11
Power and Temperature

In the last two parts of the book we have studied the design of a modern out-of-order processor, on-chip network, and memory system in detail. We started with a simple 5-stage in-order processor, and increased its performance several times by using sophisticated optimisations to enhance all aspects of the design. The natural question that we need to ask now is, "Are there any limits to doing this?" If there were indeed no limits, then we can keep on proposing newer and better optimisations, and keep on increasing the performance. However, nature has its way of applying brakes. For at least the last 15 years, the single largest factor that has kept processors from scaling in terms of additional features and complexity is the issue of high power consumption and consequent temperature rise. Both these issues are connected, and have effectively stymied many design features.

Let us motivate the reader by providing some facts that indicate the extent of the problem. While running a floating point application the die temperature of a CPU exceeds 100°C, which is enough to boil a cup of water, or fry an egg. If the CPU remains at this temperature long enough, then the wires and transistors can get seriously damaged. As we shall see in Chapter 12, the probability of failure is an exponential function of the die temperature. The problem of high temperature not only bedevils large processors that are used in desktops and servers, it is also important for mobile phones. Imaging using a mobile phone that feels like a warm face pack in a hot and humid climate. A consistently high temperature increases the probability of the battery of the mobile phone exploding!

Let us now look at energy/power consumption. We all have been in situations, where we just have 3% of the total battery charge left in our phones, and we have to either finish a call, or go back home using Google Maps™, and there is no charger in sight. In such cases, one would wish that she had a phone that consumed very little energy such that she could finish all her work with the remaining energy in the battery. Nobody likes to charge their phones or laptops very frequently.

In the case of large server processors, we might be misled to believe that temperature rise and power consumption are not an issue because the servers are isolated in a server room or a data centre. Since we are not physically present in the building, temperature rise is not an issue, and since there is no battery, energy usage is also not a concern. However, this is not correct. The energy required to maintain the optimal temperature in a large data centre is roughly 20-40% of the total energy consumption. For a large corporation such as Facebook® or Google® this can translate to billions of dollars. Even for a much smaller setup, the electricity bill is a large part of the operating expenditure of a data centre, and thus reducing the energy consumption by even 10% is significant. Consider the fact that in 2017, data centres in the US consumed 90 billion units (kilowatt-hours) of energy, and globally 3% of the energy was used to power data

centres. This number is growing and is expected to double by 2021, which will put a significant strain on the planet's resources [Danilak, 2017].

There are two major components of power consumption: dynamic power and static (or leakage) power. The dynamic power is dependent on the clock frequency, and the activity inside the CPU. However, temperature-dependent leakage power is dissipated by transistors via mechanisms that are normally considered to be associated with no power dissipation. As of 2020, leakage power is roughly 20-40% of the total power dissipation in high-end processors. The relationships between power and temperature are complex: an increase in dynamic power increases the total power, which increases the temperature, which increases the leakage power, which increases the total power, which increases the temperature (and so on). There is thus a cyclic dependence between power and temperature, which means that a small increase in power leads to a small increase in temperature, which further leads to a small increase in power, and so on. In most cases, this process converges; however, in some cases it does not and this leads to a very dangerous condition called *thermal runaway*. The system needs to be immediately shut down if such a situation arises. Let us thus summarise what we have learnt up till now.

Way Point 14

1. *The dynamic power consumption of a processor is dependent upon its design and the workloads that execute on it.*
2. *An increase in dynamic power increases the temperature, which increases the leakage power. The leakage power further increases the temperature. There is thus a cyclic dependence between temperature and leakage power.*
3. *It is necessary to limit the power consumption for controlling the peak temperature.*

We shall proceed as follows in this chapter. We shall first look at methods to model power and temperature in Sections 11.1 and 11.2 , then describe methods for power management in Section 11.3, and finally conclude by briefly discussing methods to reduce temperature in Section 11.4.

11.1 Power Consumption Model

There are many power dissipation mechanisms in modern processors. The mechanisms can broadly be divided into two types: dynamic power dissipation and static power dissipation (also known as *leakage power* dissipation) [1]. Dynamic power refers to traditional power consumption where power is consumed because of the current flow between transistors in the circuit whenever an input changes. The resistive loss in the different circuit elements leads to this kind of power dissipation.

Starting from the early 2000s, static power (or leakage power) has increasingly become more important. In the ideal scenario, we only expect current flow between the drain and source of a transistor when it is switching its state. However, this is often not the case. For example, in modern transistors and capacitors there is a small amount of current flow between the drain and source even in the off state. This is not expected, and is thus categorised as *leakage power*. Similarly, we never expect any current flow between the gate of a transistor and the body; however, this also is not true in practice. When we integrate the effects of such small sources of power dissipation over billions of transistors, the total amount of power dissipated can be significant. As of 2020, static power accounts for roughly 20-40% of the overall CPU power budget. The important point to understand is that even though the leakage current and the resultant leakage power in a

[1] In this book the terms *leakage power* and *static power* will be used synonymously and interchangeably.

given transistor are at least an order of magnitude lower than the dynamic power, they become significant when we consider the cumulative sum across the billions of transistors. Additionally, note that a rather small fraction of transistors are active or switching their state at a given point of time. The total number of transistors is actually an order of magnitude more. If we take both of these effects into account, the leakage power becomes disproportionately important.

It is important to mention that to understand the rest of the discussion in this section, it is necessary to go through Section 7.3.3, and also understand the basic structure of a transistor. For the latter, interested readers can refer to a standard text on digital electronics [Taub and Schilling, 1977].

11.1.1 Dynamic Power

The Notion of Voltage Transitions

As we discussed in Section 7.3.3, a lot of circuits in modern processors including memories can be approximated as RC networks. Every circuit element can be replaced with an equivalent RC model and the subsequent calculations are reasonably accurate – at least to get a broad estimate and get an idea of the overall trends. Consider a pair of two inverters in series as shown in Figure 11.1. The gates of the two transistors in inverter I_2 have an associated gate capacitance. These capacitors need to be charged if we need to set the voltage at the gates of I_2 to a logical 1. If they were at a logical 0, then current needs to flow via the PMOS transistor (T_1) of inverter I_1 to charge these capacitors. Whenever there is a current flow in a circuit, it leads to power dissipation across the resistive elements. Even though we cannot see any resistor in the circuit diagram of Figure 11.1, however the equivalent circuit of these transistors has resistive elements (described in detail in Section 7.3.3). In general, whenever there is a current flow across any device (transistor or capacitor), we shall encounter an associated resistance. We observe a resultant power dissipation – I^2R loss.

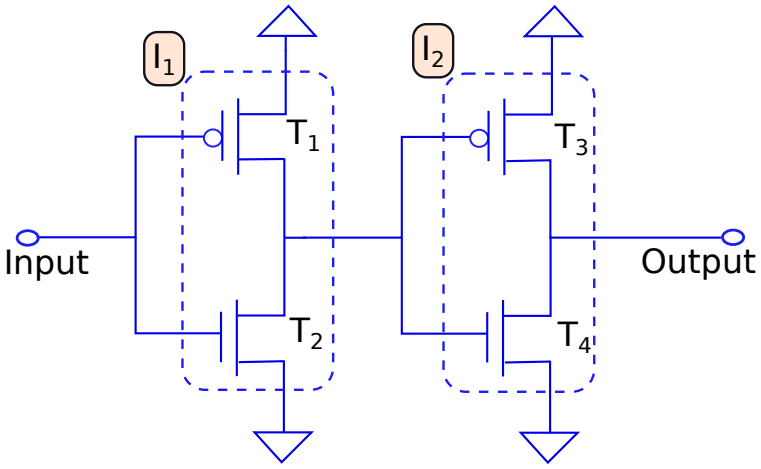

Figure 11.1: Circuit with two inverters in series

We can make a generalised observation here. Whenever there is a current flow in a circuit to effect a transition in the voltage level (logical $0 \rightarrow 1$, or $1 \rightarrow 0$), there is some amount of dynamic power dissipation because of the passage of current through resistive elements in the circuits and in the devices themselves. Hence, to compute the dynamic power dissipation of a circuit, we need to locate all the voltage transitions in a cycle, compute the current flow required to effect the transitions, and then compute the sum of the resistive (I^2R) losses throughout the circuit.

Let us consider a functional unit that has n input and m output terminals. In a cycle if none of the n inputs change, which means that they maintain their previous values, then there will be no transitions within the circuit because none of the inputs changed. If a point inside the circuit was charged to a logical

1, it will continue to maintain that voltage (assuming no leakage). Likewise is the case for points at a logical 0. There will be no transitions because the inputs did not change, and thus the dynamic power dissipation will be zero. We can thus conclude that the dynamic power dissipation of a circuit is dependent not only on its inputs but also on the previous inputs and the previous state of the entire circuit. Essentially, the only thing that we care while computing dynamic power is the locations of all the *voltage transitions in the circuit*.

The power dissipated per voltage transition also depends on the values of resistances and capacitances in the vicinity of the point at which there was a voltage transition. Given the sheer complexity of this problem with the multitude of variables, it is necessary to analyse simple situations to understand the broad trends that affect power consumption. Let us thus look at a simple model for computing the effect of voltage transitions.

Mathematical Model of Power Dissipation

Consider the circuit shown in Figure 11.2. It has a resistor and a capacitor in series.

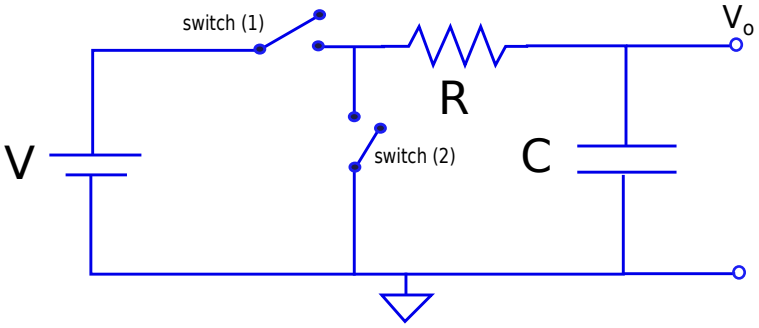

Figure 11.2: A simple RC circuit

Let's say at $t = 0$, we connect the voltage source V to the circuit. The current across the capacitor at any point of time is $I = CdV_o/dt$, where V_o is the voltage at the upper terminal of the capacitor. From Kirchhoff's laws we have $V_o = V - IR$.

Let us solve for the current I.

$$
\begin{aligned}
& I = C\frac{dV_o}{dt} \\
& \Rightarrow I = C\frac{-dIR}{dt} \\
& \Rightarrow I = -RC\frac{dI}{dt} \\
& \Rightarrow \frac{dI}{I} = -\frac{dt}{RC} \\
& \Rightarrow \int \frac{dI}{I} = -\int \frac{dt}{RC} \\
& \Rightarrow ln(I) = -\frac{t}{RC} + k \\
& \Rightarrow I = e^{-\frac{t}{RC}} \times e^k
\end{aligned}
\tag{11.1}
$$

Here, k is the constant of integration. At $t = 0$, $I = \frac{V}{R}$, because the voltage across the capacitor is 0. We thus have $e^k = \frac{V}{R}$.

The final solution is given by

$$I = \frac{V}{R} e^{-\frac{t}{RC}} \tag{11.2}$$

Let us now compute the total energy provided by the voltage source in a typical charging cycle.

$$\begin{aligned} P_{tot} &= \int V \times I \\ &= V \times \frac{V}{R} \int_0^\infty e^{-\frac{t}{RC}} dt \\ &= CV^2 \int_0^\infty e^{-\frac{t}{RC}} d\frac{t}{RC} \\ &= CV^2 \end{aligned} \tag{11.3}$$

Thus the total energy that is provided is CV^2. Out of this energy, some of it is dissipated as heat via the resistor and the rest is stored as energy across the capacitor. The final voltage across the capacitor is equal to V. From basic physics, we know that the energy stored across a capacitor is equal to $\frac{1}{2}CV^2$. This means that the energy lost as heat via the resistor is equal to the difference: $\frac{1}{2}CV^2$.

Now while discharging, the current flows along the discharge path (via switch(2) in the figure with switch(1) turned off). The capacitor will lose all of its charge and its stored energy. By the simple rules of energy conservation, we can deduce that all this energy will be dissipated as heat via the resistor. This energy is $\frac{1}{2}CV^2$.

We can thus make several important conclusions.

- The total energy provided by the voltage source to charge the capacitor is equal to CV^2.
- Half of this energy is dissipated as heat while charging the capacitor, regardless of the value of the resistance, R.
- The rest of the energy is also dissipated as heat while discharging the capacitor. This is also equal to $\frac{1}{2}CV^2$.
- The total energy dissipated in a charge-discharge cycle is equal to CV^2.

Given that $\frac{1}{2}CV^2$ units of energy are dissipated in every cycle (charging or discharging), the power consumption is equal to $\frac{1}{2}CV^2/\tau$, where τ is the cycle time. $1/\tau = f$, where f is the frequency. Thus, we arrive at the most important equation in dynamic power consumption. The dynamic power consumption, P_{dyn}, is given by

$$P_{dyn} \propto CV^2 f \tag{11.4}$$

Here, the proportional sign $\propto$ is very important. For a simple RC network, it is equal to half assuming that in every cycle either we charge or discharge the capacitor. However, in a complex circuit we might have hundreds of transistors and many of them would be switching their state in a cycle. Additionally, all the transistors will not be switching their state in a given cycle. We thus need to factor in the level of activity of the circuit. Hence, we need to modify Equation 11.4 as follows:

$$P_{dyn} \propto \beta CV^2 f \tag{11.5}$$

Here, β, is the activity factor, which is a number between 0 and 1. A value of "1" indicates that all the transistors in the circuit change their state in any given cycle, and a "0" means that none of the transistors change their state. The activity factor basically determines the mean fraction of transistors that change their state every cycle.

If we know the constant of proportionality, then we can find the exact value of power for the functional unit, and we can also find the total power dissipation of the processor by summing up the values for each functional unit.

However, even without knowing the constant of proportionality, Equation 11.5 is extremely valuable. If we know the power for a given level of activity (β), we can easily estimate the power consumption at a different level of activity. This will tell us how the power consumption of a functional unit changes as we vary the activity. Using this insight we can design micro-architectures that modify β dynamically such that the power consumption changes.

Sadly changing the activity factor, β, isn't always possible. It is far more practical to tune the voltage and frequency, which are arguably the most powerful knobs to contain power consumption.

Dynamic Voltage and Frequency Scaling (DVFS)

The key learning from Equation 11.5 is that the power consumption is dependent on the voltage and the frequency. It is possible to change the power consumption by tuning the voltage and frequency, which is known as Dynamic Voltage and Frequency Scaling (DVFS).

In the early days when the behaviour of transistors was governed by very simple equations, the time it took for a transistor to change its state was roughly inversely proportional to the supply voltage. Hence, we assumed that the relation $V \propto f$ is roughly true. This can be intuitively reasoned as follows. More is the voltage, higher is the current, and thus it will take a proportionately lesser amount of time to charge and discharge capacitors in the circuit. However, this simple relationship has ceased to hold. The relationship is now governed by the Alpha Power Law [Sarangi et al., 2008].

$$f \propto \frac{(V - V_{th})^{\alpha}}{V} \tag{11.6}$$

V_{th} is the threshold voltage and α is a constant. We observe that if $V \gg V_{th}$ and $\alpha = 2$, then the traditional relation $f \propto V$ holds. However, in modern technologies V/V_{th} is between 4 and 6, which is a comparatively much smaller ratio. Secondly, the value of α has reduced. It is now between 1.1 and 1.5 [Sarangi et al., 2008].

In practice, modern processors do not use such formulae. They have a table that contains a few voltage and frequency settings. These are known as the DVFS settings. The processor is only allowed to operate in any one of these settings.

ED^2 Metric

Let us first understand the traditional view of thinking when people assumed that $f \propto V$. We can still use this model to derive many insights and results. Assuming a constant activity factor and no leakage power. We have,

$$\begin{aligned} P_{dyn} &\propto CV^2 f \\ &\propto Cf^2 \times f \quad (V \propto f) \\ &\propto f^3 \end{aligned} \tag{11.7}$$

We thus have: $\boxed{P_{dyn} \propto f^3}$.

The key result that we derive is that the dynamic power is proportional to a cube of the frequency. If we double the frequency, the power will increase by 8 times. This explains why the processor frequency was not able to cross 4 GHz: the power consumption was prohibitive.

Let us also look at one more key metric that has survived from the old days. Consider the product ED^2, which is alternatively $P_{dyn}D^3$. Here, E is the energy, and D is the delay; we are assuming that there is no leakage power. If we consider the power for the entire system, we need to sum up the per-component dynamic power. Since the factor $V^2 f$ is common to all the power values for the components we can write that

$P_{dyn} \propto V^2 f$, where the constant of proportionality incorporates all the activity factors and capacitances. Let us make one more simplifying approximation that is not exactly correct, however it still can be used to derive an approximate relationship between the frequency(f) and the delay(D) (program execution time). We assume that $f \propto 1/D$, which is approximately true for programs that are CPU-bound – we need to discount the effect of the main memory latency, which does not scale with the frequency.

We thus have,

$$\begin{aligned} ED^2 = P_{dyn} D^3 &\propto V^2 f \times D^3 \\ &\propto f^3 \times D^3 \quad (V \propto f) \\ &\propto \frac{f^3}{f^3} \quad (D \propto 1/f) \\ &\propto 1 \end{aligned} \tag{11.8}$$

The key result is that ED^2 is a constant with respect to any DVFS based scaling. In other words, regardless of the values of the frequency and voltage, ED^2 remains a constant as long as the voltage and frequency are scaled according to our assumptions. ED^2 is essentially a function of the activity factors, the capacitances, and other artefacts of the architecture and the circuit. It is an inherent property of an architecture, and is immune to voltage and frequency scaling.

This discussion answers a very important question in computer architecture. Let us say that we have two designs: A and B. As compared to design A, B is 10% faster yet consumes 20% more energy. Which design is better? We previously did not have any method to answer this question, however we can answer this question now using the ED^2 metric (see Example 13). The lower is the ED^2, better is the design.

Example 13
We have two designs: A and B. As compared to design A, B is 10% faster, yet consumes 20% more energy. Which design is better in terms of its ED^2?

Answer: *Let the ED^2 of design A be 1 (arbitrary units). Then the ED^2 of B is 1.2×0.9^2, which is equal to 0.97. The ED^2 of design B is lower, hence it is a better design.*

In all of this discussion, one fact should always be kept in mind that such metrics are approximate and date back to an era where many of these assumed relationships used to strictly hold. However, the reason that they are still used today is because they provide a very intuitive method of approximately comparing two designs that have different power consumption and performance values.

Short-Circuit Power

Let us now specifically discuss a minor component of the dynamic power, which is known as the short-circuit power. It is typically 10% of the overall dynamic power [Li et al., 2009], however in some designs where the threshold voltage is relatively high as compared to the supply voltage, this can increase to roughly 25% [Nose and Sakurai, 2000].

A typical inverter as shown in Figure 11.3 has two transistors: the pull-up PMOS transistor and the pull-down NMOS transistor. When it is going through an input transition there is a brief period when both the transistors are conducting. During this time there is a short circuit between the supply and the ground. The corresponding current that flows is known as the *short-circuit current*, and the corresponding power is known as the *short-circuit power*.

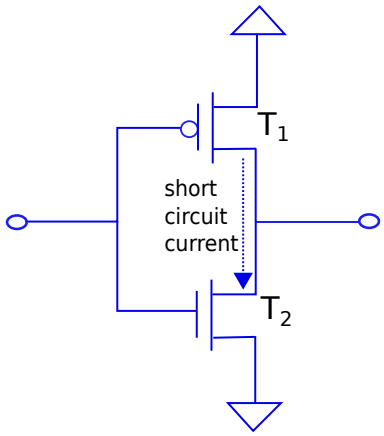

Figure 11.3: A single CMOS inverter (note the short circuit current)

11.1.2 Leakage Power

Sadly dynamic power dissipation is not the only power dissipation mechanism. 20-40% of the overall power dissipation is accounted for by leakage power in high-end processors.

Leakage (or static) power is any source of power dissipation that is not present in an ideal transistor. For example, in a typical NMOS transistor we do not expect any current to flow from the gate to the substrate, or from the drain to the substrate. However, real devices have such non-idealities, and thus they *leak* some power via interfaces that are assumed to be perfect insulators in simplistic models.

If we add the effect of these small sources of power dissipation across all the transistors in the chip, then the net power dissipation can be significant. Leakage is particularly an issue in the large L2 and L3 caches. There are several different leakage power dissipation mechanisms, and their relative predominance depends on the transistor technology.

Overview of Different Leakage Mechanisms

Figure 11.4 shows the diagram of an NMOS transistor. It shows six different leakage mechanisms: currents I_1 to I_6 in the figure. Let us discuss them one by one.

I_1: P-N Junction Reverse-Bias Current

A p-n junction is said to be reverse-biased if the p side is connected to a low voltage source, and the n side is connected to a high voltage source. Most of the time during the operation of an NMOS transistor, this is the case for the drain to body junction. The drain voltage (connected to the n-type doped region) is much higher than the body voltage. This makes the drain-body junction reverse-biased. The main component of the reverse-biased current is band-to-band tunnelling (BTBT).

Band-to-Band Tunnelling

Typically in today's transistors, there is a very high electric field across the drain-to-body p-n junction. This high electric field causes electrons to "tunnel" from the valence band of the p-type doped region to the conduction band of the n-type doped region.

It is necessary to explain the terms *valence band* and *conduction band* here. As per the rules of quantum mechanics, an electron in a system of atoms can only occupy a fixed set of energy levels. It is not possible for it to occupy any intermediate energy level because quantum state is discrete. Over the long run any material's electrons will migrate into the low-energy states. However, by Pauli's exclusion principle two electrons cannot have the same state. As a result, some electrons will have to be in high-energy states. Now,

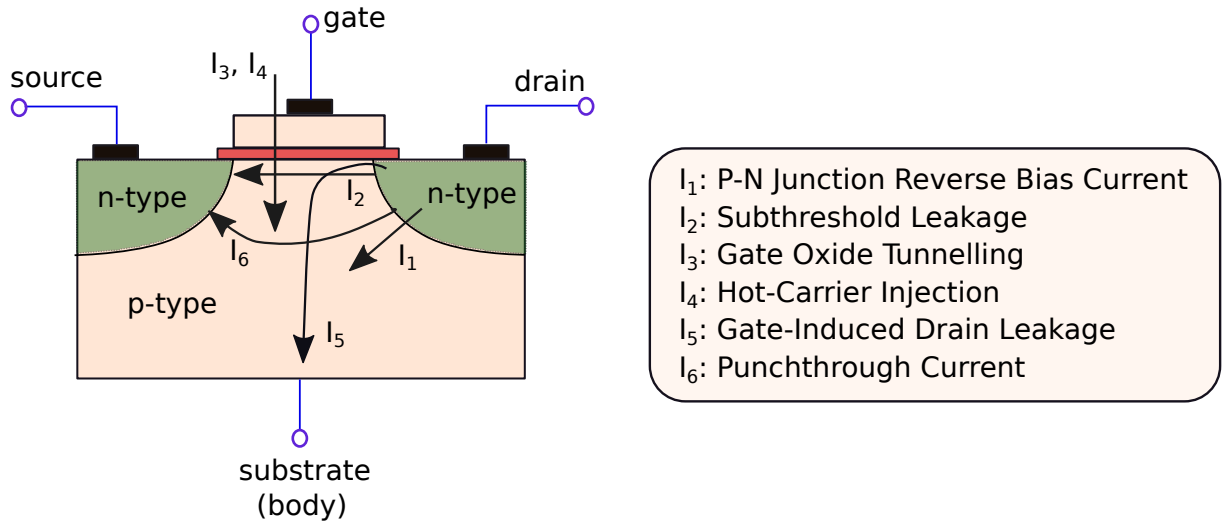

Figure 11.4: Different leakage mechanisms

every solid-state material can be characterised by the *Fermi level*, which is defined as follows. When the temperature is 0 K (absolute zero) all the energy levels below the Fermi level are occupied with electrons, and no energy level above the Fermi level is occupied with an electron. At higher temperatures, a fraction of electrons occupy energy levels *above* the Fermi level.

The *valence band* is a range of energy levels just below the Fermi level. Likewise, the *conduction band* is a range of energy levels just above the Fermi level. In semiconductor materials, typically there is a band gap between the valence and conduction bands. An electron can only act as a charge carrier if it gets transferred to the conduction band, and likewise is the case for holes. If the energy supplied to an electron because of the potential difference across the drain-body junction is more than the band gap between the conduction and valence bands, then it is possible for the resultant electric field to accelerate the electrons and move them into the conduction band of the n-type region. They can then flow towards the positively charged drain terminal.

I_2: Subthreshold Leakage

Till a few years ago (before 2015) subthreshold leakage used to be the dominant mechanism for dissipating leakage power.

As we can see in Figure 11.4, this current flows from the drain to the source via the channel. When the gate voltage is lower than the threshold voltage (V_{th}), we traditionally assume that there is no current flow between the drain and the source. However, a small amount of current still flows and this is known as the subthreshold leakage current [Roy et al., 2003]. The subthreshold current, I_{ds}, is given by the following equation.

$$I_{ds} = \mu_0 C_{ox} \frac{W}{L} (m-1) v_T^2 \times e^{(V_g - V_{th})/m v_T} \times \left(1 - e^{-\frac{V_{DS}}{v_T}}\right) \tag{11.9}$$

The terms are defined as follows.

Term	Meaning	Term	Meaning
μ_0	Zero bias mobility	C_{ox}	Gate oxide capacitance
W	Width	L	Length
m	Subthreshold swing coefficient	v_T	kT/q
V_g	Gate voltage	V_{th}	Threshold voltage
V_{DS}	Drain-source voltage	k	Boltzmann's constant
q	1.6×10^{-19} C	T	Temperature (in Kelvins)

One of the most important terms in this equation is $v_T = kT/q$, which is known as the thermal voltage that is proportional to the temperature. We see that I_{ds} is proportional to the square of v_T via the term v_T^2, and is exponentially dependent on it via the two terms that have v_T as a part of the exponent. Because of these relationships, for a long time it was assumed that the leakage current is an exponential function of the temperature. However, this relationship is not strictly true as of 2020.

Sultan et al. [Sultan et al., 2018] in their study show that in most cases a linear model of leakage is reasonably accurate (within 6%) in the temperature range 40°C-80°C. This is because Equation 11.9 is approximately linear in this temperature range. For more accuracy it is wise to either use a piecewise linear model or a quadratic model. The errors for both models are within 1%.

In short-channel devices the source and drain regions are relatively close to each other. As a result, the threshold voltage of the transistor becomes dependent on the drain-source voltage, V_{DS}. This effect is known as drain induced barrier lowering (DIBL). As V_{DS} increases, it reduces the threshold voltage, which further increases the subthreshold leakage current as per Equation 11.9.

The subthreshold leakage current is also dependent on the body to source voltage (body bias). If it is reverse biased, then the threshold voltage increases, and this reduces the leakage current. Likewise, if we forward bias this junction, then the threshold voltage reduces – the transistor becomes faster at the cost of a higher leakage current. The exact relation between the threshold voltage and the body bias is dependent on the transistor technology. Most circuit simulation tools including Spice can model body bias. Body biasing is an important technique to modulate the leakage power consumption in circuits.

I_3: Gate Oxide Tunnelling

With increasing miniaturisation, the thickness of the gate oxide is reducing, and this is increasing the electric field across the gate oxide. This causes a quantum mechanical effect known as *tunnelling*. Here, electrons (or holes) in the channel can directly escape into the gate oxide or the gate. Any flow of electrons (or holes) is associated with a current. In this case, it is a leakage current because in an ideal transistor the gate oxide is assumed to be a perfect insulator.

There are two mechanisms for gate oxide tunnelling: Fowler-Nordheim tunnelling and direct handling.

In Fowler-Nordheim tunnelling, electrons move into to the conduction band of the oxide layer because of the external electric field. Note that in the conduction band, electrons have significant mobility and can be used to conduct current.

The other mechanism is known as direct tunnelling, which is becoming increasingly more prevalent as the gate oxide size thickness reduces to a few nanometers ($<$ 3-4 nm). In this case, electrons and holes tunnel directly from the surface of the silicon layer to the gate (typically made of polysilicon).

I_4: Hot-Carrier Injection

This is similar to Gate Oxide Tunnelling in terms of the fact that here also electrons or holes escape across the Si-SiO_2 (silicon and gate oxide) interface. However, the mechanism is different. Because of the large electric field, some electrons gain sufficient kinetic energy (become *hot carriers*) to cross the interface and move from the substrate to the gate (in the case of a positively charged gate). In Chapter 12, we shall study the effect of such hot carriers on the reliability of the gate, and appreciate how hot carrier injection gradually causes a breakdown of the gate oxide. This phenomenon is also associated with transistor ageing where the parameters of the transistor gradually change over time.

I_5: Gate-Induced Drain Leakage (GIDL)

Gate-Induced Drain Leakage (GIDL) is particularly concerning when the gate-to-drain voltage ($V_{GD} < 0$). In this case, positively charged holes migrate towards the gate oxide and get accumulated on the surface of the silicon. Holes get crowded at the drain-gate boundary. Because of this, minority carriers (holes in this case) migrate into the substrate (body of the transistor), where their concentration is relatively lower. This causes a current flow from the drain to the substrate. This is a leakage current, and happens because of the negative gate-to-drain voltage.

I_6: Punchthrough Current

In transistors with short channels we see this effect. It is possible that the drain-substrate and source-substrate interfaces are reverse biased. If the channel is short enough, then it is possible that the depletion regions of both of these regions can *merge*. This causes a breakdown in the substrate because now the drain and source are connected by a single and large depletion region. This decreases the potential barrier between the source and drain allowing carriers to flow between the source and drain terminals. Note that this current is very poorly controlled by the gate voltage because the conductive path is located deep within the substrate. This gets added to the subthreshold leakage current.

11.1.3 Summary

Let us summarise this section by going through the definitions of the key concepts once again.

Definition 93

- *Most of the power in a digital circuit is dissipated when we have* voltage transitions *at different points in the circuit. Whenever we have a transition there is a current flow across a CMOS transistor, and the resultant resistive power loss is known as* dynamic power*.*
- *A subclass of dynamic power consumption is short-circuit power consumption that specifically refers to the power that is dissipated when both the NMOS and PMOS transistors in a CMOS circuit are in a conducting state (while going through a state transition).*
- *Ideally we assume that the current flow in a CMOS transistor is only between the source and drain terminals, and that too when the gate or drain-source voltage changes. However, a small amount of current* leaks through *interfaces such as the drain-body junction, and body-gate junction, even when the transistor is in the* off *state. If we aggregate these small sources of power consumption across the billions of transistors, the total power consumption is sizeable. This is known as* leakage power *or static power.*

11.2 Temperature Model

11.2.1 Overview of the System

Figure 11.5 shows the diagram of a semiconductor package. The silicon die itself is fairly small. The typical die size for a server processor is 400 mm^2. Most of the power is dissipated in the logic layer (layer that contains transistors) of the chip. It is thus beneficial if the chip is placed upside down – transistor layer on top and metal layers at the bottom. Moreover, it is possible that temperature hotspots might form inside the chip that may adversely impact the reliability. Hence to homogenise the temperature profile, most packages

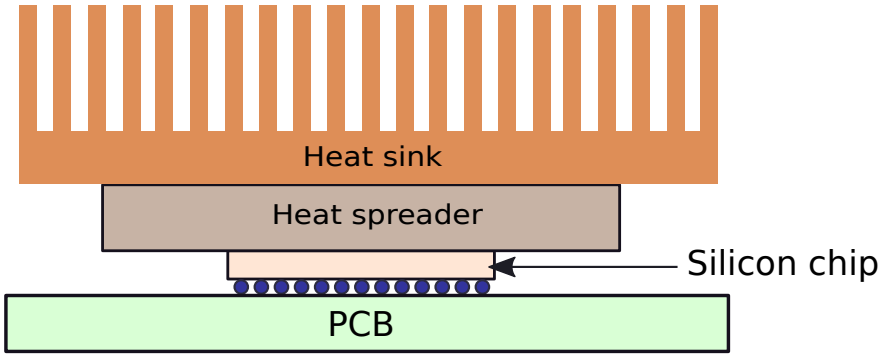

Figure 11.5: A semiconductor package

have a heat spreader that is made of a copper-nickel alloy. A typical heat spreader's area is 4cm by 4cm, and it is roughly 5 mm thick. Subsequently, to dissipate the heat we need a structure with a large surface area such that the heat can be dissipated to the surrounding environment. For this purpose, we use a heat sink with a large number of fins as shown in Figure 11.5. The fins increase the effective surface area and the resultant cooling capacity. Adjoining structures are separated by a thermal interface material (TIM). The role of the TIM is to ensure proper conductive heat transfer between the structures. For example, we have a TIM layer between the chip and the spreader, or the spreader and the heat sink. It needs to be a gel like material to ensure that it has good contact with the solid structures that it connects.

At the bottom, the semiconductor package is connected to the printed circuit board (PCB) via a ball-grid array. Each ball is a metallic connector that is used to transfer I/O signals or transfer current to the power and ground grids.

Most of the heat is transferred via the top surface (through the heat sink). The heat transfer through the sidewalls or via the PCB is comparatively much lower.

The goal of thermal modelling is to predict the temperature on the silicon die for a given power profile. In other words, if we know the topology of the chip and its power consumption, it should be possible to compute the temperature map of the entire silicon die. In this process some standard assumptions are made. We assume that the bottom surface (the side that touches the PCB) and the sides are *adiabatic*, which means that there is no heat flow across the boundary. The heat sink dissipates power to the *ambient* (surrounding environment), which is assumed to have a constant temperature that is known as the *ambient temperature*. This is an isothermal boundary (having a constant temperature). Note that these are idealistic assumptions, and in practice they are not completely true; however, for the sake of architecture level thermal modelling they are sufficient.

Definition 94
There is no heat flow across an adiabatic boundary. We have heat flow through an isothermal boundary where one of the sides has a constant temperature such as the boundary between the heat sink and the surrounding air (in a simplistic model that is suitable for architectural simulation).

Steady-state and Transient Analysis

While computing the temperature map, there are two kinds of analyses that can be done. The first is *steady-state analysis* where we assume that the power profile remains constant for a long period of time, and we are interested in finding the steady-state temperature. This is the most common kind of analysis

that is performed. In comparison, we might be interested in *transient analysis*. Here we are interested in the variation of temperature across time. For example, we might be interested to know how long it takes the temperature to rise to a given value after the power consumption of a functional unit is increased. Thermal time constants are typically of the order of milliseconds, and thus whenever we are interested in finer timescales we opt for transient analysis. Note that transient analysis is far more expensive in terms of time and computational resources as compared to steady-state analysis.

Definition 95
When the power profile does not vary and we are interested in the temperature profile after it has stabilised, we perform steady-state analysis. In comparison, if we wish to compute the variation of temperature over time, we opt for transient analysis. This is more expensive than steady-state analysis in terms of both computational time and computational resources.

11.2.2 Basic Physics

It is necessary to provide a brief overview of the physics of heat transfer to understand how temperature is modelled in modern chips. There are three primary mechanisms for heat transfer.

Heat Transfer Mechanisms

Conduction This is a process of heat transfer between objects that are in direct contact. Heat is transferred from the object that has a higher temperature to the object that has a lower temperature. For example, if we touch a hot electric iron, heat is transferred from it to our fingers. In a semiconductor chip, this is the primary mode of heat transfer within the package.

Convection This kind of heat transfer occurs within a fluid (liquid or gas). For example, near an active volcano, hot air and hot gases rise up into the atmosphere, effectively heating up the upper reaches of the atmosphere. Cold air takes its place, its temperature rises, and then it rises up in the same manner. The heat sink behaves in a similar manner. When the processor is running, the temperature of the heat sink rises which further increases the temperature of the nearby air. This hot air expands and moves through the cabinet, and cold air takes its place. The heat sink is effective because of this process of convective cooling. Sometimes, it is necessary to increase the velocity of air such that this process becomes more efficient, and this is why fans are required in laptop, desktop, and server processors.

Radiation Radiative heat transfer happens in free space. Any heated object emits radiation at different frequencies, and when this radiation is absorbed by remote objects their temperature rises. This method of heat transfer sustains our life. This is how the sun transfers its energy to our planet!

In semiconductor chips we are mostly interested in conductive cooling. There are two kinds of conductive processes in modern systems.

Vertical and Lateral Heat Conduction

Most of the heat escapes *vertically*, i.e., through the heat spreader and the heat sink. This is a desirable feature because the rest of the boundaries are mostly adiabatic. However, we do have some *lateral* heat conduction, where the heat flows sideways. This can happen either through the silicon die, or via the heat spreader. Even though lateral heat conduction is not as large as vertical heat conduction, it is still an important heat transfer mechanism and determines the placement of tasks and components on the chip as we shall see in Section 11.4.

Definition 96
Vertical heat conduction via the heat spreader and heat sink is the dominant heat transfer mechanism in packages. However, from the point of view of temperature management, a lesser mechanism called lateral heat transfer is also very important, where heat is transferred laterally on the silicon die.

Fourier's Law of Heat Conduction

The basic law that governs heat conduction is the Fourier's law. Consider an infinitesimally small rectangular area A (see Figure 11.6(a)). Let the temperature on the left side be T_1 and the temperature on the right side be T_2 ($T_1 > T_2$). Assume that the region's normal vector is along the x-axis, and the thickness of the region is Δx. Let the power entering the left side be q_x units. Then the Fourier's law of heat conduction says that $(T_1 - T_2)/\Delta x$ (temperature gradient) is proportional to q_x/A (heat flux).

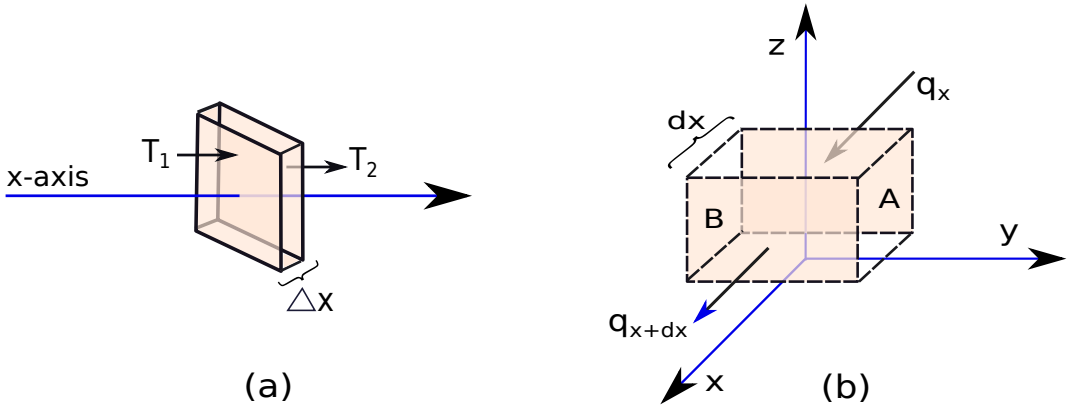

Figure 11.6: Setting for explaining the Fourier's law of heat conduction (a) rectangular slab whose normal vector is oriented along the x-axis, (b) A small 3D region

Given that the temperature gradient, and the direction of heat flow are in opposite directions, we can write the following equation, where k is the constant of proportionality. Note the negative sign.

$$q_x = -kA\frac{dT}{dx} \tag{11.10}$$

Now consider a very small three-dimensional volume with dimensions dx, dy, and dz in Figure 11.6(b). We show the heat transfer on the x-axis (likewise is the case for other dimensions). q_x units of thermal power (in Watts) enter face A (normal vector along the x-axis), and q_{x+dx} Watts leave face B. The difference is $q_x - q_{x+dx}$, and the area of the face is $dydz$. Let us now use Equation 11.10. We have,

$$q_x = -k.(dydz).\frac{\partial T}{\partial x} \tag{11.11}$$

Here the "." operator is used to denote multiplication. Note that we use a partial derivative because we are only interested in the temperature difference along the x-axis. Let us use this result in the following

derivation.

$$\begin{aligned}
&\frac{q_x - q_{x+dx}}{dx} = -\frac{\partial q_x}{\partial x} \\
\Rightarrow & q_x - q_{x+dx} = -dx.\frac{\partial}{\partial x}\left(-k.dy.dz.\frac{\partial T}{\partial x}\right) \quad (\textit{by Equation } 11.11) \\
\Rightarrow & q_x - q_{x+dx} = k.dx.dy.dz\frac{\partial^2 T}{\partial x^2} = kd\mathcal{V}\frac{\partial^2 T}{\partial x^2}
\end{aligned} \tag{11.12}$$

We will use the term $\mathcal{V}$ to represent volume and the term V to represent voltage.

In Equation 11.12, the term $q_x - q_{x+dx}$ is the heat that remains within the volume if we only consider heat flow along the x-axis. Note that we assume that k is a constant in all directions and $d\mathcal{V} = dx.dy.dz$. If we sum up the power that remains within the volume because of heat flow along all the three axes, we arrive at the expression $kd\mathcal{V}\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}\right)$. Let the operator $\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)$ be represented as ∇^2. Thus the thermal power that remains within the volume is $k.d\mathcal{V}.\nabla^2 T$.

Let us now consider the internal heat generation. It is possible that we have a power source within the volume that generates Q' units of energy per unit volume and per unit time. It can be alternatively described as the internal power generated per unit volume. Thus the total power generated is $Q'd\mathcal{V}$. Note that we will use the term Q' to denote the internal heat generation rate and the term Q for denoting the charge in an electrical circuit.

Hence, the sum of the total power that enters the volume and is generated within it equals $P_{st} = (k.\nabla^2 T + Q').d\mathcal{V}$. All of this stored thermal power, P_{st}, must increase the temperature of the small volume that we are considering. Let ρ be the density. Thus, the mass of the region is equal to $\rho.d\mathcal{V}$. The relation between the temperature rise per unit time, $\frac{\partial T}{\partial t}$, the mass, the specific heat (c_p), and the net thermal power entering and being generated within the volume is given by

$$P_{st} = \rho.d\mathcal{V}.c_p.\frac{\partial T}{\partial t} \tag{11.13}$$

From the laws of the conservation of energy, the total power entering and being generated within the volume equals the value of P_{st} shown in Equation 11.13. After cancelling the common factor $d\mathcal{V}$, we have,

$$k\nabla^2 T + Q' = (\rho c_p)\frac{\partial T}{\partial t} \tag{11.14}$$

This can be simplified to:

$$\nabla^2 T + \frac{Q'}{k} = \frac{1}{\alpha}\frac{\partial T}{\partial t} \tag{11.15}$$

Here, $\alpha = k/\rho c_p$. If we consider only the steady state problem we have,

$$\nabla^2 T = -\frac{Q'}{k} \tag{11.16}$$

Here, T is the temperature map and Q' is the power map. We now need to find a general solution to this differential equation, and then need to factor in the effect of boundary conditions.

11.2.3 The Finite Difference Method (FDM)

Let us convert Equations 11.15 and 11.16 into a set of simple algebraic equations. Consider the silicon die and adjoining structures; let us break it into a 3D grid consisting of very small cube-shaped cells. Assume that the cells are numbered with an integer.

Consider cell i at time t. Assumption: cells $i-1$, and $i+1$ are the neighbours of cell i on the x-axis.

$$\begin{aligned}\left.\frac{\partial^2 T}{\partial x^2}\right|_{i,t} &\approx \frac{\left.\frac{\partial T}{\partial x}\right|_{i+1,t} - \left.\frac{\partial T}{\partial x}\right|_{i,t}}{\Delta x} \\ &\approx \frac{\frac{T_{i+1,t}-T_{i,t}}{\Delta x} - \frac{T_{i,t}-T_{i-1,t}}{\Delta x}}{\Delta x} \\ &= \frac{T_{i+1,t} - 2T_{i,t} + T_{i-1,t}}{\Delta x^2}\end{aligned} \tag{11.17}$$

We can create a similar set of equations for the y and z axes. The important point to note is that these are all *linear equations*, and these equations hold for a point that is not on the boundary. We need to formulate special equations for points that lie on the boundary based on the boundary conditions. Then we can solve the entire system of equations using standard linear algebra techniques.

As an example, consider the steady-state problem. We need not consider the time axis. We thus have a set of linear equations where the only set of variables are of the form T_i as shown in Equation 11.17. i varies from 1 to N, where N is the number of cells. Let us create an N-element vector representing the power map, $\mathbf{P}$ (one entry for each cell). On similar lines, let $\mathbf{T}$ (N elements) represent the temperature map. We now know that we can obtain $\mathbf{T}$ from $\mathbf{P}$ (Equation 11.16) by applying a linear transformation. Assuming these are column vectors, we can write this relationship as follows (note that a linear transformation can be written as a matrix-vector product):

$$\mathbf{T} = \mathbf{AP} \tag{11.18}$$

$\mathbf{A}$ is an $N \times N$ matrix. It can either be derived from first principles by creating a set of algebraic equations (see Equation 11.17), or this matrix can be learnt from sample values.

Transient Analysis

Let us now solve the transient problem by creating a similar matrix based formulation. Let us first start with the Fourier equation with the terms slightly rearranged.

$$k\nabla^2 T + Q' = \frac{k}{\alpha}\frac{\partial T}{\partial t} \tag{11.19}$$

As per Equation 11.17 $k\nabla^2 T$ is a linear transformation, which can be represented as the product of a matrix and the vector of N temperatures. Let us refer to the time derivative of temperatures as $\mathbf{T'}$, which is also an N-element vector. We are multiplying each of its elements with a scalar, which can be different for each element because it depends on its density and specific heat. We can thus represent this product as the product of a diagonal matrix $\mathbf{C}$ with an N-element vector $\mathbf{T'}$. Finally, note that Q' is the internal power generated per unit volume across the chip, which we are representing by the vector $\mathbf{P}$. We can thus write Equation 11.19 as a linear equation.

$$\mathbf{GT} + \mathbf{P} = \mathbf{CT'} \tag{11.20}$$

Here, $\mathbf{G}$ and $\mathbf{C}$ are matrices, $\mathbf{P}$ and $\mathbf{T}$ represent the power and temperature vectors respectively. This equation can be solved using numerical methods.

For example, we can divide time into discrete time-steps, and starting from $t = 0$ we can keep solving Equation 11.20 using standard linear algebra techniques. We can replace $\mathbf{T'}$ with an expression of the form $(\mathbf{T_{k+1}} - \mathbf{T_k})/\Delta t$, where $\mathbf{T_k}$ is the temperature at all the points in the k^{th} time step.

11.2.4 Electrical Analogue of the Heat Transfer Problem

Closely take a look at Equation 11.20: $\mathbf{GT} + \mathbf{P} = \mathbf{CT'}$. We can replace the thermodynamic quantities with analogous electrical quantities that have similar relationships. We can replace temperature with voltage,

and power with current. This is know as the electrical analogue of a thermal problem. Let us now try to map the rest of the terms.

The matrix **G** can be thought of as conductance matrix. Recall from high school physics that conductance is the reciprocal of resistance. By the Ohm's law the product of conductance and voltage is equal to the current (power in this case). The second term in the LHS, **P**, can be thought of as a the output of a set of current sources. Thus, the sum of the two terms on the left-hand side indicates the total amount of current being injected into the circuit. In an electrical circuit when current is injected the voltage rises, which is being captured by the expression on the right-hand side.

To understand the right-hand side, let us consider the equations related to capacitance: $Q = VC$ (Q is the stored charge, V is the voltage, and C is the capacitance). If we differentiate it, we get $dQ/dt = I = CdV/dt$. The time derivative of voltage can be equated with the time derivative of temperature T'. The matrix **C** in Equation 11.20 can thus be thought of as the thermal capacitance.

Let us now describe the electrical analogue in a simpler setting. Consider a 1-dimensional rod. By Equation 11.10 the temperature difference across its ends $T_1 - T_2$ is equal to the inlet power multiplied by the term $\Delta x/kA$ (ignoring the negative sign). Here, Δx is the length of the one-dimensional rod. This is very similar to the formula for electrical resistance: $\rho l/A$ (ρ is the resistivity, l is the length, and A is the cross-sectional area). We can interpret the constant k as the thermal conductivity.

Similarly, for capacitance let us start with the equation: $P_{st} = \rho.d\mathcal{V}.c_p\frac{\partial T}{\partial t}$. In this case, P_{st} can be mapped to the electrical current and $\frac{\partial T}{\partial t}$ to the time derivative of voltage (dV/dt). Since $I = CdV/dt$, the thermal capacitance can be represented by $\rho.d\mathcal{V}.c_p$. Note that this is a capacitance to ground.

The advantage of this approach is that we can replace a thermal problem with an analogous electrical problem, and then we can use traditional circuit simulators to find the voltages and currents at different places. These values can then be mapped to temperature and power values. Since there are very efficient methods to simulate circuits, this method is very efficient.

Consider an example in Figure 11.7. The cell in the centre is connected to the neighbouring cells via thermal resistances. Additionally, note the capacitance to ground. For cells at the edges we will not add a resistance to the ambient if the boundaries are adiabatic (no heat flow).

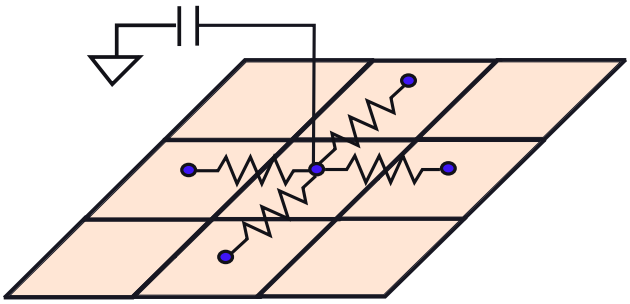

Figure 11.7: The electrical analogue of a thermal problem

11.2.5 The Finite Element Method (FEM)

The main advantage of the finite difference based approaches is that they are simple. However, for complicated thermal models that consider a variety of heat transfer paths, the effect of the thermal interface materials and heat transfer via the PCB, often finite element-based methods are found to be more accurate. Along with the effect of capturing complex geometries, this method is also very effective when the properties of materials tend to vary over space. The cost of such additional accuracy is slower speed.

Let us now provide an outline of the basic steps that are involved. For a deeper discussion readers can refer to the book by Logan et al. [Logan, 1986].

1. The first step is to convert the differential equation into an integral equation. This is an analytical exercise and is done by multiplying the original differential equation with a test function, $v()$, which has continuous derivatives till a certain point.

2. The next step is to discretise the domain, which means dividing it into a set of triangles, rectangles, cuboids or tetrahedrons. This process is known as *meshing*. Consider a triangular mesh. Here, the size of each triangle determines the accuracy of the solution and the time the method takes to complete. Each such triangle is called an *element*, and the points at which elements are connected are known as *nodes*. We can choose an appropriate subset of elements to reduce the size of the problem.

3. We take the integral equations generated in step (1) and convert them into a set of algebraic equations. Here we consider different values of the function v for each node. These are known as basis functions, and typically satisfy several desirable properties. For example, we can have linear basis functions, where v is a linear function of the coordinates of the nodes around an element. We can also use quadratic or polynomial basis functions. The intuition is as follows.

   We wish to represent $f(x)$ (where f is the function to be evaluated) as $\sum_{k=1}^{n} f_k v_k(x)$. Here, $v_k(x)$ is the basis function corresponding to node k and f_k is the value of $f(x)$ at node k. The assumption here is that the values of $f(x)$ within an element are only a function of the values at the nodes surrounding the element and the functions of the form $v_k(x)$ (basis function for node k). We have essentially assumed that the final solution is a weighted sum of simple basis functions multiplied by the values of the function at the nodes. If there are other functions in a differential equation, they can be represented with a similar sum.

4. Given this assumption, the task that remains is to find the values of the function f at the nodes. We convert the set of equations derived in step (3) into linear equations in the matrix form, and solve them using standard linear algebra techniques. We thus get an estimate of the function $f()$ at all the nodes.

It is important to note that the final accuracy is dependent upon the choice of basis functions. There is a very rich area of research for determining the right set of basis functions for different problems. However, given the slow speed of such methods they are seldom used in architectural simulation.

11.2.6 Green's Functions

Let us now consider the fastest methods in the space, Green's functions. We can use some unique features of the thermal problem to design a very fast method to compute the temperature profile. The key features that we use are *linearity* and *shift invariance*. Linearity means that if T_x is the thermal profile for the power profile P_x, and T_y is the thermal profile for the power profile P_y, then the thermal profile for the power profile $P_x + P_y$ is $T_x + T_y$. This can be easily derived from Fourier's heat equation, which is a linear equation. Shift invariance means that the thermal profile generated at any point because of a point power source remains the same with respect to the power source, even if the power source is shifted. A disclaimer is due here. Shift invariance does not hold if the power source is applied too close to the boundary of the chips. For the rest of the points if the power source is small enough, we can assume that they are very far away from the boundaries and thus the localised temperature spread is roughly the same.

To use these relationships, let us proceed as follows. Let us divide the entire silicon die into a set of very small square grid-shaped cells (an $N \times N$ grid). For a cell at the centre, let us apply 1W of power. Let its temperature profile be given by the function $G(x, y)$. Consider this cell at the centre as the origin. Because of the property of shift invariance we will have the same thermal profile if the power source is applied at any other point (x_p, y_q). For a set of power sources and by the properties of linearity and shift invariance we have,

$$T(x_p, y_q) = \sum_{i=1}^{N} \sum_{j=1}^{N} P(x_i, y_j) G(x_p - x_i, y_q - y_j) \tag{11.21}$$

This considers the cumulative effect of a set of power sources on the temperature at a single point (x_p, y_q). Let us consider the integral form of this equation at the point (x', y').

$$T(x', y') = \int_x \int_y P(x, y) G(x' - x, y' - y).dxdy \tag{11.22}$$

This is precisely the convolution operation ($\star$), and we can thus simplify this equation to,

$$T = P \star G \tag{11.23}$$

If we consider an infinitesimally small cell, the power source becomes the Dirac delta function $\delta(x)$. The Dirac delta function is defined for a point x_0, and has a value of 0 elsewhere. It satisfies the property $\int \delta(x)dx = 1$. The Green's function thus becomes the temperature profile of the Dirac delta function, which is known as its *impulse response*.

The Green's function can be computed analytically or can be estimated by simulating the thermal profile of a very small power source using a traditional FEM or FDM based tool. Computing the temperature profile is as simple as computing a simple convolution operation using Equation 11.23. Most methods convert Equation 11.23 into the Fourier transform domain, where a convolution becomes a multiplication. This is a very fast operation as compared to finite difference based methods.

Definition 97
The Green's function is the impulse response of a power profile that is a Dirac delta function. The temperature profile is the power profile convolved with the Green's function.

Note that this approach has its limitations because it is applicable to cells that are not at the edges and corners. At the rim of the chip the heat does not dissipate through the side walls because they are considered to be adiabatic. Hence, there is a disproportionate temperature rise. Here, we use the method of images. If a power source is x units away from a side wall, then we add another power source of the same magnitude at the point $-x$, which is x units away from the side wall on the opposite side. This is known as an *image source*. The temperature profile is approximately a superposition of both the temperature fields.

11.3 Power Management

11.3.1 Managing Dynamic Power

DVFS

As discussed in Section 11.1.1, the relationship between voltage and frequency in modern processors is not linear. Most processors define a set of power states, where each state is a voltage and frequency pair. The CPU can move between these states depending upon the ambient temperature, BIOS settings, and the power usage of the application. In most power efficient processors it is possible to run different cores in different power states. The only problem of doing so is that it becomes hard to send a message across voltage-frequency boundaries. We need a special circuit that shifts the level of the voltage, and synchronises the transfer. Once this is done we can achieve a fine grain control of the voltage and frequency of each core. It is possible to extend this idea further and have separate DVFS (dynamic voltage and frequency scaling) settings at the level of individual functional units. However, given the overheads, such choices are often not justified.

The DVFS settings for the Intel® Pentium® M Processor are shown in Table 11.1. It has six different power states. It is also possible for the operating system to make the CPU migrate between power states if the architecture makes special instructions available. The advantage of having software control is that

we can leverage the additional visibility that is there at the level of the operating system. This can also be done at the level of individual programs. For example, in Linux, the *cpufreq* utility allows us to adjust the voltage and frequency on-the-fly. Many software applications use such facilities to reduce their power consumption.

Frequency	Voltage
1.6 GHz	1.484 V
1.4 GHz	1.420 V
1.2 GHz	1.276 V
1.0 GHz	1.164 V
0.8 GHz	1.036 V
0.6 GHz	0.956 V

Table 11.1: Voltage and frequency settings for the Intel Pentium M processor [Intel, 2004]

In any such processor, to transition from one state to the other it is necessary to do the following.

Higher frequency We first increase the supply voltage by programming the voltage regulators on the motherboard. Note that the process of increasing the voltage takes time because the capacitors associated with the chip's power grid, and power pins need to be charged. After this is done, we pause the execution and re-tune the PLL (Phase locked loop) based clock generator of the processor to generate the new high-frequency clock signal. On the motherboard we typically have a quartz-based oscillator that generates a clock signal at a fixed frequency – typically 133 MHz. Based on the frequency setting we multiply this clock signal with a fixed value and set the PLL to oscillate at that frequency. After the PLL locks to the new frequency, we can start the execution.

Lower frequency In this case, we first pause the system, and then relock the PLLs to the new clock frequency. After that the voltage regulators gradually reduce the voltage.

Note that the process of changing the DVFS settings is asymmetric in nature. We never have a situation where the frequency is higher than what the supply voltage can support. This can cause unpredictable behaviour in the circuits. Furthermore, the process of changing the voltage takes time because large capacitors in the on-chip power grid need to be charged or discharged. Additionally, it takes time for the PLL to relock to the new frequency. In very aggressive implementations, this process takes roughly 10-20 μs. During most of this period, the chip is not operational. Hence, DVFS is an expensive optimisation and power states should be changed infrequently.

DVFS is done at the level of a core or at the level of the entire chip. Additionally, it is dependent on the behaviour of an application and the expectation of the end-user. For example, if we are playing a video we want every frame to be processed within 33 ms. Assume that it takes only 20 ms to process a frame. In this case, we can apply DVFS to deliberately slow down the processor such that it takes roughly 33 ms, and we reduce power as much as possible. In this case, it is clearly known that the user wants to see jitter-free video; hence, there is a strict deadline for processing a frame. As long as we stay within that, we can use DVFS to lower the voltage and frequency and consequently save power. In mobile phones, DVFS can be applied based on the user's requirements and the available battery power. We can apply DVFS more aggressively if the user wants to be in a power save mode.

Let us next look at the space of decisions that can be taken at a much lower level – at the level of functional units.

Clock Gating

Clock gating is a simple technique where we simply disable the clock of a functional unit (see Figure 11.8) that is not expected to be used in the near future. Here, we can either follow a deterministic policy or a

non-deterministic policy. In the first case, we know for sure that a given functional unit is not going to be used. For example, if we see a divide instruction we can be very sure that the functional unit for subtraction is not going to be used till we execute one more instruction. In this case, we can confidently gate the clock of the subtract unit.

However, in some cases we might not be very sure (non-deterministic). In such cases it is necessary to design a predictor, which can predict if a functional unit is expected to be used in the near future or not. If it predicts that a functional unit will remain idle then its clock can be gated. By gating the clock we are not allowing the latches that feed data to the functional unit to change their values. This means that there will be no voltage transitions in the inputs of the functional unit and thus there will be no current flow or resultant dynamic power dissipation. Even if the inputs remain the same, then also there is a benefit because the clock is routed to all the latches in a circuit, and we need to periodically charge and discharge the clock inputs of the latches. This is also avoided.

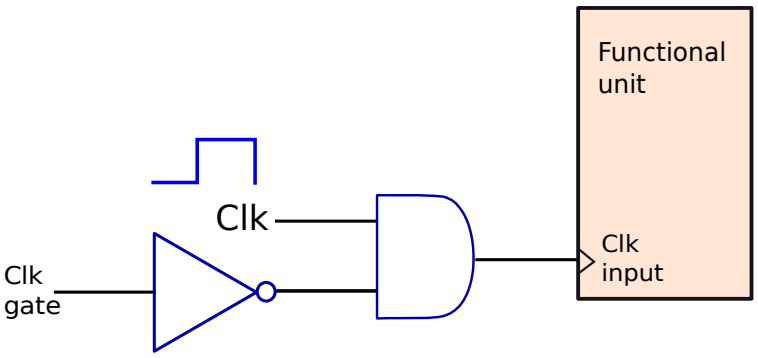

Figure 11.8: A simple clock gating circuit

There are different micro-architectural techniques to find out when a unit is going to be used or not. We can embed a small counter within a unit. If the unit is idle, the counter starts counting down to 0. Once it reaches 0, we can gate the clock of the associated functional unit. We can also find the usage of functional units in the select stage, where we know which execution units are not going to be used. The clocks of the unused functional units can be gated. Similarly, after decoding instructions we know which parts of the rename and dependence check logic will be used. The remaining parts can also be clock gated. For the caches, we can predict cache activity and clock gate the decoders.

The primary problem with clock gating is that it makes the circuit complicated. For example, we need to be very careful with regards to when we gate the clock. If the value of the clock is being set to 0 when its value is 1, we are introducing a clock transition. The rest of the circuit will perceive a negative edge prematurely. No correctness problem should be created by this. Secondly, verifying the design becomes more complex, because it is possible that when an input arrives at a functional unit, its clock is gated. We need to wait till the unit is enabled. This can lead to many unforeseen errors. To avoid such situations many processors only opt for deterministic clock gating where no such delays are introduced.

Issue and Fetch Throttling

Let us now consider an even simpler mechanism that does not involve major changes to the circuit, and does not introduce correctness errors. We can simply reduce (*throttle*) the issue or fetch rate. Instead of issuing six instructions per cycle, we can issue four instructions per cycle. This will reduce the dynamic power consumption immediately. Additionally, implementing this is very simple and requires minimal changes to the select logic. On similar lines, we can throttle the fetch and commit rates as well.

Even though this mechanism is very effective in reducing dynamic power, it has a direct impact on the performance. It slows down the program and does not necessarily reduce the total energy requirement.

There is an important point to note here with regards to leakage power. See Example 14.

Example 14
Consider two designs A and B. Assume they require the same amount of dynamic energy (power×delay). The dynamic power of design A is 50% more, and it is 3 times faster than design B. Which design should we prefer?

Answer: *The temperature is a mildly super-linear function of the dynamic power. If temperature is a concern then we should prefer design B. However, if we want to reduce the total amount of energy then we need to consider the energy consumed by leakage as well. The leakage energy is equal to the leakage power multiplied by the program's execution time (delay). In this case, A consumes more leakage power because of the higher temperature, however we cannot say the same about the leakage energy. This is because A's total delay is one-third of B's delay. It is possible that the sum of the dynamic energy and leakage energy for A is less than that of B. In this case, we will prefer design A.*

If we also want to bring the performance into the picture, then we should minimise the energy-delay^2 or (power-delay^3) product. If the power of design B is P and delay is D, then the power-delay^3 product for design B is $1.5 \times P \times (D/3)^3 = 0.05PD^3$. The corresponding product for design A is PD^3. We would clearly prefer design B with these metrics.

11.3.2 Managing Leakage Power

Power Gating

The most effective way of controlling leakage power is *power gating*. If we stop the flow of current into the circuit, then there will be no leakage, and consequently no leakage power dissipation.

There are three ways of implementing power gating. We either disconnect the supply voltage, V_{dd}, disconnect the connection to the ground, or disconnect both. In all the three cases we are breaking the connection between the supply and the ground and thus ideally there will be no current flow. Out of these approaches, disconnecting the supply voltage or disconnecting both the supply and ground are more common. Disconnecting only the ground terminals does eliminate most sources of leakage power, however in a large circuit we always have some leakage paths to ground via the package. Some current can leak through them.

The design of both the circuits is shown in Figure 11.9. The idea is very simple. We have two grids for supplying and removing current: one for the supply voltage (power grid), and one for ground (ground grid). If a chip has higher power requirements then we can have multiple grids for the supply voltage and ground. However, for the sake of simplicity let us assume that we have a single grid for each type of voltage, and each grid has its dedicated VLSI layer. The reason that we need a grid is such that if a given functional unit needs more power, it can draw current from other parts of the grid. Having a grid helps us balance the current flow across the power and ground pins of the chip. Furthermore, a large grid has a large capacitance, which can further be augmented by adding additional capacitors called decoupling capacitors to limit voltage fluctuations.

Each functional unit has several connections to the power and ground grids respectively. To effectively decouple the functional unit we need to add pass transistors on all these connections. To power gate the functional unit all that we need to do is disable these transistors, which are also commonly known as *sleep transistors*. When a circuit is power gated, it is said to be in the *sleep state*. However, this is easier said than done. We need to keep several things in mind.

1. The current that is available to a functional unit should not reduce after adding sleep transistors. This means that to carry the requisite amount of current these transistors have to be large.

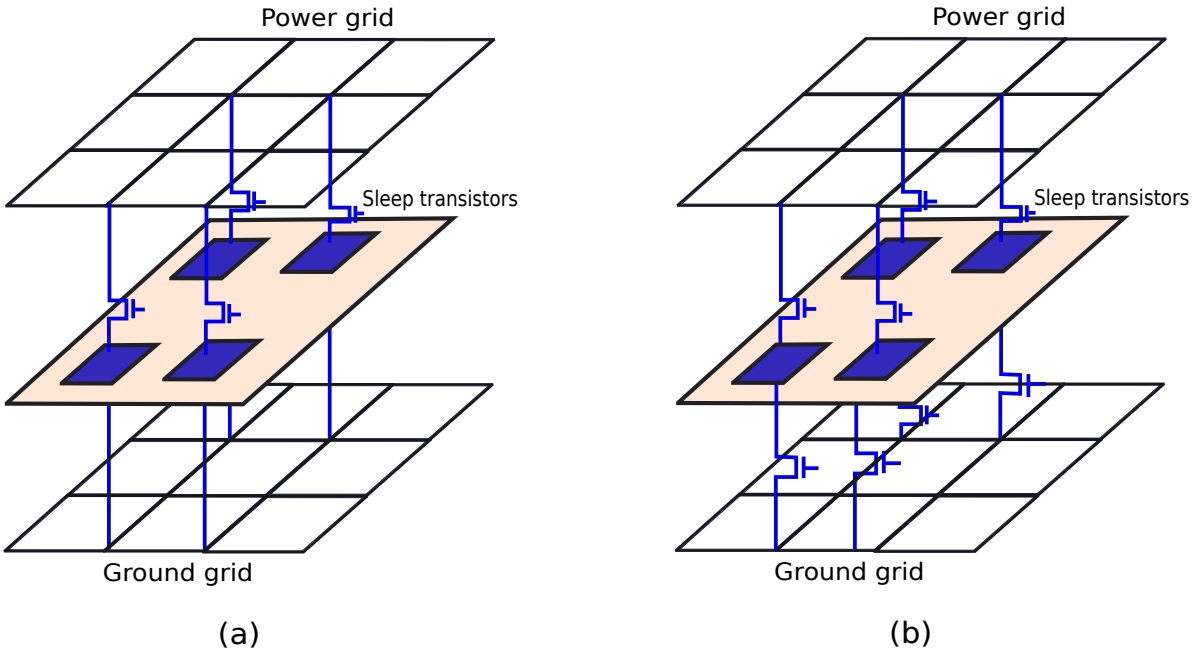

Figure 11.9: Power gating. (a) Disconnecting the supply voltage, (b) Disconnecting the supply and ground

2. The problem with large transistors is that they take a long time to switch on and switch off. This puts a limit on how frequently we can power gate a circuit and then enable it back again.

3. There will invariably be some amount of leakage through the sleep transistors. In most designs this is expected to be insignificant though.

4. When we enable and disable power gating, the current flow in the power and ground grids changes rather abruptly. As per Lenz's law, any electrical conductor opposes the change of current through it by introducing a back EMF – referred to as the *power gating noise*. To taper the effect of such voltage fluctuations, we need to add large decoupling capacitors to the power grid and also slow down the process of entering and exiting sleep states. The decoupling capacitors will also increase the leakage power rather disproportionately.

5. The sleep transistors will also have a potential difference across their terminals. This will decrease the effective supply voltage for the transistors in the functional unit. As a result, the supply voltage needs to be increased to compensate for this reduction.

6. When a functional unit is power gated, it loses its state. As a result, functional units need to be designed keeping this in mind, which is sometimes difficult.

7. We are assuming that we know for sure when a functional unit is not expected to be used. We need a predictor for this purpose, and if the predictor is not accurate, then there will be a loss in performance. We will waste valuable cycles in waking up the functional unit from the sleep state.

8. Cycling between the active and sleep states is associated with drawing in large currents from the power grid. The resultant dynamic power consumption needs to be taken into consideration.

To summarise, the support for power gating does not come for free, it has its associated costs. As a result, we need to take a very judicious decision regarding whether a functional unit should be power gated

or not. An important point to note is that we need to have a reliable predictor, which can predict long periods of inactivity in the future based on either past history or from programmer/compiler inputs. It does not make sense to power gate a circuit for a short period of time because the overheads of doing so might overshadow the gains in a reduction of leakage power. Over the past decade, many algorithms have been proposed for effective prediction of inactive periods, and this is still an area of active research.

Drowsy Caches

A major limitation of power gating is that we completely turn off the supply voltage and this leads to a complete loss of state. Power gating is thus not a very effective approach for cache banks, which take up a large amount of area on the die. The issue is that when we turn off the supply voltage, the caches lose all their data, which is not acceptable. Hence, it is a much better idea to define an intermediate *drowsy state* [Kim et al., 2004], where the supply voltage is reduced but it is not so low that the state elements lose their values. The state elements can be either regular latches or SRAM arrays.

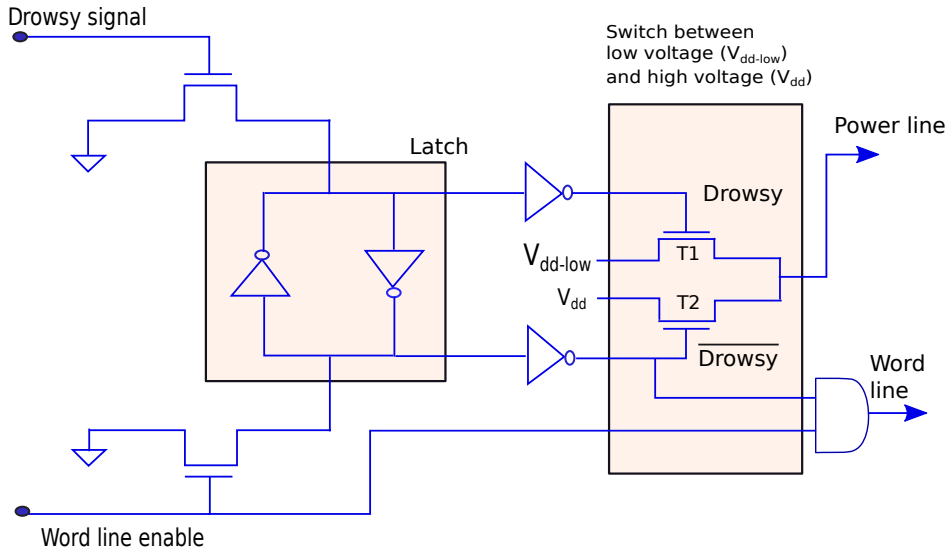

Figure 11.10: Voltage regulator of a drowsy cache (adapted from [Kim et al., 2004])

Figure 11.10 shows the changes that need to be made to make a regular cache bank a *drowsy cache bank*. We have a set-reset latch made of an inverter-pair where the drowsy signal sets the latch, and the word line enable signal resets the latch. It is assumed that when the cache is in drowsy mode, the word line enable signal is off. In other words, we do not access the cache. Similarly, when we intend to access the cache by setting the word line enable bit, the cache should not be in drowsy mode (the drowsy signal should be reset). Let us elaborate.

Consider the case when the drowsy signal is a logical 1, and the word line enable bit is a logical 0. In this case the gate voltage of transistor $T1$ is a logical 1, and this transistor is enabled. However, transistor $T2$ is disabled. Thus the voltage of the power line (powering the cache bank) is V_{dd-low}, which means that the cache bank is in the drowsy state. On similar lines, we can argue that when the drowsy signal is a logical 0, and the word line enable signal is 1, transistor $T2$ is enabled and the cache bank gets the full supply voltage, V_{dd}. This is when the bank can be accessed. The AND gate also enforces the condition that a word line in a bank can be enabled only if the drowsy signal is 0, and the word line enable signal is 1.

Finally, note that in the default state when both the signals are a logical 0 the latch maintains its value. If the cache bank is in a drowsy state, it continues to be in that state.

There are several design decisions that need to be made here. We need to decide the granularity at which we add this circuitry. Starting from an individual cache line to an entire cache bank we can create groups of lines and together assign them to the drowsy state. There is a trade-off between area and flexibility here.

The second design decision that needs to be made is regarding prediction. At one end of the spectrum we have an on-demand approach, where we might decide to set every cache line to the drowsy state and only make it active when there is an access. If we disregard the power required to enter and exit the states, then this is the most power efficient approach. However, the cache access time increases because we need to allocate at least one cycle to changing the state of the cache line. This results in a slow down. The other option is to predict periods of inactivity for groups of cache lines, and then collectively set them to the drowsy state. In this case the accuracy of the predictor is very important.

Multiple V_{th} Designs

The leakage power is a function of the threshold voltage. If the threshold voltage is low, the leakage power is high and vice versa. Moreover, lower is the threshold voltage, faster is the transistor. This property can be exploited in many different ways.

In a typical circuit all the transistors are not on the critical path (the longest path that determines the overall timing). There are many paths in the circuit that can be slowed down without changing the length of the critical path. On such paths we can have transistors with higher threshold voltages. They will be slower, yet will be more power efficient. Let us say that in a given circuit, the length of the critical path measured in terms of the time it takes a signal to propagate is 300 ps. If we have a path in the circuit, which is 200 ps, we can slow the transistors down such that the length of the path becomes equal to 300 ps. By slowing down the transistors we will save on leakage power.

There are different methods to slow down the transistors. One of the most popular methods is body biasing. In this case we set the voltage of the body of the transistor to a given value. If the body-to-source voltage is positive, then this is known as forward body biasing. In this case the threshold voltage reduces. Likewise if the body-to-source voltage is negative, then this is known as reverse body biasing, and this leads to an increase in the threshold voltage. It is possible to create islands of transistors where their bodies are connected to a single voltage source, and we can thus change the threshold voltage (speed and power) of a group of transistors by just adjusting the body voltage.

Static techniques to change the threshold voltage include changing the dopant density, and the dimensions of the transistors.

Regardless of the approach, the overarching idea is that we can deliberately slow down transistors to save leakage power, when the transistors are not on the critical path.

11.4 Temperature Management

In general any power management scheme is effective for temperature as well. Since the overall power decreases, the temperature and the leakage power also decrease. However, sometimes temperature hotspots can form if a functional unit is being heavily used. For example, it is possible that in some workloads a given core or a given functional unit such as a multiplier may be very heavily used. Coupled with other factors, this can cause a local temperature rise, which can be detrimental to the health of the chip in the long run. Hence, it is necessary to monitor the temperature in as many places as possible and take appropriate action. In this case, the action would be to either move the computation to another core or another multiplier, or throttle the computation rate. Both of these would be effective in reducing the intensity of the temperature hotspot. A common approach is called stop-and-go, where we stop the execution for some time till the chip cools down, and then we restart it. This is an extreme case of throttling.

Hence, for a long time we did not have separate schemes for temperature reduction. Additionally, most of the heat transfer happens in a vertical direction (through the heat sink) and thus the spatial location of activity was not that important because lateral heat conduction was less of an issue.

However, off late this problem has been receiving more attention because of increasing power density, and process variation where localised regions of the die can have a very high leakage power with large non-linear effects. Let us elaborate.

1. The power density is increasing, and transistors are getting smaller and more unreliable. In addition, because of process variations (variation in transistor dimensions caused due to imperfections in the fabrication process), the leakage power tends to vary across the chip, and thus it is possible to have some areas of the chip where the leakage power is very high. Because of this, the probability of forming temperature hotspots is also elevated.

2. Modern hardware such as GPUs and neural network accelerators consume a lot of power (typically 2X that of server processors), hence for such settings, temperature issues are far more important.

3. We are moving into an era of 3D chips that will have multiple transistor and DRAM layers. Here heat transfer is an issue for the inner layers where the temperature can increase significantly. Hence, there is a need to lay out the computations in such a manner such that at all points the temperature is below a specified threshold. Nowadays, the GPU has also come into the package; this further increases the power dissipation and causes temperature problems.

4. For 3D chips traditional air cooling might not be sufficient. There is a lot of research underway in creating water based cooling solutions and microchannel based cooling. A microchannel is a very small tube (cross-sectional area: $50 \times 100 \mu m^2$) within the chip that carries a coolant. The idea is to create an array of microchannels to carry a cooling fluid, particularly for cooling the inner layers of a 3D chip, where modelling and mitigating temperature hotspots is of vital importance.

Given these reasons, mapping jobs to cores has become an important problem. Such problems are known as mapping or job placement problems.

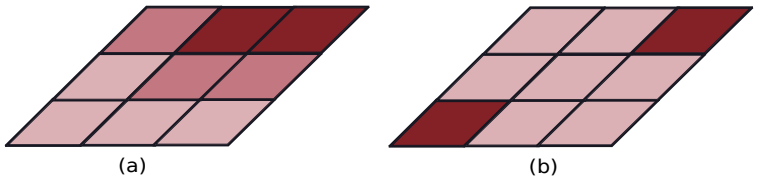

Figure 11.11: Effect of the placement of jobs. (a) Two hot jobs placed side by side, (b) The hot jobs placed at opposite corners

11.4.1 A Typical Placement Problem

A typical job placement problem in this space as follows. Assume we have N jobs and N cores. The power consumption of each job (thread) is known. We can have many objective functions. Consider a simple one: minimise the peak temperature subject to a constraint on the minimum performance.

First consider a problem where the performance is not dependent on the placement. Then we have several options. We can either place the hot jobs (high power consumption) side by side as shown in Figure 11.11(a). In this case, there is an unacceptable temperature rise in the nearby cores because of lateral heat conduction and additional leakage. However, if the hot jobs are placed far apart as in Figure 11.11(b) then the temperature rise is much lower, and the leakage power is also commensurately lower.

Let us complicate this situation by bringing in performance constraints. Let's say that some jobs need to be placed in close proximity because they access similar data. Now if they are placed far apart, performance will suffer. We can thus add additional performance constraints.

Now, we can further complicate this picture if we consider DVFS, more jobs than cores, variability in leakage power, and dynamic migration of jobs. We thus have a very rich research area in this space.

11.5 Summary and Further Reading

11.5.1 Summary

Summary 10

1. *Power consumption is a very significant issue in modern processors. There are two primary modes of power consumption: dynamic power consumption and leakage power consumption.*
   (a) *Circuits consume dynamic power when there is a transition in the voltage levels (*$0 \rightarrow 1$ *or* $1 \rightarrow 0$*). Upon a transition there is a flow of current to charge or discharge gate capacitances and the resulting resistive loss is known as dynamic power consumption.*
   (b) *An important component of the dynamic power is the* short-circuit power*. When there is a transition in a CMOS transistor's input voltage, there is a short period of time when both the NMOS and PMOS transistors are conducting. For this brief period, there is a short circuit from the supply to the ground. The resultant current flow and power dissipation is known as the short-circuit power consumption.*
   (c) *In a transistor, we typically assume that many interfaces are ideal, which means that there is no current flow through them. However, this is not the case in modern transistors. Some amount of current* leaks through *such interfaces and this is known as the* leakage current*. The resulting power consumption is known as leakage power.*
2. *The dynamic power consumption is proportional to* $\beta CV^2 f$*, where* β *is the activity factor (varies from 0 to 1),* C *is the lumped capacitance,* V *is the supply voltage, and* f *is the frequency.*
3. *The relationship between the voltage and frequency is typically given by the Alpha-power law.*
$$f \propto \frac{(V - V_{th})^{\alpha}}{V}$$
   V *is the supply voltage,* V_{th} *is the threshold voltage, and* α *is a constant between 1.1 and 1.5.*
4. *The process of changing the voltage and frequency to either increase performance or reduce power is known as dynamic voltage frequency scaling (DVFS).*
5. *To compare systems that have different frequencies, typically the energy-delay-square (*ED^2*) metric is used. It is independent of the DVFS setting if some simplistic assumptions are made.*
6. *There are six main sources of leakage power: P-N junction reverse bias current, subthreshold leakage, gate-oxide tunnelling, hot-carrier injection, gate-induced drain leakage, and punchthrough current.*
7. *There is a feedback loop between the temperature and leakage power particularly the subthreshold leakage power. If the temperature increases, the leakage power also increases, and the resultant increase in overall power increases the temperature.*
8. *The basic equation in temperature modelling is the Fourier's equation: the heat flux (power per unit area) is proportional to the gradient of the temperature.*

9. *The final form of the Fourier's equation in Cartesian coordinates is:*

$$k\nabla^2 T + Q' = \rho c_p \frac{\partial T}{\partial t}$$

k is a constant, Q' is the rate of internal power generation, T is the temperature field, ρ is the density, and c_p is the specific heat.

10. *The finite difference approach for solving this equation is to convert it to a linear transformation. The Fourier equation can be simplified as follows for the steady-state case.*

$$\mathbf{T} = \mathbf{AP}$$

11. *An equivalent expression for the transient case is as follows where $\mathbf{C}$ is a diagonal matrix.*

$$\mathbf{GT} + \mathbf{P} = \mathbf{CT}'$$

12. *We can create an electrical analogue of the problem where we can designate the constants of proportionality in our equations as equivalent* thermal resistances *and* thermal capacitances. *Voltage can be mapped to the temperature, and current to power. We can use a standard circuit simulator to compute the thermal profile.*

13. *Using the properties of linearity and superposition, we can define the Green's function that is the impulse response of a unit power source. The temperature profile is given as $T = P \star G$. Here, $\star$ is the convolution operator.*

14. *The common approaches to manage dynamic power are voltage-frequency scaling, clock gating (setting the clock equal to 0), and issue/fetch throttling (reducing the issue or fetch rate).*

15. *For reducing leakage power we can opt for power gating, which means disconnecting the circuit from the power and ground lines. Another approach for reducing leakage in caches is to implement the* drowsy *mode, where the supply voltage is reduced such that the stored value can be maintained but the cell cannot be accessed.*

16. *In general, any approach that reduces power reduces temperature as well. However, there are dedicated schemes to reduce temperature hotspots as well. For example, we can stop the execution completely, or map the jobs in such a manner that high-temperature hotspots do not form. This can lead to very interesting optimisation problems where we can place constraints on the performance and reduce the peak temperature as much as possible. We can additionally couple these problems with energy and DVFS based constraints.*

11.5.2 Further Reading

For power estimation techniques, readers can refer to the extensive survey by Sultan et al. [Sultan et al., 2014], and the following books: [Kaxiras and Martonosi, 2008,Själander et al., 2014]. To get a better understanding of the tools, readers can refer to the design of the Wattch [Brooks et al., 2000] and McPAT [Li et al., 2009] tools. For modelling leakage power one of the most authoritative references is a paper by Roy et al. [Roy et al., 2003].

For temperature estimation, readers can refer to another survey paper by Sultan et al. [Sultan et al., 2019]. Along with this, readers can look at the popular temperature estimation tools such as HotSpot [Huang et al., 2006], HS3D [Hung et al., 2006], and 3D-ICE [Sridhar et al., 2010]. To understand how Green's function

based methods are used in temperature estimation, readers can refer to [Sarangi et al., 2014, Park et al., 2010]. For estimating temperature in 3D chips using the Green's function, readers can find the following reference useful: [Sultan and Sarangi, 2017].

Finally, for understanding power management schemes, some of the most comprehensive references are the e-books [Kaxiras and Martonosi, 2008, Själander et al., 2014], and for dynamic thermal management schemes these references [Kong et al., 2012, Coskun et al., 2008] can be consulted.

Exercises

Ex. 1 — Understand the working of a thermal simulator such as Hotspot or 3D-ICE. Use an architectural simulator to generate the power profile of a 32-core chip running the Parsec benchmarks. Generate the temperature map of the chip for the power profile.

Ex. 2 — Write an algorithm for dynamically assigning threads to cores based on temperature and leakage power in an architectural simulator.

Ex. 3 — Implement clock gating and power gating in an architectural simulator.

Ex. 4 — Implement DVFS for GPUs in an architectural simulator such as GPUTejas.