 of $WLC_{ox}$ (cut-off) over 2/3 $WLC_{ox}$ (saturation) to the full $WLC_{ox}$ (linear). During the first half of the transient, it may be assumed that one of the load devices is always in linear mode, while the other transistor evolves from the off-mode to saturation. Ignoring the capacitance variation results in a pessimistic estimation with an error of approximately 10%, which is acceptable for a first order analysis.

Example 5.4 Capacitances of a 0.25 $\mu$m CMOS Inverter

A minimum-size, symmetrical CMOS inverter has been designed in the 0.25 $\mu$m CMOS technology. The layout is shown in Figure 5.15. The supply voltage $V_{DD}$ is set to 2.5 V. From the layout, we derive the transistor sizes, diffusion areas, and perimeters. This data is summarized in Table 5.1. As an example, we will derive the drain area and perimeter for the NMOS transistor. The drain area is formed by the metal-diffusion contact, which has an area of $4 \times 4 \lambda^2$, and the rectangle between contact and gate, which has an area of $3 \times 1 \lambda^2$. This results in a total area of $19 \lambda^2$, or 0.30 $\mu$m$^2$ (as $\lambda = 0.125$ $\mu$m). The perimeter of the drain area is rather involved and consists of the following components (going counterclockwise): $5 + 4 + 4 + 1 + 1 = 15 \lambda$, or PD = $15 \times 0.125 = 1.875$ $\mu$m. Notice that the gate side of the drain perimeter is not included, as this is not considered a part of the side-wall. The drain area and perimeter of the PMOS transistor are derived similarly (the rectangular shape makes the exercise considerably simpler): $AD = 5 \times 9 \lambda^2 = 45 \lambda^2$, or 0.7 $\mu$m$^2$; $PD = 5 + 9 + 5 = 19 \lambda$, or 2.375 $\mu$m.
This physical information can be combined with the approximations derived above to come up with an estimation of $C_L$. The capacitor parameters for our generic process were summarized in Table 3.5, and repeated here for convenience:

- Overlap capacitance: $C_{GD0}(\text{NMOS}) = 0.31 \text{ fF/} \mu\text{m}$; $C_{GD0}(\text{PMOS}) = 0.27 \text{ fF/} \mu\text{m}$
- Bottom junction capacitance: $C_{J}(\text{NMOS}) = 2 \text{ fF/} \mu\text{m}^2$; $C_{J}(\text{PMOS}) = 1.9 \text{ fF/} \mu\text{m}^2$
- Side-wall junction capacitance: $C_{JSW}(\text{NMOS}) = 0.28 \text{ fF/} \mu\text{m}$; $C_{JSW}(\text{PMOS}) = 0.22 \text{ fF/} \mu\text{m}$
- Gate capacitance: $C_{gs}(\text{NMOS}) = C_{gs}(\text{PMOS}) = 6 \text{ fF/} \mu\text{m}^2$

Finally, we should also consider the capacitance contributed by the wire, connecting the gates and implemented in metal 1 and polysilicon. A layout extraction program typically...
will deliver us precise values for this parasitic capacitance. Inspection of the layout helps us to form a first-order estimate and yields that the metal-1 and poly silicon areas of the wire, that are not over active diffusion, equal $42 \lambda^2$ and $72 \lambda^2$, respectively. With the aid of the interconnect parameters of Table 4.2, we find the wire capacitance — observe that we ignore the fringing capacitance in this simple exercise. Due to the short length of the wire, this contribution is ignorable compared to the other parasitics.

$$C_{\text{wire}} = 42/8 \mu m^2 \times 30 \text{ aF/}\mu m^2 + 72/8 \mu m^2 \times 88 \text{ aF/}\mu m^2 = 0.12 \text{ fF}$$

Bringing all the components together results in Table 5.2. We use the values of $K_{eq}$ derived in Example 5.3 for the computation of the diffusion capacitances. Notice that the load capacitance is almost evenly split between its two major components: the intrinsic capacitance, composed of diffusion and overlap capacitances, and the extrinsic load capacitance, contributed by wire and connecting gate.

<table>
<thead>
<tr>
<th>Capacitor</th>
<th>Expression</th>
<th>Value (fF) (H→L)</th>
<th>Value (fF) (L→H)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{gd1}$</td>
<td>$2 \text{ CGD}_0 W_n$</td>
<td>0.23</td>
<td>0.23</td>
</tr>
<tr>
<td>$C_{gd2}$</td>
<td>$2 \text{ CGD}_0 W_p$</td>
<td>0.61</td>
<td>0.61</td>
</tr>
<tr>
<td>$C_{dh1}$</td>
<td>$K_{eqn} AD_n CJ + K_{eqprop} PD_n CJSW$</td>
<td>0.66</td>
<td>0.90</td>
</tr>
<tr>
<td>$C_{dh2}$</td>
<td>$K_{eqp} AD_p CJ + K_{eqprop} PD_p CJSW$</td>
<td>1.5</td>
<td>1.15</td>
</tr>
<tr>
<td>$C_{gd}$</td>
<td>$(\text{CGD}_0 + \text{CGSO}<em>1) W_n + C</em>{\text{ox}} L_n$</td>
<td>0.76</td>
<td>0.76</td>
</tr>
<tr>
<td>$C_{gd}$</td>
<td>$(\text{CGD}_0 + \text{CGSO}<em>1) W_p + C</em>{\text{ox}} L_p$</td>
<td>2.28</td>
<td>2.28</td>
</tr>
<tr>
<td>$C_w$</td>
<td>From Extraction</td>
<td>0.12</td>
<td>0.12</td>
</tr>
<tr>
<td>$C_L$</td>
<td>$\Sigma$</td>
<td>6.1</td>
<td>6.0</td>
</tr>
</tbody>
</table>

### 5.4.2 Propagation Delay: First-Order Analysis

One way to compute the propagation delay of the inverter is to integrate the capacitor (dis)charge current. This results in the expression of Eq. (5.16).

$$t_p = \int_{v_1}^{v_2} \frac{C_L(v)}{i(v)} dv \quad (5.16)$$

with $i$ the (dis)charging current, $v$ the voltage over the capacitor, and $v_1$ and $v_2$ the initial and final voltage. An exact computation of this equation is untractable, as both $C_L(v)$ and $i(v)$ are nonlinear functions of $v$. We rather fall back to the simplified switch-model of the inverter introduced in Figure 5.6 to derive a reasonable approximation of the propagation delay adequate for manual analysis. The voltage-dependencies of the on-resistance and the load capacitor are addressed by replacing both by a constant linear element with a value averaged over the interval of interest. The preceding section derived precisely this value.
for the load capacitance. An expression for the average on-resistance of the MOS transis-
tor was already derived in Example 3.8, and is repeated here for convenience.

\[
R_{eq} = \frac{1}{\frac{V_{DD}}{2}} \int_{\frac{V_{DD}}{2}}^{V} \frac{V}{I_{DSAT}(1 + \lambda V)} dV = \frac{3}{4} \frac{V_{DD}}{4I_{DSAT}} \left( 1 - \frac{7}{9} \frac{V_{DD}}{V_{DSAT}} \right)
\]  
(5.17)

with \( I_{DSAT} = k' \frac{W}{L} (V_{DD} - V_t) \frac{V_{DSAT}}{2} \)

Deriving the propagation delay of the resulting circuit is now straightforward, and is
nothing more than the analysis of a first-order linear \( RC \)-network, identical to the exercise
of Example 4.5. There, we learned that the propagation delay of such a network for a volt-
age step at the input is proportional to the time-constant of the network, formed by pull-
down resistor and load capacitance. Hence,

\[
t_{pHL} = \ln(2) R_{eqn} C_L = 0.69 R_{eqn} C_L
\]  
(5.18)

Similarly, we can obtain the propagation delay for the low-to-high transition,

\[
t_{pLH} = 0.69 R_{eqp} C_L
\]  
(5.19)

with \( R_{eqp} \) the equivalent on-resistance of the PMOS transistor over the interval of interest.
This analysis assumes that the equivalent load-capacitance is identical for both the high-
to-low and low-to-high transitions. This has been shown to be approximately the case in
the example of the previous section. The overall propagation delay of the inverter is
defined as the average of the two values, or

\[
t_p = \frac{t_{pHL} + t_{pLH}}{2} = 0.69 C_L \left( \frac{R_{eqn} + R_{eqp}}{2} \right)
\]  
(5.20)

Very often, it is desirable for a gate to have identical propagation delays for both rising
and falling inputs. This condition can be achieved by making the on-resistance of the
NMOS and PMOS approximately equal. Remember that this condition is identical to the
requirement for a symmetrical VTC.

Example 5.5 Propagation Delay of a 0.25 \( \mu \)m CMOS Inverter

To derive the propagation delays of the CMOS inverter of Figure 5.15, we make use of Eq.
(5.18) and Eq. (5.19). The load capacitance \( C_L \) was already computed in Example 5.4, while
the equivalent on-resistances of the transistors for the generic 0.25 \( \mu \)m CMOS process were
derived in Table 3.3. For a supply voltage of 2.5 V, the normalized on-resistances of NMOS
and PMOS transistors equal 13 k\( \Omega \) and 31 k\( \Omega \), respectively. From the layout, we determine
the (W/L) ratios of the transistors to be 1.5 for the NMOS , and 4.5 for the PMOS. We
assume that the difference between drawn and effective dimensions is small enough to be
ignorable. This leads to the following values for the delays:
The accuracy of this analysis is checked by performing a SPICE transient simulation on the circuit schematic, extracted from the layout of Figure 5.15. The computed transient response of the circuit is plotted in Figure 5.16, and determines the propagation delays to be 39.9 psec and 31.7 for the HL and LH transitions, respectively. The manual results are good considering the many simplifications made during their derivation. Notice especially the overshoots on the simulated output signals. These are caused by the gate-drain capacitances of the inverter transistors, which couple the steep voltage step at the input node directly to the output before the transistors can even start to react to the changes at the input. These overshoots clearly have a negative impact on the performance of the gate, and explain why the simulated delays are larger than the estimations.

\[ t_{pHL} = 0.69 \times \left( \frac{13k\Omega}{1.5} \right) \times 6.1fF = 36 \text{ psec} \]
\[ t_{pLH} = 0.69 \times \left( \frac{31k\Omega}{4.5} \right) \times 6.0fF = 29 \text{ psec} \]
and
\[ t_p = \left( \frac{36 + 29}{2} \right) = 32.5 \text{ psec} \]

The accuracy of this analysis is checked by performing a SPICE transient simulation on the circuit schematic, extracted from the layout of Figure 5.15. The computed transient response of the circuit is plotted in Figure 5.16, and determines the propagation delays to be 39.9 psec and 31.7 for the HL and LH transitions, respectively. The manual results are good considering the many simplifications made during their derivation. Notice especially the overshoots on the simulated output signals. These are caused by the gate-drain capacitances of the inverter transistors, which couple the steep voltage step at the input node directly to the output before the transistors can even start to react to the changes at the input. These overshoots clearly have a negative impact on the performance of the gate, and explain why the simulated delays are larger than the estimations.

**WARNING:** This example might give the impression that manual analysis always leads to close approximations of the actual response. This is not necessarily the case. Large deviations can often be observed between first- and higher-order models. The purpose of the manual analysis is to get a basic insight in the behavior of the circuit and to determine the dominant parameters. A detailed simulation is indispensable when quantitative data is required. Consider the example above a stroke of good luck.
The obvious question a designer asks herself at this point is how she can manipulate and/or optimize the delay of a gate. To provide an answer to this question, it is necessary to make the parameters governing the delay explicit by expanding $R_{eq}$ in the delay equation. Combining Eq. (5.18) and Eq. (5.17), and assuming for the time being that the channel-length modulation factor $\lambda$ is ignorable, yields the following expression for $t_{pHL}$ (a similar analysis holds for $t_{pLH}$):

$$t_{pHL} = 0.693 \frac{C_L V_{DD}}{4 I_{DSATn}} = 0.52 \frac{C_L V_{DD}}{(W/L)n_k'V_{DSATn}(V_{DD} - V_{Tn} - V_{DSATn}/2)}$$  (5.21)

In the majority of designs, the supply voltage is chosen high enough so that $V_{DD} >> V_{Tn} + V_{DSATn}/2$. Under these conditions, the delay becomes virtually independent of the supply voltage (Eq. (5.22)). Observe that this is a first-order approximation, and that increasing the supply voltage yields an observable, albeit small, improvement in performance due to a non-zero channel-length modulation factor.

$$t_{pHL} \approx 0.52 \frac{C_L}{(W/L)n_k'V_{DSATn}}$$  (5.22)

This analysis is confirmed in Figure 5.17, which plots the propagation delay of the inverter as a function of the supply voltage. It comes as no surprise that this curve is virtually identical in shape to the one of Figure 3.27, which charts the equivalent on-resistance of the MOS transistor as a function of $V_{DD}$. While the delay is relatively insensitive to supply variations for higher values of $V_{DD}$, a sharp increase can be observed starting around $\approx 2V_T$. This operation region should clearly be avoided if achieving high performance is a premier design goal.

### Design Techniques

From the above, we deduce that the propagation delay of a gate can be minimized in the following ways:

Figure 5.17 Propagation delay of CMOS inverter as a function of supply voltage (normalized with respect to the delay at 2.5 V). The dots indicate the delay values predicted by Eq. (5.21). Observe that this equation is only valid when the devices are velocity-saturated. Hence, the deviation at low supply voltages.
Section 5.4 Performance of CMOS Inverter: The Dynamic Behavior

- **Reduce** $C_L$. Remember that three major factors contribute to the load capacitance: the internal diffusion capacitance of the gate itself, the interconnect capacitance, and the fan-out. Careful layout helps to reduce the diffusion and interconnect capacitances. **Good design practice requires keeping the drain diffusion areas as small as possible.**

- **Increase the W/L ratio of the transistors.** This is the most powerful and effective performance optimization tool in the hands of the designer. Proceed however with caution when applying this approach. Increasing the transistor size also raises the diffusion capacitance and hence $C_L$. In fact, once the intrinsic capacitance (i.e. the diffusion capacitance) starts to dominate the extrinsic load formed by wiring and fanout, increasing the gate size does no longer help in reducing the delay, and only makes the gate larger in area. This effect is called “self-loading”. In addition, wide transistors have a larger gate capacitance, which increases the fan-out factor of the driving gate and adversely affects its speed.

- **Increase $V_{DD}$.** As illustrated in Figure 5.17, the delay of a gate can be modulated by modifying the supply voltage. This flexibility allows the designer to trade-off energy dissipation for performance, as we will see in a later section. However, increasing the supply voltage above a certain level yields only very minimal improvement and hence should be avoided. Also, reliability concerns (oxide breakdown, hot-electron effects) enforce firm upper-bounds on the supply voltage in deep sub-micron processes.

**Example 5.6 Device sizing for performance**

Let us explore the performance improvement that can be obtained by device sizing in the design of Example 5.5. We assume the wire and fanout capacitance to be unaffected by the resizing. An insight in the potential improvement can be obtained by partitioning the load capacitance into an intrinsic (diffusion and miller) and an extrinsic (wiring and fanout) component, or

$$ C_L = C_{int} + C_{ext} = C_{int}(1 + \alpha) \quad (5.23) $$

with $\alpha$ the ratio between extrinsic and intrinsic capacitance. Widening both NMOS and PMOS of the driving inverter with a factor $S$ reduces their equivalent resistance by an identical factor, but also raises the intrinsic capacitance of the gate by approximately the same ratio. The propagation delay of the redesigned gate can be estimated

$$ t_p = 0.69(S + \alpha)C_{int}\left(\frac{R_{eqo} + R_{eqp}}{2S}\right) = \left(1 + \frac{\alpha}{\alpha}\right) t_{p0} \quad (5.24) $$

with $t_{p0}$ the **intrinsic delay of the gate** (this is, no extrinsic load, or $\alpha = 0$). Making $S$ infinitely large yields the maximum obtainable performance gain, equal to $1/(1+\alpha)$. Yet, any sizing factor $S$ that is sufficiently larger than $\alpha$ will produce similar results at a substantial gain in silicon area.

For the example in question, we find from Table 5.2 that $\alpha \approx 1.05$ ($C_{int} = 3.0 \text{ fF}$, $C_{ext} = 3.15 \text{ fF}$). This would predict a maximum performance gain of 2.05. A scaling factor of 10 allows us to get within 10% of this optimal performance, while larger device sizes only yield ignorable performance gains.
This is confirmed by simulation results, which predict a maximum obtainable performance improvement of 1.9 ($t_{p0} = 19.3$ psec). From the graph of Figure 5.18, we observe that the bulk of the improvement is already obtained for $S = 5$, and that sizing factors larger than 10 barely yield any extra gain.

**Problem 5.4 Propagation Delay as a Function of (dis)charge Current**

So far, we have expressed the propagation delay as a function of the equivalent resistance of the transistors. Another approach would be replace the transistor by a current source with value equal to the average (dis)charge current over the interval of interest. Derive an expression of the propagation delay using this alternative approach.

### 5.4.3 Propagation Delay Revisited

A detailed analysis of the transient response of the complementary MOS inverter yields some extra observations and design trade-offs, worth analyzing.

**Impact of Fanout**

Eq. (5.23) states that the load capacitance of the inverter can be divided into an intrinsic and an extrinsic component. The latter factor is an obvious function of the fanout of the gate: the larger the fanout, the larger the external load. Assuming that each fanout gate presents an identical load, and that the wiring capacitance is proportional to the fanout, we can rewrite the delay equation as a function of the fanout $N$.

$$t_p(N) = t_{p0}(1 + \alpha N)$$

Equation 5.25
A linear dependence can be observed. Large fanout factors should hence be avoided if performance is an issue. From the preceding discussions, it is furthermore apparent that increasing the sizing factor $S$ of the driving inverter is appropriate and recommendable in the presence of fanout.

**NMOS/PMOS Ratio**

So far, we have consistently widened the PMOS transistor so that its resistance matches that of the pull-down NMOS device. This typically requires a ratio of 3 to 3.5 between PMOS and NMOS width. The motivation behind this approach is to create an inverter with a symmetrical VTC, and to equate the high-to-low and low-to-high propagation delays. However, this does not imply that this ratio also yields the minimum overall propagation delay. If symmetry and reduced noise margins are not of prime concern, it is actually possible to speed up the inverter by reducing the width of the PMOS device!

The reasoning behind this statement is that, while widening the PMOS improves the $t_{PLH}$ of the inverter by increasing the charging current, it also degrades the $t_{PHL}$ by cause of a larger parasitic capacitance. When two contradictory effects are present, there must exist a transistor ratio that optimizes the propagation delay of the inverter.

This optimum ratio can be derived through the following simple analysis. Consider two identical, cascaded CMOS inverters. The load capacitance of the first gate equals approximately

$$C_L = (C_{dp1} + C_{dn1}) + (C_{gp2} + C_{gn2}) + C_W$$  \hspace{1cm} (5.26)

where $C_{dp1}$ and $C_{dn1}$ are the equivalent drain diffusion capacitances of PMOS and NMOS transistors of the first inverter, while $C_{gp2}$ and $C_{gn2}$ are the gate capacitances of the second gate. $C_W$ represents the wiring capacitance.

When the PMOS devices are made $\beta$ times larger than the NMOS ones ($\beta = (W/L)_p / (W/L)_n$), all transistor capacitances will scale in approximately the same way, or $C_{dp1} \approx \beta C_{dn1}$, and $C_{gp2} \approx \beta C_{gn2}$. Eq. (5.26) can then be rewritten:

$$C_L = (1 + \beta)(C_{dn1} + C_{gn2}) + C_W$$  \hspace{1cm} (5.27)

An expression for the propagation delay can be derived, based on Eq. (5.20).

$$t_p = \frac{0.69}{2}((1 + \beta)(C_{dn1} + C_{gn2}) + C_W)(R_{eqn} + R_{eqp})$$

$$= 0.345((1 + \beta)(C_{dn1} + C_{gn2}) + C_W)R_{eqn}(1 + \frac{r}{\beta})$$  \hspace{1cm} (5.28)

$r = R_{eqp}/R_{eqn}$ represents the resistance ratio of identically-sized PMOS and NMOS transistors. The optimal value of $\beta$ can be found by setting $\frac{\partial t_p}{\partial \beta}$ to 0, which yields

$$\beta_{opt} = \sqrt{\frac{1 + \frac{C_W}{C_{dn1} + C_{gn2}}}{\beta}}$$  \hspace{1cm} (5.29)
This means that when the wiring capacitance is negligible \((C_{dn1} + C_{gn2} >> C_W)\), \(\beta_{opt}\) equals \(2\), in contrast to the factor \(r\) normally used in the noncascaded case. If the wiring capacitance dominates, larger values of \(\beta\) should be used. The surprising result of this analysis is that smaller device sizes (and hence smaller design area) yield a faster design at the expense of symmetry and noise margin.

**Example 5.7 Sizing of CMOS Inverter Loaded by an Identical Gate**

Consider again our standard design example. From the values of the equivalent resistances (Table 3.3), we find that a ratio \(\beta\) of 2.4 (= 31 kΩ / 13 kΩ) would yield a symmetrical transient response. Eq. (5.29) now predicts that the device ratio for an optimal performance should equal 1.6. These results are verified in Figure 5.19, which plots the simulated propagation delay as a function of the transistor ratio \(\beta\). The graph clearly illustrates how a changing \(\beta\) trades off between \(t_{pLH}\) and \(t_{pHL}\). The optimum point occurs around \(\beta = 1.9\), which is somewhat higher than predicted. Observe also that the rising and falling delays are identical at the predicted point of \(\beta\) equal to 2.4.

![Figure 5.19](image)

**The rise/fall time of the input signal**

All the above expressions were derived under the assumption that the input signal to the inverter abruptly changed from 0 to \(V_{DD}\) or vice-versa. Only one of the devices is assumed to be on during the (dis)charging process. In reality, the input signal changes gradually and, temporarily, PMOS and NMOS transistors conduct simultaneously. This affects the total current available for (dis)charging and impacts the propagation delay. Figure 5.20 plots the propagation delay of a minimum-size inverter as a function of the input signal slope—as obtained from SPICE. It can be observed that \(t_p\) increases (approximately) linearly with increasing input slope, once \(t_s > t_p(t_s=0)\).

While it is possible to derive an analytical expression describing the relationship between input signal slope and propagation delay, the result tends to be complex and of limited value. From a design perspective, it is more valuable to relate the impact of the finite slope on the performance directly to its cause, which is the limited driving capability of the preceding gate. If the latter would be infinitely strong, its output slope would be zero, and the performance of the gate under examination would be unaffected. The major
Section 5.4 Performance of CMOS Inverter: The Dynamic Behavior

advantage of this approach is that it realizes that a gate is never designed in isolation, and that its performance is both affected by the fanout, and the driving strength of the gate(s) feeding into its inputs. This leads to a revised expression for the propagation delay of an inverter \( i \) in a chain of inverters [Hedenstierna87]:

\[
 t_p^i = t_{\text{step}}^i + \eta t_{\text{step}}^{i-1}
\]  

(5.30)

Eq. (5.30) states that the propagation delay of inverter \( i \) equals the sum of the delay of the same gate for a step input (\( t_{\text{step}}^i \) (i.e. zero input slope) augmented with a fraction of the step-input delay of the preceding gate (\( i-1 \)). The fraction \( \eta \) is an empirical constant. This expression has the advantage of being very simple, yet it exposes all relationships necessary for global delay computations of complex circuits.

Example 5.8 Delay of Inverter embedded in Network

Consider for instance the circuit of . All inverters in this example are assumed to be identical, and to have an intrinsic propagation delay \( t_{p0} \). With the aid of Eq. (5.30) and Eq. (5.25), we can derive an expression for the delay of inverter \( i \):

\[
 t_p^i = t_{p0}(1 + \alpha N) + \eta t_{p0}(1 + \alpha M)
 = t_{p0}(1 + \eta + \alpha(N + \eta M))
\]  

(5.31)

with \( N \) and \( M \) the fanout factors of inverters \( i \) and \( i-1 \), respectively. Typical values for the parameters \( \alpha \) and \( \eta \) are around 1 and 0.25, respectively. Experiments have demonstrated that the model of Eq. (5.31) forms a good approximation of the actual dependencies, although some important deviations can be observed for small values of \( N \) and \( M \).
It is advantageous to keep the signal rise times smaller than or equal to the gate propagation delays. This proves to be true not only for performance, but also for power consumption considerations as will be discussed later. Keeping the rise and fall times of the signals small and of approximately equal values is one of the major challenges in high-performance design, and is often called ‘slope engineering’.

Problem 5.5 Impact of input slope
Determine if reducing the supply voltage increases or decreases the influence of the input signal slope on the propagation delay. Explain your answer.

Delay in the Presence of (Long) Interconnect Wires
The interconnect wire has played a minimal role in our analysis so far. When gates get farther apart, the wire capacitance and resistance can no longer be ignored, and may even dominate the transient response. Earlier delay expressions can be adjusted to accommodate these extra contributions by employing the wire modeling techniques introduced in the previous Chapter. The analysis detailed in Example 4.9 is directly applicable to the problem at hand. Consider the circuit of Figure 5.22, where an inverter drives a single fanout through a wire of length $L$. The driver is represented by a single resistance $R_{dr}$, which is the average between $R_{eqn}$ and $R_{eqp}$. $C_{int}$ and $C_{fan}$ account for the intrinsic capacitance of the driver, and the input capacitance of the fanout gate, respectively.

The propagation delay of the circuit can be obtained by applying the Ellmore delay expression.
Section 5.5 Power, Energy, and Energy-Delay

\[ t_p = 0.69 R_{dr} C_{int} + (0.69 R_{dr} + 0.38 R_w) C_w + 0.69 (R_{dr} + R_w) C_{fan} \]

\[ = 0.69 R_{dr} (C_{int} + C_{fan}) + 0.69 (R_{dr} c_w + r_w C_{fan}) L + 0.38 r_w c_w L^2 \] (5.32)

The 0.38 factor accounts for the fact that the wire represents a distributed delay. \( C_w \) and \( R_w \) stand for the total capacitance and resistance of the wire, respectively. The delay expressions contains a component that is linear with the wire length, as well a quadratic one. It is the latter that causes the wire delay to rapidly become the dominant factor in the delay budget for longer wires.

Example 5.9 Inverter delay in presence of interconnect

Consider the circuit of Figure 5.22, and assume the device parameters of Example 5.5: \( C_{int} = 3 \, \text{fF}, C_{fan} = 3 \, \text{fF}, \) and \( R_w = 0.5(13/1.5 + 31/4.5) = 7.8 \, \text{kΩ} \). The wire is implemented in metal1 and has a width of 0.4 \( \mu \text{m} \) —the minimum allowed. This yields the following parameters: \( c_w = 92 \, \text{aF/μm}, \) and \( r_w = 0.19 \, \text{Ω/μm} \) (Example 4.4). With the aid of Eq. (5.32), we can compute at what wire length the delay of the interconnect becomes equal to the intrinsic delay caused purely by device parasitics. Solving the following quadratic equation yields a single useful solution.

\[ 6.6 \times 10^{-18} L^2 + 0.5 \times 10^{-12} L = 32.29 \times 10^{-12} \]

or

\[ L = 65 \, \mu \text{m} \]

Observe that the extra delay is solely due to the linear factor in the equation, and more specifically due to the extra capacitance introduced by the wire. The quadratic factor (this is, the distributed wire delay) only becomes dominant at much larger wire lengths (\( > 7 \, \text{cm} \)). This is due to the high resistance of the (minimum-size) driver transistors. A different balance emerges when wider transistors are used. Analyze, for instance, the same problem with the driver transistors 100 times wider, as is typical in high-speed, large fan-out drivers.

5.5 Power, Energy, and Energy-Delay

So far, we have seen that the static CMOS inverter with its almost ideal VTC—symmetrical shape, full logic swing, and high noise margins—offers a superior robustness, which simplifies the design process considerably and opens the door for design automation. Another major attractor for static CMOS is the almost complete absence of power consumption in steady-state operation mode. It is this combination of robustness and low static power that has made static CMOS the technology of choice of most contemporary digital designs. The power dissipation of a CMOS circuit is instead dominated by the dynamic dissipation resulting from charging and discharging capacitances.
5.5.1 Dynamic Power Consumption

Dynamic Dissipation due to Charging and Discharging Capacitances

Each time the capacitor $C_L$ gets charged through the PMOS transistor, its voltage rises from 0 to $V_{DD}$, and a certain amount of energy is drawn from the power supply. Part of this energy is dissipated in the PMOS device, while the remainder is stored on the load capacitor. During the high-to-low transition, this capacitor is discharged, and the stored energy is dissipated in the NMOS transistor.\(^3\)

A precise measure for this energy consumption can be derived. Let us first consider the low-to-high transition. We assume, initially, that the input waveform has zero rise and fall times, or, in other words, that the NMOS and PMOS devices are never on simultaneously. Therefore, the equivalent circuit of Figure 5.23 is valid. The values of the energy $E_{VDD}$ taken from the supply during the transition, as well as the energy $E_C$ stored on the capacitor at the end of the transition, can be derived by integrating the instantaneous power over the period of interest. The corresponding waveforms of $v_{out}(t)$ and $i_{VDD}(t)$ are pictured in Figure 5.24.

$$E_{VDD} = \int_{0}^{\infty} i_{VDD}(t)V_{DD} \, dt = V_{DD} \int_{0}^{\infty} C_L \frac{dv_{out}}{dt} \, dt = C_L V_{DD} \int_{0}^{v_{oo}} dv_{out} = C_L V_{DD}^2 \quad (5.33)$$

$$E_C = \int_{0}^{\infty} i_{VDD}(t)v_{out} \, dt = \int_{0}^{\infty} C_L \frac{dv_{out}}{dt} v_{out} \, dt = C_L \int_{0}^{v_{oo}} v_{out} \, dv_{out} = \frac{C_L V_{DD}^2}{2} \quad (5.34)$$

These results can also be derived by observing that during the low-to-high transition, $C_L$ is loaded with a charge $C_L V_{DD}$. Providing this charge requires an energy from the supply equal to $C_L V_{DD}^2$ ($\approx Q \times V_{DD}$). The energy stored on the capacitor equals $C_L V_{DD}^2/2$. This means that only half of the energy supplied by the power source is stored on $C_L$. The other half has been dissipated by the PMOS transistor. Notice that this energy dissipation is independent of the size (and hence the resistance) of the PMOS device! During the discharge phase, the charge is removed from the capacitor, and its energy is dissipated in the NMOS device. Once again, there is no dependence on the size of the device. In summary, each switching cycle (consisting of an L$\rightarrow$H and an H$\rightarrow$L transition) takes a fixed amount of energy, equal to $C_L V_{DD}^2$. In order to compute the power consumption, we have to take

---

\(^3\) Observe that this model is a simplification of the actual circuit. In reality, the load capacitance consists of multiple components some of which are located between the output node and GND, others between output node and $V_{DD}$. The latter experience a charge-discharge cycle that is out of phase with the capacitances to GND, i.e. they get charged when $V_{out}$ goes low and discharged when $V_{out}$ rises. While this distributes the energy delivery by the supply over the two phases, it does not impact the overall dissipation, and the results presented in this section are still valid.
into account how often the device is switched. If the gate is switched on and off \( f_{0 \rightarrow 1} \) times per second, the power consumption equals

\[
P_{\text{dyn}} = C_L V_{\text{DD}}^2 f_{0 \rightarrow 1}
\]

(5.35)

\( f_{0 \rightarrow 1} \) represents the frequency of energy-consuming transitions, this is \( 0 \rightarrow 1 \) transitions for static CMOS.

Advances in technology result in ever-higher of values of \( f_{0 \rightarrow 1} \) (as \( t_p \) decreases). At the same time, the total capacitance on the chip (\( C_L \)) increases as more and more gates are placed on a single die. Consider for instance a 0.25 \( \mu \)m CMOS chip with a clock rate of 500 Mhz and an average load capacitance of 15 fF/gate, assuming a fanout of 4. The power consumption per gate for a 2.5 V supply then equals approximately 50 \( \mu \)W. For a design with 1 million gates and assuming that a transition occurs at every clock edge, this would result in a power consumption of 50 W! This evaluation presents, fortunately, a pessimistic perspective. In reality, not all gates in the complete IC switch at the full rate of 500 Mhz. The actual activity in the circuit is substantially lower.

**Example 5.10 Capacitive power dissipation of inverter**

The capacitive dissipation of the CMOS inverter of Example 5.4 is now easily computed. In Table 5.2, the value of the load capacitance was determined to equal 6 fF. For a supply voltage of 2.5 V, the amount of energy needed to charge and discharge that capacitance equals

\[
E_{\text{dyn}} = C_L V_{DD}^2 = 37.5 \text{ fJ}
\]

Assume that the inverter is switched at the maximum possible rate ( \( T = 1/f = t_{p_{\text{LH}}} + t_{p_{\text{HL}}} = 2 t_p \)). For a \( t_p \) of 32.5 psec (Example 5.5), we find that the dynamic power dissipation of the circuit is

\[
P_{\text{dyn}} = E_{\text{dyn}} / (2t_p) = 580 \text{ \( \mu \)W}
\]

Of course, an inverter in an actual circuit is rarely switched at this maximum rate, and even if done so, the output does not swing from rail-to-rail. The power dissipation will hence be substantially lower. For a rate of 4 GHz (\( T = 250 \) psec), the dissipation reduces to 150 \( \mu \)W. This is confirmed by simulations, which yield a power consumption of 155 \( \mu \)W.
Computing the dissipation of a complex circuit is complicated by the $f_{0 \rightarrow 1}$ factor, also called the switching activity. While the switching activity is easily computed for an inverter, it turns out to be far more complex in the case of higher-order gates and circuits. One concern is that the switching activity of a network is a function of the nature and the statistics of the input signals: If the input signals remain unchanged, no switching happens, and the dynamic power consumption is zero! On the other hand, rapidly changing signals provoke plenty of switching and hence dissipation. Other factors influencing the activity are the overall network topology and the function to be implemented. We can accommodate this by another rewrite of the equation, or

$$P_{dyn} = C_L V_{DD}^2 f_{0 \rightarrow 1} = C_L V_{DD}^2 P_{0 \rightarrow 1} f = C_{EFF} V_{DD}^2 f$$

(5.36)

where $f$ now presents the maximum possible event rate of the inputs (which is often the clock rate) and $P_{0 \rightarrow 1}$ the probability that a clock event results in a $0 \rightarrow 1$ (or power-consuming) event at the output of the gate. $C_{EFF} = P_{0 \rightarrow 1} C_L$ is called the effective capacitance and represents the average capacitance switched every clock cycle. For our example, an activity factor of 10% ($P_{0 \rightarrow 1} = 0.1$) reduces the average consumption to 5 W.

Example 5.11 Switching activity

Consider the waveforms on the right where the upper waveform represents the idealized clock signal, and the bottom one shows the signal at the output of the gate. Power consuming transitions occur 2 out of 8 times, which is equivalent to a transition probability of 0.25 (or 25%).

![Figure 5.25 Clock and signal waveforms](image)

Low Energy/Power Design Techniques

With the increasing complexity of the digital integrated circuits, it is anticipated that the power problem will only worsen in future technologies. This is one of the reasons that lower supply voltages are becoming more and more attractive. **Reducing $V_{DD}$ has a quadratic effect on $P_{dy}$**. For instance, reducing $V_{DD}$ from 2.5 V to 1.25 V for our example drops the power dissipation from 5 W to 1.25 W. This assumes that the same clock rate can be sustained. Figure 5.17 demonstrates that this assumption is not that unrealistic as long as the supply voltage is substantially higher than the threshold voltage. An important performance penalty occurs once $V_{DD}$ approaches 2 $V_T$.

When a lower bound on the supply voltage is set by external constraints (as often happens in real-world designs), or when the performance degradation due to lowering the supply voltage is intolerable, the only means of reducing the dissipation is by lowering the effective capacitance. This can be achieved by addressing both of its components: the physical capacitance and the switching activity.

A **reduction in the switching activity** can only be accomplished at the logic and architectural abstraction levels, and will be discussed in more detail in later Chapters. **Lowering the**
Section 5.5 Power, Energy, and Energy-Delay

Physical capacitance is an overall worthwhile goal, which also helps to improve the performance of the circuit. As most of the capacitance in a combinational logic circuit is due to transistor capacitances (gate and diffusion), it makes sense to keep those contributions to a minimum when designing for low power. This means that transistors should be kept to minimal size whenever possible or reasonable. This definitely affects the performance of the circuit, but the effect can be offset by using logic or architectural speed-up techniques. The only instances where transistors should be sized up is when the load capacitance is dominated by extrinsic capacitances (such as fan-out or wiring capacitance). This is contrary to common design practices used in cell libraries, where transistors are generally made large to accommodate a range of loading and performance requirements.

The above observations lead to an interesting design challenge. Assume we have to minimize the energy dissipation of a circuit with a specified lower-bound on the performance. The obvious approach is to lower the supply voltage as much as possible, and to compensate the loss in performance by increasing the transistor sizes. Yet, the latter causes the capacitance to increase. It may be foreseen that at a low enough supply voltage, the latter factor may start to dominate and cause energy to increase with a further drop in the supply voltage.

To analyze the transistor-sizing for minimum energy problem, let us analyze the simple case of a static CMOS inverter driving a load capacitance consisting of an intrinsic ($C_{int}$) and an extrinsic component ($C_{ext}$) (Figure 5.26a). While the former represents the diffusion capacitances, the latter stands for wiring capacitance and fan-out. It is assumed that the ratio between PMOS and NMOS transistors is constant. The factor $S$ stands for the inverter sizing factor, where $S$ is equal to 1 for an inverter constructed of minimum-size devices. We can see that the intrinsic capacitance of the scaled device is proportional to $S$ (or $C_{int}(scaled) = SC_{int}$). Figure 5.26b plots the normalized energy (per transition) as a function of the scaling factor $S$ with the ratio between the extrinsic and intrinsic capacitance as a parameter: $\alpha = C_{ext}/C_{int}$. The speed of all implementations is kept constant by appropriately adjusting the supply voltage: larger values of $S$ normally mean lower values of the supply voltage.

Figure 5.26 Normalized energy of a MOS inverter with load capacitance $C_L$, as a function of the inverter size $S$ and the ratio between the extrinsic and intrinsic capacitance $\alpha (C_{ext}/C_{int})$. (assuming a reference supply voltage of 2.5 V).
When $\alpha = 0$ (or the load capacitance is zero), the lowest energy consumption is obtained when using minimum-size devices. Only when the extrinsic capacitances dominate ($\alpha \geq 1$) does it make sense to widen the devices. This result should come as no surprise: transistor sizing to increase performance—and reduce the energy by lowering the supply voltage—only makes sense as long as performance is dominated by the extrinsic capacitance. Once the intrinsic capacitance becomes the primary factor, further increases in the device sizes only raise the energy consumption while no longer lowering the propagation delay. For example, a sizing factor $S$ of 3.75 minimizes the energy for a load of $\alpha = 5$. The energy-reduction—with a factor of 4 with respect to the circuit instance with minimum-size devices—requires that the supply voltage be reduced to 1.03 V.

**Example 5.12 Transistor Sizing for Inverter**

We derive a simplified expression for the normalized energy of the inverter of Figure 5.26 as a function of $S$ and $\alpha$. The energy is normalized with respect to the case for $S = 1$, which is called the reference.

An expression for the propagation delay of the gate was already derived in Eq. (5.24), and is repeated here for convenience.

$$t_p = 0.69(S + \alpha)C_{\text{int}} \left( \frac{R_{\text{eq}} + R_{\text{op}}}{2S} \right) = \left( 1 + \frac{\alpha}{S} \right) t_{p0}$$

$t_{p0}$ stands for the intrinsic delay of the gate at the reference voltage $V_{DD}$. Its dependence upon $V_{DD}$ is approximated by the following expression, derived from Eq. (5.21).

$$t_{p0} = \frac{V_{DD}}{V_{DD} - V_{TE}} = \frac{1}{1 - \frac{V_{TE}}{V_{DD}}}$$

with $V_{TE} = V_T + \frac{V_{DSAT}}{2}$ (assume a value averaged over NMOS and PMOS).

Keeping the propagation delay of the scaled inverter constant with respect to the reference case means lowering the supply voltage:

$$\frac{V_{DD}}{V_{TE}} = \frac{S(1 + \alpha)}{\alpha(S - 1) + (S + \alpha)(V_{TE}/V_{DD})}$$

where $V'_{DD}$ and $V_{DD}$ are the supply voltages of the scaled and reference inverters, respectively. The (normalized) dissipated energy of the scaled inverter is now derived:

$$\frac{E'}{E_{\text{ref}}} = \frac{(S + \alpha)C_{\text{int}}(V'_{DD})^2}{(1 + \alpha)C_{\text{int}}(V_{DD})^2} = \frac{(S + \alpha)S(1 + \alpha)}{(\alpha(S - 1)(V_{DD}/V_{TE}) + (S + \alpha))^2}$$

The charts for $V_{DD} = 2.5$ V and $V_{TE} = 0.75$ V are plotted in Figure 5.26. Observe that for $S \gg \alpha$, the dissipation increases linearly with $S$. The reader should further be aware that the presented model is somewhat optimistic, as it ignores the extra energy dissipation related to the increased gate capacitance of the driving transistors.
Dissipation Due to Direct-Path Currents

In actual designs, the assumption of the zero rise and fall times of the input wave forms is not correct. The finite slope of the input signal causes a direct current path between $V_{DD}$ and GND for a short period of time during switching, while the NMOS and the PMOS transistors are conducting simultaneously. This is illustrated in Figure 5.27. Under the (reasonable) assumption that the resulting current spikes can be approximated as triangles and that the inverter is symmetrical in its rising and falling responses, we can compute the energy consumed per switching period,

$$E_{dp} = V_{DD} \frac{I_{peak} t_{sc}}{2} + V_{DD} \frac{I_{peak} t_{sc}}{2} = t_{sc} V_{DD} I_{peak}$$  \hspace{1cm} (5.38)

as well as the average power consumption

$$P_{dp} = t_{sc} V_{DD} I_{peak} f = C_{sc} V_{DD}^2 f$$  \hspace{1cm} (5.39)

The direct-path power dissipation is proportional to the switching activity, similar to the capacitive power dissipation. $t_{sc}$ represents the time both devices are conducting. For a linear input slope, this time is reasonably well approximated by Eq. (5.40) where $t_s$ represents the 0-100% transition time.

$$t_{sc} = \frac{V_{DD} - 2 VT_I}{V_{DD}} \approx \frac{V_{DD} - 2VT}{V_{DD}} \times t_s \left( \frac{0.8}{0.8} \right)$$  \hspace{1cm} (5.40)

$I_{peak}$ is determined by the saturation current of the devices and is hence directly proportional to the sizes of the transistors. The peak current is also a strong function of the ratio between input and output slopes. This relationship is best illustrated by the following simple analysis: Consider a static CMOS inverter with a $0 \rightarrow 1$ transition at the input. Assume first that the load capacitance is very large, so that the output fall time is significantly larger than the input rise time (Figure 5.28a). Under those circumstances, the input moves through the transient region before the output starts to change. As the source-drain voltage of the PMOS device is approximately 0 during that period, the device shuts off without ever delivering any current. The short-circuit current is close to zero in this case.
Consider now the reverse case, where the output capacitance is very small, and the output fall time is substantially smaller than the input rise time (Figure 5.28b). The drain-source voltage of the PMOS device equals $V_{DD}$ for most of the transition period, guaranteeing the maximal short-circuit current (equal to the saturation current of the PMOS). This clearly represents the worst-case condition. The conclusions of the above analysis are confirmed in Figure 5.29, which plots the short-circuit current through the NMOS transistor during a low-to-high transition as a function of the load capacitance.

This analysis leads to the conclusion that the short-circuit dissipation is minimized by making the output rise/fall time larger than the input rise/fall time. On the other hand, making the output rise/fall time too large slows down the circuit and can cause short-circuit currents in the fan-out gates. This presents a perfect example of how local optimization and forgetting the global picture can lead to an inferior solution.

**Design Techniques**

A more practical rule, which optimizes the power consumption in a global way, can be formulated (Veendrick84):
The power dissipation due to short-circuit currents is minimized by matching the rise/fall times of the input and output signals. At the overall circuit level, this means that rise/fall times of all signals should be kept constant within a range.

Making the input and output rise times of a gate identical is not the optimum solution for that particular gate on its own, but keeps the overall short-circuit current within bounds. This is shown in Figure 5.30, which plots the short-circuit energy dissipation of an inverter (normalized with respect to zero-input rise-time dissipation) as a function of the ratio \( r \) between input and output rise/fall times. When the load capacitance is too small for a given inverter size (\( r > 2\ldots3 \) for \( V_{DD} = 5 \text{ V} \)), the power is dominated by the short-circuit current. For very large capacitance values, all power dissipation is devoted to charging and discharging the load capacitance. When the rise/fall times of inputs and outputs are equalized, most power dissipation is associated with the dynamic power and only a minor fraction (< 10%) is devoted to short-circuit currents.

Observe also that the impact of short-circuit current is reduced when we lower the supply voltage, as is apparent from Eq. (5.40). In the extreme case, when \( V_{DD} < V_{TN} + |V_{TP}| \), short-circuit dissipation is completely eliminated, because both devices are never on simultaneously. With threshold voltages scaling at a slower rate than the supply voltage, short-circuit power dissipation is becoming of a lesser importance in deep-submicron technologies. At a supply voltage of 2.5 V and thresholds around 0.5 V, an input/output slope ratio of 2 is needed to cause a 10% degradation in dissipation.

Finally, it is worth observing that the short-circuit power dissipation can be modeled by adding a load capacitance \( C_{sc} = t_{sc} I_{peak}/V_{DD} \) in parallel with \( C_L \), as is apparent in Eq. (5.39). The value of this short-circuit capacitance is a function of \( V_{DD} \), the transistor sizes, and the input-output slope ratio.
5.5.2 Static Consumption

The static (or steady-state) power dissipation of a circuit is expressed by Eq. (5.41), where \( I_{\text{stat}} \) is the current that flows between the supply rails in the absence of switching activity

\[
P_{\text{stat}} = I_{\text{stat}} V_DD
\]  

(5.41)

Ideally, the static current of the CMOS inverter is equal to zero, as the PMOS and NMOS devices are never on simultaneously in steady-state operation. There is, unfortunately, a leakage current flowing through the reverse-biased diode junctions of the transistors, located between the source or drain and the substrate as shown in Figure 5.31. This contribution is, in general, very small and can be ignored. For the device sizes under consideration, the leakage current per unit drain area typically ranges between 10-100 pA/\( \mu \)m\(^2\) at room temperature. For a die with 1 million gates, each with a drain area of 0.5 \( \mu \)m\(^2\) and operated at a supply voltage of 2.5 V, the worst-case power consumption due to diode leakage equals 0.125 mW, which is clearly not much of an issue.

However, be aware that the junction leakage currents are caused by thermally generated carriers. Their value increases with increasing junction temperature, and this occurs in an exponential fashion. At 85°C (a common junction temperature limit for commercial hardware), the leakage currents increase by a factor of 60 over their room-temperature values. Keeping the overall operation temperature of a circuit low is consequently a desirable goal. As the temperature is a strong function of the dissipated heat and its removal mechanisms, this can only be accomplished by limiting the power dissipation of the circuit and/or by using chip packages that support efficient heat removal.

An emerging source of leakage current is the subthreshold current of the transistors. As discussed in Chapter 3, an MOS transistor can experience a drain-source current, even when \( V_{\text{GS}} \) is smaller than the threshold voltage (Figure 5.32). The closer the threshold voltage is to zero volts, the larger the leakage current at \( V_{\text{GS}} = 0 \) V and the larger the static power consumption. To offset this effect, the threshold voltage of the device has generally been kept high enough. Standard processes feature \( V_T \) values that are never smaller than 0.5-0.6V and that in some cases are even substantially higher (~0.75V).

This approach is being challenged by the reduction in supply voltages that typically goes with deep-submicron technology scaling as became apparent in Figure 3.40. We con-
Section 5.5 Power, Energy, and Energy-Delay

cluded earlier (Figure 5.17) that scaling the supply voltages while keeping the threshold voltage constant results in an important loss in performance, especially when $V_{DD}$ approaches $2V_T$. One approach to address this performance issue is to scale the device thresholds down as well. This moves the curve of Figure 5.17 to the left, which means that the performance penalty for lowering the supply voltage is reduced. Unfortunately, the threshold voltages are lower-bounded by the amount of allowable subthreshold leakage current, as demonstrated in Figure 5.32. The choice of the threshold voltage hence represents a trade-off between performance and static power dissipation. The continued scaling of the supply voltage predicted for the next generations of CMOS technologies will however force the threshold voltages ever downwards, and will make subthreshold conduction a dominant source of power dissipation. Process technologies that contain devices with sharper turn-off characteristic will therefore become more attractive. An example of the latter is the SOI (Silicon-on-Insulator) technology whose MOS transistors have slope-factors that are close to the ideal 60 mV/decade.

**Example 5.13 Impact of threshold reduction on performance and static power dissipation**

Consider a minimum size NMOS transistor in the 0.25 µm CMOS technology. In Chapter 3, we derived that the slope factor $S$ for this device equals 90 mV/decade. The off-current (at $V_{GS} = 0$) of the transistor for a $V_T$ of approximately 0.5V equals $10^{-11}$ A (Figure 3.22). Reducing the threshold with 200 mV to 0.3 V multiplies the off-current of the transistors with a factor of 170! Assuming a million gate design with a supply voltage of 1.5 V, this translates into a static power dissipation of $10^6 \times 170 \times 10^{-11} \times 1.5 = 2.6$ mW. A further reduction of the threshold to 100 mV results in an unacceptable dissipation of almost 0.5 W! At that supply voltage, the threshold reductions correspond to a performance improvement of 25% and 40%, respectively.

This lower bound on the thresholds is in some sense artificial. The idea that the leakage current in a static CMOS circuit has to be zero is a preconception. Certainly, the presence of leakage currents degrades the noise margins, because the logic levels are no longer equal to the supply rails. As long as the noise margins are within range, this is not a compelling issue. The leakage currents, of course, cause an increase in static power dissipa-
tion. This is offset by the drop in supply voltage, that is enabled by the reduced thresholds at no cost in performance, and results in a quadratic reduction in dynamic power. For a 0.25 μm CMOS process, the following circuit configurations obtain the same performance: 3 V supply–0.7 V $V_T$; and 0.45 V supply–0.1 V $V_T$. The dynamic power consumption of the latter is, however, 45 times smaller [Liu93]! Choosing the correct values of supply and threshold voltages once again requires a trade-off. The optimal operation point depends upon the activity of the circuit. In the presence of a sizable static power dissipation, it is essential that non-active modules are powered down, lest static power dissipation would become dominant. Power-down (also called standby) can be accomplished by disconnecting the unit from the supply rails, or by lowering the supply voltage.

5.5.3 Putting It All Together

The total power consumption of the CMOS inverter is now expressed as the sum of its three components:

$$P_{tot} = P_{dyn} + P_{dp} + P_{stat} = (C_L V_{DD}^2 + V_{DD} I_{peak} t_p) f_{0 \rightarrow 1} + V_{DD} I_{leak}$$

(5.42)

In typical CMOS circuits, the capacitive dissipation is by far the dominant factor. The direct-path consumption can be kept within bounds by careful design, and should hence not be an issue. Leakage is ignorable at present, but this might change in the not too distant future.

The Power-Delay Product, or Energy per Operation

In Chapter 1, we introduced the power-delay product (PDP) as a quality measure for a logic gate.

$$PDP = P_{av} t_p$$

(5.43)

The PDP presents a measure of energy, as is apparent from the units (Wsec = Joule). Assuming that the gate is switched at its maximum possible rate of $f_{max} = 1/(2 t_p)$, and ignoring the contributions of the static and direct-path currents to the power consumption, we find

$$PDP = C_L V_{DD}^2 f_{max} t_p = C_L V_{DD}^2$$

(5.44)

The PDP stands for the average energy consumed per switching event (this is, for a 0→1, or a 1→0 transition). Remember that earlier we had defined $E_{av}$ as the average energy per switching cycle (or per energy-consuming event). As each inverter cycle contains a 0→1, and a 1→0 transition, $E_{av}$ hence is twice the PDP.

Energy-Delay Product

The validity of the PDP as a quality metric for a process technology or gate topology is questionable. It measures the energy needed to switch the gate, which is an important
Section 5.5 Power, Energy, and Energy-Delay

property for sure. Yet for a given structure, this number can be made arbitrarily low by reducing the supply voltage. From this perspective, the optimum voltage to run the circuit at would be the lowest possible value that still ensures functionality. This comes at the major expense in performance, at discussed earlier. A more relevant metric should combine a measure of performance and energy. The energy-delay product (EDP) does exactly that.

\[
EDP = PDP \times t_p = P_{av}t_p^2 = \frac{C_l V_{DD}^2}{2} \quad (5.45)
\]

It is worth analyzing the voltage dependence of the EDP. Higher supply voltages reduce delay, but harm the energy, and the opposite is true for low voltages. An optimum operation point should hence exist. Assuming that NMOS and PMOS transistors have comparable threshold and saturation voltages, we can simplify the propagation delay expression Eq. (5.21).

\[
t_p \approx \frac{\alpha C_l V_{DD}}{V_{DD} - V_{TE}} \quad (5.46)
\]

where \( V_{TE} = V_T + \frac{V_{DSAT}}{2} \), and \( \alpha \) technology parameter. Combining Eq. (5.45) and Eq. (5.46),

\[
EDP = \frac{\alpha C_l^2 V_{DD}^2}{2(V_{DD} - V_{TE})} \quad (5.47)
\]

The optimum supply voltage can be obtained by taking the derivative of Eq. (5.47) with respect to \( V_{DD} \), and equating the result to 0.

\[
V_{DDopt} = \left( \frac{3}{2} \right) V_{TE} \quad (5.48)
\]

The remarkable outcome from this analysis is the low value of the supply voltage that simultaneously optimizes performance and energy. For sub-micron technologies with thresholds in the range of 0.5 V, the optimum supply is situated around 1 V.

\* Example 5.14 Optimum supply voltage for 0.25 \( \mu \text{m} \) CMOS inverter

From the technology parameters for our generic CMOS process presented in Chapter 3, the value of \( V_{TE} \) can be derived.

\[
V_Tn = 0.43 \text{ V}, V_{Dn} = 0.63 \text{ V}, V_{TEn} = 0.74 \text{ V}, \\
V_Tp = -0.4 \text{ V}, V_{Dp} = -1 \text{ V}, V_{TEp} = -0.9 \text{ V}, \\
V_{TE} \approx (V_{TEn} - V_{TEp})/2 = 0.8 \text{ V}
\]

Hence, \( V_{DDopt} = (3/2) \times 0.8 \text{ V} = 1.2 \text{ V} \). The simulated graphs of Figure 5.33, plotting normalized delay, energy, and energy-delay product, confirm this result. The optimum supply volt-

\* This equation is only accurate as long as the devices remain in velocity saturation, which is probably not the case for the lower supply voltages. This introduces some inaccuracy in the analysis, but will not distort the overall result.
age is predicted to equal 1.1 V. The charts clearly illustrate the trade-off between delay and energy.

**WARNING:** While the above example demonstrates that there exists a supply voltage that minimizes the energy-delay product of a gate, this voltage does not necessarily represent the optimum voltage for a given design problem. For instance, some designs require a minimum performance, which requires a higher voltage at the expense of energy. Similarly, a lower-energy design is possible by operating by circuit at a lower voltage and by obtaining the overall system performance through the use of architectural techniques such as pipelining or concurrency.

### 5.5.4 Analyzing Power Consumption Using SPICE

A definition of the average power consumption of a circuit was provided in Chapter 1, and is repeated here for the sake of convenience.

\[
P_{av} = \frac{1}{T} \int_0^T p(t) dt = \frac{V_{DD}}{T} \int_0^T i_{DD}(t) dt
\]  

(5.49)

with \( T \) the period of interest, and \( V_{DD} \) and \( i_{DD} \) the supply voltage and current, respectively. Some implementations of SPICE provide built-in functions to measure the average value of a circuit signal. For instance, the HSPICE `.MEASURE TRAN I(VDD) AVG` command computes the area under a computed transient response (\( I(VDD) \)) and divides it by the period of interest. This is identical to the definition given in Eq. (5.49). Other implementations of SPICE are, unfortunately, not as extensive. This is not as bad as it seems, as long as one realizes that SPICE is actually a differential equation solver. A small circuit can easily be conceived that acts as an integrator and whose output signal is nothing but the average power.
Consider, for instance, the circuit of Figure 5.34. The current delivered by the power supply is measured by the current-controlled current source and integrated on the capacitor $C$. The resistance $R$ is only provided for DC-convergence reasons and should be chosen as high as possible to minimize leakage. A clever choice of the element parameter ensures that the output voltage $P_{av}$ equals the average power consumption. The operation of the circuit is summarized in Eq. (5.50) under the assumption that the initial voltage on the capacitor $C$ is zero.

$$C \frac{dP_{av}}{dt} = k_iDD$$

or

$$P_{av} = k \int_{0}^{T} V_{DD} dt$$

Equating Eq. (5.49) and Eq. (5.50) yields the necessary conditions for the equivalent circuit parameters: $k/C = V_{DD}/T$. Under these circumstances, the equivalent circuit shown presents a convenient means of tracking the average power in a digital circuit.

Figure 5.34 Equivalent circuit to measure average power in SPICE.

Example 5.15 Average Power of Inverter

The average power consumption of the inverter of Example 5.4 is analyzed using the above technique for a toggle period of 250 psec ($T = 250$ psec, $k = 1$, $V_{DD} = 2.5$ V, hence $C = 100$ pF). The resulting power consumption is plotted in Figure 5.35, showing an average power consumption of approximately 157.3 $\mu$W. The .MEAS AVG command yields a value of 160.32 $\mu$W, which demonstrates the approximate equivalence of both methods. These numbers are equivalent to an energy of 39 fJ (which is close to the 37.5 fJ derived in Example 5.10). Observe the slightly negative dip during the high-to-low transition. This is due to the injection of current into the supply, when the output briefly overshoots $V_{DD}$ as a result of the capacitive coupling between input and output (as is apparent from the transient response of Figure 5.16).
5.6 Perspective: Technology Scaling and its Impact on the Inverter Metrics

In section 3.5, we have explored the impact of the scaling of technology on some of the important design parameters such as area, delay, and power. For the sake of clarity, we repeat here some of the most important entries in the resulting scaling table (Table 3.8).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Relation</th>
<th>Full Scaling</th>
<th>General Scaling</th>
<th>Fixed-Voltage Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area/Device</td>
<td>( WL )</td>
<td>( 1/S^2 )</td>
<td>( 1/S^2 )</td>
<td>( 1/S^2 )</td>
</tr>
<tr>
<td>Intrinsic Delay</td>
<td>( R_{\text{on}}C_{\text{gate}} )</td>
<td>( 1/S )</td>
<td>( 1/S )</td>
<td>( 1/S )</td>
</tr>
<tr>
<td>Intrinsic Energy</td>
<td>( C_{\text{gate}}V^2 )</td>
<td>( 1/S^3 )</td>
<td>( 1/SU^2 )</td>
<td>( 1/S )</td>
</tr>
<tr>
<td>Intrinsic Power</td>
<td>Energy/Delay</td>
<td>( 1/S^3 )</td>
<td>( 1/U^2 )</td>
<td>( 1 )</td>
</tr>
<tr>
<td>Power Density</td>
<td>( P/\text{Area} )</td>
<td>( 1 )</td>
<td>( S^2/U^2 )</td>
<td>( S^2 )</td>
</tr>
</tbody>
</table>

Figure 5.35 Deriving the power consumption using SPICE.
Section 5.6  Perspective: Technology Scaling and its Impact on the Inverter Metrics

To validity of these theoretical projections can be verified by looking back and observing the trends during the past decades. From , we can derive that the gate delay indeed decreases exponentially at a rate of 13%/year, or halving every five years. This rate is on course with the prediction of Table 5.3, since $S$ averages approximately 1.15 as we had already observed in Figure 3.39. The delay of a 2-input NAND gate with a fanout of four has gone from tens of nanoseconds in the 1960s to a tenth of a nanosecond in the year 2000, and is projected to be a few tens of picoseconds by 2010.

Reducing power dissipation has only been a second-order priority until recently. Hence, statistics on dissipation-per-gate or design are only marginally available. An interesting chart is shown in Figure 5.37, which plots the power density measured over a large number of designs produced between 1980 and 1995. Although the variation is large—even for a fixed technology—it shows the power density to increase approximately with $S^2$. This is in correspondence with the fixed-voltage scaling scenario presented in Table 5.3. For more recent years, we expect a scenario more in line with the full-scaling model—which predicts a constant power density—due to the accelerated supply-voltage scaling and the increased attention to power-reducing design techniques. Even under these circumstances, power dissipation-per-chip will continue to increase due to the ever-larger die sizes.

The presented scaling model has one fatal flaw however: the performance and power predictions produce purely “intrinsic” numbers that take only device parameters into account. In Chapter 4, it was concluded that the interconnect wires exhibit a different scaling behavior, and that wire parasitics may come to dominate the overall performance. Similarly, charging and discharging the wire capacitances may dominate the energy bud-
get. To get a crisper perspective, one has to construct a combined model that considers device and wire scaling models simultaneously. The impact of the wire capacitance and its scaling behavior is summarized in Table 5.4. We adopt the fixed-resistance model introduced in Chapter 4. We furthermore assume that the resistance of the driver dominates the wire resistance, which is definitely the case for short to medium-long wires.

Table 5.4 Scaling scenarios for wire capacitance. $S$ and $U$ represent the technology and voltage scaling parameters, respectively, while $S_L$ stands for the wire-length scaling factor. $\varepsilon_c$ represents the impact of fringing and interwire capacitances.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Relation</th>
<th>General Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wire Capacitance</td>
<td>$WL/t$</td>
<td>$\varepsilon_c/S_L$</td>
</tr>
<tr>
<td>Wire Delay</td>
<td>$R_{on}C_{tot}$</td>
<td>$\varepsilon_c/S_L$</td>
</tr>
<tr>
<td>Wire Energy</td>
<td>$C_{tot}V^2$</td>
<td>$\varepsilon_c/S_L U^2$</td>
</tr>
<tr>
<td>Wire Delay / Intrinsic Delay</td>
<td>$\varepsilon_c S/S_L$</td>
<td></td>
</tr>
<tr>
<td>Wire Energy / Intrinsic Energy</td>
<td>$\varepsilon_c S/S_L$</td>
<td></td>
</tr>
</tbody>
</table>

The model predicts that the interconnect-caused delay (and energy) gain in importance with the scaling of technology. This impact is limited to an increase with $\varepsilon_c$ for short wires ($S = S_L$), but it becomes increasingly more outspoken for medium-range and long wires ($S_L < S$). These conclusions have been confirmed by a number of studies, an example of which is shown in Figure 5.38. How the ratio of wire over intrinsic contributions will actually evolve is debatable, as it depends upon a wide range of independent parameters such as system architecture, design methodology, transistor sizing, and interconnect materials. The doomday scenario that interconnect may cause CMOS performance to saturate in the very near future hence may be exaggerated. Yet, it is clear that increased attention to interconnect is an absolute necessity, and may change the way the next-generation circuits are designed and optimized (e.g. Sylvester99).
Section 5.7 Summary

This chapter presented a rigorous and in-depth analysis of the static CMOS inverter. The key characteristics of the gate are summarized:

- The static CMOS inverter combines a pull-up PMOS section with a pull-down NMOS device. The PMOS is normally made wider than the NMOS due to its inferior current-driving capabilities.

- The gate has an almost ideal voltage-transfer characteristic. The logic swing is equal to the supply voltage and is not a function of the transistor sizes. The noise margins of a symmetrical inverter (where PMOS and NMOS transistor have equal current-driving strength) approach $V_{DD}/2$. The steady-state response is not affected by fan-out.

- Its propagation delay is dominated by the time it takes to charge or discharge the load capacitor $C_L$. To a first order, it can be approximated as
  \[
  t_p = 0.69 C_L \left( \frac{R_{eqP} + R_{eqN}}{2} \right)
  \]
  Keeping the load capacitance small is the most effective means of implementing high-performance circuits. Transistor sizing may help to improve performance as long as the delay is dominated by the extrinsic (or load) capacitance of fanout and wiring.

- The power dissipation is dominated by the dynamic power consumed in charging and discharging the load capacitor. It is given by $P_{0<i} C_L V_{DD}^2 f$. The dissipation is proportional to the activity in the network. The dissipation due to the direct-path currents occurring during switching can be limited by careful tailoring of the signal slopes. The static dissipation can usually be ignored but might become a major factor in the future as a result of subthreshold currents.

- Scaling the technology is an effective means of reducing the area, propagation delay and power consumption of a gate. The impact is even more striking if the supply voltage is scaled simultaneously.

- The interconnect component is gradually taking a larger fraction of the delay and performance budget.

5.8 To Probe Further

The operation of the CMOS inverter has been the topic of numerous publications and textbooks. Virtually every book on digital design devotes a substantial number of pages to the analysis of the basic inverter gate. An extensive list of references was presented in Chapter 1. Some references of particular interest that were explicitly quoted in this chapter are given below.
REFERENCES


5.9 Exercises and Design Problems

For all problems, use the device parameters provided in Chapter 3 (as well as the inside back cover), unless otherwise mentioned.

1. [M, SPICE, 3.3.3] The layout of a static CMOS inverter is given in Figure 5.39. (1$\lambda = 0.6$ $\mu$m).
   a. Determine the sizes of the NMOS and PMOS transistor.
   b. Derive the VTC and its parameters ($V_{OH}$, $V_{OL}$, $V_M$, $V_{IH}$, and $V_{IL}$).
   c. Is the VTC affected when the output of the gates is connected to the inputs of 4 similar gates?
CHAPTER 6

DESIGNING COMBINATIONAL LOGIC GATES IN CMOS

In-depth discussion of logic families in CMOS—static and dynamic, pass-transistor, nonratioed and ratioed logic

Optimizing a logic gate for area, speed, energy, or robustness

Low-power and high-performance circuit-design techniques

6.1 Introduction
6.2 Static CMOS Design
  6.2.1 Complementary CMOS
  6.5 Leakage in Low Voltage Systems
  6.2.2 Ratioed Logic
  6.2.3 Pass-Transistor Logic
6.3 Dynamic CMOS Design
  6.3.1 Dynamic Logic: Basic Principles
6.3.2 Speed and Power Dissipation of Dynamic Logic
6.3.3 Issues in Dynamic Design
6.3.4 Cascading Dynamic Gates
6.4 Perspective: How to Choose a Logic Style
6.6 Summary
6.7 To Probe Further
6.8 Exercises and Design Problems
Chapter 6

6.1 Introduction

The design considerations for a simple inverter circuit were presented in the previous chapter. In this chapter, the design of the inverter will be extended to address the synthesis of arbitrary digital gates such as NOR, NAND and XOR. The focus will be on combinational logic (or non-regenerative) circuits that have the property that at any point in time, the output of the circuit is related to its current input signals by some Boolean expression (assuming that the transients through the logic gates have settled). No intentional connection between outputs and inputs is present.

In another class of circuits, known as sequential or regenerative circuits — to be discussed in a later chapter —, the output is not only a function of the current input data, but also of previous values of the input signals (Figure 6.1). This is accomplished by connecting one or more outputs intentionally back to some inputs. Consequently, the circuit “remembers” past events and has a sense of history. A sequential circuit includes a combinational logic portion and a module that holds the state. Example circuits are registers, counters, oscillators, and memory.

![Figure 6.1 High level classification of logic circuits.](image)

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

6.2 Static CMOS Design

The most widely used logic style is static complementary CMOS. The static CMOS style is really an extension of the static CMOS inverter to multiple inputs. In review, the primary advantage of the CMOS structure is robustness (i.e, low sensitivity to noise), good performance, and low power consumption (with no static power consumption). As we will
see, most of those properties are carried over to large fan-in logic gates implemented using the same circuit topology.

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

In this section, we sequentially address the design of various static circuit flavors including complementary CMOS, ratioed logic (pseudo-NMOS and DCVSL), and pass-transistor logic. The issues of scaling to lower power supply voltages and threshold voltages will also be dealt with.

6.2.1 Complementary CMOS

A static CMOS gate is a combination of two networks, called the pull-up network (PUN) and the pull-down network (PDN) (Figure 6.2). The figure shows a generic \( N \) input logic gate where all inputs are distributed to both the pull-up and pull-down networks. The function of the PUN is to provide a connection between the output and \( V_{DD} \) anytime the output of the logic gate is meant to be 1 (based on the inputs). Similarly, the function of the PDN is to connect the output to \( V_{ss} \) when the output of the logic gate is meant to be 0. The PUN and PDN networks are constructed in a mutually exclusive fashion such that one and only one of the networks is conducting in steady state. In this way, once the transients have settled, a path always exists between \( V_{DD} \) and the output \( F \), realizing a high output (“one”), or, alternatively, between \( V_{ss} \) and \( F \) for a low output (“zero”). This is equivalent to stating that the output node is always a low-impedance node in steady state.

In constructing the PDN and PUN networks, the following observations should be kept in mind:

![Figure 6.2 Complementary logic gate as a combination of a PUN (pull-up network) and a PDN (pull-down network).](image-url)
• A transistor can be thought of as a switch controlled by its gate signal. An NMOS switch is on when the controlling signal is high and is off when the controlling signal is low. A PMOS transistor acts as an inverse switch that is on when the controlling signal is low and off when the controlling signal is high.

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

![Diagram](image)

**Figure 6.3** Simple examples illustrate why an NMOS should be used as a pull-down transistor, while a PMOS should be used as a pull-up device.

• A set of construction rules can be derived to construct logic functions (Figure 6.4). NMOS devices connected in series corresponds to an AND function. With all the inputs high, the series combination conducts and the value at one end of the chain is transferred to the other end. Similarly, NMOS transistors connected in parallel represent an OR function. A conducting path exists between the output and input terminal if at least one of the inputs is high. Using similar arguments, construction rules for PMOS networks can be formulated. A series connection of PMOS conducts if both
Section 6.2 Static CMOS Design

201 inputs are low, representing a NOR function \((\overline{A \cdot B} = \overline{A} + \overline{B})\), while PMOS transistors in parallel implement a NAND \((\overline{A + B} = \overline{A} \cdot \overline{B})\).

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

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

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

**Example 6.1 Two input NAND Gate**

Figure 6.5 shows a two-input NAND gate \((F = \overline{A \cdot B})\). The PDN network consists of two NMOS devices in series that conduct when both \(A\) and \(B\) are high. The PUN is the dual network, and consists of two parallel PMOS transistors. This means that \(F\) is 1 if \(A = 0\) or \(B = 0\), which is equivalent to \(F = \overline{A + B}\). The truth table for the simple two input NAND gate is given in Table 6.1. It can be verified that the output \(F\) is always connected to either \(V_{DD}\) or \(GND\), but never to both at the same time.

**Example 6.2 Synthesis of complex CMOS Gate**

Using complementary CMOS logic, consider the synthesis of a complex CMOS gate whose function is \(F = \overline{D + A \cdot (B + C)}\). The first step in the synthesis of the logic gate is to derive the pull-down network as shown in Figure 6.6a by using the fact that NMOS devices in series implements the AND function and parallel device implements the OR function. The next step is to use duality to derive the PUN in a hierarchical fashion. The PDN network is broken into smaller networks (i.e., subset of the PDN) called sub-nets that simplify the derivation of the PUN. In Figure 6.6b, the subnets (SN) for the pull-down network are identified. At the top level, SN1 and SN2 are in parallel so in the dual network, they will be in series. Since SN1
consists of a single transistor, it maps directly to the pull-up network. On the other hand, we need to recursively apply the duality rules to SN2. Inside SN2, we have SN3 and SN4 in series so in the PUN they will appear in parallel. Finally, inside SN3, the devices are in parallel so they will appear in series in the PUN. The complete gate is shown in Figure 6.6c. The reader can verify that for every possible input combination, there always exists a path to either \(V_{DD}\) or \(GND\).

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 6.5** Two-input NAND gate in complementary static CMOS style.

**Figure 6.6** Complex complementary CMOS gate.

**Static Properties of Complementary CMOS Gates**

Complementary CMOS gates inherit all the nice properties of the basic CMOS inverter, discussed earlier. They exhibit rail to rail swing with \(V_{OH} = V_{DD}\) and \(V_{OL} = GND\). The circuits also have no static power dissipation, since the circuits are designed such that the pull-down and pull-up networks are mutually exclusive. The analysis of the DC voltage transfer characteristics and the noise margins is more complicated than for the inverter, as these parameters depend upon the data input patterns applied to gate.

Consider the static two-input NAND gate shown in Figure 6.7. Three possible input combinations switch the output of the gate from high-to-low: (a) \(A = B = 0 \rightarrow 1\), (b) \(A = 1\),
Section 6.2 Static CMOS Design

$B = 0 \rightarrow 1$, and (c) $B = 1, A = 0 \rightarrow 1$. The resulting voltage transfer curves display significant differences. The large variation between case (a) and the others (b & c) is explained by the fact that in the former case both transistors in the pull-up network are on simultaneously for $A=B=0$, representing a strong pull-up. In the latter cases, only one of the pull-up devices is on. The VTC is shifted to the left as a result of the weaker PUN.

The difference between (b) and (c) results mainly from the state of the internal node $\text{int}$ between the two NMOS devices. For the NMOS devices to turn on, both gate-to-source voltages must be above $V_{Tn}$, with $V_{GS2} = V_A - V_{DS1}$ and $V_{GS1} = V_B$. The threshold voltage of transistor $M_2$ will be higher than transistor $M_1$ due to the body effect. The threshold voltages of the two devices are given by:

$$V_{Tn2} = V_{tn0} + \gamma((\sqrt{2\phi_f} + V_{\text{int}}) - \sqrt{2\phi_f})$$

(6.1)

$$V_{Tn1} = V_{tn0}$$

(6.2)

For case (b), $M_3$ is turned off, and the gate voltage of $M_2$ is set to $V_{DD}$. To a first order, $M_2$ may be considered as a resistor in series with $M_1$. Since the drive on $M_2$ is large, this resistance is small and has only a small effect on the voltage transfer characteristics. In case (c), transistor M1 acts as a resistor, causing body effect in $M_2$. The overall impact is quite small as seen from the plot.

### Design Consideration

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

**Propagation Delay of Complementary CMOS Gates**

The computation of propagation delay proceeds in a fashion similar to the static inverter. For the purpose of delay analysis, each transistor is modeled as a resistor in series with an ideal switch. The value of the resistance is dependent on the power supply voltage and an equivalent large signal resistance, scaled by the ratio of device width over length, must be used. The logic is transformed into an equivalent RC network that includes the effect of internal node capacitances. Figure 6.8 shows the two-input NAND gate and its equivalent RC switch level model. Note that the internal node capacitance $C_{int}$—attributable to the source/drain regions and the gate overlap capacitance of $M_2/M_1$—is included. While complicating the analysis, the capacitance of the internal nodes can have quite an impact in some networks such as large fan-in gates.

![Two-input NAND gate](image)

**Figure 6.8** Equivalent RC model for a NAND gate.

We will initially ignore the effect of the internal capacitance (for a first pass). The most important observation is that delay is also dependent on the input patterns. Consider for instance the low-to-high transition. Three possible input scenarios can be identified for charging the output to $V_{DD}$. If both inputs are driven low, the two PMOS devices are on. The delay in this case is $0.69 \times (R_{p}/2) \times C_L$, since the two resistors are in parallel. This is not the worst-case low-to-high transition, which occurs when only one device turns on, and is given by $0.69 \times R_{p} \times C_L$. For the pull-down path, the output is discharged only if both $A$ and $B$ are switched high, and the delay is given by $0.69 \times (2R_{N}) \times C_L$ to a first order. In other words, adding devices in series slows down the circuit, and devices must be made wider to avoid a performance penalty. When sizing the transistors in a gate with multiple fan-in’s, we should pick the combination of inputs that triggers the worst-case conditions.

For example, for a NAND gate to have the same pull-down delay ($t_{phl}$) as a minimum sized inverter (NMOS: $0.375\mu m/0.25\mu m$ and PMOS: $1.125\mu m/0.25\mu m$), the
NMOS devices in the NAND stack must be made twice as large (i.e., NMOS of NAND should be 0.75μm/0.25μm) so that the equivalent resistance the NAND pull-down is the same as the inverter. The PMOS device can remain unchanged.

This first-order analysis assumes that the extra capacitance introduced by widening the transistors can be ignored. This is not a good assumption in general, but allows for a reasonable first cut at device sizing.

**Example 6.3 Delay dependence on input patterns**

Consider the NAND gate of Figure 6.8a. Assume NMOS and PMOS devices of 0.5μm/0.25μm and 0.75μm/0.25μm, respectively. This sizing should result in approximately equal worst-case rise and fall times (since the effective resistance of the pull-down is designed to be equal to the pull-up resistance).

Figure 6.9 shows the simulated low-to-high delay for different input patterns. As expected, the case where both inputs transition go low (A = B = 1→0) results in a smaller delay, compared to the case where only one input is driven low. Notice that the worst-case low-to-high delay depends upon which input (A or B) goes low. The reason for this involves the internal node capacitance of the pull-down stack (i.e., the source of M2). For the case that B = 1 and A transitions from 1→0, the pull-up PMOS device only has to charge up the output node capacitance since M2 is turned off. On the other hand, for the case where A = 1 and B transitions from 1→0, the pull-up PMOS device has to charge up the sum of the output and the internal node capacitances, which slows down the transition.

The table in Figure 6.9 shows a compilation of various delays for this circuit. The first-order transistor sizing indeed provides approximately equal rise and fall delays. An important point to note is that the high-to-low propagation delay depends on the state of the internal nodes. For example, when both inputs transition from 0→1, it is important to establish the state of the internal node. The worst-case happens when the internal node is charged up to $V_{DD} - V_TN$. The worst case can be ensured by pulsing the A input from 1 →0→1, while input B only makes the 0→1. In this way, the internal node is initialized properly.

The important point to take away from this example is that estimation of delay can be fairly complex, and requires a careful consideration of internal node capacitances and data patterns. Care must be taken to model the worst-case scenario in the simulations. A brute force approach that applies all possible input patterns, may not always work as it is important to consider the state of internal nodes.

<table>
<thead>
<tr>
<th>Input Data Pattern</th>
<th>Delay (pS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = B = 0→1</td>
<td>69</td>
</tr>
<tr>
<td>A = 1, B = 0→1</td>
<td>62</td>
</tr>
<tr>
<td>A = 0→1, B = 1</td>
<td>50</td>
</tr>
<tr>
<td>A = B = 1→0</td>
<td>35</td>
</tr>
<tr>
<td>A = 1, B = 1→0</td>
<td>76</td>
</tr>
<tr>
<td>A = 1→0, B = 1</td>
<td>57</td>
</tr>
</tbody>
</table>
The CMOS implementation of a NOR gate \( F = A + B \) is shown in Figure 6.10. The output of this network is high, if and only if both inputs \( A \) and \( B \) are low. The worst-case pull-down transition happens when only one of the NMOS devices turns on (i.e., if either \( A \) or \( B \) is high). Assume that the goal is to size the NOR gate such that it has approximately the same delay as an inverter with the following device sizes: NMOS 0.5\( \mu \)m/0.25\( \mu \)m and PMOS 1.5\( \mu \)m/0.25\( \mu \)m. Since the pull-down path in the worst case is a single device, the NMOS devices \((M_1 \text{ and } M_2)\) can have the same device widths as the NMOS device in the inverter. For the output to be pulled high, both devices must be turned on. Since the resistances add, the devices must be made two times larger compared to the PMOS in the inverter (i.e., \( M_3 \) and \( M_4 \) must have a size of 3\( \mu \)m/0.25\( \mu \)m). Since PMOS devices have a lower mobility relative to NMOS devices, stacking devices in series must be avoided as much as possible. A NAND implementation is clearly preferred over a NOR implementation for implementing generic logic.

![Figure 6.10](image)

**Figure 6.10** Sizing of a NOR gate to produce the same delay as an inverter with size of NMOS: 0.5\( \mu \)m/0.25\( \mu \)m and PMOS: 1.5\( \mu \)m/0.25\( \mu \)m.

**Problem 6.1 Transistor Sizing in Complementary CMOS Gates**

Determine the transistor sizes of the individual transistors in Figure 6.6c such that it has approximately the same \( t_{plh} \) and \( t_{phl} \) as an inverter with the following sizes: NMOS: 0.5\( \mu \)m/0.25\( \mu \)m and PMOS: 1.5\( \mu \)m/0.25\( \mu \)m.

So far in the analysis of propagation delay, we have ignored the effect of internal node capacitances. This is often a reasonable assumption for a first-order analysis. However, in more complex logic gates that have large fan-in, the internal node capacitances can become significant. Consider a 4-input NAND gate as shown in Figure 6.11, which shows the equivalent RC model of the gate, including the internal node capacitances. The internal capacitances consist of the junction capacitance of the transistors, as well as the gate-to-source and gate-to-drain capacitances. The latter are turned into capacitances to ground using the Miller equivalence. The delay analysis for such a circuit involves solving distributed RC networks, a problem we already encountered when analyzing the delay of interconnect networks. Consider the pull-down delay of the circuit. The output is discharged when all inputs are driven high. The proper initial conditions must be placed on the internal nodes (this is, the internal nodes must be charged to \( V_{DD} \) before the inputs are driven high.)
The propagation delay can be computed using the Elmore delay model and is approximated as:

\[ t_{pHL} = 0.69 (R_1 \cdot C_1 + (R_1 + R_2) \cdot C_2 + (R_1 + R_2 + R_3) \cdot C_3 + (R_1 + R_2 + R_3 + R_4) \cdot C_L) \]  

(6.3)

Notice that the resistance of \( M_i \) appears in all the terms, which makes this device especially important when attempting to minimize delay. Assuming that all NMOS devices have an equal size, Eq. (6.3) simplifies to

\[ t_{pHL} = 0.69 R_N (C_1 + 2 \cdot C_2 + 3 \cdot C_3 + 4 \cdot C_L) \]  

(6.4)

**Example 6.4 A Four-Input Complementary CMOS NAND Gate**

In this example, the intrinsic propagation delay of the 4 input NAND gate (without any loading) is evaluated using hand analysis and simulation. Assume that all NMOS devices have a \( W/L \) of 0.5\( \mu \)m/0.25\( \mu \)m, and all PMOS devices have a device size of 0.375\( \mu \)m/0.25\( \mu \)m. The layout of a four-input NAND gate is shown in Figure 6.12. The devices are sized such that the worst case rise and fall time are approximately equal (to first order ignoring the internal node capacitances).

Using techniques similar to those employed for the CMOS inverter in Chapter 3, the capacitances values can be computed from the layout. Notice that in the pull-up path, the PMOS devices share the drain terminal in order to reduce the overall parasitic contribution to the output. Using our standard design rules, the area and perimeter for various devices can be easily computed as shown in Table 6.1.

In this example, we will focus on the pull-down delay, and the capacitances will be computed for the high-to-low transition at the output. While the output make a transition from \( V_{DD} \) to 0, the internal nodes only transition from \( V_{DD} - V_{th} \) to GND. We would need to linearize the internal junction capacitances for this voltage transition, but, to simplify the analysis, we will use the same \( K_{eq} \) for the internal nodes as for the output node.

It is assumed that the output connects to a single, minimum-size inverter. The effect of intracell routing, which is small, is ignored. The different contributions are summarized in Table 6.2. For the NMOS and PMOS junctions, we use \( K_{eq} = 0.57, K_{eqw} = 0.61 \), and \( K_{eq} = 0.79, K_{eqw} = 0.86 \), respectively. Notice that the gate-to-drain capacitance is multiplied by a factor of two for all internal nodes and the output node to account for the Miller effect (this ignores the fact that the internal nodes have a slightly smaller swing due to the threshold drop).
Using Eq. (6.4), we can compute the propagation delay as:

$$t_{pHL} = 0.69 \left( \frac{13K\Omega}{2} \right) (0.85fF + 2 \cdot 0.85fF + 3 \cdot 0.85fF + 4 \cdot 3.47fF) = 85ps \quad (6.5)$$

The simulated delay for this particular transition was found to be 86 psec! The hand analysis gives a fairly accurate estimate given all assumptions and linearizations made. For example, we assume that the gate-source (or gate-drain) capacitance only consists of the overlap component. This is not entirely the case, as during the transition some other contributions come in place depending upon the operating region. Once again, the goal of hand analysis is not to provide a totally accurate delay prediction, but rather to give intuition into what factors influence the delay and to aide in initial transistor sizing. Accurate timing analysis and transistor optimization is usually done using SPICE. The simulated worst-case low-to-high delay time for this gate was 106ps.

While complementary CMOS is a very robust and simple approach for implementing logic gates, there are two major problems associated with using this style as the com-

---

**Table 6.1** Area and perimeter of various transistors for 4 input NAND gate.

<table>
<thead>
<tr>
<th>Transistor</th>
<th>W (µm)</th>
<th>AS (µm²)</th>
<th>AD (µm²)</th>
<th>PS (µm)</th>
<th>PD(µm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.3125</td>
<td>0.0625</td>
<td>1.75</td>
<td>0.25</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>0.0625</td>
<td>0.0625</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>0.0625</td>
<td>0.0625</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
<td>0.0625</td>
<td>0.3125</td>
<td>0.25</td>
<td>1.75</td>
</tr>
<tr>
<td>5</td>
<td>0.375</td>
<td>0.296875</td>
<td>0.171875</td>
<td>1.875</td>
<td>0.875</td>
</tr>
<tr>
<td>6</td>
<td>0.375</td>
<td>0.171875</td>
<td>0.171875</td>
<td>0.875</td>
<td>0.875</td>
</tr>
<tr>
<td>7</td>
<td>0.375</td>
<td>0.171875</td>
<td>0.171875</td>
<td>0.875</td>
<td>0.875</td>
</tr>
<tr>
<td>8</td>
<td>0.375</td>
<td>0.296875</td>
<td>0.171875</td>
<td>1.875</td>
<td>0.875</td>
</tr>
</tbody>
</table>

**Figure 6.12** Layout a four-input NAND gate in complementary CMOS.
plexity of the gate (i.e., fan-in) increases. First, the number of transistors required to implement an N fan-in gate is $2N$. This can result in significant implementation area. The second problem is that propagation delay of a complementary CMOS gate deteriorates rapidly as a function of the fan-in. The large number of transistors ($2N$) increases the overall capacitance of the gate. For an N-input NAND gate, the output capacitance increases linearly with the fan-in since the number of PMOS devices connected to the output node increases linearly with the fan-in. Also, a series connection of transistors in either the PUN or PDN slows the gate as well, because the effective (dis)charging resistance is increased. For the same N-input NAND gate, the effective resistance of the PDN path increases linearly with the fan-in. Since the output capacitance increase linearly and the pull-down resistance increases linearly, the high-to-low delay can increase in a quadratic fashion.

The fan-out has a large impact on the delay of complementary CMOS logic as well. Each input to a CMOS gate connects to both an NMOS and a PMOS device, and presents a load to the driving gate equal to the sum of the gate capacitances.

The above observations are summarized by the following formula, which approximates the influence of fan-in and fan-out on the propagation delay of the complementary CMOS gate

$$t_p = a_1FI + a_2F^2 + a_3FO$$ (6.6)

where $FI$ and $FO$ are the fan-in and fan-out of the gate, respectively, and $a_1$, $a_2$ and $a_3$ are weighting factors that are a function of the technology.

At first glance, it would appear that the increase in resistance for larger fan-in can be solved by making the devices in the transistor chain wider. Unfortunately, this does not improve the performance as much as expected, since widening a device also increases its gate and diffusion capacitances, and has an adverse affect on the gate performance. For the N-input NAND gate, the low-to-high delay only increases linearly since the pull-up resistance remains unchanged and only the capacitance increases linearly.

Table 6.2 Computation of capacitances (for high-to-low transition at the output). The circuit shows the intrinsic delay of the gate with no extra loading. Any fan-out capacitance would simply be added to the $C_L$ term.

<table>
<thead>
<tr>
<th>Capacitor</th>
<th>Contributions (H→L)</th>
<th>Value (fF) (H→L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>$C_{d1} + C_{s2} + 2 \cdot C_{gd1} + 2 \cdot C_{gs2}$</td>
<td>$(0.57 \cdot 0.0625 \cdot 2 + 0.61 \cdot 0.25 \cdot 0.28) + 2 \cdot (0.31 \cdot 0.5) + 2 \cdot (0.31 \cdot 0.5) = 0.85 fF$</td>
</tr>
<tr>
<td>$C_2$</td>
<td>$C_{d2} + C_{s3} + 2 \cdot C_{gd2} + 2 \cdot C_{gs3}$</td>
<td>$(0.57 \cdot 0.0625 \cdot 2 + 0.61 \cdot 0.25 \cdot 0.28) + 2 \cdot (0.31 \cdot 0.5) + 2 \cdot (0.31 \cdot 0.5) = 0.85 fF$</td>
</tr>
<tr>
<td>$C_3$</td>
<td>$C_{d3} + C_{s4} + 2 \cdot C_{gd3} + 2 \cdot C_{gs4}$</td>
<td>$(0.57 \cdot 0.0625 \cdot 2 + 0.61 \cdot 0.25 \cdot 0.28) + 2 \cdot (0.31 \cdot 0.5) + 2 \cdot (0.31 \cdot 0.5) = 0.85 fF$</td>
</tr>
<tr>
<td>$C_L$</td>
<td>$C_{d4} + 2 \cdot C_{gd4} + C_{d5} + C_{d6} + C_{d7} + C_{d8} + 2 \cdot C_{gsd5} + 2 \cdot C_{gsd7} + 2 \cdot C_{gsd8} + 2 \cdot C_{gsd9} + C_{s10}$</td>
<td>$(0.57 \cdot 0.3125 \cdot 2 + 0.61 \cdot 1.75 \cdot 0.28) + 2 \cdot (0.31 \cdot 0.5) + 4 \cdot (0.79 + 0.175 \cdot 0.28) + 4 \cdot (0.79 + 0.175 \cdot 0.28) = 3.47 fF$</td>
</tr>
</tbody>
</table>
Figure 6.13 show the propagation delay for both transitions as a function of fan-in assuming a fixed fan-out (NMOS: 0.5µm and PMOS: 1.5µm). As predicted above, the $t_{pLH}$ increases linearly due to the linearly-increasing value of the output capacitance. The simultaneous increase in the pull-down resistance and the load capacitance results in an approximately quadratic relationship for $t_{pHL}$. Gates with a fan-in greater than or equal to 4 become excessively slow and must be avoided.

Several approaches may be used to reduce delays in large fan-in circuits.

1. **Transistor Sizing**

   The most obvious solution is to increase the overall transistor size. This lowers the resistance of devices in series and lowers the time constant. However, increasing the transistor size, results in larger parasitic capacitors, which do not only affect the propagation delay of the gate in question, but also present a larger load to the preceding gate. This technique should, therefore, be used with caution. If the load capacitance is dominated by the intrinsic capacitance of the gate, widening the device only creates a “self-loading” effect, and the propagation delay is unaffected.

2. **Progressive Transistor Sizing**

   An alternate approach to uniform sizing (in which each transistor is scaled up uniformly), is to use progressive transistor sizing (Figure 6.14). Referring back to Eq. (6.3), we see that the resistance of $M_1$ ($R_1$) appears $N$ times in the delay equation, the resistance of $M_2$ ($R_2$) appears $N$-1 times, etc. From the equation, it is clear that $R_1$ should be made the smallest, $R_2$ the next smallest, etc. Consequently, a progressive scaling of the transistors is beneficial: $M_1 > M_2 > M_3 > ... > M_N$. Basically, in this approach, the important resistance is reduced while reducing capacitance. For an excellent treatment on the optimal sizing of transistors in a complex network, we refer the interested reader to [Shoji88, pp. 131–143].

3. **Input Re-Ordering**
Section 6.2 Static CMOS Design

Some signals in complex combinational logic blocks might be more critical than others. Not all inputs of a gate arrive at the same time (due, for instance, to the propagation delays of the preceding logical gates). An input signal to a gate is called critical if it is the last signal of all inputs to assume a stable value. The path through the logic which determines the ultimate speed of the structure is called the critical path.

Putting the critical-path transistors closer to the output of the gate can result in a speed-up. This is demonstrated in Figure 6.15. Signal \( \text{In}_1 \) is assumed to be a critical signal. Suppose further that \( \text{In}_2 \) and \( \text{In}_3 \) are high and that \( \text{In}_1 \) undergoes a \( 0 \rightarrow 1 \) transition. Assume also that \( C_L \) is initially charged high. In case (a), no path to GND exists until \( M_1 \) is turned on, which is unfortunately the last event to happen. The delay between the arrival of \( \text{In}_1 \) and the output is therefore determined by the time it takes to discharge \( C_L \), \( C_1 \) and \( C_2 \). In the second case, \( C_1 \) and \( C_2 \) are already discharged when \( \text{In}_1 \) changes. Only \( C_L \) still has to be discharged, resulting in a smaller delay.

### 4. Logic Restructuring

Manipulating the logic equations can reduce the fan-in requirements and hence reduce the gate delay, as illustrated in Figure 6.16. The quadratic dependency of the gate delay on fan-in makes the six-input NOR gate extremely slow. Partitioning the NOR-gate into two three-input gates results in a significant speed-up, which offsets by far the extra delay incurred by turning the inverter into a two-input NAND gate.
Power Consumption in CMOS Logic Gates

The sources of power consumption for the complementary CMOS inverter was discussed in detail. Many of issues apply directly to complex CMOS gates. The power dissipation is a strong function of transistor sizing (which affects physical capacitance), input and output rise/fall times (which affects the short-circuit power), device thresholds and temperature (which affect leakage power) and switching activity. The switching power of a CMOS gate is given by $\alpha_{0\rightarrow1} C_L V_{DD}^2 f$ and this section will focus on the switching activity ($\alpha_{0\rightarrow1}$) of a logic gate. There are two components to switching activity: a static component (which does not take into account the timing behavior) and a dynamic (or glitching) component (which takes into account the timing behavior of the circuit). The major factors that affect activity is listed below.

Logic Function — The amount of transition activity is a strong function of the logic function being implemented. In static CMOS gates, the static transition probability assuming independent inputs is the probability that the output will be in the zero state in one cycle multiplied by the probability that the output will be in the one state in the next cycle:

$$\alpha_{0\rightarrow1} = p_0 \cdot p_1 = p_0 \cdot (1 - p_0)$$ (6.7)

where $p_0$ is the probability that the output is in the zero state and $p_1$ is the probability that the output will is in the one state. Assuming that the inputs are independent and uniformly distributed, any $N$-input static gate will have a transition probability that corresponds to:

$$\alpha_{0\rightarrow1} = \frac{N_0}{2^N} \cdot \frac{N_1}{2^N} = \frac{N_0 \cdot (2^N - N_0)}{2^{2N}}$$ (6.8)

where $N_0$ is the number of zero entries and $N_1$ is the number of one entries in the truth table for the output of the $N$-input function. To illustrate, consider a static 2-input NOR gate whose truth table is shown in Table 6.3. Assume that only one input transition is possible during a clock cycle and that the inputs to the NOR gate have a uniform input distribution (i.e., the four possible states for inputs $A$ and $B$ (00, 01, 10, 11) are equally likely).

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$Out$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
From Table 6.3 and Eq. (6.8), the output transition probability of a 2-input static CMOS NOR gate is given by:

\[
\alpha_{0 \rightarrow 1} = \frac{N_0 \cdot (2^N - N_0)}{2^{2N}} = \frac{3 \cdot (2^2 - 3)}{2^3 \cdot 2} = \frac{3}{16} \tag{6.9}
\]

**Problem 6.2** \(N\) input XOR gate

Assuming the inputs to an \(N\)-input XOR gate are uncorrelated and uniformly distributed, derive the expression for the switching activity factor.

**Signal Statistics**—The switching activity of a logic gate is a strong function of the signal statistics. Using a uniform input distribution to compute activity is not a good one since the propagation through logic gates can significantly modify the signal statistics. For example, consider once again a 2-input static NOR gate, and let \(p_a\) and \(p_b\) be the probabilities that the inputs \(A\) and \(B\) are one. Assume that the inputs are not correlated. The probability that the output node is a one is given by:

\[
p_1 = (1-p_a) (1-p_b) \tag{6.10}
\]

Therefore, the probability of a transition from 0 to 1 is:

\[
\alpha_{0 \rightarrow 1} = p_0 p_1 = (1-(1-p_a) (1-p_b)) (1-p_a) (1-p_b) \tag{6.11}
\]
Figure 6.17 shows the transition probability as a function of $p_a$ and $p_b$. Observe how this graph degrades into the simple inverter case when one of the input probabilities is set to 0. From this plot, it is clear that understanding the signal statistics and their impact on switching events can be used to significantly impact the power dissipation.

Problem 6.3  Power Dissipation of Basic Logic Gates

Derive the $0 \rightarrow 1$ output transition probabilities for the basic logic gates (AND, OR, XOR). The results to be obtained are given in Table 6.4.

Table 6.4  Output transition probabilities for static logic gates.

<table>
<thead>
<tr>
<th></th>
<th>$\alpha_{0\rightarrow1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>$(1-p_a p_b)p_a p_b$</td>
</tr>
<tr>
<td>OR</td>
<td>$(1-p_a)(1-p_a)(1-p_a)(1-p_b)$</td>
</tr>
<tr>
<td>XOR</td>
<td>$[1-(p_a + p_b - 2p_a p_b)p_a p_b]$</td>
</tr>
</tbody>
</table>

Inter-signal Correlations—The evaluation of the switching activity is further complicated by the fact that signals exhibit correlation in space and time. Even if the primary inputs to a logic network are uncorrelated, the signals become correlated or 'colored', as they propagate through the logic network. The example of Figure 6.18 provides a simple example. Consider the circuit shown in Figure 6.18a, and assume that the primary inputs, $A$ and $B$, are uncorrelated and uniformly distributed. Node $C$ has a 1 (0) probability of 1/2, and a 0->1 transition probability of 1/4. The probability that the node $Z$ undergoes a power consuming transition is then determined using the AND-gate expression of Table 6.4.

$$p_{0\rightarrow1} = (1-p_a p_b)p_a p_b = (1-1/2 \cdot 1/2) 1/2 \cdot 1/2 = 3/16$$  (6.12)

The computation of the probabilities is straightforward: signal and transition probabilities are evaluated in an ordered fashion, progressing from the input to the output node. This approach, however, has two major limitations: (1) it does not deal with circuits with feedback, as found in sequential circuits; (2) it assumes that the signal probabilities at the input of each gate are independent. This is rarely the case in actual circuits, where reconvergent fanout often causes inter-signal dependencies. For instance, the inputs to the AND gate in Figure 6.18b ($C$ and $B$) are interdependent, as both are a function of $A$. The

![Figure 6.18](image-url)  
**Figure 6.18** Example illustrating the effect of signal correlations.
approach to compute probabilities, presented previously, fails under these circumstances. Traversing from inputs to outputs yields a transition probability of 3/16 for node Z, similar to the previous analysis. This value for transition probability is clearly false, as logic transformations show that the network can be reduced to $Z = C \cdot B = A \cdot \overline{A} = 0$, and no transition will ever take place.

To get the precise results in the progressive analysis approach, it is essential to take signal inter-dependencies into account. This can be accomplished with the aid of conditional probabilities. For an AND gate, $Z$ equals 1 if and only if $B$ and $C$ are equal to 1.

$$p_Z = p(Z=1) = p(B=1, C=1)$$

where $p(B=1, C=1)$ represents the probability that $B$ and $C$ are equal to 1 simultaneously. If $B$ and $C$ are independent, $p(B=1, C=1)$ can be decomposed into $p(B=1) \cdot p(C=1)$, and this yields the expression for the AND-gate, derived earlier: $p_Z = p(B=1) \cdot p(C=1) = p_B p_C$. If a dependency between the two exists (as is the case in Figure 6.18b), a conditional probability has to be employed, such as

$$p_Z = p(C=1|B=1) \cdot p(B=1)$$

The first factor in Eq. (6.14) represents the probability that $C=1$ given that $B=1$. The extra condition is necessary as $C$ is dependent upon $B$. Inspection of the network shows that this probability is equal to 0, since $C$ and $B$ are logical inversions of each other, resulting in the signal probability for $Z$, $p_Z = 0$.

Deriving those expressions in a structured way for large networks with reconvergent fanout is complex, especially when the networks are contain feedback loops. Computer support is therefore essential. To be meaningful, the analysis program has to process a typical sequence of input signals, as the power dissipation is a strong function of statistics of those signals.

**Dynamic or Glitching Transitions**—When analyzing the transition probabilities of complex, multistage logic networks in the preceding section, we ignored the fact that the gates have a non-zero propagation delay. In reality, the finite propagation delay from one logic block to the next can cause spurious transitions, called *glitches, critical races, or dynamic hazards*, to occur: a node can exhibit multiple transitions in a single clock cycle before settling to the correct logic level.

A typical example of the effect of glitching is shown in Figure 6.19, which displays the simulated response of a chain of NAND gates for all inputs going simultaneously from 0 to 1. Initially, all the outputs are 1 since one of the inputs was 0. For this particular transition, all the odd bits must transition to 0 while the even bits remain at the value of 1. However, due to the finite propagation delay, the higher order even outputs start to discharge and the voltage drops. When the correct input ripples through the network, the output goes high. The glitch on the even bits causes extra power dissipation beyond what is required to strictly implement the logic function. Although the glitches in this example are only partial (i.e., not from rail to rail), they contribute significantly to the power dissipation. Long chains of gates often occur in important structures such as adders and multipliers and the glitching component can easily dominate the overall power consumption.
The dynamic power of a logic gate can be reduced by minimizing the physical capacitance and the switching activity. The physical capacitance can be minimized in a number ways, including circuit style selection, transistor sizing, placement and routing, and architectural optimizations. The switching activity, on the other hand, can be minimized at all level of the design abstraction, and is the focus of this section. Logic structures can be optimized to minimize both the fundamental transitions required to implement a given function, and the spurious transitions. This can be accomplished in the following ways:

1. **Logic Restructuring**

   The topology of a logic network can affect the overall power dissipation. To illustrate this point consider two alternate implementations of \( F = A \cdot B \cdot C \cdot D \), as shown in Figure 6.20. Ignore glitching and assume that all primary inputs \((A,B,C,D)\) are uncorre-

---

**Figure 6.19** Glitching in a chain of NAND gates.

**Figure 6.20** Simple example to demonstrate the influence of circuit topology on activity.
lated and uniformly distributed (i.e., $p_{1(a,b,c,d)} = 0.5$). For an AND gate, the probability that the output is 1 is $p_1 = p_x p_y$ and the transition probability is:

$$\alpha_{0\rightarrow1} = p_0 p_1 = p_0 (1-p_0) = (1-p_x p_y) p_x p_y$$  \hspace{1cm} (6.15)

Given this, the activity can be computed for the two topologies as shown in Table 6.5. The results indicate that the chain implementation will have an overall lower switching activity than the tree implementation for random inputs. However, as mentioned before, it is also important to consider the timing behavior to accurately make power trade-offs. In this example the tree topology will have lower (no) glitching activity since the signal paths are balanced to all the gates.

**Table 6.5** Probabilities for tree and chain topologies.

<table>
<thead>
<tr>
<th></th>
<th>$O1$</th>
<th>$O2$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$ (chain)</td>
<td>$1/4$</td>
<td>$1/8$</td>
<td>$1/16$</td>
</tr>
<tr>
<td>$p_0 = 1-p_1$ (chain)</td>
<td>$3/4$</td>
<td>$7/8$</td>
<td>$15/16$</td>
</tr>
<tr>
<td>$p_{0\rightarrow1}$ (chain)</td>
<td>$3/16$</td>
<td>$7/64$</td>
<td>$15/256$</td>
</tr>
<tr>
<td>$p_1$ (tree)</td>
<td>$1/4$</td>
<td>$1/4$</td>
<td>$1/16$</td>
</tr>
<tr>
<td>$p_0 = 1-p_1$ (tree)</td>
<td>$3/4$</td>
<td>$3/4$</td>
<td>$15/16$</td>
</tr>
<tr>
<td>$p_{0\rightarrow1}$ (tree)</td>
<td>$3/16$</td>
<td>$3/16$</td>
<td>$15/256$</td>
</tr>
</tbody>
</table>

2. **Input ordering**

Consider the two static logic circuits of Figure 6.21. The probabilities of $A$, $B$ and $C$ being 1 is listed in the figure. Since both circuits implement identical logic functionality, it is obvious that the activity at the output node $Z$ is equal in both cases. The difference is the activity at the intermediate node. In the first circuit, this activity equals $(1 - 0.5 \times 0.2) (0.5 \times 0.2) = 0.09$. In the second case, the probability that a 0 $\rightarrow$ 1 transition occurs equals $(1 - 0.2 \times 0.1) (0.2 \times 0.1) = 0.0196$. This is substantially lower. From this we learn that it is beneficial to postpone the introduction of signals with a high transition rate (i.e., signals with a signal probability close to 0.5). A simple reordering of the input signals is often sufficient to accomplish that goal.

![Figure 6.21 Reordering of inputs affects the circuit activity.](image)

3. **Time-multiplexing resources**
Another important design consideration is the amount of resources required to implement a given function. To conserve area, it is often desirable to minimize the amount of physical hardware (logic units or data busses). Unfortunately, the minimum area solution does not always result in the lowest switching activity. For example, consider the transmission of two input bits ($A$ and $B$) using dedicated resources and a time-multiplexed approach as shown in Figure 6.22. To first order, it would seem that the degree of time-multiplexing should not affect the overall switched capacitance since the time-multiplexed solution has half the capacitance switched at twice the frequency (for a fixed throughput).

If data being transmitted were random, it will make no difference what architecture is used. However if data is not correlated, the power dissipation of the time-multiplexed solution can be significantly higher. For example, suppose $A$ was mostly low and $B$ was mostly high. In the parallel solution, the switched capacitance should be very low since there are very few transitions on the data bits. However, in the time-multiplexed solution, the bus is going to toggle between 0 and 1. Care must be taken in digital systems to avoid time-multiplexing data stream that are not correlated.

4. Glitch Reduction by balancing signal paths

The occurrence of glitching in a circuit is mainly due to a mismatch in the path lengths in the network. If all input signals of a gate change simultaneously, no glitching occurs. On the other hand, if input signals change at different times, a dynamic hazard might develop. Such a mismatch in signal timing is typically the result of different path lengths with respect to the primary inputs of the network. This is illustrated in Figure 4.20.

Assume that the XOR gate has a unit delay. The first network (a) suffers from glitching as a result of the wide disparity between the arrival times of the input signals for a gate. For example, for gate $F_3$, one input settles at time 0, while the second one only arrives at time
2. Redesigning the network so that all arrival times are identical can dramatically reduce the number of transitions (network b).

### 6.2.2 Ratioed Logic

The CMOS logic style described in the previous section is highly robust and scalable with technology, but requires $2N$ transistors to implement a $N$-input logic gate. Also, the load capacitance is significant since each gate drives two devices (a PMOS and an NMOS) per fan-out. Ratioed logic is an attempt to reduce the number of transistors required to implement a given logic function, at the cost of reduced robustness and extra power dissipation.

The purpose of the PUN in complementary CMOS is to provide a conditional path between $V_{DD}$ and the output when the PDN is turned off. In ratioed logic, the entire PUN is replaced with a single load device that pulls up the output when the PDN is turned off.

Figure 6.24 shows an example of ratioed logic which uses a grounded PMOS load and referred to as a pseudo-NMOS style. Instead of a combination of active pull-down and pull-up networks, such a gate consists of an NMOS pull-down network that realizes the logic function, and a simple load device.

The clear advantage of pseudo-NMOS is the reduced number of transistors ($N+1$ vs. $2N$ for complementary CMOS). The nominal high output voltage ($V_{OH}$) for this gate is $V_{DD}$ since the pull-down devices is turned off when the output is pulled high (assuming that $V_{OL}$ is below $V_{Tn}$). On the other hand, the nominal low output voltage is not 0V since there is a fight between the devices in the PDN and the load grounded PMOS device. This results in reduced noise margins and more importantly static power dissipation. The sizing of the load device relative to the pull-down devices can be used to trade-off parameters such as noise margin, propagation delay and power dissipation. Since the voltage swing on the output and overall functionality of the gate is dependent on the device size, the circuit is called ratioed. This is in contrast to the ratioless logic styles, such as complementary CMOS, where the low and high levels do not depend upon transistor sizes.

Computing the dc transfer characteristic of the pseudo-NMOS proceeds along paths similar to those used for its complementary CMOS counterpart. The value of $V_{OL}$ is obtained by equating the currents through the driver and load devices for $V_{in} = V_{DD}$. At
this operation point, it is reasonable to assume that the NMOS device resides in linear mode (since the output should ideally be close to 0V), while the PMOS load is saturated.

\[
k_n(V_{DD} - V_{Tn})V_{OL} - \frac{V^2_{DL}}{2} = k_p(-V_{DD} - V_{Tp}) \cdot V_{DSAT} - \frac{V^2_{DSAT}}{2}
\]  

(6.16)

Assuming that \(V_{OL}\) is small relative to the gate drive \((V_{DD} - V_{T})\) and that \(V_{Tp}\) is equal to \(V_{Tn}\) in magnitude, \(V_{OL}\) can be approximated as:

\[
V_{OL} \approx \frac{k_p(-V_{DD} - V_{Tp}) \cdot V_{DSAT}}{k_n(V_{DD} - V_{Tn})} \approx \frac{\mu_p \cdot W_p}{\mu_n \cdot W_n} \cdot |V_{DSAT}|
\]  

(6.17)

In order to make \(V_{OL}\) as small as possible, the PMOS device should be sized much smaller than the NMOS pull-down devices. Unfortunately, this has a negative impact on the propagation delay for charging up the output node since the current provided by the PMOS device is limited.

An important disadvantage of pseudo-NMOS gates is static power that happens when the output is low, because a direct current path exists between \(V_{DD}\) and GND through the load and driver devices. The static power consumption in the low-output mode is easily derived

\[
P_{low} = V_{DD}I_{low} \approx V_{DD} \cdot k_p(-V_{DD} - V_{Tp}) \cdot V_{DSATp} \cdot \frac{V^2_{DSATp}}{2}
\]  

(6.18)

Example 6.5 Pseudo-NMOS Inverter

Consider a simple pseudo-NMOS inverter (where the PDN network in Figure 6.24 degerates to a single transistor) with an NMOS size of 0.5µm/0.25µm. The effect of sizing the PMOS device is studied in this example to demonstrate the impact on various parameters. The \(W/L\) ratio of the grounded PMOS is varied for values of 4, 2, 1, 0.5 and 0.25. The devices less than \(W/L < 1\) is contructed by making the length longer than the width. The voltage transfer curve for the different sizes is plotted in Figure 6.25.

![Figure 6.25 Voltage transfer curves for sizes of the pseudo-NMOS devices.](image-url)

Table 6.6 summarizes the nominal output voltage \((V_{OL})\), static power dissipation, and the low-to-high propagation delay. The low-to-high delay is measured as the time to
Section 6.2 Static CMOS Design

reach 1.25V from $V_{OL}$ (which is not 0V for this inverter). This is chosen since the load gate is a CMOS inverter with a switching threshold of 1.25V. The trade-off between the static and dynamic properties is clearly illustrated. A larger pull-up device improves performance, but increases static power dissipation and lower noise margins (i.e., higher $V_{OL}$).

Notice that the simple first order model to predict $V_{OL}$ is reasonably valid. For a PMOS $W/L$ of 4, $V_{OL}$ is given by $(30/115) (4) (0.63V) = 0.66V$.

The static power dissipation of pseudo-NMOS has limited its use. However, pseudo-NMOS still finds use in large fan-in circuits. Figure 6.26 shows the implementation of pseudo-NMOS NOR and NAND gates. When area is most important, such an approach is attractive.

<table>
<thead>
<tr>
<th>Size</th>
<th>$V_{OL}$</th>
<th>Static Power Dissipation</th>
<th>$I_{ph}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.693V</td>
<td>564µW</td>
<td>14ps</td>
</tr>
<tr>
<td>2</td>
<td>0.273V</td>
<td>298µW</td>
<td>56ps</td>
</tr>
<tr>
<td>1</td>
<td>0.133V</td>
<td>160µW</td>
<td>123ps</td>
</tr>
<tr>
<td>0.5</td>
<td>0.064V</td>
<td>80µW</td>
<td>268ps</td>
</tr>
<tr>
<td>0.25</td>
<td>0.031V</td>
<td>41µW</td>
<td>569ps</td>
</tr>
</tbody>
</table>

![Table 6.6](image)

![Figure 6.26](image)

Problem 6.4 NAND Versus NOR in Pseudo-NMOS

Given the choice between NOR or NAND logic, which one would you prefer for implementation in pseudo-NMOS?

How to Build Even Better Loads

It is possible to create a ratioed logic style that allows us to completely eliminate static currents and provide rail-to-rail swing. This requires the use of feedback concepts. In this particular style of logic, complementary inputs are fed into the gate and the gates
provide complementary outputs. Such a gate, called *Differential Cascade Voltage Switch Logic* (or DCVSL) is presented conceptually in Figure 6.27a.

The pull-down networks PDN1 and PDN2 are designed using NMOS devices and are mutually exclusive (i.e., when PDN1 conducts, PDN2 is off and when PDN1 is off, PDN2 conducts). The mutually exclusive pull-down devices allow the implementation of the required logic function and its inverse. Assume now that, for a given set of inputs, PDN1 conducts while PDN2 does not. Also assume that $Out$ was initially high and $Out$ initially low. Node $Out$ is pulled down and initially there is a fight between $M_1$ and PDN1 PMOS as the pull-down device is turned on. Notice that initially, $Out$ is actually in a high impedance state since both $M_2$ and PDN2 are turned off. PDN1 must be strong enough to bring $Out$ down to $V_{DD} - |V_{TP}|$, at which point, $M_2$ turns on and charges $Out$ to $V_{DD}$. This in turn enables $Out$ to discharge all the way to GND. The circuit is still ratioed since the sizing of the PMOS devices relative to the pull-down devices is critical to functionality, not just performance. Figure 6.27b shown an example of an XOR/XNOR gate. Notice that it is possible to share transistors among the two pull-down networks.

In addition to the problem of increase complexity in design, this circuit style has the problem of increased power dissipation due to coss-over current. There is a period of time when the PMOS and PDN is turned on simultaneously, producing a short circuit path. However, notice that the static power dissipation has been eliminated since in steady state, one of the pull-down networks and other PMOS device are turned off.

**Example 6.6 DCVSL Transient Response**

An example transient response is shown for an AND/NAND gate in DCVSL. Notice that as $Out$ is pulled down to $V_{DD} - |V_{TP}|$, $Out$ starts to charge up to $V_{DD}$ quickly. The
Section 6.2 Static CMOS Design

delay from the input to Out is 197ps and to \overline{Out} is 321 ps. A static CMOS AND gate (NAND followed by an inverter) has a delay of 200ps.

6.2.3 Pass-Transistor Logic

Pass-Transistor Basics

A popular and widely used alternative to complementary CMOS is pass transistor logic. Pass transistor logic attempts to reduce the number of transistors required to implement logic by allowing the primary inputs to drive gate terminals as well as source/drain terminals [Radhakrishnan85]. This is in contrast to logic families that we have studied so far that only allow primary inputs to drive the gate terminals of MOSFETS. Figure 6.29 shows a transistor level implementation of the AND function constructed using NMOS transistors. In this gate, if the B input is high, the top transistor is turned on and copies the input A to the output F. When input B is low, the bottom pass transistor is turned on and passes a 0. The switch driven by \overline{B} seems to be redundant at first glance. Its presence is essential to ensure that a low-impedance path exists to the supply rails under all circumstances, or, in this particular case, when B is low.

The potential advantage of pass transistor is that a fewer number of transistors are required to implement a given function. For example, the implementation of the AND gate in Figure 6.29b requires 4 transistors (including the inverter required to invert B) while a complementary CMOS implementation would require 6 transistors.

Pass transistor logic uses fewer devices and therefore has lower physical capacitance. Unfortunately, as we have discussed earlier, a NMOS device is effective at passing

A popular and widely used alternative to complementary CMOS is pass transistor logic. Pass transistor logic attempts to reduce the number of transistors required to implement logic by allowing the primary inputs to drive gate terminals as well as source/drain terminals [Radhakrishnan85]. This is in contrast to logic families that we have studied so far that only allow primary inputs to drive the gate terminals of MOSFETS. Figure 6.29 shows a transistor level implementation of the AND function constructed using NMOS transistors. In this gate, if the B input is high, the top transistor is turned on and copies the input A to the output F. When input B is low, the bottom pass transistor is turned on and passes a 0. The switch driven by \overline{B} seems to be redundant at first glance. Its presence is essential to ensure that a low-impedance path exists to the supply rails under all circumstances, or, in this particular case, when B is low.

The potential advantage of pass transistor is that a fewer number of transistors are required to implement a given function. For example, the implementation of the AND gate in Figure 6.29b requires 4 transistors (including the inverter required to invert B) while a complementary CMOS implementation would require 6 transistors.

Pass transistor logic uses fewer devices and therefore has lower physical capacitance. Unfortunately, as we have discussed earlier, a NMOS device is effective at passing

A popular and widely used alternative to complementary CMOS is pass transistor logic. Pass transistor logic attempts to reduce the number of transistors required to implement logic by allowing the primary inputs to drive gate terminals as well as source/drain terminals [Radhakrishnan85]. This is in contrast to logic families that we have studied so far that only allow primary inputs to drive the gate terminals of MOSFETS. Figure 6.29 shows a transistor level implementation of the AND function constructed using NMOS transistors. In this gate, if the B input is high, the top transistor is turned on and copies the input A to the output F. When input B is low, the bottom pass transistor is turned on and passes a 0. The switch driven by \overline{B} seems to be redundant at first glance. Its presence is essential to ensure that a low-impedance path exists to the supply rails under all circumstances, or, in this particular case, when B is low.

The potential advantage of pass transistor is that a fewer number of transistors are required to implement a given function. For example, the implementation of the AND gate in Figure 6.29b requires 4 transistors (including the inverter required to invert B) while a complementary CMOS implementation would require 6 transistors.

Pass transistor logic uses fewer devices and therefore has lower physical capacitance. Unfortunately, as we have discussed earlier, a NMOS device is effective at passing
a 0 but is poor at pulling a node to $V_{DD}$. In pass transistor logic, the pass transistors are used to pass high and low voltages. Therefore, when the pass transistor pulls a node high, the output only charges up to $V_{DD} - V_{Tn}$. In fact, the situation is worsened by the fact that the devices experience body effect since there is a significant source to body voltage when pulling high since the body is tied to GND and the source charge up close to $V_{DD}$.

Consider the case when the pass transistor is charging up a node to $V_{DD}$ where the gate and drain terminals are set at $V_{DD}$. Let the source the NMOS pass transistor be labeled $x$. Node $x$ will charge up to $V_{DD} - V_{Tn}$ where, the threshold must account for body effect as shown in Eq. (6.19). This maximum voltage swing on the output node is given by:

$$V_x = V_{DD} - (V_{th0} + \gamma((\sqrt{2}\phi_f + V_x) - \sqrt{2}\phi_f))$$  \hspace{1cm} (6.19)

**Example 6.7 Voltage swing for pass transistors circuits**

Assuming a power supply voltage of 2.5V, the transient response of Figure 6.30 shows the output of a NMOS charging up (where the drain voltage is at $V_{DD}$ and the gate voltage in is ramped from 0V to $V_{DD}$). Assume that node $x$ was initially 0. Also notice that if $IN$ is low, node $x$ is in a high impedance state (not driven to one of the rails using a low resistance path). Extra transistors can be added to provide a path to GND, but for this discussion, the simplified circuit is sufficient. Notice that the output charges up quickly initially, but has slow tail. This is attributed to the fact that the drive (gate to source voltage) reduces significantly as the output approaches $V_{DD} - V_{Tn}$ and the current available to charge up node $x$ reduces drastically. Hand calculation using Eq. (6.19), results in an output voltage of 1.8V, which comes close to the simulated value.

**WARNING:**

The above example demonstrates that pass transistor gates cannot be cascaded by connecting the output of a pass gate to the gate terminal of another pass transistor. This is illustrated by the simple example of Figure 6.31. In Figure 6.31a, the output of $M_i$ (node $x$) drives the gate of another MOS device. Node $x$ can charge up to $V_{DD} - V_{Tn}$. If node $C$ has a rail to rail swing, node $Y$ only charges up to the voltage on node $x - V_{Tn2}$ which works out to $V_{DD} - V_{Tn1} - V_{Tn2}$. Figure 6.31b on the other hand has the output of $M_i$ ($x$) driving the junc-
Static CMOS Design

Section 6.2

Example 6.8 VTC of the pass transistor AND gate

The voltage transfer curve of a pass-transistor gate shows little resemblance to complementary CMOS. Consider the AND gate shown in Figure 6.32. Similar to complementary CMOS, the VTC of pass transistor logic is data dependent. For the case when $B = V_{DD}$, the top pass transistor is turned on while the bottom one is turned off. In this case, the output just follows the input $A$ until the input is high enough to turn off the top pass transistor (i.e., reaches $V_{DD} - V_{T1}$). Next consider the case when $A = V_{DD}$ and $B$ makes a transition from 0 $\rightarrow$ 1. Since the inverter has a threshold of $V_{DD}/2$, the bottom pass transistor is turned on until then and the output is close to zero. Once the bottom pass transistor turns off, the output follows the input $B$ minus a threshold drop. A similar behavior is observed when both inputs $A$ and $B$ transition from 0 $\rightarrow$ 1. Note that pass transistor logic gates will need to be restored by placing inverters after every few pass transistors in series. With the inclusion of an inverter in the signal path, the VTC resembles the one of CMOS gates.

Pass transistors require lower switching energy to charge up a node due to its reduced voltage swing. For the pass transistor circuit in Figure 6.30 assume that the drain voltage is at $V_{DD}$ and the gate voltage transitions to $V_{DD}$. The output node charges from 0V.
to $V_{DD}-V_{Tn}$ (assuming that node $x$ was initially at 0V) and the energy drawn from the power supply for charging the output of a pass transistor is given by:

$$E_{0 \rightarrow 1} = \int_{0}^{T} P(t) \, dt = V_{DD} \int_{0}^{T} \text{supply}(t) \, dt - V_{DD} \int_{0}^{T} C_{L} \cdot dV_{out} = C_{L} \cdot (V_{DD} - V_{Tn})$$  

(6.20)

While the circuit exhibits lower switching power, it consumes static power when the output is high since the PMOS device of the connecting inverter is not fully turned off.

**Differential Pass Transistor Logic**

For high performance design, a differential pass transistor logic family, called CPL or DPL, is commonly used. The basic idea (similar to DCVSL) is to accept true and complementary inputs and produce true and complementary outputs. A number of CPL gates (AND/NAND, OR/NOR, and XOR/NXOR) are shown in Figure 4.38. These gates possess a number of interesting properties:

- Since the circuits are differential, complementary data inputs and outputs are always available. Although generating the differential signals requires extra circuitry, the differential style has the advantage that some complex gates such as XORs and adders can be realized efficiently with a small number of transistors. Furthermore,
the availability of both polarities of every signal eliminates the need for extra inverters, as is often the case in static CMOS or pseudo-NMOS.

- CPL belongs to the class of static gates, because the output-defining nodes are always connected to either \( V_{DD} \) or \( GND \) through a low resistance path. This is advantageous for the noise resilience.
- The design is very modular. In effect, all gates use exactly the same topology. Only the inputs are permuted. This makes the design of a library of gates very simple. More complex gates can be built by cascading the standard pass-transistor modules.

**Example 6.9 Four-input NAND in CPL**

Consider the implementation of a four-input AND/NAND gate using CPL. Based on the associativity of the boolean AND operation \( (A \cdot B \cdot C \cdot D) = ((A \cdot B) \cdot (C \cdot D)) \), a two-stage approach has been adopted to implement the gate (Figure 6.34). The total number of transistors in the gate (including the final buffer) is 14. This is substantially higher than previously discussed gates. This factor, combined with the complicated routing requirements, makes this circuit style not particularly efficient for this gate. One should, however, be aware of the fact that the structure simultaneously implements the AND and the NAND functions, which might reduce the transistor count of the overall circuit.

In summary, CPL is a conceptually simple and modular logic style. Its applicability depends strongly upon the logic function to be implemented. The availability of a simple XOR as well of the ease of implementing some specific gate structures makes it attractive for structures such as adders and multipliers. Some extremely fast and efficient implementations have been reported in that application domain [Yano90]. When considering CPL, the designer should not ignore the implicit routing overhead of the complementary signals, which is apparent in the layout of Figure 6.34.
Robust and Efficient Pass-Transistor Design

Unfortunately, differential pass transistor logic, like single-ended pass transistor logic suffers from static power dissipation since the high input to the inverter only charges up to $V_{DD} - V_T$. Static power is highly undesirable since in many portable electronics, the devices are idle for extended periods of time. Therefore, the voltage drop of pass transistors that causes lower noise margins and static power is not acceptable. There are several solutions proposed to deal with this problem as outlined below.

Solution 1: Level Restoration

A common solution to the voltage drop of pass transistors is the use of a level restorer, which is a single PMOS configured in a feedback path (Figure 6.35). The gate of the PMOS device is connected to the output of the inverter, its drain connected to the input of the inverter and the source to $V_{DD}$. Assume that node $X$ is at 0V ($out$ is at $V_{DD}$ and the $M_r$ is turned off) with $B = V_{DD}$ and $A = 0$. If input $A$ makes a 0 to $V_{DD}$ transition, $M_n$ only charges up node $X$ to $V_{DD} - V_T$. This is, however, enough to switch the output of the inverter low, turning on the feedback device $M_r$ and pulling node $X$ all the way to $V_{DD}$. This eliminates any static power dissipation in the inverter. Furthermore, no static current path can exist through the level restorer and the pass-transistor, since the restorer is only active when $A$ is high. In summary, this circuit has the advantage that all voltage levels are either at GND or $V_{DD}$, and no static power is consumed.

![Figure 6.35 Level-restoring circuit.](image)

While this solution is appealing in terms of eliminating static power dissipation, it is more complex since the circuit is now ratioed. The problem arises during the transition of node $X$ from high-to-low. The pass transistor network attempts to pull-down node $X$ while the level restorer pulls now $X$ to $V_{DD}$. Therefore, the pull-down device must be stronger than the pull-up device to switch node $X$ and the output. We use the notation $R_1$ to denote the equivalent on-resistance of transistor $M_1, R_2$ for $M_2$, and so on. Some careful transistor sizing is necessary to make the circuit function correctly: when $R_1$ is made too small, it is impossible to bring the voltage at node $X$ below the switching threshold of the inverter. Hence, the inverter output never switches to $V_{DD}$, and the level-restoring transistor stays on. This sizing problem can be reformulated in the following way:

The resistance of $M_n$ and $M_r$ must be such that the voltage at node $X$ drops below the threshold of the inverter, $V_M = f(R_1, R_2)$. This condition is sufficient to guarantee a switching of the output voltage $V_{out}$ to $V_{DD}$ and a turning off of the level-restoring transistor.
Example 6.10 Sizing of a Level Restorer

Analyzing the circuit as a whole is nontrivial, because the restoring transistor acts as a feedback device. One way to simplify the circuit for manual analysis is to open the feedback loop and to ground the gate of the restoring transistor when determining the switching point (this is a reasonable assumption, as the feedback only becomes effective once the inverter starts to switch). Hence, $M_r$ and $M_n$ form a “pseudo-NMOS-like” configuration, with $M_r$ the load transistor and $M_n$ acting as a pull-down device to GND. Assume that the inverter $M_1$, $M_2$ is sized to have the switching point at $V_{DD}/2$ (NMOS: 0.5µm/0.25µm and PMOS: 1.5µm/0.25µm). Therefore, node $X$ must be pulled below $V_{DD}/2$ in order to switch the inverter and shut off $M_r$.

This is confirmed in Figure 6.38, which shows the transient response as the size of the level restorer is varied while keeping the size of $M_n$ fixed (0.5µm/0.25µm). As the simulation indicates, for sizes above 1.5µm/0.25µm, node $X$ can’t be brought below the switching threshold of the inverter and can’t switch the output. The detailed derivation of sizing requirement will be presented in the sequential design chapter. An important point to observe here is that the sizing of $M_r$ is critical for DC functionality, not just performance!

Another concern is the influence of the level restorer on the switching speed of the device. Adding the restoring device increases the capacitance at the internal node $X$, slowing down the gate. The rise time of the gate is further negatively affected, since, the level-restoring transistor $M_1$ fights the decrease in voltage at node $X$ before being switched off.
On the other hand, the level restorer reduces the fall time, since the PMOS transistor, once turned on, speeds the pull-up action.

**Problem 6.5  Device Sizing in Pass Transistors**

For the circuit shown in Figure 6.36, assume that the pull-down device consists of 6 pass transistors in series each with a device size of 0.5µm/0.25µm (replacing transistor $M_n$). Determine the maximum $W/L$ size for the level restorer transistor for correct functionality.

A modification of the level restorer to differential pass transistors is shown in Figure 6.38, known as swing restored pass transistor logic. Instead of a simple inverter or half latch at the output of the pass transistor network, two back-to-back inverters configured in a cross-coupled fashion are used for level restoration and performance improvement. Inputs are fed to both the gate and source/drain terminals as in the case of conventional pass transistor networks. Figure 6.38 shows a simple XOR/XNOR gate of three variables $A$, $B$ and $C$. Notice that the complementary network can be optimized by sharing transistors between the true and complementary outputs.

**Solution 2: Multiple Threshold Transistors**

A technology solution to the voltage drop problem associated with pass transistor logic is the use of multiple threshold devices. Pass transistors only pass $V_{DD} - V_{Th}$, degrading the high voltage level. One solution is to use zero threshold devices for the NMOS pass transistors, enabling passing signal close to $V_{DD}$. Notice that even if the devices threshold was implanted to be exactly equal to zero, the body effect of the device prevents a swing to $V_{DD}$. All devices other than the pass transistors (i.e., the inverters) are implemented using standard high threshold devices. The use of multiple threshold transistors is becoming more common and involves simple modifications to existing process flows.
The use of zero-threshold transistors can be dangerous due to the subthreshold currents that can flow through the pass-transistors, even if $V_{GS}$ is slightly below $V_T$. This is demonstrated in Figure 6.39, which points out a potential sneak dc-current path. While these leakage paths are not critical when the device is switching constantly, they do pose a significant energy overhead when the device is in the idle state.

**Solution 3: Transmission Gate Logic**

The most widely used solution to deal with the voltage drops induced by pass transistors is the use of transmission gates. The primary limitation of NMOS or PMOS only pass gate is the threshold drop (NMOS pass device pass a strong 0 while passing a weak 1 and PMOS pass devices pass a strong 1 while passing a weak 0). The ideal approach is to use the NMOS device to pull-down and the PMOS device to pull-up. The transmission gate combines the best of both device flavors by placing a NMOS device in parallel with a PMOS device (Figure 6.40a). The control signals to the transmission gate (C and $\overline{C}$)

\[
A = B \quad \text{if} \quad C = 1
\]  

(6.21)
On the other hand, \( C = 0 \) places both transistors in cutoff, creating an open circuit between nodes \( A \) and \( B \). Figure 6.40b shows a commonly used transmission-gate symbol.

Consider the case of charging node \( B \) to \( V_{DD} \) for the transmission gate circuit in Figure 6.41a. Node \( A \) is driven to \( V_{DD} \) and transmission gate is enabled \((C = 1 \text{ and } \overline{C} = 0)\). If only the NMOS pass device were present, node \( B \) will charge up to \( V_{DD} - V_{Tn} \) at which point the NMOS device turns off. However, since the PMOS device is present and turned on \((V_{GS} = -V_{DD})\), node \( B \) charge all the way up to \( V_{DD} \). Figure 6.41b shows the case for discharging node \( B \) to 0. \( B \) is initially at \( V_{DD} \) and node \( A \) is driven low. The PMOS pass transistor by itself can only pull-down node \( B \) to \( V_{Tn} \) at which point the PMOS device turns off. the parallel NMOS device however is turned on (since its \( V_{GS} = V_{DD} \)) and pulls down node \( B \) all the way to \( GND \). Though the transmission gate requires two transistors and more control signals, it enables rail-to-rail swing.

Transmission gates can be used to build some complex gates very efficiently. Figure 6.42 shows an example of a simple inverting two-input multiplexer. This gate either selects input \( A \) or \( B \) based on the value of the control signal \( S \), which is equivalent to implementing the following Boolean function:

\[
F = (A \cdot S + B \cdot \overline{S})
\]  
(6.22)

A complementary implementation of the gate requires eight transistors instead of six.

Another example of the effective use of transmission gates is the popular XOR circuit shown in Figure 6.43. The complete implementation of this gate requires only six transistors (including the inverter used for the generation of \( \overline{B} \)), compared to the twelve transistors required for a complementary implementation. To understand the operation of this circuit, we have to analyze the \( B = 0 \) and \( B = 1 \) cases separately. For \( B = 1 \), transistors \( M_1 \) and \( M_2 \) act as an inverter while the transmission gate \( M_3/M_4 \) is off; hence \( F = \overline{A}B \). In the opposite case, \( M_1 \) and \( M_2 \) are disabled, and the transmission gate is operational, or \( F = A \overline{B} \). The combination of both results in the XOR function. Notice that, regardless of the values of \( A \) and \( B \), node \( F \) always has a connection to either \( V_{DD} \) or \( GND \) and is hence a low-impedance node. When designing static-pass transistor networks, it is essential to adhere to the low-impedance rule under all circumstances. Other examples where transmission-gate logic is effectively used are fast adder circuits and registers.
Resistance and Delay of Transmission Gate Logic

The transmission gate is, unfortunately, not an ideal switch, and has a series resistance associated with it. To quantify the resistance, consider the circuit in Figure 6.44, which involves charging a node from 0 V to $V_{DD}$. In this discussion, we will use the large signal definition of resistance which involves dividing the voltage across the switch by the drain current. The effective resistance of the switch is modeled as a parallel connection of the resistances $R_n$ and $R_p$ of the NMOS and PMOS devices, defined as $(V_{DD} - V_{out})/I_n$ and $(V_{DD} - V_{out})/I_p$, respectively. The currents through the devices are obviously dependent on the value of $V_{out}$ and the operating mode of the transistors. During the low-to-high transition, the pass-transistors traverse through a number of operation modes. Figure 6.44 shows the individual resistances and the combined parallel resistance. For low values of $V_{out}$, the NMOS device is saturated and the resistance is approximated as:
The resistance goes up for increasing values of $V_{out}$, and approaches infinity when $V_{out}$ reaches $V_{DD} - V_{TN}$, this is when the device shuts off. Similarly, we can analyze the behavior of the PMOS transistor. When $V_{out}$ is small, the PMOS is saturated, but it enters the linear mode of operation for $V_{out}$ approaching $V_{DD}$. The resistance then approximated by:

$$R_p = \frac{V_{DD} - V_{out}}{I_p} = \frac{V_{DD} - V_{out}}{k_p}\left(V_{DSAT} - \frac{(V_{out} - V_{DD})^2}{2}\right)$$

$$\approx k_p(V_{DD} - V_{out})V_{DSAT}$$

(6.24)

The simulated value of $R_{eq} = R_p \parallel R_n$ as a function of $V_{out}$ is plotted in Figure 6.44. It can be observed that $R_{eq}$ is relatively constant ($\approx 8k\Omega$ in this particular case). The same is true in other design instances (for instance, when discharging $C_L$). When analyzing transmission-gate networks, the simplifying assumption that the switch has a constant resistive value is therefore acceptable.

![Figure 6.44](image-url) Simulated equivalent resistance of transmission gate for low-to-high transition (for $(W/L)_n = (W/L)_p = 0.5\mu m/0.25\mu m$). A similar response for overall resistance is obtained for the high-to-low transition.

### Problem 6.6 Equivalent Resistance During Discharge

Determine the equivalent resistance by simulation for the high-to-low transition of a transmission gate (this is, produce a plot similar to the one presented in Figure 6.44).
An important consideration is the delay associated with a chain of transmission gates. Figure 6.45 shows a chain of \( n \) transmission gates. Such a configuration often occurs in circuits such as adders or deep multiplexors. Assume that all transmission gates are turned on and a step is applied at the input. To analyze the propagation delay of this network, the transmission gates are replaced by their equivalent resistances \( R_{eq} \). This produces the network of Figure 6.45b.

![Figure 6.45](image)

(a) A chain of transmission gates

(b) Equivalent RC network

**Figure 6.45** Speed optimization in transmission-gate networks.

The exact analysis of delay is not simple, but as discussed earlier, we can estimate the dominant time constant at the output of a chain of \( n \) transmission gates as follows:

\[
\tau(V_n) = \sum_{k=0}^{n} C R_{eq} k = C R_{eq} \frac{n(n + 1)}{2}
\] (6.25)

This means that the propagation delay is proportional to \( n^2 \) and increases rapidly with the number of switches in the chain.

**Example 6.11 Delay through 16 transmission gates**

Consider 16 minimum sized transmission gates with an average resistance of 8 k\( \Omega \). The node capacitance consists of the capacitance of two NMOS devices (junction and gate) and the capacitance two PMOS devices (junctions and gate). Since the gate inputs are assumed to be fixed, there is no Miller multiplication. The capacitance can be calculated to be approximately 3.6 fF for the low-to-high transition. The delay is given by:

\[
t_p = 0.69 \cdot C R_{eq} \frac{n(n + 1)}{2} = 0.69 \cdot (3.6 fF)(8 k\Omega)(\frac{16(16 + 1)}{2}) \approx 2.7 ns
\] (6.26)

The transient response for this particular example is shown in Figure 6.46. The simulated delay is 2.7 ns. It is remarkable that a simple RC model predicts the delay accu-
It is also clear that the use of long pass transistor chains causes significant delay degradation.

The most common approach for delaying with the long delay is to break the chain every \( m \) switches and to insert buffers (Figure 6.46). Assuming a propagation delay \( t_{buf} \) for each buffer, the overall propagation delay of the transmission-gate/buffer network can be computed as follows,

\[
t_p = 0.69 \left[ \frac{2}{m} C R_{eq} \frac{m(m+1)}{2} \right] + \left( \frac{n}{m} - 1 \right) t_{buf}
\]

\[
= 0.69 \left[ C R_{eq} \frac{n(m+1)}{2} \right] + \left( \frac{n}{m} - 1 \right) t_{buf}
\]

The resulting delay exhibits only a linear dependence on the number of switches \( n \), in contrast to the unbuffered circuit, which is quadratic in \( n \). The optimal number of switches \( m_{opt} \) between buffers can be found by setting the derivative

\[
\frac{d t_p}{dm}
\]

to 0, which yields

\[
m_{opt} = 1.7 \sqrt{\frac{t_{buf}}{C R_{eq}}}
\]
Obviously, the number of switches per segment grows with increasing values of $t_{\text{buf}}$. In current technologies, $m_{\text{opt}}$ typically around 3.

Example 6.12 Transmission Gate Chain

Consider the same 16 transmission gate chain. The buffers shown in Figure 6.47 can be implemented as inverters (instead of two cascaded inverters). In some cases, it might be necessary to add an extra inverter to produce the correct polarity. Assuming that each inverter is sized such that the NMOS is $0.5\mu\text{m}/0.25\mu\text{m}$ and PMOS is $0.5\mu\text{m}/0.25\mu\text{m}$, Eq. (6.28) predicts that an inverter must be inserted every 3 transmission gates. The simulated delay when placing an inverter every two transmission gates equals 154ps, for every three transmission gates is 154ps and for four transmission gates is 164ps. The insertion of buffering inverters reduces the delay with a factor of almost 2.

CAUTION: Although many of the circuit styles discussed in the previous sections sound very exciting, and might be superior to static CMOS in many respects, none of them has the robustness and ease of design of complementary CMOS. Therefore, use them sparingly and with caution. For designs that have no extreme area, complexity, or speed constraints, complementary CMOS is the recommended design style.

6.3 Dynamic CMOS Design

It was noted earlier that static CMOS logic with a fan-in of $N$ requires $2N$ devices. A variety of approaches were presented to reduce the number of transistors required to implement a given logic function including pseudo-NMOS, pass transistor logic, etc. The pseudo-NMOS logic style requires only $N + 1$ transistors to implement an $N$ input logic gate, but unfortunately it has static power dissipation. In this section, an alternate logic style called dynamic logic is presented that obtains a similar result, while avoiding static power consumption. With the addition of a clock input, it uses a sequence of precharge and conditional evaluation phases to realize complex logic functions.
6.3.1 Dynamic Logic: Basic Principles

The basic construction of a N-type dynamic logic gate is shown in Figure 6.48a. The PDN (pull-down network) is constructed exactly in the same fashion as a complementary CMOS. The operation of this circuit can be divided into two major phases: precharge and evaluation, with the mode of operation determined by the clock signal.

Precharge

When $CLK = 0$, the output node $Out$ is precharged to $V_{DD}$ by the PMOS transistor $M_p$. During that time, the evaluate NMOS transistor $M_e$ is off, so the pull-down path does not fight the pull-up path. The evaluation FETS also eliminate any static power that would be consumed during the precharge period (i.e., if the pull-down path was turned on and the precharge device was turned on, static current would flow between the supplies).

Evaluation

When $CLK = 1$, the precharge transistor $M_p$ is off, and the evaluation transistor $M_e$ is turned on. The output is conditionally discharged based on the input values and the pull-down topology. If the inputs are such that the PDN conducts, then a low resistance path exists between $Out$ and $GND$ and the output is discharged to $GND$. If the PDN is turned off, the precharged value remains stored on the output capacitance $C_L$, which is a combination of junction capacitances, the wiring capacitance, and the input capacitance of the fan-out gates. During the evaluation phase, the only possible path between the output node and a supply rail is to $GND$. Consequently, once $Out$ is discharged, it cannot be charged again till then next precharge operation. The inputs to the gate can therefore make at most one transition during evaluation. Notice that the output can be in the high impedance state during the evaluation period if the pull-down network is turned off and this behavior is fundamentally different than the static counterpart that always has a low resistance path between the output and one of the power rails.
As an example of dynamic logic, consider the circuit shown in Figure 6.48b. During the precharge phase \((CLK=0)\), the output is precharged to \(V_{DD}\) regardless of the input values since the evaluation device is turned off. During evaluation \((CLK=1)\), a conducting path is created between \(Out\) and \(GND\) if (and only if) \(A \cdot B + C\) is TRUE. Otherwise, the output remains at the precharged state of \(V_{DD}\). The following function is thus realized:

\[
Out = A \cdot B + C \quad \text{(when } CLK = 1) \tag{6.29}
\]

A number of important properties can be derived for the dynamic logic gate:

- The logic function is implemented by the NMOS pull-down network. The construction of the PDN proceeds just as it does for static CMOS.
- The number of transistors (for complex gates) is substantially lower than in the static case: \(N + 2\) versus \(2N\).
- It is nonratioed. The sizing of the PMOS precharge device is not important for realizing proper functionality of the gate. The size of the precharge device can be made large to improve the low-to-high transition time (of course, at a cost to the high-to-low transition time). There is however, a trade-off with power dissipation since a larger precharge device directly increases clock power dissipation.
- It only consumes dynamic power. Ideally, no static current path ever exists between \(V_{DD}\) and \(GND\). The overall power dissipation, however, can be significantly higher compared to a static logic gate.
- The logic gates have faster switching speeds. There are two main reasons for this. The first (obvious) reason is due to the reduced load capacitance attributed to the number of transistors per gate and the single-transistor load per fan-in. Second, the dynamic gate does not have short circuit current, and all the current provided by the pull-down devices go into discharging the load capacitance.

The low and high output levels \(V_{OL}\) and \(V_{OH}\) are easily identified as \(GND\) and \(V_{DD}\) and are not dependent upon the transistor sizes. The other VTC parameters are dramatically different from static gates. Noise margins and switching thresholds have been defined as static quantities, which are not influenced by time. To be functional, a dynamic gate requires a periodic sequence of precharges and refreshes. Pure static analysis, therefore, does not apply. During the evaluate period, the pull-down network of a dynamic inverter starts to conduct when the input signal exceeds the threshold voltage \((V_{TH})\) of the NMOS pull-down transistor. Therefore, it is reasonable to set the switching threshold \((V_{TH})\) as well as \(V_{IH}\) and \(V_{IL}\) of the gate equal to \(V_{TH}\). This translates to a low value for the \(NM_{L}\).

### 6.3.2 Speed and Power Dissipation of Dynamic Logic

The main advantage of dynamic logic is speed and potentially smaller implementation area.Fewer devices to implement a given logic function implies that the overall load capacitance is much smaller. The analysis of the switching behavior of the gate has some interesting peculiarities to it. After the precharge phase, the output is high. For a low input signal, no additional switching occurs. As a result, \(t_{pLU} = 0\) ! The high-to-low transition, on
the other hand, requires the discharging of the output capacitance through the pull-down network. Therefore, $I_{pd}$ is proportional to $C_L$ and the current-sinking capabilities of the PDN. The presence of the evaluation transistor slows the gate somewhat, as it presents an extra series resistance to the pull-down network. Omitting this transistor, while functionally not forbidden, results in significant performance loss and static power dissipation.

The above analysis is somewhat unfair, because it ignores the influence of the precharge time on the switching speed of the gate. The precharge time is determined by the time it takes to charge $C_L$ through the PMOS precharge transistor. During this time, the logic in the gate cannot be utilized. However, very often, the overall digital system can be designed in such a way that the precharge time coincides with other system functions. For instance, the precharge of the arithmetic unit in a microprocessor can coincide with the instruction decode. The designer has to be aware of this “dead zone” in the use of dynamic logic, and should carefully consider the pros and cons of its usage, taking the overall system requirements into account.

**Example 6.13 A Four-Input Dynamic NAND Gate**

Figure 6.49 shows the design of a four-input NAND example designed using the dynamic-circuit style. Due to the dynamic nature of the gate, the derivation of the voltage-transfer characteristic diverges from the traditional approach. As we had discussed above, we will assume that the switching threshold of the gate equals the threshold of the NMOS pull-down transistor. This results in asymmetrical noise margins, as shown in Table 6.7.

The dynamic behavior of the gate is simulated with SPICE. It is assumed that all inputs are set high as the clock transitions high. On the rising edge of the clock, the output node is discharged. The resulting transient response is plotted in Figure 4.35. The resulting propagation delays are summarized in Table 6.7. The length of the precharge time can be adjusted by changing the size of the PMOS precharge transistor. Making the PMOS too large should be avoided, however, as it both slows down the gate and increases the capacitive load on the clock line. For large designs, the latter factor might become a major design concern because the clock load can become excessive and hard to drive.
As mentioned earlier, the static parameters are time dependent. To illustrate this, consider the four input NAND gate with all inputs tied together. Assume that the inputs make a partial low-to-high transition. Figure 6.50 shows a simulation of the output voltage for three different input voltages (when input transitions to 0.45V, 0.5V and 0.55V). We have previously defined the switching threshold of the dynamic gate as the device threshold. However, notice that the amount by which the output voltage drops is a strong function of the input voltage and the available evaluation time. In this example, a larger input voltage is necessary to corrupt the output. So the switching threshold is really a function of the evaluation time.

When evaluating the power dissipation of a dynamic gate, it would appear that dynamic logic presents significant advantage. There are three reasons for this. First, since dynamic logic uses fewer transistors to implement a given function, it should have a lower physical capacitance. Basically the load seen for each fanout is one transistor instead of two. Second, dynamic logic gates by construction can at most have one transition per clock cycle. The glitching transitions seen in static gates are not seen in dynamic gates. Finally, dynamic gates do not exhibit short circuit power since the pull-up path is not turned on when the gate is evaluating. While the above statements are generally true, several important second order effects causes the number of transistors to be higher than the minimal set required for implementing the logic and short circuit power does exist if the logic is required to be pseudo-static. Also, the clock power of dynamic logic can be significant particularly since the clock node has a guaranteed transition on every single clock cycle.

<table>
<thead>
<tr>
<th>Transistors</th>
<th>( V_{OH} )</th>
<th>( V_{OL} )</th>
<th>( V_{M} )</th>
<th>( N_{M_H} )</th>
<th>( N_{M_L} )</th>
<th>( t_{pHL} )</th>
<th>( t_{pLH} )</th>
<th>( t_{pre} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2.5 V</td>
<td>0 V</td>
<td>( V_{TN} )</td>
<td>2.5- ( V_{TN} )</td>
<td>( V_{TN} )</td>
<td>110 ps</td>
<td>0 assec</td>
<td>83 ps</td>
</tr>
</tbody>
</table>

Table 6.7 The dc and ac parameters of a four-input dynamic NAND.

Figure 6.50 Effect of an input glitch on the output. The switching threshold depends on the time for evaluation. A larger glitch is acceptable if the evaluation phase is smaller. In this example, the input glitches high during evaluation and stays high during the whole period.
Dynamic logic generally has higher activity due to constant *precharge* and *discharge* operations. Earlier, the transition probability for a static gate was shown to be $p_0 p_1 = p_0 (1-p_0)$. For dynamic logic, the output transition probability does not depend on the state (history) of the inputs but rather on just the signal probabilities. For an N-tree dynamic gate, the output will make a 0 to 1 transition during the precharge phase only if the output was discharged by the N-tree logic during the evaluate phase. The zero to one transition probability for an N-tree structure is therefore

$$\alpha_0 \rightarrow 1 = p_0$$  \hspace{1cm} (6.30)

where $p_0$ is the probability that the output is in the zero state. For uniformly distributed inputs, this means that the transition probability is:

$$\alpha_0 \rightarrow 1 = \frac{N_0}{2^N}$$  \hspace{1cm} (6.31)

where $N_0$ is the number of zero entries in the truth table of the logic function.

**Example 6.14 Activity estimation in dynamic logic**

To illustrate the increased activity for a dynamic gate, once again consider a 2 input NOR gate. An N-tree dynamic NOR gate is shown in Figure 6.51 along with its static counterpart. For the dynamic implementation, power is consumed during the precharge operation for the times when the output capacitor was discharged the previous cycle. For equi-probable input, there is then a 75% probability that the output node will discharge immediately after the precharge phase, implying that the activity for such a gate is 0.75 (i.e. $P_{NOR} = 0.75 C_L V_{dd}^2 f_{clk}$). The corresponding activity is a lot smaller, 3/16, for a static implementation. Note that for the dynamic case, the activity depends only on the signal probability, while for the static case the transition probability depends on previous state. If the inputs to a static CMOS gate do not change from the previous sample period, then the gate does not switch. This is not true in the case of dynamic logic in which gates can switch. For a dynamic NAND gate, the transition probability is 1/4 (since there is a 25% probability the output will be discharged) while it is 3/16 for a static implementation. Though this example illustrates that the switching activity can be higher using dynamic
logic, it should be noted that dynamic logic has lower physical capacitance. Both factors must be accounted for when choosing a logic style.

Problem 6.7 Activity Computation

For the 4-input dynamic NAND gate, compute the activity factor with the following assumption for the inputs. Assume that the inputs are independent and $p_A = 0.2$, $p_B = 0.3$, $p_C = 0.5$, and $p_D = 0.4$.

6.3.3 Issues in Dynamic Design

Dynamic logic clearly can result in high performance solutions compared to static circuits. However, there are several important considerations that must be taken into account to make dynamic circuits function properly. This includes charge leakage, charge sharing, backgate (and in general capacitive) coupling, and clock feedthrough. Some of these issues are highlighted in this section.

Charge Leakage

The operation of a dynamic gate relies on the dynamic storage of the output value on a capacitor. During the evaluation period, if the pull-down network is off, then ideally the output should remain at the precharged state of $V_{DD}$. However, due to leakage currents, this charge gradually leaks away, resulting eventually in malfunctioning of the gate. Figure 6.52a shows the different sources of leakage for a simple dynamic inverter circuit.

Source 1 and 2 are the reverse-biased diode and sub-threshold leakage of the NMOS pull-down device $M_1$ respectively. The charge stored on $C_L$ will slowly leak away due these leakage sources, assuming that the input is in the low state during evaluation. Charge leakage causes a degradation in the high level (Figure 6.52b). Dynamic circuits therefore require a minimal clock rate, which is typically on the order of a few kHz. This makes the usage of dynamic techniques unattractive for certain low performance products such as watches or processors that need to provide conditional clocks (where there are no

---

Figure 6.51 Static NOR vs. N-tree based dynamic NOR.
guarantees on minimum clock rates). Note that the PMOS precharge device also contributes some leakage due to the reverse bias diode (source 3) and subthreshold conduction (source 4). To some extent, the leakage current of the PMOS counteracts the leakage due to the pull-down path. As a result the output voltage is going to be set by the resistive divider composed of the pull-down and pull-up paths.

**Example 6.15 Example of leakage**

Consider the simple inverter with all devices set at 0.5\(\mu\)m/0.25\(\mu\)m. Assume that the input is low during the evaluation period. Ideally, the output should remain at the precharged state of \(V_{DD}\). However, as seen from Figure 6.53 the output voltage drops. Once the output drops below the switching threshold of the fan-out logic gate, the output is interpreted as a low voltage. Notice that the output settles to an intermediate voltage. This is due to the leakage provided by the PMOS pull up device.

**Figure 6.52 Leakage issues in dynamic circuits.**

(a) Leakage sources

(b) Effect on waveforms

**Figure 6.53 Impact of charge leakage.** The output settles to an intermediate voltage determined by a resistive divider of the pull-down and pull up devices.

---

Leakage is caused by the high impedance state of the output node when the pull down path is turned off during the evaluate mode. To deal with the leakage problem, the impedance on the output node must be reduced during evaluate period. This is often done
by adding a bleeder transistor to the output node, as shown in Figure 6.54. The bleeder compensates for the charge lost due to the pull-down leakage paths. When the clock is high and the pull-down network is turned off, the output remains high. In order to avoid the ratioed problems associated with this circuit, the bleeder resistance is made high (small device size). This allows the pull-down devices to be strong enough to pull-down the Out node below the switching threshold of the inverter. The circuit does have static power dissipation when Out is pulled low during the evaluation period. Often, the bleeder is implemented in a feedback configuration to eliminate static power dissipation.

**Figure 6.54** Static bleeder to compensate for the charge leakage problem.

### Charge Sharing

Another important consideration in dynamic logic is charge sharing. Consider the circuit of Figure 6.55. During the precharge phase, the output node is precharged to $V_{DD}$. Assume that all inputs are set to 0 during precharge and that the capacitance $C_a$ is discharged. Assume further that input $B$ remains at 0 during evaluation, while input $A$ makes a $0 \rightarrow 1$ transition, turning transistor $M_a$ on. The charge stored originally on capacitor $C_L$ is redistributed over $C_L$ and $C_a$. This causes a drop in the output voltage, which cannot be recovered due to the dynamic nature of the circuit.

The influence on the output voltage is readily calculated. Under the above assumptions, the following initial conditions are valid: $V_{out}(t = 0) = V_{DD}$ and $V_x(t = 0) = 0$. Two cases must be considered:

1. $\Delta V_{out} < V_{Th}$—In this case, the final value of $V_X$ equals $V_{DD} - V_{Th}(V_X)$. Charge conservation yields

$$C_L V_{DD} = C_L V_{out}(t) + C_a [V_{DD} - V_{Th}(V_X)]$$

or

$$\Delta V_{out} = V_{out}(t) - V_{DD} = \frac{C_a}{C_L} [V_{DD} - V_{Th}(V_X)]$$

(6.32)

2. $\Delta V_{out} > V_{Th}$—$V_{out}$ and $V_X$ reach the same value:
Overall, it is desirable to keep the value of $\Delta V_{\text{out}}$ below $|V_{\text{th}}|$. The output of the dynamic gate might be connected to a static inverter, in which case the low level of $V_{\text{out}}$ would cause static power consumption. One major concern is circuit malfunction if the output voltage is brought below the switching threshold of the gate it drives.

**Example 6.16 Charge Sharing Example**

Consider the dynamic logic gate shown in Figure 6.56. It can be easily verified that the function implemented by the logic gate is $y = A \oplus B \oplus C$. To analyze charge sharing, the question we will ask is for the case when the output is nominally supposed to stay high, what is the worst case change in voltage on node $y$. For simplicity, ignore the load inverter, and assume that all inputs are low during the precharge operation and that all isolated internal nodes ($V_a$, $V_b$, $V_c$, and $V_d$) are initially at 0V.

Overall, it is desirable to keep the value of $\Delta V_{\text{out}}$ below $|V_{\text{th}}|$. The output of the dynamic gate might be connected to a static inverter, in which case the low level of $V_{\text{out}}$ would cause static power consumption. One major concern is circuit malfunction if the output voltage is brought below the switching threshold of the gate it drives.

**Figure 6.55** Charge sharing in dynamic networks.

$$\Delta V_{\text{out}} = -V_{\text{DD}} \left( \frac{C_a}{C_a + C_L} \right)$$  \hspace{1cm} (6.33)

**Figure 6.56** Example illustrating the charge sharing effect in dynamic logic.
There are four possible cases when the output remains high and the challenge is to find the combination of inputs that results in the maximum change of the output voltage. The worst case change in output is obtained by exposing the maximum amount of internal capacitance to the output node during the evaluation period. This happens when \( \overline{A} \overline{B} C \) or \( A \overline{B} \overline{C} \). The voltage change can be easily obtained by equating the initial charge with the final charge as done with equation Eq. (6.33). Doing this results in a worst case change of \( \frac{30}{(30+50)} \times 2.5V = 0.94V \). To ensure the circuit functions correctly, the switching threshold of the inverter should be placed below \( 2.5 - 0.94 = 1.56V \).

The most common and effective approach to deal with the charge redistribution is to precharge the (critical) internal nodes as well, as shown in Figure 6.57b. Since the internal nodes are charged to \( V_{DD} \) during precharge, there is no problem with charge sharing. However, this solution obviously comes at the cost of increased area and capacitance.

Figure 6.57 Dealing with the charge sharing problem by precharging internal nodes. An NMOS precharge transistor may also be used, however, this requires an inverted clock.

**Capacitive Coupling**

Capacitive coupling is another major problem in dynamic circuits. There are many forms of capacitive coupling that arise due to floating nodes in dynamic circuits. For example, a wire routed over a dynamic node can capacitively couple and destroy the state of a floating node. Another equally important form of capacitive coupling is backgate coupling. Consider the circuit shown in Figure 6.58 in which a dynamic two input NAND gate drives a static NAND gate. Assume that the input \( IN \) is initially low during the precharge operation. Also assume that the inputs \( A \) and \( B \) are low during the entire precharge and evaluate period. Therefore, \( Out_1 \) should remain ideally in the high state (ignoring leakage). If \( IN \) goes high during the evaluate period, the output of the static gate, \( Out_2 \) should be pulled low. In this process, due to capacitive backgate coupling between the internal and output node of the static gate and the output of the dynamic gate, \( Out_1 \) node voltages reduces. A simulation of this is shown in Figure 6.59. As seen from this simulation, the output of the dynamic gate can drop significantly. As a result, the output of the
static NAND gate does not drop all the way down to 0V and a small amount of static power is dissipated. If the voltage drop is large enough, the circuit can evaluate incorrectly since the NAND output may not go low. In general, care must be taken to design circuits to minimize noise introduced by capacitive coupling.

Clock Feedthrough
A special case of capacitive coupling is clock feedthrough. Clock feedthrough is an effect caused by the coupling between the dynamic output storage node and the gate input of the precharge device due to the gate to drain capacitance (which includes both the overlap and the channel capacitance). During the precharge phase, the output of the dynamic gate precharges high. On the low the high transition of the clock, there should be no effect on the output (assuming the pull-down network is turned off). However, due to the capacitive coupling, the voltage on the output node can rise above $V_{DD}$. The fast rising and falling
edges of the clock couple into the signal node, as is adequately demonstrated in the simulation of Figure 6.59.

The danger of clock feedthrough is that it causes the signal level to rise sufficiently above the supply voltage that the (normally reverse-biased) junction diodes become forward-biased. This causes electron injection into the substrate, which can be collected by a nearby high impedance node in the 1 state, eventually resulting in faulty operation. CMOS latchup might be another result of this injection. For all purposes, high-speed dynamic circuits should be carefully simulated to ensure that clock feedthrough effects stay within bounds.

All the above considerations demonstrate that the design of dynamic circuits is rather tricky and requires extreme care. It should therefore only be attempted when high performance is required.

### 6.3.4 Cascading Dynamic Gates

So far, we have focused on the basic functionality and constraints of individual dynamic logic gates. Unfortunately, the way the circuit is implemented, dynamic gates cannot be directly cascaded. To illustrate this, consider two simple N-type dynamic inverters cascaded together, as shown in Figure 6.60a. During the precharge phase (i.e., $CLK = 0$), the output of both inverters are precharged up to $V_{DD}$. Assume that the primary input $In$ makes a $0 \rightarrow 1$ transition (Figure 6.60b). On the rising edge of the clock, output $Out_1$ starts to discharge. The second output should remain in the precharged state of $V_{DD}$ since $Out_1$ transitions to 0 during evaluation. However, since there is a finite propagation delay for the input to discharge $Out_1$ to GND, the second output also starts to discharge. As long as $Out_1$ exceeds the switching threshold of the second gate, which approximately equals $V_{Tn}$, a conducting path exists between $Out_2$ and GND. $Out_2$ therefore discharges as well, resulting in incorrect evaluation. This conducting path is only turned off when $Out_1$ reaches $V_{Tn}$ and shuts off the NMOS pull-down transistor. This leaves $Out_2$ at an intermediate voltage level. The correct level will not be recovered, since dynamic gates rely on capacitive stor-

![Figure 6.60](cascade.png)

---

**Figure 6.60** Cascade of dynamic N-type blocks.
age, in contrast to static gates, which have dc restoration. The charge loss leads to reduced noise margins and eventual malfunctioning.

It is obvious that the cascading problems arise because the output (and hence the input to the next stage) is precharged to 1. Setting the inputs to 0 during precharge could solve this problem. In doing so, all logic transistors of the next function block are turned off after precharge, and no inadvertent discharging of the storage capacitors can occur during evaluation. In other words, correct operation is guaranteed (ignoring charge redistribution and leakage) as long as the inputs can only make a single 0 → 1 transition during the evaluation period. This eliminates the inadvertent discharging since transistors will only be turned on when needed and at most one time per cycle. A number of design styles complying with the above rule have been developed. The two most important ones are discussed below.

**Domino Logic**

A Domino logic module [Krambeck82] consists of an N-type dynamic logic block followed by a static inverter (Figure 6.61). During precharge, the output of the N-type dynamic gate is charged up to $V_{DD}$ and the output of the inverter is set to 0. During evaluation, based on the inputs, the dynamic gate conditionally discharges and the output of the inverter makes a conditional transition from 0 → 1. The input to a Domino gate always comes from the output of another Domino gate. This ensures that all inputs to the Domino gate are set to 0 at end of the precharge period. Hence, the only possible transition for the input during the evaluation period is the 0 → 1 transition, so that the formulated rule is obeyed. The introduction of the static inverter has the additional advantage that the fan-out of the gate is driven by a static inverter with a low-impedance output, which increases noise immunity. The buffer furthermore reduces the capacitance of the dynamic output node by separating internal and load capacitances.

Consider now the operation of a chain of Domino gates. During precharge, all inputs are set to 0. During evaluation, the output of the first Domino block either stays at 0 or makes a 0 → 1 transition, affecting the second Domino. This effect might ripple through the whole chain, one after the other, as with a line of falling dominoes—hence the name. Domino CMOS has the following properties:

![Figure 6.61 DOMINO CMOS logic.](image-url)
Since each dynamic gate has a static inverter, only noninverting logic can be implemented. This is a major limiting factor, and though there are ways to deal with this (as will be discussed), pure Domino design have become rare.

Very high speeds can be achieved: only a rising edge delay exists, while \( t_{puH} \) equals zero (as the output node is precharged low). The static inverter can be optimized to match the fan-out, which is already much smaller than in the complimentary static CMOS case (only a single gate capacitance per input).

Since the inputs to a Domino gate are low during precharge, it is tempting to eliminate the evaluation transistor as it reduces clock load and increases pull-down drive. However, eliminating the evaluation device results in a performance degradation since the precharge has to ripple through the critical path. Consider the simple logic network shown in Figure 6.62, where the evaluation device has been eliminated. If the primary input \( I_{n1} \) is 1 during evaluation, the output of each dynamic gate is 0 and the output of each static inverter is 1. On the falling edge of the clock, we start the precharge operation assuming \( I_{n1} \) makes a high-to-low transition. Unfortunately, the circuit exhibits ripple precharge. The input to the second gate is initially high and it takes two gate delay before \( I_{n2} \) is driven low. During this time, the second gate cannot precharge its output to \( V_{DD} \) since the pull-down device is fighting the precharge device. Similarly, the third gate has to wait till the second gate precharges before it can start precharging, etc. Therefore the time taken to precharge the logic circuit is equal to the critical path of the logic circuit. Another important problem here is the static power dissipation due to the fight between the pull-up and pull-down devices. As a result of this, the evaluation device is almost always placed in the circuit.

**Dealing with the Non-inverting Property of Domino Logic**

A major limitation in Domino logic is that only non-inverting logic can be implemented. This is due to the inclusion of the static inverter at the output of each dynamic gate. This requirement has limited the widespread use of pure Domino logic. There are several ways to deal with the problem of non-inverting logic requirement. Figure 6.63 shows one approach to the problem, which basically involves reorganizing the logic using

---

**Figure 6.62** Effect of ripple precharge when the evaluation transistor is removed. The circuit also exhibits static power dissipation.
simple boolean transforms, such as De Morgan’s Law. Unfortunately, this sort of optimization is not always possible, and more general schemes must be used.

A general, but expensive, approach to solving the problem of the non-inverting logic requirement is the use of dual-rail coding. Dual-rail Domino is similar in concept to the DCVS structure discussed earlier, but uses a precharged load instead of a static cross-coupled PMOS load. Figure 6.64 shows the circuit schematic of a simple AND/NAND differential logic gate. Note that all inputs come from other differential Domino gates and therefore, all inputs are low during the precharge phase and make a conditional transition from 0 to 1. Using differential Domino, it is possible to implement any arbitrary function. Differential Domino gates consume significant power since they have a guaranteed transition every single clock cycle, regardless of the input values since either $O$ or $\overline{O}$ will make a 0 to 1 transition. The function of transistors $M_{f1}$ and $M_{f2}$ is to keep the circuit static when the clock is high for extended periods of time. Notice that though there is a cross coupled PMOS pair, this circuit is not ratioed! Such a differential approach is very popular and is used in several commercial microprocessors.

**Optimization of Domino Logic Gates**

There are several optimizations that can be performed on Domino logic gates. The most obvious performance optimization involves the sizing of transistors in the static
inverter. With the inclusion of the evaluation devices in Domino circuits, all gates pre-
charge in parallel and the precharge operation is only two gates as the output of the
dynamic gate charges to $V_{DD}$ and the inverter output is driven low. The critical path during
evaluation happens through the pull-down path of the dynamic gate and the PMOS pull-up
path of the static inverter. Therefore, to speed up the circuit, the beta ratio of the static
inverter should be made high so that the switching threshold is close to $V_{DD}$. This can be
accomplished by using a small (minimum) sized NMOS and a large PMOS device. The
minimum sized NMOS does not affect the performance since the precharge happens in
parallel. The only disadvantage of using a large beta ratio is a reduction in noise margin is
reduced. Issues such as charge sharing and backgate coupling can cause the dynamic gate
voltage to drop below $V_{DD}$ and hence a high switching threshold can cause an incorrect
evaluation. The device sizing of the inverter should simultaneously consider the reduced
noise margin and performance.

Numerous circuit variations of Domino circuits have been proposed [Bernstein98].
One optimization that reduces area is Multiple Output Domino Logic. The basic concept is
illustrated is Figure 6.65. The idea is to exploit the partial trees in the pull-down network
and the fact that certain outputs are subsets of other outputs. In this example, $O3 = C+D$ is
used in all three outputs, and hence it is implemented in the bottom of the pull-down net-
work. Since $O2$ is simply $B \cdot O3$, it is connected in series with the logic for $O3$. Notice that
the internal nodes have to be precharged to $V_{DD}$ since the outputs are based on internal
nodes. Given that the internal node precharge to $V_{DD}$, the number of devices driving pre-
charge devices is not reduced. However, the number of devices driving the evaluation
switch is reduced since the overhead of evaluation device is amortized over multiple out-
puts. Also the number of transistors required in the circuit is clearly reduced since the
logic for $O3$ and $O2$ must be duplicated in implementing $O1$ if the three logic functions
were implemented independently.

Another optimization of the generic Domino logic gate is Compound Domino (Figure
6.66). The basic goal of this style is to minimize the number of devices in a dynamic
logic gate. Instead of each dynamic gate driving a static inverter, it is possible to combine
the output of multiple dynamic gates using complex static CMOS gates as shown in Figure
6.66. In this example, we have three dynamic gates which include $o1 = A B \overline{C}, o2 = D
\overline{E F}$ and $o3 = \overline{G H}$. The output of three dynamic structures are combined using a complex

![Figure 6.65 Multiple output Domino](image-url)
CMOS static gate whose function is \( O = (o_1 + o_2) o_3 \). For this example, this equates to \( O = A B C D E F + G H \).

Compound Domino is a useful tool for constructing complex dynamic logic gates. Large dynamic stacks are replaced using using parallel small fan-in structures and complex CMOS gates. For example, a large fan-in Domino AND can be implemented as parallel dynamic NAND structures with lower fan-in that are combined using a static NOR gate. One important consideration in Compound Domino is the problem associated with backgate coupling. Care must be taken to ensure that the dynamic nodes are not affected by the coupling between the output of the static gates and the output of dynamic nodes.

**Figure 6.66** Compound Domino logic where complex static gates can be placed at the output of dynamic gates.

**np-CMOS**

The Domino logic presented in the previous section has the disadvantage that each dynamic gate requires an extra static inverter in the critical path to make the circuit functional. np-CMOS, provides an alternate race-free approach to cascading dynamic logic by using two flavors (N-tree and P-tree) of dynamic logic. In a P-tree logic gate, PMOS devices are used to build a pull-up network that has a series evaluation device (Figure 6.67) ([Goncalvez83, Friedman84, Lee86]). A predischarge device drives the output low during precharge and the output conditionally makes a \( 0 \rightarrow 1 \) transition based on its inputs.

np-CMOS logic exploits the duality between N-tree and P-tree logic gates to eliminate races. The N-tree gates are controlled by CLK and P-tree gates are controlled using \( CLK \). N-tree gates can directly drive P-tree gates, but similar to Domino, N-tree outputs must go through an inverter if another N-tree gate is to be driven. During the precharge phase (\( CLK = 0 \)), the output of N-tree logic, \( Out_1 \), is charge up to \( V_{DD} \) and the output of P-tree, \( Out_2 \), is predischarged to 0V. Since N-tree logic drives PMOS pull-up devices, the PUN of the P-tree is turned off. If some of the P-tree inputs (i.e., the N-tree outputs) are discharged during the evaluation period (\( CLK =1 \)), and the PUN turns on, the output of P-type gates make a 0 to 1 transition. In a similar way, N-tree blocks can follow P-tree
blocks without any problems, as the precharge value of inputs equals 0. A disadvantage of the \textit{np}-CMOS logic style is that the P-tree blocks are slower than the N-tree modules, due to the lower mobility of the PMOS transistors in the logic network. Equalizing the propagation delays requires extra area.

6.4 Perspective: How to Choose a Logic Style

In the preceding sections, we have discussed several gate-implementation approaches using the CMOS technology. Each of the circuit styles has its advantages and disadvantages. Which one to select depends upon the primary requirement: ease of design, robustness, area, speed, or power dissipation. No single style optimizes all these measures at the same time. Even more, the approach of choice can vary from logic function to logic function.

The static approach has the advantage of being robust in the presence of noise. This makes the design process rather trouble-free and amenable to a high degree of automation. This ease-of-design does not come for free: for complex gates with a large fan-in, complementary CMOS becomes expensive in terms of area and performance. Alternative static logic styles have therefore been devised. Pseudo-NMOS is simple and fast at the expense of a reduced noise margin and static power dissipation. Pass-transistor logic is attractive for the implementation of a number of specific circuits, such as multiplexers and XOR-dominated logic such as adders.

Dynamic logic, on the other hand, makes it possible to implement fast and small complex gates. This comes at a price. Parasitic effects such as charge sharing make the design process a precarious job. Charge leakage forces a periodic refresh, which puts a lower bound on the operating frequency of the circuit.

The current trend is towards an increased use of complementary static CMOS. This tendency is inspired by the increased use of design-automation tools at the logic design level. These tools emphasize optimization at the logic rather than the circuit level and put
a premium on robustness. Another argument is that static CMOS is more amenable to voltage scaling than some of the other approaches discussed in this chapter.

6.5 Leakage in Low Voltage Systems

As power supply voltage scale, the device thresholds must scale to maintain performance. Figure 6.68a shows a plot of power supply and \( V_T \) required in order to maintain a fixed performance level. This tradeoff is not without penalty however, as subthreshold leakage currents increase exponentially as \( V_T \) is reduced. The leakage can be approximated as follows:

\[
I_{\text{leakage}} = I_0 e^{(V_{GS} - V_T) / nV_{th}} = I_0 10^{-\frac{V_{GS} - V_T}{S}}
\]

(6.34)

The subthreshold \( S = nV_{th} \ln(10) \). For a typical technology with a subthreshold slope of 100 mV/decade, each 100mV decrease in \( V_T \) will cause an order magnitude change in leakage currents. The leakage of an inverter is current of the NMOS when \( V_{in} = 0V \) and the output is at \( V_{DD} \). The exponential increase in leakage when the threshold is decreased is demonstrated in Figure 4.26b.

Leakage currents are particularly a concern for event driven computation in which intermittent computation activity triggered by external events is separated by long periods of inactivity (e.g., the processor in a cellular phone or PDA remains in the idle mode for majority of the time). While the processor is shutdown, the system should ideally consume near zero power. This is only possible if the devices consume low levels of leakage power - i.e., the devices have a high threshold voltage. However for low voltage high-performance operation, reduced threshold devices are required. To satisfy the contradicting requirements of high-performance during active periods and low-standby leakage, several device technologies have recently been introduced. This includes the control of threshold voltages in triple-well CMOS using backgate effect and the use of multiple threshold devices.

\[ I_{\text{leakage}} = I_0 e^{(V_{GS} - V_T) / nV_{th}} = I_0 10^{-\frac{V_{GS} - V_T}{S}} \]

Figure 6.68 Voltage Scaling (\( V_{DD}/V_T \) on delay and leakage)
As with propagation delay, static properties, and switching activity, leakage of logic gates has a strong dependence on the input patterns. The leakage is a function of the circuit topology and the value of inputs. This is due to the fact that $V_T$ depends on body bias ($V_{BS}$) we observe that sub-threshold leakage of a long channel MOS transistor depends on gate drive ($V_{GS}$) and body bias ($V_{BS}$). So, in an inverter with $IN = 0$ the sub-threshold leakage of the inverter will be set by the NMOS transistor with its $V_{GS} = V_{BS} = 0$ V. In more complex CMOS gates the leakage current will depend on the input vector, for example, one can show (Eq. (6.35)) that the sub-threshold leakage current of a two-input NAND gate will be the least when $A = B = 0$. Under this condition the intermediate node in will settle to,

$$V_X \approx V_T \ln (1 + n)$$  \hspace{1cm} (6.35)

The NAND gate sub-threshold leakage then will be set by the NMOS transistor with its $V_{GS} = V_{BS} = -V_X$. Clearly, the sub-threshold leakage under this condition will be slightly smaller than that of the inverter or a stand-alone NMOS transistor’s $I_{OFF}$. This reduction in sub-threshold leakage due to stacked transistors is called the stack effect. Figure 6.69 shows the leakage components for a simple two input NAND gate.

![Figure 6.69 Sub-threshold leakage reduction due to stack effect in a two-input NAND gate using long channel transistors.](image)

In short channel MOS transistors the sub-threshold leakage current depends not only on the gate drive ($V_{GS}$) and body bias ($V_{BS}$), but also depends strongly on the drain voltage ($V_{DS}$). Threshold voltage of short channel MOS transistors decrease with increase in $V_{DS}$ due to drain induced barrier lowering (DIBL). Typical value for DIBL can range from 20-150 mV change in $V_T$ per volt change in $V_{DS}$.

Figure 6.70 shows the decrease in sub-threshold leakage due to: (i) decrease in gate drive - point $A$ to $B$ and (ii) increase in body bias - point $A$ to $C$, similar to long channel MOS transistors. It also illustrates the increase in sub-threshold leakage due to increase in drain voltage - point $A$ to $D$. Because of this reason the impact of stack effect for leakage reduction will be more significant in short channel MOS transistors. Consider the two-input NAND gate in Figure 6.69 when both $M_1$ and $M_2$ are off. From the load line in Figure 6.71 we can see that in steady state $V_X$ will settle to $\sim 100$ mV. So the steady state sub-threshold leakage in the NAND gate will be due to $V_{GS} = V_{BS} = -100$ mV and $V_{DS} = V_{DD} - 100$ mV which is 20X smaller than that leakage of a stand-alone NMOS transistor with $V_{GS} = V_{BS} = 0$ mV and $V_{DS} = V_{DD}$ [Ye98]. Because of enhanced stack effect in short channel MOS transistors the sub-threshold leakage in circuits with stacks will be significantly smaller than individual devices. Note that maximum leakage reduction due to stack effect
happens when all the transistors in the stack are off and the intermediate node voltage reaches its steady state value. So one can reduce standby leakage in a VLSI system by forcing stack effect in as many gates as possible. Since the intermediate node gets charged or discharged to its steady state value through sub-threshold currents time constant to realize maximum stack effect can be long, depending on the initial node value. A key challenge is to determine the primary input vector that minimizes the leakage effect.

**Problem 6.8 Computing \( V_X \)**

Eq. (6.35) represents intermediate node voltage for a two-input NAND with less than 10% error, when \( A = B = 0 \). Derive Eq. (6.35) assuming (i) \( V_T \) and \( I_o \) of \( M_1 \) and \( M_2 \) are approximately equal, (ii) NMOS transistors are identically sized, and (iii) \( n \approx 1.5 \). Explain the tem-
6.6 Summary

In this chapter, we have extensively analyzed the behavior and performance of combinational CMOS digital circuits with regard to area, speed, and power.

- Static complementary CMOS combines dual pull-down and pull-up networks, only one of which is enabled at any time.

- The performance of a CMOS gate is a strong function of fan-in. Techniques to deal with fan-in include transistor sizing, input reordering, and partitioning. The speed is also a linear function of the fan-out. Extra buffering is needed for large fan-outs.

- The ratioed logic style consists of an active pull-down (up) network connected to a load device. This results in a substantial reduction in gate complexity at the expense of static power consumption and an asymmetrical response. Careful transistor sizing is necessary to maintain sufficient noise margins. The most popular approaches in this class are the pseudo-NMOS techniques and the differential DCVSL, which requires complementary signals.

- Pass-transistor logic implements a logic gate as a simple switch network. This results in very simple implementations for some logic functions. Long cascades of switches are to be avoided due to a quadratic increase in delay with respect to the number of elements in the chain. NMOS-only pass-transistor logic produces even simpler structures, but might suffer from static power consumption and reduced noise margins. This problem can be addressed by adding a level-restoring transistor.

- The operation of dynamic logic is based on the storage of charge on a capacitive node and the conditional discharging of that node as a function of the inputs. This calls for a two-phase scheme, consisting of a precharge followed by an evaluation step. Dynamic logic trades off noise margin for performance. It is sensitive to parasitic effects such as leakage, charge redistribution, and clock feedthrough. Cascading dynamic gates can cause hazards and should be addressed carefully.

- The power consumption of a logic network is strongly related to the switching activity of the network. This activity is a function of the input statistics, the network topology, and the logic style.

- Sources of power consumption such as glitches and short-circuit currents can be minimized by careful circuit design and transistor sizing.

- Power consumption is minimized by reducing the supply voltage, which increases the delay. Trading off area for power is a way to compensate for that performance loss.
• Threshold voltage scaling is required for low-voltage operation. Leakage control is critical for low-voltage operation

6.7 To Probe Further

The topic of (C)MOS logic styles is treated extensively in the literature. Numerous texts have been devoted to the issue. Some of the most comprehensive treatments can be found in [Glasser85], [Annaratone86], [Elmasry91], [Uyemura92], and [Weste93]. Regarding the intricacies of high-performance design, [Shoji88] offers the most in-depth discussion of the optimization and analysis of digital MOS circuits. The topic of power minimization is relatively new. Excellent reference works are [Chandrakasan95] and [Rabaey95].

Innovations in the MOS logic area are typically published in the proceedings of the ISSCC Conference and the VLSI circuits symposium, as well as the IEEE Journal of Solid State Circuits (especially the November issue).

REFERENCES

Section 6.7 To Probe Further


Implementation techniques for flip-flops, latches, oscillators, pulse generators, and Schmitt triggers

Static versus dynamic realization

Choosing clocking strategies

7.1 Introduction
7.2 Timing Metrics for Sequential Circuits
7.3 Classification of Memory Elements
7.4 Static Latches and Registers
  7.4.1 The Bistability Principle
  7.4.2 SR Flip-Flops
  7.4.3 Multiplexer Based Latches
  7.4.4 Master-Slave Based Edge Triggered Register
  7.4.5 Non-ideal clock signals
  7.4.6 Low-Voltage Static Latches
7.5 Dynamic Latches and Registers
  7.5.1 Dynamic Transmission-Gate Based Edge-triggered Registers
  7.5.2 C2MOS Dynamic Register: A Clock Skew Insensitive Approach
  7.5.3 True Single-Phase Clocked Register (TSPCR)
7.6 Pulse Registers
  6.4.2 The C2MOS Latch
  7.8.2 NORA-CMOS—A Logic Style for Pipelined Structures
  7.5.3 True Single-Phase Clocked Register (TSPCR)
7.7 Sense-Amplifier Based Registers
7.8 Pipelining: An approach to optimize sequential circuits
Section 271

7.8.1 Latch- vs. Register-Based Pipelines
7.8.2 NORA-CMOS—A Logic Style for Pipelined Structures

7.9 Non-Bistable Sequential Circuits
7.9.1 The Schmitt Trigger
7.9.2 Monostable Sequential Circuits
7.9.3 Astable Circuits

7.10 Perspective: Choosing a Clocking Strategy

7.11 Summary

7.12 To Probe Further

7.13 Exercises and Design Problems
7.1 Introduction

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

Figure 7.1 shows a block diagram of a generic finite state machine (FSM) that consists of combinational logic and registers that hold the system state. The system depicted here belongs to the class of synchronous sequential systems, in which all registers are under control of a single global clock. The outputs of the FSM are a function of the current inputs and the current state. The next state is determined based on the current state and the current inputs and is fed to the inputs of registers. On the rising edge of the clock, the next state bits are copied to the outputs of the registers (after some propagation delay), and a new cycle begins. The register then ignores changes in the input signals until the next rising edge. In general, registers can be positive edge-triggered (where the input data is copied on the positive edge) or negative edge-triggered (where the input data is copied on the negative edge of the clock, as is indicated by a small circle at the clock input).

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

7.2 Timing Metrics for Sequential Circuits

There are three important timing parameters associated with a register as illustrated in Figure 7.2. The set-up time \( t_{su} \) is the time that the data inputs \( D \) input must be valid before the clock transition (this is, the 0 to 1 transition for a positive edge-triggered register). The hold time \( t_{hold} \) is the time the data input must remain valid after the clock edge. Assum-
Section 7.3 Classification of Memory Elements

273

ing that the set-up and hold-times are met, the data at the D input is copied to the Q output after a worst-case propagation delay (with reference to the clock edge) denoted by \( t_{c-q} \).

Given the timing information for the registers and the combination logic, some system-level timing constraints can be derived. Assume that the worst-case propagation delay of the logic equals \( t_{\text{logic}} \), while its minimum delay (also called the contamination delay) is \( t_{c,d} \). The minimum clock period \( T \), required for proper operation of the sequential circuit is given by

\[
T \geq t_{c-q} + t_{\text{logic}} + t_{su}
\]  

(7.1)

The hold time of the register imposes an extra constraint for proper operation,

\[
t_{c,d} + t_{c,d} \geq t_{\text{hold}}
\]  

(7.2)

where \( t_{c,d} \) is the minimum propagation delay (or contamination delay) of the register.

As seen from Eq. (7.1), it is important to minimize the values of the timing parameters associated with the register, as these directly affect the rate at which a sequential circuit can be clocked. In fact, modern high-performance systems are characterized by a very-low logic depth, and the register propagation delay and set-up times account for a significant portion of the clock period. For example, the DEC Alpha EV6 microprocessor [Gieseke97] has a maximum logic depth of 12 gates, and the register overhead stands for approximately 15% of the clock period. In general, the requirement of Eq. (7.2) is not hard to meet, although it becomes an issue when there is little or no logic between registers, (or when the clocks at different registers are somewhat out of phase due to clock skew, as will be discussed in a later Chapter).

7.3 Classification of Memory Elements

Foreground versus Background Memory

At a high level, memory is classified into background and foreground memory. Memory that is embedded into logic is foreground memory, and is most often organized as individual registers of register banks. Large amounts of centralized memory core are referred to as background memory. Background memory, discussed later in this book, achieves
DESIGNING SEQUENTIAL LOGIC CIRCUITS  Chapter 7

higher area densities through efficient use of array structures and by trading off performance and robustness for size. In this chapter, we focus on foreground memories.

Static versus Dynamic Memory

Memories can be static or dynamic. Static memories preserve the state as long as the power is turned on. Static memories are built using positive feedback or regeneration, where the circuit topology consists of intentional connections between the output and the input of a combinational circuit. Static memories are most useful when the register won’t be updated for extended periods of time. An example of such is configuration data, loaded at power-up time. This condition also holds for most processors that use conditional clocking (i.e., gated clocks) where the clock is turned off for unused modules. In that case, there are no guarantees on how frequently the registers will be clocked, and static memories are needed to preserve the state information. Memory based on positive feedback fall under the class of elements called multivibrator circuits. The bistable element, is its most popular representative, but other elements such as monostable and astable circuits are also frequently used.

Dynamic memories store state for a short period of time—on the order of milliseconds. They are based on the principle of temporary charge storage on parasitic capacitors associated with MOS devices. As with dynamic logic discussed earlier, the capacitors have to be refreshed periodically to annihilate charge leakage. Dynamic memories tend to be simpler, resulting in significantly higher performance and lower power dissipation. They are most useful in datapath circuits that require high performance levels and are periodically clocked. It is possible to use dynamic circuitry even when circuits are conditionally clocked, if the state can be discarded when a module goes into idle mode.

Latches vs. Registers

A latch is an essential component in the construction of an edge-triggered register. It is level-sensitive circuit that passes the $D$ input to the $Q$ output when the clock signal is high. This latch is said to be in transparent mode. When the clock is low, the input data sampled on the falling edge of the clock is held stable at the output for the entire phase, and the latch is in hold mode. The inputs must be stable for a short period around the falling edge of the clock to meet set-up and hold requirements. A latch operating under the above conditions is a positive latch. Similarly, a negative latch passes the $D$ input to the $Q$ output when the clock signal is low. The signal waveforms for a positive and negative latch are shown in Figure 7.3. A wide variety of static and dynamic implementations exists for the realization of latches.

Contrary to level-sensitive latches, edge-triggered registers only sample the input on a clock transition — 0-to-1 for a positive edge-triggered register, and 1-to-0 for a negative edge-triggered register. They are typically built using the latch primitives of Figure 7.3. A most-often recurring configuration is the master-slave structure that cascades a positive and negative latch. Registers can also be constructed using one-shot generators of the clock signal (“glitch” registers), or using other specialized structures. Examples of these are shown later in this chapter.
7.4 Static Latches and Registers

7.4.1 The Bistability Principle

Static memories use positive feedback to create a bistable circuit — a circuit having two stable states that represent 0 and 1. The basic idea is shown in Figure 7.4a, which shows two inverters connected in cascade along with a voltage-transfer characteristic typical of such a circuit. Also plotted are the VTCs of the first inverter, that is, $V_{o1}$ versus $V_{i1}$, and the second inverter ($V_{o2}$ versus $V_{i1}$). The latter plot is rotated to accentuate that $V_{i2} = V_{o1}$.

Assume now that the output of the second inverter $V_{o2}$ is connected to the input of the first $V_{i1}$, as shown by the dotted lines in Figure 7.4a. The resulting circuit has only three possi-
ble operation points \((A, B,\) and \(C\)), as demonstrated on the combined VTC. The following important conjecture is easily proven to be valid:

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

Suppose that the cross-coupled inverter pair is biased at point \(C\). A small deviation from this bias point, possibly caused by noise, is amplified and \textit{regenerated} around the circuit loop. This is a consequence of the gain around the loop being larger than 1. The effect is demonstrated in Figure 7.5a. A small deviation \(\delta\) is applied to \(V_{i1}\) (biased in \(C\)). This deviation is amplified by the gain of the inverter. The enlarged divergence is applied to the second inverter and amplified once more. The bias point moves away from \(C\) until one of the operation points \(A\) or \(B\) is reached. In conclusion, \(C\) is an unstable operation point. Every deviation (even the smallest one) causes the operation point to run away from its original bias. The chance is indeed very small that the cross-coupled inverter pair is biased at \(C\) and stays there. Operation points with this property are termed \textit{metastable}.

On the other hand, \(A\) and \(B\) are stable operation points, as demonstrated in Figure 7.5b. In these points, the \textit{loop gain is much smaller than unity}. Even a rather large deviation from the operation point is reduced in size and disappears.

Hence the cross-coupling of two inverters results in a \textit{bistable} circuit, that is, a circuit with two stable states, each corresponding to a logic state. The circuit serves as a memory, storing either a 1 or a 0 (corresponding to positions \(A\) and \(B\)).

In order to change the stored value, we must be able to bring the circuit from state \(A\) to \(B\) and vice-versa. Since the precondition for stability is that the loop gain \(G\) is smaller than unity, we can achieve this by making \(A\) (or \(B\)) temporarily unstable by increasing \(G\) to a value larger than 1. This is generally done by applying a trigger pulse at \(V_{i1}\) or \(V_{i2}\). For instance, assume that the system is in position \(A\) \((V_{i1} = 0, V_{i2} = 1)\). Forcing \(V_{i1}\) to 1 causes both inverters to be on simultaneously for a short time and the loop gain \(G\) to be larger than 1. The positive feedback regenerates the effect of the trigger pulse, and the circuit moves to the other state (\(B\) in this case). The width of the trigger pulse need be only a little
larger than the total propagation delay around the circuit loop, which is twice the average propagation delay of the inverters.

In summary, a bistable circuit has two stable states. In absence of any triggering, the circuit remains in a single state (assuming that the power supply remains applied to the circuit), and hence remembers a value. A trigger pulse must be applied to change the state of the circuit. Another common name for a bistable circuit is flip-flop (unfortunately, an edge-triggered register is also referred to as a flip-flop).

### 7.4.2 SR Flip-Flops

The cross-coupled inverter pair shown in the previous section provides an approach to store a binary variable in a stable way. However, extra circuitry must be added to enable control of the memory states. The simplest incarnation accomplishing this is the well-known SR—or set-reset—flip-flop, an implementation of which is shown in Figure 7.6a. This circuit is similar to the cross-coupled inverter pair with NOR gates replacing the inverters. The second input of the NOR gates is connected to the trigger inputs (S and R), that make it possible to force the outputs Q and $\bar{Q}$ to a given state. These outputs are complimentary (except for the $SR = 11$ state). When both $S$ and $R$ are 0, the flip-flop is in a quiescent state and both outputs retain their value (a NOR gate with one of its input being 0 looks like an inverter, and the structure looks like a cross coupled inverter). If a positive (or 1) pulse is applied to the $S$ input, the $Q$ output is forced into the 1 state (with $\bar{Q}$ going to 0). Vice versa, a 1 pulse on $R$ resets the flip-flop and the $Q$ output goes to 0.

These results are summarized in the characteristic table of the flip-flop, shown in Figure 7.6c. The characteristic table is the truth table of the gate and lists the output states as functions of all possible input conditions. When both $S$ and $R$ are high, both $Q$ and $\bar{Q}$ are forced to zero. Since this does not correspond with our constraint that $Q$ and $\bar{Q}$ must be complementary, this input mode is considered to be forbidden. An additional problem with this condition is that when the input triggers return to their zero levels, the resulting state of the latch is unpredictable and depends on whatever input is last to go low. Finally, Figure 7.6 shows the schematics symbol of the SR flip-flop.
Problem 7.1  *SR* Flip-Flop Using NAND Gates

An *SR* flip-flop can also be implemented using a cross-coupled NAND structure as shown in Figure 7.7. Derive the truth table for such an implementation.

The *SR* flip-flops discussed so far are asynchronous, and do not require a clock signal. Most systems operate in a synchronous fashion with transition events referenced to a clock. One possible realization of a clocked *SR* flip-flop—a level-sensitive positive latch—is shown in Figure 7.8. It consists of a cross-coupled inverter pair, plus 4 extra transistors to drive the flip-flop from one state to another and to provide clocked operation. Observe that the number of transistors is identical to the implementation of Figure 7.6, but the circuit has the added feature of being clocked. The drawback of saving some transistors over a fully-complimentary CMOS implementation is that transistor sizing becomes critical in ensuring proper functionality. Consider the case where *Q* is high and an *R* pulse is applied. The combination of transistors *M*₄, *M*₇, and *M*₈ forms a ratioed inverter. In order to make the latch switch, we must succeed in bringing *Q* below the switching threshold of the inverter *M*₁-*M*₂. Once this is achieved, the positive feedback causes the flip-flop to invert states. This requirement forces us to increase the sizes of transistors *M*₅, *M*₆, *M*₇, and *M*₈.

The presented flip-flop does not consume any static power. In steady-state, one inverter resides in the high state, while the other one is low. No static paths between *V*₉ and GND can exist except during switching.

Example 7.1  Transistor Sizing of Clocked SR Latch

Assume that the cross-coupled inverter pair is designed such that the inverter threshold *V*₉ is located at *V*₂/₂. For a 0.25 µm CMOS technology, the following transistor sizes were selected: (*W/L*)₄₁ = (*W/L*)₄₂ = (0.5µm/0.25µm), and (*W/L*)₄₃ = (*W/L*)₄₄ = (1.5µm/0.25µm). Assuming *Q* = 0, we determine the minimum sizes of *M*₅, *M*₆, *M*₇, and *M*₈ to make the device switchable.
Section 7.4 Static Latches and Registers

To switch the latch from the $Q = 0$ to the $Q = 1$ state, it is essential that the low level of the ratioed, pseudo-NMOS inverter ($M_5$-$M_6$) be below the switching threshold of the inverter $M_2$-$M_4$ that equals $V_{DD}/2$. It is reasonable to assume that as long as $V_Q > V_{th}$, $V_Q$ equals 0, and the gate of transistor $M_2$ is grounded. The boundary conditions on the transistor sizes can be derived by equating the currents in the inverter for $V_Q = V_{DD}/2$, as given in Eq. (7.3) (this ignores channel length modulation). The currents are determined by the saturation current since 

$$k'_{p} \left( \frac{W}{L} \right)_{5-6} \left( V_{DD} - V_{th} \right) V_{DSAT} - \frac{V_{DSAT}^2}{2} = k'_{p} \left( \frac{W}{L} \right)_{2} \left( - V_{DD} - V_{th} \right) V_{DSAT} - \frac{V_{DSAT}^2}{2}$$ (7.3)

Using the parameters for the 0.25 µm process, Eq. (7.3) results in the constraint that the effective ($W/L$)$_{5-6}$ ≥ 2.26. This implies that the individual device ratio for $M_5$ or $M_6$ must be larger that approximately 4.5. Figure 7.9a shows the DC plot of $V_Q$ as a function of the individual device sizes of $M_5$ and $M_6$. We notice that the individual device ratio of greater than 3 is sufficient to bring the $\overline{Q}$ voltage to the inverter switching threshold. The difference between the manual analysis and simulation arises due to second order effects such as DIBL and channel length modulation. Figure 7.9b, which shows the transient response for different device sizes. The plot confirms that an individual $W/L$ ratio of greater than 3 is required to overpower the feedback and switch the state of the latch.

The positive feedback effect makes a manual derivation of propagation delay of the $SR$ latch non-trivial. Some simplifications are therefore necessary. Consider, for instance, the latch of Figure 7.8, where $Q$ and $\overline{Q}$ are set to 0 and 1, respectively. A pulse is applied at node $S$, causing the latch to toggle. In the first phase of the transient, node $\overline{Q}$ is being pulled down by transistors $M_5$ and $M_6$. Since node $Q$ is initially low, the PMOS device $M_2$ is on while $M_4$ is off. The transient response is hence determined by the pseudo-NMOS inverter formed by ($M_5$-$M_6$) and $M_2$. Once $\overline{Q}$ reaches the switching threshold of the CMOS inverter $M_2$-$M_4$, this inverter reacts and the positive feedback comes into action, turning $M_2$ off and $M_4$ on. This accelerates the pulling down of node $\overline{Q}$. From this analysis, we can
derive that the propagation delay of node $\overline{Q}$ is approximately equal to the delay of the pseudo-NMOS inverter formed by ($M_5$-$M_6$) and $M_2$. To obtain the delay for node $Q$, it is sufficient to add the delay of the complementary CMOS inverter $M_3$-$M_4$.

**Example 7.2  Propagation Delay of Static SR Flip-Flop**

The transient response of the latch in Figure 7.8, as obtained from simulation, is plotted in Figure 7.10. The devices are sized as described in Example 7.1, and a load of a single inverter is assumed for each latch output. The flip-flop is initially in the reset state, and an $S$-pulse is applied. As we can observe, this results first in a discharging of the $\overline{Q}$ output while $Q$ stays at 0. Once the switching threshold of the inverter $M_3$-$M_4$ is reached, the $Q$ output starts to rise. The delay of this transient is solely determined by the $M_3$-$M_4$ inverter, which is hampered by the slow rise time at its input. From the simulation results, we can derive that $t_{\overline{Q}}$ and $t_Q$ equal 120 psec and 230 psec, respectively.

**Problem 7.2  Complimentary CMOS SR FF**

Instead of using the modified SR FF of Figure 7.8, it is also possible to use complementary logic to implement the clocked SR FF. Derive the transistor schematic (which consists of 12 transistors). This circuit is more complex, but switches faster and consumes less switching power. Explain why.

### 7.4.3 Multiplexer Based Latches

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

Figure 7.11 shows an implementation of static positive and negative latches based on multiplexers. For a negative latch, when the clock signal is low, the input 0 of the mult-
tplexer is selected, and the $D$ input is passed to the output. When the clock signal is high, the input 1 of the multiplexer, which connects to the output of the latch, is selected. The feedback holds the output stable while the clock signal is high. Similarly in the positive latch, the $D$ input is selected when clock is high, and the output is held (using feedback) when clock is low.

A transistor level implementation of a positive latch based on multiplexers is shown in Figure 7.12. When $CLK$ is high, the bottom transmission gate is on and the latch is transparent - that is, the $D$ input is copied to the $Q$ output. During this phase, the feedback loop is open since the top transmission gate is off. Unlike the $SR$ FF, the feedback does not have to be overridden to write the memory and hence sizing of transistors is not critical for realizing correct functionality. The number of transistors that the clock touches is important since it has an activity factor of 1. This particular latch implementation is not particularly efficient from this metric as it presents a load of 4 transistors to the $CLK$ signal.

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

The most common approach for constructing an edge-triggered register is to use a master-slave configuration as shown in Figure 7.14. The register consists of cascading a negative latch (master stage) with a positive latch (slave stage). A multiplexer based latch is used in this particular implementation, though any latch can be used to realize the master and slave stages. On the low phase of the clock, the master stage is transparent and the D input is passed to the master stage output, Q_M. During this period, the slave stage is in the hold mode, keeping its previous value using feedback. On the rising edge of the clock, the master slave stops sampling the input, and the slave stage starts sampling. During the high phase of the clock, the slave stage samples the output of the master stage (Q_M), while the master stage remains in a hold mode. Since Q_M is constant during the high phase of the clock, the output Q makes only one transition per cycle. The value of Q is the value of D right before the rising edge of the clock, achieving the positive edge-triggered effect. A negative edge-triggered register can be constructed using the same principle by simply switching the order of the positive and negative latch (i.e., placing the positive latch first).

A complete transistor level implementation of a the master-slave positive edge-triggered register is shown in Figure 7.15. The multiplexer is implemented using transmission gates as discussed in the previous section. When clock is low (CLK = 1), T_1 is on and T_2 is off, and the D input is sampled onto node Q_M. During this period, T_3 is off and T_4 is on and the cross-coupled inverters (I_5, I_6) holds the state of the slave latch. When the clock goes high, the master stage stops sampling the input and goes into a hold mode. T_1 is off and T_2...
is on, and the cross coupled inverters $I_3$ and $I_4$ holds the state of $Q_M$. Also, $T_3$ is on and $T_4$ is off, and $Q_M$ is copied to the output $Q$.

![Figure 7.15 Transistor-level implementation of a master-slave positive edge-triggered register using multiplexers.]

Problem 7.3 Optimization of the Master Slave Register

It is possible to remove the inverters $I_1$ and $I_2$ from Figure 7.3 without loss of functionality. Is there any advantage in including these inverters in the implementation?

**Timing Properties of the multiplexer Bases Master-Slave Register.** As discussed earlier, there are three important timing metrics in registers: the *set up time*, the *hold time* and the *propagation delay*. It is important to understand these factors that affect the timing parameters and develop the intuition to manually estimate the parameters. Assume that the *propagation delay* of each inverter is $t_{pd\_inv}$ and the *propagation delay* of the transmission gate is $t_{pd\_tx}$. Also assume that the *contamination delay* is 0 and the inverter delay to derive $CLK$ from $CLK$ has a delay equal to 0.

The *set-up time* is the time before the rising edge of the clock that the input data $D$ must become valid. Another way to ask the question is how long before the rising edge does the $D$ input have to be stable such that $Q_M$ samples the value reliably. For the transmission gate multiplexer-based register, the input $D$ has to propagate through $I_1$, $T_1$, $I_3$ and $I_2$ before the rising edge of the clock. This is to ensure that the node voltages on both terminals of the transmission gate $T_2$ are at the same value. Otherwise, it is possible for the cross-coupled pair $I_2$ and $I_3$ to settle to an incorrect value. The *set-up time* is therefore equal to $3 \cdot t_{pd\_inv} + t_{pd\_tx}$.

The *propagation delay* is the time for the value of $Q_M$ to propagate to the output $Q$. Note that since we included the delay of $I_2$ in the *set-up time*, the output of $I_4$ is valid before the rising edge of clock. Therefore the delay $t_{c-q}$ is simply the delay through $T_3$ and $I_6$ ($t_{c-q} = t_{pd\_tx} + t_{pd\_inv}$).

The *hold time* represents the time that the input must be held stable after the rising edge of the clock. In this case, the transmission gate $T_1$ turns off when clock goes high and therefore any changes in the $D$-input after clock going high are not seen by the input. Therefore, the *hold time* is 0.
Example 7.3 Timing analysis using SPICE.

To obtain the set-up time of the register using SPICE, we progressively skew the input with respect to the clock edge until the circuit fails. Figure 7.16 shows the set-up time simulation assuming a skew of 210 psec and 200 psec. For the 210 psec case, the correct value of input $D$ is sampled (in this case, the $Q$ output remains at the value of $V_{DD}$). For a skew of 200 psec, an incorrect value propagates to the output (in this case, the $Q$ output transitions to 0). Node $Q_M$ starts to go high while the output of $I_2$ (the input to transmission gate $T_2$) starts to fall. However, the clock is enabled before the two nodes across the transmission gate ($T_2$) settle to the same value and therefore, results in an incorrect value written into the master latch. The set-up time for this register is therefore 210 psec.

In a similar fashion, the hold time can be simulated. The $D$ input edge is once again skewed relative to the clock signal till the circuit stop functioning. For this design, the hold time is 0 - i.e., the inputs can be changed on the clock edge. Finally, for the propagation delay, the inputs are transitioned at least a set-up time before the rising edge of the clock and the delay is measured from the 50% point of the CLK edge to the 50% point of the $Q$ output. From this simulation (Figure 7.17), $t_{c-q(lh)}$ was 160 psec and $t_{c-q(hl)}$ was 180 psec.
As mentioned earlier, the drawback of the transmission gate register is the high capacitive load presented to the clock signal. The clock load per register is important since it directly impacts the power dissipation of the clock network. Ignoring the overhead required to invert the clock signal (since the buffer inverter overhead can be amortized over multiple register bits), each register has a clock load of 8 transistors. One approach to reduce the clock load at the cost of robustness is to make the circuit ratioed. Figure 7.18 shows that the feedback transmission gate can be eliminated by directly cross coupling the inverters.

![Figure 7.18 Reduced load clock load static master-slave register.](image)

The penalty for the reduced clock load is increased design complexity. The transmission gate ($T_1$) and its source driver must overpower the feedback inverter ($I_2$) to switch the state of the cross-coupled inverter. The sizing requirements for the transmission gates can be derived using a similar analysis as performed for the SR flip-flop. The input to the inverter $I_1$ must be brought below its switching threshold in order to make a transition. If minimum-sized devices are to be used in the transmission gates, it is essential that the transistors of inverter $I_2$ should be made even weaker. This can be accomplished by making their channel-lengths larger than minimum. Using minimum or close-to-minimum-size devices in the transmission gates is desirable to reduce the power dissipation in the latches and the clock distribution network.

Another problem with this scheme is the reverse conduction — this is, the second stage can affect the state of the first latch. When the slave stage is on (Figure 7.19), it is possible for the combination of $T_2$ and $I_4$ to influence the data stored in $I_1$-$I_2$ latch. As long as $I_4$ is a weak device, this is fortunately not a major problem.

![Figure 7.19 Reverse conduction possible in the transmission gate.](image)

### 7.4.5 Non-ideal clock signals

So far, we have assumed that $\overline{CLK}$ is a perfect inversion of $CLK$, or in other words, that the delay of the generating inverter is zero. Even if this were possible, this would still not be a good assumption. Variations can exist in the wires used to route the two clock signals, or
the load capacitances can vary based on data stored in the connecting latches. This effect, known as clock skew is a major problem, and causes the two clock signals to overlap as is shown in Figure 7.20b. Clock-overlap can cause two types of failures, as illustrated for the NMOS-only negative master-slave register of Figure 7.20a.

- When the clock goes high, the slave stage should stop sampling the master stage output and go into a hold mode. However, since CLK and \( \overline{CLK} \) are both high for a short period of time (the overlap period), both sampling pass transistors conduct and there is a direct path from the \( D \) input to the \( Q \) output. As a result, data at the output can change on the rising edge of the clock, which is undesired for a negative edge-triggered register. This is known as a race condition in which the value of the output \( Q \) is a function of whether the input \( D \) arrives at node \( X \) before or after the falling edge of \( CLK \). If node \( X \) is sampled in the metastable state, the output will switch to a value determined by noise in the system.

- The primary advantage of the multiplexer-based register is that the feedback loop is open during the sampling period, and therefore sizing of devices is not critical to functionality. However, if there is clock overlap between \( CLK \) and \( \overline{CLK} \), node \( A \) can be driven by both \( D \) and \( B \), resulting in an undefined state.

Those problems can be avoided by using two non-overlapping clocks \( PHI_1 \) and \( PHI_2 \) instead (Figure 7.21), and by keeping the nonoverlap time \( t_{non\_overlap} \) between the clocks large enough such that no overlap occurs even in the presence of clock-routing delays. During the nonoverlap time, the FF is in the high-impedance state—the feedback loop is open, the loop gain is zero, and the input is disconnected. Leakage will destroy the state if this condition holds for too long a time. Hence the name pseudostatic: the register employs a combination of static and dynamic storage approaches depending upon the state of the clock.
Problem 7.4 Generating non-overlapping clocks

Figure 7.22 shows one possible implementation of the clock generation circuitry for generating a two-phase non-overlapping clocks. Assuming that each gate has a unit gate delay, derive the timing relationship between the input clock and the two output clocks. What is the non-overlap period? How can this period be increased if needed?

Figure 7.21 Pseudostatic two-phase D register.

7.4.6 Low-Voltage Static Latches

The scaling of supply voltages is critical for low power operation. Unfortunately, certain latch structures don’t function at reduced supply voltages. For example, without the scaling of device thresholds, NMOS only pass transistors (e.g., Figure 7.21) don’t scale well with supply voltage due to its inherent threshold drop. At very low power supply voltages, the input to the inverter cannot be raised above the switching threshold, resulting in incorrect evaluation. Even with the use of transmission gates, performance degrades significantly at reduced supply voltages.

Scaling to low supply voltages hence requires the use of reduced threshold devices. However, this has the negative effect of exponentially increasing the sub-threshold leakage power as discussed in Chapter 6. When the registers are constantly accessed, the leak-
age energy is typically insignificant compared to the switching power. However, with the use of conditional clocks, it is possible that registers are idle for extended periods and the leakage energy expended by registers can be quite significant.

Many solutions are being explored to address the problem of high leakage during idle periods. One approach for this involves the use of Multiple Threshold devices as shown in Figure 7.23 [Mutoh95]. Only the negative latch is shown here. The shaded inverters and transmission gates are implemented in low-threshold devices. The low-threshold inverters are gated using high threshold devices to eliminate leakage.

During normal mode of operation, the sleep devices are tuned on. When clock is low, the $D$ input is sampled and propagates to the output. When clock is high, the latch is in the hold mode. The feedback transmission gate conducts and the cross-coupled feedback is enabled. Note there is an extra inverter, needed for storage of state when the latch is in the sleep state. During idle mode, the high threshold devices in series with the low threshold inverter are turned off (the SLEEP signal is high), eliminating leakage. It is assumed that clock is in the high state when the latch is in the sleep state. The feedback low-threshold transmission gate is turned on and the cross-coupled high-threshold devices maintains the state of the latch.

![Figure 7.23](image)

**Figure 7.23** One solution for the leakage problem in low-voltage operation using MTCMOS.

---

**Problem 7.5 Transistor minimization in the MTCMOS register**

Unlike combination logic, both flavors of high threshold devices in series are required to eliminate the leakage of low threshold gates. Explain why this is the case. Hint: Eliminate the high $V_T$ NMOS or high $V_T$ PMOS of the low threshold inverter on the right of Figure 7.23 and investigate potential leakage paths.
7.5 Dynamic Latches and Registers

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

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

7.5.1 Dynamic Transmission-Gate Based Edge-triggered Registers

A fully dynamic positive edge-triggered register based on the master-slave concept is shown in Figure 7.24. When \( \text{CLK} = 0 \), the input data is sampled on storage node 1, which has an equivalent capacitance of \( C_1 \) consisting of the gate capacitance of \( I_1 \), the junction capacitance of \( T_1 \), and the overlap gate capacitance of \( T_1 \). During this period, the slave stage is in a hold mode, with node 2 in a high-impedance (floating) state. On the rising edge of clock, the transmission gate \( T_2 \) turns on, and the value sampled on node 1 right before the rising edge propagates to the output \( Q \) (note that node 1 is stable during the high phase of the clock since the first transmission gate is turned off). Node 2 now stores the inverted version of node 1. This implementation of an edge-triggered register is very efficient as it requires only 8 transistors. The sampling switches can be implemented using NMOS-only pass transistors, resulting in an even-simpler 6 transistor implementation. The reduced transistor count is attractive for high-performance and low-power systems.

The set-up time of this circuit is simply the delay of the transmission gate, and corresponds to the time it takes node 1 to sample the \( D \) input. The hold time is approximately zero, since the transmission gate is turned off on the clock edge and further inputs changes are ignored. The propagation delay (\( t_{c-q} \)) is equal to two inverter delays plus the delay of the transmission gate \( T_2 \).
One important consideration for such a dynamic register is that the storage nodes (i.e., the state) has to be refreshed at periodic intervals to prevent a loss due to charge leakage, due to diode leakage as well as sub-threshold currents. In datapath circuits, the refresh rate is not an issue since the registers are periodically clocked, and the storage nodes are constantly updated.

Clock overlap is an important concern for this register. Consider the clock waveforms shown in Figure 7.25. During the 0-0 overlap period, the NMOS of $T_1$ and the PMOS of $T_2$ are simultaneously on, creating a direct path for data to flow from the $D$ input of the register to the $Q$ output. This is known as a race condition. The output $Q$ can change on the falling edge if the overlap period is large — obviously an undesirable effect for a positive edge-triggered register. The same is true for the 1-1 overlap region, where an input-output path exists through the PMOS of $T_1$ and the NMOS of $T_2$. The latter case is taken care off by enforcing a hold time constraint. That is, the data must be stable during the high-high overlap period. The former situation (0-0 overlap) can be addressed by making sure that there is enough delay between the $D$ input and node 2 ensuring that new data sampled by the master stage does not propagate through to the slave stage. Generally the built in single inverter delay should be sufficient and the overlap period constraint is given as:

$$t_{\text{overlap}0-0} < t_{T_1} + t_{T_2}$$

(7.4)

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

$$t_{\text{hold}} > t_{\text{overlap}1-1}$$

(7.5)

7.5.2 C2MOS Dynamic Register: A Clock Skew Insensitive Approach

The C2MOS Register

Figure 7.26 shows an ingenious positive edge-triggered register based on the master-slave concept which is insensitive to clock overlap. This circuit is called the C2MOS (Clocked CMOS) register [Suzuki73]. The register operates in two phases.

1. $CLK = 0$ ($CLK = 1$): The first tri-state driver is turned on, and the master stage acts as an inverter sampling the inverted version of $D$ on the internal node $X$. The master stage is in the evaluation mode. Meanwhile, the slave section is in a high-impedance mode, or in a hold mode. Both transistors $M_7$ and $M_8$ are off, decoupling the output from the input. The output $Q$ retains its previous value stored on the output capacitor $C_{L2}$. 

![Figure 7.25 Impact of non-overlapping clocks.](image)
The roles are reversed when $CLK = 1$: The master stage section is in hold mode ($M_3 - M_4$ off), while the second section evaluates ($M_7 - M_8$ on). The value stored on $C_{L1}$ propagates to the output node through the slave stage which acts as an inverter.

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

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

To prove the above statement, we examine both the (0-0) and (1-1) overlap cases (Figure 7.25). In the (0-0) overlap case, the circuit simplifies to the network shown in Figure 7.27a in which both PMOS devices are on during this period. The question is does any new data sampled during the overlap window propagate to the output $Q$. This is not desirable since data should not change on the negative edge for a positive edge-triggered register. Indeed new data is sampled on node $X$ through the series PMOS devices $M_2 - M_4$, and node $X$ can make a 0-to-1 transition during the overlap period. However, this data cannot propagate to the output since the NMOS device $M_7$ is turned off. At the end of the overlap period, $\overline{CLK} = 1$ and both $M_7$ and $M_8$ turn off, putting the slave stage in the hold mode. Therefore, any new data sampled on the falling clock edge is not seen at the slave output $Q$, since the slave state is off till the next rising edge of the clock. As the circuit consists of a cascade of inverters, signal propagation requires one pull-up followed by a pull-down, or vice-versa, which is not feasible in the situation presented.

The (1-1) overlap case (Figure 7.27b), where both NMOS devices $M_1$ and $M_5$ are turned on, is somewhat more contentious. The question is again if new data sampled during the overlap period (right after clock goes high) propagates to the $Q$ output. A positive edge-triggered register may only pass data that is presented at the input before the rising
edge. If the $D$ input changes during the overlap period, node $X$ can make a 1-to-0 transition, but cannot propagate to the output. However, as soon as the overlap period is over, the PMOS $M_8$ is turned on and the 0 propagates to output. This effect is not desirable. The problem is fixed by imposing a hold time constraint on the input data, $D$, or, in other words, the data $D$ should be stable during the overlap period.

In summary, it can be stated that the C$^2$MOS latch is insensitive to clock overlaps because those overlaps activate either the pull-up or the pull-down networks of the latches, but never both of them simultaneously. If the rise and fall times of the clock are sufficiently slow, however, there exists a time slot where both the NMOS and PMOS transistors are conducting. This creates a path between input and output that can destroy the state of the circuit. Simulations have shown that the circuit operates correctly as long as the clock rise time (or fall time) is smaller than approximately five times the propagation delay of the register. This criterion is not too stringent, and is easily met in practical designs. The impact of the rise and fall times is illustrated in Figure 7.28, which plots the simulated transient response of a C$^2$MOS $D$ FF for clock slopes of respectively 0.1 and 3 nsec. For slow clocks, the potential for a race condition exists.
Section 7.5 Dynamic Latches and Registers

Dual-edge Triggered Registers

So far, we have focused on edge-triggered registers that sample the input data on only one of the clock edges (rising or falling). It is also possible to design sequential circuits that sample the input on both edges. The advantage of this scheme is that a lower frequency clock (half of the original rate) is distributed for the same functional throughput, resulting in power savings in the clock distribution network. Figure 7.29 shows a modification of the C2MOS register to enable sampling on both edges [REFERENCE]. It consists of two parallel master-slave based edge-triggered registers, whose outputs are multiplexed using the tri-state drivers.

When clock is high, the positive latch composed of transistors $M_1$-$M_4$ is sampling the inverted $D$ input on node $X$. Node $Y$ is held stable, since devices $M_9$ and $M_{10}$ are turned off. On the falling edge of the clock, the top slave latch $M_5$-$M_8$ turns on, and drives the inverted value of $X$ to the $Q$ output. During the low phase, the bottom master latch ($M_1$, $M_4$, $M_9$, $M_{10}$) is turned on, sampling the inverted $D$ input on node $Y$. Note that the devices $M_1$ and $M_4$ are reused, reducing the load on the $D$ input. On the rising edge, the bottom slave latch conducts, and drives the inverted version of $Y$ on node $Q$. Data hence changes on both edges. Note that the slave latches operate in a complementary fashion — this is, only one of them is turned on during each phase of the clock.

![Figure 7.29 C2MOS based dual-edge triggered register.](image_url)

Problem 7.6 Dual-edge Registers

Determine how the adoption of dual-edge registers influences the power-dissipation in the clock-distribution network.
7.5.3 True Single-Phase Clocked Register (TSPCR)

In the two-phase clocking schemes described above, care must be taken in routing the two clock signals to ensure that overlap is minimized. While the C²MOS provides a skew-tolerant solution, it is possible to design registers that only use a single phase clock. The True Single-Phase Clocked Register (TSPCR) proposed by Yuan and Svensson uses a single clock (without an inverse clock) [Yuan89]. The basic single-phase positive and negative latches are shown in Figure 7.30. For the positive latch, when \( CLK \) is high, the latch is in the transparent mode and corresponds to two cascaded inverters; the latch is non-inverting, and propagates the input to the output. On the other hand, when \( CLK = 0 \), both inverters are disabled, and the latch is in hold-mode. Only the pull-up networks are still active, while the pull-down circuits are deactivated. As a result of the dual-stage approach, no signal can ever propagate from the input of the latch to the output in this mode. A register can be constructed by cascading positive and negative latches. The clock load is similar to a conventional transmission gate register, or C²MOS register. The main advantage is the use of a single clock phase. The disadvantage is the slight increase in the number of transistors — 12 transistors are required.

TSPC offers an additional advantage: the possibility of embedding logic functionality into the latches. This reduces the delay overhead associated with the latches. Figure 7.31a outlines the basic approach for embedding logic, while Figure 7.31b shows an example of a positive latch that implements the AND of \( \text{In}_1 \) and \( \text{In}_2 \) in addition to performing the latching function. While the set-up time of this latch has increased over the one shown in Figure 7.30, the overall performance of the digital circuit (that is, the clock period of a sequential circuit) has improved: the increase in set-up time is typically smaller than the delay of an AND gate. This approach of embedding logic into latches has been used extensively in the design of the EV4 DEC Alpha microprocessor [Dobberpuhl92] and many other high performance processors.

**Example 7.4 Impact of embedding logic into latches on performance**

Consider embedding an AND gate into the TSPC latch, as shown in Figure 7.31b. In a 0.25 \( \mu \)m, the set-up time of such a circuit using minimum-size devices is 140 psec. A conventional approach, composed of an AND gate followed by a positive latch has an effective set-up time of 600 psec (we treat the AND plus latch as a black box that performs the AND-latching...
functions). The embedded logic approach hence results in significant performance improvements.

The TSPC latch circuits can be further reduced in complexity as illustrated in Figure 7.32, where only the first inverter is controlled by the clock. Besides the reduced number of transistors, these circuits have the advantage that the clock load is reduced by half. On the other hand, not all node voltages in the latch experience the full logic swing. For instance, the voltage at node \( A \) (for \( V_{in} = 0 \) V) for the positive latch maximally equals \( V_{DD} - V_{Tn} \), which results in a reduced drive for the output NMOS transistor and a loss in performance. Similarly, the voltage on node \( A \) (for \( V_{in} = V_{DD} \)) for the negative latch is only driven down to \( |V_{Tp}| \). This also limits the amount of \( V_{DD} \) scaling possible on the latch.

Figure 7.33 shows the design of a specialized single-phase edge-triggered register. When \( CLK = 0 \), the input inverter is sampling the inverted \( D \) input on node \( X \). The second (dynamic) inverter is in the precharge mode, with \( M_6 \) charging up node \( Y \) to \( V_{DD} \). The third inverter is in the hold mode, since \( M_4 \) and \( M_5 \) are off. Therefore, during the low phase of the clock, the input to the final (static) inverter is holding its previous value and the output \( Q \) is stable. On the rising edge of the clock, the dynamic inverter \( M_4-M_6 \) evaluates. If \( X \) is high on the rising edge, node \( Y \) discharges. The third inverter \( M_7-M_9 \) is on during the high phase, and the node value on \( Y \) is passed to the output \( Q \). On the positive phase of the clock, note that node \( X \) transitions to a low if the \( D \) input transitions to a high level. Therefore, the input must be kept stable till the value on node \( X \) before the rising edge of the clock propagates to \( Y \). This represents the hold time of the register (note that the hold time less than 1 inverter delay since it takes 1 delay for the input to affect node \( X \)).


gation delay of the register is essentially three inverters since the value on node \( X \) must propagate to the output \( Q \). Finally, the set-up time is the time for node \( X \) to be valid, which is one inverter delay.

![Diagram of TSPC Circuit]

**Figure 7.33** Positive edge-triggered register TSPC.

**WARNING:** Similar to the \( \text{C}^2\text{MOS} \) latch, the TSPC latch malfunctions when the slope of the clock is not sufficiently steep. Slow clocks cause both the NMOS and PMOS clocked transistors to be on simultaneously, resulting in undefined values of the states and race conditions. The clock slopes should therefore be carefully controlled. If necessary, local buffers must be introduced to ensure the quality of the clock signals.

**Example 7.5 TSPC Edge-Triggered Register**

Transistor sizing is critical for achieving correct functionality in the TSPC register. With improper sizing, glitches may occur at the output due to a race condition when the clock transitions from low to high. Consider the case where \( D = \text{low} \) and \( \overline{Q} = 1 \) (\( Q = 0 \)). While CLK is low, \( Y \) is pre-charged high turning on \( M_7 \). When CLK transitions from low to high, nodes \( Y \) and \( Q \) start to discharge simultaneously (through \( M_4 \)-\( M_5 \) and \( M_7 \)-\( M_8 \), respectively). Once \( Y \) is sufficiently low, the trend on \( Q \) is reversed and the node is pulled high anew through \( M_9 \). In a sense, this chain of events is comparable to what would happen if we chain dynamic logic gates. Figure 7.34 shows the transient response of the circuit of Figure 7.34 for different sizes of devices in the final two stages.

![Graph showing transient response]

**Figure 7.34** Transistor sizing issues in TSPC (for the register of Figure 7.33).

This glitch may be the cause of fatal errors, as it may create unwanted events (for instance, when the output of the latch is used as a clock signal input to another register). It also reduces the contamination delay of the register. The problem can be corrected by resizing the relative strengths of the pull-down paths through \( M_4 \)-\( M_5 \) and \( M_7 \)-\( M_8 \), so that \( Y \) discharges
Section 7.6 Pulse Registers

much faster than $Q$. This is accomplished by reducing the strength of the $M_7$-$M_8$ pulldown path, and by speeding up the $M_4$-$M_5$ pulldown path.

7.6 Pulse Registers

Until now, we have used the master-slave configuration to create an edge-triggered register. A fundamentally different approach for constructing a register uses pulse signals. The idea is to construct a short pulse around the rising (or falling) edge of the clock. This pulse acts as the clock input to a latch (e.g., a TSPC flavor is shown in Figure 7.35a), sampling the input only in a short window. Race conditions are thus avoided by keeping the opening time (i.e., the transparent period) of the latch very short. The combination of the glitch-generation circuitry and the latch results in a positive edge-triggered register.

Figure 7.35b shows an example circuit for constructing a short intentional glitch on each rising edge of the clock [Kozo96]. When $CLK = 0$, node $X$ is charged up to $V_{DD}$ ($M_N$ is off since $CLKG$ is low). On the rising edge of the clock, there is a short period of time when both inputs of the AND gate are high, causing $CLKG$ to go high. This in turn activates $M_N$, pulling $X$ and eventually $CLKG$ low (Figure 7.35c). The length of the pulse is controlled by the delay of the AND gate and the two inverters. Note that there exists also a delay between the rising edges of the input clock ($CLK$) and the glitch clock ($CLKG$) — also equal to the delay of the AND gate and the two inverters. If every register on the chip uses the same clock generation mechanism, this sampling delay does not matter. However, process variations and load variations may cause the delays through the glitch clock circuitry to be different. This must be taken into account when performing timing verification and clock skew analysis (which is the topic of a later Chapter).

![Glitch latch - timing generation and register.](image-url)
If set-up time and hold time are measured in reference to the rising edge of the glitch clock, the set-up time is essentially zero, the hold time is equal to the length of the pulse (if the contamination delay is zero for the gates), and the propagation delay \( t_{c-q} \) equals two gate delays. The advantage of the approach is the reduced clock load and the small number of transistors required. The glitch-generation circuitry can be amortized over multiple register bits. The disadvantage is a substantial increase in verification complexity. This has prevented a wide-spread use. They do however provide an alternate approach to conventional schemes, and have been adopted in some high performance processors (e.g., [Kozo96]).

Another version of the pulsed register is shown in Figure 7.36 (as used in the AMD-K6 processor [Partovi96]). When the clock is low, \( M_3 \) and \( M_6 \) are off and device \( P_1 \) is turned on. Node \( X \) is precharged to \( V_{DD} \), the output node \( (Q) \) is decoupled from \( X \) and is held at its previous state. \( CLKD \) is a delay-inverted version of \( CLK \). On the rising edge of the clock, \( M_3 \) and \( M_6 \) turn on while devices \( M_4 \) and \( M_5 \) stay on for a short period, determined by the delay of the three inverters. During this interval, the circuit is transparent and the input data \( D \) is sampled by the latch. Once \( CLKD \) goes low, node \( X \) is decoupled from the \( D \) input and is either held or starts to precharge to \( V_{DD} \) by PMOS device \( P_2 \). On the falling edge of the clock, node \( X \) is held at \( V_{DD} \) and the output is held stable by the cross-coupled inverters.

Note that this circuit also uses a one-shot, but the one-shot is integrated into the register. The transparency period also determines the hold time of the register. The window must be wide enough for the input data to propagate to the \( Q \) output. In this particular circuit, the set-up time can be negative. This is the case if the transparency window is longer than the delay from input to output. This is attractive, as data can arrive at the register even after the clock goes high, which means that time is borrowed from the previous cycle.

**Example 7.6 Set-up time of glitch register**

The glitch register of Figure 7.36 is transparent during the (1-1) overlap of \( CLK \) and \( CLKD \). As a result, the input data can actually change after the rising edge of the clock, resulting in a negative set-up time (Figure 7.37). The \( D \)-input transitions to low after the rising edge of the clock, and transitions high before the falling edge of \( CLKD \) (this is, during the transparency period). Observe how the output follows the input. The output \( Q \) does go to the correct value of \( V_{DD} \) as long as the input \( D \) is set up correctly some time before the falling edge of \( CLKD \). When the negative set-up time is exploited, there can be no guarantees on the monotonic behavior of the output. That is, the output can have multiple transitions around the rising edge, and therefore, the output of the register should not be used as a clock to other registers.
Section 7.7 Sense-Amplifier Based Registers

7.7 Sense-Amplifier Based Registers

So far, we have presented two fundamental approaches towards building edge-triggered registers: the master-slave concept and the glitch technique. Figure 7.38 introduces another technique that uses a sense amplifier structure to implement an edge-triggered register [Montanaro96]. Sense amplifier circuits accept small input signals and amplify them to generate rail-to-rail swings. As we will see, sense amplifier circuits are used extensively in memory cores and in low swing bus drivers to amplify small voltage swings.
present in heavily loaded wires. There are many techniques to construct these amplifiers, with the use of feedback (e.g., cross-coupled inverters) being one common approach. The circuit shown in Figure 7.38 uses a precharged front-end amplifier that samples the differential input signal on the rising edge of the clock signal. The outputs of front-end are fed into a NAND cross-coupled SR FF that holds the data and guarantees that the differential outputs switch only once per clock cycle. The differential inputs in this implementation don’t have to have rail-to-rail swing and hence this register can be used as a receiver for a reduced swing differential bus.

The core of the front-end consists of a cross-coupled inverter ($M_5$- $M_8$) whose outputs ($L_1$ and $L_2$) are precharged using devices $M_9$ and $M_{10}$ during the low phase of the clock. As a result, PMOS transistors $M_7$ and $M_8$ to be turned off and the NAND FF is holding its previous state. Transistor $M_1$ is similar to an evaluate switch in dynamic circuits and is turned off ensuring that the differential inputs don’t affect the output during the low phase of the clock. On the rising edge of the clock, the evaluate transistor turns on and the differential input pair ($M_2$ and $M_3$) is enabled, and the difference between the input signals is amplified on the output nodes on $L_1$ and $L_2$. The cross-coupled inverter pair flips to one of its the stable states based on the value of the inputs. For example, if $IN$ is 1, $L_1$ is pulled to 0, and $L_2$ remains at $V_{DD}$. Due to the amplifying properties of the input stage, it is not necessary for the input to swing all the way up to $V_{DD}$ and enables the use of low-swing signaling on the input wires.

The shorting transistor, $M_4$, is used to provide a DC leakage path from either node $L_3$, or $L_4$, to ground. This is necessary to accommodate the case where the inputs change their value after the positive edge of CLK has occurred, resulting in either $L_3$ or $L_4$ being left in a high-impedance state with a logical low voltage level stored on the node. Without the leakage path that node would be susceptible to charging by leakage currents. The latch
could then actually change state prior to the next rising edge of \( CLK \)! This is best illustrated graphically, as shown in Figure 7.39.

### 7.8 Pipelining: An approach to optimize sequential circuits

Pipelining is a popular design technique often used to accelerate the operation of the data-paths in digital processors. The idea is easily explained with the example of Figure 7.40a. The goal of the presented circuit is to compute \( \log(|a - b|) \), where both \( a \) and \( b \) represent streams of numbers, that is, the computation must be performed on a large set of input values. The minimal clock period \( T_{\text{min}} \) necessary to ensure correct evaluation is given as:

\[
T_{\text{min}} = t_{c-q} + t_{p,d,\text{logic}} + t_{su}
\]

where \( t_{c-q} \) and \( t_{su} \) are the propagation delay and the set-up time of the register, respectively. We assume that the registers are edge-triggered D registers. The term \( t_{p,d,\text{logic}} \) stands for the worst-case delay path through the combinatorial network, which consists of the adder, absolute value, and logarithm functions. In conventional systems (that don’t push the edge of technology), the latter delay is generally much larger than the delays associated with the registers and dominates the circuit performance. Assume that each logic module has an equal propagation delay. We note that each logic module is then active for only 1/3 of the clock period (if the delay of the register is ignored). For example, the adder unit is active during the first third of the period and remains idle—this is, it does no useful computation— during the other 2/3 of the period. Pipelining is a technique to improve the resource utilization, and increase the functional throughput. Assume that we introduce registers between the logic blocks, as shown in Figure 7.40b. This causes the computation for one set of input data to spread over a number of clock periods, as shown in Table 7.1. The

![Figure 7.40 Datapath for the computation of \( \log(|a + b|) \).](image-url)
result for the data set \((a_1, b_1)\) only appears at the output after three clock-periods. At that time, the circuit has already performed parts of the computations for the next data sets, \((a_2, b_2)\) and \((a_3, b_3)\). The computation is performed in an assembly-line fashion, hence the name pipeline.

### Table 7.1 Example of pipelined computations.

<table>
<thead>
<tr>
<th>Clock Period</th>
<th>Adder</th>
<th>Absolute Value</th>
<th>Logarithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(a_1 + b_1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(a_2 + b_2)</td>
<td>(</td>
<td>a_1 + b_1</td>
</tr>
<tr>
<td>3</td>
<td>(a_3 + b_3)</td>
<td>(</td>
<td>a_2 + b_2</td>
</tr>
<tr>
<td>4</td>
<td>(a_4 + b_4)</td>
<td>(</td>
<td>a_3 + b_3</td>
</tr>
<tr>
<td>5</td>
<td>(a_5 + b_5)</td>
<td>(</td>
<td>a_4 + b_4</td>
</tr>
</tbody>
</table>

The advantage of pipelined operation becomes apparent when examining the minimum clock period of the modified circuit. The combinational circuit block has been partitioned into three sections, each of which has a smaller propagation delay than the original function. This effectively reduces the value of the minimum allowable clock period:

\[
T_{\text{min,pipe}} = \frac{t_{c-q} + \max(t_{pd,\text{add}}, t_{pd,\text{abs}}, t_{pd,\log})}{3} \quad (7.7)
\]

Suppose that all logic blocks have approximately the same propagation delay, and that the register overhead is small with respect to the logic delays. The pipelined network outperforms the original circuit by a factor of three under these assumptions, or \(T_{\text{min,pipe}} = \frac{T_{\text{min}}}{3}\). The increased performance comes at the relatively small cost of two additional registers, and an increased latency. This explains why pipelining is popular in the implementation of very high-performance datapaths.

### 7.8.1 Latch- vs. Register-Based Pipelines

Pipelined circuits can be constructed using level-sensitive latches instead of edge-triggered registers. Consider the pipelined circuit of Figure 7.41. The pipeline system is implemented based on pass-transistor-based positive and negative latches instead of edge-triggered registers. That is, logic is introduced between the master and slave latches of a master-slave system. In the following discussion, we use without loss of generality the \(CLK-\overline{CLK}\) notation to denote a two-phase clock system. Latch-based systems give significantly more flexibility in implementing a pipelined system, and often offers higher performance. When the clocks \(CLK\) and \(\overline{CLK}\) are nonoverlapping, correct pipeline operation is obtained. Input data is sampled on \(C_1\) at the negative edge of \(CLK\) and the computation of logic block \(F\) starts; the result of the logic block \(F\) is stored on \(C_2\) on the falling edge of \(\overline{CLK}\), and the computation of logic block \(G\) starts. The nonoverlapping of the clocks

---

1 Latency is defined here as the number of clock cycles it takes for the data to propagate from the input to the output. For the example at hand, pipelining increases the latency from 1 to 3. An increased latency is in general acceptable, but can cause a global performance degradation if not treated with care.
Section 7.8 Pipelining: An approach to optimize sequential circuits

ensures correct operation. The value stored on $C_2$ at the end of the $CLK$ low phase is the result of passing the previous input (stored on the falling edge of $CLK$ on $C_1$) through the logic function $F$. When overlap exists between $CLK$ and $CLK$, the next input is already being applied to $F$, and its effect might propagate to $C_2$ before $CLK$ goes low (assuming that the contamination delay of $F$ is small). In other words, a race develops between the previous input and the current one. Which value wins depends upon the logic function $F$, the overlap time, and the value of the inputs since the propagation delay is often a function of the applied inputs. The latter factor makes the detection and elimination of race conditions non-trivial.

### 7.8.2 NORA-CMOS—A Logic Style for Pipelined Structures

The latch-based pipeline circuit can also be implemented using $C^2$MOS latches, as shown in Figure 7.42. The operation is similar to the one discussed above. This topology has one additional, important property:

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

The reasoning for the above argument is similar to the argument made in the construction of a $C^2$MOS register. During a (0-0) overlap between $CLK$ and $CLK$, all $C^2$MOS latches, simplify to pure pull-up networks (see Figure 7.27). The only way a signal can race from stage to stage under this condition is when the logic function $F$ is inverting, as illustrated in Figure 7.43, where $F$ is replaced by a single, static CMOS inverter. Similar considerations are valid for the (1-1) overlap.

Based on this concept, a logic circuit style called NORA-CMOS was conceived [Goncalves83]. It combines $C^2$MOS pipeline registers and NORA dynamic logic function blocks. Each module consists of a block of combinational logic that can be a mixture of static and dynamic logic, followed by a $C^2$MOS latch. Logic and latch are clocked in such a way that both are simultaneously in either evaluation, or hold (precharge) mode. A block
that is in evaluation during $CLK = 1$ is called a $CLK$-module, while the inverse is called a $\overline{CLK}$-module. Examples of both classes are shown in Figure 7.44 a and b, respectively. The operation modes of the modules are summarized in Table 7.2.

### Table 7.2 Operation modes for NORA logic modules.

<table>
<thead>
<tr>
<th>$CLK$ block</th>
<th>$CLK$ block</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CLK = 0$</td>
<td>$CLK = 1$</td>
</tr>
<tr>
<td>Logic</td>
<td>Evaluate</td>
</tr>
<tr>
<td>Latch</td>
<td>Evaluate</td>
</tr>
</tbody>
</table>

A NORA datapath consists of a chain of alternating $CLK$ and $\overline{CLK}$ modules. While one class of modules is precharging with its output latch in hold mode, preserving the previous output value, the other class is evaluating. Data is passed in a pipelined fashion from module to module.

NORA offers designers a wide range of design choices. Dynamic and static logic can be mixed freely, and both $CLK_p$ and $CLK_n$ dynamic blocks can be used in cascaded or in pipelined form. With this freedom of design, extra inverter stages, as required in DOMINO-CMOS, are most often avoided.

### Design Rules

In order to ensure correct operation, two important rules should always be followed:
Section 7.8 Pipelining: An approach to optimize sequential circuits

- The dynamic-logic rule: Inputs to a dynamic $CLK_n$ block are only allowed to make a single $0 \rightarrow 1$ ($1 \rightarrow 0$) transition during the evaluation period (Chapter 6).

- The $C^2$MOS rule: In order to avoid races, the number of static inversions between $C^2$MOS latches should be even.

The presence of dynamic logic circuits requires the introduction of some extensions to the latter rule. Consider the situation pictured in Figure 7.45a. During precharge ($CLK = 0$), the output register of the module has to be in hold mode, isolating the output node from the internal events in the module. Assume now that a (0-0) overlap occurs. Node $A$ gets precharged to $V_{DD}$, while the latch simplifies to a pull-up network (Figure 7.45b). It can be observed that under those circumstances the output node charges to $V_{DD}$, and the stored value is erased! This malfunctioning is caused by the fact that the number of static inversions between the last dynamic node in the module and the latch is odd, which creates an active path between the precharged node and the output. This translates into the following rule: The number of static inversions between the last dynamic block in a logic function and the $C^2$MOS latch should be even. This and similar considerations lead to a reformulated $C^2$MOS rule [Goncalvez83].
Chapter 7

### Revised $C^2$MOS Rule

- The number of static inversions between $C^2$MOS latches should be even (in the absence of dynamic nodes); if dynamic nodes are present, the number of static inverters between a latch and a dynamic gate in the logic block should be even. The number of static inversions between the last dynamic gate in a logic block and the latch should be even as well.

Adhering to the above rules is not always trivial and requires a careful analysis of the logic equations to be implemented. This often makes the design of an operational NOR-A-CMOS structure cumbersome. Its use should only be considered when maximum circuit performance is a must.

#### 7.9 Non-Bistable Sequential Circuits

In the preceding sections, we have focused on one single type of sequential element, this is the latch (and its sibling the register). The most important property of such a circuit is that it has two stable states, and is hence called bistable. The bistable element is not the only sequential circuit of interest. Other regenerative circuits can be catalogued as astable and monostable. The former act as oscillators and can, for instance, be used for on-chip clock generation. The latter serve as pulse generators, also called one-shot circuits. Another interesting regenerative circuit is the Schmitt trigger. This component has the useful property of showing hysteresis in its dc characteristics—it's switching threshold is variable and depends upon the direction of the transition (low-to-high or high-to-low). This peculiar feature can come in handy in noisy environments.
Section 7.9 Non-Bistable Sequential Circuits

7.9.1 The Schmitt Trigger

Definition

A Schmitt trigger [Schmitt38] is a device with two important properties:

1. It responds to a slowly changing input waveform with a fast transition time at the output.

2. The voltage-transfer characteristic of the device displays different switching thresholds for positive- and negative-going input signals. This is demonstrated in Figure 7.46, where a typical voltage-transfer characteristic of the Schmitt trigger is shown (and its schematics symbol). The switching thresholds for the low-to-high and high-to-low transitions are called $V_{M+}$ and $V_{M-}$, respectively. The hysteresis voltage is defined as the difference between the two.

![Figure 7.46](image) Non-inverting Schmitt trigger.

One of the main uses of the Schmitt trigger is to turn a noisy or slowly varying input signal into a clean digital output signal. This is illustrated in Figure 7.47. Notice how the hysteresis suppresses the ringing on the signal. At the same time, the fast low-to-high (and high-to-low) transitions of the output signal should be observed. For instance, steep signal slopes are beneficial in reducing power consumption by suppressing direct-path currents. The “secret” behind the Schmitt trigger concept is the use of positive feedback.

![Figure 7.47](image) Noise suppression using a Schmitt trigger.
CMOS Implementation

One possible CMOS implementation of the Schmitt trigger is shown in Figure 7.48. The idea behind this circuit is that the switching threshold of a CMOS inverter is determined by the \((k_n/k_p)\) ratio between the NMOS and PMOS transistors. Increasing the ratio results in a reduction of the threshold, while decreasing it results in an increase in \(V_M\). Adapting the ratio depending upon the direction of the transition results in a shift in the switching threshold and a hysteresis effect. This adaptation is achieved with the aid of feedback.

Suppose that \(V_{in}\) is initially equal to 0, so that \(V_{out} = 0\) as well. The feedback loop biases the PMOS transistor \(M_4\) in the conductive mode while \(M_3\) is off. The input signal effectively connects to an inverter consisting of two PMOS transistors in parallel (\(M_2\) and \(M_4\)) as a pull-up network, and a single NMOS transistor (\(M_1\)) in the pull-down chain. This modifies the effective transistor ratio of the inverter to \(k_M/(k_M + k_{M4})\), which moves the switching threshold upwards.

Once the inverter switches, the feedback loop turns off \(M_4\), and the NMOS device \(M_3\) is activated. This extra pull-down device speeds up the transition and produces a clean output signal with steep slopes.

A similar behavior can be observed for the high-to-low transition. In this case, the pull-down network originally consists of \(M_1\) and \(M_3\) in parallel, while the pull-up network is formed by \(M_2\). This reduces the value of the switching threshold to \(V_M-\).

**Example 7.7 CMOS Schmitt Trigger**

Consider the schmitt trigger with the following device sizes. Devices \(M_1\) and \(M_2\) are \(1\mu m/0.25\mu m\) and \(3\mu m/0.25\mu m\), respectively. The inverter is sized such that the switching threshold is around \(V_{DD}/2\) (= 1.25 V). Figure 7.49a shows the simulation of the Schmitt trigger assuming that devices \(M_1\) and \(M_2\) are \(0.5\mu m/0.25\mu m\) and \(1.5\mu m/0.25\mu m\), respectively. As apparent from the plot, the circuit exhibits hysteresis. The high-to-low switching point \((V_{M^-} = 0.9 V)\) is lower than \(V_{DD}/2\), while the low-to-high switching threshold \((V_{M^+} = 1.6 V)\) is larger than \(V_{DD}/2\).

It is possible to shift the switching point by changing the sizes of \(M_1\) and \(M_4\). For example, to modify the low-to-high transition, we need to vary the PMOS device. The high-to-low threshold is kept constant by keeping the device width of \(M_4\) at 0.5 \(\mu m\). The device width of \(M_4\) is varied as \(k \times 0.5\mu m\). Figure 7.49b demonstrates how the switching threshold increases with raising values of \(k\).
Problem 7.8 An Alternative CMOS Schmitt Trigger

Another CMOS Schmitt trigger is shown in Figure 7.50. Discuss the operation of the gate, and derive expressions for $V_{M-}$ and $V_{M+}$.

7.9.2 Monostable Sequential Circuits

A monostable element is a circuit that generates a pulse of a predetermined width every time the quiescent circuit is triggered by a pulse or transition event. It is called monostable because it has only one stable state (the quiescent one). A trigger event, which is either a signal transition or a pulse, causes the circuit to go temporarily into another quasi-stable state. This means that it eventually returns to its original state after a time period determined by the circuit parameters. This circuit, also called a one-shot, is useful in generating pulses of a known length. This functionality is required in a wide range of applications. We have already seen the use of a one-shot in the construction of glitch registers. Another
notorious example is the address transition detection (ATD) circuit, used for the timing generation in static memories. This circuit detects a change in a signal, or group of signals, such as the address or data bus, and produces a pulse to initialize the subsequent circuitry.

The most common approach to the implementation of one-shots is the use of a simple delay element to control the duration of the pulse. The concept is illustrated in Figure 7.51. In the quiescent state, both inputs to the XOR are identical, and the output is low. A transition on the input causes the XOR inputs to differ temporarily and the output to go high. After a delay $t_d$ (of the delay element), this disruption is removed, and the output goes low again. A pulse of length $t_d$ is created. The delay circuit can be realized in many different ways, such as an RC-network or a chain of basic gates.

![Figure 7.51 Transition-triggered one-shot.](image)

### 7.9.3 Astable Circuits

An astable circuit has no stable states. The output oscillates back and forth between two quasi-stable states with a period determined by the circuit topology and parameters (delay, power supply, etc.). One of the main applications of oscillators is the on-chip generation of clock signals. This application is discussed in detail in a later chapter (on timing).

The ring oscillator is a simple, example of an astable circuit. It consists of an odd number of inverters connected in a circular chain. Due to the odd number of inversions, no stable operation point exists, and the circuit oscillates with a period equal to $2 \times t_p \times N$, with $N$ the number of inverters in the chain and $t_p$ the propagation delay of each inverter.

**Example 7.8 Ring oscillator**

The simulated response of a ring oscillator with five stages is shown in Figure 7.52 (all gates use minimum-size devices). The observed oscillation period approximately equals 0.5 nsec, which corresponds to a gate propagation delay of 50 psec. By tapping the chain at various points, different phases of the oscillating waveform are obtained (phases 1, 3, and 5 are dis-

![Figure 7.52 Simulated waveforms of five-stage ring oscillator. The outputs of stages 1, 3, and 5 are shown.](image)
Section 7.9 Non-Bistable Sequential Circuits

A wide range of clock signals with different duty-cycles and phases can be derived from those elementary signals using simple logic operations.

The ring oscillator composed of cascaded inverters produces a waveform with a fixed oscillating frequency determined by the delay of an inverter in the CMOS process. In many applications, it is necessary to control the frequency of the oscillator. An example of such a circuit is the voltage-controlled oscillator (VCO), whose oscillation frequency is a function (typically non-linear) of a control voltage. The standard ring oscillator can be modified into a VCO by replacing the standard inverter with a current-starved inverter as shown in Figure 7.53 [Jeong87]. The mechanism for controlling the delay of each inverter is to limit the current available to discharge the load capacitance of the gate.

![Figure 7.53 Voltage-controlled oscillator based on current-starved inverters.](image)

In this modified inverter circuit, the maximal discharge current of the inverter is limited by adding an extra series device. Note that the low-to-high transition on the inverter can also be controlled by adding a PMOS device in series with $M_2$. The added NMOS transistor $M_3$, is controlled by an analog control voltage $V_{cntl}$ which determines the available discharge current. Lowering $V_{cntl}$ reduces the discharge current and, hence, increases $t_{pHL}$. The ability to alter the propagation delay per stage allows us to control the frequency of the ring structure. The control voltage is generally set using feedback techniques. Under low operating current levels, the current-starved inverter suffers from slow fall times at its output. This can result in significant short-circuit current. This is resolved by feeding its output into a CMOS inverter or better yet a Schmitt trigger. An extra inverter is needed at the end to ensure that the structure oscillates.

**Example 7.9 Current-Starved Inverter Simulation**

Figure 7.54 show the simulated delay of the current-starved inverter as a function of the control voltage $V_{cntl}$. The delay of the inverter can be varied over a large range. When the control voltage is smaller than the threshold, the device enters the sub-threshold region. This results in large variations of the propagation delay, as the drive current is exponentially dependent on the drive voltage. When operating in this region, the delay is very sensitive to variations in the control voltage, and, hence, to noise.
Another approach to implement the delay element is to use a differential element as shown in Figure 7.55a. Since the delay cell provides both inverting and non-inverting outputs, an oscillator with an even number of stages can be implemented. Figure 7.55b shows a two-stage differential VCO, where the feedback loop provides 180° phase shift through two gate delays, one non-inverting and the other inverting, therefore forming an oscillation. The simulated waveforms of this two stage VCO are shown in Figure 7.55c. The in-phase and quadrature phase outputs are available simultaneously. The differential type VCO has better immunity to common mode noise (for example, supply noise) compared to the common ring oscillator. However, it consumes more power due to the increased complexity, and the static current.

![Figure 7.54](image)

Figure 7.54 $t_{pdL}$ of current-starved inverter as a function of the control voltage.

![Figure 7.55](image)

Figure 7.55 Differential delay element and VCO topology.
Section 7.10 Perspective: Choosing a Clocking Strategy

7.10 Perspective: Choosing a Clocking Strategy

A crucial decision that must be made in the earliest phases of a chip design is to select the appropriate clocking methodology. The reliable synchronization of the various operations occurring in a complex circuit is one of the most intriguing challenges facing the digital designer of the next decade. Choosing the right clocking scheme affects the functionality, speed and power of a circuit.

A number of widely-used clocking schemes were introduced in this chapter. The most robust and conceptually simple scheme is the two-phase master-slave design. The predominant approach is to use the multiplexer-based register, and to generate the two clock phases locally by simply inverting the clock. More exotic schemes such as the glitch register are also used in practice. However, these schemes require significant hand tuning and must only be used in specific situations. An example of such is the need for a negative set-up time to cope with clock skew.

The general trend in high-performance CMOS VLSI design is therefore to use simple clocking schemes, even at the expense of performance. Most automated design methodologies such as standard cell employ a single-phase, edge-triggered approach, based on static flip-flops. But the tendency towards simpler clocking approaches is also apparent in high-performance designs such as microprocessors. The use of latches between logic is also very common to improve circuit performance.

7.11 Summary

This chapter has explored the subject of sequential digital circuits. The following topics were discussed:

- The cross-coupling of two inverters creates a bistable circuit, called a flip-flop. A third potential operation point turns out to be metastable; that is, any diversion from this bias point causes the flip-flop to converge to one of the stable states.
- A latch is a level-sensitive memory element that samples data on one phase and holds data on the other phase. A register (sometimes also called a flip-flop) on the other hand samples the data on the rising or falling edge. A register has three important parameters: the set-up time, the hold time, and the propagation delay. These parameters must be carefully optimized since they may account for a significant portion of the clock period.
- Registers can be static or dynamic. A static register holds state as long as the power supply is turned on. It is ideal for memory that is accessed infrequently (e.g., reconfiguration registers or control information). Dynamic memory is based on temporary charge store on capacitors. The primary advantage is the reduced complexity and higher performance/lower power. However, charge on a dynamic node leaks away with time, and hence dynamic circuits have a minimum clock frequency.
- There are several fundamentally different approaches towards building a register. The most common and widely used approach is the master-slave configuration which involves cascading a positive latch and negative latch (or vice-versa).
• Registers can also be constructed using the pulse or glitch concept. An intentional pulse (using a one shot circuit) is used to sample the input around an edge. Generally, the design of such circuits requires careful timing analysis across all process corners. Sense-amplifier based schemes are also used to construct registers and are to be used when high performance or low signal swing signalling is required.

• Choice of clocking style is an important consideration. Two phase design can result in race problems. Circuit techniques such as C2MOS can be used to eliminate race conditions in two-phase clocking. Another option is to use true single phase clocking. However, the rise time of clocks must be carefully optimized to eliminate races.

• The combination of dynamic logic with dynamic latches can produce extremely fast computational structures. An example of such an approach, the NORA logic style, is very effective in pipelined datapaths.

• Monostable structures have only one stable state. They are useful as pulse generators.

• Astable multivibrators, or oscillators, possess no stable state. The ring oscillator is the best-known example of a circuit of this class.

• Schmitt triggers display hysteresis in their dc characteristic and fast transitions in their transient response. They are mainly used to suppress noise.

7.12 To Probe Further

The basic concepts of sequential gates can be found in many logic design textbooks (e.g., [Mano82] and [Hill74]). The design of sequential circuits is amply documented in most of the traditional digital circuit handbooks.

References


[Mutoh95] S. Mutoh et al., “1-V Power Supply High-Speed Digital Circuit Technology with Multi-
[Yuan89] J. Yuan and Svensson C., “High-Speed CMOS Circuit Technique,” *IEEE JSSC*, vol. 24,
no. 1, February 1989, pp. 62–70.
INSTRUCTIONS FOR DESIGNERS

DRAW IN UNITS OF LAMBDA.

ON ANY LAYER NO SPACE OR FEATURE SIZE MAY BE LESS THAN 2 LAMBDA WIDE.

FOR LAMBDA = 1.5 MICRONS, ALL FEATURE EDGES MUST BE ON A LAMBDA GRID.
HALF LAMBDA GRID ALLOWED FOR METALS.

FOR LAMBDA = 1.0 MICRONS AND BELOW,
ALL FEATURE EDGES MUST BE ON A HALF LAMBDA GRID.

WHAT YOU DRAW WILL BE VERY CLOSE TO WHAT YOU GET. MOSIS WILL TELL YOU THE DIFFERENCES.

SCALE YOUR CIF TO CENTIMICRONS. NEVER SUBMIT A DESIGN IN CENTILAMBDA.
WHAT VALUES LAMBDA?

LAMBDA=1.5 MICRONS
FOR 3 MICRON FABRICATORS

LAMBDA=1.0 MICRONS
FOR 2 MICRON FABRICATORS

LAMBDA=0.8 MICRONS
FOR 1.6 MICRON FABRICATORS

LAMBDA=0.6 MICRONS
FOR 1.2 MICRON FABRICATORS
TECHNOLOGIES AND REQUIRED LAYERS

<table>
<thead>
<tr>
<th>PROCESS</th>
<th>TECHNOLOGY</th>
<th>REQUIRED LAYERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMELL AND N SUBS THIN TUB</td>
<td>SCP</td>
<td>CWP, CSP</td>
</tr>
<tr>
<td>N WELL AND P SUBS THIN TUB</td>
<td>SCN</td>
<td>CWN, CSN OR CSP*</td>
</tr>
<tr>
<td>ALL **</td>
<td>SCG</td>
<td>CGW, CSG</td>
</tr>
<tr>
<td>ALL ***</td>
<td>SCE</td>
<td>CWP, CWN, CSP, CSN</td>
</tr>
</tbody>
</table>

* CSP may allow lower resistivity fld poly

** For a P Well or N Subs Thin Tub process, MOSIS sets CWP=CGW and CSP=CSG.

For an NWell or P Subs Thin Tub process, MOSIS sets CWN=CGW and CSN=CSG.

*** For a P Well or N Subs Thin Tub process, MOSIS ignores CWN and CSN.
For an NWell or P Subs Thin Tub process, MOSIS ignores CWP and uses CSP or CSN.
## Layer Names and Colors

<table>
<thead>
<tr>
<th>Layer</th>
<th>CIF</th>
<th>CALMA*</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>WELL</td>
<td>CWG</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>PWELL</td>
<td>CWP</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>NWELL</td>
<td>CWN</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>ACTIVE</td>
<td>CRA</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>SELECT</td>
<td>CSG</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>PSELECT</td>
<td>CSP</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>NSELECT</td>
<td>CSN</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>POLY</td>
<td>CPG</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>CONT TO POLY</td>
<td>CCP</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>CONT TO ACT</td>
<td>CCA</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>METAL1</td>
<td>CKF</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>VIA</td>
<td>CVR</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>METAL2</td>
<td>CMS</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>CONT TO ELEC</td>
<td>CCE</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>ELECTRODE</td>
<td>CEL</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>OVERGLASS</td>
<td>COG</td>
<td>52</td>
<td></td>
</tr>
</tbody>
</table>
1. WELL (NWELL, PWELL)

LAMBDAS

1.1 WIDTH 10

1.2 SPACE DIFF. POT. 9

1.3 SPACE SAME POT. 0 or 6

→ 1.1 ←

→ 1.3 ← → 1.2 ←

SAME POT. DIFF. POT.

Note: If both P and N wells submitted, they may not overlap but they may be coincident.
2. ACTIVE

2.1 Width

2.2 Space

2.3 Source/Drain Active to Well Edge

2.4 Subs./Well Contact, Active to Well Edge

Lambdas

3

3

5

3
3. POLY

3.1 Width 2
3.2 Space 2
3.3 Gate Overlap of Active 2
3.4 Active Overlap of Gate 3
3.5 Field Poly to Active 1
4. SELECT (PSELECT, NSELECT) LAMBDA S

4.1 Select space (overlap) to (op) channel to ensure adequate source/drain width

4.2 Select space (overlap) to (op) active

4.3 Select space (overlap) to (op) contact to well or substrate

4.4 Min width and space

NOTE: If both pselect and nselect submitted, they may be coincident but must not overlap.
5A. SIMPLER CONTACT TO POLY LAMBDAS

5A.1 Contact size exactly 2x2

5A.2 Poly overlap 2

5A.3 Spacing 2
5B. DENSER CONTACT TO POLY LAMBDAS

5B.1 Contact size, exactly 2x2
5B.2 Poly overlap of contact 1
5B.3 Spacing on same poly 2
5B.4 Spacing on diff poly 5
5B.5 Space to other poly 4
5B.6 Space to act, one contact 2
5B.7 Space to act, many contacts 3

Note: Your associating contacts with poly or active allows MOSIS to independently bloat the layer and the layer overlap of the contact.
6A. SIMPLER CONTACT TO ACTIVE

6A.1 Contact size exactly 2 x 2

6A.2 Active overlap 2

6A.3 Spacing 2

6A.4 Space to Gate 2
6B. DENSER CONTACT TO ACTIVE

6B.1 Contact size, exactly 2x2
6B.2 Active Overlap 1
6B.3 Spacing on same active 2
6B.4 Spacing on diff active 6
6B.5 Space to diff active 5
6B.6 Space to gate 2
6B.7 Space to field poly, one cont. 2
6B.8 Space to field poly, many cont. 3
6B.9 Space to contact to poly 4

Diagram:
- One contact
- Many contacts
- Many contacts
7. METAL1

7.1 Width 3

7.2 Space to Metal1 3

7.3 Overlap of Contact to Poly 1

7.4 Overlap of Contact to Active 1
8. VIA

8.1 Size: exactly 2x2

8.2 Separation to Via 3

8.3 Overlap by Metal1 1

8.4 Space to Poly or Active Edge 2

8.5 Space to Contact 2

Note: Objective is Via on a Flat Surface. Via Stacked Over Contact NOT allowed.
9. METAL2

9.1 Width 3

9.2 Space to Metal2 4

9.3 Overlap of Via 1