To be presented at the High Performance Computer Conference, Bangalore, India, Dec. 00

Power: A First Class Design Constraint for Future Architectures

Trevor Mudge

Computer Science and Electrical Engineering
University of Michigan, Ann Arbor[1]

Abstract. In many mobile and embedded environments power is already the leading design constraint. This paper argues that power will also be a limiting factor in general purpose high-performance computers. It should therefore be considered a "first class" design constraint on a par with performance. A corollary of this view is that the impact of architectural design decisions on power consumption must be considered early in the design cycle — at the same time that their performance impact is considered. In this paper we summarize the key equations governing power and performance and use them to illustrate some simple architectural ideas for power savings. The paper then presents two contrasting research directions where power is important. We conclude with a discussion of the tools needed to conduct research into architecture-power trade-offs.

1 Introduction

The limits imposed by power consumption are becoming an issue in most areas of computing. The need to limit power consumption is readily apparent in the case of portable and mobile computer platforms — the laptop and the cell phone being the most common examples. But the need to limit power in other computer settings is becoming important too. A good example is the case of server farms. They are the warehouse-sized buildings that internet service providers fill with servers. A recent analysis presented in [1] has shown that a 25,000 sq. ft. server farm with about 8,000 servers will consume 2MW. Further it was shown that about 25% of the total running cost of such a facility is attributable to power consumption, either directly or indirectly.

It is frequently reported that the net is "growing exponentially," it follows then that the server farms will match this growth and with them the demand for power. A recent article in the Financial Times noted that 8% of power consumption in the US is for information technology (IT). If this component is set to grow exponentially without check it will not be long before the power for IT will be greater than for all other uses combined.

1. *Author's address:* Dept. EECS, The University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109-2122, USA. *Tel:* +1 (734) 764.0203. *Fax:* +1 (734) 468.0152. *E-mail:* tnm@eecs.umich.edu. *WWW:* http://www.eecs.umich.edu/~tnm.

7th International Conference on High Performance Computing. December, 2000. Bangalore, India pp. 215-224.

To get an idea of the trends in power consumption of today's processors consider the following table taken from [2]. The rapid growth in power consumption is obvious.

Table 1: Power Trends for the Compaq Alpha

Alpha model	Power (W)	Frequency (MHz)	Die size (mm^2)	Supply voltage
21064	30	200	234	3.3
21164	50	300	299	3.3
21264	90	575	314	2.2
21364	>100	>1000	340	1.5

The growth in power density of the die is equally alarming. It is growing linearly, so that the power density of the 21364 has reached about 30 W/cm^2 — three times that of a typical hot plate. This growth has occurred in spite of process and circuit improvements. Clearly, trading high power for high performance cannot continue, and architectural improvements will also have to be added to process and circuit improvements if the growth in power is to be contained. The only exceptions will be one-of-a-kind supercomputers built for special tasks like weather modelling.

In this introductory discussion we have briefly illustrated that power will be a limitation for most types of computers, not just ones that are battery powered. Furthermore, we have argued that system and architectural improvements will also have to be added to process and circuit improvements to contain the growth in power. It therefore seems reasonable that power should be dealt with at the same stage in the design process as architectural trade-offs

In the remainder of the paper we will expand on this theme starting, in the next section, with a simple model of power consumption for CMOS logic.

2 Power equations for CMOS logic

There are three equations that provide a model of the power-performance trade-offs for CMOS logic circuits. The equations are frequently discussed in the low-power literature and are simplifications that capture the essentials for logic designers, architects, and systems builders. We focus on CMOS because it will likely remain the dominant technology for the next 5-7 years. In addition to applying directly to processor logic and caches, the equations are also relevant for some aspects of DRAM chips. The first equation defines power consumption:

$$P = ACV^2f + \tau AVI_{short}f + VI_{leak} \qquad (1)$$

There are three components to this equation. The first is perhaps the most familiar. It measures the dynamic power consumption caused by the charging and discharging of the capacitive load on the output of each gate. It is proportional to the frequency of the operation of the system, *f*, the activity of the gates in the system, *A* (some gates may not switch every clock), the total capacitance seen by the gate outputs, *C*, and the square of the supply voltage, *V*. The second term captures the power expended due to the short-circuit current, I_{short}, that momentarily, τ, flows between the supply voltage and ground when the output of a CMOS logic gate switches. The third term measures the power lost due to leakage current that is present regardless of the state of the gate.

In today's circuits the first term dominates, immediately suggesting that the most effective way to reduce power consumption is to reduce the supply voltage, *V*. In fact, the quadratic dependence on *V* means that the savings can be significant: halving the voltage will reduce the power consumption to one quarter of it original value. Unfortunately, this comes at the expense of performance, or, more accurately, maximum operating frequency, as the next equations shows:

$$f_{max} \propto (V - V_{threshold})^2 / V \qquad (2)$$

In other words, the maximum frequency of operation is roughly linear in *V*. Reducing it will limit the circuit to a lower frequency. But reducing the power to one quarter of its original value will only cut the maximum frequency in half. There is an important corollary to equations (1) and (2) that has been widely noticed: If a computation can be split in two and run as two parallel independent tasks, this form of parallel processing has the potential to cut the power in half without slowing the computation.

We can lessen the effect of reducing voltage *V* in equation (2) by reducing $V_{threshold}$. In fact this must occur to allow proper operation of low voltage logic circuits. Unfortunately, reducing $V_{threshold}$ increases the leakage current, as the third equation shows:

$$I_{leak} \propto \exp(-V_{threshold} / (35mV)) \qquad (3)$$

Thus, this is a limited option for countering the effect of reducing *V*. It makes the leakage term in the first equation appreciable.

Summary: There are three important points that can be taken away from the above model. They are:

1. Reducing voltage has a significant effect on power consumption — $P \propto V^2$.
2. Reducing activity does as well — in the simplest case, this means turning off parts of the computer that are not being used.
3. Parallel processing is good if it can be done efficiently — independent tasks are ideal.

2.1 Other figures of merit related to power

The equation for power consumption given in (1) is an average value. There are two important cases where more information is needed. The first is peak power. Typically, systems have an upper limit, which if exceeded will lead to some form of damage — average power does not account for this. The second is dynamic power. Sharp changes in power consumption can result in inductive effects that can result in circuit malfunction. The effect is seen in "di/dt noise"— again, average power does not account for this.

The term "power" is often used quite loosely to refer to quantities that are not really power. For example, in the case of portable devices, the amount of energy used to perform a computation may be a more useful measure, because a battery stores a quantity of energy, not a quantity of power. To refer to a processor as being “lower power” than another may be misleading if the computation takes longer to perform. The total energy expended may be the same in both cases — the battery will be run down by the same amount in both cases[1]. This leads to the idea of energy/operation. A processor with a low energy/operation is sometimes incorrectly referred to as “low power.” In fact, the inverse of this measure, MIPS/W is frequently used as a figure of merit for processors intended for mobile applications [3].

Although the MIPS/W is widely used as a figure of merit (higher numbers are better), it too can be misleading because cutting the power consumption in half reduces the frequency of operation by much less, because of the quadratic term in (1), as we have discussed. This has lead Gonzalez and Horowitz [4] to propose a third figure of merit: energy*delay. This measure takes into account that, in systems whose power is modelled by (1), it is possible to trade a decrease in speed for higher MIPS/W. Unfortunately, the bulk of the literature uses MIPS/W or simply Watts, so we will continue this convention recognizing that occasionally it may suggest misleading trade-offs where “quadratic” devices like CMOS are concerned. Finally, it should be noted that if the computation under consideration must finish by a deadline, slowing the operation may not be an option. In these cases a measure that combines total energy with a deadline is more appropriate.

3 Techniques for reducing power consumption

In this section we will survey some of the most common techniques that systems designers have proposed to save power. The scope of this brief overview includes logic, architecture and operating systems.

3.1 Logic

There are a number of techniques at the logic level for saving power. The clock tree can consume 30% of the power of a processor — the early Alpha 21064 exceeded this. Therefore, it is not surprising that this is an item where a number of power saving techniques have been developed.

3.1.1 Clock gating

This is a technique that has been widely employed. The idea is to turn off those parts of the clock tree to latches or flip-flops that are not being used. Until a few years ago gated clocks were considered poor design practice because the gates in the clock tree can exacerbate clock skew. However, more accurate timing analyzers and more flexible design tools have made it possible to produce reliable designs with gated clocks and this technique is no longer frowned upon.

1. This is a simplification because the total energy that can be drawn from a battery after it has been charged depends, to some extent, on the rate at which the energy is drawn out. We will ignore this effect for our simplified analysis.

3.1.2 Half-frequency and half-swing clocks

The idea with the half-frequency clock is to use both edges of the clock to synchronize events. The clock can then run at half the frequency of a conventional clock. The drawbacks are that the latches are much more complex and occupy more area, and the requirements on the clock are more stringent.

The half-swing clock swings only half of *V*. It also increases the requirements on latch design, and it is difficult to employ in systems where *V* is low in the first place. However, the gains from lowering clock swing are usually greater than for clocking on both edges.

3.1.3 Asynchronous logic

The proponents of asynchronous logic have pointed out that their systems do not have a clock and therefore stand to save the considerable power that goes into the clock tree. However, there are drawbacks with asynchronous logic design. The most notable is the need to generate completion signals. This means that additional logic must be employed at each register transfer — in some cases a double rail implementation is employed. Other drawbacks include difficulty of testing and absence of design tools.

There have been several projects to demonstrate the power saving of asynchronous systems. The Amulet, an asynchronous implementation of the ARM instruction set architecture, is one of the most successful [5]. It is difficult to draw definitive conclusions because it is important to compare designs that are realized in the same technologies. Furthermore, the asynchronous designer is at a disadvantage because, as noted, today's design tools are geared for synchronous design. In any case, asynchronous design does not appear to offer sufficient advantages for there to be a wholesale switch to it from synchronous designs.

The area where asynchronous techniques are likely to prove important is in globally asynchronous, locally synchronous systems (GALS). These reduce clock power and help with the growing problem of clock skew across a large chip, while still allowing conventional design techniques for most of the chip.

3.2 Architecture

The focus of computer architecture research, typified by the work presented at the International Symposia on Computer Architecture and the International Symposia on Microarchitecture, has been on high performance. There have been two important themes pursued by this research. One has been to exploit parallelism, which we have seen can help reduce power. The other is to employ speculation — computations are allowed to proceed beyond dependent instructions that may not have yet completed. Clearly, if the speculation is wrong, energy has been wasted executing useless instructions. Branch prediction is perhaps the best known example of speculation. If there is a high degree of confidence that the speculation will be correct, then it can provide an increase in the MIPS/W figure. However, for this to be the case the confidence level must often be so high that speculation is rarely employed as a means to reduce MIPS/W or power [7].

The area where new architectural ideas can most profitably contribute to reducing power is in reducing the dynamic power consumption term, specifically the activity factor, *A*, in (1).

3.2.1 Memory systems

The memory system is a significant source of power consumption. For systems with relatively unsophisticated processors, cache memory can dominate the chip area. There are two sources of power loss in memory systems. First there is the dynamic power loss, due to the frequency of memory access. This is modelled by the first term[1] in (1). The second is the leakage current — the third term in (1).

There have been several proposals for limiting the dynamic power loss in memories by organizing memory so that only parts of it are activated on a memory access. Two examples are the filter cache and memory banking [8]. The filter cache is a small cache placed in front of the L1 cache. Its purpose is to intercept signals intended for the main cache. Its hit rate does not have to be very high, and its access time need be no faster than the L1 cache. Even if it is hit only 50% of the time, then the power saved is half the difference between activating the main cache and the filter cache. This can be significant. The second example, memory banking, is currently employed in some low power designs. The idea is to split the memory into banks and activate only the bank presently in use. It relies on the reference pattern having a lot of spatial locality and, thus, is more suitable for instruction cache organization.

There is not much that can be done by the architect or systems designer to limit leakage, except to shut the memory down. This is only practical if the memory is going to be unused for a relatively long time because it will lose state and therefore must be backed up to disk. This type of shut down (often referred to as sleep mode) is usually handled by the operating system.

3.2.2 Buses

Buses are a significant source power loss. This is especially true for inter-chip buses. These are often very wide — the standard PC memory bus includes 64 data lines and 32 address lines. Each requires substantial drivers. It is not unusual for a chip to expend 15-20% of its power on these inter-chip drivers.

There have been several proposal for limiting this swing. One idea is to encode the address lines into a Gray code. The reasoning is that address changes (particularly from cache refills) are often sequential and counting in Gray code switches the least number of signals [10].

It is straightforward to adapt other ideas to this problem. Transmitting the difference between successive address values achieves a similar result to the Gray code. More generally, it has been observed that the address lines can be reduced by compressing the information in them [9]. These techniques are best suited to inter-chip signalling, because the encoding can be integrated into the memory controllers.

1. The second term in *(1)* can also be lumped together with this.

Continuing with the code compression concept, it has been shown that significant instruction memory savings results if the program is stored in compressed form and decompressed on the fly (typically on a cache miss) [11]. The reduction in memory size can translate to power savings. It also reduces the frequency of code overlays, another source of power loss and a technique still used in many digital signal processing (DSP) systems.

3.2.3 Parallel Processing and Pipelining

As we noted above, a corollary of our power model is that parallel processing can be an important technique for reducing power consumption in CMOS systems. Pipelining does not share this advantage, because the concurrency in pipelining is achieved through increasing the clock frequency which limits the ability to scale the voltage (2). This is an interesting reversal, because the microarchitecture for pipelining is simpler than for parallel instruction issue and, therefore, it has traditionally been the more common of the two techniques employed to speed up execution.

The degree to which computations can be parallelized varies widely. Some are "embarrassingly parallel." They are usually characterized by identical operations on array data structures. However, for general purpose computations typified by the SPEC benchmark suite, there has been little progress on discovering parallelism. This is reflected in the fact that successful general purpose microprocessors rarely issue more than three or four instructions at once. Increasing instruction level parallelism is not likely to offset the loss due to hazards inhibiting efficient parallel execution. However, it is likely that future desktop architectures will have shorter pipes.

In contrast, common signal processing algorithms often possess a significant degree of parallelism. This is reflected in the architecture of DSP chips, which is notably different from desktop or workstation architectures. DSPs typically run at much lower frequencies and exploit a much higher degree of parallelism. Parallelism and direct support for a multiply-accumulate (MAC) operation, which occurs with considerable frequency in signal processing algorithms, means that in spite of their lower clock rates they can achieve high MIPS ratings. An example is given by Analog Devices 21160 SHARC DSP. It can achieve 600 Mflops using only 2W on some DSP kernels.

3.3 Operating System

The quadratic voltage term in (1) means, as we have noted several times, that reducing voltage has great benefit for power savings. A processor does not have to run at it maximum frequency all the time to get its work done. If the deadline of a computation is known it may be possible to adjust the frequency of the processor and reduce the supply voltage. For example, a simple MPEG decode runs at a fixed rate determined by the screen refresh (usually once every 1/30th of a second). A processor responsible for this work could be adjusted to run so that it does not finish ahead of schedule and waste power.

It is very difficult to automatically detect periods where voltage can be scaled back, so current proposals are to provide an interface to the operating system that allows it to control the voltage. The idea is for the application to use these operating system functions to "schedule" its voltage needs [12]. This is a very effect way to save power because it works directly on the quadratic term in equation (1). Support for voltage scaling has already found its way into the next generation of StrongARM microprocessors from Intel, the XScale.

4 What can we do with a high MIPS/W device?

The obvious applications for processors with a high MIPS/W are in mobile computing. The so-called 3G mobile phones that we can expect in the near future will have to possess remarkable processing capabilities. The 3G phones will communicate over a packet-switched wireless link at up to 2 Mbs. The link will support both voice and data and will be connected all the time. There are plans to support MPEG4 video transmission as well as other data intensive applications.

Today’s cell phones are built around two processors: a general purpose computer and a DSP engine. Both have to be low power, the lower the better. A common solution is to use an ARM processor for the general purpose machine and a Texas Instrument’s DSP chip. For 3G systems both of these processors will have to be much more powerful without sacrificing battery life. In fact, the processing requirements, given the power constraints, are beyond the present state-of-the-art. The design of such systems where power is a first class design constraint will be one of the next major challenge for computer architects. Furthermore, the immense number of units that will be sold — there are hundreds of millions of cell phones in use today — means that this platform will take over from the desktop as the defining application environment for computers as a whole.

The two-processor configuration of the cell platform has arisen out of the need to have a low power system that can perform significant amounts of signal processing and possess general purpose functionality for low-resolution display support, simple data base functions, and the protocols associated with cell-to-phone communications. From an architectural point of view this is not a particularly elegant solution and a “convergent” architecture that can handle the requirements of both signal processing and general purpose computing may be a cleaner solution. However, from a power perspective it may be easier to manage the power to separate components; either one can be easily turned off when not required. There are many more trade-offs to be studied.

While the cell phone and its derivatives will become the leading user of power efficient systems, it is by no means the only place where power is key, as we saw in the introduction. To go back to the server farm example, consider one of the servers: It has a workload of independent programs. Thus parallelism can be used without the inefficiencies often introduced by the need for intra-program communication and synchronization — multiprocessors are an attractive solution. A typical front-end server that handles mail, web pages, and news, has an Intel compatible processor, 32M bytes of memory, an 8G byte disk and requires about 30W of power. Assume the processor is an AMD Mobile K6 with a total of 64K bytes of cache running at 400MHz. It is rated at 12W (typical). Compare this to the recently announced Intel XScale, which is Intel’s next generation of StrongARM processor. It has the same total cache size but consumes only 450mW at 600MHz. (It can run from about 1GHz to below 100MHz; at 150Mhz it consumes just 40mW.) If we replace the K6 with 24 XScales we have not increased power consumption. For the K6 to process as many jobs as the 24-headed multiprocessor, it will have to have an architectural efficiency (e.g., SPECmarks) that is about 24 times that of an XScale.

There is a lot to disagree with in the above analysis. For example, the much more complex processor-memory interconnect required by the multiprocessor is not accounted for, nor has any consideration been given to the fact that the individual jobs may have an unacceptable response time. However, the point is to show that if power is the chief design constraint, then a low power but non-trivial processor like the XScale can introduce a new perspective into computer architecture. Consider the above analysis if we replaced the K6 with a 100W Compaq 21364: it would need to be 200 times as efficient.

5 Conclusion

We have made an argument for power to be a first class design constraint. We have also listed a number of ways that systems designers can contribute to satisfying this constraint. There is much more research to be done. Power aware design is no longer exclusively the province of the process engineer and circuit designer, although their contributions are crucial. By one account (see the talk by Deo Singh in [1]) we need architects and systems level designers to contribute a 2x improvement in power consumption per generation. This improvement is required in addition to the gains from process and circuit improvements.

To elevate power to a first class constraint, it needs to be dealt with early on in the design flow, at the same point that architectural trade-offs are being made. This is the point where cycle accurate simulation is performed. This is problematic because accurate power determination can only be made after chip layout has been performed. However, very approximate values are usually acceptable early on in the design flow provided they accurately reflect trends. In other words, if a change is made in the architecture the approximate power figure should reflect a change in power that is in the correct direction.

Several research efforts are under way to insert power estimators into cycle level simulators. They typically employ event counters to obtain frequency measures for components of the architecture. These components are items such as adders, caches, decoders, and buses, for which an approximate model of power can be obtained. An early example was developed by researchers at Intel [13]. Others are Wattch [14] and SimplePower [15]. All three are based on the SimpleScalar simulator that is widely used in academe [16]. A fourth effort, PowerAnalyzer, under development by the author, T. Austin and D. Grunwald is expanding on the work in [13]. PowerAnalyzer will also provide estimates for di/dt noise and peak power [17].

Acknowledgment. The author was supported by contract from the Power Aware Computing and Communications program at DARPA. #F33615-00-C-1678

References

[1] D. Singh and V. Tiwari. Power Challenges in the internet World. *Cool Chips Tutorial: An Industrial Perspective on Low Power Processor Design*, Eds. T. Mudge, S. Manne, D. Grunwald, held in conjunction with MICRO 32, Haifa Israel, Nov. 1999, pp.8-15. (http://www.eecs.umich.edu/~tnm/cool.html)

[2] K.Wilcox and S. Manne. Alpha processors: A history of power issues and a look to the future. *Cool Chips Tutorial: An Industrial Perspective on Low Power Processor Design*, Eds. T. Mudge, S. Manne, D. Grunwald, held in conjunction with MICRO 32, Haifa, Israel, Nov. 1999, pp.16-37. (http://www.eecs.umich.edu/~tnm/cool.html)

[3] Chart Watch: Mobile Processors. Microprocessor Rept, vol. 14, archive 3, Mar. 2000, p.43

[4] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Jour. of Solid-State Circuits*, Sep. 1996, pp. 1277-1284.

[5] S. Furber, J. Garside, and S. Temple. *Power Saving Features in Amulet2e. Power-Driven Microarchitecure Workshop*, Eds. T. Mudge, S. Manne, D. Grunwald, held in conjunction with ISCA 98, Barcelona, Spain, June 1998. (http://www.cs.colorado.edu/~grunwald/LowPowerWorkshop/agenda.html)

[6] S. Moore, et al. Self Calibrating Clocks for Globally Asynchronous Locally Synchronous Systems. *Int. Conf. on Computer Design*, Austin, Texas, Sep. 2000.

[7] S. Manne, A. Klauser and D. Grunwald. Pipeline gating: Speculation control for energy reduction. *Proc. 25th Int. Symp. Computer Architecture,* Barcelona, Spain, June 1998, pp. 132-141.

[8] M. Johnson, M. Gupta and W. Mangione-Smith. Filtering memory references to increase energy efficiency. *IEEE Trans. on Computers*, vol. 49, no. 1, Jan. 2000, pp. 1-15.

[9] A. Park, M. Farrens and G. Tyson,. Modifying VM hardware to reduce address pin requirements. *Proc. 25th Int. Symp. Computer Architecture,* Portland, Oregon, Dec. 1992, pp. 1-4.

[10] L. Benini, et al. Address bus encoding techniques for system-level power optimization. *Proc. 1998 Design Automation and Test in Europe (DATE '98),* pp. 861-866.

[11] C. Lefurgy, E. Piccininni, and T. Mudge. Reducing code size with run-time decompression. *Proc. 6th Int. Symp. on High-Performance Computer Architecture*, Jan. 2000, pp. 218-227.

[12] T. Pering, T. Burd, and R. Brodersen, Voltage scheduling in the lpARM microprocessor System, *Proc. 2000 Int. Symp. on Low Power Electronics and Desig*n, July 2000.

[13] G. Cai and C. Lim. Architectural level power/performance optimization and dynamic power estimation. *Cool Chips Tutorial: An Industrial Perspective on Low Power Processor Design*, Eds. T. Mudge, S. Manne, D. Grunwald, held in conjunction with MICRO 32, Haifa Israel, Nov. 1999, pp.90-113. (http://www.eecs.umich.edu/~tnm/cool.html)

[14] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. *Proc. 27th Int. Symp. Computer Architecture*, Vancouver, Canada, pp. 83-94.

[15] N. Vijaykrishnan, et al. Energy-driven integrated hardware-software optimizations using SimplePower. *Proc. 27th Int. Symp. Computer Architecture*, Vancouver, British Columbia, Canada, June 2000, pp. 95-106.

[16] D. Burger and T. Austin. *The SimpleScalar toolset, version 2.0.* Tech Rept. Computer Science Dept., Univ. Wisconsin, June 1997. (see also http://www.simplescalar.org)

[17] http://www.eecs.umich.edu/~tnm/power/power.html