2

Transistors and Layout

2.1 Introduction

We will start our study of VLSI design by learning about transistors and wires and how they are fabricated. The basic properties of transistors are clearly important for logic design. Going beyond a minimally-functional logic circuit to a high-performance design requires the consideration of **parasitic circuit elements**—capacitance and resistance. Those parasitics are created as necessary by-products of the fabrication process which creates the wires and transistors, which gives us a very good reason to understand the basics of how integrated circuits are fabricated. We will also study the rules which must be obeyed when designing the masks used to fabricate a chip and the basics of layout design.

Our first step is to understand the basic fabrication techniques as described in Section 2.2. This material will describe how the basic structures for transistors and wires are made. We will then study transistors and wires, both as integrated structures and as circuit elements, in Section 2.3 and Section 2.4, respectively. We will study design rules for layout in Section 2.5. Finally, we will introduce some basic concepts and tools for layout design in Section 2.6.

2.2 Fabrication Processes

We need to study fabrication processes and the design rules that govern layout. Examples are always helpful. We will use as our example the SCMOS rules, which have been defined by MOSIS, the MOS Implementation Service. (MOSIS is supported by the United States National Science Foundation. Similar services, such as EuroChip/EuroPractice in the European Community, VDEC in Japan, and CIC in Taiwan, serve educational VLSI needs in other countries.) SCMOS is unusual in that it is not a single fabrication process, but a collection of rules that hold for a family of processes. Using generic technology rules gives greater flexibility in choosing a manufacturer for your chips. It also means that the SCMOS technology is less aggressive than any particular fabrication process developed for some special purpose—some manufacturers may emphasize transistor switching speed, for example, while others emphasize the number of layers available for wiring. We will point out advanced technology features that are not part of the SCMOS specification but may be found in a particular fabrication process.

2.2.1 Overview

A cross-section of an integrated circuit is shown in Figure 2-1. Integrated circuits are built on a silicon **substrate**. Components are formed by a combination of processes:

- **doping** the substrate with impurities to create areas such as the n+ and p+ regions;
- adding or cutting away insulating glass (**silicon dioxide**, or SiO_2) on top of the substrate;
- adding wires made of polycrystalline silicon (**polysilicon**, also known as **poly**) or metal, insulated from the substrate by SiO_2.

A pure silicon substrate contains equal numbers of two types of electrical carriers: electrons and holes. While we will not go into the details of device physics here, it is important to realize that the interplay between electrons and holes is what makes transistors work. The goal of doping is to create two types of regions in the substrate: an **n-type** region which contains primarily electrons and a **p-type** region which is dominated by holes. (Heavily doped regions are referred to as n+ and p+.) Transistor action occurs at properly formed boundaries between n-type and p-type regions.

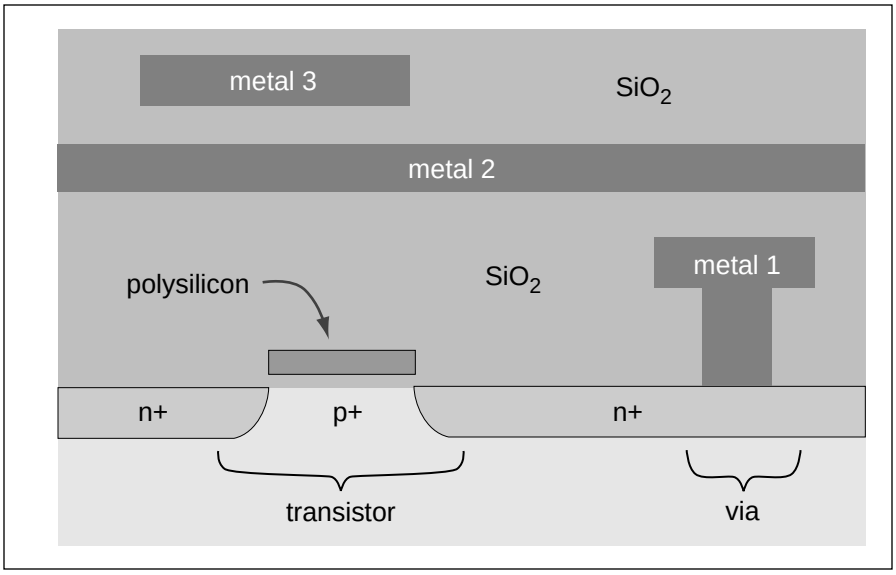


Figure 2-1: *Cross-section of an integrated circuit.*

The n-type and p-type regions can be used to make wires as well as transistors, but polysilicon (which is also used to form transistor gates) and metal are the primary materials for wiring together transistors because of their superior electrical properties. There may be several levels of metal wiring to ensure that enough wires can be made to create all the necessary connections. Glass insulation lets the wires be fabricated on top of the substrate using processes like those used to form transistors. The integration of wires with components (the invention of Robert Noyce), which eliminates the need to mechanically wire together components on the substrate, was one of the key inventions that made the integrated circuit feasible.

The key figure of merit for a fabrication process is the size—more specifically, the channel length—of the smallest transistor it can manufacture. Transistor size helps determine both circuit speed and the amount of logic that can be put on a single chip. Fabrication technologies are usually identified by their minimum transistor length, so a process which can produce a transistor with a 0.5 µm minimum channel length is called a 0.5 µm process. When we discuss design rules, we will recast the on-chip dimensions to a scalable quantity λ. our λ = 0.25 µm CMOS process is also known as a 0.5 µm CMOS process; if λ is not referred to explicitly, the size of the process gives the minimum channel length.

2.2.2 Fabrication Steps

Features are patterned on the wafer by a photolithographic process; the wafer is covered with light-sensitive material called **photoresist**, which is then exposed to light with the proper pattern. The patterns left by the photoresist after development can be used to control where SiO_2 is grown or materials are placed on the surface of the wafer.

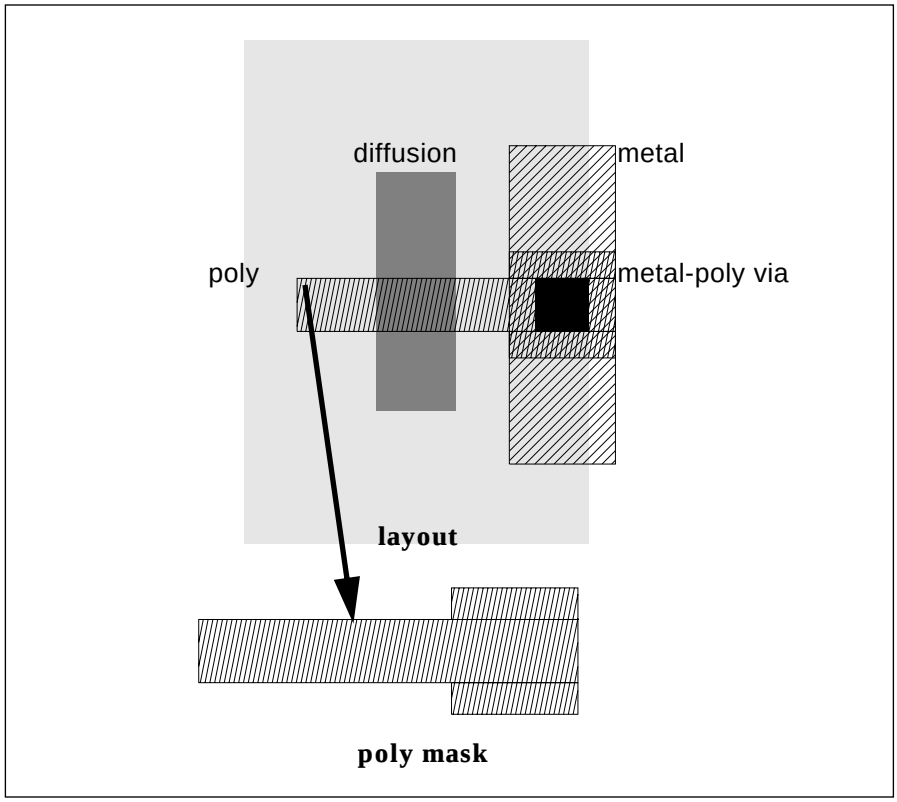


Figure 2-2: *The relationship between layouts and fabrication masks.*

A layout contains summary information about the patterns to be made on the wafer. Photolithographic processing steps are performed using **masks** which are created from the layout information supplied by the designer. In simple processes there is roughly one mask per layer in a layout, though in more complex processes some masks may be built from several layers while one layer in the layout may contribute to several masks. Figure 2-2 shows a simple layout and the mask used to form the polysilicon pattern.

Transistors are fabricated within regions called **tubs** or **wells**: an n-type transistor is built in a p-tub, and a p-type transistor is built in an n-tub. The wells prevent undesired conduction from the drain to the substrate. (Remember that the transistor type refers to the minority carrier which forms the inversion layer, so an n-type transistor pulls electrons out of a p-tub.) There are three ways to form tubs in a substrate:

- start with a p-doped wafer and add n-tubs;
- start with an n-doped wafer and add p-tubs;
- start with an undoped wafer and add both n- and p-tubs.

CMOS processes were originally developed from nMOS processes, which use p-type wafers into which n-type transistors are added. However, the **twin-tub process**, which uses an undoped wafer, has become the most commonly used process because it produces tubs with better electrical characteristics. We will therefore use a twin-tub process as an example.

Figure 2-3 illustrates important steps in a twin-tub process. Details can vary from process to process, but these steps are representative. The first step is to put tubs into the wafer at the appropriate places for the n-type and p-type wafers. Regions on the wafer are selectively doped by implanting ionized dopant atoms into the material, then heating the wafer to heal damage caused by ion implantation and further move the dopants by diffusion. The tub structure means that n-type and p-type wires cannot directly connect. Since the two diffusion wire types must exist in different type tubs, there is no way to build a via which can directly connect them. Connections must be made by a separate wire, usually metal, which runs over the tubs.

The next steps form an oxide covering of the wafer and the polysilicon wires. The oxide is formed in two steps: first, a thick field oxide is grown over the entire wafer. The field oxide is etched away in areas directly over transistors; a separate step grows a much thinner oxide which will form the insulator of the transistor gates. After the field and thin oxides have been grown, the polysilicon wires are formed by depositing polysilicon crystalline directly on the oxide.

Note that the polysilicon wires have been laid down before the diffusion wires were made—that order is critical to the success of MOS processing. Diffusion wires are laid down immediately after polysilicon deposition to create **self-aligned** transistors—the polysilicon masks the formation of diffusion wires in the transistor channel. For the transistor to work properly, there must

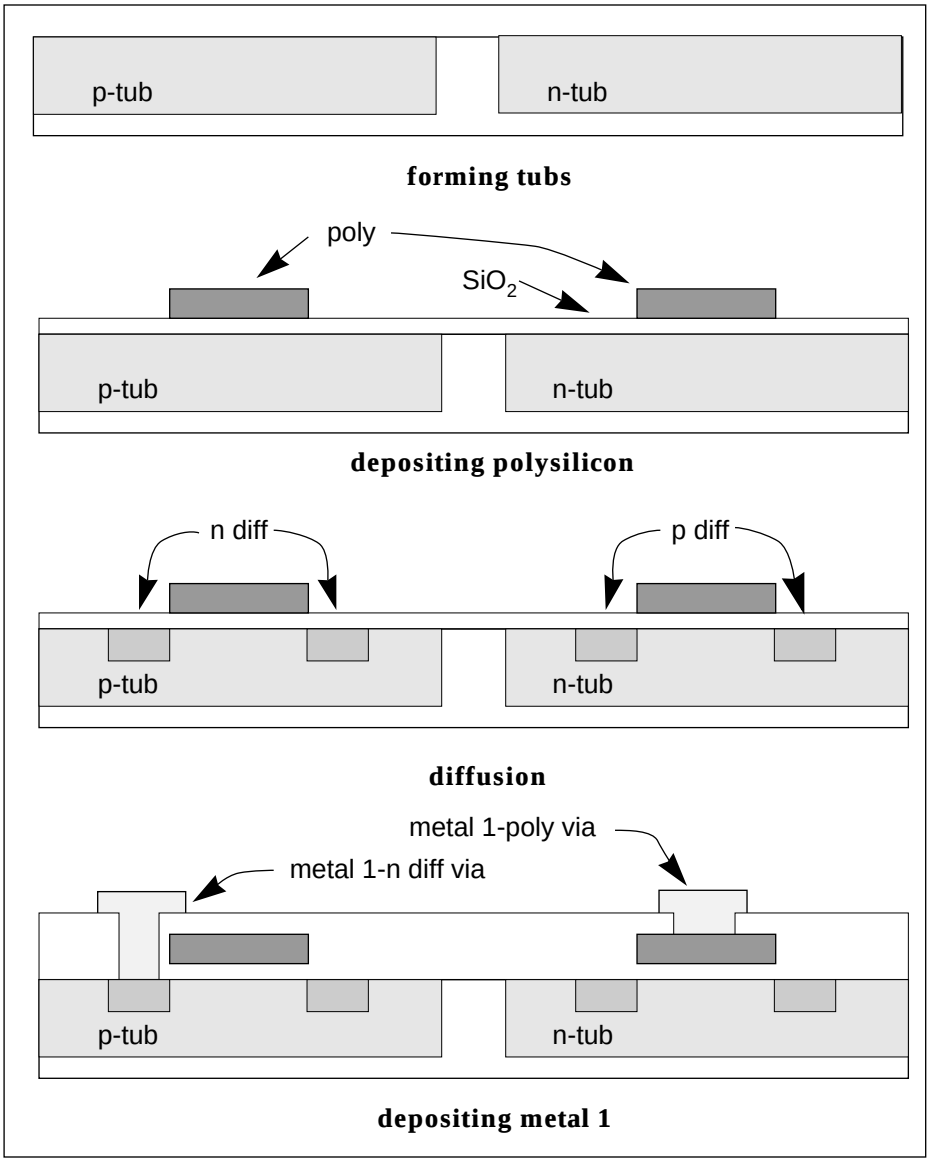


Figure 2-3: *Steps in processing a wafer.*

be no gap between the ends of the source and drain diffusion regions and the start of the transistor gate. If the diffusion were laid down first with a hole left for the polysilicon to cover, it would be very difficult to hit the gap with a polysilicon wire unless the transistor were made very large. Self-aligned processing allows much smaller transistors to be built.

After the diffusions are complete, another layer of oxide is deposited to insulate the polysilicon and metal wires. Although copper wires promise to provide substantially lower-resistance wires, at this writing metal connections are made using aluminum. Holes are cut in the field oxide where vias are desired, then metal 1 is deposited where desired. The metal fills the cuts to make connections between layers. The metal 2 layer requires an additional oxidation / cut / deposition sequence. After all the important circuit features have been formed, the chip is covered with a final **passivation layer** of SiO_2 to protect the chip from chemical contamination.

2.3 Transistors

2.3.1 Structure of the Transistor

Figure 2-4 shows the cross-section of an n-type MOS transistor. (The name MOS is an anachronism. The first such transistors used a metal wire for a gate, making the transistor a sandwich of metal, silicon dioxide, and the semiconductor substrate. Even though transistor gates are now made of polysilicon, the name MOS has stuck.) An n-type transistor is embedded in a p-type substrate; it is formed by the intersection of an n-type wire and a polysilicon wire. The region at the intersection, called the **channel**, is where the transistor action takes place. The channel connects to the two n-type wires which form the source and drain, but is itself doped to be p-type. The insulating silicon dioxide at the channel (called the **gate oxide**) is much thinner than it is away from the channel (called the **field oxide**); having a thin oxide at the channel is critical to the successful operation of the transistor.

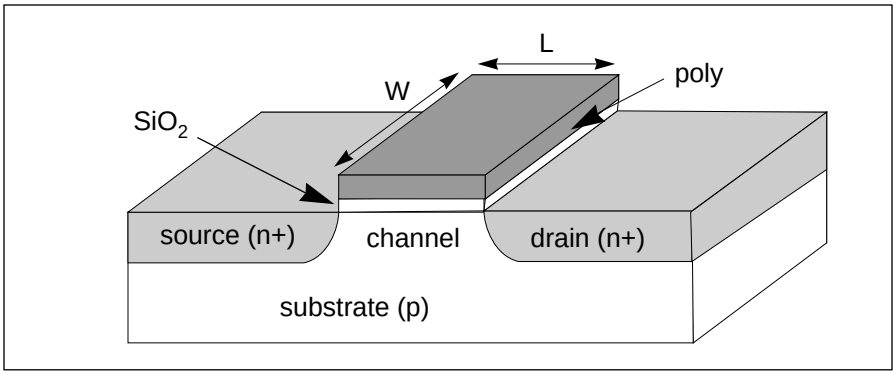


Figure 2-4: *Cross-section of an n-type transistor.*

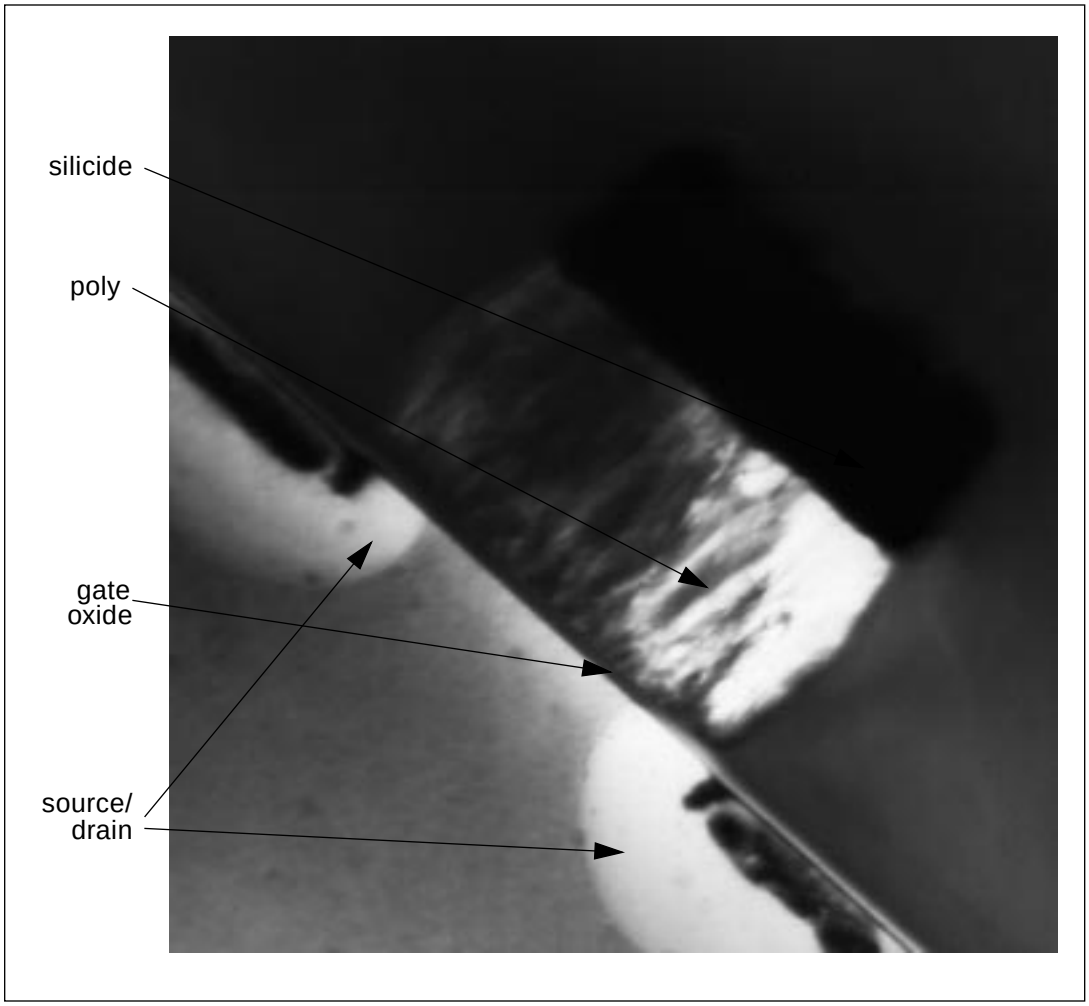


Figure 2-5: *Photomicrograph of a submicron MOS transistor (courtesy Lucent).*

Figure 2-5 shows a photomicrograph of an MOS transistor's cross-section. The photograph makes clear just how thin and sensitive the gate oxide is. The gate of this transistor is made of a sandwich of polysilicon and silicide. The sandwich's resistance is much lower than that of straight polysilicon.

The transistor works as a switch because the gate-to-source voltage modulates the amount of current that can flow between the source and drain. When the gate voltage (V_{gs}) is zero, the p-type channel is full of holes, while the n-type source and drain contain electrons. The p-n junction at the source termi-

nal forms a diode, while the junction at the drain forms a second diode that conducts in the opposite direction. As a result, no current can flow from the source to the drain.

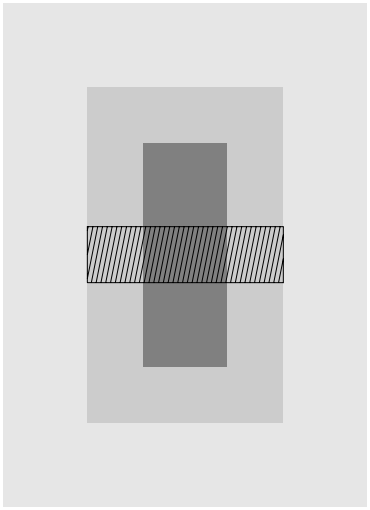
As V_{gs} rises above zero, the situation starts to change. While the channel region contains predominantly p-type carriers, it also has some n-type carriers. The positive voltage on the polysilicon which forms the gate attracts the electrons. Since they are stopped by the gate oxide, they collect at the top of the channel along the oxide boundary. At a critical voltage called the **threshold voltage** (V_t), enough electrons have collected at the channel boundary to form an **inversion layer**—a layer of electrons dense enough to conduct current between the source and the drain.

The size of the channel region is labeled relative to the direction of current flow: the channel **length** (L) is along the direction of current flow between source and drain, while the **width** (W) is perpendicular to current flow. The amount of current flow is a function of the W/L ratio, for the same reasons that bulk resistance changes with the object's width and length: widening the channel gives a larger cross-section for conduction, while lengthening the channel increases the distance current must flow through the channel. Since we can choose W and L when we draw the layout, we can very simply design the transistor current magnitude.

P-type transistors have identical structures but complementary materials: trade p's and n's in Figure 2-4 and you have a picture of a p-type transistor. The p-type transistor conducts by forming an inversion region of holes in the n-type channel; therefore, the gate-to-source voltage must be negative for the transistor to conduct current.

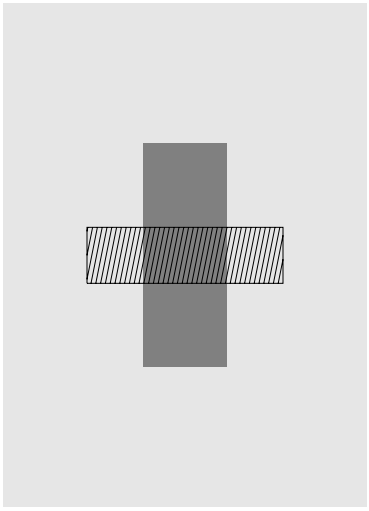
Example 2-1: Layout of n-type and p-type transistors

The basic layout of an n-type transistor is simple:



This layout is of a minimum-size transistor. Current flows through the channel vertically.

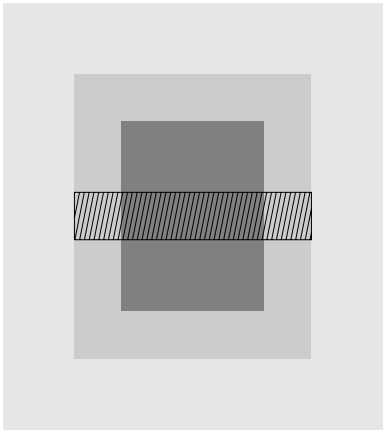
The layout of a p-type transistor is very similar:



In both cases, the tub rectangles are added as required. The details of which tub must be specified vary from process to process; many designers use simple programs to generate the tubs required around rectangles.

Fabrication engineers may sometimes refer to the **drawn length** of a transistor. Photolithography steps may affect the length of the channel. As a result, the actual channel length may not be the drawn length. The drawn length is usually the parameter of interest to the digital designer, since that is the size of rectangle that must be used to get a transistor of the desired size.

We can also draw a wider n-type transistor, which delivers more current:



2.3.2 A Simple Transistor Model

The behavior of both n-type and p-type transistors is described by two equations and two physical constants; the sign of one of the constants distinguishes the two types of transistors. The variables that describe a transistor's behavior, some of which we have already encountered, are:

- V_{gs}—the gate-to-source voltage;
- V_{ds}—the drain-to-source voltage (remember that $V_{ds} = -V_{sd}$);
- I_d—the current flowing between the drain and source.

The constants that determine the magnitude of source-to-drain current in the transistor are:

- V_t—the transistor threshold voltage, which is positive for an n-type

transistor and negative for a p-type transistor;

- k'—the transistor transconductance, which is positive for both types of transistors;
- W/L—the width-to-length ratio of the transistor.

Both V_t and k' are measured, either directly or indirectly, for a fabrication process. W/L is determined by the layout of the transistor, but since it does not change during operation, it is a constant of the device equations.

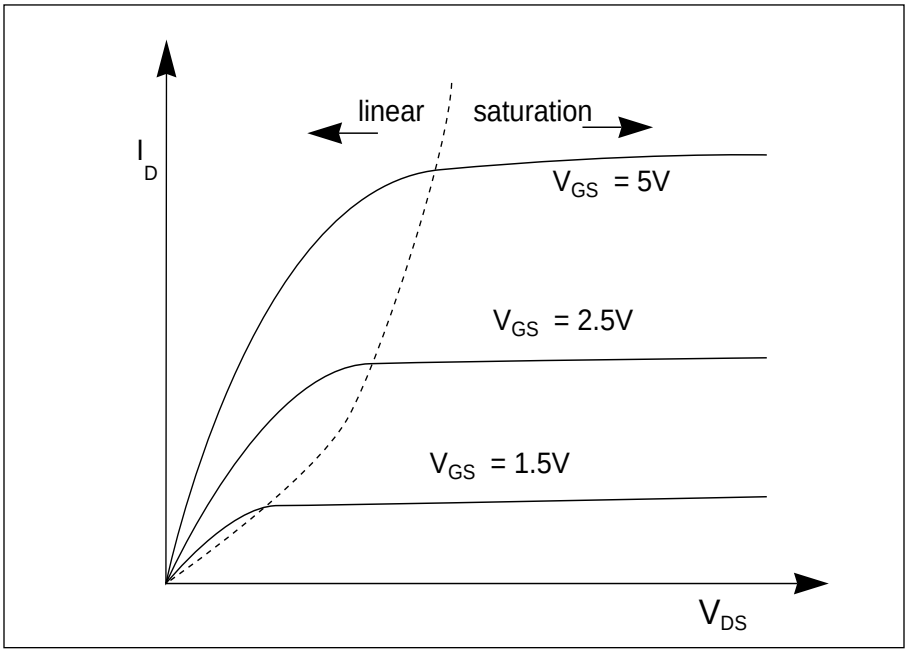


Figure 2-6: *The I_d curves of an n-type transistor.*

The equations that govern the transistor's behavior are traditionally written to show the drain current as a function of the other parameters. A reasonably accurate model for the transistor's behavior, written in terms of the drain current I_d, divides operation into **linear** and **saturated** [Yan78]. For an n-type transistor, we have:

- *Linear region* $V_{ds} < V_{gs} - V_t$:

$$I_d = k'\frac{W}{L}\left[(V_{gs} - V_t)V_{ds} - \frac{1}{2}V_{ds}^2\right] \qquad \textbf{(EQ 2-1)}$$

- *Saturated region* $V_{ds} \geq V_{gs} - V_t$:

$$I_d = \frac{1}{2}k'\frac{W}{L}(V_{gs} - V_t)^2 \qquad \textbf{(EQ 2-2)}$$

For a p-type transistor, the drain current is negative and the device is on when V_{gs} is below the device's negative threshold voltage. Figure 2-6 plots these equations over some typical values for an n-type device. Each curve shows the transistor current as V_{gs} is held constant and V_{ds} is swept from 0 V to a large voltage.

The transistor's switch action occurs because the density of carriers in the channel depends strongly on the gate-to-substrate voltage. For $|V_{gs}| < |V_t|$, there are not enough carriers in the inversion layer to conduct an appreciable current. (To see how much current is conducted in the subthreshold region, check Section 2.3.5.) Beyond that point and until saturation, the number of carriers is directly related to V_{gs}: the greater the gate voltage applied, the more carriers are drawn to the inversion layer and the greater the transistor's conductivity.

The relationship between *W/L* and source-drain current is equally simple. As the channel width increases, more carriers are available to conduct current. As channel length increases, however, the drain-to-source voltage diminishes in effect. V_{ds} is the potential energy available to push carriers from drain to source; as the distance from drain to source increases, it takes longer to push carriers across the transistor for a fixed V_{ds}, reducing current flow.

Table 2-1 *Typical transistor parameters for our 0.5 µm process.*

	k'	V_t
n-type	$k'_n = 73\mu A/V^2$	$0.7V$
p-type	$k'_p = 21\mu A/V^2$	$-0.8V$

Table 2-1 shows typical values of *k*' and V_t for a 0.5 µm process. The next example calculates the current through a transistor.

Example 2-2: Current through a transistor

A minimum-size transistor in the SCMOS rules is formed by a $L = 2\lambda$ and $W = 3\lambda$. Given this size of transistor and the 0.5 µm transistor characteristics, the current through a minimum-sized n-type transistor at the boundary between the linear and saturation regions when the gate is at the low voltage $V_{gs} = 2\text{V}$ would be

$$I_d = \frac{1}{2}\left(73\frac{\mu A}{V^2}\right)\left(\frac{3\mu\text{m}}{2\mu\text{m}}\right)(2V - 0.7V)^2 = 93\mu A \quad .$$

The saturation current when the transistor's gate is connected to a 5 V power supply would be

$$I_d = \frac{1}{2}\left(73\frac{\mu\text{A}}{\text{V}}\right)\left(\frac{3\mu\text{m}}{2\mu\text{m}}\right)(5\text{V} - 0.7\text{V})^2 = 1.0m\text{A} \quad .$$

2.3.3 Transistor Parasitics

Real devices have parasitic elements which are necessary artifacts of the device structure. Since the transistor is a non-linear device, we are primarily concerned with its capacitances as parasitics, although the source and drain regions have significant resistance.

The transistor itself introduces significant **gate capacitance**, C_g. This capacitance, which comes from the parallel plates formed by the poly gate and the substrate, forms the majority of the capacitive load in small logic circuits; $C_g = 0.9\text{fF}/\mu\text{m}^2$ for both n-type and p-type transistors in a typical 2 µm process. The total gate capacitance for a transistor is computed by measuring the area of the active region (or $W \times L$) and multiplying the area by the unit capacitance C_g. We don't worry about fringing because the edges of the gate are capacitively coupled to the source and drain and the total gate capaci-

tance contributed by all sources remains relatively constant throughout the transistor's operating regime.

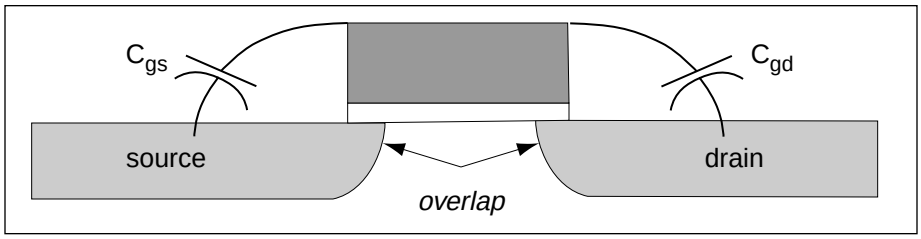


Figure 2-7: *Parasitic capacitances from the gate to the source/drain-overlap regions.*

We may, however, want to worry about the **source/drain overlap capacitances**. During fabrication, the dopants in the source / drain regions diffuse in all directions, including under the gate as shown in Figure 2-7. The source / drain overlap region tends to be a larger fraction of the channel area in deep submicron devices. The overlap region is independent of the transistor length, so it is usually given in units of Farads per unit gate width. Then the total source overlap capacitance for a transistor would be

$$Cgs = C_{ol}W. \qquad \textbf{(EQ 2-3)}$$

There is also a **gate/bulk overlap capacitance** due to the overhang of the gate past the channel and onto the bulk.

The source and drain regions also have a non-trivial capacitance to the substrate and a very large resistance. Circuit simulation may require the specification of source / drain capacitances and resistances. However, the techniques for measuring the source / drain parasitics at the transistor are the same as those used for measuring the parasitics of long diffusion wires. Therefore, we will defer the study of how to measure these parasitics to Section 2.4.1.

2.3.4 Tub Ties and Latchup

An MOS transistor is actually a four-terminal device, but we have up to now ignored the electrical connection to the substrate. The substrates underneath the transistors must be connected to a power supply: the p-tub (which contains n-type transistors) to V_{SS} and the n-tub to V_{DD}. These connections are made by special vias called **tub ties**.

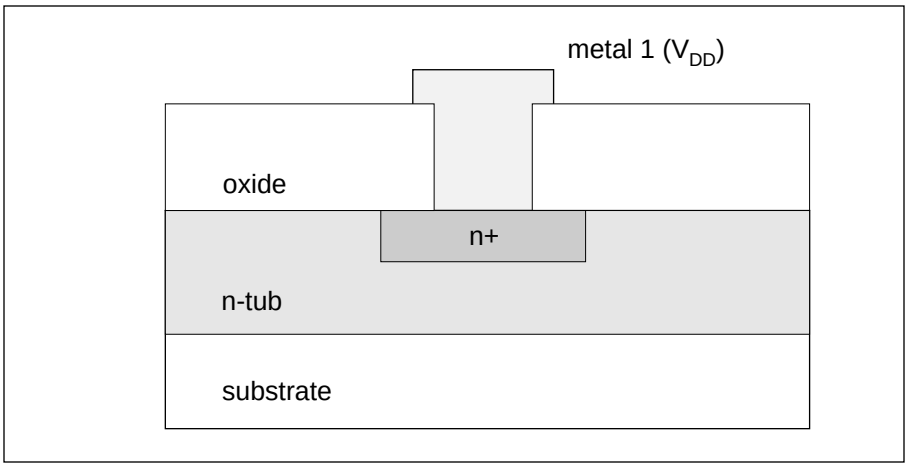


Figure 2-8: *Cross-section of an n-tub tie.*

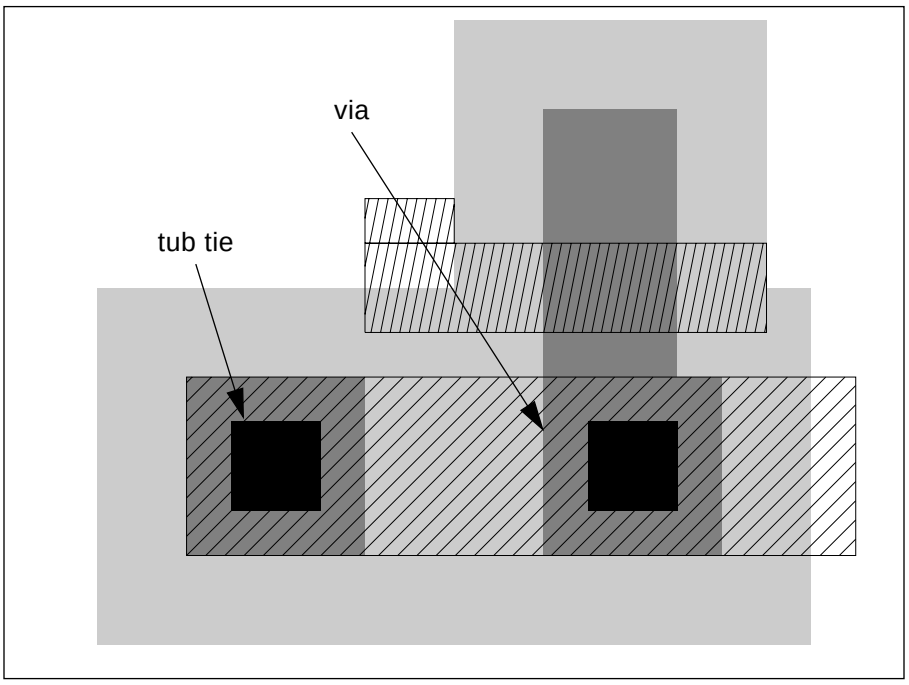


Figure 2-9: *A layout section featuring a tub tie.*

Figure 2-8 shows the cross-section of a tub tie connecting to an n-tub and Figure 2-9 shows a tub tie next to a via and an n-type transistor. The tie connects a metal wire connected to the V_{DD} power supply directly to the substrate. The connection is made through a standard via cut. The substrate underneath

the tub tie is heavily doped with n-type dopants (denoted as n+) to make a low-resistance connection to the tub. The SCMOS rules make the conservative suggestion that tub ties be placed every one to two transistors. Other processes may relax that rule to allow tub ties every four to five transistors. Why not place one tub tie in each tub—one tub tie for every 50 or 100 transistors? Using many tub ties in each tub makes a low-resistance connection between the tub and the power supply. If that connection has higher resistance, parasitic bipolar transistors can cause the chip to **latch-up**, inhibiting normal chip operation.

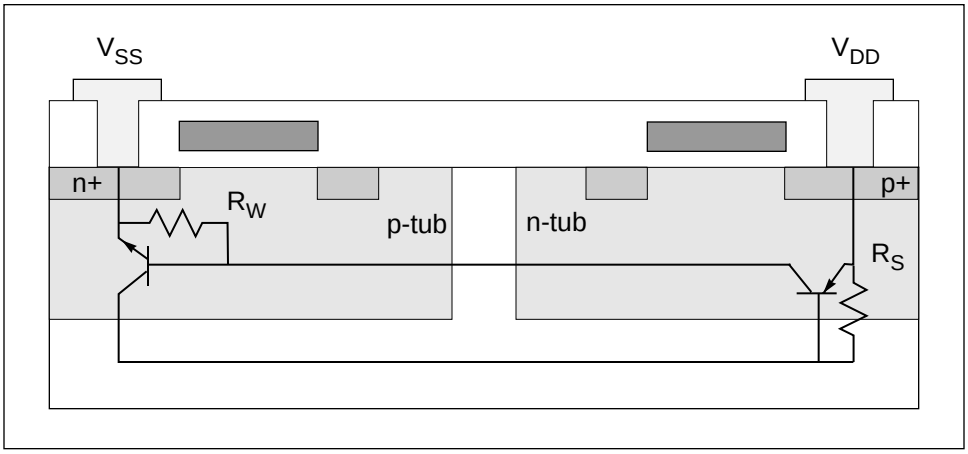


Figure 2-10: *Parasitics which cause latch-up.*

Figure 2-10 shows a chip cross-section which might be found in an inverter or other logic gate. The MOS transistor and tub structures form parasitic bipolar transistors: npn transistors are formed in the p-tub and pnp transistors in the n-tub. Since the tub regions are not physically isolated, current can flow between these parasitic transistors along the paths shown as wires. Since the tubs are not perfect conductors, some of these paths include parasitic resistors; the key resistances are those between the power supply terminals and the bases of the two bipolar transistors.

The parasitic bipolar transistors and resistors create a parasitic **silicon-controlled rectifier**, or SCR. The schematic for the SCR and its behavior are shown in Figure 2-11. The SCR has two modes of operation. When both bipolar transistors are off, the SCR conducts essentially no current between its two terminals. As the voltage across the SCR is raised, it may eventually turn on and conducts a great deal of current with very little voltage drop. The SCR formed by the n- and p-tubs, when turned on, forms a high-current, low-volt-

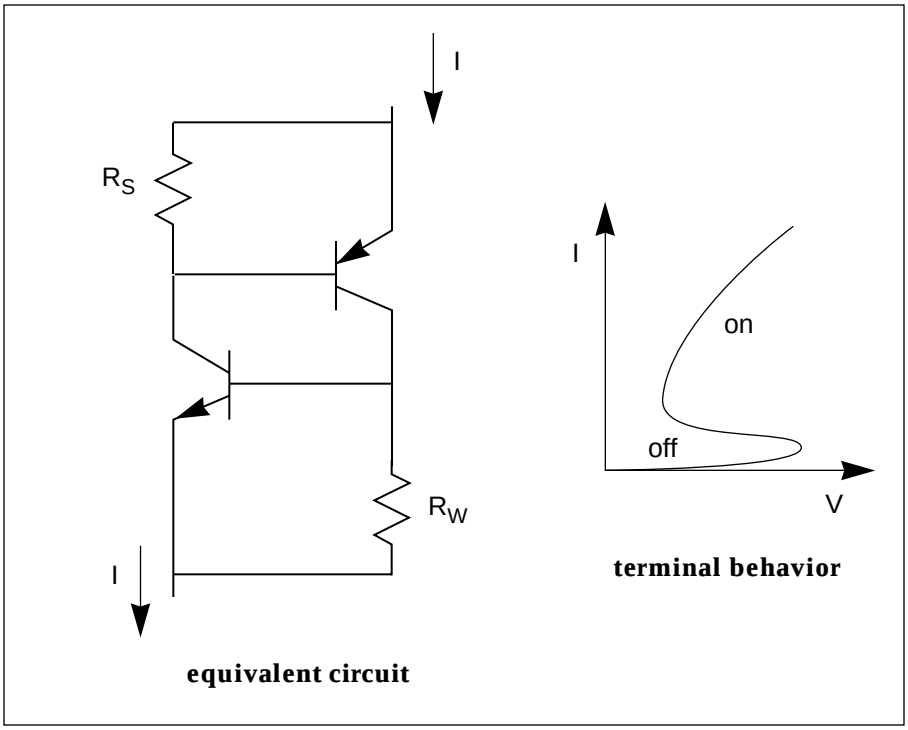


Figure 2-11: *Characteristics of a silicon-controlled rectifier.*

age connection between V_{DD} and V_{SS}. Its effect is to short together the power supply terminals. When the SCR is on, the current flowing through it floods the tubs and prevents the transistors from operating properly. In some cases, the chip can be restored to normal operation by disconnecting and then reconnecting the power supply; in other cases the high currents cause permanent damage to the chip.

The switching point of the SCR is controlled by the values of the two power supply resistances R_s and R_w. Each bipolar transistor in the SCR turns on when its base-to-emitter voltage reaches 0.7 V; that voltage is controlled by the voltage across the two resistors. The higher the resistance, the less stray current through the tub is required to cause a voltage drop across the parasitic resistance that can turn on the associated transistor. Adding more tub ties reduces the values of R_s and R_w. The maximum distance between tub ties is chosen to ensure that the chip will not latch-up during normal operation.

2.3.5 Advanced Transistor Characteristics

In order to better understand the transistor, we will derive the basic device characteristics that were stated in Section 2.3.2. Along the way we will be able to identify some second-order effects that can become significant when we try to optimize a circuit design.

The parallel place capacitance of the gate determines the characteristics of the channel. We know from basic physics that the parallel-plate oxide capacitance per unit area (in units of Farads per cm^2) is

$$C_{ox} = \varepsilon_{ox} / x_{ox}, \qquad \textbf{(EQ 2-4)}$$

where ε_{ox} is the permittivity of silicon dioxide (about $3.9\varepsilon_0$, where ε_0, the permittivity of free space, is $8.854 \times 10^{-14} F/cm$) and x_{ox} is the oxide thickness in centimeters.

Table 2-2 *Values of some physical constants.*

charge of an electron	q	$1.6 \times 10^{-19} C$
Si intrinsic carrier concentration	n_i	$1.45 \times 10^{12} C/cm^3$
permittivity of free space	ε_0	$8.854 \times 10^{-14} F/cm^2$
permittivity of Si	ε_{Si}	$3.9\varepsilon_0$
thermal voltage (300K)	kT/q	$0.026 V$

The intrinsic carrier concentration of silicon is denoted as n_i. N-type doping concentrations are written as N_d (donor) while p-type doping concentrations are written as N_a (acceptor). Table 2-2 gives the values of some important physical constants.

Applying a voltage of the proper polarity between the gate and substrate pulls minority carriers to the lower plate of the capacitor, namely the channel region near the gate oxide. The threshold voltage is defined as the voltage at which the number of minority carriers (electrons in an n-type transistor) in the channel region equals the number of majority carriers in the substrate. (This actually defines the **strong threshold condition**.) So the threshold voltage may be computed from the component voltages which determine the

number of carriers in the channel. The threshold voltage (assuming that the source / substrate voltage is zero) has four major components:

$$V_{t0} = V_{fb} + \phi_s + \frac{Q_b}{C_{ox}} + V_{II}. \quad \textbf{(EQ 2-5)}$$

Let us consider each of these terms.

- The first component, V_{fb}, is the **flatband voltag**e, which in modern processes has two main components:

  $$V_{fb} = \Phi_{gs} - Q_f / C_{ox} \quad \textbf{(EQ 2-6)}$$

  Φ_{gs} is the difference in work functions between the gate and substrate material, while Q_f is the fixed surface charge. (Trapped charge used to be a significant problem in MOS processing which increased the flatband voltage and therefore the threshold voltage. However, modern processing techniques control the amount of trapped charge.)

  If the gate polysilicon is n-doped at a concentration of N_{dp}, the formula for the work function difference is

  $$\Phi_{gs} = -\frac{kT}{q}\ln\left(\frac{N_a N_{dp}}{n_i^2}\right). \quad \textbf{(EQ 2-7)}$$

  If the gate is p-doped at a concentration of N_{ap}, the work function difference is

  $$\Phi_{gs} = \frac{kT}{q}\ln\left(\frac{N_{ap}}{N_a}\right). \quad \textbf{(EQ 2-8)}$$

- The second term is the surface potential. At the threshold voltage, the surface potential is twice the **Fermi potential** of the substrate:

  $$\phi_s \approx 2|\phi_F| = 2\frac{kT}{q}\ln\frac{N_a}{n_i}. \quad \textbf{(EQ 2-9)}$$

- The third component is the voltage across the parallel plate capacitor. The value of the charge on the capacitor Q_b is

  $$\sqrt{2q\varepsilon_{si}N_a\phi_s}. \quad \textbf{(EQ 2-10)}$$

(We will not derive this value, but the square root comes from the value for the depth of the depletion region.)

- An additional ion implantation step is also performed to adjust the threshold voltage—the fixed charge of the ions provides a bias voltage on the gate. The voltage adjustment V_{II} has the value qD_I/C_{ox}, where D_I is the ion implantation concentration; the voltage adjustment may be positive or negative, depending on the type of ion implanted.

When the source/substrate voltage is not zero, we must add another term to the threshold voltage. Variation of threshold voltage with source/substrate voltage is called **body effect**, which can significantly affect the speed of complex logic gates. The amount by which the threshold voltage is increased is

$$\Delta V_t = \gamma_n(\sqrt{\phi_s + V_{sb}} - \sqrt{\phi_s}) \quad \textbf{(EQ 2-11)}$$

The term γ_n is the **body effect factor**, which depends on the gate oxide thickness and the substrate doping:

$$\gamma_n = \frac{\sqrt{2q\varepsilon_{Si}N_A}}{C_{ox}}. \quad \textbf{(EQ 2-12)}$$

(To compute γ_p, we substitute the n-tub doping N_D for N_A.) We will see how body effect must be taken into account when designing logic gates in Section 3.3.4.

Example 2-3: Threshold voltage of a transistor

First, we will calculate the value of the threshold voltage of an n-type transistor at zero source/substrate bias. First, some reasonable values for the parameters:

- $x_{ox} = 200\mathring{A}$;
- $\varepsilon_{ox} = 3.5 \times 10^{-13} F/cm$;
- $\phi_s = 0.6V$;

- $Q_f = q \times 10^{11} = 1.6 \times 10^{-8} F/cm^2$;
- $\varepsilon_{si} = 1.0 \times 10^{-12}$;
- $N_A = 10^{15}$ cm^{-3};
- $N_{ap} = 10^{19}$ cm^{-3};
- $N_{II} = 1 \times 10^{12}$.

Let's compute each term of V_{t0}:

- $C_{ox} = \varepsilon_{ox}/x_{ox}$

  $= 3.45 \times 10^{-13}/2 \times 10^{-6} = 1.73 \times 10^{-7} F/cm^2$.
- $\Phi_{gs} = -\frac{kT}{q}\ln\left(\frac{N_a N_{dp}}{n_i^2}\right)$

  $= -0.026\left(\frac{10^{15}10^{19}}{(1.45 \times 10^{10})^2}\right)$

  $= -0.82\,V$
- $V_{fb} = \Phi_{gs} + Q_f/C_{ox}$

  $= -0.82 - 1.6 \times 10^{-8}/1.73 \times 10^{-7}$

  $= -0.91\,V$.
- $\phi_s = 2\frac{kT}{q}\ln\frac{N_a}{n_i}$

  $= 2 \times 0.026 \times \ln\left(\frac{10^{15}}{1.45 \times 10^{10}}\right)$

  $= 0.58\,V$
- $Q_b = \sqrt{2q\varepsilon_{si}N_a\phi_s}$

  $= \sqrt{2 \times (1.6 \times 10^{-19}) \times 1.0 \times 10^{-12} \times 10^{15} \times 0.58}$

  $= 1.4 \times 10^{-8}$
- $V_{II} = qD_I/C_{ox}$

  $= (1.6 \times 10^{-19}) \times (1 \times 10^{12})/(1.73 \times 10^{-7})$

$$= 0.92V$$

So,

$$V_{t0} = V_{fb} + \phi_s + \frac{Q_b}{C_{ox}} + V_{II}$$

$$= -0.91V + 0.58V + \frac{1.4 \times 10^{-8}}{1.73 \times 10^{-7}} + 0.92V$$

$$= 0.68V .$$

Note that it takes a significant ion implantation to give a threshold voltage that is reasonable for digital circuit design.

What is the value of the body effect at a source / substrate voltage of 5 V? That is the voltage the source will be raised to when it is in a chain of transistors in a logic gate. First, we compute the body effect factor:

$$\gamma_n = \frac{\sqrt{2q\varepsilon_{Si}N_A}}{C_{ox}}$$

$$= \frac{\sqrt{2 \times (1.6 \times 10^{-19}) \times 1.0 \times 10^{-12} \times 10^{15}}}{1.73 \times 10^{-7}}$$

$$= 0.1 .$$

Then

$$\Delta V_t = \gamma_n(\sqrt{\phi_s + V_{sb}} - \sqrt{\phi_s})$$

$$= 0.1(\sqrt{0.58V + 5} - \sqrt{0.58V})$$

$$= 0.16V .$$

While 0.16 V may not seem like much, it is 24% of the threshold voltage, a value large enough to cause delays under some conditions.

The drain current equation of Equation 2-1 can be found by integrating the charge over the channel. The charge at a point *y* is given simply by the definition of a parallel plate capacitance:

$$Q(y) = C_{ox}(V_{gs} - V_t - Vy). \quad \textbf{(EQ 2-13)}$$

The voltage differential over a differential distance in the channel is

$$dV = \frac{I_d dy}{\mu Q W}, \quad \textbf{(EQ 2-14)}$$

where μ is the (n- or p-) mobility at the surface and *W* is, of course, the channel width. Therefore, the total channel current is

$$I_d = \mu C_{ox}\frac{W}{L}\int_0^V (V_{gs} - V_t - V)(dV)ds. \quad \textbf{(EQ 2-15)}$$

The factor μC_{ox} is given the name *k'* or **process transconductance**. We sometimes call *k'W/L* the **device transconductance** β. This integral gives us the linear-region drain current formula of Equation 2-1. At saturation, our first-order model assumes that the drain current becomes independent of the drain voltage and maintains that value as V_{ds} increases. As shown in Figure 2-12, the depth of the inversion layer varies with the voltage drop across the length of the channel and, at saturation, its height has been reduced to zero.

But this basic drain current equation ignores the small dependence of drain current on V_{ds} in saturation. Increasing V_{ds} while in saturation causes the channel to shorten slightly, which in turn slightly increases the drain current. This phenomenon can be modeled by multiplying Equation 2-2 by a factor (1 + λV_{ds}). (Unfortunately, the **channel length modulation parameter** λ is given the same symbol as the scaling factor λ which will be introduced in the next chapter.) The value of λ is measured empirically, not derived. This gives us the new drain current equation for the saturation region

$$I_d = \frac{1}{2}k'\frac{W}{L}(V_{gs} - V_t)^2(1 + \lambda V_{ds}). \quad \textbf{(EQ 2-16)}$$

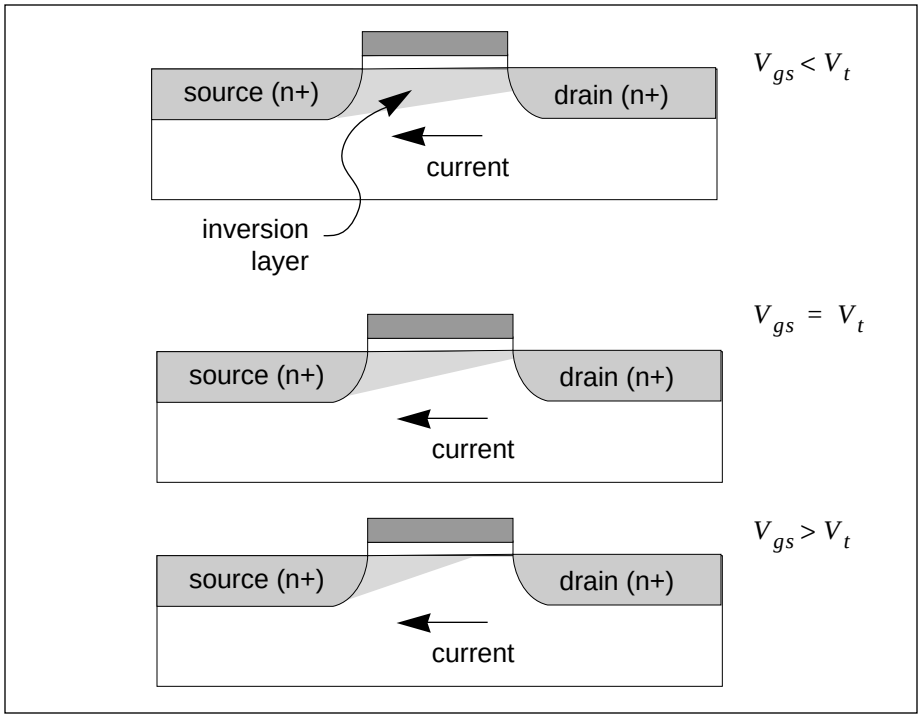


Figure 2-12: *Shape of the inversion layer as a function of gate voltage.*

Unfortunately, the λ term causes a slight discontinuity between the drain current equations in the linear and saturation regions—at the transition point, the λVds term introduces a small jump in I_d. A discontinuity in drain current is clearly not physically possible, but the discontinuity is small and usually can be ignored during manual analysis of the transistor's behavior. Circuit simulation, however, may require using a slightly different formulation that keeps drain current continuous.

It is also possible to model the subthreshold conduction of the transistor—the current carried when the gate voltage is below the threshold. The threshold voltage is chosen somewhat arbitrarily and the transistor does carry a very small current. Furthermore, that current drops off exponentially as the gate voltage decreases as shown in Figure 2-13. We can derive the subthreshold current by modeling the device as a bipolar transistor [Sze85], since the drain current is dominated by diffusion, not drift. In this case, the channel forms a very wide base for the bipolar transistor.

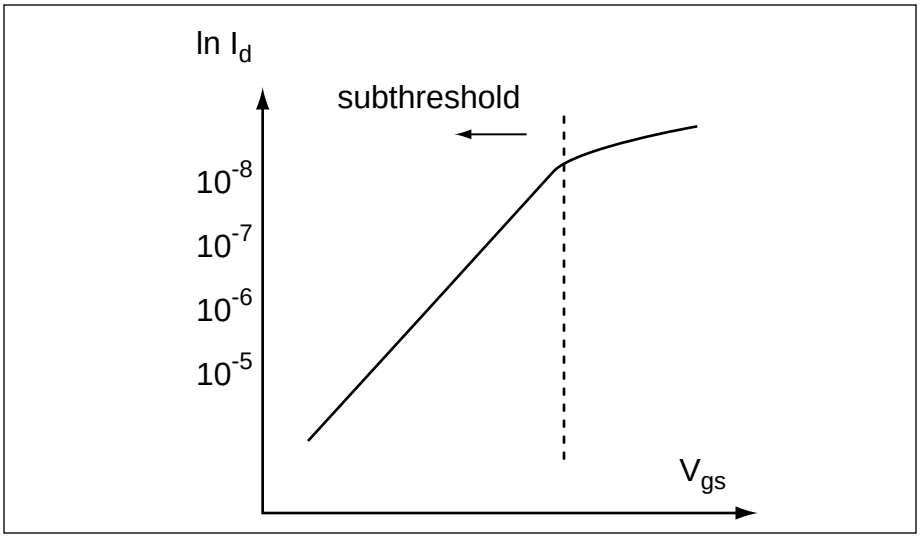


Figure 2-13: *Characteristics of subthreshold current.*

Then the current through the channel is

$$I_d = -qAD_n\frac{dn}{dy} = -qAD_n\frac{n(0)-n(L)}{L}. \quad \textbf{(EQ 2-17)}$$

In this formula, D_n is the diffusion coefficient, A is the area of the channel cross section (perpendicular to current flow). *n(0)* and *n(L)* are the electron densities at the source and drain, whose values are:

$$n(0) = n_i e^{q((\Psi_S-\Psi_B)/kT)}, \quad \textbf{(EQ 2-18)}$$

$$n(0L) = n_i e^{q((\Psi_S-\Psi_B-V_d)/kT)}. \quad \textbf{(EQ 2-19)}$$

By substituting into Equation 2-17, we can see that the subthreshold current is in fact an exponential function of threshold voltage. We also see that the subthreshold current is temperature-dependent.

The relationship between I_d and V_{gs} is approximately

$$\left(\frac{d}{dV_{gs}}\ln I_d\right)^{-1} = \frac{kT}{q}\ln 10(1+\alpha) \quad \textbf{(EQ 2-20)}$$

where α is a physical constant greater than 1. The details of subthreshold conduction are not important for most digital circuits. However, the fact that the transistor is not a perfect switch with infinite off impedance affects the design of dynamic gates because the subthreshold current changes the charge stored in the gate.

In contrast to the subthreshold drain current, **leakage currents** flow from the source or drain to the substrate. The source/drain and the substrate form a diode which conducts whenever the voltage across the diode is non-zero. The general form of the leakage current comes from the diode current law:

$$I_l = I_{l0}(e^{V_d/kT} - 1). \quad \textbf{(EQ 2-21)}$$

I_{l0} is the reverse saturation current, typically on the order of a few tenths of a nanoamp. Leakage currents are the source of static power dissipation in properly-designed CMOS circuits, as we will see in Section 3.3.5.

2.3.6 Advanced Transistor Structures

The modern MOS transistor is more complex than the basic transistor shown in Figure 2-4. A number of improvements to the basic MOS structure have been introduced over time to increase performance and permit the construction of efficient, short-channel devices [Bre90]. These new structures increase process complexity. The region between the source and drain is more heavily doped than the region underneath, allowing the source and drain to be put closer together. An epitaxial layer—a crystalline layer grown on top of the wafer during processing—allows fine control of doping in large regions; the lightly-doped region created underneath the source and drain by epitaxial growth reduces the junction capacitance between the two. The contacts to the source and drain are also designed to reduce source/drain capacitance by reducing the contact area. The diffusion is made thin near the channel area to reduce the penetration of the source/drain electric fields into the channel area. Many transistors also use lightly-doped drains to reduce the generation of **hot electrons**—high-energy electrons which can physically damage the drain region. We saw a silicided poly gate in Figure 2-5; silicides can also be applied to diffusions to reduce their resistance.

2.3.7 Spice Models

A circuit simulator, of which Spice [Nag75] is the prototypical example, provides the most accurate description of system behavior by solving for voltages and currents over time. The basis for circuit simulation is Kirchoff's laws, which describe the relationship between voltages and currents. Linear elements, like resistors and capacitors, have constant values in Kirchoff's laws, so the equations can be solved by standard linear algebra techniques. However, transistors are non-linear, greatly complicating the solution of the circuit equations. The circuit simulator uses a model—an equivalent circuit whose parameters may vary with the values of other circuits voltages and currents—to represent a transistor. Unlike linear circuits, which can be solved analytically, numerical solution techniques must be used to solve non-linear circuits. The solution is generated as a sequence of points in time. Given the circuit solution at time t, the simulator chooses a new time $t+\delta$ and solves for the new voltages and currents. The difficulty of finding the $t+\delta$ solution increases when the circuit's voltages and currents are changing very rapidly, so the simulator chooses the time step δ based on the derivatives of the Is and Vs. The resulting values can be plotted in a variety of ways using interactive tools.

A circuit simulation is only as accurate as the model for the transistor. Most versions of Spice offer three MOS transistor models, called naturally enough level 1, level 2, and level 3 [Gei90, Rab96]. The level 1Spice model is roughly the device equations of Section 2.3. The level 2Spice model provides more accurate determination of effective channel length and the transition between the linear and saturation regions, but is used less frequently today. The level 3Spice model uses empirical parameters to provide a better fit to the measured device characteristics. Each model requires a number of parameters which should be supplied by your fabrication vendor. The level 4 model, also known as the BSIM model, uses some extracted parameters, but is smaller and more efficient. Even more recent models, including level 28 (BSIM2) and level 47 (BSIM3) have been developed recently to more accurately model deep submicron transistors.

Table 2-3 gives the Spice names for some common parameters of Spice models and their correspondence to names used in the literature. Process vendors typically supply customers with Spice model parameters directly. You should use these values rather than try to derive them from some other parameters.

Table 2-3 *Names of some Spice parameters.*

parameter	**symbol**	**Spice name**
channel drawn length	L	L
channel width	W	W
source, drain areas		AS, AD
source, drain perimeters		PS, PD
source / drain resistances	R_s, R_d	RS, RD
source / drain sheet resistance		RSH
zero-bias bulk junction capacitance	C_{j0}	CJ
bulk junction grading coefficient	m	MJ
zero-bias sidewall capacitance	C_{jsw0}	CJSW
sidewall grading coefficient	m_{sw}	MJSW
gate-bulk / source / drain overlap capacitances	C_{gb0} / C_{gs0} / C_{gd0}	CGBO, CGSO, CGDO
bulk junction leakage current	I_s	IS
bulk junction leakage current density	J_s	JS
bulk junction potential	ϕ_0	PB
zero-bias threshold voltage	V_{t0}	VT0
transconductance	k'	KP
body bias factor	γ	GAMMA
channel modulation	λ	LAMBDA
oxide thickness	t_{ox}	TOX
lateral diffusion	x_d	LD
metallurgical junction depth	x_j	XJ
surface inversion potential	$2\lvert\phi_F\rvert$	PHI
substrate doping	N_A, N_D	NSUB
surface state density	Q_{ss}/q	NSS
surface mobility	μ_0	U0
maximum drift velocity	v_{max}	VMAX
mobility critical field	E_{crit}	UCRIT
critical field exponent in mobility degradation		UEXP
type of gate material		TPG

2.4 Wires and Vias

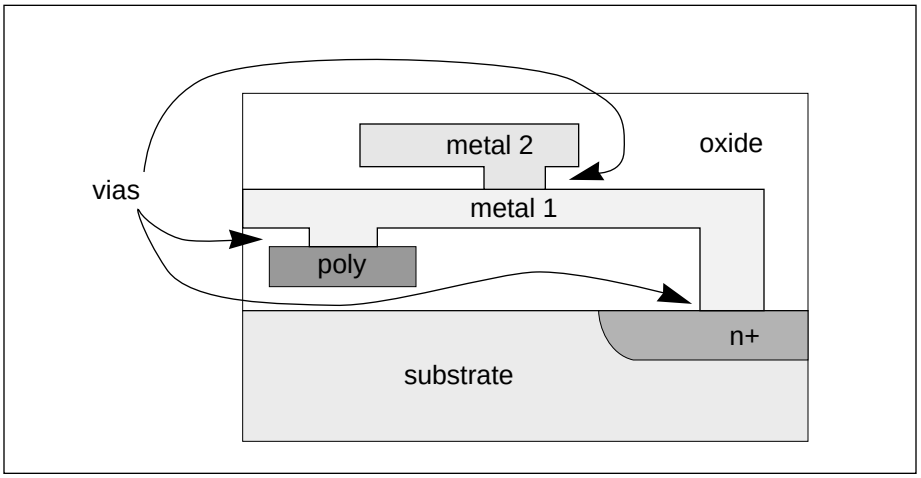


Figure 2-14: *A cross-section of a chip showing wires and vias.*

Figure 2-14 illustrates the cross-section of a nest of wires and vias. N-diffusion and p-diffusion wires are created by doping regions of the substrate. Polysilicon and metal wires are laid over the substrate, with silicon dioxide to insulate them from the substrate and each other. Wires are added in layers to the chip, alternating with SiO_2: a layer of wires is added on top of the existing silicon dioxide, then the assembly is covered with an additional layer of SiO_2 to insulate the new wires from the next layer. Vias are simply cuts in the insulating SiO_2; the metal flows through the cut to make the connection on the desired layer below. Figure 2-15 shows a photomicrograph of a multi-level interconnect structure, including four levels of metal and one level of polysilicon.

In addition to carrying signals, metal lines are used to supply power throughout the chip. On-chip metal wires have limited current-carrying capacity, as does any other wire. (Poly and diffusion wires also have current limitations, but since they are not used for power distribution those limitations do not affect design.) Electrons drifting through the voltage gradient on a metal line collide with the metal grains which form the wire. A sufficiently high-energy collision can appreciably move the metal grain. Under high currents, electron collisions with metal grains cause the metal to move; this process is called **metal migration** (also known as **electromigration**) [Mur93].

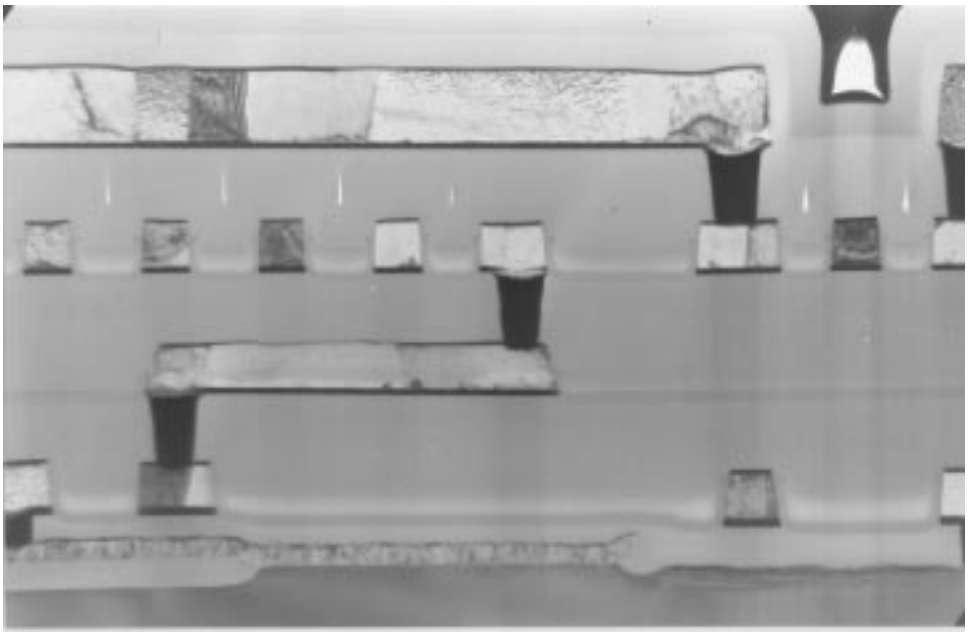


Figure 2-15: *Cross-section of 4-level metal/1-level poly interconnect (courtesy UMC).*

The **mean time to failure** (**MTF**) for metal wires—the time it takes for 50% of testing sites to fail—is a function of current density:

$$\text{MTF} \propto j^{-n} e^{Q/kT}, \qquad \textbf{(EQ 2-22)}$$

where j is the current density, n is a constant between 1 and 3, and Q is the diffusion activation energy. This equation is derived from the drift velocity relationship.

Metal wires can handle 1.5 mA of current per micron of wire width under the SCMOS rules. (Width is measured perpendicular to current flow.) A 3 µm wire can handle 4.5 mA of current. We will see in Chapter 3 that 4.5 mA is enough current to supply a large number of logic gates, so in small designs, metal migration limits are not a major problem. In larger designs, however, sizing power supply lines is critical to ensuring that the chip does not fail once it is installed in the field.

2.4.1 Wire Parasitics

Wires, vias and transistors all introduce parasitic elements into our circuits. Inductance is not a significant problem in current generations of integrated circuit technology (though, as we will see in Chapter 7, it is an important concern in IC packaging), so the parasitics of interest are resistance and capacitance. It is important to understand the structural properties of our components that introduce parasitic elements, and how to measure parasitic element values from layouts.

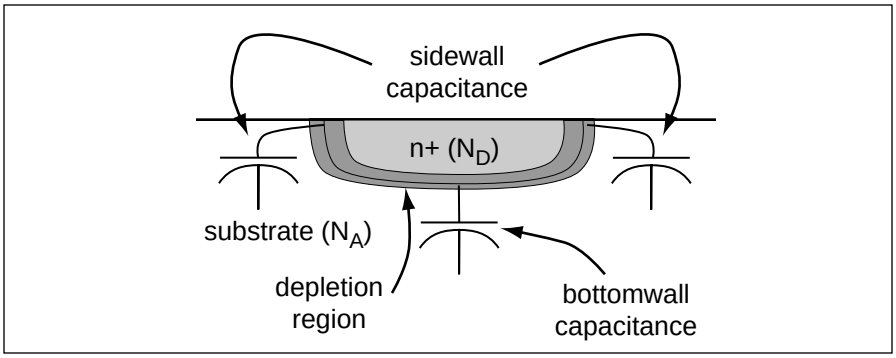


Figure 2-16: *Sidewall and bottomwall capacitances of a diffusion region.*

Diffusion wire capacitance is introduced by the p-n junctions at the boundaries between the diffusion and underlying tub or substrate. While these capacitances change with the voltage across the junction, which varies during circuit operation, we generally assume worst-case values. An accurate measurement of diffusion wire capacitance requires separate calculations for the bottom and sides of the wire—the doping density, and therefore the junction properties, vary with depth. To measure total capacitance, we measure the diffusion area, called **bottomwall** capacitance, and perimeter, called **sidewall** capacitance, as shown in Figure 2-16, and sum the contributions of each.

The **depletion region capacitance** value is given by

$$C_{j0} = \frac{\varepsilon_{si}}{x_d}. \qquad \textbf{(EQ 2-23)}$$

This is the **zero-bias depletion capacitance**, assuming zero voltage and an abrupt change in doping density from N_a to N_d. The depletion region width x_{d0} is shown in Figure 2-16 as the dark region; the depletion region is split between the n+ and p+ sides of the junction. Its value is given by

$$x_{d0} = \sqrt{\left(\frac{1}{N_A} + \frac{1}{N_D}\right)\frac{2\varepsilon_{si}V_{bi}}{q}}, \quad \textbf{(EQ 2-24)}$$

where the built-in voltage V_{bi} is given by

$$V_{bi} = \frac{kT}{q}\ln\left(\frac{N_A N_D}{n_i^2}\right). \quad \textbf{(EQ 2-25)}$$

The junction capacitance is a function of the voltage across the junction V_r:

$$C_j(V_r) = \frac{C_{j0}}{\sqrt{1 + \frac{V_r}{V_{bi}}}}. \quad \textbf{(EQ 2-26)}$$

So the junction capacitance decreases as the reverse bias voltage increases.

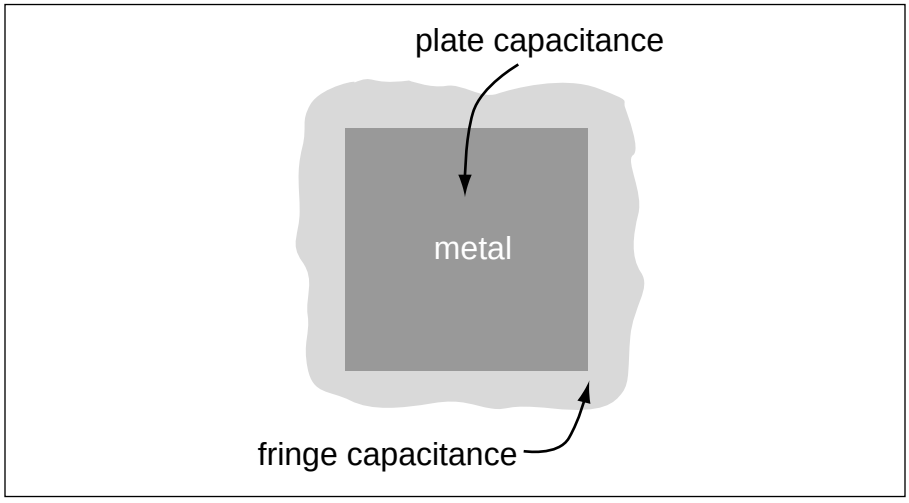


***Figure 2-17**: Plate and fringe capacitances of a parallel-plate capacitor.*

The capacitance mechanism for poly and metal wires is, in contrast, the parallel plate capacitor from freshman physics. We must also measure area and perimeter on these layers to estimate capacitance, but for different reasons. The **plate capacitance** per unit area assumes infinite parallel plates. We take

into account the changes in the electrical fields at the edges of the plate by adding in a **fringe capacitance** per unit perimeter. These two capacitances are illustrated in Figure 2-17. Capacitances can form between signal wires. In conservative technologies, the dominant parasitic capacitance is between the wire and the substrate, with the silicon dioxide layer forming the insulator between the two parallel plates.

However, as the number of metal levels increases and the substrate capacitance decreases, wire-to-wire parasitics are becoming more important. Both capacitance between two different layers and between two wires on the same layer are basic parallel plate capacitances. The parasitic capacitance between two wires on different layers, such as C_{m1m2} in Figure 2-18, depends on the area of overlap between the two wires. In a typical 0.5 µm process, the plate capacitance between metal 1 and metal 2 is 0.3 fF/cm^2 and the metal 1-metal3 plate capacitance is 0.1 fF/cm^2. When two wires run together for a long distance, with one staying over the other, the layer-to-layer capacitance can be very large. The capacitance between two wires on the same layer, C_{w1w2} in the figure, is formed by the vertical sides of the metal wires. Metal wires can be very tall in relation to their width, so the vertical wall coupling is non-negligible. However, this capacitance depends on the distance between two wires. The values given in process specifications are for minimum-separation wires, and the capacitance decreases by a factor of $1/x$ as distance increases. When two wires on the same layer run in parallel for a long distance, the coupling capacitance can become very large.

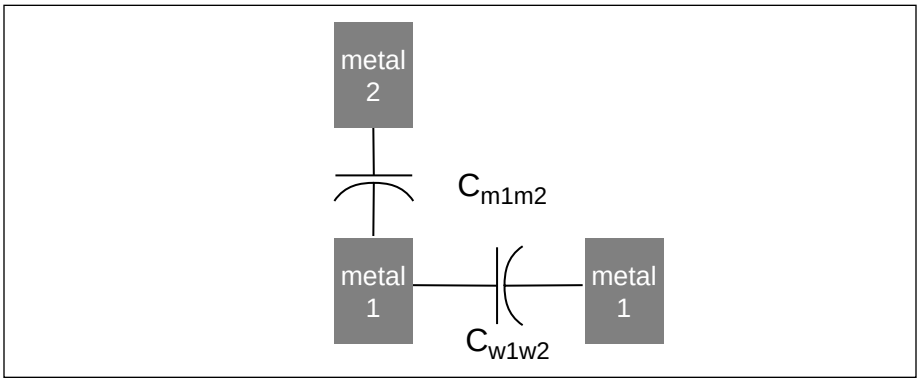


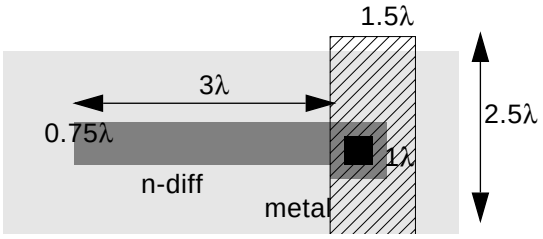
Figure 2-18: *Capacitive coupling between signals on the same and different layers.*

The following example illustrates how to measure parasitic capacitance from a layout.

Example 2-4: Parasitic capacitance measurement

N-diffusion wires in our typical 0.5 µm process have a bottomwall capacitance of 0.6 fF/mm^2 and a sidewall capacitance of 0.2 fF/µm. P-diffusion wires have bottomwall and sidewall capacitances of 0.9 fF/µm^2 and 0.3 fF/µm, respectively. The sidewall capacitance of a diffusion wire is typically as large or larger its bottomwall capacitances because the well/substrate doping is highest near the surface. Typical metal 1 capacitances in a process are 0.04 fF/µm^2 for plate and 0.09 fF/µm for fringe; typical poly values are 0.09 fF/µm^2 plate and 0.04 *f*F/µm fringe. The fact that diffusion capacitance is an order of magnitude larger than metal or poly capacitance suggests that we should avoid using large amounts of diffusion.

Here is our example wire, made of n-diffusion and metal connected by a via:



To measure wire capacitance of a wire, simply measure the area and perimeter on each layer, compute the bottomwall and sidewall capacitances, and add them together. The only potential pitfall is that our layout measurements are probably, as in this example, in λ units, while unit capacitances are measured in units of µm. The n-diffusion section of the wire occupies $3\,\mu m \times 0.75\,\mu m + 1\,\mu m \times 1\,\mu m = 3.25\,\mu m^2$, for a bottomwall capacitance of 2 *fF*. In this case, we count the n-diffusion which underlies the via, since it contributes capacitance to the substrate. The n-diffusion's perimeter is, moving counterclockwise from the upper left-hand corner, $0.75\mu m + 3\mu m + 0.25\mu m + 1\mu m + 1\mu m + 4\mu m = 10\mu m$, giving a total sidewall capacitance of 2 *fF*. Because the sidewall and bottomwall capacitances are in parallel, we add them to get the n-diffusion's contribution of 4 fF.

The metal 1 section has a total area of $2.5\mu m \times 1.5\mu m = 3.75\mu m^2$, giving a plate capacitance of 0.15 fF. The metal's perimeter is $2.5\mu m \times 2 + 1.5\mu m \times 2 = 8\mu m$ for a fringe capacitance of 0.72 fF and a total metal contribution of 0.87 fF. A slightly more accurate measurement would

count the metal area overlying the n-diffusion differently—strictly speaking, the metal forms a capacitance to the n-diffusion, not the substrate, since the diffusion is the closer material. However, since the via area is relatively small, approximating the metal 1-n-diffusion capacitance by a metal 1-substrate capacitance doesn't significantly change the result.

The total wire capacitance is the sum of the layer capacitances, since the layer capacitors are connected in parallel. The total wire capacitance is 4.9 fF; the n-diffusion capacitance dominates the wire capacitance, even though the metal 1 section of the wire is larger.

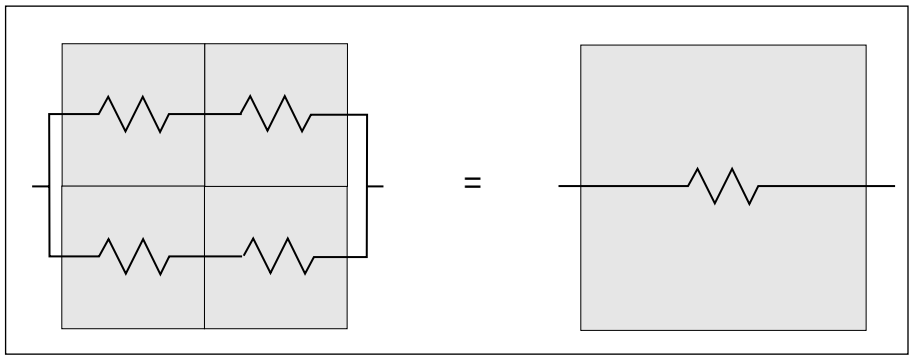


Figure 2-19: *Resistance per unit square is constant.*

Wire resistance is also computed by measuring the size of the wire in the layout, but the unit of resistivity is **ohms per square** ($\Omega/\square$), not ohms per square micron. The resistance of a square unit of material is the same for a square of any size; to understand, consider Figure 2-19. Assume that a unit square of material has a resistance of 1Ω. Two squares of material connected in parallel have a total resistance of $1/2\Omega$. Connecting two such rectangles in series creates a 2×2 square with a resistance of $1\ \Omega$. We can therefore measure the resistance of a wire by measuring its aspect ratio.

Figure 2-20 shows two example wires. The upper wire is made of polysilicon, which has a resistivity of $4\ \Omega/\square$ in our 0.5 µm process. Current flows in the direction shown; wire length is along the direction of current flow, while wire width is perpendicular to the current. The wire is composed of 18/3 squares connected in series, giving a total resistance of $24\ \Omega$.

The second wire is more interesting because it is bent. A 90° bend in a wire offers less resistance because electrons nearer the corner travel a shorter distance. A simple and common approximation is to count each square corner

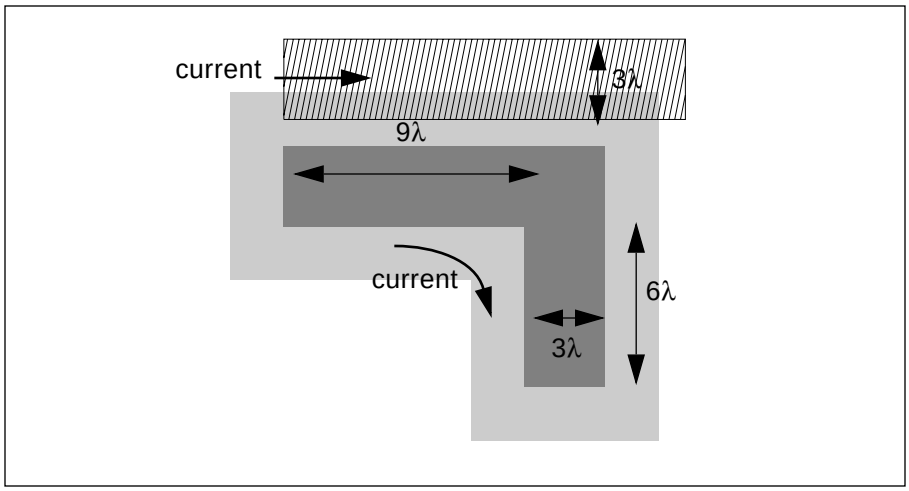


Figure 2-20: *An example of resistance calculation.*

rectangle as 1/2 squares of resistance. The wire can be broken into three pieces: 9/3 = 3 squares, 1/2 squares, and 6/3 = 2 squares. P-diffusion resistivity is approximately 2 Ω/□ , giving a total resistance of 11 Ω.

In our typical 0.5 µm process, an n-diffusion wire has a resistivity of approximately 2 Ω/□ , with metal 1, metal 2, and metal 3 having resistivities of about 0.08, 0.07, and 0.03 Ω/□ , respectively. Note that p-diffusion wires in particular have higher resistivity than polysilicon wires, and that metal wires have low resistivities.

The source and drain regions of a transistor have significant capacitance and resistance. These parasitics are, for example, entered into a Spice simulation as device characteristics rather than as separate wire models. However, we measure the parasitics in the same way we would measure the parasitics on an isolated wire, measuring area and perimeter up to the gate-source/drain boundary.

Vias have added resistance because the cut between the layers is smaller than the wires it connects and because the materials interface introduces resistance. The resistance of the via is usually determined by the resistance of the materials: a metal 1-metal 2 via has a typical resistance of less than 0.5 Ω while a metal1-poly contact has a resistance of 2.5 Ω. We rarely worry about the exact via resistance in layout design; instead, we try to avoid introducing unnecessary vias in current paths for which low resistance is critical.

2.5 Design Rules

Layouts are built from three basic component types: transistors, wires, and vias. We have seen the structures of these components created during fabrication. Now we will consider the design of the layouts which determine the circuit that is fabricated. **Design rules** govern the layout of individual components and the interactions—spacings and electrical connections—between those components. Design rules determine the low-level properties of chip designs: how small individual logic gates can be made; how small the wires connecting gates can be made, and therefore, the parasitic resistance and capacitance which determine delay.

Design rules are determined by the conflicting demands of component packing and chip yield. On the one hand, we want to make the components as small as possible, to put as many functions as possible on-chip. On the other hand, since the individual transistors and wires are about as small as the smallest feature that our manufacturing process can produce, errors during fabrication are inevitable: wires may short together or never connect, transistors may be faulty, etc. One common model for yield of a single type of structure is a Gamma distribution [Mur93]:

$$Y_i = \left(\frac{1}{1 + A\beta_i}\right)^{\alpha_i}. \qquad \textbf{(EQ 2-27)}$$

The total yield for the process is then the product of all the yield components:

$$Y = \prod_{i=1}^{n} Y_i. \qquad \textbf{(EQ 2-28)}$$

This formula suggests that low yield for even one of the process steps can cause serious final yield problems. But being too conservative about design rules leads to chips that are too large (which itself reduces yield) and too slow as well. We try to balance chip functionality and manufacturing yield by following rules for layout design which tell us what layout constructs are likely to cause the greatest problems during fabrication.

2.5.1 Fabrication Errors

The design rules for a particular process can be confusing unless you understand the motivation for the rules—the types of errors that are likely to occur while the chip is being manufactured. The design rules for a process are formulated to minimize the occurrence of common fabrication problems and bring the yield of correct chips to an acceptable level.

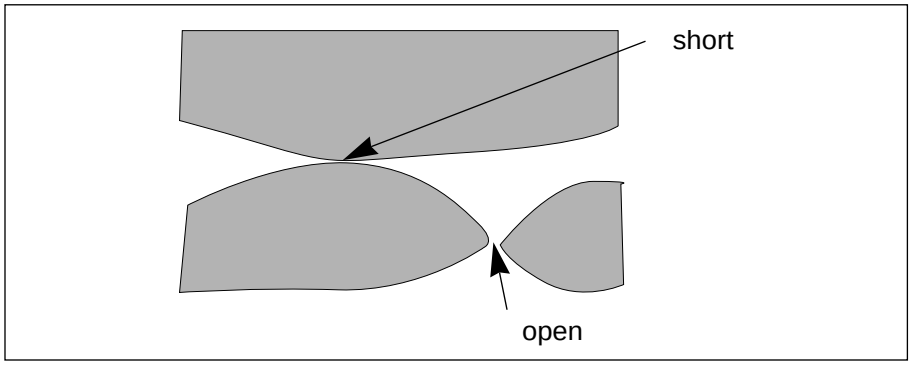


Figure 2-21: *Problems when wires are too wide or narrow.*

The most obvious type of fabrication problem is a wire or other feature being made too wide or too narrow. This problem can occur for a variety of reasons: photolithographic errors may leave an erroneous pattern for later steps; local materials variations may cause different rates of diffusion or deposition; processing steps at a nearby feature may cause harmful interactions. One important problem in fabrication is **planarization** [Gha94]—poly and metal wires leave hills in the oxide. The bumps in the oxide can be smoothed by several different chemical or mechanical methods; failure to do so causes s**tep coverage** problems which may lead to breaks in subsequent metallization layers. In any case, the result is a wire that is too narrow or too wide. As shown in Figure 2-21, a wire that is too narrow may never conduct current, or may burn out after some use. A too-wide wire may unintentionally short itself to another wire or, as in the case of a poly wire overlapping a parallel diffusion wire, cut into another element.

The simplest remedy for these problems is the introduction of **spacing** and **minimum-width** rules, which take a variety of forms in our design rules. Minimum width rules give a minimum size for a layout element; they help ensure that even with minor variations in the position of the lines that form the element, the element will be of an acceptable size. Spacing rules give a minimum distance between the edges of layout elements, so that minor pro-

cessing variations will not cause the element to overlap nearby layout elements.

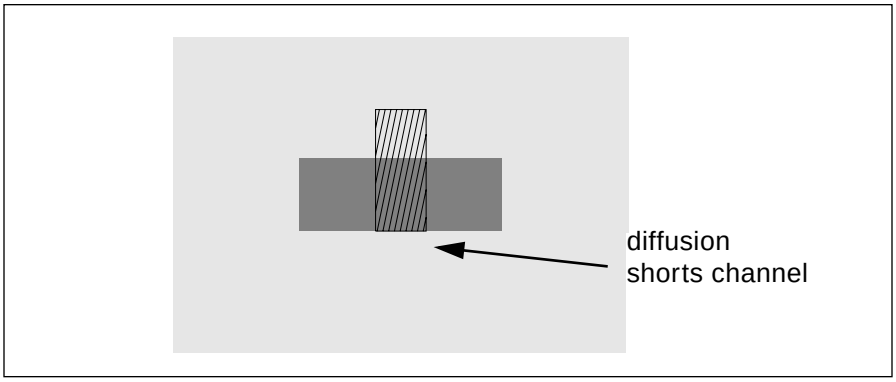


Figure 2-22: *Potential problems in transistor fabrication.*

We also have a number of **composition** rules to ensure that components are well-formed. Consider the transistor layout in Figure 2-22—the transistor action itself takes place in the channel, at the intersection of the polysilicon and diffusion regions, but a valid transistor layout requires extensions of both the poly and diffusion regions beyond the boundary. The poly extensions ensure that no strand of diffusion shorts together the source and drain. The diffusion extensions ensure that adequate contact can be made to the source and drain.

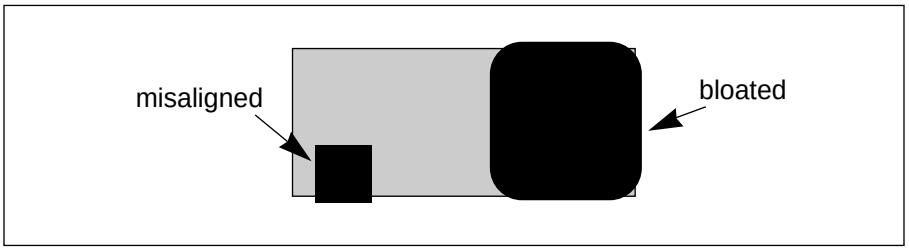


Figure 2-23: *Potential problems in via fabrication.*

Vias have construction rules as well: the material on both layers to be connected must extend beyond the SiO_2 cut itself; and the cut must be of a fixed size. As shown in Figure 2-23, the overlap requirement simply ensures that the cut will completely connect the desired layout elements and not mistakenly connect to the substrate or another wire. The key problem in via fabrication, however, is making the cuts. A large chip may contain millions of vias, all of which must be opened properly for the chip to work. The acid etching process which creates cuts must be very uniform—cuts may be neither too small and shallow nor too large. It isn't hard to mount a bookcase in a wall with an electric drill—it is easy to accurately size and position each hole

required. Now imagine making those holes by covering the wall with acid at selected points, then wiping the wall clean after a few minutes, and you should empathize with the problems of manufacturing vias on ICs. The cut must also be filled with material without breaking as the material flows over the edge of the cut. The size, shape, and spacing of via cuts are all strictly regulated by modern fabrication processes to give maximum via yield.

2.5.2 Scalable Design Rules

Manufacturing processes are constantly being improved. The ability to make ever-smaller devices is the driving force behind Moore's Law. But many characteristics of the fabrication process do not change as devices shrink—layouts do not have to be completely redesigned, simply shrunk in size. We can take best advantage of process scaling by formulating our design rules to be explicitly scalable.

We will scale our design rules by expressing them not in absolute physical distances, but in terms of λ, the size of the smallest feature in a layout. All features can be measured in integral multiples of λ. By choosing a value for λ we set all the dimensions in a scalable layout.

Scaling layouts makes sense because chips actually get faster as layouts shrink. As a result, we don't have to redesign our circuits for each new process to ensure that speed doesn't go down as packing density goes up. If circuits became slower with smaller transistors, then circuits and layouts would have to be redesigned for each process.

Digital circuit designs scale because the capacitive loads that must be driven by logic gates shrink faster than the currents supplied by the transistors in the circuit [Den74]. To understand why, assume that all the basic physical parameters of the chip are shrunk by a factor $1/x$:

- lengths and widths: $W \rightarrow W/x, L \rightarrow L/x$;
- vertical dimensions such as oxide thicknesses: $t_{ox} \rightarrow t_{ox}/x$;
- doping concentrations: $N_d \rightarrow N_d/x$;
- supply voltages: $V_{DD} - V_{SS} \rightarrow (V_{DD} - V_{SS})/x$.

We now want to compute the values of scaled physical parameters, which we will denote by variables with hat symbols. One result is that the transistor

transconductance scales: since $k' = (\mu_{eff}\varepsilon_{ox})/t_{ox}$ [Mul77], $\hat{k}'/k' = x$. (μ_{eff} is the carrier mobility and ε_{ox} is the dielectric constant.) The threshold voltage scales with oxide thickness, so $\hat{V}_t = V_t/x$. Now compute the scaling of the saturation drain current W/L:

$$\frac{\hat{I}_d}{I_d} = \left(\frac{\hat{k}'}{k'}\right)\left(\frac{\hat{W}/\hat{L}}{W/L}\right)\left[\frac{(\hat{V}_{gs}-\hat{V}_t)^2}{(V_{gs}-V_t)^2}\right]$$
$$= x\left(\frac{1/x}{1/x}\right)\left(\frac{1}{x}\right)^2 \qquad \textbf{(EQ 2-29)}$$
$$= \frac{1}{x}$$

The scaling of the gate capacitance is simple to compute: $C_g = \varepsilon_{ox}WL/t_{ox}$, so $\hat{C}_g/C_g = 1/x$. The total delay of the logic circuit depends on the capacitance to be charged, the current available, and the voltage through which the capacitor must be charged; we will use CV/I as a measure of the speed of a circuit over scaling. The voltage through which the logic circuit swings is determined by the power supply, so the voltage scales as $1/x$. When we plug in all our values,

$$\frac{\hat{C}\hat{V}/\hat{I}}{CV/I} = \frac{1}{x}. \qquad \textbf{(EQ 2-30)}$$

So, as the layout is scaled from λ to $\hat{\lambda} = \lambda/x$, the circuit is actually speeded up by a factor x.

In practice, few processes are perfectly λ-scalable. As process designers learn more, they inevitably improve some step in the process in a way that does not scale. High-performance designs generally require some modification when migrated to a smaller process as detailed timing properties change. However, the scalability of VLSI systems helps contain the required changes.

2.5.3 SCMOS Design Rules

Finally, we reach the SCMOS design rules themselves[1]. We will cast these rules in terms of λ. For the SCMOS rules, a 0.5 µm process, the nominal value for λ is 0.25 µm. However, fabrication lines often make minor adjustments to the masks (bloating or shrinking) to adjust the final physical dimensions of

the features. We will concentrate on the rules for the basic two-level metal process, with some discussion of the rules for metal 3, since the rules for the higher levels of metal are not as well standardized as for the two-level metal process. At this writing, MOSIS provides up to four metal layers in some processes.

Design rules are generally specified as pictures illustrating basic situations, with notes to explain features not easily described graphically. While this presentation may be difficult to relate to a real layout, practice will teach you to identify potential design rule violations in a layout from the prototype situations in the rules. Many layout editor programs, such as Magic [Ost84], have built-in design-rule checkers which will identify design-rule violations on the screen for you. Using such a program is a big help in learning the process design rules.

Figure 2-24 summarizes the design rules. Classifying the situations described in these pictures as separation, minimum size, or composition will help you distinguish and learn the rules. Many of these rules hold for any tub structure: n-tub, p-tub, or twin-tub. The rules regarding tubs and tub ties necessarily depend on the tub structure, however.

The basic separation and minimum size rules are:

- **metal 1** Minimum width is 3 λ, minimum separation is 3 λ.
- **metal 2** Minimum width is 3 λ, minimum separation is 4 λ.
- **polysilicon** Minimum width is 2 λ, minimum poly–poly separation is 2 λ.
- **p-, n-diffusion** Minimum width is 3 λ, minimum separation between same-type diffusions is 3 λ, minimum p-diff–n-diff separation is 10 λ.
- **tubs** Tubs must be at least 10 λ wide. The minimum distance from the tub edge to source / drain active area is 5 λ.

The basic construction rules are:

---

1. We will describe the most recent set of rules available to us. Since these rules change over time, it is always best to consult MOSIS (http://www.mosis.org) before starting a design that will be fabricated using the SCMOS rules.

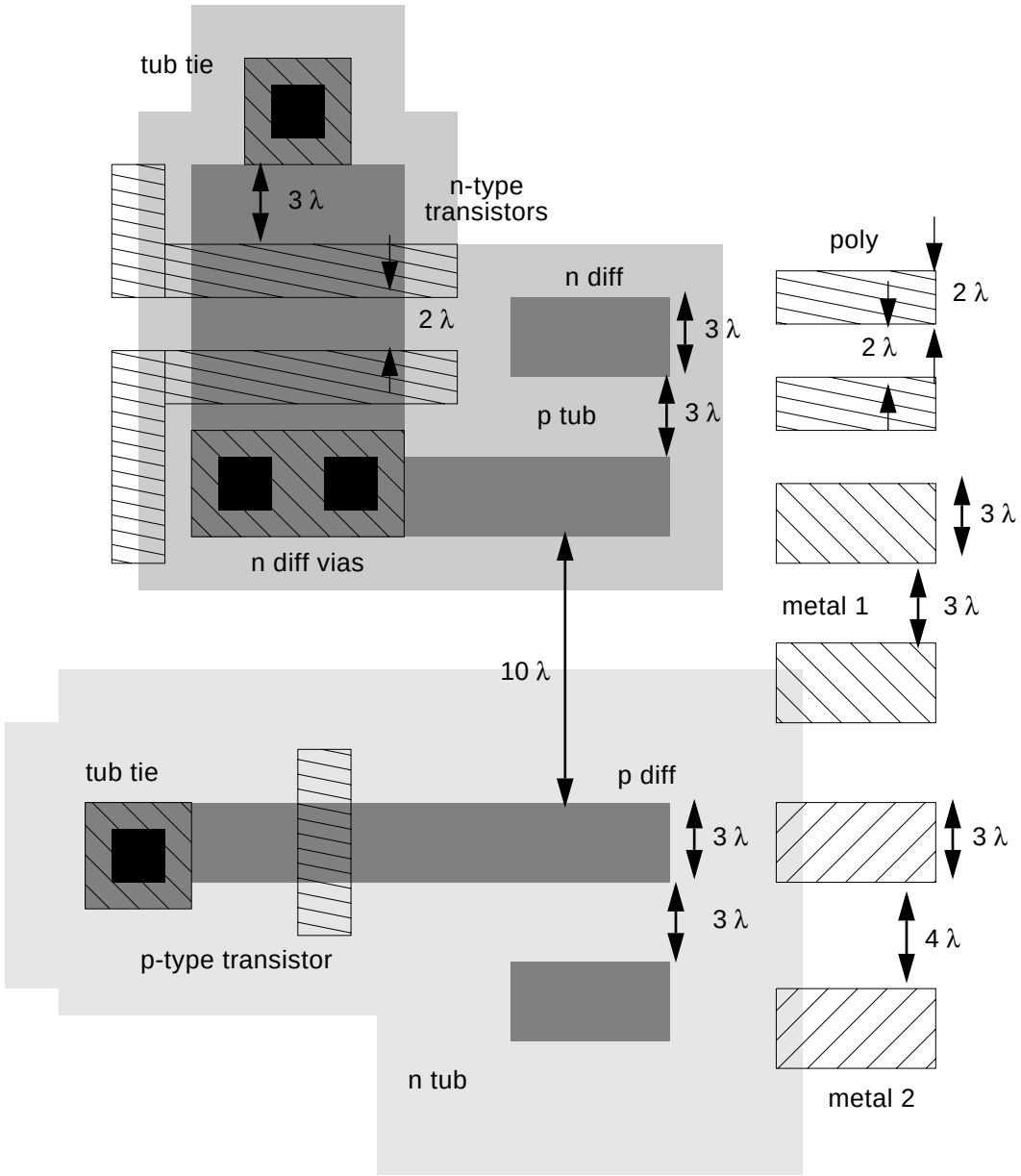


Figure 2-24: *A summary of the SCMOS design rules for two-level metal processes.*

- **transistors** The smallest transistor is of width 3 λ and length 2 λ; poly extends 2 λ beyond the active region and diffusion extends 3 λ. The active region must be at least 1 λ from a poly-metal via, 2 λ from another transistor, and 3 λ from a tub tie.
- **vias** Cuts are 2 λ × 2 λ; the material on both layers to be connected extends 1 λ in all directions from the cut, making the total via size 4 λ × 4 λ. (MOSIS also suggests another via construction with 1.5 λ of material around the cut. This construction is safer but the fractional design rule may cause problems with some design tools.) Available via types are:
  - n / p-diffusion–poly;
  - poly–metal 1;
  - n / p-diffusion–metal 1;
  - metal 1–metal 2;

If several vias are placed in a row, successive cuts must be at least 2 λ apart. Spacing to a via refers to the complete 4 λ × 4 λ object, while spacing to a via cut refers to the 2 × 2 λ cut.

- **tub ties** A p-tub tie is made of a 2 λ × 2 λ cut, a 4 λ × 4 λ metal element, and a 4 λ × 4 λ p^+ diffusion. An n-tub tie is made with an n^+ diffusion replacing the p^+ diffusion. A tub tie must be at least 2 λ from a diffusion contact.

It is important to remember that different rules have different dependencies on electrical connectivity. Spacing rules for wires, for example, depend on whether the wires are on the same electrical node. Two wire segments on the same electrical node may touch. However, two via cuts must be at least 2 λ apart even if they are on the same electrical net. Similarly, two active regions must always be 2 λ apart, even if they are parallel transistors.

The rules for metal 3 are:

- Minimum metal 3 width is 6 λ, minimum separation is 4 λ.
- Available via from metal 3 is to metal 2. Connections from metal 3 to other layers must be made by first connecting to metal 2.

There are some specialized rules that do not fit into the separation / minimum size / composition categorization.

- A cut to polysilicon must be at least 3 λ from other polysilicon.
- Polysilicon cuts and diffusion cuts must be at least 2 λ apart.
- A cut must be at least 2 λ from a transistor active region.
- A diffusion contact must be at least 4 λ away from other diffusion.
- A metal 2 via must not be directly over polysilicon.

One final special rule is to avoid generating small **negative features**. Consider the layout of Figure 2-25: the two edges of the notch are 1 λ apart, but both sides of the notch are on the same electrical node. The two edges are not in danger of causing an inadvertent short due to a fabrication error, but the notch itself can cause processing errors. Some processing steps are, for convenience, done on the negative of the mask given, as shown in the figure. The notch in the positive mask forms a 1λ wide protrusion on the negative mask. Such a small feature in the photoresist can break off during processing, float around the chip, and land elsewhere, causing an unwanted piece of material. We can minimize the chances of stray photoresist causing problems by requiring all negative features to be at least 2 λ in size.

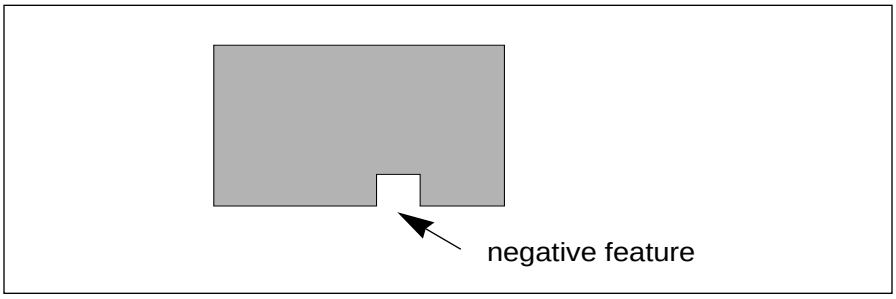


Figure 2-25: *A negative mask feature.*

2.5.4 Typical Process Parameters

Typical values of process parameters for a 0.5 μm fabrication process are given in Table 2-4. We use the term typical loosely here; these are estimated values which do not reflect a particular manufacturing process and the actual parameter values can vary widely. You should always request process parameters from your vendor when designing a circuit that you intend to fabricate.

n-type transconductance	k'_n	$73\mu A/V^2$
p-type transconductance	k'_p	$-21\mu A/V^2$
n-type threshold voltage	V_{tn}	0.7V
p-type threshold voltage	V_{tp}	–0.8V
gate capacitance	C_g	$0.9fF/\mu m^2$
n-diffusion bottomwall capacitance	$C_{ndiff,bot}$	$0.6fF/\mu m^2$
n-diffusion sidewall capacitance	$C_{ndiff,side}$	$0.2fF/\mu m$
p-diffusion bottomwall capacitance	$C_{pdiff,bot}$	$0.9fF/\mu m^2$
p-diffusion sidewall capacitance	$C_{pdiff,side}$	$0.3fF/\mu m$
n-type source/drain resistivity	R_{ndiff}	$2\Omega/\square$
p-type source/drain resistivity	R_{pdiff}	$2\Omega/\square$
poly-substrate plate capacitance	$C_{poly,plate}$	$0.09fF/\mu m^2$
poly-substrate fringe capacitance	$C_{poly,fringe}$	$0.04fF/\mu m$
poly resistivity	R_{poly}	$4\Omega/\square$
metal 1-substrate plate capacitance	$C_{metal1,plate}$	$0.04fF/\mu m^2$
metal 1-substrate fringe capacitance	$C_{metal1,fringe}$	$0.09fF/\mu m$
metal 2-substrate capacitance	$C_{metal2,plate}$	$0.02fF/\mu m^2$
metal 2-substrate fringe capacitance	$C_{metal2,fringe}$	$0.06fF/\mu m$
metal 3-substrate capacitance	$C_{metal3,plate}$	$0.009fF/\mu m^2$
metal 3-substrate fringe capacitance	$C_{metal3,fringe}$	$0.02fF/\mu m$
metal 1 resistivity	R_{metal1}	$0.08\Omega/\square$
metal 2 resistivity	R_{metal2}	$0.07\Omega/\square$
metal 3 resistivity	R_{metal3}	$0.03\Omega/\square$
metal current limit	$I_{m,max}$	$1.5mA/\mu m$

Table 2-4 *Typical parameters for our 0.5 μm process.*

2.6 Layout Design and Tools

2.6.1 Layouts for Circuits

We ultimately want to design layouts for circuits. Layout design requires not only a knowledge of the components and rules of layout, but also strategies for designing layouts which fit together with other circuits and which have good electrical properties.

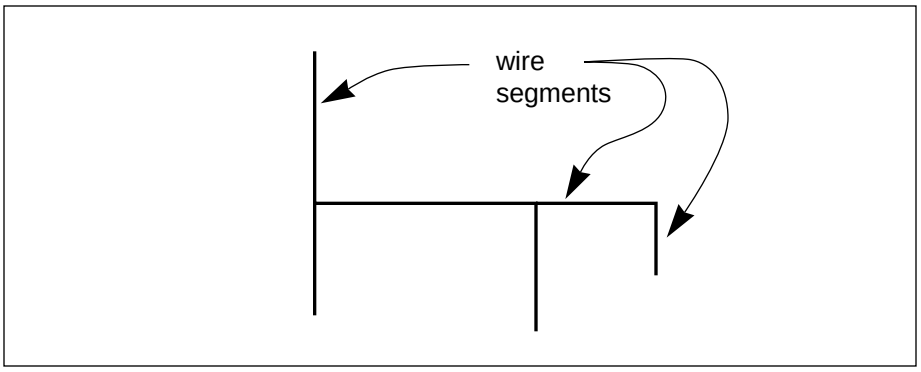


Figure 2-26: *Wires and wire segments.*

Since layouts have more physical structure than schematics, we need to augment our terminology. Chapter 1 introduced the term *net* to describe a set of electrical connections; a net corresponds to a variable in the voltage equations, but since it may connect many pins, it is hard to draw. A **wire** is a set of point-to-point connections; as shown in Figure 2-26, a wire may contain many branches. The straight sections are called **wire segments**.

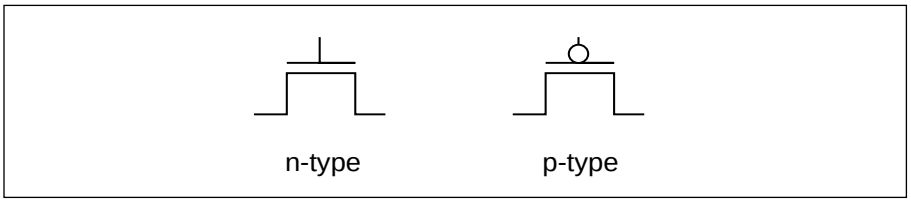


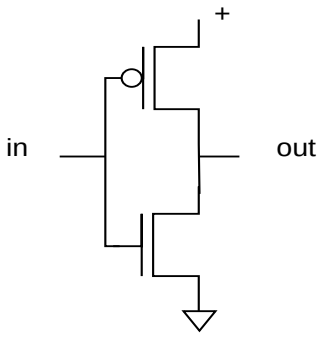
Figure 2-27: *Schematic symbols for transistors.*

The starting point for layout is a **circuit schematic**. The schematic symbols for n- and p-type transistors are shown in Figure 2-27. The schematic shows all electrical connections between transistors (except for tub ties, which are often omitted to simplify the diagram); it must also be annotated with the W/L of each transistor. We will discuss the design of logic circuits from transistors in detail in Chapter 3. At this point, we will treat the circuit schematic

as a specification for which we must implement the transistors and connections in layout. (Most professional layout designers, in fact, have no training in electrical engineering and treat layout design strictly as an artwork design problem.) The next example walks through the design of an inverter's layout.

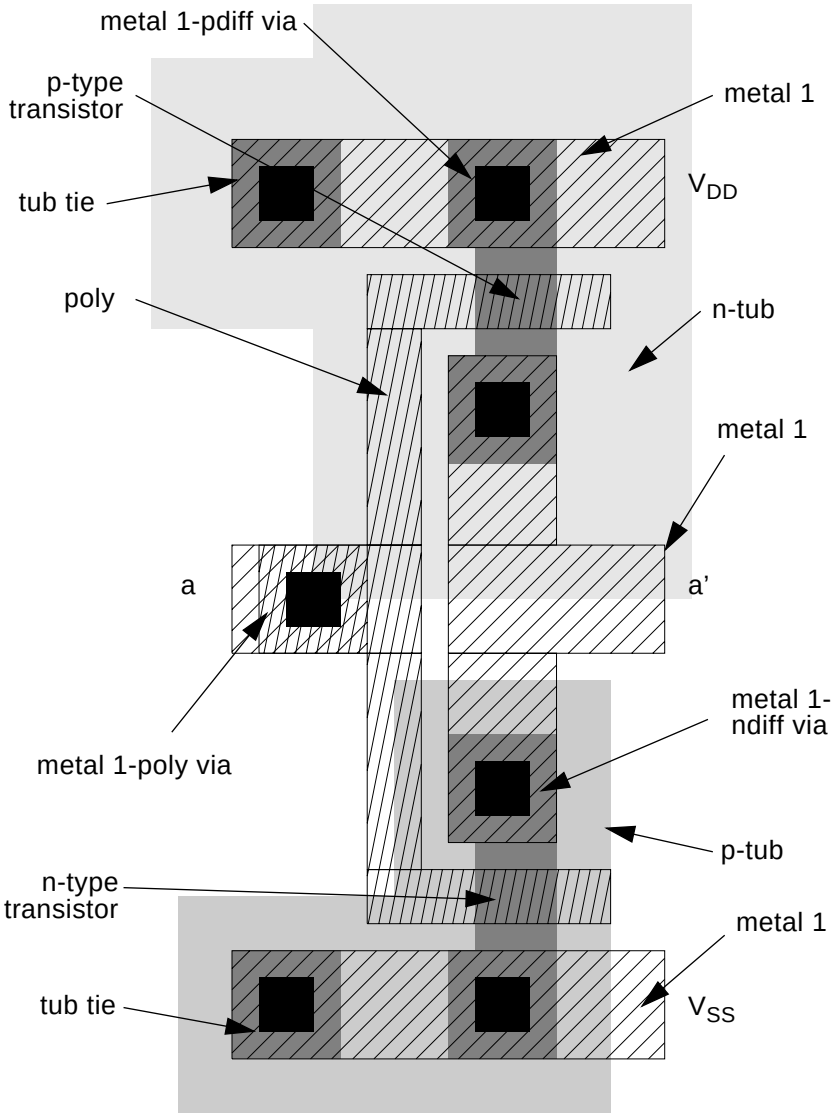
Example 2-5: Design of an inverter layout

The inverter circuit is simple (+ is V_{DD} and the triangle is V_{SS}):



In thinking about how the layout will look, a few problems become clear. First, we cannot directly connect the p-type and n-type transistors with pdiff and ndiff wires. We must use vias to go from ndiff to metal and then to pdiff. Second, the *in* signal is naturally in polysilicon, but the *out* signal is naturally in metal, since we must use a metal strap to connect the transistors' source and drain. Third, we must use metal for the power and ground connections. We probably want to place several layouts side-by-side, so we will run the power/ground signals from left to right across the layout.

Assuming that both transistors are minimum size, here is one layout for the inverter:



We chose to put a metal-poly via at the inverter's input so the signal would be on the same layer at input and output; we might want to connect the output of one inverter directly to the input of another. We ran power and ground along the top and bottom of the cell, respectively, placing the p-type transis-

tor in the top half and the n-type in the bottom half. Larger layouts with many transistors follow this basic convention: p-type on the top, n-type on the bottom. The large tub spacing required between p-type and n-type devices makes it difficult to mix them more imaginatively. We also included a tub tie for both the n-tub and p-tub.

2.6.2 Stick Diagrams

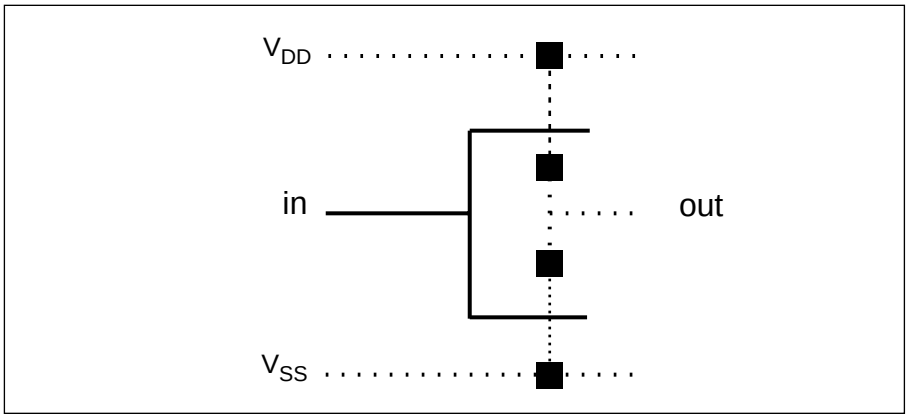


***Figure 2-28**: A stick diagram for an inverter.*

We must design a complete layout at some point, but designing a complex system directly in terms of rectangles can be overwhelming. We need an abstraction between the traditional transistor schematic and the full layout to help us organize the layout design. A **stick diagram** is a cartoon of a chip layout. Figure 2-28 shows a stick diagram for an inverter. The stick diagram represents the rectangles with lines which represent wires and component symbols. While the stick diagram does not represent all the details of a layout, it makes some relationships much clearer and it is simpler to draw.

Layouts are constructed from rectangles, but stick diagrams are built from cartoon symbols for components and wires. The symbols for wires used on various layers are shown in Figure 2-29. You probably want to draw your own stick diagrams in color: red for poly, green for n-diffusion, yellow for p-diffusion, and shades of blue for metal are typical colors. A few simple rules for constructing wires from straight-line segments ensure that the stick diagram corresponds to a feasible layout. First, wires cannot be drawn at arbitrary angles—only horizontal and vertical wire segments are allowed. Second, two wire segments on the same layer which cross are electrically connected. Vias to connect wires that do not normally interact are drawn as black

dots. Figure 2-30 shows the stick figures for transistors—each type of transistor is represented as poly and diffusion crossings, much as in the layout.

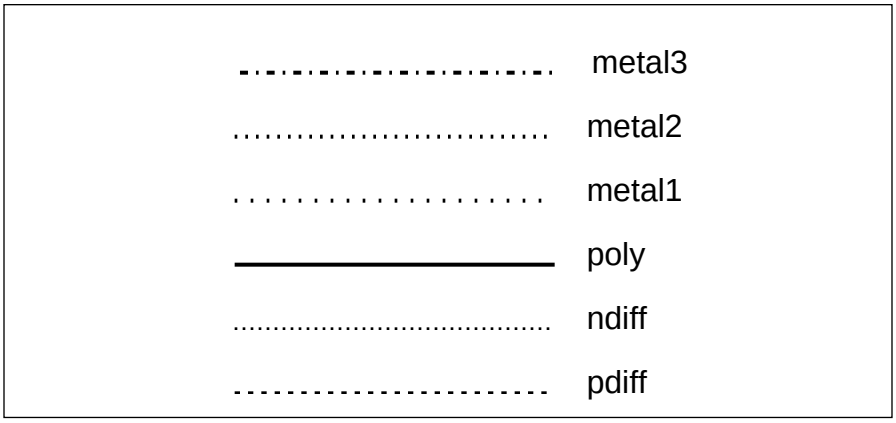


Figure 2-29: *Stick diagram symbols for wires.*

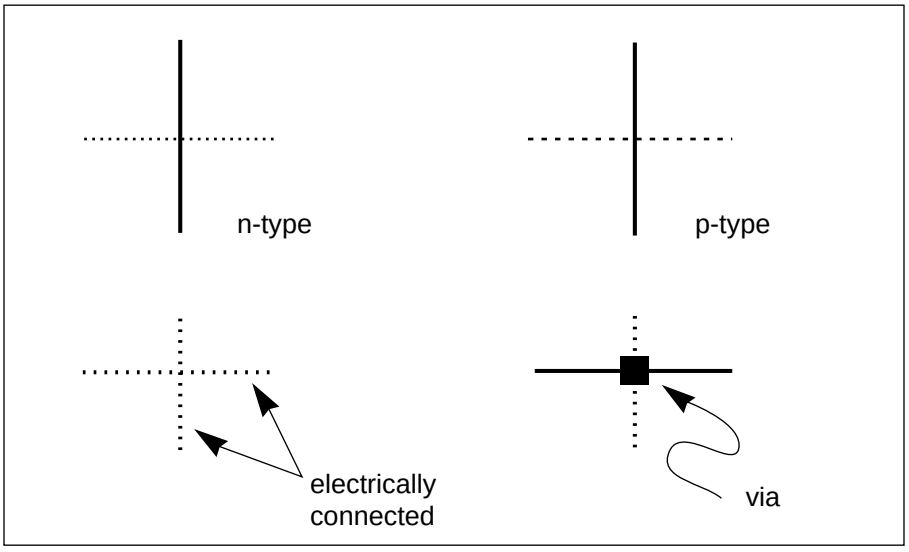


Figure 2-30: *Symbols for components in stick diagrams.*

The complete rules which govern how wires on different layers interact are shown in Table 2-5; they tell whether two wires on given layers are allowed to cross and, if so, the electrical properties of the new construct. This table is derived from the manufacturing design rules.

Stick diagrams are not exact models of layouts. Most of the differences are caused by the use of zero-width lines and zero-area transistors in stick diagrams. When you draw a layout using a stick diagram as a guide, you may find it necessary to move transistors and vias and to reroute wires. Area and

metal3	metal2	metal1	poly	ndiff	pdiff	
short	open	open	open	open	open	metal3
	short	open	open	open	open	metal2
		short	open	open	open	metal1
			short	n-type	p-type	poly
				short	illegal	ndiff
					short	pdiff

Table 2-5 *Rules for possible interactions between layers.*

aspect ratio are also difficult to estimate from stick diagrams. But a stick diagram can be drawn much faster than a full-fledged layout and lets you evaluate a candidate design with relatively little effort. Stick diagrams are especially important tools for layouts built from large cells and for testing the connections between cells—tangled wiring within and between cells quickly becomes apparent when you sketch the stick diagram of a cell.

2.6.3 Hierarchical Stick Diagrams

Drawing a large chip as a single stick diagram—covering a huge sheet of butcher paper with arcane symbols—usually leads to a spaghetti layout. We can make use of hierarchy to organize stick diagrams and layouts just as with schematics. Components in a layout or hierarchical stick diagram are traditionally called **cells**. In schematics, we either invent a symbol for a type (*e.g.*, logic gate symbols) or we use a box; however, the shape of the component symbol has no physical significance. Layouts and stick diagrams have physical extent. The simplest representation for a cell is its **bounding box**: a rectangle which just encloses all the elements of the cell. Bounding boxes are easy to generate; some layout tools require that cells be represented by rectangular bounding boxes. However, in some cases, we use non-rectangular **cell boundaries** to represent cells with very non-rectangular shapes.

Figure 2-31 shows a hierarchical stick diagram built from two copies of an inverter cell. The top-level cell in the hierarchy, *pair1*, includes some wires used to connect the cells together and to make external connections. Note that *pair1*'s wiring implies that the *inv1* stick diagram has been redesigned so that, unlike the stick diagram of Figure 2-28, its input and output are both on the

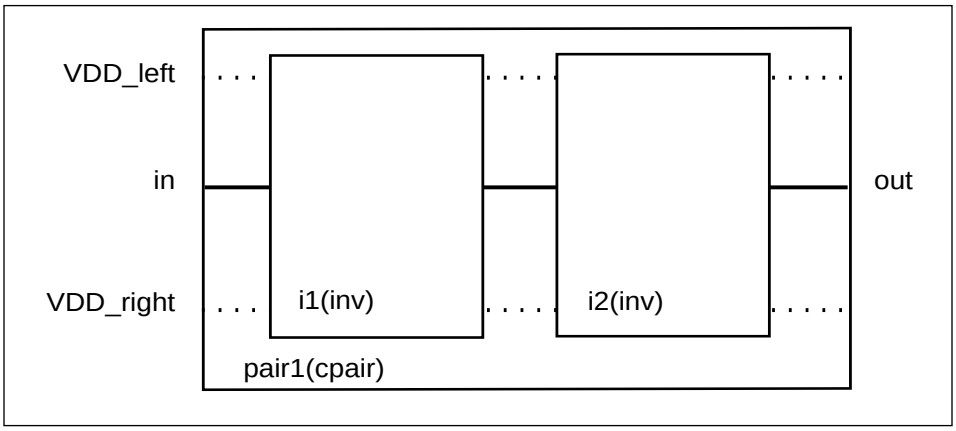
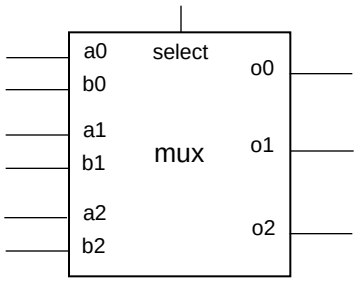


Figure 2-31: *A hierarchical stick diagram.*

polysilicon layer. We sometimes want to show sticks cells in their entirety, and sometimes as outlines—some relationships between cells are apparent only when detail within a cell is suppressed. Hierarchical design is particularly useful in layout and sticks design because we can reuse sections of layout. Many circuits are designed by repeating the same elements over and over. Repeating cells saves work and makes it easier to correct mistakes in the design of cells.

Example 2-6: Sticks design of a multiplexer

A more interesting example of a stick diagram which takes advantage of hierarchy is a multiplexer (also known as a mux):

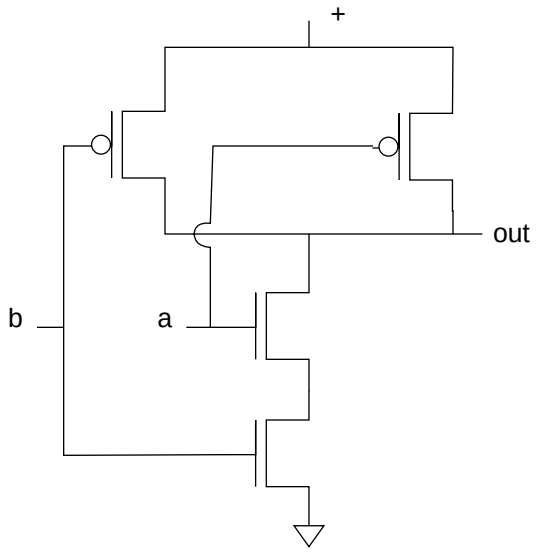


A two-input, *n*-bit multiplexer (in this case, *n* = 3) has two *n*-bit data inputs and a select input, along with an *n*-bit data output. When select = 0, the data

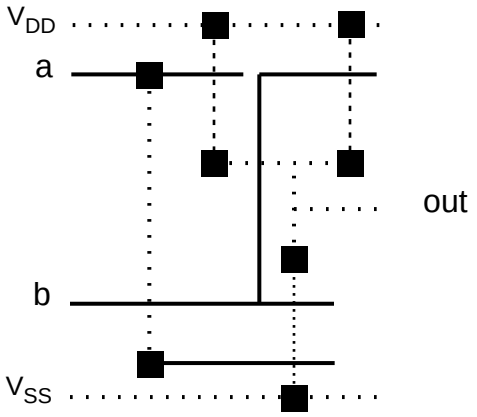
output's value is equal to the a data input's value; if select = 1, the data output's value is equal to b.

The multiplexer can be designed as a one-bit slice which is replicated to create an *n*-bit system. The Boolean logic formula which determines the output value of one bit is $o_i = (a_i\ select) + (b_i\ select')$; the value of o_i depends only on a_i, b_i, and *select*. We can rewrite this formula in terms of two-input NAND gates: $o_i = NAND(NAND(a_i, select), NAND(b_i, select'))$. Since we know how to design the stick diagram for a NAND gate, we can easily design the one-bit multiplexer out of NAND cells.

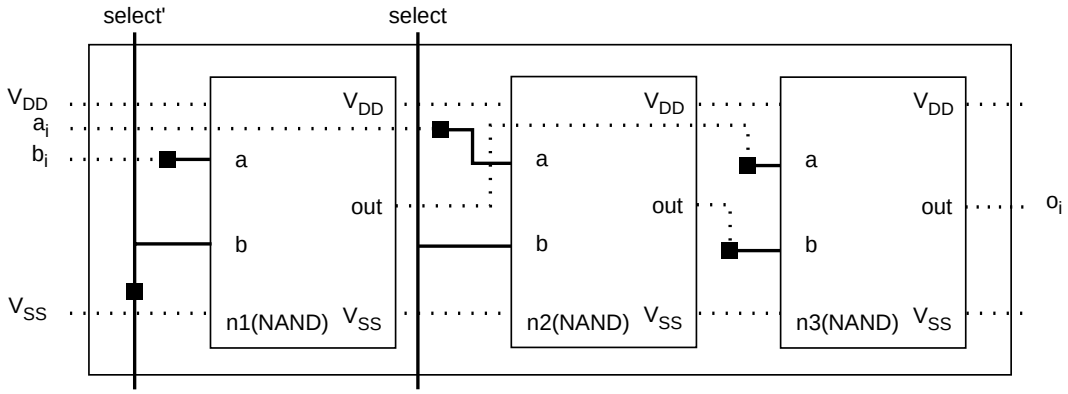
Here is the transistor schematic for a two-input NAND gate:



And here is a stick diagram for the two-input NAND:

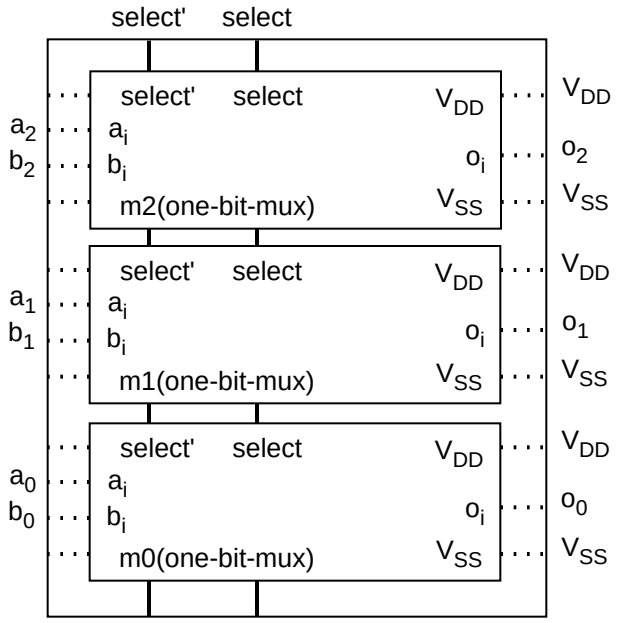


We can use the NAND cell to build a one-bit multiplexer cell:



In this case we've drawn the hierarchical stick diagram using bounding boxes; to design the complete layout we would have to look into the cells. The connections designed between NAND cells were designed to avoid creating unwanted shorts with wires inside the NANDs; to be completely sure the intercell wires do not create problems, you must expand the view of the bit slice to include the internals of the NAND cells. However, making an initial wiring design using the NANDs as boxes, remembering the details of their internals as you work, makes it easier to see the relationships between wires that go between the cells.

We can build a three-bit multiplexer from our bit slice by stacking three instances of the slice cell along with a few wires:



The select signal was designed to run vertically through the cell so vertical connections could easily be made between stacked cells. The multiplexer inputs arrive at the left edge of the stack, while the multiplexer's outputs leave at the right edge.

Constructing this three-bit multiplexer required very little labor—given a NAND cell, we were able to construct the bit slice with only a few extra wires; and given the bit slice building the complete multiplexer was almost trivial. Changing *n*, the width of the data word, is very simple. And last but not least, building large stick diagrams out of previously-designed smaller cells means the complete design is more likely to be correct: cells we have used before are likely to have been previously checked, and repeating cells gives us fewer opportunities to make simple mistakes while copying simple constructs.

2.6.4 Layout Design and Analysis Tools

A variety of CAD tools help us design and verify layouts. The most important tools are **layout editors**, **design rule checkers**, and **circuit extractors**.

A layout editor is an interactive graphic program that lets you create and delete layout elements. Most layout editors work on hierarchical layouts, organizing the layout into cells which may include both primitive layout elements and other cells. Some layout editing programs, such as Magic, work on **symbolic layouts**, which include somewhat more detail than do stick diagrams but are still more abstract than pure layouts. A via, for example, may be represented as a single rectangle while you edit the symbolic layout; when a final physical layout is requested, the symbolic via is fleshed out into all the rectangles required for your process. Symbolic layout has several advantages: the layout is easier to specify because it is composed of fewer elements; the layout editor ensures that the layouts for the symbolic elements are properly constructed; and the same symbolic layout can be used to generate several variations, such as n-tub, p-tub, and twin-tub versions of a symbolic design.

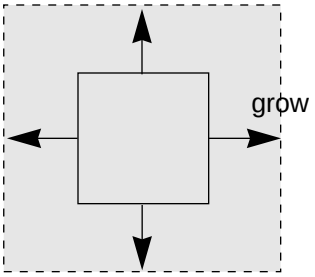
A design rule checker (often called a **DRC** program), as the name implies, looks for design rule violations in the layout. It checks for minimum spacing and minimum size and ensures that combinations of layers form legal components. The results of the DRC are usually shown as highlights. on top of the layout. Some layout editors, including Magic, provide on-line design rule checking. Design rule checking algorithms are briefly described in Chapter 10.

Circuit extraction is an extension of design rule checking and uses similar algorithms. A design rule checker must identify transistors and vias to ensure proper checks—otherwise, it might highlight a transistor as a poly-diffusion spacing error. A circuit extractor performs a complete job of component and wire extraction. It produces a net list which lists the transistors in the layout and the electrical nets which connect their terminals. Vias do not appear in the net list—a via simply merges two nets into a single larger net. The circuit extractor usually measures parasitic resistance and capacitance on the wires and annotates the net list with those parasitic values. The next example describes how we can extract a circuit from a layout.

Example 2-7: Circuit extraction

We will extract the circuit by successively identifying, then deleting components. After all component types have been extracted, only the wires will remain.

Identifying components from the layout requires manipulating masks singly and in combination. **Grow** and **shrink** are two important operations:



The grow operation increases the extent of each polygon in the mask by a fixed amount in every direction; the shrink operation does the converse. We will also need to form Boolean combinations of masks: the NOT of a mask covers all the area not covered by the mask itself; the AND of two masks covers only the area under both masks; and the OR includes the area covered by either mask. Boolean and grow / shrink operations generate new masks.

When we extract the circuit, we will assume the layout has no design-rule errors; we can always DRC the layout before extraction. We can identify all the transistors in the layout very easily: the n-type transistors' active areas are exactly the AND of the poly and the n-diff masks, with analogous definition for the p-type transistors. After identifying the transistors, we can remove them from the layout of the active-area mask, which leaves the gate, source, and drain connections hanging. We will mark and remember the locations of the transistors' terminals for the final step of extraction.

Identifying vias requires a little more effort. To identify poly-metal1 vias, we first grow the cut mask by 2λ, then we form the AND of the grown-cut, metal, and poly masks. The result is one 4λ-by-4λ square for each poly-metal1 via. After identifying all the vias, we remove them while marking their place. We

can identify tub ties, but we won't need them for the later stages of analysis, since they don't make new electrical connections.

At this point, only the wires are left in the layout. A polygon on one layer forms an electrically connected region. However, we're not quite done, because connections may have been made by vias or by wires through transistors. To take account of all connections, we must first identify where each wire touches a connection point to a via or transistor. We then form the transitive closure of all the connection points: if one wire connects points A and B, and another wire connects B and C, then A, B, and C are all electrically connected.

Once we have traced through all the connections, we have a basic circuit description. We have not yet taken parasitics into account. To do so, we must count parasitics for each wire, via, and transistor, then mark each electrical node appropriately. However, for simple functional analysis, extracting parasitics may not be necessary. Here is a fragment of an extracted circuit written in Magic's *ext* format:

```
node "6_38_29#" 122 55 19 -14 green 0 0 0 0 54 34 0 0 92 62 0 0 0 0 0 0
node "6_50_15#" 120 10 25 -7 green 0 0 0 0 12 16 0 0 0 0 0 0 0 0 0 0
node "6_50_7#" 521 92 25 -3 green 0 0 60 44 30 22 0 0 80 64 0 0 0 0 0 0
node "6_36_19#" 825 12 18 -9 p 110 114 0 0 0 0 0 0 0 0 0 0 0 0 0 0
node "6_36_11#" 690 9 18 -5 p 92 96 0 0 0 0 0 0 0 0 0 0 0 0 0 0
node "6_40_40#" 559 83 20 20 brown 0 0 80 54 0 0 0 0 68 58 0 0 0 0 0 0
cap "6_36_19#" "6_50_7#" 1
fet nfet 25 -9 26 -8 12 16 "GND!" "6_36_19#" 4 0 "6_38_29#" 6 0 "6_50_15#" 6 0
fet nfet 25 -5 26 -4 12 16 "GND!" "6_36_11#" 4 0 "6_50_15#" 6 0 "6_50_7#" 6 0
fet pfet 39 17 40 18 12 16 "Vdd!" "6_36_19#" 4 0 "6_50_7#" 6 0 "6_40_40#" 6 0
fet pfet 25 17 26 18 12 16 "Vdd!" "6_36_11#" 4 0 "6_50_7#" 6 0 "6_40_40#" 6 0
```

The exact format of this file isn't important, but a few details should help make this information less forbidding. A `node` record defines an electrical node in the circuit—explicit declaration of the nodes simplifies the program which reads the file. The record gives total resistance and capacitance for the node, an *x*, *y* position which can be used to identify the node in the layout, and area and perimeter information for resistance extraction. A `cap` record gives two nodes and the capacitance between them. A `fet` record describes the type of transistor, the corners of its channel, and the electrical nodes to which the source, drain, and gate are connected.

The simplest extraction algorithm works on a layout without cells—this is often called flat circuit extraction because the component hierarchy is flattened to a single level before extraction. However, a flattened layout is very large: a layout built of one 100-rectangle cell repeated 100 times will have 100 rectangles plus 100 (small) cell records; the same layout flattened to a single cell will have 10,000 rectangles. That added size claims penalties in disk storage, main memory, and CPU time.

Hierarchical circuit extraction extracts circuits directly on the hierarchical layout description. Dealing with cell hierarchies requires more sophisticated algorithms which are beyond our scope. Hierarchical extraction may also require design restrictions, such as eliminating overlaps between cells. However, one problem which must be solved illustrates the kinds of problems introduced by component hierarchies.

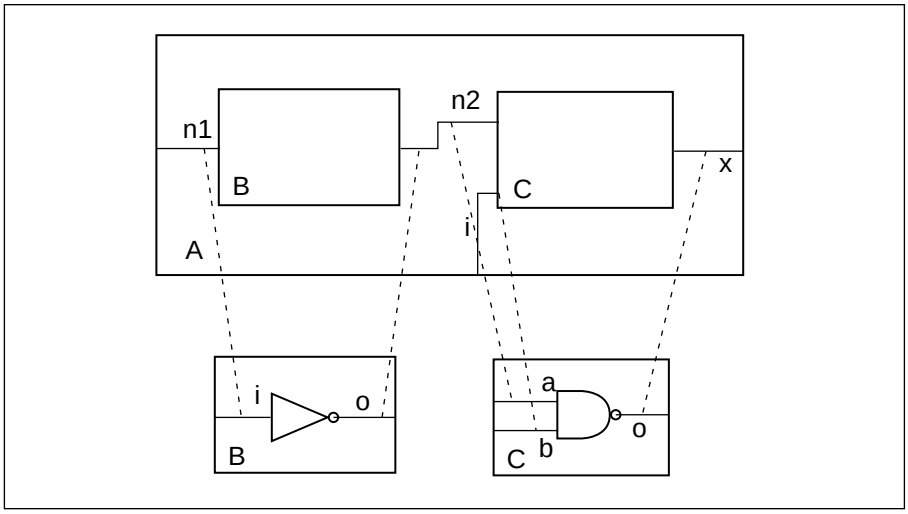


***Figure 2-32**: Tracing nets for hierarchical circuit extraction.*

Consider the example of Figure 2-32. Each cell has its own net list. The net lists of leaf cells make sense on their own, but A's net list is written in terms of its components. We often want to generate a flattened net list—flattening the net list after extraction makes sense because the net list is much smaller than the layout. To create the flattened net list, we must make correspondences between nets in the cells and nets in the top-level component. Once again, we use transitive closure: if net o in cell B is connected to n2 in A, which in turn is connected to net a in C, then B.o, A.n2, and C.a are all connected. Flattening algorithms can be very annoying if they choose the wrong names for combined elements. In this case, n2, the top-level component's name for the net, is probably the name most recognizable to the designer.

An extracted layout has two important uses. First, the extracted circuit can be simulated and the results compared to the specified circuit design. Serious layout errors, such as a missing transistor or wire, should show up as a difference in the specified and extracted circuits. Second, extracted parasitics can be used to calculate actual delays. Circuit performance may have been estimated using standard parasitic values or parasitics may have been ignored entirely, but long wires can slow down logic gates. Comparing the actual performance of the extracted layout to the predicted performance tells you whether the logic and circuits need to be modified and, if so, where critical delay problems exist.

2.6.5 Automatic Layout

Hierarchical stick diagrams are a good way to design large custom cells. But you will probably design large cells from scratch infrequently. You are much more likely to use layouts generated by one of two automated methods: **cell generators** (also known **macrocell generators**), which create optimized layouts for specialized functions such as ALUs; or **standard cell placement and routing**, which use algorithms to build layouts from gate-level cells.

A cell generator is a parameterized layout—it is a program written by a person to generate the layout for a particular cell or a families of cells. The generator program is usually written textually, though some graphical layout editors provide commands to create parameterized layouts. If the generator creates only one layout, it may as well have been created with a graphical layout editor. But designers often want to create variations on a basic cell: changing the sizes of transistors, choosing the number of busses which run through a cell, perhaps adding simple logic functions. Specialized functions like ALUs, register files, and RAMs often require careful layout and circuit design to operate at high speed. Generator languages let skilled designers create parameterized layouts for such cells which can be used by chip designers whose expertise is in system design, not circuit and layout design.

Figure 2-33 shows a fragment of a layout generator written in the CLL language [Sax83]. The `wire` command works like a pen—it lays down material on a layer as it moves from point to point. Movement may be specified by absolute coordinate or by relative movement; a layer switch instantiates a via at the point of the switch. This small example doesn't show how to build layout procedures, which can take parameters much like procedures in programming languages. Parameters passed in at run time can control whether layout elements are placed at all and where they are placed.

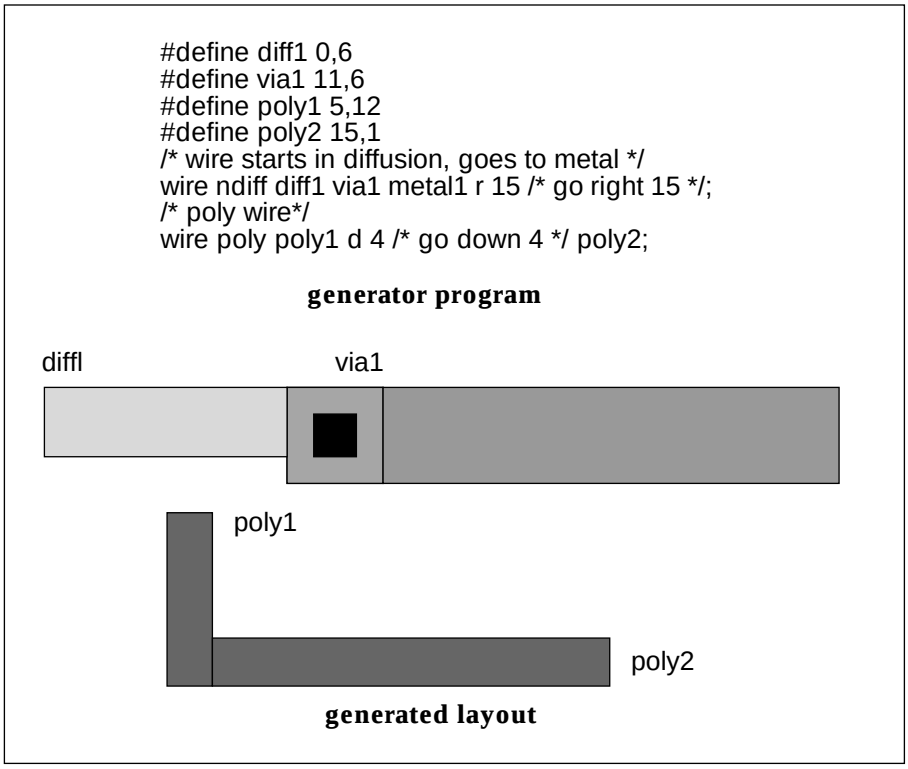


Figure 2-33: *A layout generation program.*

Place-and-route programs take a very different approach to layout synthesis: they break the problem into placing components on the plane, then routing wires to make the necessary connections. Placement and routing algorithms may not be able to match the quality of hand-designed layouts for some specialized functions, but they often do better than people on large random logic blocks because they have greater patience to search through large, unstructured problems to find good solutions.

The most common placement-and-routing systems use **standard cells**, which are logic gates, latches, flip-flops, or occasionally slightly larger functions like full adders. Figure 2-34 shows the architecture of a standard cell layout: the component cells, which are of standard height but of varying width, are arranged in rows; wires are run in **routing channels** between the cell rows, along the sides, and occasionally through **feedthroughs** (spaces left open for wires in the component cells). The layout is designed in two stages: components are placed using approximations to estimate the amount of wire required to make the connections; then the wires are routed. We will discuss

standard cell layout synthesis in more detail in Chapter 6. Figure 2-35 shows a small standard cell layout generated by the *wolfe* program [San84, Sec85].

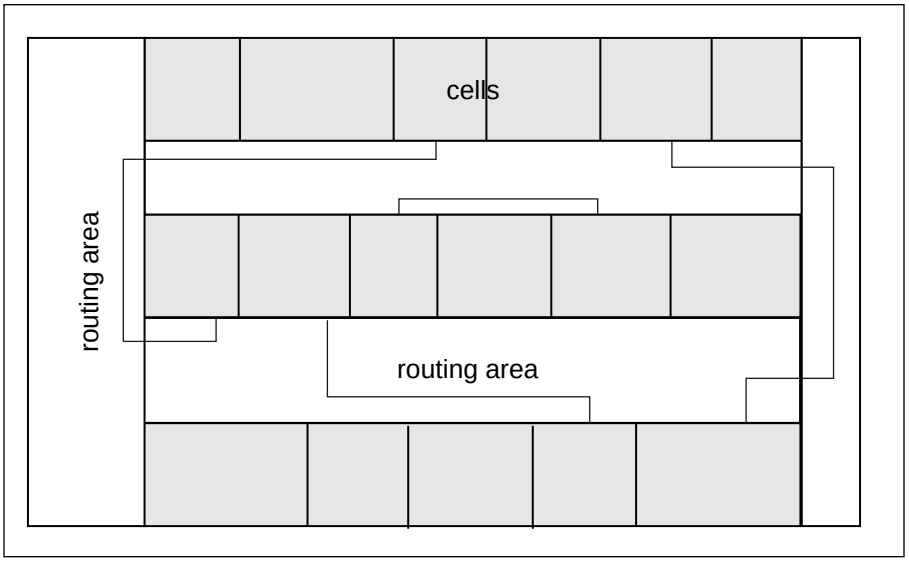


Figure 2-34: *Architecture of a standard cell layout.*

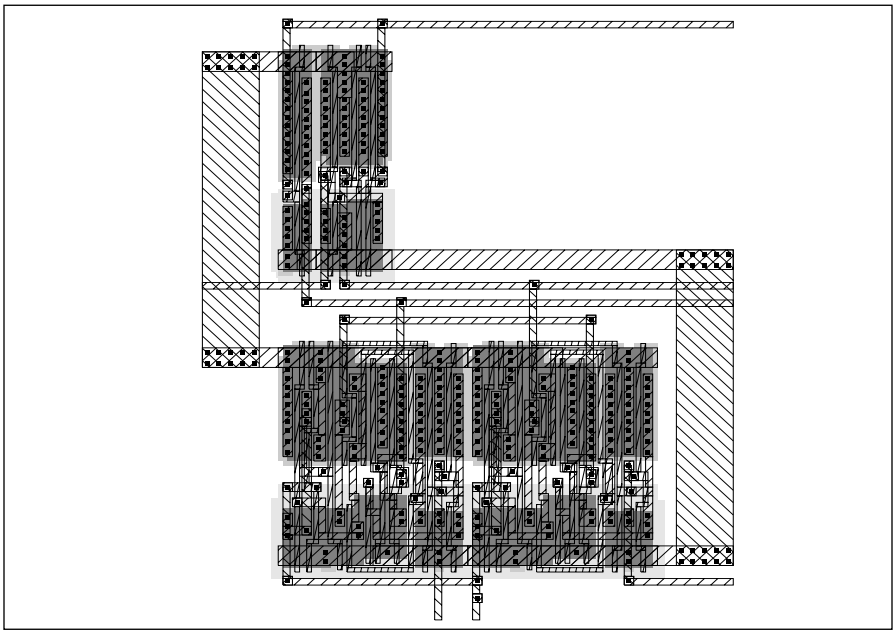


Figure 2-35: *An example of standard cell layout.*

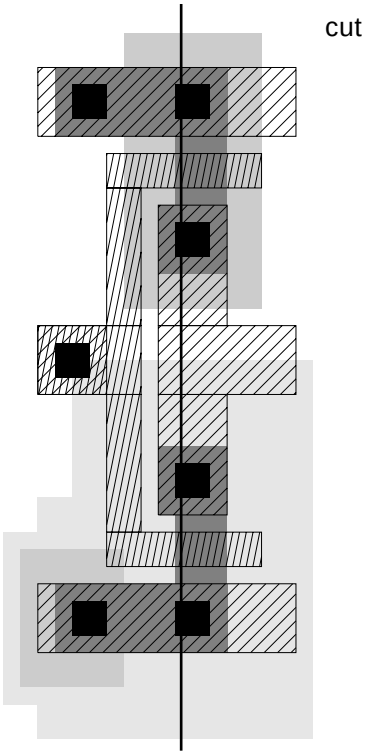
2.7 References

As mentioned in the last chapter, Dennard *et al.* [Den74] first explained why shrinking IC line widths led to higher performance as well as smaller chips. That observation led to the development of scalable design rules, which were first introduced by Mead and Conway [Mea80]. The specifications for the MOSIS SCMOS process were derived from MOSIS data. Complete documentation on the SCMOS rules is available on the World Wide Web at http://www.isi.edu/mosis or by sending electronic mail to mosis@mosis.edu. The MOSIS SCMOS rules do occasionally change, so it is always best to consult MOSIS for the latest design rules before starting a design.

2.8 Problems

Use process parameters from Table 2-4 as required.

2-1. Draw the cross-section of the inverter shown below along a cut through the middle of the p-type and n-type transistors.



2-2. Assuming that V_{gs} = 3.3V, compute the drain current through n-type transistors of these sizes at V_{ds} values of 1V, 2V, 3.3V, and 5V:

a) W / L = 5 / 2.

b) W / L = 8 / 2.

c) W / L =12 / 2.

d) W / L = 25 / 2.

2-3. Using the parameter values given in Example 2-2:

a) How much would you have to change the gate oxide thickness to reduce the threshold voltage by 0.1 V?

b) How much would you have to change the doping N_A to reduce the threshold voltage by 0.1 V?

2-4. Plot I_d vs. V_{gs} at V_{ds} = 5V for two values of the channel length modulation parameter: $\lambda = 0$ and $\lambda = 0.03$.

2-5. Evaluate the effects of leakage currents:

a) First, compute the gate capacitance of a minimum-size transistor (ignoring source / drain overlap capacitances).

b) Assuming that I_{l0} = 0.1 nA, how long will it take leakage currents to drain half the charge of that gate capacitance, assuming that it starts at 5 V?

c) How long will it take for leakage currents to drain the capacitor assuming it starts at 3.3 V?

2-6. Justify each of these design rules:

a) 2λ poly-poly spacing;

b) no required poly-metal spacing;

c) 1λ of diffusion and metal surrounding cut;

d) 2λ overhang of poly at transistor gate.

2-7. Explain:

a) Why is ndiff-to-pdiff spacing so large?

b) Why is metal-metal spacing larger than poly-poly spacing?

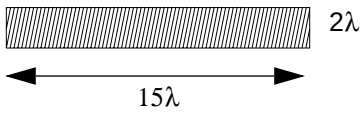
c) Why is metal2-metal2 spacing larger than metal1-metal1 spacing?

2-8. What distinguishes a tub tie from an ndiff-metal1 via?

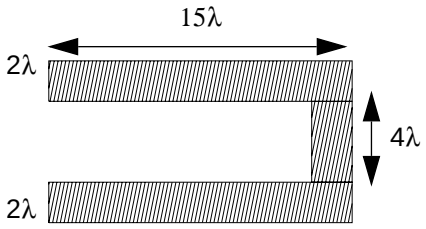
2-9. Design the layout for an ndiff wire connected to a pdiff wire.

2-10. Compute the resistance and capacitance for each polysilicon wire below:

a)



b)



2-11. Compute the parasitic resistance and capacitance of the source / drain region of transistors of the types and sizes given below, assuming that these source drain-regions have the required 2 λ overhang past the gate:

a) p-type; W / L = 3 / 2;

b) n-type, W / L = 4 / 2;

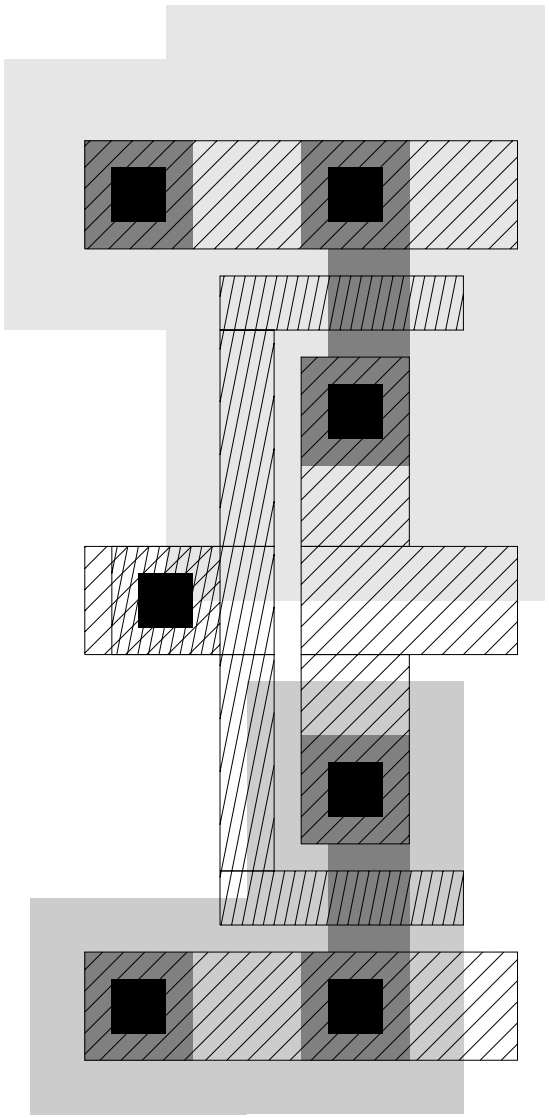
c) p-type; W / L = 6 / 2;

d) n-type; W / L = 12 / 2.

2-12. Recompute the parasitic resistances and capacitances of the transistors of Question 2-11. Assume in this case that the source / drain regions have the required overhang but also include as many diffusion / metal 1 vias as will fit in the given gate width. Measure the total resistance including the via resistance.

2-13. Measure the parasitic resistance and capacitance on the input to the inverter shown on the next page. The transistors have W / L = 3 λ / 2 λ and the V_{DD} and V_{SS} lines are each 4 λ wide. Give the total resistance and capaci-

tance of all the material connected to the inverter's input, showing how you broke the layout elements into sections for measurement.

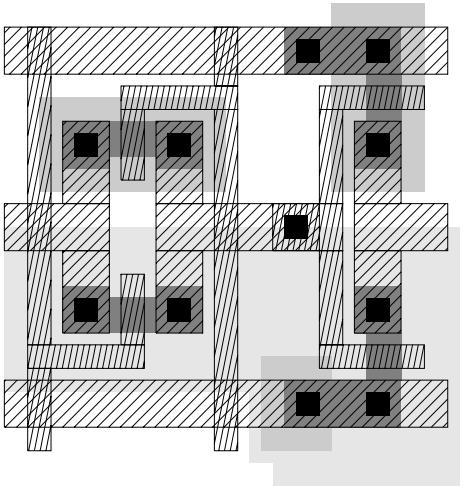


2-14. Reverse-engineer the layout shown on the next page. Draw:

a) a stick diagram corresponding to the layout;

b) a transistor schematic.

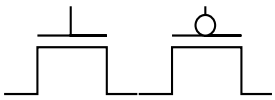
Label inputs and outputs in your drawings in accordance with the labels in the layout.



2-15. Design a layout for the circuit fragment below in two stages:

a) a stick diagram;

b) the complete layout.

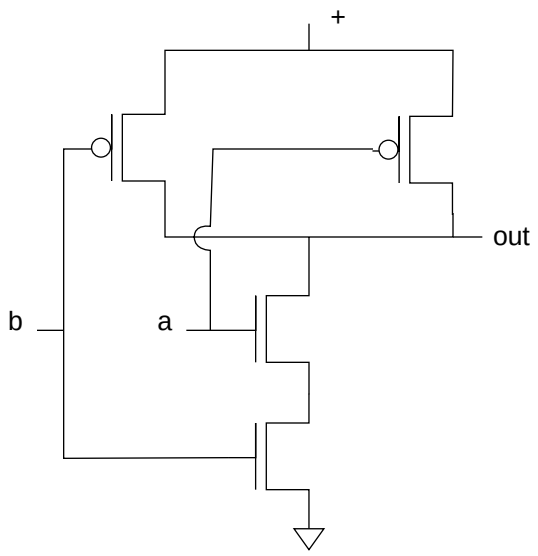


2-16. Design two different stick diagrams for the inverter circuit of Example 2-5. Your only constraint is that both V_{DD} and V_{SS} must be available on two opposite sides of the cell.

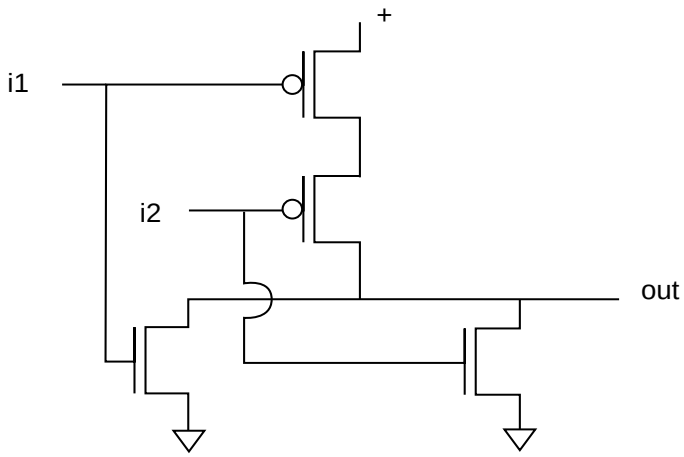
2-17. Expand the stick diagram for the one-bit-mux cell of Example 2-6 to include the wires in the NAND cells.

2-18. Redesign the three-bit multiplexer of Example 2-6 so that adjacent one-bit-mux cells share V_{DD} and V_{SS} lines. You can overlap cells to force sharing of wires.

2-19. Design two different stick diagrams for this circuit, such that both V_{DD} and V_{SS} are available on two opposite sides of the cell:

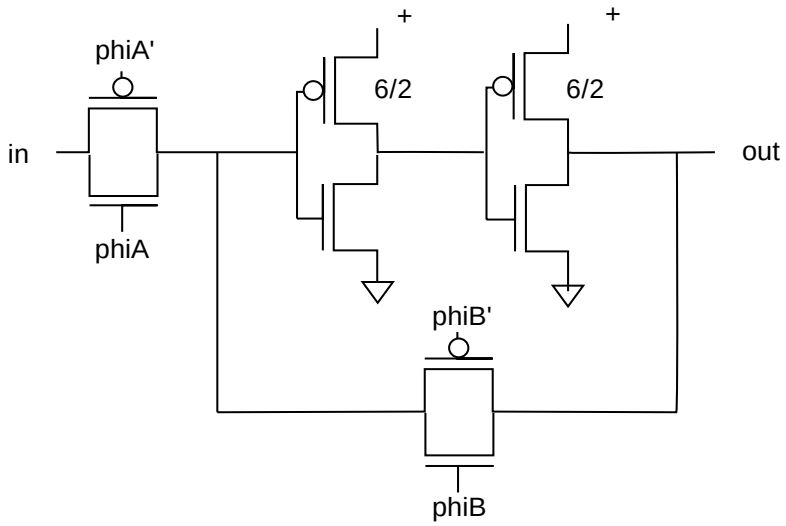


2-20. Design a layout for the circuit shown below in two stages: first draw a stick diagram, then design the complete layout. Assume that all transistors are 3 λ / 2 λ. The power supply lines should run through the cell in metal 1, available on opposite sides of the cell.



2-21. Design a stick diagram for the circuit shown below, assuming that transistor sizes have W/L = 3 λ / 2 λ except as indicated. The cell should be designed to be area-efficient and to be horizontally tilable—you should be

able to place two of your cells side-by-side and make all electrical connections without adding any wires.



2-22. Design a layout for the stick diagram you designed in Question 2-21. Show that the cell tiles horizontally.