Low Power Characteristics of Single-Core and Multi-core Systems

Mark Ewan Adam
ECE Department
George Mason University
Fairfax, VA
madam2@gmu.edu

***Abstract* - Power and thermal reduction techniques are presented for hardware level, operating system level and application level. While the majority of control resides within the hardware level, it is shown that allowing all three to operate harmoniously and giving control to the level that has the best understanding of the current process will yield the best results. Topics such as DVFS, Clock gating, power metrics and custom component design are covered.**

I. Introduction

The thermal characteristics of microprocessors does not sound like a very riveting topic, but the ability to shape these characteristics and lessen the power/heat output is one of the most important challenges that is facing the computer industry today. Not only does this affect the chip manufacturer, but it also affects the OS Supplier and the computer builder. Within these three segments lie three different ways that the thermal output of a processor can be handled: by design, by control and by force.

The recent trend involved the latter, i.e. making the processor operate at a lower temperature using liquid cooling, Freon cooling, or an immense amount of fans. The purpose of this paper is to highlight the ways chips are being designed and manipulated as to decrease the need for these external cooling systems. The total power of the processor encompasses two parts; the dynamic power (power consumption because of workload) and static power (the innate power consumption that occurs with absolutely no workload). Both types of power affect the overall power, with the static power becoming more and more important as transistors decrease in size.

The power consumption of the computer has started to become more important for two reason. Firstly, because the costs associated with controlling the power have increased significantly [6, 10]. The first sector to feel the brunt force of this is the enterprise industry. Huge server rooms require expensive cooling units and as processors become smaller, and the static power significantly increases, the end customer will also inevitably face added fees to help control the power and thermal output of the processor. The second reason is that of static power. As shown in the power consumption and thermal loss section of this paper, the dynamic power will continue to play a role in the total power of a system, but as sizes decrease past .1 micron, static power drastically increases [18] and thus plays a more important role.

From the different techniques that chip architects can use, to upcoming and proposed methods for software applications to control and assist in dynamic power management, there is a multitude of different techniques, of varying degrees of effectiveness, that can be and will be utilized to lessen the power/heat output of a computer system.

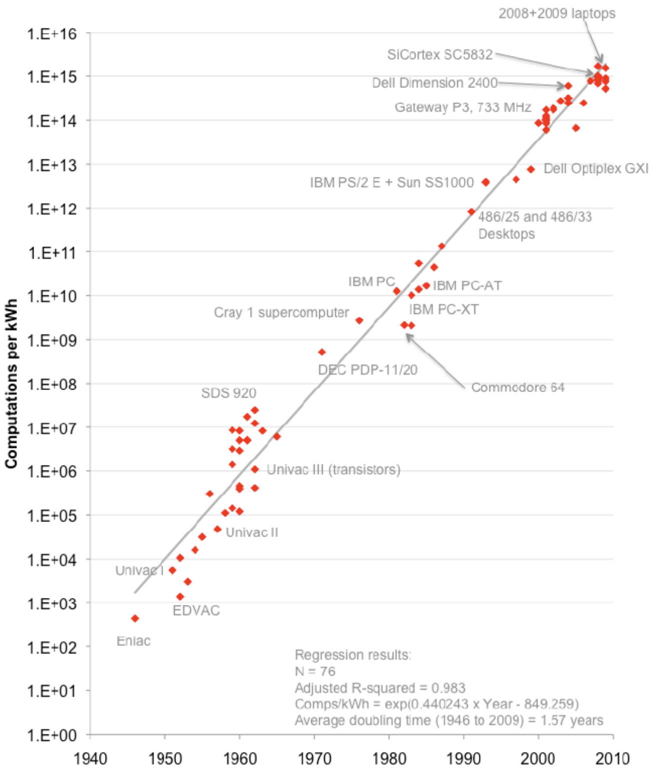


Figure 1 Moore's Power Law [5]

II. Processor Level

A. Determining Power Characteristics

Moore's law is the statistic that has driven microprocessors to the level they are at today. Moore stated that the number of transistors in a processor will double every 1.8 years [5]. Another lesser known statistic of Moore is his Power Law. Figure 1 shows the graph of this law. At first examination, this graph appears well placed and within the bounds of processing technology. The trend merely shows that as time has progressed and transistor counts increased, the operations per watt have also increased. This is very intuitive, but points out an important characteristic of processors. Operations per Watt or more specifically, millions of instructions per Watt (MIPS/Watt) is a very common metric used to convey the power characteristics of both single core processors and multi core processors. This metric,

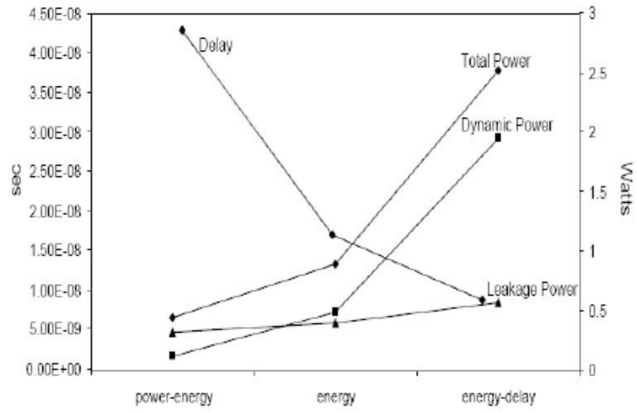


Figure 2 Comparison of different power metrics [19]

however, does not emphasize performance as strongly as the energy delay product (Power x Delay x Delay) or EDP, nor does it provide the ability to weigh a particular power/delay component. The EDP metric can also be used alongside the power delay product (Power x Delay) or PDP, and the power energy product (Power x Power x Delay) or PEP, to more efficiently architect a processor [19]. The ability of the processor designer to utilize these metrics effectively plays a large part in the inevitable power consumption of the processor as these all place either a priority on power or speed.

An interesting point is made in [19], which states that when designing for lower dynamic power, this can cause the static power to increase. This tradeoff is something that will affect designers more and more as the static power becomes more weighted in the overall power equation. [19] also compares EDP, PDP and PEP. Results show stark differences between these metrics. The EDP puts a high priority on delay, so the dynamic power and static power are very high. The PEP puts a priority on power, so the dynamic and static power are low, where as the delay is very high. The EP on the other hand is a compromise between the two as shown in in figure 2.

Beyond these metrics, there are two major ways that single core and multi core processors are designed from a power standpoint; power-limited design and hotspot limited design. Power limited design deals with the average power of the system. The system referred to in this context is the processor and all of its architectural components (cache, ALU, registers...). Hotspot limited deals with only the areas of the processor that consume the most power and thus have higher local temperatures. There is currently a division within the processing community as to which one of these is more applicable[12]. The power limited side believes that by decreasing the overall power consumption, you in return decrease the hotspots. On the other hand, the hotspot side believes that this average power reduction may not reduce the hotspots and in fact decreasing the average power does not linearly decrease the hotspots [12].

Figure 2 exemplifies these two different design techniques very well. The first row shows two different processors, the first being power-limited and the second being hotspot-limited. The second row, which is the row of

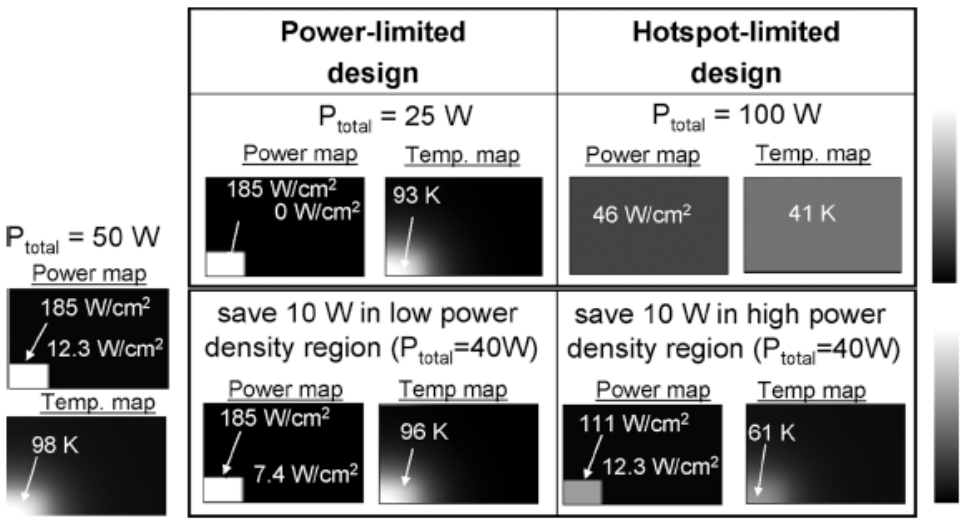


Figure 3 Comparison of Power-limited design and Hotspot-limited design [11]

most importance in our current topic, shows the same processor (original, non-power aware design on the left) with the power-limiting and hotspot limiting design applied. While both designs save 10 W in total power, it is truly the power and temperature maps that bring to light their differences. The power-limited design saves 10 watts by reducing the power in the low power density region of the chip from 12.3 W/cm2 down to 7.4 W/cm2, but the high power region, remains untouched. The hotspot-limited design on the other hand reduces power in the high power density region from 185 W/cm2 down to 111 W.cm2 while leaving the low power region the same. While this difference accounts for the same total power loss, the temperature maps show a completely different story. The power-limited design maintains the 96 K above room temperature region in the bottom left of the die, where as the hot-spot limited design has reduced this down to 61 K for a savings of 36%. This information, coupled with the forth coming sections, will help highlight why this high temperature region of the chip is of the utmost importance.

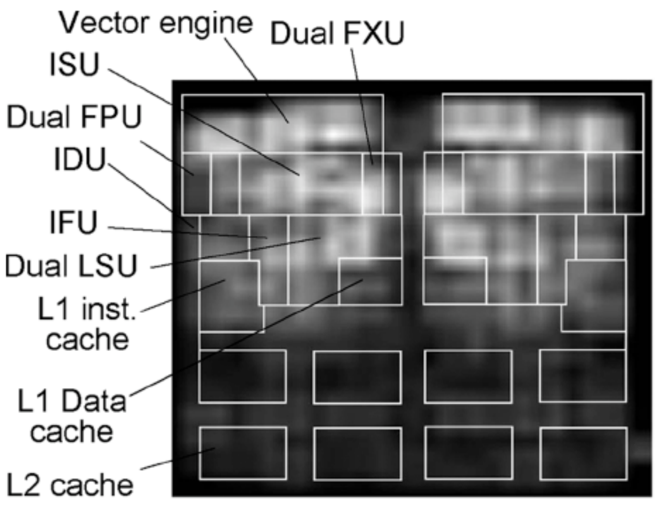


Figure 4 Thermal output of IBM Dual Power5 [11]

B. Power Consumption and Thermal Loss

Power consumption has been generalized into two different categories for single and multi-core processors: dynamic power and static power. Dynamic power deals with the power consumption of the processor and static power deals with the innate power consumption of a processor regardless of its workload. Dynamic power has long been the most significant with processor architects but as transistors get smaller, and thus chips do too, static power will increasingly play a more crucial role.

As dynamic power and static power comprise the power consumption of a system, deriving a formula for this system's characteristics is not only helpful, but quite essential. The equation below formulates this overall power consumption. The terms to the right of the addition sign constitute the dynamic power and the terms to the left constitute the static power. The dynamic power terms in this equation are as follows: A is the fraction of gates actively switching, C is the total capacitive load of all gates, V is the transistors supply voltage and f is the operating frequency.

$$P = ACV^2f + VI_{\text{leak}}$$

The dynamic power portion of this equation is relatively straightforward and is equal to the total power of all actively switching gates at speed f and supply voltage V. The static power component on the other hand, needs further clarification. The leakage current (I_{leak}) is comprised of the summation of the sub-threshold power leakage (I_{sub}) and of the gate-oxide power leakage (I_{ox}). The two equations below, as well as the latter, are presented in [18] and are very useful in conveying the static power consumption. The terms for the static power equations are as follows: W is the gate width, V_Θ is the thermal voltage (~25 mV at room temp), V_{th} is the threshold voltage, T_{ox} is the oxide thickness, V is the general transistor supply voltage and K, n and α are experimentally derived

$$I_{\text{sub}} = K_1 W e^{-V_{\text{th}}/nV_\theta}\left(1 - e^{-V/V_\theta}\right)$$

$$I_{\text{ox}} = K_2 W \left(\frac{V}{T_{\text{ox}}}\right)^2 e^{-\alpha T_{\text{ox}}/V}$$

The ability to minimize the sub-threshold power leakage as well as the gate-oxide power leakage will result in a much more substantial decrease in the total system power. Changing the value of the different components in these two equations will of course change the power consumption and

Table 1 Custom ALU Power Savings [17]

Application	design	qsort	aes	crc	des	RC4	rsa	dijkstra	stringsearch	sha	average
CPU clk time	non-custom	7.77	7.77	7.77	7.77	7.77	7.77	7.77	7.77	7.77	
(ns)	custom	7.77	7.77	7.77	7.77	7.77	7.77	7.77	7.77	7.77	
ALU power	non-custom.	1.0170	0.7826	1.2880	1.6590	1.0110	1.1160	1.1170	1.4010	1.3980	
(mW)	custom	0.5557	0.4418	0.6534	0.8367	0.5214	0.6132	0.6073	0.7403	0.7302	
pow.red(%)		45.4	43.5	49.3	49.6	48.4	45.1	45.6	47.2	47.8	46.9

performance of the overall system. We could increase V_{th} but that substantially lowers performance of the system, and we could turn off the supply voltage V, but that would result in a loss of state. Another option would be to increase the gate oxide thickness, $T_{ox,}$ but again that results in performance degradation and reliability issues [18]. Given the later, research is being done into high K dielectrics for the gate oxide. The dielectric is the gate oxide material that is placed in between the gate and the substrate in the transistor. By increasing the dielectric of the gate oxide, you substantially reduce the leakage current [18].

C. Processor Component Power

There are particular components of a single core or multi-core system that affect the temperature and power consumption much more significantly than others. The figure on the previous page, figure 4, shows the thermal landscape of a PowerPC970MP dual core processor running under a high power workload. From this picture, and from an understanding of the processor micro-architecture, we know that the ALU, Registers and Floating Point Unit represent very active parts of the processor. Minimizing the power consumption and heat dissipation from these parts will lessen the overall systems thermal and power characteristics as pointed out in the section above. While the floating point unit is shown in this figure, it will be covered in the next section.

1) Arithmetic Logic Unit

Completely redesigning any part of a system is more often than not frowned upon as it is monetarily and temporally consuming. A novel idea presented in [17] suggests that complete re-design is not needed, just an understanding of what aspects of the ALU are used more frequently. By determining the most frequently used operations of a common ALU (ADD, AND, OR, XOR), iterating through the different combinations of the placement of these components in the chain and placing the most commonly used operations closest to the output (OR), the ALU saved on average 49.6%. Table 1 on the previous page shows the results for different types of operations coded in VHDL and implemented in the design tool ASIPMesiter.

As demonstrated, this simple technique saved power across multiple different operations such as sorting (qsort), shortest path algorithm (dijkstra) and hashing (CRC). Implementation of this in new processors would be very straightforward as it simple in both logic and hardware requirements.

2) Registers

The registers on the processor are the one of the most active memory elements on the processor. During an operation it is very possible, and actually highly likely that the registers are not being full utilized [15]. A particular method for dealing with this would be to essentially trick the operation into believing that the registers requested exist, when in reality they do not. According to research and results from [15], by reducing the number of registers from 2^n to n and changing the memory address decoder from an n to 2^n to an n to n, you save on average 30.625% power.

This power savings is possible because of two crucial principles, one of which is challenging to implement in the real world. Firstly, the ability to reduce the number of registers merely comes down to logic. The simplest way to think of this is the modulo operator exp. if address 15 is requested, then address 15 Mod n is given. This cuts the number of registers required dramatically. The second principle, is the foresight of knowing what operations are to be used and how many registers are indeed needed. The research and conclusions from [15] had a certain degree of omniscience, in their knowledge of how many registers were needed and being able to physically remove them from their VHDL code. Two different types of updates were made, one for only the registers and a second time for the registers and decoder. By updating the decoder, the power savings was increased even more as it did not have to reference extra registers. In practice, this ability is of course not feasible. The dynamic changing of registers and the address decoder would require extra hardware and physically removing an item is not possible.

On the other hand, the ability to turn off registers is currently practiced [13] and could be coupled with this new research to produce a more power and thermal efficient design. By determining how many registers are to be used, and utilizing more advanced decode logic, the registers could be turned off and the decoder could consume less power. The trade-off would indeed have to be the overhead associated with determining which registers are turned off and dynamically changing the decoder logic.

D. Controlling the Power

There are many different techniques that have been used to lessen the power consumption and thus the thermal characteristics of a processor. A majority of processors use thresholds (triggers) based upon digital thermometers placed around the processor and the die. Once a threshold is passed, these methods are typically implemented, up to point when the processor is completely shutdown to avoid thermal runaway and potential harm. On the opposite side of this, are techniques used to reduce the power when the processor is idle. Both concepts actually overlap on their techniques to reduce the power and thermal output of the system.

1) Clock gating

Clock gating allows the ability to turn off different parts of the processor when they are not being used and is normally done by ANDing the clock with the gating control logic. For example, a set of registers could have their clock source "turned off" so that the gates do not switch, there is no charging or dis-charging and thus no wasted power. The AND gate will of course dissipate some power, but this is much less than the register or block of registers [20].

Relatively recent research into the measuring of the operating temperature of a processor under different workloads has determined that clock gating can save 14% in the hotspot temperature while simultaneously saving 18% in total power for an IBM Dual Power5 processor[11]. Clock gating has also been applied to the cache of the processor. This has been done to blocks of cache and with single registers as well, but it is the later of these two techniques that is used more frequently.

Clock gating is also used in the pipeline of the processor. The pipeline of the processor varies from manufacturer to manufacturer and even within the different chips that these companies produce. The same idea is present behind the pipeline in a superscalar machine. The main stages of a pipeline include Fetch, Decode, Execute, Memory and Writeback. Between each stage are latches to hold data from the previous stage until the next stage is ready.

The pipeline represents one of the most active and power consuming regions of the processor [20]. A current method to control the power consumption of this area is the pipeline balancing technique (PLB). There are two different versions of PLB, original and extended, and as extended has a higher power savings, we will mainly be dealing with it and it should be assumed that we are referring to PLB extended when using the abbreviation PLB. PLB utilizes predictive clock gating to decide whether a programs instruction level parallelism (ILP) will be less or more for a specific window of 256 cycles. ILP can be simply thought of as a measure of how many operations can be performed simultaneously for a given program. If PLB predicts that the next windows ILP is less than the current, then certain areas of the processor can

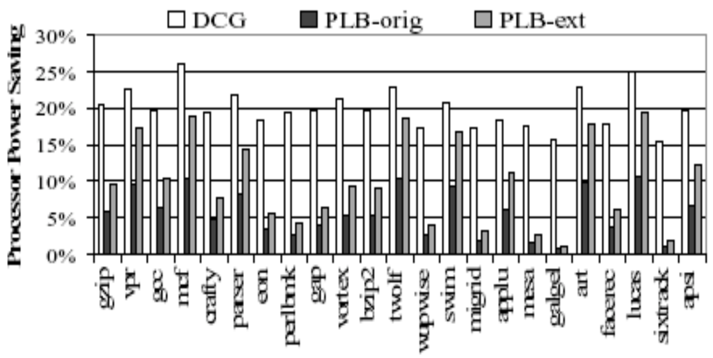


Figure 5 PIpeline Clock Gating Power Results [16]

be turned off. What is turned off is dependent upon the program but the areas of the processor that are gated in PLB are the Issue queue, pipeline-latches, d-cache wordline decoders, result bus and execution units. The clock gating of these components saves on average 11.0% for integer operations and 8.7% for floating point operations but imposes delay as it is predictive in nature, and a wrong prediction will result in a rollback of the gates that are turned off.

[20] proposes a more efficient solution using a deterministic approach. If you are able to determine ahead of time what the different stages of the pipeline are doing, you can then deterministically clock gate those components.

Deterministic Clock Gating (DCG) is the name given to this method. DCG utilizes the knowledge of the fetch and decode stages to effectively clock gate parts of the later stages such as the pipe-line latches, d-cache wordline decoders, results bus and execution units. This method is able to turn units off for 1-2 cycles at a time and able to determine which components to clock gate a few cycles ahead of time. Results from [20] show that DCG has much more power savings than PLB. DCG saves on average 20.9% for integer operations and 18.8% for floating point operations. While this is an almost doubling of the power savings over PLB, it is within the Floating Point Unit that the savings are truly seen. Figure 4 showed the most active units in the PowerPC970MP processor and one of the most power consuming units was the Floating Point unit, which is in line with [11, 13]. The floating point savings from DCG is 77.2% while the PLB savings is only 23% for floating point benchmarks [20]. The savings from DCG over PLB primarily come from the DCG ability to clock gate at 1-2 cycle duration where as the PLB approach uses a 256 cycle duration.

This triple the amount of power savings of DCG, coupled with its negligible delay provides a very good alternative to PLB and one that shows substantial savings for both the power and the thermal output of the system.

2) Voltage and Frequency Scaling

This is a very commonly used control, but results in a performance loss when it is implemented as voltage and frequency have a direct relationship. A decrease in the voltage (whether supply or threshold) will result in an at least similar, if not more drastic decrease in the operating frequency.

Table 2 Cycle loss for changes in DVFS [14]

		to frequency (MHz)				
		300	400	533	600	667
from frequency (MHz)	300		1101	1099	1086	1066
	400	1125		1095	1086	1066
	533	1117	1104		1073	1066
	600	1125	1101	1092		1066
	667	1117	1101	1088	1077	

Voltage and Frequency scaling will lessen the frequency and voltage of a processor once a certain power/thermal threshold has been reached. This requires an amount of delay as exemplified in Table 2 for a 700 MHz processor that supports voltage and frequency scaling. There are only certain levels that the voltage and frequency can be scaled to i.e. scaling is more of a step-function than linear or exponential.

Conclusions and research from [1] have taken a different approach to DVFS (dynamic voltage and frequency scaling). Instead of utilizing thresholds as triggers for DVFS, the clock frequency is instead started out at a much lower speed and, as demonstrated in the above sections, initially produces much less heat and power output. The users were then allowed to up the speed of the CPU as they saw fit which is how this technique has been deemed UDFS (User Driven Frequency Scaling). This incorporation of the user as a performance metric is relatively new, provides insight into what is actually necessary and poses a compelling questions; is just enough good enough? Results from these tests showed that on average, 22.1% of total system power was saved compared to the typical Windows XP DVFS. Even though this incorporates the OS and is not truly a hardware solution, it shows the necessity for all levels (hardware, OS, application) to work together and not rely solely on one entity.

3) Processor Sleep Cycles and ACPI

Advanced Configuration and Power Interface (ACPI) is by now a standard in systems and allows the Operating System to control the processor during its idle stages. ACPI is the technique used when you decide to put your computer in hibernate mode or suspend mode. The different modes of ACPI affect all the different components (Hard drives, memory, processor, input devices, power supply..). This is being covered here because of the different processor components that are affected during the different C stages of ACPI.

The first C stage is C0. In this state the processor is operating at a specific frequency with a workload. The next state is C1. In this state the processor is idle and some processors, such as the Intel Core Duo Processor, gates the core clocks for reduced power savings as described in the clock gating section of this paper. The C1 state can change to the C0 state almost instantaneously as the cache and chipset are still active. In the next state, C2, the processor needs approval from the chipset to access the bus, so processor, chipset and front-side bus can enter a lower power state. The next state is the C3 state in which the processor disables all internal phase locked-loops and does not need to refresh its internal cache, so the L1 cache is typically flushed as it is assumed that almost all of the data in the L1 cache is contained in the L2 cache. A new state found in the Intel Core Duo processor is the C4 state [9]. The C4 state primarily deals with the L2 cache. The L2 cache is the largest cache on the processor and therefore contains a large amount of data. Completely flushing the L2 cache would result in a substantial performance loss as this data would need to be fetched from higher memory such as the main memory or, worse yet, the hard drive. This C4 technique checks for interrupts and then flushes out the blocks of L2 cache that are not affected. This is repeated until the processor needs to moved to a higher state or until all of the applicable blocks have been flushed.

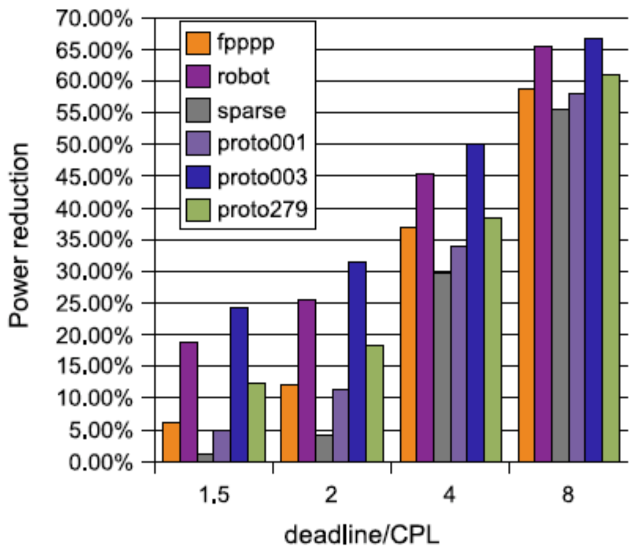
The ACPI technique uses a finite state machine (FSM) to change states back and forth as described above. When the processor must jump up a stage i.e. move from a lower state to a higher one (C4 to C3), the preceding actions are undone. This method also applies to multi-core processors, but adds an extra layer of complexity. In a multi-core system, there must be a check in the FSM as to whether or not the L2 cache is being utilized by the other cores as a majority of current processors have a shared L2 cache. Beyond this, the ACPI method is standard for single core and multi-core systems and allows a substantial amount of power savings.

All of these sections have described different ways that the hardware can within itself optimize the components as well as providing the higher levels more options for controlling the power. Together this allows for the further lowering of the power output and as a result, the thermal output of the processor.

III. Operating System Level

The amount of control and potential effect that the Operating System has on the power and thermal characteristics of a system is substantial. ACPI is of course utilized in most major operating systems, and was covered in the previous section. One of the main tasks that the OS handles is that of task scheduling. From the preceding section on hardware topics, we showed that dynamic power is currently the major consumer of power, but with static power gaining ground as time progresses. As the last section dealt with both of these topics (static and dynamic) as they are both hardware related, this section deals primarily with dynamic power but addresses static power as well.

Figure 6 LAMP Power Reduction Results [14]



A. Power and Task Scheduling

There has been a lot of research on the power consumption and proposed power savings for embedded systems. Within this lies the realm of real-time embedded systems. At first glance, the concepts and scheduling used in real time embedded systems seems not to apply to our current end user operating system (Windows XP, Mac OS X...) as they do not have strict deadlines, but this idea combined with ideas from the hardware section could yield some truly applicable and power/thermal efficient results.

Research and conclusions from [4, 16] provides us with a starting point on applying real time task scheduling to that of non-real time OS's. In a real time system, a task has a certain deadline that it must be completed by. It does not matter if the task finishes in unison with the deadline or finishes far before it. [4, 16] demonstrated, using similar concepts, that a concept similar to DVFS could be applied before the task starts based upon its deadline. The algorithm, Leakage Aware Multiple Processor Scheduling (LAMPS), developed and used in [16] determines the number of processors required to finish the algorithm. This differs from the Schedule and Stretch routine commonly used in realtime operating systems, in that Lamps defines at what speed the processors should run, how many should be used and uses the earliest deadline first technique. Both finish the task in unison with the deadline. Based upon these results, the frequency of the processors is reduced to the amount calculated and the task is scheduled. This calculation requires a small amount of overhead, but as shown in the figure below, results in a substantial savings over typical stretch and schedule routines.

As figure 6, on the previous page demonstrates, the savings for particular operations are very significant The abscissa above is of unit deadline/CPL or deadline/Critical Path Length and is how many times larger the deadline is than the critical path. The looser the deadline, the more the power savings.

By extrapolating the above results to non real time operating systems, we can see a potential for power savings. All that needs to be done is to determine the deadline for a given operation and apply the LAMPS algorithm to it. While simple sounding, this deadline of course is dependent upon multiple factors such as the hardware performance, OS interrupts and the user themselves which makes determining a deadline quite problematic. A solution to this can be taken from the application section of this paper, mainly making the deadline 50 ms (20 Hz) which is on the threshold of human vision [16].

IV. Application Level

The control of power consumption from an application level is a very new concept, but presents a lot of promise. In the recent paper [14], many of the ideas expressed in the Operating System section of this paper are relinquished to the Application and User level.

Chameleon [16] is the name of the application level power management scheme given to the following techniques. Firstly, the estimated demand of the processor is determined using previous CPU statistics and knowledge of the current operations demands. Secondly, the processor availability due to concurrent applications is determined, and

Table 3 Chameleon Application Level Power Savings. [14]

	Chameleon	LongRun	PAST	AVG_n	FULL
Mix M1	2.25W	3.27W	3.98W	4.42W	5.3W
Mix M2	2.47W	3.08W	3.79W	3.83W	5.3W
Mix M3	3.81W	5.27W	5.26W	5.27W	5.3W
Mix M4	3.71W	5.22W	5.23W	5.23W	5.3W

Mix M1 = encoded DVD playback and web browser
Mix M2 = encoded DVD playback and word processor
Mix M3 = encoded DVD playback and batch compilations (make and pnice)
Mix M4 = batch compilations and word processor

thirdly the processor speed setting, similarly to the schedule and stretch technique descried in the preceding section, is determined. Chameleon also uses the accepted concept that to a user, instantaneous response is anything that occurs under 50 ms [16]. Utilizing this, Chameleon will use the calculated speed for a task, and at defined interval's check that it will meet this 50 ms deadline. If the task will not, then the speed is increased to the next setting. As described in the hardware section, processors have preset levels for DVFS and a majority of systems have less than 6 different levels. This increase occurs until the task is either finished on time, or the deadline has passed in which case the CPU is increased to full speed.

The results from Chameleon are very appealing. Chameleon was able to save between 32% and 50% compared to no DVFS scheme and an average of 20% savings over the Longrun power savings as demonstrated in table 3. (Longrun is a hardware technique developed for Transmeta Crusoe processors that utilizes an implementation of DVFS and FULL is no DVFS scheme at all.)

While Chameleon is just one application domain power management system, it allows the ability of applications to rely upon it, rather than the OS to determine the proper DVFS settings. Chameleon utilizes information provided from the application as well keeping a historical summary of what settings were used for each application. Overall, Chameleon has shown that it can provide substantial savings to the power and thermal output of the processor as well as the system as a whole.

V. Conclusion

The sections in this paper represent different parts of the overall problem that is power consumption and thermal dissipation. At the beginning we analyzed different metrics used by chip designers to determine the power consumption and potential speed of a processor while highlighting the need to understand and utilize a metric that combines both dynamic power, static power and speed into one concise, fair metric.

The second section of the paper overviewed the different techniques currently used for lowering the power consumption of the processor. New techniques were presented that offer substantial savings over the current techniques. The optimizing of certain components, such as the L1 cache and the ALU were described and shown that their results are both substantial and easy to implement.

The third section focused on the Operating System and the main function of task scheduling. While the examples used were for real-time systems, it was shown that these are indeed applicable to non-real time operating systems and could indeed present a beneficial amount of power savings.

Finally, the possibilities for the Application to lower power consumption was investigated. While this section primarily discussed the Chameleon Power Management, it also highlighted the vast amount of savings that is possible from a better understanding of what the application needs in terms of speed and performance.

Combining all of these sections together allows a concise example of where the industry as a whole should be headed. By optimizing different processor components and deterministically deciding what units of the processor should be clock-gated, a DVFS enabled processor can provide a very substantial base for the OS and Application levels to help control the power consumption. As the application level of the system inevitably determines what operations are going to be run and hence the system load in a power and speed sense, it would seem only fitting that an application level power management solution would be the most efficient and energy aware.

Currently, the primary goal of reducing the power and thermal output of a system is to save money. In the near future, the increasing importance of the static power in the power equation will result in a much more basic need to control the output of the system so it does not spike out of control. It will take all levels of the computer i.e. hardware, operating system and application, to work together, and further technological improvements, such as high k di-electrics, to help solve the problem of decreasing the power and thermal output.

VI. References

The following papers were either directly referenced in the paper or were used for general background knowledge on a specific topic.

[1] Arindam Mallik, Bin Lin, Gokhan Memik, Peter Dinda, Robert P. Dick, "User-Driven Frequency Scaling," IEEE Computer Architecture Letters, vol. 5, no. 2, pp. , July-December, 2006.

[2] Aviad Cohen, Finkelstein Finkelstein, Avi Mendelson, Ronny Ronen, Dmitry Rudoy, "On Estimating Optimal Performance of CPU Dynamic Thermal Management," IEEE Computer Architecture Letters, vol. 2, no. 1, pp. , January-December, 2003.

[3] Sazeides, Y.; Kumar, R.; Tullsen, D.M.; Constantinou, T., "The Danger of Interval-Based Power Efficiency Metrics: When Worst Is Best," Computer Architecture Letters , vol.4, no.1, pp.1-1, January-December 2005

[4] Jejurikar, R.; Gupta, R., "Energy-aware task scheduling with task synchronization for embedded real-time systems," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.25, no.6, pp.1024-1037, June 2006

[5] Jonathan G. Koomey, Stephen Beard, Marla Sanchez, Henry Wong, “Assessing trends in the electrical efficiency of computation over time”, IEEE Annals of the History of Computing, Submitted August 5, 2009

[6] Jonathan G. Koomey, Christina Belady, Michael Patterson, Anthony Santos, Klaus-Dieter Lange, “Assessing trends over time in performance, costs and energy use for servers.” Release on web August 17, 2009.

[7] Eren Kursun, Chen-Yong Cher, "Temperature Variation Characterization and Thermal Management of Multicore Architectures," IEEE Micro, vol. 29, no. 1, pp. 116-126, January/February, 2009.

[8] David Brooks, Robert P. Dick, Russ Joseph, Li Shang, "Power, Thermal, and Reliability Modeling in Nanometer-Scale Microprocessors," IEEE Micro, vol. 27, no. 3, pp. 49-62, May/June, 2007.

[9] Alon Naveh, Efi Rotem, Avi Mendelson, Simcha Gochman, Rajshree Cabuksmar, Karthik Krishan, Arun Kumar, “Power and Thermal Management in the Intel Core Duo Processor”, May 15, 2006, retrieved October 2009

[10] Intel, “Intel is leading the way in in designing energy-efficine platforms”, retrieved October 2009,

[11] Hamann, H. F.; Weger, A.; Lacey, J. A.; Hu, Z.; Bose, P.; Cohen, E.; Wakil, J., "Hotspot-Limited Microprocessors: Direct Temperature and Power Distribution Measurements," Solid-State Circuits, IEEE Journal of , vol.42, no.1, pp.56-65, Jan. 2007

[12] Kevin Skadron, Pradip Bose, Kanad Ghose, Resit Sendag, Joshua J. Yi, Derek Chiou, "Low-Power Design and Temperature Management," IEEE Micro, vol. 27, no. 6, pp. 46-57, November/December, 2007.

[13] Brooks, D.; Martonosi, M., "Dynamic thermal management for high-performance microprocessors," High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on , vol., no., pp.171-182, 2001

[14] Xiaotao Liu; Shenoy, P.; Corner, M.D., "Chameleon: Application-Level Power Management," Mobile Computing, IEEE Transactions on , vol.7, no.8, pp.995-1010, Aug. 2008

[15] Yu Zhou, Hui Guo, Ji Gu, "Register File customization for low power embedded processors," Computer Science and Information Technology, International Conference on, pp. 92-96, 2009 2nd IEEE International Conference on Computer Science and Information Technology, 2009.

[16] P. de Langen, B. Juurlink, "Leakage-aware multiprocessor scheduling for low power," Parallel and Distributed Processing Symposium, International, pp. 60, Proceedings 20th IEEE International Parallel & Distributed Processing Symposium, 2006.

[17] Yu Zhou, Hui Guo, "Application Specific Low Power ALU Design," Embedded and Ubiquitous Computing, IEEE/IFIP International Conference on, vol. 1, pp. 214-220, 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2008.

[18] Nam Sung Kim, Todd Austin, David Blaauw, Trevor Mudge, Kriszti Flautner, Jie S. Hu, Mary Jane Irwin, Mahmut Kandemir, Vijaykrishnan Narayanan, "Leakage Current: Moore's Law Meets Static Power," Computer, vol. 36, no. 12, pp. 68-75, December, 2003.

[19] Sengupta, D.; Saleh, R., "Power-delay metrics revisited for 90 nm CMOS technology," Quality of Electronic Design, 2005. ISQED 2005. Sixth International Symposium on , vol., no., pp. 291-296, 21-23 March 2005

[20] Hai Li; Bhunia, S.; Chen, Y.; Vijaykumar, T.N.; Roy, K., "Deterministic clock gating for microprocessor power reduction," High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on , vol., no., pp. 113-122, 8-12 Feb. 2003