A Caveat from the Editors

This chapter covers the most time-dependent of all the topics in these lectures — the advances in silicon technology over the past decade have been truly startling. Nonetheless, we believe it worthwhile to include Feynman's overview of the status of the subject in the early 1980's — despite the fact that some of the technological goalsposts have moved considerably since Feynman looked at the subject. In particular, the mid 1980's saw the widespread adoption of CMOS technology and Feynman's discussion of devices in terms of nMOS technology now looks somewhat dated; we have therefore edited out a few of his more complex nMOS examples. Nonetheless, his brief discussion of CMOS devices does concentrate on their favorable energetics and savings in power compared to nMOS. Feynman's discussion of design rules is restricted to single metal layer nMOS — as specified by Mead and Conway in their classic 1980 book on VLSI systems. Rather than attempt to update the material to a CMOS context, we have decided to remain faithful to Feynman's original presentation, apart from some minor editorial updates. In this way we hope that Feynman's unique ability to offer valuable physical insight into complex physical processes still comes through. Moreover, it should be remembered that, in actuality, Feynman's lectures were supplemented by lectures from experts from many fields. It is intended to capture this element of Feynman's course in a forthcoming accompanying volume containing state-of-the-art lectures and papers by some of the same experts who contributed to his original courses in the early 1980's. Now read on.

The unifying theme of this course has been what we can and cannot do with computers, and why. We have considered restrictions arising from the organization of the basic elements within machines, the limitations imposed by fundamental mathematics, and even those resulting from the laws of Nature themselves. In this final chapter, we come to address perhaps the most practical of obstacles: the constraints that arise from the technology we employ to actually build our machines — both from the materials we use and from the way in which we arrange the elementary component parts.

Presently, the majority of computers are based on semiconductor technology, which is used to fashion the basic building blocks of machines — devices such as transistors and diodes. VLSI (Very Large Scale Integration), the field of microelectronics dealing with the construction and utilization of silicon chips — and hence of central importance to computing — is a vast subject in itself and we can only scratch the surface here. The reader will certainly find what follows easier to understand if he or she has some knowledge of electronics. However, we hope that our presentation will be intelligible to those with only a passing acquaintance with electricity and magnetism, and we provide several references in the section on suggested reading for the curious to take their interest further.

To begin with, we shall take a look at one simple kind of device, the diode. This is a cunning device which allows current to flow in one direction only. We shall consider the physical phenomena involved in its operation, and how it works in the engineering context of a Field Effect Transistor.

7.1: The Physics of Semiconductor Devices

Our current understanding of the electrical properties of metals and other materials is based on the so-called "Band Theory" of solids. Loosely speaking, this theory predicts that the possible physical states that can be occupied by electrons within a material are arranged into a series of (effectively continuous) strata called "bands", each characterized by a specific range of energies for the allowed electron energy levels within it. These bands arise from the complex interplay of electrons with their parent atoms located within the atomic lattice of the material and are an intrinsically quantum mechanical effect. Electrons in different atomic states occupy different bands. In a general substance, we can identify two essentially distinct types of band relevant to the conduction of electric current: these are the "filled" or "valence" band, and the "conduction" band. States in the filled band correspond to electrons which are bound to their parent atoms, and are effectively confined to a certain region within the material — they are not free to roam around. Electrical conduction occurs when electrons leave their parent atoms and are able to move freely through the conductor. Mobile electrons of this type are said to occupy states within the "conduction band". Typically, there will be a discrete energy gap between the filled and conduction bands. The size of this gap largely determines whether our material
is to be classified as a conductor or an insulator, as we'll see. Let us examine the energy band structure shown in Figure 7.1:

![Electron Energy Diagram](image)

**Fig. 7.1 Band Structure**

As you can see, we have valence and conduction bands separated by an energy gap — in the diagram, the energy associated with the bands increases as we move up vertically. When the lower band is full, the material acts like an insulator: there are no available energy states for electrons to gain energy from the applied electric field and form a current. To support an electrical current, we need electrons in the conduction band where there are plenty of empty states available. To produce such electrons, enough energy must be supplied to occupants of the valence band to help them leap above the gap and make the transition into the conduction zone. This minimum energy is called the "band gap energy" and its value largely determines the electrical properties of a substance, as I've said. Good conductors have a plentiful supply of free electrons under normal conditions, the band gap energy being tiny or non-existent (filled and conduction bands can even overlap). Hence it will not be difficult to excite a current in such a material. Insulators, however, have prohibitively wide gaps (several eV) and only conduct under pretty extreme conditions. There is, however, a third class of material needing consideration, and that is a sort of hybrid of conductors and insulators — the semiconductor — for which the energy gap is relatively small (1 eV or so).

The primary mechanism responsible for getting electrons out of the filled band and into the conduction band is thermal excitation (neglecting the application of external electric fields). This is simply the process whereby the energy changes of random thermal fluctuations are themselves enough to supply the energy required to enable electrons to make a transition. A typical thermal energy might be of the order of 25 meV and if this exceeds the band gap energy, it will be sufficient to cause transitions. This is the case for metals but not for insulators — with their large band gap energies of several eV. For any given material, we can calculate how likely it is for a thermal fluctuation to result in a conduction electron. If the temperature of the substance is $T$, and $E$ is the band gap energy, then the rate at which electrons spontaneously pop up to the higher band is determined by the Boltzmann distribution and is proportional to $\exp(-E/kT)$, where $k$ is Boltzmann's constant. At room temperature ($T=300K$), we have $kT=1.4$ of an electron-volt or 25 meV. Note that, due to the exponential in the formula, this transition rate rises rapidly with temperature. Nonetheless, for most insulators, this rate remains negligible right up to near the melting point.

Let us take a look at a semiconductor. At zero degrees (and low temperatures generally), the semiconductor silicon (henceforth Si) is effectively an insulator. Its band gap is of the order of 1.1 eV and thermal transitions are rare. However, we can certainly excite a current by supplying energy to the valence electrons and when we do, we find something interesting happening, something which is of central importance in our study of semiconductors. When we excite an electron to the conduction band, not only does it become free to run around and give rise to some conductivity, but it leaves behind, in the lower band, a hole. This hole has an effective positive charge and, like the electron in the conduction band, is also able to move about and carry electric current: if a nearby electron fills the space vacated by the thermally excited particle, it will leave a positive charge in its own original location, as if the hole had moved sites. Holes are not "real" free particles — they are just empty spaces in the valence band that behave as if they are particles with positive charge. Holes also appear in insulators but rarely in metals.

There is a special trick that we can perform with Si which modifies its properties so that it is ideal for use in computers. This is the process of doping. Doping involves adding atoms of another substance (an "impurity") into the Si lattice. A common dopant is the element phosphorus (P), which sits next to Si in the Periodic Table. P has a valency of five rather than the four of Si: this means it has five electrons in its outer shell compared to silicon's four. In an ordinary silicon crystal lattice, all four of these valence electrons play a role in holding the atom in place in the lattice and they are not free to move through the crystal — the valence band is fully occupied. When some impurity P atoms are introduced, each impurity atom bonds to four silicon atoms using up four of

---

1 An undoped semiconductor is usually referred to as an *intrinsic* semiconductor. If it is doped, it is *extrinsic*. [Editors]
the five valence shell electrons of the phosphorus. This leaves an extra electron per P atom free to roam through the material and carry a current (Fig. 7.2).

![Fig. 7.2 Liberation of a Phosphorus Electron During Doping](image)

The resulting material is called an "n-type" semiconductor, as there is an excess of negative charge carriers. At modest levels of doping, substances of this sort conduct quite weakly compared to metals; the latter may have one or two free electrons per metal atom whereas an n-type semiconductor has but one electron for each phosphorus atom.

There are very few holes in n-type Si, even when the temperature is high enough to dislodge electrons thermally, because holes in the lower band are filled in by the P electrons preferentially before they fill levels in the conduction band. The venerable "Law of Mass Action", as used for chemical reactions, gives an important relation between electron and hole densities, \( n_e \) and \( n_h \), respectively, and one which, interestingly enough, is actually independent of the fraction of dopant in the material:

\[
\frac{n_e}{n_h} = n_i^2
\]

(7.1)

where \( n_i \) is the density of electrons and holes at that temperature for pure, undoped Si. (This relationship is pretty obvious for undoped Si, since we must have \( n_e = n_h \).) Ideally, we would like to be able to design components which still work when material specifications such as \( n_i \), or the temperature are slightly but unpredictably changed.

Another type of doping involves replacing selected Si atoms with atoms from group 3 of the Periodic Table. Thus we could add an impurity atom such as Boron (B) which has one less electron than Si in its outer shell. If we do this, then clearly we will find ourselves with an excess of holes rather than electrons, and another type of semiconductor. Due to the wonders of the laws of electromagnetism, holes can be viewed rather like bubbles of positive charge in an electric field — just as air bubbles in a liquid go up in a gravitational field (having an effective "negative weight"), so do holes go the "wrong way" in an electric field. Since they act like positive charges, B-doped Si is called "p-type" Si to indicate this. Note that, once again, relation (7.1) still holds.

### 7.1.1: The np Junction Diode and npn Transistor

We will now look at what it is about semiconductors that makes them useful in the manufacture of parts for computers. We start by examining the particularly interesting situation that occurs when slabs of p-type and n-type silicon are brought into contact with each other. This forms the basis of a device called a diode. We will give an idealized, qualitative discussion, and not allow ourselves to get bogged down in the murky details. We envisage a situation like that shown in Figure 7.3:

![Fig. 7.3 The np Junction](image)

On the left hand side we have the n-type material, which we can view as comprising a bunch of fixed positive charges and free negative charges. On the right hand side, we have the opposite situation. We know how many free electrons and free holes there are since they match up with the extra fixed B and P atoms — one electron per P, and one hole per B atom. At room temperature there will also be extra carriers due to thermal fluctuations.

---

\(^2\) Feynman actually used Aluminum instead of Boron in his lectures. Aluminum immediately precedes Si in the Periodic Table but has rarely, if ever, been used as a valence-3 dopant. [Editors]
In a moment we’ll stick this device into a circuit and put a voltage across it. First, let’s see what’s going on in the absence of any such field. The charge carriers will not only move about within their respective halves but will billow out, like steam escaping, into the adjacent material. However, this process of diffusion — of electrons into the p-type material and holes into the n-type — does not go unopposed. The fixed positive charges in the left-hand block will create an electric field that tends to pull the escaping electrons back; this field is increased by the holes percolating into the n-type material. These holes also experience a “tug”, from the fixed negative charges in their original half, and the electrons that have migrated over the barrier. We can actually list four separate phenomena operating at the join of our two slabs:

1. Creation and annihilation of electron-hole pairs by thermal fluctuations,
2. Conductivity (carrier drift prompted by electric forces),
3. Diffusion (carriers trying to smooth out the charge density), and
4. Electrostatic processes (due to the fixed charges).

After a while, this complicated physical system will settle down into an equilibrium state in which there is a concentration of fixed charges either side of the junction (Fig. 7.4):

![Fig. 7.4 The Equilibrium Charge Distribution for an np Junction](image)

The central region is actually depleted of charge carriers and is referred to as the depletion region. The density of the fixed charges in this region is not quite mirror-symmetric, as $P$ and $B$ have a different effective mass, but we will treat them as the same. We can add the signed densities of electrons, holes, and the fixed charges to obtain the net charge density in the device (Fig. 7.5):

![Fig. 7.5 The Net Charge Density](image)

The physical situation in equilibrium, it must be stressed, is not static. There are currents flowing due to diffusion. However, in equilibrium, the current flowing to the right cancels that to the left, resulting in no net current flow.

Let us see what happens if we apply a voltage across this system. We have two choices as to how we hook up our battery: we can connect the positive terminal to either the n-type or the p-type material. Let us consider connecting it to the p-type material first. If you think about it, you should be able to see that the effect will be to reduce the opposition to current flow caused by the fixed charges in the depletion region — the positive potential on the right will attract the electrons from the n-type material into the p-type. As the voltage increases, more and more electrons are able to diffuse across the boundary, and more and more holes, of course, can go in the opposite direction. Put bluntly, if we wire it right our “device” conducts madly! (An important rider to this bald statement is that it is essential for the maintenance of a current that “external” free electrons be drawn into the n-type part of the material from the point of contact with the battery, to continually replenish the flow. This is necessary because many of the “indigenous” carriers in the semiconductor will recombine with their opposite charges once over the boundary.)

What happens if we apply the voltage in the opposite direction? Now it gets interesting — we find that the material does not conduct! Why so? Well, any free electrons in the n-type material can happily go left, away from the junction region, and free holes in the p-type can go right, flowing out of the
semiconductor and into the circuit. However, the application of the voltage has increased the height of the potential barrier across the depletion zone, in fact, to the point where electrons in the p-type material cannot traverse it. (Needless to say, neither can any electrons that might be sourced from the point of contact with the battery.) Similarly, holes cannot maintain a current to the right, so after an initial blip, the current just drops off. There are too few free carriers available in the right places to sustain it. We say that the voltage reverse-biases the junction: in the current flow condition, the junction is said to be forward-biased. We call this device a junction diode and it has the fundamental property that it conducts in one direction but not the other.

Is there absolutely no current when the junction is reverse-biased? Well, not quite — there will be some flow due to thermal electron-hole pair creation at the junction. We wait for it to happen, it happens, and the electron scoots off in one direction, the hole in the other. The magnitude of the current created will clearly be temperature dependent — it actually increases exponentially with temperature — and largely independent of the applied reverse voltage. If we wish, we can aid the thermal process in a reverse-biased diode by creating electron-hole pairs ourselves. The thermal current is typically so small that, if we do create any pairs, we can easily detect them over the thermal background. How we go about producing pairs depends on the magnitude of the semiconductor’s band gap. In gallium arsenide (GaAs), for example, we can create pairs utilizing photons (in fact, this process is quite efficient even in Si). Naturally, the reverse is true: if we flip the electric field so that the diode becomes “forward-biased” and current can flow freely, electrons and holes move towards each other and annihilate, producing photons (in GaAs) or phonons (in Si). In this way, we can make semiconductor LASERs and LEDs.

Recall that electrons and holes annihilate at a rate $n_e n_h^2$. If this were zero, then in the reverse-biased case there would be no current since, by Equation 7.1, $n_e$ and $n_h$ would have to vanish also! However, in the forward-biased case, current could flow by filling up the p-type material farther and farther from the junction with electrons. If the applied voltage were to flip again, these electrons would have to all be laboriously brought back, and the diode would no longer prevent current flowing the wrong way — the device would be acting like a large capacitor. Annihilation solves this problem by letting the electrons in the p-type material fill in holes rather than be stored at increasingly great distances from the junction. So in fact, the diode would not really work without this annihilation process.

Let us return to the case when the field allows current to flow fully. It is possible to calculate the current $I$ that flows through the diode as a function of the applied voltage $V$. The math isn’t exceptionally difficult, but we will not go into it here (see suggested reading). The relationship turns out to be non-linear:

$$I(V) = I_0 \left[ \exp\left(\frac{qV}{kT}\right) - 1 \right]$$ (7.2)

where $V$ is the effective potential difference across the device — the voltage after allowing for a voltage in the “wrong” direction caused by the fixed charges at the boundary — and $q$ is the magnitude of the charge on the carriers (in Coulombs) and $I_0$ is a constant. The graph of $I$ against $V$ can now be drawn (Fig. 7.6):

In the real world, $I(V)$ cannot just keep on growing exponentially with $V$; other phenomena will come into play, and the potential difference across the junction will differ from that applied. Note also that the current trickle that exists in the reverse-biased case, catastrophically increases (negatively) at a certain voltage, the so-called breakdown voltage. This can vary from as few as five volts to hundreds, and can actually be exploited in some situations to limit the voltages in a circuit.

The next step in this chapter is to look at the operation of another semiconductor device, the famous transistor. Now transistors come in all shapes and sizes, and those that are deployed extensively in VLSI chips are quite different from those that are used by hobbyists. As an example of this latter type we mention very briefly the venerable npn bipolar junction transistor, one of
several historical antecedents of more modern transistor devices. This transistor is formed by sandwiching an extremely thin slice of p-type material between two of n-type material (hence "npn"). The various slabs are denoted the base, the collector, and the emitter, as shown in Figure 7.7:

![Diagram of npn Bipolar Junction Transistor](image)

**Fig. 7.7 Structure of the npn Bipolar Junction Transistor**

The "base" got its name from the fact that the transistor was built of this material: the terms "emitter" and "collector" both derive from prehistoric vacuum technology. Note the relative thickness of the base to the slabs on either side of it — this feature is essential to the transistor's correct operation. What this device is, if operated properly, is an amplifier: small changes in the current to the base are amplified at the emitter. It can also act as a switch and can be used in all of the transistor circuits discussed thus far in this book. However, bipolar transistors are not the most commonly used transistors in modern VLSI chips and I will therefore not discuss the (complicated) mode of operation of this transistor at the electron and hole level here. Good discussions can be found in many standard texts (see suggested reading). Instead, we will take a look at the type of transistor that is most commonly employed in VLSI systems — the sort of transistor that is usually built onto silicon chips. This is the MOSFET, an acronym for Metal Oxide Semiconductor Field Effect Transistor.

### 7.1.2: The MOSFET

We begin with a sketch of the structure of the MOSFET (and will worry about how to actually build such a device on a chip later):

![Diagram of MOSFET](image)

**Fig. 7.8 The MOSFET**

The bulk of a silicon chip consists of a slab of lightly doped silicon, the substrate on which the transistors and whatnot are laid down. If the doping is of the p-type, we are dealing with so-called nMOS technology; if the substrate is n-type, we have pMOS. We will focus on nMOS, as shown in Figure 7.8. We can identify three ports for our MOSFET: the gate is a conducting layer of "polysilicon" (a substance rather like a metal), which is separated from the silicon by a thin layer of non-conducting oxide. To either side of the gate, also separated from it by the oxide layer, are two n-type diffusion layers, materials so-called because they have doping diffused into them (as illustrated): they are referred to as the source and drain — as opposed to the emitter and collector of the npn device. These diffusion layers also conduct.

The transistor works as easy as pie. From our discussion of the np junction diode, we can see roughly what’s happening inside the device before we apply any external voltages. The substrate material is lightly p-type, and is usually kept grounded. The substrate forms a diode-like structure with the n-type layers — in essence, the MOSFET is built from two back-to-back diodes. As before, an equilibrium state will arise in which there are depletion regions at the n-p junctions with very few electrons and holes diffusing across them. So, the source and drain are effectively cut off from one another and if a voltage is put across them no current will flow. However, if we now put a positive voltage on the gate, things are different. The effect of such a voltage is to attract electrons to the underside of the oxide (they won't conduct into the gate because the oxide layer is an insulator). These electrons chiefly come from the source and drain — the positive potential having lowered the barriers at the depletion layers...
which prevented their free movement. The electrons under the oxide form what is called an inversion layer. If we now put a voltage across the source and drain, we will get a current — the inversion layer is essentially a channel that allows electrons to flow freely between the two contacts. So we have a fantastic device — a switch! The voltage that controls it is the gate-source voltage \( V_{gs} \), the bigger this is, the more charge carriers there are under the gate, and the more current that can flow. Note, however, that if \( V_{gs} < 0 \), the MOSFET will not conduct. In such a case, electrons are repelled from beneath the gate, and an inversion layer cannot be formed.

Actually, we have been a little simplistic here. Simply having \( V_{gs} > 0 \) does not automatically allow a current to flow. It is necessary for \( V_{gs} \) to exceed a certain minimum voltage, the threshold voltage \( V_{th} \), before this happens (typically, \( V_{th} \) is of the order of \( 0.2V_{DD} \), where \( V_{DD} \) is the supply voltage, say five volts). Because of this, it turns out to be convenient to define a shifted gate-source voltage:

\[
V'_{gs} = V_{gs} - V_{th} \tag{7.3}
\]

in terms of which the condition for current flow is \( V'_{gs} > 0 \). We can actually design our MOSFET to make this threshold voltage either positive or negative. A negative threshold voltage is obtained by doping the silicon slab so that there is a thin conducting layer of n-type semiconductor under the gate, connecting the source and the drain. Transistors with \( V_{th} > 0 \) such as we discussed above are called "enhancement mode" transistors; if \( V_{th} < 0 \), they are called "depletion mode" transistors. (Depletion mode transistors turn out to be useful for fabricating resistors in nMOS VLSI, as we shall see later.)

Let's make all this a bit more quantitative. Suppose we want to find the drain-source current \( I_d \), for given gate-source and source-drain voltages. Those readers not interested in details can skip this as we will not need it subsequently — it's just a nice bit of physics! We can consider the gate/oxide/silicon sandwich to be a capacitor, modeled by two conducting plates of area \( A = WL \) (where \( W \) is the width of, and \( L \) the distance between source and drain), separated by a material of depth \( D \) and permittivity \( \varepsilon \). Let us denote the charge on this capacitor (i.e. the charge under the gate) at any time by \( q_g \). We can calculate the capacitance \( C_g \) for this system, using the well-known general formula for a parallel plate capacitor \( C = \varepsilon A/D \). We have:

\[
q_g = C_g V_{gs} \tag{7.5}
\]

Using the standard relationship between the voltage across a capacitor and the charge stored in it, we can write:

\[
q_g = C_g V'_{gs} \tag{7.6}
\]

Let us suppose first that the drain-source voltage \( V_{ds} \) is small. We know that the current \( I_d \) is just the charge under the gate divided by the time it takes for the electrons to drift from the source to the drain. This is a standard result in electricity. How long is this drift time? Drawing on engineering practice, we can write the drift velocity \( v_{drift} \) in terms of the "mobility" \( \mu \) of the charge carriers as \( v_{drift} = \mu E \), where \( E \) is the electric field across the drain/source. \( E \) is easily seen to be given by \( V_{ds}/L \). We can now straightforwardly find the drift time, which we denote by \( \tau \):

\[
\tau = L/\mu E = L^2/\mu VT \tag{7.7}
\]

Combining this expression with that for the charge \( q_g \) in Equation 7.5, we find that the current (charge divided by time) is given by:

\[
I_d = (\mu \varepsilon W/L)VT_g V_{ds} \tag{7.8}
\]

Our calculation has been a little simplistic: this result strictly only holds for small values of the source-drain voltage \( V_{ds} \). However, we can see that as long as \( V_{ds} \) is fairly small, the transistor has the interesting property that the current through it is proportional to the applied voltage. In other words, it effectively functions as a resistor (remember \( V = IR \)), with the resistance proportional to \( 1/V_{gs} \).

As the drain-source voltage increases, matters become more complicated. The drift velocity of the charge carriers depends upon the electric field \( E \), and this in turn is determined by both \( V_{ds} \) and \( V_{gs} \). However, it so happens that, if \( V_{ds} \) gets too big, the current across the transistor actually becomes independent of \( V_{gs} \), a phenomenon known as saturation — at this point, the current becomes proportional to \( V_{ds}^2 \). We can understand this strange phenomenon better by use of a fluid model analogy (described in more detail in the book by Mead and Conway). If you like water and gravity better than electricity, you should like
this!

Let us envisage two reservoirs of water, separated by a partition. We will actually take this water to be without inertia — a little like honey — so it will flow slowly, and not splash about all over the place. To begin, consider the state of affairs shown in Figure 7.9 where the "water" is on either side of a movable central partition:

![Figure 7.9 Fluid Analogy for the Forward-Biased MOSFET](image)

The diagram has been annotated with MOSFET-related words to force the analogy on the reader. In this case, there will clearly be a flow of water from the "source" to the "drain": in the transistor, this equates to a current, with the partition playing the role of the potential barrier the charge carriers have to overcome. In this situation the transistor is said to be "forward-biased". The height of the water column above the partition on the right represents $V'_{gs}$ of the left column above the partition, $V'_{gd}$ and so on. Note that the precise nature of the flow across the partition will depend on the level of the "water" in the drain reservoir. Consider now a second state shown in Figure 7.10:

![Figure 7.10 Fluid Analogy for the Back-Biased MOSFET](image)

In this case there will be no flow at all: for the MOSFET, this represents the "back-biased" case, where $V_{gs} < 0$. Thirdly, consider the case of saturation (Fig. 7.11):

![Figure 7.11 Fluid Analogy for the Case of Saturation of the MOSFET](image)

In this instance, where the level of the drain reservoir is below the level of the partition, the water from the source will simply "waterfall" into the drain, at a rate independent of the actual drain-level. In the MOSFET, this behavior would begin to occur when the "drain reservoir" and the "partition" were at the same height, e.g. when $V_{ds} = V_{gs}$. At such a point, the current flow will become constant, irrespective of $V_{ds}$.

We can implement this analogy physically to gain some insight into the process of saturation. This is an alternative derivation of the magnitude of the current flowing through the MOSFET. Let us take Figure 7.11 to define a voltage $\phi(x)$ beneath the gate which, due to the energy level structure of Si, happens to be proportional to the number of free electrons under the gate. Now the electric field under the gate is proportional to the derivative of this voltage $(\Delta\phi/dx)$. Hence, since the current must depend on the density of electrons multiplied by the electric field (which controls the electron drift velocity), we must have:

$$I = K \frac{d\phi}{dx} = K \frac{dx}{d\phi} \frac{d\phi}{dx}$$  \hspace{1cm} (7.8)

where $K$ is a constant. Now if you think about it, you should be able to see that the current $I$ has to be a constant, independent of $x$. This means that the function $\phi$ must be linear in $x$ and thus $\phi \propto \sqrt{x}$. We therefore obtain the general
expression:

$$I = (K/2L) (\phi[0] - \phi[L])$$  \hspace{1cm} (7.9)

However, at saturation we have $\phi(0) = V_{gs}'$ and $\phi(L) = 0$ so that we recover the quadratic dependence of $I$ on $V_{gs}'$ that we flagged earlier. What happens if we are not at saturation? Will we rederive Equation 7.7? Actually, no. We get something better. In the unsaturated case, $\phi(L) = V_{sd}$, the gate-drain voltage, and we find that:

$$I \propto (1/2) (V_{gs} + V_{sd})V_{ds}$$  \hspace{1cm} (7.10)

where $V_{ds} = V_{gs} - V_{sd}$. Why is this an improvement? It removes an anomaly in our earlier expression (Equation 7.7). In that expression, the current $I$ was strangely asymmetric between source and drain. Intuitively, one might prefer to replace it with an average over the gate-source voltage and the gate-drain voltage — that is just what Equation 7.10 amounts to. For the record, we give here a plot of the variation of the current with the various voltages we have considered in this section (Fig. 7.12):

![Graph showing the variation of current $I_{ds}$ with gate-source voltage $V_{gs}$ and drain-source voltage $V_{ds}$, highlighting the saturation region.](image)

Fig. 7.12 Current-Voltage Variations for MOSFET

Another question we can ask about the MOSFET is: in the state for which no current flows, how good an insulator is the device? You will remember that the junction diode permitted a small thermally-induced current to flow even when reverse-biased. Let us briefly discuss this effect in the transistor. Recall Figure 7.10, the "water" diagram for the back-biased case:

The probability of an electron jumping from the source to the drain must be proportional to the probability for it to have enough thermal energy for it to overcome the potential barrier $V_{gs}'$, namely, $\exp(-eV_{gs}'/kT)$. Thus the forward current will also be proportional to this factor. There will also be a backward current — we can similarly see that this will be proportional to $\exp(-e(V_{gs} + V_{sd})/kT)$. There will therefore be an overall current given by

$$I = \text{(constant)} \cdot \exp(-eV_{gs}'/kT) \text{[1 - exp(-eV_{sd}/kT)]}$$  \hspace{1cm} (7.11)

when $V_{gs}' < 0$. So, if we switch from a current-flow to a back-biased case, the current will not switch off instantly (note that the temperature of the device is likely to be higher than room temperature). However, if we have $kT \approx 1/40$ electron-volt, then the current turns off quite quickly for circuitry run at about 5 volts. We can amend Figure 7.12 to include the reverse current:

![Graph showing the transistor current as a function of source-drain voltage.](image)

Fig. 7.13 Transistor Current as a Function of Source-Drain Voltage
Let us take stock and summarize what we have learned about the properties of MOSFETs. Firstly, rather than redraw the physical picture of Figure 7.8 every time we want to discuss the device and draw circuits involving it, we'll need a diagrammatic symbol. There are many such symbols for the differing varieties of transistors encountered in VLSI technology. Figure 7.14 illustrates the fairly common convention for the MOSFET that we shall adopt.

![MOSFET Symbol](image)

**Fig. 7.14 Conventional MOSFET Symbol**

This can represent either an n- or p-type transistor. (We will show later how to amend this figure to indicate which — this is necessary in CMOS technology which uses both types.) The rule for both is that they act like switches — when the gate voltage $V_g$ is positive enough they conduct. For the n-type MOSFET, we have the following rules: the most negative of $A$ and $B$ is called the source, the other called the drain; and if the gate voltage is more positive than a certain threshold voltage, $V_{th}$, above the source voltage the device conducts — the switch is "closed" and current flows. For a p-type device, the most positive of $A$ and $B$ is the source and the transistor conducts if the voltage on the gate is more negative than a certain threshold below the source voltage. We also defined two modes of operation for a MOSFET — the enhancement and depletion modes. The former is the case when the threshold voltage, $V_{th}$, is positive for n-type and negative for p-type. In depletion mode, it is the other way around. Now, a nice feature of depletion mode MOSFETs is that, if $V_g = V_s$ (the source voltage), the device always conducts. Thus, if we directly connect the source to the gate so that each is automatically at the same voltage, we find our transistor acting no longer as a switch but as a resistance (Fig. 7.15):

![Depletion Mode MOSFET](image)

**Fig. 7.15 Depletion Mode MOSFET operating as a Resistor**

Why should we want to do this? It's a matter of simple economics and design. Implementing a standard resistance on a chip is both expensive and takes up a lot of space, neither of which are obstacles to the use of depletion mode MOSFETs. (Note, incidentally, that this trick would not work with an enhancement mode device.)

**Problem 7.1:** I will now set you a problem — I will actually give you the answer shortly, but you might like to work through the math to get some practice on the theory of the internal guts of these things. The question has to do with capacitance. In the diode, not only was the current a non-linear function of the voltage but so was its capacitance. This is also the case for the MOSFET. Now although most of the capacitance is in the oxide, the overall set up turns out to be actually highly non-linear — and quite interesting! The problem I am about to give you is designed to illustrate the nature of this non-linearity.

We model the electrode contact at the gate in the MOSFET as shown below in Figure 7.16. Suppose we have a large mass of lightly doped p-type material — in principle, this material should be of infinite depth (measured vertically in the diagram). On top of this we place a metal plate, also of very large extent (but now in the horizontal plane):

---

3 With the advent of CMOS technology (see later) nMOS depletion mode transistors are now rarely used; they are replaced by p-channel enhancement mode transistors. [Editors]
We put a positive voltage \( V \) on the plate and, as a result, negative charge carriers are attracted to its underside. Deep into the material, where the electrostatic forces due to the plate are negligible, the number of negative carriers will just equal the number of doping ions. However, because of the voltage on the plate, the density of carriers near to it will be greater, falling off to a constant as we go further away. If we label the depth into the material by \( x \), we can define a position-dependent carrier density \( n(x) \). We can also define a resulting electrical potential within the material, \( \phi(x) \). Finally, let us denote by \( n_0 \) the initial positive doping concentration. Now the question I want you to answer is: how much charge is there on the electrode? Put differently, what is the capacitance of this physical set up?

To help you practice, I'll give you some hints. Firstly, you have to take \( \phi(0) \) (i.e., at the electrode) to be the plate voltage \( V \), and take the idealized value \( \phi(\infty) \) to be zero. I will hand you on a plate a relationship between \( n(x) \) and \( \phi(x) \) resulting from thermodynamical considerations:

\[
    n(x) = n_0 \exp\left[\frac{q \phi(x)}{kT}\right]
\]

(7.12)

where \( q \) is the charge on the negative carriers, and \( T \) is the temperature, as usual. \( n_0 \) is a constant. Another essential relationship is that between the rate of change of \( \phi(x) \) as we go deeper into the material (in other words, the electric field within the semiconductor) and the charge density on the plate, \( Q \). Note that defining a charge density here is important — it would be meaningless to discuss the total charge for an infinite plate. We find (cf. Equation 7.4 or by Gauss' theorem) the result that at \( x = 0 \):

\[
    \frac{\partial \phi}{\partial x} = \frac{Q}{\varepsilon}.
\]

(7.13)

where \( \varepsilon \) is the permittivity of the doped material and determines how rapidly the electric field drops off with distance from the plate. Using the standard Poisson equation \( \frac{\partial^2 \phi}{\partial x^2} = -\rho(x) \), where \( \rho(x) \) is the charge density within the material (which you can find in terms of \( n(x) \) and \( N \)) and integrating using the boundary conditions, you should find the result of the form of Equation 7.14:

\[
    Q = V[2(\varepsilon' - V - 1)V^2]^{1/2}
\]

(7.14)

in some set of units \( (kT/q = 1 \text{ and } n_0 q e = 1) \). The rather odd appearance of a \( V^2 \) within the square root, which you might think ought to cancel with the \( V \) outside it, is necessary to get the correct sign for \( Q \). Now you can see by comparison with the standard formula defining capacitance, \( C = qV \), that the capacitance of this system displays an extremely non-linear relationship with the plate voltage, \( V \). To my knowledge, this property is not much exploited in VLSI — although there have been recent applications in “hot clocking” (which we discuss later).

Thus far, we have considered an isolated MOSFET device on a silicon substrate. The next stage in our journey into the heart of VLSI is to take a look at how these transistors might actually be put together on chips to make logic circuits. We now come to real machines!

### 7.1.3: MOSFET Logic Gates and Circuit Elements

To build logic circuits we need to be able to build logic gates and we have already seen, in Chapter Two, how to do this using generic "transistors". We use the same approach with MOSFETS. Consider what happens when we hook up a transistor to a supply voltage, \( V_{dd} \) across a resistance as shown in Figure 7.17:
We will take our transistor to be of the nMOS variety, operating in enhancement mode. (There are many types of VLSI design, and we cannot consider all of them — it makes sense to focus on one in particular.) If terminal \( X \) (the gate) is near zero, then the transistor is an insulator, and the output voltage at \( Y \) is near the supply voltage, \( V_{DD} \). We interpret this state of affairs as meaning that the output \( Y \) is at logical 1. However, if \( X \) is near \( V_{DD} \), then the transistor conducts. If we suppose it conducts much better than the resistance, then \( Y \) is near zero; this state we equate with logical 0. As a rule, we do not operate between these extremes, except perhaps temporarily. This single MOSFET device therefore operates as a NOT gate (an inverter), as we saw in Chapter Two, since it just flips the input signal.

We can follow Chapter Two's lead for the other canonical gates. For example, the NAND (NOT AND) gate is built as follows (Fig. 7.18):

![Fig. 7.18 The NAND Gate Realized by MOSFETS](image)

In this system, both inputs \( A \) and \( B \) must be logical 1 for the output \( Y \) to be 0. To get the AND gate, we obviously just tag an inverter onto the output. To remind yourself of how to get a NOR gate, check out Chapter Two!

One can build other useful elements onto chips using MOSFETs apart from logic gates. Consider the matter of resistors — as I stated earlier, it turns out to be expensive and area-consuming to put standard forms of resistor onto silicon chips so it is normal practice to employ depletion-mode transistors in this role. Thus, in nMOS technology, the MOSFET structure of our inverter would actually be that shown in Figure 7.19:

![Fig. 7.19 nMOS Inverter with MOSFET Resistance](image)

Now there is another essential property of MOSFETs that is not evident from strictly logical considerations. This is their behavior as amplifiers. Consider what happens if we place two inverters in sequence (Fig. 7.20):

![Fig. 7.20 "Follower" Circuit](image)

From a logical viewpoint, this is a pretty trivial operation — we have just produced the identity. We're not doing any computing. However, from the viewpoint of machinery, we have to be careful; transistors dissipate energy, and one might naively think that the output of a chain of devices such as this would ultimately dwindle away to nothing as the power dropped at each successive stage. This would indeed be disastrous! However, this clearly doesn't happen: the input current to the second transistor may drop slightly, but it will not be enough to alter the mode of operation of this transistor (i.e., conducting or not), and the output \( Y \) will still be pulled up to the supply voltage (or down to
ground, whichever is appropriate). In other words, the output will always represent a definite logical decision, being relatively insensitive to minor power fluctuations along the chain. This circuit is an extremely effective so-called "follower", which jack-ups the power or impedance behind the line (if you like, it is a double amplifier). In a sense, we can control the whole dog just by controlling its tail. Needless to say, this amplifying property is crucial to the successful operation of circuits containing thousands or millions of transistors, where we are constantly needing to restore the signals through them. The presence of amplification is essential for any computing technology.

With VLSI, as with other areas of computing, we are often concerned with matters of timing. In this regard, it is interesting to ask how fast an inverter can go. That is, if we switch the input at the far left of a chain of connected inverters, what happens at the output on the right? The switching certainly won't be instantaneous: the output of each transistor must feed the input of another and charge up its gate, and this will take time. Each gate voltage must be changed by some value V with the gate having some effective capacitance $C_g$, say. If we can find how long the process takes, and maybe think up ways of speeding it up, we might be able to get better machines. We can shed some light on this process by examining the circuit depicted in Figure 7.21, in which we have explicitly inserted a capacitor to represent the gate capacitance:

![Fig. 7.21 Effective Electrical Analogue of a Follower Circuit](image)

Suppose the accumulated charge needed for a decision (i.e. for the gate voltage to be adequate for the transistor to switch) is $Q$. Then, $Q = C_g V$. How fast can we deliver this charge, or take it away? Firstly, note that the state $X = 1$ does not correspond to the first transistor's output being exactly at ground; the transistor will have a certain minimum resistance (which we call $R_{min}$) resulting in a slight voltage drop across the device. Now it is a standard result in electronics that the discharge time is determined by the product $R_{min}C_g$, assuming an analogy with the standard RC circuit shown in Figure 7.22:

![Fig. 7.22 Equivalent RC Circuit](image)

Again from standard circuit theory, the charge $Q$ on the capacitor at time $t$, $Q(t)$, obeys

$$Q(t) \propto \exp(-t/R_{min}C_g) = \exp(-t/\tau),$$  \hspace{1cm} (7.15)

with $\tau = R_{min}C_g$.

Clearly, if we were interested solely in getting the inverter to go faster, then we could achieve this by decreasing both $R$ and $C$, something we could do by making the circuit smaller. However, there is a limit to this: recall that, even in an inactive state, electrons from the source and drain nonetheless see a small distance into the silicon substrate of the MOSFET. As we shrink the device down, these carriers drift closer and closer to the opposite pole, until there comes a point where they actually short-circuit the region under the gate, and we will get a current flowing without having to manipulate the gate voltage. When this happens, it is back to the drawing board: a redesign is now needed, as the transistor will no longer work the old way. This is a nice example of how Nature places limitations on our technology!

So what do we do if we want to build smaller machines? Well, when the rules change, redesign, as I have said. Consider, for example, the case of aeronautical engineering with incompressible air and low-speed aircraft. A detailed analysis concluded that propeller-based machines would not work for
speeds in excess of that of sound: there was a "sound barrier". To get a faster-than-sound plane, it was necessary to go back pretty much to square one. At this moment in time, we have yet to find a fundamental limit on sizes for Si computers — there is no analogue of the sound barrier. This problem is an instance of how thinking differently from everyone else might pay dividends — you might blunder into something new! Currently, state-of-the-art devices have \( RC \approx 10 \) picoseconds\(^4\). By the time you have managed to reduce this significantly, you will probably find that others will have undercut you using some other technology! This actually happened with superconducting computing devices: as researchers were working in this area, its advantages were continually disappearing as advances were made in conventional VLSI technology. This sort of thing is quite a common occurrence.

Thus far in this chapter, we have reviewed the structure of various semiconductor devices used in computing but have so far had little to say on the practical limitations in this area. We address some examples of this now, beginning with a discussion of the important topics of heat generation and power loss in computers.

### 7.2: Energy Use and Heat Loss in Computers

In Chapter Five, we pointed out that a typical transistor dissipates some \( 10^{14} \text{kT} \) in heat per switch. This is a phenomenal amount — if we could get it done by a factor of ten or a hundred, we could simplify our machines considerably just by getting rid of all the fans! One particularly annoying problem with the nMOS technology we have discussed up to now is that even in the steady state of a MOSFET's operation — when \( X=1 \) (\( Y=0 \)), say, and the transistor is merely holding this value, not changing it — current flows continuously. So even if our transistors aren't doing anything, they're throwing away power! Obviously, any technology that offers the hope of more economical behavior is worth exploring, and the Complementary Metal Oxide Semiconductor (CMOS) technology that we will look at in this section is just such a technology.

\(^4\) When Feynman delivered his course, the value of \( RC \) was actually of the order of 4 nanoseconds. This 400-fold improvement in timing is an illustration of the extraordinary rate at which VLSI technology has advanced. [Editors]

---

#### 7.2.1: The CMOS Inverter

In the CMOS approach, we employ a mixture of n-type and p-type MOSFETs in our circuitry. The way in which we combine these to make a standard CMOS inverter is shown in Figure 7.23. As with the nMOS inverter, logical 1 is held to be near +V, for some voltage \( V \), but logical 0 is not at ground but can be chosen to be at –V:

![Fig. 7.23 The CMOS Inverter](image)

To indicate the doping type of each MOSFET (n or p) we have followed the convention of writing the appropriate letter adjacent to its symbol. Note that the nMOS depletion mode transistor has effectively been replaced by a conventional p-channel transistor. Is this circuit worth building? Yes, for the following reason. Suppose the input \( X \) is positive. Then the n-type MOSFET has its gate voltage above that of the source and it conducts: the p-type device, on the other hand, is reverse biased and therefore doesn't conduct. The output \( Y \) is pulled down to –V. Now switch \( X \) to zero. As you can see from Figure 7.23, the upper transistor now conducts and the lower doesn't; the voltage \( Y \) rises to the supply. So far, nothing new — this is just the standard operation of an inverter. However, this circuit has a novel feature: specifically, after the transition occurs, no current flows through the circuit! The route to –V is cut off by the insulating n-type MOSFET. (I'll leave it to the reader to see what happens when the input is switched back again.)

This is a remarkable property. In a CMOS inverter, no energy is required...
to hold a state, just to change it\(^5\). The CMOS inverter can also serve as a useful simple ‘laboratory’ for investigating some of the energetics of logic gates. The matter of how much energy is required for a logical process was considered in the abstract in Chapter Five, but it is obviously important to get a handle on the practicalities of the matter. We would, unsurprisingly, like our devices to use the very minimum of energy to function — and to this end we will have to take into account the amount of energy required to make a decision, the time taken in switching, the reliability of our components, their size, and so forth. Let’s start by considering in more detail the electrical behavior of a CMOS inverter as part of a chain. This will enable us to examine also the amplification properties of CMOS devices. To proceed we will employ a simplified (and none too accurate) model due to Mead and Conway.

In this model, we treat the two transistors simply as controlled resistors. We thus have the following equivalence (Fig. 7.24):

![Diagram of CMOS inverter]

This CMOS device is to be visualized as one in a linear sequence. The input at \(X\) is fed in from the previous gate; the output at \(Y\) is to be considered the input to the next gate, which has an effective capacitance to ground of \(C\), say (which we take to be a constant, although this isn’t strictly true). Ultimately, we want to examine the behavior of the output voltage as we vary the gate voltage, \(V_g\), at \(X\) — i.e., as we perform a switch. Let us first consider the simple case where we keep the voltage to the \(X\) input gate constant. This will prompt a flow of current. What will the final, equilibrium voltage at the output be? Denote the current through the transistors by \(I_1\) and \(I_2\), and define the difference between them (that is, the current that transfers charge to any subsequent component connected to \(Y\)) to be \(I = I_1 - I_2\). The voltage at \(Y\) is a function of time, say \(v(t)\). Let us also take the charge accumulating at \(Y\) to be \(Q(t)\). From standard circuit theory, we know that:

\[
dQ/dt = I_1 - I_2 = CdV/dt
\]

(7.16)

and from Equation 7.11, we can see that for small drain-source voltages the currents \(I_n\) are given by:

\[
I_n = V_{ds,n}/R_n
\]

(7.17)

where the interpretation of \(V_{ds,n}\) is obvious, and the effective resistances \(R_n\) are given by

\[
R_1 = R_0 \exp(qV_g/kT), \quad R_2 = R_0 \exp(-qV_g/kT).
\]

(7.18)

Note that we have sneakily removed all sign of the threshold voltage in \(V_g\) — we are considering our devices to be somewhat ideal.

If we now combine the basic equations for \(I_1\) and \(I_2\) given below:

\[
I_1 = (V - v)/R_1, \quad I_2 = (v + V)/R_2
\]

(7.19)

with the Equations 7.16 and 7.18, we can straightforwardly derive a differential equation relating \(C, v(t)\) and \(R\):

\[
CdV/dt = -2(V_{ds,n}/R_0) \sinh(V_g/V_p) - 2(V_{ds,n}/R_0) \cosh(V_g/V_p).
\]

(7.20)

So, if we keep the voltage on the gate fixed, what is the equilibrium value at the output, i.e. the value it has when everything has settled down? Well, when everything has stopped sloshing about, \(dv/dt = 0\), and we see directly that the equilibrium value, \(v\), say, is given by:

\[
5\text{ Strictly speaking, there will be a small current flowing through the reverse-biased transistor, but we largely neglect this in our considerations. [RPF]}
\]
\[ v_e = -V \tanh(V_g/V_p) \]  

(7.21)

where \( v_e \) is a constant. Since \( V/V_p \) is typically a large positive number or a large negative number, the equilibrium voltage asymptotically approaches \(+V\) or \(-V\).

We can use this result to analyze the amplification properties of a chain of CMOS inverters. Suppose we vary the gate input slightly, say \( V_i \rightarrow V_g + \delta V_g \). In response to this, the output will vary by some amount which we denote \( \delta v_e = A \delta v_g \). In response to this, the output of the gate fed by \( v_e \), \( v' \) say, will itself vary, by \( \delta v' = A v_e = A^2 v_g \); and so on down the chain. Clearly, if this CMOS device is to work, it must be the case that the magnitude of this factor \( |A| \) is greater than one; if it were not, then any change of input at the left-hand end of the chain would not propagate all the way through and eventually peter out. The amplification factor \( A \) is the slope of the graph of \( v_e \) against \( V_g \) at the origin, \( V_i = 0 \) (Fig. 7.25):

![Fig. 7.25 Gate versus Output Voltages for the CMOS Inverter](image)

The slope at the origin is \(-V/V_T\) (as you can show). Hence, we only need our DC supply voltage to exceed the order of \( 1/40^e \) of a volt for the chain to work. In practice, of course, the supply voltage is much higher (say five or six volts) so we see that the amplification is quite significant. The output voltage is an extremely sensitive function of the input since small input changes are magnified many times at the output.

**Problem 7.2**: Here are some problems, not easy, for you to try. So far, we have considered the equilibrium behavior of a CMOS circuit. What I'd like you to do now is analyze its behavior in time, by solving the equation (7.20) to find how long it takes the output to switch if we switch the input. The general solution, for which \( V_g \) is an arbitrary function of time, is obviously too difficult, so assume in your calculation that the input voltage switches infinitely rapidly.

Next, consider the dissipation of energy in the inverter. I stated earlier that, while it is a useful qualitative idealization to think of no current flowing through the circuit in equilibrium, this is not actually the case (indeed, our previous calculation presumes otherwise). The reverse-biased transistor just has a very high resistance. This results in a small perpetual power loss, which you can find using the standard electrical formula of the power dissipated by a voltage drop \( V \) across a resistance \( R \) (alternatively, you could use \( P = I^2 R \), where \( I \) is the leakage current). There will also be power loss in the switching process — this occurs when we dump the charge on the gate through the (now conducting) resistance. You should find the energy lost during switching to be \( 2C V_T^2 \). Also, what is the time constant \( \tau \) of the effective gate capacitance?

Although we are interested in CMOS technology chiefly for what it can tell us about the energetics of VLSI, for completeness I will briefly digress to illustrate how CMOS inverters can be used to construct general logic gates. Consider the implementation of a NAND gate — remember, if we can build one of these, we can build everything. A NAND gate then results from the arrangement shown in Figure 7.26:

![Fig. 7.26 NAND Gate Realised in CMOS](image)

Let us see how this works. Recall that, for a NAND gate, the output is zero if both inputs are one, and one for all other inputs. That is clearly what will occur here: the output voltage in Figure 7.26 can only be pulled down to \(-V\), i.e., logical zero, if both of the lower transistors conduct. This can only occur when
both inputs are positive. If either input is negative, the respective transistor will fail to conduct, and the output voltage will stay at $+V$.

Let us return to the matter of energy dissipation in CMOS devices. In practice, the energy dissipated per switch is of the order of $10^4kT$. This is very big so here is an opportunity for people to make a splash in the engineering world: *there is no reason why it should be so high.* Obviously, the voltage must be a certain size depending on the technology implemented in our devices, but this is not a fundamental limitation, and it should be possible to decrease the energy dissipated. (Remember our analysis in Chapter Five where we saw that $kT$ log 2 was theoretically attainable.) Let us discuss what can be done in this area.

Consider what actually happens in the switching process. Before we make the switch, there is a voltage on the input capacitance and a certain energy stored there. After we switch, the voltage is reversed, but the energy in the capacitance is the *same* energy. So we have done the stupid thing of getting from one energy condition to the same energy condition by dumping all the juice out of the circuit into the sewer, and then recharging from the power supply! This is rather analogous to driving along the highway at great speed, slamming on the brakes — screeech! — until we come to a halt; and then pushing the car back up to speed again in the opposite direction! We start off at sixty miles an hour, and we end up there, but we dissipate an awful lot of energy in the process. Now, in principle it should be possible to put the energy of the car into (say) a flywheel and store the energy. Then, having stopped, we can get started again by drawing power from the flywheel rather than from a fresh source. We shouldn't have to throw the energy away. Is there some parallel in VLSI to this flywheel?

One suggestion is to store the energy in an inductance, the electrical analogue of inertia. We build the circuit so that the energy is not thrown away, but stored "in a box" so that we can get it out again subsequently. Is this possible? Let's see. To explain the concept of inductance, I'll turn to another useful analogy using water. Those of you who are electrically-minded are used to analogies between water and electricity; those more comfortable with mechanics than electricity will also find water is easy! Imagine we have the arrangement shown in Figure 7.27, consisting of a large water-holding vessel with a couple of pipes leading into it:

![Fig. 7.27 Water Analogy for the CMOS Switch](image)

Each pipe is connected to an essentially bottomless reservoir (not shown), into or from which water can flow — this flow is regulated by a valve on each pipe. The analogy here is that the pipes plus valves represent the transistors, and the water in the reservoirs is charge from the power supply just waiting to be dumped through them. The upper reservoir corresponds to the voltage $+V$, the lower to voltage $-V$, and the height of the water in the tank can be interpreted as the voltage through which the charge will be dumped. To keep the analogy meaningful, the valves are rigged so that if one is open (conducting), the other is closed (insulating). To model the switching process in an inverter, we open and close the valves in this system and see what happens.

The initial condition is that shown in Figure 7.27, with the upper valve closed and the lower valve open. The water sits at some equilibrium level. Suppose we now switch the system by closing the lower valve and opening the upper (corresponding to a negative gate voltage). The water from the upper reservoir rushes in — splasch! — filling up the tank up until a new equilibrium depth is reached. In the process there is noise, friction, turbulence and whatnot, and energy is dissipated. There is a power loss. Eventually, everything settles down to a fresh equilibrium point. We now want to go back to our initial situation, so we switch again, opening the lower valve and closing the upper. Down comes the water level, dissipating energy in a variety of ways, until the water in the tank reaches its original height. We are back where we started, but we have used up a heck of a lot of energy in getting there!

We would like to alter this set-up so that we don't lose so much energy every time we switch. One way we could do this is as follows. We put another tank next to the first, and join the two by a tube containing a valve (Fig. 7.28):
Suppose we have the upper valve open so the water level of our original tank is as shown in Figure 7.28. If we now close the upper valve and open up the valve into the adjoining tank, the water goes splashing through the connecting tube into the new tank. When the water level reaches its maximum height in this tank, we close the valve. If we were to just leave the adjoining valve open, the water would slosh back and forth, back and forth between the two tanks and eventually settle down into a state where the height in both tanks was the same. In this case the pressure would be equalized but this finite time to stability results from the fact that water has inertia. When the valve is first opened, the water level reaches a height in the new tank that is higher than what would be the equilibrium value if we let the system continue sloshing about. Likewise, the initial level in our original tank will be lower than its equilibrium value. By closing the valve just after this high point is reached, we have actually managed to catch most of the water, and hence its potential energy, in the new tank. Not all of it, of course — there will be losses due to friction, etc., and we might have to top the new tank up a bit. But now if we want the energy of the water back, we just have to open up the adjoining valve to the adjoining tank when the right-hand tank is at a low ebb.

To implement this in silicon we need the electrical analogue of this and that means we need the analogue of inertia. As I’ve said, for electricity this is inductance. One way to implement the above idea can be seen by considering the following circuit (Fig. 7.29):

This circuit contains a capacitor, an inductance, a resistance and two “check valves”, based on diodes. When one of the switches is closed, the diode ensures that the current can only flow one way, mimicking the one-way flow of water through the two pipes in the water model. You should be familiar with the basic equation defining the behavior of the circuit:

$$L \frac{d^2Q}{dt^2} + R \frac{dQ}{dt} + \frac{Q}{C} = V$$

(7.22)

where V is the voltage across the circuit. I will leave it to you to see if you can implement this sort of idea using CMOS as the basis. Unfortunately, it turns out that it is extremely difficult to make appreciable inductances with silicon technology. You need long wires and coils and there’s no room! So it turns out that this is not a practical way of getting the energy losses down. However, that need not mean we have to abandon the basic idea — a very clever thing we can try is to have just one inductance, off the chip, instead of many small ones, as in one per switch.

7.2.2: Hot-Clocking

Here is a completely different, and very clever, way to get the energy dissipation down. It is a technique known as hot-clocking. In this approach, we try to save the energy by varying the power supply voltages. How and why might something like this work? Let’s return to our water analogy. Earlier, we saw that if we opened the upper valve while the level of water in the tank was low, then
we would lose energy as water flooded in from above and cascaded down. Where we are going wrong is in opening the switch while there was a difference in water level. If we do that, we will unavoidably lose energy. In principle, however, there are other ways of filling tanks which aren't nearly so wasteful. For example, suppose we have a tank to which is attached a single switched pipe, at the end of which is a water reservoir. If we fill the tank by the gradual process shown in Figure 7.30, opening the switch and moving the pipe up the tank as it pours so that it is always at the height of the water, then we will dissipate no energy:

![Fig. 7.30 Non-dissipative Filling of a Tank](image)

Of course, we would have to perform the operation infinitesimally slowly to completely avoid a dissipative waterfall (this type of argument was used frequently in Chapter Five). However, it is clear that if we could move things so slowly, then we could really get the energy loss down as long as we never opened the switch when there was a difference in level between the pipe and the head of water in the tank. There is an analogous principle in electricity: Never open or close a switch when there's a voltage across it. But that's exactly what we've been doing!

Here's the basic principle of hot-clocking. Consider the amended inverter circuit in Figure 7.31:

![Fig. 7.31 Sample "Hot-Clocking" Circuit](image)

In Figure 7.31, the upper and lower voltages $V_{\text{TOP}} (= V, \text{say})$ and $V_{\text{BOTTOM}} (= -V_{\text{TOP}})$ are not to be considered constant: they can, and will, vary, so watch out! We will define the two main states in which they can be as the quiescent state, which corresponds to the upper voltage being negative and the lower positive, and the hot state, the inverse of this, with the upper voltage positive and the lower negative. (These designations are arbitrary — we could just as well have them the other way around.) The principle of operation of this device is this. Suppose we start in the quiescent state, so the upper voltage in Figure 7.31 is negative, and have X positive ($= +V$). Then, the p-MOSFET is open, the n-MOSFET is closed, and no current flows (there is no voltage across the n device). In fact, even if X is negative, no current will flow due to the rectification property of the diode. So we can switch the input willy-nilly while in the quiescent state — the circuit is quite insensitive to the input voltage. This clearly leaves us free to choose our initial state for Y; we will take this to be positive.

Now, we let the voltages go hot — we gradually turn them around. Now a positive voltage gradually grows across the bottom diode, which conducts. This draws the output Y down to that of the lower voltage (which is now negative). The energy dissipation as this occurs is small as resistance of the diode is low. When this lower voltage bottoms out and things have settled down, we switch back to the quiescent state again: the output Y would like to revert to its previous value but cannot, as the diode prevents any current from flowing.
We can change X, that is, make a switch, as we please once in this stable state. It is necessary to run the first part of the cycle, when Y changes, rather slowly; the second stage, the return to quiescence, can be performed rapidly.

Now the output of Y must feed another gate. Clearly, we cannot use it while it is changing so the voltage cycle of the next gate must take place somewhat out of phase, with a different power supply (rather like a two-phase clock). It is possible, as is common with flip-flops, to have the second signal simply the inverse of the first, and hence use just the one supply — but this is dangerous, and slightly confusing, as going back to quiescence allows Y to vary a little. It is safer to design conservatively, with two separate power supplies. We can exhibit this diagrammatically by plotting the voltage changes of the two supplies (Fig. 7.32):

![Fig. 7.32 The Supply Voltages](image)

Note that the leading edge of each pulse is more leisurely than the trailing edge, reflecting the differing times of switching in the two stages. Let us also point out that these power supplies are universal to the entire chip or chip; otherwise, we could see that the amount of energy required to vary the supply voltage would effectively offset any savings we might make. We store outflowing energy in the power supply machinery.

Let's go back to the diode arrangement and calculate the energy lost during the switch. Let's suppose that the "rise time" we are allowing for the supply voltage to shift is \( t \). The charge that we have to move during the change is \( Q = CV \) and hence the current that will flow is (on average) just \( Q/t = CV/t \). If we further suppose that the resistance we encounter in the diodes when we close them is \( R \), a small quantity similar to that of the transistors, then the rate of energy loss, i.e. the power loss, is just \( P = \dot{Q}R = Q^2R/t^2 \). Hence, the total energy loss in switching is:

\[
\Delta E = Pt = CV^2Rt = (CV^2)(CR)t = (CV^2)t. \tag{7.23}
\]

where \( t \) is the time constant of the original, naive CMOS inverter circuit. Also, recall that \( CV^2 \) was the energy loss during switching in that circuit. Therefore, we see that the energy loss multiplied by the time in which this loss takes place is the same for both the old and the new circuits. This is suggestive of a general relationship of the form:

\[
\text{(Energy loss)(Time of loss)} = \text{Constant} \tag{7.24}
\]

for each switching step or simple logical operation. This expression certainly appears to be in sympathy with the findings of Chapter Five: the slower we go, the less energy we lose. In actual circuits, the clocks are much slower than the transistors (e.g., a factor of fifty to one), and so clocking enables us to save a great deal of energy in various computations. Unfortunately, such is the current obsession with speed that full advantage is not being taken of the opportunities that power savings might offer. Yes, the machines would be slower, and bigger because of the extra components, but this might be offset by the fact that they would be cheaper to run, and there would be considerably less need for all the pumps and the fans and so forth needed to keep the things cool!

Now although I used diodes in my example of Figure 7.31, I ought to point out that a more realistic set-up, if we don't want to use too many different types of component, is that shown in Figure 7.33 below, in which the diodes are replaced by transistors:

![Fig. 7.33 "Diode-less" Hot-Clocking Circuit](image)
We have looked at just one of the so-called "hot-clocking" methods for reducing energy dissipation. These techniques (developed largely at CalTech\textsuperscript{6}) allow clock lines to deliver power but were not originally intended for trading time for energy. Let me finish this section by pointing out that hot-clocking is a fairly recent development, and there are still many unanswered questions about it — so you have a chance to actually do something here, to make a contribution! The circuit I drew was my own, different from others that have been designed and built, and I'm not sure if it has any advantages over them. But you can check out all manner of ideas. For example: what if the supply voltage was AC, i.e., sinusoidal? Could we perhaps use two power supplies, both AC and out of phase? Why not let the voltage across the logic elements be AC? Perhaps we could define two states, in phase with the power supply (logical one) and out of phase (logical zero). There are many opportunities, and perhaps if you delved further and kept at it, you might uncover something interesting.

7.2.3: Some General Considerations and an Interesting Relationship

One of the central discoveries of the previous section, which might be general, is that the energy needed to do the switching, multiplied by the time used for this switching, is a constant — at least for resistive systems. We will call this constant the "dissipated action" (a new phrase I just made up). Now the typical time constant $\tau$ of an inverter is of the order of 0.3ns, which is pretty small. Does it have to be so tiny? Well, yes, if we want to go as fast as possible. But we can approach the matter from a different angle. Because of delays on the lines, and because each element might have to feed others, and so forth, the actual clock cycles used are a hundred times greater in length — you can't have everything changing too quickly, or you'll get a jam. Now it is not obvious that we cannot slow the inverter down a bit — if we do so, it is not necessarily true that we will lose time overall in our computation in proportion to this reduction. Since this is unclear, it is interesting to find out exactly what is the value of our constant, which we shall write as $(Ei)_{\text{av}}$.

One way to do this is to work out the value of the constant for a specific switch for which it is directly calculable. We will therefore focus on the fastest possible switch and evaluate it for this — this is as good as any other choice.

\textsuperscript{6} A 1985 paper on 'Hot-Clock NMOS' by Chuck Seitz and colleagues at CalTech ends with the following acknowledgement: "We have enjoyed and benefited from many interesting discussions about 'hot-clocking' with our CalTech colleagues Alain J. Martin and Richard P. Feynman." [Editors]

Let's first recap our basic equations. Our switch, a single transistor, will have a certain capacitance $C_g$, and we put a voltage $V$ on it, and hence a charge $Q = C_g V$. This gives us a switching energy $E_{sw} = \frac{1}{2} C_g V^2$. Now $\tau = C_g R$, so $(Ei)_{\text{av}} = C_g V^2 R = Q^2 R$, that is, the square of the charge needed to make the switch work multiplied by the minimum resistance we get when the switch is turned on. (You can also understand this in terms of power losses, working with currents.) Now we're naturally interested in asking what this dissipated action constant is for our ordinary transistors. We want to know this to see if, by redesigning such devices, we can get it down a bit, and perhaps use less energy or less time.

In order to proceed with the calculation, which is rather easy, we will need some physical parameters. Firstly, the electron charge $e = 1.6 \times 10^{-19}$ C. Also, at room temperature we have $k T/e = 1/40$ Volt. Using the kinetic energy relationship $(1/2)mv^2 = (3/2)kT$ (where $m$ is the effective mass of the electron), we can define a "thermal velocity" $v_t$ — this turns out to be roughly $1.2 \times 10^7$ cm/s. We also need some of the properties of weakly-doped silicon material: the electron carriers have surface channel mobility $\mu = 300 \text{ cm}^2\text{V}^{-1}\text{s}^{-1}$, and a mean free path $l_{\text{ef}} = 5 \times 10^{-6}$ cm. As with our earlier analysis of the MOSFET, we take the silicon under the gate to be $L$ cm in length, $W$ in width: in 1978, a typical value for $L$ was 6 microns, falling to 3 by 1985\textsuperscript{7}.

![Fig. 7.34 The Simple MOSFET](image)

Suppose we have electrons sloshing about under the gate, and we impose a force $F$ on them, for a time $t_{\text{col}}$. This latter quantity we take to be the average time between electron collisions, which is a natural choice given the physics of the situation. There is an intuitively satisfying relationship between the mean free path and the collision time: $l_{\text{ef}} = v_t t_{\text{col}}$. Now, from standard mechanics, at the end of this time an electron will have gained a momentum $mv_d = Ft_{\text{col}}$ where

\textsuperscript{7} Standard technology in 1995 is now 0.5 micron with 0.35 micron available to major manufacturers like Intel. [Editors.]
the velocity $v_D$ is the "drift velocity" in the direction of the force, and is quite independent of (and much smaller than) $v_{th}$. Since mobility is defined by the relation, $v_D = \mu F$, we have $\mu = \tau_{col}/m$. Now, take the current flowing under the gate to be $I$. We have $I = Q/(\text{time of passage across the gate}) = Q/(L/v_D) = Q/L.(\mu e).(V_{sd}/L)$. However, the source-drain voltage $V_{sd} = IR$, so we have, for the resistance, $R = L/(Qe\mu) = mL^2/(Qe\tau_{col})$. (Incidentally, the effective mass of an electron moving through Si is within 10% of its free mass, so we can take $m$ to be the latter.) Now, using our expression for the dissipated action in terms of $Q$ and $R$, and using the relationships we have derived, we find:

$$\langle E \rangle_{av} = N(L/\tau_{col})^2(3kT)\tau_{col}$$

(7.25)

where we have introduced the number $N$ of (free) electrons under the gate, $N = Qe$. Now focus on the last two factors on the right hand side of Equation 7.25. $3kT$ is an energy, of the order of the kinetic energy of an electron, and $\tau_{col}$ is a time, the time between collisions. Maybe it will help us to understand what is going on here if we define the product of these two terms to itself be a dissipated action — just that dissipated during a single collision. Isn't it forced on us; we'll just see what happens. Let us call such an action $E_{col}$. Then we have:

$$\langle E \rangle_{av} = N(L/\tau_{col})^2 E_{col}$$

(7.26)

So we find that the $\langle E \rangle$ that we need for the whole switch is larger than the $E_{col}$ for a single collision by two factors. One is the number of electrons under the gate, and the other is the ratio of the width of the gate to the mean free path. Taking $L$ to be 6 microns (hence $L/\tau_{col}$ to be about 100), and the number of electrons $N$ under the gate to be about $10^6$, we find:

$$\langle E \rangle_{av} = 10^{10}\langle E \rangle_{col} = 10^{10}kT\tau_{col}$$

(7.27)

This ties in with what we have quoted before. Now this is an awful amount, and we would certainly hope that we can improve things somehow! Why is this number so large? We know from the considerations of the Chapter Five that it in no way reflects a fundamental energetic limit. What can we do to get it down a bit?

Of course, all of our calculations thus far have been rooted in the conventional silicon VLSI approach — so perhaps what we ought to do is step outside that technology and look at another. Let us take a more general, and somewhat abstract, look at this question. Suppose that you design for someone a beautiful switch, the fundamental part of a computational device, which has a certain switching energy $E_{part}$, and corresponding switching time $t_{part}$. Now you give this guy a pile of these parts, and he proceeds to build a circuit with them. But he does this in a most absurdly inefficient manner. He does this as follows (this might all seem a bit abstract at first, but bear with me). Firstly, he connects up, say, $p$ switches in parallel, and hooks them all up to the same input:

Fig. 7.35 A Possible Parallel Connection of Fundamental Parts

These switches all operate simultaneously, the signal propagating from left to right. Clearly, the energy dissipated in switching all of these parts is $E_{av} = pE_{part}$, and the time for it to occur is just $t_{av} = t_{part}$. In other words, $\langle E \rangle_{av} = p\langle E \rangle_{part}$. This is ridiculous, given that they all give out the same answer.

Next, the guy does something even dumber and connects up some parts in series as well, in chains $s$ parts long:

Fig. 7.36 A Serial Connection of Parts
This really is dumb! Each switch in the chain just inverts the previous one, so all he has overall is a simple switch, effectively no better than the parallel arrangement he started out with! Yet compared to that, now \( (E_t)_{11} = \rho s^2 (E_t)_{1e} \) as you should be able to see. So of what relevance is all this? Well, an electron colliding is rather like a 1-electron switch, with which we can associate a quantity \( (E_t)_{1e} = 3kT \tau_{col} \). We can consider such a collision to be the fundamental operation. Now all the electrons jiggling beneath the gate are doing the same thing, bumping into one another, drifting and so forth, and so we can consider them to be operating effectively in parallel; with the number of parallel parts \( p = N \), the number of electrons beneath the gate. Of course, one collision is not sufficient to account for the whole of an electron’s activity between poles -- the actual number of hits, on average, is \( (\langle l \rangle_{col} \sim s \), using the serial analogy here. So we can actually interpret our result for \( (E_t)_{1e} \) in terms of the crazy handiwork of our engineer: \( 10^{-6} \rho s^2 \) ! Surely room for improvement?

Okay -- so how can we improve on this? Firstly, is it completely silly to put things in parallel?! Not at all: it’s good for accuracy. It might be the case that we are working with parts that are extremely sensitive and which can easily be flippd the wrong way by thermal fluctuations and whatnot. Putting such parts in parallel and deciding the output on the basis of averaging, or by a majority vote, improves system reliability. If we have a part whose probability of malfunctioning is \( 1/4 \), then with just 400 of these in parallel, we can guarantee that the chance of the system spitting out a wrong answer is about 1 in \( 10^{14} \) -- wo! And what about putting parts in series? Well, I’ve thought a lot about this, but have yet to come up with anything resulting advantage. It wouldn’t help with reliability -- all it does is increase the lag. In fact, I can see no reason for having anything other than \( s = 1 \).

**Problem 7.3:** In our electron model, \( s = 1 \) would correspond to getting the fundamental ratio \( (\langle l \rangle_{col} \) down to unity. An interesting question arises if we actually take this notion seriously. In fact, I would like you to consider the most extreme case, that where the mean free path of the electrons below the gate is *infinite*; in other words, they suffer no collisions. Analyze the characteristics and behavior of such a device. Sure, on first impression, such a device could never function as a switch -- it would always conduct. But we have forgotten about *inertia*: in order to conduct, the electrons have to speed up and change their speed, and can only start at zero; so there is a certain density of charge beneath the gate anyhow. In fact, this whole analysis, with infinite mean free path, was originally made for vacuum tubes, and these certainly worked. So a switch of this kind can be devised, and analyzed -- it’s just that we can’t do it with silicon

(in which the electrons can be thought of as moving through some sort of "honey").

Generally, however we do it, we should make every effort to increase the mean free path and to decrease \( L \). There is a factor of 100 to be found in \( (E_t)_{fe} \) (not \( 10^4 \), because if we change the mean free path we change \( \tau_{col} \) as well). Current hardware design stinks! The energy loss is huge and there is no physical reason why we shouldn’t be able to get that down at the same time as speeding things up. So go for it -- you’re only up against your imagination not Nature.

An obvious suggestion is to simply reduce the size of our machines. We can make good gains this way. Let us scale \( L \) by a factor of \( \alpha < 1: L \rightarrow \alpha L \). We then find \( (\langle l \rangle_{col})^2 \rightarrow \alpha^2 (\langle l \rangle_{col})^2 \) as \( l_{col} \) does not scale. The number of electrons under the gate scales with area: \( N \rightarrow \alpha^2 N \). Hence, we arrive at the result:

\[
(E_t)_{1e} \rightarrow \alpha^2 (E_t)_{1e}
\]

(7.28)

This is excellent scaling behavior, and though we cannot trust it down to too small values of \( \alpha \), it shows that simply shrinking our components will be advantageous.

The \( (E_t) \) ideas I’ve put forward here are my own way of looking at things and might be wrong. The idea that \( (E_t) \) might be a constant is very reminiscent of the Uncertainty Principle in quantum mechanics, and I would love to have a fundamental explanation for it, if it turns out to be so. There is certainly room for you to look into such questions to see if you can notice something. Anything you can do to criticize or discuss these ideas could be valuable. If nothing else, because the simple relationship:

\[
Power = E_t = (E_t)_{fe} \mu^2,
\]

(7.29)

shows that reducing the dissipative action \( (E_t) \) should reduce the power loss from faster machines.

**7.3: VLSI Circuit Construction**

We now come, at last, to discuss the actual physical technology underlying VLSI. How are transistors actually made? How do we, being so big, get all this stuff onto such tiny chips? The answer is: very, very cleverly -- although the
The basic idea is conceptually quite simple. The whole VLSI approach is a triumph of engineering and industrial manufacture, and it's a pity that ordinary people in the street don't appreciate how marvelous and beautiful it all is! The accuracy and skill needed to make chips is quite fantastic. People talk about being able to write on the head of a pin as if it is still in the future, but they have no idea of what is possible today! We can now easily get a whole book, such as an encyclopedia or the Bible, onto a pinhead—rather than angels! In this section, we will examine, at a fairly simplistic level of analysis, the basic processes used to make VLSI components. We shall once again focus solely on nMOS technology.

### 7.3.1: Planar Process Fabrication Technology

The process all begins with a very pure crystal of silicon. This material was known and studied for many years before an application in electronics was found, and at first, it tended to be both rare and, when unearthed, riddled with impurities—nowadays, in the laboratory, we are able to make it extremely pure. We start with a block of the stuff, about four inches square, and deep, and we slice this into thin wafers. Building integrated circuits on this substrate involves a successive layering of a wafer, laying down the oxide, polysilicon and metals that we need according to our design. Remember from our earlier discussion that the source and drain of a MOSFET were n-type regions seeded into, rather than grafted onto, lightly doped p-type Si material—it is important to keep in mind that the silicon wafer we are using is actually this p-type stuff. To see the sort of thing that goes on, we'll explain in some detail the first step, which is to create and manipulate the non-conducting oxide layer on the silicon that will ultimately play a role in constructing the insulation layer under the gate of a transistor. We start by passing oxygen over the surface of the wafer, at high temperature, which results in the growth of a layer of silicon oxide (SiO$_2$). This oxide layer is shown in Figure 7.37. We now want to get rid of this oxide in a selective fashion. We do this very cunningly. On top of the oxide we spread a layer of "resist," an organic material which we bake to make sure it stays put. A property of this resist is that it breaks down under ultraviolet light, and we use this property to etch an actual outline of our circuitry on the wafer. We take a template—a "mask"—and lay this over the material. The mask comprises a transparent material overlaid with an ultraviolet opaque substance, occupying regions beneath which, on the chips, we will want SiO$_2$ to remain. (Usually, the mask will repeat this pattern over its area many times, enabling us to produce many chips on one wafer, which may be cut into separate chips later.) We next bombard the wafer with UV light (or X-rays). The affected resist, that not shielded by the opaque regions of the mask, breaks down, and can be sluiced off. This exposes channels of SiO$_2$ which we can now remove by application of a strong acid, such as hydrofluoric acid. The beauty of the resist is that it is not removed by the acid so that it protects the layer of SiO$_2$ beneath it—unlike the stuff we've just sluiced off—we want to keep in place. After this stage, we have an upper grid of resist, under which lies SiO$_2$, and beneath this a bare grid of the original silicon. We now apply an organic solvent to the wafer which removes the resist and leaves the underlying oxide intact. The result is, if you like, a layer of oxide with "silicon holes" in it (Fig. 7.37):
Such a transistor is perpetually closed and current can always flow unless we place a negative charge on the gate to stem this current flow and open the switch (hence, $V_n < 0$, as stated earlier). To put such transistors on the chip, it is necessary to lay down their foundations before we go any further: this entails first delineating their gate regions and then creating a very thin region of n-type doped Si over these areas. To do this, we cover the wafer with resist again, and place on it a mask whose transparent regions represent the depletion areas. Once again, we blast the wafer with UV or X-radiation, and this time we are left with a wafer comprising a covering of resist, dotted among which are spots of exposed silicon substrate. These open areas we dope with phosphorus, arsenic or antimony, to create the required depletion region. The resist prevents these ions from penetrating into the rest of the silicon. This done, we wash off the remaining resist.

The next layer to be taken care of is the polysilicon (polycrystalline silicon) layer. Recall that highly-doped polysilicon conducts well, although not as well as a metal, and will be used to construct, among other things, the gates of transistors. As these gates are separated from the substrate by a thin layer of insulating oxide (see Figure 7.8), it should come as no surprise to you that before we do anything with our polysilicon, we have to coat the wafer with another thin layer of oxide as we did initially. As before, we do this by heating the wafer in oxygen (note that this will leave the depth of oxide across the wafer uneven). The wafer is then coated in polysilicon and another mask overlaid — this time designed to enable us to remove unwanted polysilicon. Having done this, we have to build the drains and sources (and, generally, the diffusion layer) of our transistors — and we do this by doping all of the remaining silicon appropriately (i.e. with phosphorus). We achieve this by removing any oxide that is not lying under the polysilicon and mass-doping the exposed Si regions. The depletion layers beneath the polysilicon are protected will not be additionally contaminated.

We can now see how an enhancement mode transistor will arise from this process. To draw a diagram, we will adopt the conventions for the various layers shown in Figure 7.39 (the conventions in most common usage are actually color-coded):

- Polysilicon (red)
- Conductive (diffusion) Si (green)
- Depletion regions (yellow)
- Metal (blue)

We have added one more layer here — that of metal\(^2\). This layer comprises the "flat wires" we use to carry current a sizeable distance, in preference to polysilicon or the diffusion layer. (The power supply is usually drawn from metal paths.) It will also be necessary to add contact points to enable the current to flow freely between layers, as required. With this convention, we can draw an enhancement mode transistor as:

\[^2\] Three metal layers are now typical in 1996, with five available for specialists. Typically one of these "metal" layers would be polysilicon. [Editors]
The transistor is just the crossing point of a polysilicon path and a diffusion path! Of course, the two paths do not cross in the sense of making physical contact — there is a layer of insulating oxide between them.

A full inverter requires a resistance in series. As we discussed earlier, we use a depletion mode transistor for this task. The inverter circuit is shown in Figure 7.41 below:

![Figure 7.41 The Full Inverter](image)

You will note that I have included here the power supply and ground lines, both of which are metal paths. It is necessary in the fabrication process to leave patches of the diffusion paths exposed at the point where the metal crosses, so as to ensure an electrical contact. These features you cannot see from a vertical picture. (The actual circuit is not laid out wholly flat as in Figure 7.41; it's all built on top of itself, in a clever, tight little box. See Mead and Conway for more details.) A similar procedure is necessary if we want to, say, use the source or drain of a transistor as the input to another gate — we then have to connect a diffusion path to a polysilicon path. Obviously, some kind of direct contact is needed; otherwise, we would find a capacitor or transistor where the lines cross. We can use a so-called "butting contact" where we overlay a direct diffusion/poly contact with metal, as shown in Figure 7.42:

![Figure 7.42 Polysilicon-Diffusion Layer Contact](image)

To give an illustration of a more involved logic unit, we will look at the NAND gate. To make this, all we need to do is take our previous circuit, and cross the diffusion path with another polysilicon path to make another transistor (Fig. 7.43):

![Figure 7.43 The NAND Gate](image)

Note that in this circuit the polysilicon paths extend a little way beyond the diffusion path at the each of the two transistors. Why? Well, there are many design rules governing precisely how we should arrange the various paths on a
chip with regard to each other, how big the paths must be, and so on, and I'll briefly list some here. (For a fuller exposition of these 'lambda-based' design rules, see Mead & Conway). Let us begin by defining a certain unit of length, \( \lambda \), and express all lengths on the chip in terms of this variable. In 1978, \( \lambda \) was about 3 microns; by 1985, it had fallen to 1 micron, and it falls further as time progresses. The minimum width for the diffusion and polysilicon paths is \( 2\lambda \). The metal wire, however, must be at least \( 3\lambda \) across, to counter the possibility of what is known as 'electromigration', a phenomenon whereby atoms of the metal tend to drift in the direction of the current. This can be a seriously destructive effect if the wire is especially thin (Fig. 7.44):}

\[
\begin{align*}
\{2\lambda\} & \quad \{2\lambda\} & \quad \{3\lambda\}
\end{align*}
\]

**Fig. 7.44 Silicon Chip Path Widths**

Again, these are minima: the paths can be wider if we desire. Another set of rules pertains to how closely we can string wires together. Conducting paths cannot be placed too near each other because of the danger of voltage breakdown, which would allow current to criss-cross the circuit (Fig. 7.45):

\[
\begin{align*}
\{3\lambda\} & \quad \{2\lambda\} \\
\{3\lambda\} & \quad \{3\lambda\}
\end{align*}
\]

**Fig. 7.45 Silicon Chip Path Separations**

Metal paths (blue) can go on top of poly (red) and diffusion ones (green) without making contact. Where red crosses green, as we've said, there is a transistor. It is important with such devices that the poly line forming the gate extends over the edge of the diffusion region, to prevent a conducting path forming around it resulting from shorting the drain to the source. We usually require an overlap of at least \( 2\lambda \), to allow for manufacturing errors (Fig. 7.46):

**Fig. 7.46 Rules for a Transistor**

We must also consider the connections between levels. If we are hooking a metal line to another path, we must be sure the contact is good (the contact is typically made square). To ensure this, we do not just place the metal in contact with the path, area for area, but must have at least a distance \( \lambda \) of the path substance surrounding the contact to prevent leakage through the metal and into the surroundings. This is true whether we are connecting to poly, diffusion or metal lines (Fig. 7.47):

**Fig. 7.47 Rules for Contacts**

### 7.3.2: Circuit Design and Pass Transistors

To actually make a specific circuit, we would design all of the necessary masks (typically enormously complex) and send them to a manufacturer. This
manufacturer would then implement them in the construction process we have described to provide us with our product. There is a standard heuristic technique for drawing out circuits, one which tells us the topology of the layout, but not its geometry — that is, it tells us what which paths are made of, and what is connected to where, but it does not inform us as to scale, i.e. the relevant lengths of paths and so on. For example, the drawing (the so-called "stick figure") for the NAND gate is shown in Figure 7.48 (in which we have also indicated the new linear conventions we adopt for each type of path):

\[ \text{Fig. 7.48 "Stick Figure" for the NAND Gate} \]

This tells us all the important interconnections in the circuit but if we were to actually trace the final physical product, the actual scaling of the respective parts might be radically different. This latter need not concern us here and we will adopt the stick figure approach in what follows, when we want to take a look at some specific circuits. To make things simpler still, we can sometimes deploy a kind of "half and half" shorthand, in which we represent sub-circuits on the chip by black boxes (a common enough procedure). So, for example, if we had a simple chain of inverters, it would be easier, rather than drawing the entire transistor stick figures over and over, to use the scheme of Figure 7.49:

\[ \text{Fig. 7.49 Simplified Circuit Diagram for Chain of Inverters} \]

where the triangles are just the conventional symbols for inverters, and the line convention is as explained in Figure 7.48.

A common type of circuit is the shift register. We represent this in Figure 7.50 as a doubly-clocked inverter chain, crossed by polysilicon paths:

\[ \text{Fig. 7.50 A Shift Register} \]

The two (complementary) clock pulses are sent down polysilicon lines, and where these cross a diffusion line, they form what is known as a pass transistor, so-called because it only allows a current to flow from source to drain (i.e. from left to right in the above picture) if the gate is forward-biased. This occurs whenever the clock pulse to the polysilicon line is on. At the next pulse the next inverter in the chain switches and will hold its new value until the next clock pulse. The reader should be able to make contact with our discussion of clocked registers in Chapter 1 to figure out how Figure 7.50 works. It is a simpler, more accessible arrangement than a bunch of flip-flops and logic gates. Note, incidentally, that we can close such an arrangement (i.e., make it go "in a circle") if we want to use it as a memory store.

7.3.3: Programmable Logic Arrays

With Programmable Logic Arrays (PLAs), we come on to examine the issue of if-then control in machines — that is, the matter of how, given a certain set of input data, the machine should determine what it does next. For example, "if such-and-such is zero, then stop" or "if both bits are 1, then carry 1". Abstractly, there is information coming out of some part of the machine which will tell us what we're to do next. This information hits some "sensors" (my own word, not the technical one), which tell us our present state. Once we know this, we can
act on it, for example, by telling an adder to add or subtract. This instruction, or more generally, set of instructions, will take the form of data coming out on a set of lines (Fig. 7.51):

![Diagram of Sensing, Control, Instructions](image)

**Fig. 7.51 A Generic Control Device**

The first stage in designing a device to do this is, obviously, to know what set of instructions are associated with a given sensory set. This is pretty straightforward. For example, we might represent the instructions as in Table 7.1:

<table>
<thead>
<tr>
<th>SENSE LINES</th>
<th>INSTRUCTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5</td>
<td>a b c d e f</td>
</tr>
<tr>
<td>1 1 0 1 0</td>
<td>1 0 1 1 0 1</td>
</tr>
<tr>
<td>1 0 0 1 1</td>
<td>1 0 1 1 1 0</td>
</tr>
<tr>
<td>0 1 1 0 1</td>
<td>0 1 1 0 1 0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Table 7.1: Example Instruction Set for a Control Device**

What this means is as follows. Each row in the left-hand column represents some configuration of bits on the sensor lines (of which there are five in this example). The corresponding rows on the right represent the bits sent out along the instruction lines (six, in this case), given the sensor set on the left. In this column, a 1 might mean "do something if the input from this line is 1" — such as "add" or "switch on light" — while a 0 might mean do nothing, or do something else — "leave state X as it is" or "switch off light". A very direct, and very inefficient, way of making a control system would be to simply store this table in memory, with the sensing lines as memory addresses, and the control lines as the contents of these addresses. Thus we would separately store the actions to be performed for all possible combinations of sense lines. Since the contents of this memory are to be fixed, we might as well store everything on a Read Only Memory (ROM) device. The only potential hitch in this otherwise straightforward procedure arises from timing: it is conceivable that some instructions could leave the ROM device before the rest, changing the state of the machine and confusing the sensing. The effect of this might be fed back into the ROM before it has completely dealt with its previous sense. This would be pretty bad if it happened, but is usually avoided (you should be way ahead of me here) by deploying clocked registers at each end of the memory to ensure that the retrieval and use of an instruction occur at different times (Fig. 7.52):

![Clocked ROM Control System](image)

**Fig. 7.52 Clocked ROM Control System**

When \( \phi \) is on, the sense lines feed through to the memory, which looks up the corresponding control signals. These latter signals cannot get out because \( \phi \) is off. Only when we can be sure that everything has settled down — that all the sensing information is in and that the instruction set has been chosen — do we switch \( \phi \) on. \( \phi \) has meanwhile gone off to freeze the memory input. With the external clock on, the instruction set can now get out and reach the rest of the machine without affecting the memory input. And so it goes on.

Thus, we see that control can be very, very simple. However, we are dissatisfied with this kind of approach because we would also like to be efficient! As a rule, stuffing our memory with \( 2^n \) entries is somewhat extravagant. Often, for example, two or more given input states will result in the same output state, or we might always filter a few sense lines through a
multiple-OR gate before letting them into the ROM. This would leave us with a high degree of redundant information in our table, and naturally enough we find ourselves tempted to eliminate the ROM completely and go back to basics, developing a circuit involving masses of logic gates. This was how things were done in the early days, carefully building immensely complicated logic circuits, deploying theorems to find the minimum number of gates needed, and so forth, without a ROM in sight. However, these days the circuits are so complex that it is frequently necessary — given the limitations of human brain power! — to use a ROM approach. But there are intermediate cases for which a ROM is not necessary because the number of possible outputs is small enough to enable a much more compact implementation using just a logic circuit — the set up is not too complicated for us to design. To illustrate one such instance, we shall examine a so-called "Programmable Logic Array" (PLA), something we first encountered in Chapter Two. This is an ordered arrangement of logic gates into which we feed the sense input, and which then spits out the required instruction set. Ideally, such an array would exhibit no redundancy. In a "black box" scheme, a generic PLA would have the form shown in Figure 7.53:

![Diagram of PLA](image)

**Fig. 7.53 The Generic PLA**

As can be seen, the PLA comprises two main sections: the "AND-plane" — formed exclusively from AND gates — and the "OR-plane" — formed exclusively from ORs. The planes are connected by a bridge of wires, which we label $R$. The inputs are fed into the AND-plane, processed and fed into the OR-plane by the $R$-wires. A further level of processing then takes place and a signal emerges as output from the OR-plane. This output is the set of "what next" instructions corresponding to the particular input.

Let us consider a case where we have three input lines, $A$, $B$, and $C$, and four output lines, $Z_1$ to $Z_4$. Each input, before being fed into the AND-plane, is split into two pieces — itself and its complement, for example, $A$ and NOT $A$. We now have a device that can manipulate each signal with NOT, AND, and OR — in other words, it can represent any logical function whatsoever. Let us pick a specific PLA to show the actual transistor structure of such an array. We have three inputs telling us the state of certain parts of the machine and four possible outputs — pulses that will shoot off and tell the machine what to do next. Now suppose that the output $Z$'s are to be given in terms of the inputs according to the following Boolean functions ($\lor =$ OR; $\land =$ AND; $'$ = NOT):

$$
\begin{align*}
Z_1 &= A \\
Z_2 &= A \lor (A' \land B' \land C) \\
Z_3 &= B' \land C' \\
Z_4 &= (A' \land B' \land C) \lor (A' \land B \land C')
\end{align*}
$$

(7.30)

It is not immediately obvious that the particular Boolean functions of $A$, $B$, and $C$ that we need to calculate the $Z$'s can be written as the product of a series of ANDs followed by ORs. However, it is in fact the case, as we demonstrated for the general logical function in Chapter 2. In this instance, an acceptable output $R_i$ from the AND-plane must only involve the ANDs and NOTs of $A$, $B$, and $C$. Thus we can define the $R_i$ as:

$$
R_1 = A, \quad R_2 = B' \land C', \quad R_3 = A' \land B' \land C, \quad R_4 = A' \land B \land C',
$$

(7.31)

and it is now straightforward to see that the $Z$-outputs can be written purely in terms of OR operations (or identities) on these $R$'s:

$$
\begin{align*}
Z_1 &= R_1 \\
Z_2 &= R_1 \lor R_3 \\
Z_3 &= R_2 \\
Z_4 &= R_3 \lor R_4
\end{align*}
$$

(7.32)

It is a general result that any Boolean function can be factorized in this way. The PLA for this function is shown in Figure 7.54:
This is a very practical approach. Of course, if you wanted more lines you would have to look up in a manufacturer's catalog which core arrays were available. Incidentally, note, from Figure 7.54 that the generic OR-plane is essentially the AND-plane rotated through a right angle.

**Problem 7.4:** Let me now give you an interesting problem to solve. This actually arose during the design of a real device. The problem is this: we would like to switch, that is, exchange, a pair of lines $A$ and $B$ by means of a control line, $C$. We are given $C$ and its complement $C'$ — they come shooting in from somewhere and we don’t care exactly where — and if the control $C$ is hot, then $A$ and $B$ change places: if $C$ is cold, they don’t. This is a variant of our old friend the Controlled Exchange. The circuit diagram we will use is that of Figure 7.56:

![Exchange Circuit](image)

To reiterate the rules: $C = 0 \Rightarrow A' = A, B' = B; C = 1 \Rightarrow A' = B, B' = A$. You should be able to see how it all works. Here is what I want you to do:

(a) Draw a stick figure with the correct conventions for diffusion, poly and metal (Hint: the inputs $A$ and $B$ are fed in on metal lines).

(b) Draw a legitimate layout on graph paper, obeying the $\lambda$ design rules.

(c) This circuit can easily be amended to allow for more $A, B$... inputs simply by iterating its structure (and extending the $C, NOT C$ lines). Suppose now that we have eight input pairs coming in from the top. There are only fourteen $\lambda$'s available horizontally for each pair, and sixteen or twenty extra $\lambda$’s on the borders for about $132\lambda$ total width. But we are allowed $150\lambda$ deep. Now we...
want the A's and B's to go out of the circuit in metal too. Can you design it? You may assume more C's from the left if you want.

7.4: Further Limitations on Machine Design

It doesn't take much thought to realize that one of the most important components of any computer is wire. We're so used to treating wires — more generally, transmission paths, including polysilicon lines — in an idealized way that we forget they are real physical objects, with real physical properties that can affect the way our machine needs to be designed. In this final section, I'd like to look at two ways in which wires play an important role in machine design. The first relates to how wire lengths can screw up our clocking, the so-called "clock skew" problem; the second to an even simpler issue, the fact that wires take up space, and that when we build a computer, we'd better make sure we leave enough room to get the stuff in!

7.4.1: Clock Skew

Let's return to our discussion of clocking the general PLA. Remember, we employed two clock pulses, \( \Phi_1 \) and \( \Phi_2 \), taking the general form:

\[
\Phi_1 \quad \Phi_2
\]

\text{SETTLE TIME}

Fig. 7.57 The PLA Clock Pulses

The idea is that we feed data into the PLA while \( \Phi_1 \) is on, and then let things settle down for a while — let the logic gates go to work and ready their outputs, and so on. This is the reason for introducing a delay time, and not simply having the two clocks complementary. Then, we switch on \( \Phi_2 \), and during this time we allow the data to spew out. This sounds all very straightforward and simple.

However, in a real machine, there can be problems. For a start, charging up the gates of circuit elements takes a non-zero time, and this will introduce delays and time-lags. Also, of course, the clock signals are current pulses sent along wires — metal, polysilicon, whatever — and these pulses will take a finite time to travel. A clock pulse sent along a short wire will reach the end before a pulse sent along a long wire. We can actually model a simple wire in an interesting way as an infinite sequence of components as shown in Figure 7.58 (which in the finite case could be taken as modeling a chain of pass transistors):

![Diagram](image)

**Fig. 7.58 An Infinite-Limit Model of a Simple Wire**

We have a line of resistors interspersed with capacitors. If we assume we have infinitely many small capacitors and resistors, bunched up infinitely closely, then we effectively have a wire, with a resistance per unit length of \( R \), and a capacitance per unit length of \( C \). Now what we want to do is to load up one end of the line (which needn't be metal — it could be polysilicon), and wait for the signal to propagate along to the other end. Let the distance along the wire from the origin be \( x \). At each junction we can define a potential \( V(x) \), and a current flowing into it, \( I(x) \). Taking the limit as \( \Delta x \rightarrow 0 \), elementary math and electricity gives us the set of equations:

\[
\frac{\partial I(x)}{\partial x} = - \frac{C}{R} \frac{\partial V(x)}{\partial t} \tag{7.33}
\]

\[
\frac{\partial V(x)}{\partial t} = - \frac{I(x)}{R} \tag{7.34}
\]

\[
\frac{\partial^2 V(x)}{\partial x^2} = \frac{1}{RC} \frac{\partial V(x)}{\partial t} \tag{7.35}
\]

defining \( \tau = RC \). Equation 7.35 is an example of the diffusion equation. Charge
flows in at one end and diffuses through the system. The general form of the solution in terms of Green's functions is well-known. With our boundary conditions the solution is:

\[ V(x,t) = \exp(-x^2\tau/4t) \]  

(7.36)

It is easy to see from this that if the overall length of the wire is \( X \), then the time to load the wire scales as \( X^2 \). For 1 mm of polysilicon, this time comes to 100 ns. For 2 mm, it is 400 ns. This is a pretty lousy line, especially if you're more used to transmission lines for which the loading time is proportional to the distance. Metal, however, has such a low resistance that the load time is relatively much shorter — so if you want to send a signal any great distance, you should put it on metal.

The issue of clocking is of such importance to computing (indeed, much more important than you'd think given how little I've talked about it) that we are naturally encouraged to explore other avenues, other ways of controlling our information flow. The problem with standard way, so-called synchronous clocking — the only type we've considered so far — is that in designing our machine we have, at each part of the system, to allow for the "worst case scenario". For example, suppose we have to take an output from a complex adder that could take anywhere from, say \( t \) units of time up to \( 5t \) to show. Now even if the output rises through after \( t \) units, we still have to put our machine on hold for at least \( 5t \) just on the off-chance that we get a slow decision. This can lead to severe time inefficiencies. Now, another way to design machines — although one which is not yet used commercially — is an "asynchronous" method: we let the adder control the timing. Let it tell us when it's ready! It carries out its computation, and then sends a signal saying it's ready to send the data. In this way, the timing is controlled by the computing elements themselves, and not a set of external clocks.

Interestingly enough, a little thought will show you that even synchronous systems have asynchronous problems of their own to solve. For example, consider what happens if such a machine has to accept data from a keyboard, or another machine hooked up to it? Keyboards don't know anything about the "right time" to send in the data! We have to have a buffer, a little box which lets data into the machine only if the machine clocks are in the right state. It has to make a decision: whether to accept the data right now, or to wait until the next cycle, as the data came in too late. The fact that a decision has to be made introduces the theoretical possibility of a hang-up caused by the data coming in at just such a time that the buffer is not quick enough to make a decision — it can't make its mind up. It's a fascinating problem, and one well worth thinking about.

### 7.4.2: Wire Packing: Rent's Rule

Up until now we've been discussing transistors, VLSI, and this and that — and we think that's the hard part of machine design. But whenever you get to the end of a big design, and you set out to build it, you'll discover that all the algorithms and so forth that you've worked out are not enough — something always ends up getting in the way. That something is wires. We look at that now.

I would like to emphasize that wires represent a real problem in system design. We've discussed one difficulty they cause: timing problems resulting from the finite time it takes to load them. But another problem is that the space needed for the wiring, connecting this chip to that and the other\(^3\) greater than that needed for the functioning components, like transistors! Now there is no guarantee that wires will forever reign supreme: with optical fibers, for example, we can send multiple messages down single wires by using light of differing frequencies. People occasionally break down and begin to dream, having brilliant ideas such as that of building a machine, by analogy with our broadcasting system, in which each component radiates light of a particular color (say via a LED), which is broadcast throughout the machine to be picked up and acted on by frequency-sensitive components. However, at this moment in time the predominant method of current transmission is via wires, and I'd like to spend some time discussing them. Specifically, I want to address the question of how much wire we might need for a generic design.

Now there's very little I can say here about wire-packing — they're just wires, after all — but it turns out that there is an empirical rule, Rent's Rule, which purports to shed some light on this question. It's a curious rule, and I can't really vouch for how accurate it is in general, but it appears to be the case based on the experience of IBM. Here's how it goes. Let us suppose we have a unit, like a circuit board, and suppose further that we can segregate elements on the unit into "cells" — not too big, not too small. These cells could be

\[^3\]I am not now concerned so much with the "wires" on the chips, but those connecting chips together — real bunches of wires that interfere with how closely chips can be stacked, and so on. [RPF]
individual chips, for example. Now suppose that:

1. Each cell has \( t \) pins, or terminals,
2. \( N \) cells make up a unit, and
3. The number of terminals, or output pins, on our unit is \( T \).

Needless to say, these numbers have to be interpreted with a certain latitude. Let us suppose we try to connect everything up so that the components talk a lot, that is, we try to minimize the wire length by packing. Then Rent's Rule states that:

$$ T = tN $$  \hspace{1cm} (7.37)$$

where \( 0.65 \leq r \leq 0.70 \). (Since this inequality is only approximate we will take \( r = 2/3 \).) In other words, it claims to relate the number of wires leading to and from the unit (\( T \)) to the density of cell packing on the unit (\( = N \)). A naive first question to ask might be: why not just \( T = tN \)? Well, for the obvious reason that many of the wires will be internal to the unit (Fig. 7.59):

![Fig. 7.59 Schematic Depiction of Fundamental Cells on a Board](image)

We can see how an expression such as that in Equation 7.37 might arise by moving "up" in our hierarchy of units and cells. We have considered units on which cells were joined together. We now consider units joined together. So let us imagine that we have a bigger unit, a "superunit", the cells of which are the units bearing the original cells. Suppose this superunit contains \( M \) units. Now, because we have set no fundamental level of analysis, there must be some consistency of scaling between these two situations. Let the number of terminals on the superunit be \( T'_s \). Clearly, each of the \( M \) units will have \( T \) terminals. Then, Rent's Rule would say:

$$ T'_s = TM' $$  \hspace{1cm} (7.38)$$

However, returning to our initial level of analysis, we can treat the superunit as comprising \( NM \) of the original cells, each of which has \( t \) pins. Using Rent's Rule again, we get:

$$ T'_s = t(NM) $$  \hspace{1cm} (7.39)$$

Clearly, using (7.37), we see that (7.38) and (7.39) agree so that Rent's Rule has the correct scaling properties. This is very important.

Note that this treatment tells us nothing about the value of \( r \) (although it should be obvious from the form of the rule and the discussion following (7.37) that \( r \) would have to be less than 1). Where does this exponent come from? Well, you should remember that the value that was chosen was derived from experience, and this experience must have been influenced by problems of geometry in designing and connecting up logic circuits. That is, while it might be enticing to think that there is some neat logical reason for the value of \( r \), that it might drop out of a pretty mathematical treatment, it's possible that it's an artifact of conventional design approaches. But for the moment, with this caveat in mind, let's assume it is true in the general case and see what it might teach us about wire packing.

Let's go back to the two-dimensional case. Suppose we have a square board, of side length \( L \) cm, say. Let this be the unit. We pack it with cells, each of length \( l \) cm; so we can write the number of cells on the board as \( N = (L/l)^2 \) (Fig. 7.60):

![Fig. 7.60 A General Two-dimensional Unit](image)
Now suppose that there is a restriction on how many terminals we can fit on each of the cells — that we can only place them so closely together. Let the maximum number of pins per cm on a cell perimeter be $s_p$. Suppose that there is also a minimum pin separation for the board terminals, with the maximum number per cm of perimeter being $s_e$. Rent's rule then becomes:

$$T = (4s_pL) = tN^r = (4s_eL/L)^{2r}$$

(7.40)

and we have:

$$s_B = s_e(L/L)^{2r-1}.$$ 

(7.41)

It is clear from this that if $r > 1/2$ then, as we increase $L$, we need more and more pins per inch on the perimeter to take care of all the junk inside it. Therefore, we'll eventually get a jam. So as we build the machine bigger, the wiring problem becomes more serious. At the heart of this is the fact that the length of the perimeter varies as the square root of the area but the number of terminals (according to Rent) goes as the $(2/3)^{rd}$ power, a much slower scaling factor. A big incompressible mess of wires is unavoidable, and we have to increase the spacing between cells, leading to more boards, and increasing the spacing between boards, and so on, to make room. Now interestingly enough, if we were to rework this argument in three dimensions, rather than two, we get a different result: in 3-D we replace the perimeter by the surface area ($length^2$) and the area by the volume (length$^3$). Clearly, the former scales as the latter to the $(2/3)^{rd}$ power, the same as the number of terminals! So in 3-D we could just make it — we could always use the same density of pins over the surface, and we wouldn't get into a wire hassle. The problem with this sort of 3-D design, of course, is that for anyone to look at it — to see what's going on — they have to be able to get inside it, to get a handle or some tools in. At least with two dimensions we can look at our circuits from above!

Still assuming the validity of Rent's Rule, we can ask another interesting question. What is the distribution of wire lengths in a computer? Suppose we have a big, two-dimensional computer, a board covered in cells and wires. Some of the wires are short, maybe going between adjacent cells, but others may have to stretch right across the board. A natural question to ask is: if we pick a wire at random, what is the chance that it is of a certain length? With Rent's Rule we can actually have a guess at this, after a fashion. Return to the two-dimensional case shown in Figure 7.60, and now take $L$ to be the side-length of some arbitrary unit on the board. We can consider any wires connecting cells within this unit to other cells within it to be less than $L$ in length. This is not strictly true, of course, as we might have diagonals. However, if we just deal with orders of magnitude, we shall assume we can neglect this subtlety. There will also be wires going out of this unit and hooking up to other units on the board. We take these to be longer than $L$. From Rent's Rule, we can calculate the number of wires of length greater than $L$ — this will be $T$, the number of terminals on the unit, given in this case by:

$$T = t(L/L)^{2r}.$$ 

(7.42)

We can now calculate the probability that a random wire will have a length greater than $L$. It is just the right hand side of Equation 7.42 divided by the total number of wires on the unit. This is easily seen to be $\propto (L/L)^{2r}$. So, if the probability of a wire having length greater than $L$ is $P(L)$, we clearly have:

$$P(L) \propto (L/L)^{2r-2}, \text{ i.e., } P(L) \propto 1/L^{2r}.$$ 

(7.43)

using $r = 2/3$ in Rent's expression.

We can take these statistics further. Introduce the probability density $p(L)$, which is defined in the standard way — the probability of finding the wire length to lie between $L$ and $L + \delta L$ is $p(L)\delta L$. Then we have:

$$P(L) = \int_0^\infty p(L')dL'$$

(7.44)

with

$$p(L) = dp/dL = 1/L^{5/3}.$$ 

(7.45)

Let us compute a quantity of particular interest, the mean wire length. By conventional statistical reasoning, this is:

$$\overline{L} = \frac{\int_0^L Lp(L)dL}{\int_0^L p(L)dL}$$

(7.46)
Note that we have tinkered with the limits of integration in (7.46); if we let the length $L$ range from zero to infinity, then the numerator gives us trouble at its upper limit (infinity), as the integral is of positive dimension in $L$, and the denominator gives us trouble at its lower limit (zero), as its integral is of negative dimension in $L$. We hence set an upper limit for $L$, $L_{\text{max}}$, and also set a natural lower limit, the cell-size $l$. The reader can perform the integrals in (7.14) to obtain the mean wire length. The answer is:

$$2l(L_{\text{max}}/R)^{1/3}.$$  \hfill (7.47)

Note that this quantity is divergent: the bigger our machine (its size being given roughly by $L_{\text{max}}$), the bigger the mean wire length. No surprise there. However, note how it is the cell-size, $l$, that is calling the shots in (7.47); the mean length scales half as quickly with machine size as it does with cell-size (which is equivalent to cell spacing in our model). If we space our cells a little further apart, the size of the machine must balloon out of proportion.

It used to be said in the early eighties that a good designer, with a bit of ingenuity and hard work, could pack a circuit in such a way as to beat Rent’s Rule. But when it came to the finished product, something always came up—extra circuits were needed, a register had to be put here, an inductance there—and, when the machine was finally built, it would be found to obey the Rule. When it comes to the finished product, Rent’s Rule holds sway, even though it can be beaten for specific circuits. Nowadays, we have “machine packing programs”, semi-intelligent software which attempts to take the contents of a machine and arrange things so as to minimize the space it takes up.

---

**A Final Comment from the Editors**

What remains to be said? Well, there are a few scattered lectures on the Feynman tapes that we have not attempted to put into publishable form. These lectures cover interesting topics such as the physics of optical fibers and the possibilities for optical computers. In these cases, however, technical developments have been so substantial that we have thought it best to leave topics such as these for an expert up-to-date overview in the accompanying volume. With this caveat, the lectures contained in this book constitute an accurate representation of Feynman’s overview of the field of computation. Moreover, these lectures, by his choice of topics, also demonstrate the subject areas that he felt were important for the future.