Chapter 1
Low Power Design: An Introduction

1.1 The Emergence of Power as an Important Design Metric

Through most of the evolution of the electronics and computer industry in the twentieth century, technological progress was defined in terms of the inexorable march of *density* and *speed*. Increasing density resulted from process improvements that led to smaller and smaller geometries of semiconductor devices, so the number of transistors packed into a given area kept increasing. Increasing processing speed indirectly resulted from the same causes: more processing power on the same chip meant more computation resources and more data storage on-chip, leading to higher levels of parallelism and hence, more computations completed per unit time. Circuit and architectural innovations were also responsible for the increased speed. In the case of processors, deeper pipelines and expanding parallelism available led to improvements in the effective execution speed. In the case of memories, density continued to make remarkable improvements; performance improvements were not commensurate, but nevertheless, architectural innovations led to increased bandwidth at the memory interface.

In parallel to the steady march in the density and performance of high-end chips such as processors, a quiet mobile revolution was ignited in the 1990s. The specification for components targeting mobile systems were very different from the high-end processors. These chips had to fit into very tight power budgets defined by battery life considerations, which led to a significant investment in low power design techniques, methodologies, and tools. Starting early 1990s, researchers began to consider modifying the entire electronic design automation tool chain by incorporating power awareness. Even as the complexity of Systems-on-a-Chip (SoC) increased significantly and proceeded in the direction of integration of a large number of modules (cores), the individual cores were still designed to tight power constraints and intelligence was introduced in the overall power management of these SoCs in order to meet chip-level power budgets. Many of these cores were processors in themselves, but these were architected very differently from the standard high-end desktop or server processor – speed of individual cores was heavily compromised to meet power constraints by keeping the design simple. It was a prescient move – the high-end multicore processors of the early 21st century would later adopt the same philosophy.

P.R. Panda et al., *Power-efficient System Design*, DOI 10.1007/978-1-4419-6388-8_1,

Through the late 1990s and early 2000s, the parallel evolution of the two seemingly unrelated lines of products continued unabated. High end processor systems delivered on the traditional density and performance scaling promise through increased architectural sophistication, while the mobile revolution was well under way, spearheaded by the ubiquitous cell phone.

Towards the mid-2000s, important directional changes were visible in the evolution of these technologies. Cell phone makers began packing more and more functionality into their devices, making them look increasingly similar to general purpose programmable devices. All the functionality still had to be delivered within the tight power envelope of a few watts – imposed by the simple physical requirement that the device should fit and stay within the pocket without burning a hole through it. Desktop processors, at the same time, ran up against an unexpected barrier – the *power wall*, which imposed an upper limit on the power dissipation. It turned out that, if the processor power exceeded around 150 W, then the simple fan-based cooling mechanism used in desktop computers might be inadequate. More sophisticated cooling would increase the cost of the systems to inordinate levels – the consumer had got used to price drops with increasing performance over the years. Suddenly, power related innovation became a top concern even in high-end processors, altering the designer's priorities radically after decades of riding the predictable performance and density curves.

A third front had opened with the exploding Internet revolution – the world was getting more and more connected, powered by infrastructural re-organization that was not obviously visible to the end-user, but nevertheless, an important enabling innovation. Large scale distributed storage and processing capabilities had become necessary. To handle the terabytes of data continuously being generated, stored, and retrieved, data had to be organized in non-trivial ways. Similarly, the huge processing and computation power also needed similar architectural paradigm shifts. *Data centers* were the offshoot of these pressing requirements. Designers realized the economies of scale provided by aggregations of tens of thousands of computers stacked into a warehouse-style environment, which would make more efficient utilization of space than the maintenance of a large number of small server rooms in individual companies. Service models began to evolve based on this idea. There was just one issue – power. Large data centers draw megawatts of power. It is well established that the annual energy cost of operating a large data center rivals the cost of the equipment itself. The time had come to start worrying about ways to reduce the energy bills. Further, the concentration of a large number of powerful computers in a data center leads to a very high thermal density, with significant consequences on the cooling technology required to maintain the systems within the operating temperature range, ultimately impacting the cost of maintaining the infrastructure.

As we proceed to the 2010s, it is amply clear that no one can run away from the looming power wall. Far from being limited to battery-operated devices, power and energy efficiency now has to be built into products at every level of abstraction, starting from the semiconductor device level, through circuit and chip levels, all the way to compilers, operating systems, and large data centers. While earlier

generations of hardware designers invariably optimized the product for high performance and reported the power dissipation merely as a documentation exercise to generate complete data sheets for the customer, newer generations of designers now need to aggressively optimize for power – even at the expense of performance. Power is now elevated to what is referred to as a *first class design concern*.

Power awareness on the software side of the computation stack expressed itself as an issue somewhat later, as expected. This followed after appropriate hooks were provided by the hardware. It didn't take long for researchers to realize that a huge opportunity lay ahead for power aware system and application software. This followed the observation that at least some of the hardware power optimizations would work better if there were hints passed on to the hardware by the software and vice versa. For example, a higher level macroscopic view of a system as a whole is available to an application designer or a compiler that is able to view and analyze an extended piece of code, than a processor that is able to view only one small window of instructions at a time. Power efficiency demands are now placed on compilers, application developers, and operating systems.

The need for energy efficiency in electronic and computer systems tracks the global need for energy efficiency in every other walk of life. At a time when the world is acutely aware of the environmental consequences of the huge collective footprint of our individual actions, and struggling to evolve a sustainable solution, it is natural that power efficiency and energy efficiency concerns should permeate into everything we do. With electronic and computer systems beginning to play such a dominant role in our day-to-day lives, they present themselves as perfect targets on which to focus our power efficiency efforts.

1.2 Power Efficiency vs. Energy Efficiency

Since the terms *power efficiency* and *energy efficiency* are both used in literature, often interchangeably, let us look at the elementary definitions of power and energy and clarify the distinction between power and energy efficiency and between the objectives of power and energy optimizations.

Figure 1.1 shows the instantaneous power dissipated by a system S as a function of time. The power $P(t)$ will vary from time to time because the work done inside the system typically changes with time. In electronic systems, power is, in turn, determined by the current drawn by the system and the voltage at which it operates, as a function of time:

$$P(t) = V(t) \times I(t) \tag{1.1}$$

where V and I are the instantaneous voltage and current respectively. If the external supply voltage remains constant, as is common in many systems, power is directly proportional to the current drawn, and the power curve essentially tracks the variation of the current over time.

What about energy dissipation by the same system S? Power is simply the rate at which energy is dissipated, so the energy dissipated over a period $t = 0$ to $t = T$ is given by:

$$E = \int_0^T P(t)\, dt \tag{1.2}$$

This corresponds to the area under the $P(t)$ curve as indicated in Fig. 1.1.

A system S' that delivers the same work as the one in Fig. 1.1 in the same duration T, but is characterized by a different power curve $P'(t)$, as shown in Fig. 1.2(a), would be called more power-efficient because the P' curve is always lower than P. What about its energy dissipation? Its energy E' corresponds to the area under the P' curve, which is clearly less than E. Hence, E' is also more energy-efficient.

On the other hand, let us consider the power characteristic of a different system S'' shown in Fig. 1.2(b). System S'' has a power curve P'' that is under P most of

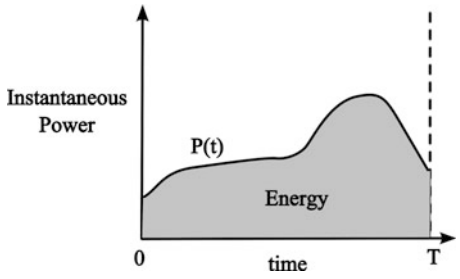

Fig. 1.1 The relationship between power and energy: the area under the power curve over a time interval gives the energy consumed

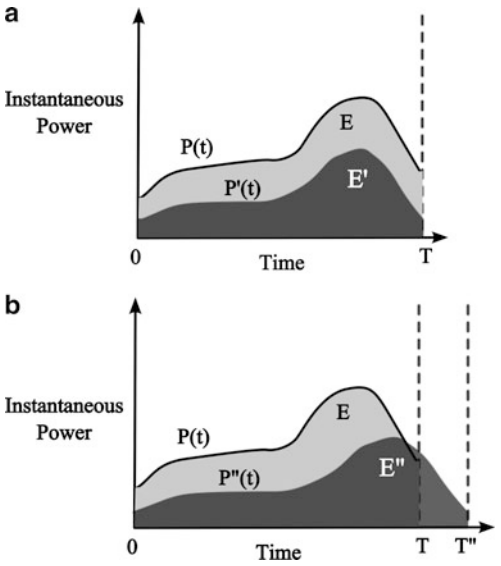

Fig. 1.2 Power efficiency and energy efficiency. Three systems S, S', and S'' have power curves P, P', and P'' respectively. (**a**) S' is more power-efficient and more energy-efficient than S. (**b**) S'' has lower peak power than S. S'' has lower average power/energy than S in time period $[0, T'']$ if $E'' < E$

the time, but it takes longer to complete its task ($T'' > T$). The energy dissipated is now given by:

$$E'' = \int_0^{T''} P''(t)\, dt \tag{1.3}$$

Which of S, S'' is more power efficient? Which is more energy-efficient?

The energy efficiency question is easier to answer – S'' is more energy efficient if $E'' < E$. This is a clearer question because energy corresponds to the total work done by a system, independent of completion time.

Power, on the other hand, is intricately tied to completion times, and the definition of power-efficiency needs to be clarified in a given context. Two major power efficiency criteria are commonly relevant:

1. Peak power
2. Average power

The peak power criterion is sometimes externally imposed on a system. If minimizing peak power is the objective, then S'' is more efficient. If a peak power constraint is imposed on the system, then the design has to be adjusted to ensure that the power curve is always under the horizontal line corresponding to the power constraint, irrespective of how long it takes for completion.

Average power is defined with respect to a time interval. In Fig. 1.2(b), if we take the time interval $[0, T'']$, then we have:

$$\text{Average Power for } \ S = \frac{E}{T''} \tag{1.4}$$

$$\text{Average Power for } \ S'' = \frac{E''}{T''} \tag{1.5}$$

Note that, for a fair comparison of the two systems, the time durations should be the same. S' has completed its task at time T, and is consuming zero power between T and T'', which should be accounted for in the average power calculation. Since the two denominators are the same, the relationship between the average power dissipation between the two systems is identical to the relationship between the total energy.

Requirements for energy efficiency or average power reduction usually stem from battery life considerations. To a level of approximation, battery capacity can be viewed in terms of a fixed amount of total energy that it can supply.

Requirements for peak power usually stem from two sources:

- external power supply may impose a constraint on the maximum instantaneous power that can be supplied; or
- there may be a cap on the maximum temperature (which is indirectly determined by peak power) that a device is allowed to reach. This could be applicable to both mobile devices as well as powerful server computers.

In this book we discuss different classes of mechanisms that target the reduction of either energy (average power) or peak power. The actual objective is clarified from the context in which it appears.

1.3 Power-Performance Tradeoff

Figure 1.2(b) illustrates an important trade-off between peak power and performance that is not merely theoretical. The trade-off is real and has tremendously influenced the direction taken by the mobile computer industry. The explosion of interest in ultra-portable computers (also referred to as *netbooks*) in recent times, along with the evolution of cell phones into more sophisticated gadgets that make them look more like handheld computers, clearly shows that there is a huge potential for the market segment where computer systems are held within a tight power budget while sacrificing performance. Netbooks run on processors with very low peak power and correspondingly weak performance; their acceptance in the market may imply that the average set of applications run by many users just don't require very high speed. Instead of raw speed, other important features such as network connectivity, mobility, and battery life are valued. The race to deliver higher and higher raw performance may have transformed into a more nuanced battle, with new battle grounds being created that squarely put power and energy at the center.

Design metrics such as *energy-delay product* are some times used to evaluate the relative merits of two implementations in a way that accounts for both energy and performance [1]. Such metrics help in discriminating between alternatives that are equivalent in total energy dissipation but different in performance. Energy-delay products for the two systems above are given by:

$$\text{Energy-Delay Product for } S = E \times T$$
$$\text{Energy-Delay Product for } S'' = E'' \times T''$$

The term *performance-per-watt* is sometimes used in the industry to characterize energy efficiency of hardware [2]. This term is effectively a direct or indirect computation of $1/E$, with E defined as above. The *performance* measure here can be interpreted in different ways, for example, execution frequency or throughput of computation. If performance is interpreted as the inverse of latency T, then we have:

$$\text{Performance-per-watt } = \frac{1}{T} \times \frac{1}{P} = \frac{1}{E}$$

Higher performance-per-watt means more energy-efficient (lower total energy) and vice versa.

The performance-per-watt metric gives equal emphasis to power and performance, while the energy-delay product ($E \times T = P \times T^2$) gives higher emphasis to performance (the latter varies quadratically with delay).

1.4 Power Density

Power density is an additional important concern that is not captured in a straightforward way by the power and energy efficiency metrics discussed above. This refers to the fact that the distribution of power dissipation across a chip is never uniform because different modules exhibit different degrees of activity in any fixed period of time. Thus, the power density – the dynamic energy dissipated per unit area – is non-uniformly distributed on the chip, leading to *hotspots* where the power density is significantly high. Higher power density leads to higher temperatures, and a non-uniform power density leads to a non-uniform temperature variation on the chip, leading to some associated problems. Even if the average and peak power numbers are acceptable, power density variations can cause regions on the chip to have unacceptably high temperatures leading to failures. Thus, the chip designer also needs to target a relatively uniform distribution of power density in the resulting chip.

1.5 Power and Energy Benchmarks

The rising importance of power and energy motivated the eventual development of benchmarking efforts targeting the evaluation of different classes of systems with respect to power and energy. *Powerstone* [3] was among the early reported efforts targeting power evaluation of mobile embedded systems. The Powerstone benchmark contained applications selected from the following domains: automobile control (*auto* and *engine*), signal and image processing (*fir_int* – integer FIR filter, *jpeg* – JPEG decompression, *g721* – voice compression, and *g3fax* – fax decode), graphics (*bilt*), and several utilities (*compress* – UNIX file compression, *crc* – cyclic redundancy check, *bilv* – bit vector operations, *des* – data encryption, etc.).

Joulesort [4] is a more modern benchmarking effort aimed at power evaluation of high-end server, cluster, and data center class computer systems. The benchmark consists of essentially an application for sorting a randomly permuted collection of records. Different categories of the benchmark contain different input data sizes (10 GB, 100 GB, and 1 TB). Thus, the benchmark has I/O as the primary emphasis, with secondary emphasis on CPU and memory. Joulesort is good at exercising whole system level power efficiencies. The proposed evaluation metric used here is the energy consumed for processing a fixed input size.

1.6 Power Optimizations at the System Level

In this book we will explore power optimizations at the *system level* of abstraction. Since the word *system* is very generic, it is important to clarify its interpretation as used in this book. The following different levels of abstraction could possibly

be considered in the context of power optimizations. They are arranged in the increasing level of granularity.

Device level. This refers to the choice of the appropriate semiconductor materials and processes used to fabricate transistors and other devices in this class which form the building blocks of electronic technology.

Circuit Level. This refers to the interconnection of transistors and the related class of components, along with the choice of appropriate geometry for these devices.

Gate Level. This refers to the logic level of abstraction, where the components (*gates*) implement simple combinational and sequential functions (such as AND, MUX, flip-flops, etc.). A logic synthesis tool generates an optimized logic netlist starting from an unoptimized gate level description, or some other simple format such as a truth table.

Register Transfer Level (RTL). This refers to a description that is usually expressed in a Hardware Description Language (HDL), containing a cycle-accurate description of the hardware to be designed. RTL synthesis tools automate the generation of a gate netlist from an RTL HDL description. The architecture view of a processor or other hardware falls in this level of abstraction.

Behavioral Level. At this level, the hardware is still described in an HDL, but the level of detail is lesser than RTL. For example, we may specify an algorithm implemented in hardware, but indicate nothing about which operation is performed in which clock cycle. A behavioral synthesis tool takes a behavioral level model as input, performs a series of *high level synthesis* optimizations, and finally generates an RTL HDL design, to be processed further.

Transaction Level. At this level – pertaining to SoC design – the design is no longer a pure hardware design. In fact, the decision of which part of the specification will go into hardware and which into software, has not been finalized yet. The details of the computation in individual blocks and communication across blocks of the SoC are not specified yet.

Application and System Software Level. At this level, we have moved to software executing on a processor hardware. This level encompasses all application software and key system software such as compilers that have a deeper view of a single application, and operating systems that have a wider view of the run-time environment.

Full System, Server, and Data Center Level. This level of abstraction refers to an aggregation of electronic and computer systems that can be considered complete in some sense - for example, cell phones, laptops, dedicated servers, all the way up to data centers.

Our focus on system-level design covers the behavioral and higher levels. That is, in this book we focus on the Behavioral Level, Transaction Level, Application and System Software Level, and Full System, Server, and Data Center Levels. Key power optimization strategies at lower levels of abstraction are summarized briefly in Chapter 2.

1.7 Organization of this Book

In this book we explore various mechanisms that have been proposed for reducing power dissipation in electronic and computer systems. Figure 1.3 gives a pictorial view of the contents of the book. The organization of the chapters is as follows.

In Chapter 2 ("Basic Low Power Digital Design") we introduce the avenues for power dissipation in electronic systems based on CMOS processes, and the basic techniques employed for reducing dynamic and static power. The chapter covers

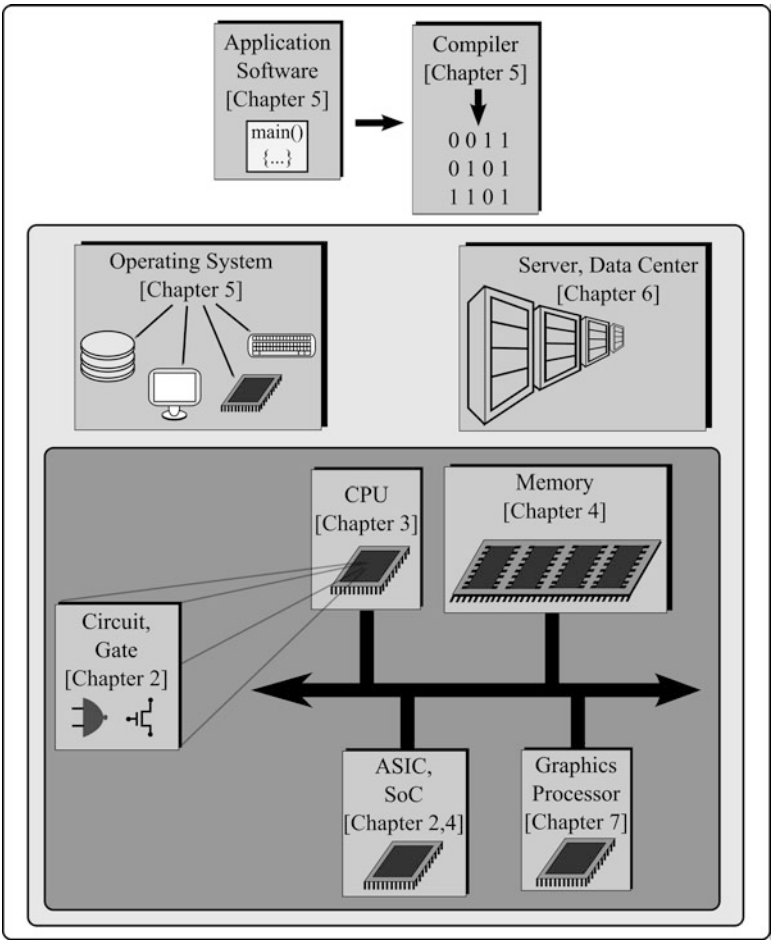

Fig. 1.3 Contents and organization of this book. Chapter 2 outlines the electronic foundations and basic synthesis/automation techniques. Chapter 3 and 4 focus on CPU and memory respectively. Chapter 5 discusses power-aware operating systems, compilers, and applications. Chapter 6 covers servers and data centers. Chapter 7 describes low power graphics processors

the electronic foundations that form the basis of most of the higher level power optimizations discussed in subsequent chapters.

In Chapter 3 ("Power-efficient Processor Architecture") we delve into the details of modern processor architecture and point out some of the major strategies for reducing power in various components. While circuit level enhancements for power reduction naturally apply to processor design, our focus here is on the architecture-level modifications that reduce power.

In Chapter 4 ("Power-efficient Memory and Cache") we discuss the basics of memory architecture and point out several techniques for reducing memory power in different contexts. In addition to power-aware cache memory design in the context of processors, we also discuss power reduction opportunities afforded by scratch pad memory and banked memory in the SoC context, as well as other components such as translation look-aside buffers and dynamic memory (DRAM).

In Chapter 5 ("Power Aware Operating Systems, Compilers, and Application Software") we move up another level in the compute stack – system software such as operating systems and compilers, and application software. Once the proper power optimization hooks are provided by the hardware, new opportunities open up with regard to their exploitation in software, starting from power-aware compilers.

In Chapter 6 ("Power Issues in Servers and Data Centers") we examine power awareness issues being introduced into high-end server-class computers and aggregations of servers constituting data centers. High level operating system features such as task allocation can now take power into account, exploiting mechanisms provided by the hardware for controlling performance and power states of individual computers.

In Chapter 7 ("Low Power Graphics Processors") we do a detailed study of system level power optimizations in graphics processors. The chapter gives a brief introduction to the graphics processing domain and proceeds to illustrate the application of several of the ideas introduced in previous chapters in the context of a complex processing system whose components bear some similarity to general purpose processors, but nevertheless, the overall architecture is application specific.

References

1. Gonzalez, R., Horowitz, M.: Energy dissipation in general purpose microprocessors. IEEE Journal of Solid-State Circuits **31**(9), 1277–1284 (1996)
2. Laudon, J.: Performance/watt: the new server focus. SIGARCH Computer Architecture News **33**(4), 5–13 (2005)
3. Malik, A., Moyer, B., Cermak, D.: The M·CORE^{TM} M340 unified cache architecture. In: Proceedings of the IEEE International Conference On Computer Design: VLSI in computers & processors, pp. 577–580 (2000)
4. Rivoire, S., Shah, M.A., Ranganathan, P., Kozyrakis, C.: Joulesort: a balanced energy-efficiency benchmark. In: SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pp. 365–376. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1247480.1247522