

The Elusive Metric for Low-Power Architecture Research

Hsien-Hsin S. Lee, Joshua B. Fryman, A. Utku Diril, Yuvraj S. Dhillon

Center for Experimental Research in Computer Systems

Georgia Institute of Technology

Atlanta, GA 30332

{leehs, fryman, utku, yuvrajsd}@ece.gatech.edu

Abstract

The Energy-Delay product or ED product was proposed as a metric to gauge design effectiveness. This metric is widely used in the area of low-power architecture research, however it is also often improperly used when reporting a new architecture design that addresses energy-performance effectiveness. In this paper, we discuss two common fallacies from the literature: (1) the way the ED product is calculated, and (2) issues of introducing additional hardware structures to reduce dynamic switching activities. When using the ED product without meticulous consideration, a seemingly energy-efficient design could turn out to be a more energy-consuming one.

1. INTRODUCTION

The Energy-Delay product (ED product), advocated by Gonzalez and Horowitz [9], was introduced as a metric capable of coupling both the energy consumption and performance of alternate architecture design choices. In this metric, the performance is measured in terms of the delay until execution is complete, whether completion constitutes the end of the program or simply meeting a constraint as in video coder. The ED product has subsequently been applied widely in the literature, analyzing everything from network protocol selection to TLB arrangements [11, 15, 30].

The Energy Delay product was also presented at a time when the common process technology was 0.25- μm or larger. Leakage current was generally an order of magnitude less – or better – than dynamic switching currents in architecture designs [6]. This is no longer the case, however, and leakage current must be accounted for in all contemporary designs that use sub-0.13- μm layouts, referred to as Deep Sub-Micron (DSM) processes. Recent research has similarly ignored or downplayed the effects of leakage current in order to study new architectural techniques for reducing dynamic energy alone, potentially leading to erroneous conclusions when applying the same technique for processors fabricated using DSM processes.

2. COMMON FALLACIES

Batteries, when used in a platform, do not only supply the energy needed for the CPU but also provide the power for the rest of the system, e.g. a notebook PC hard-disk drive and TFT display. When architects evaluate performance and energy using cycle-accurate architectural simulators with built-in power/energy analyzers [5, 8, 36], the perfor-

mance numbers in Instructions Per Cycle (IPC) typically consider the entire system effects including all the cache hierarchies and the DRAM memory. The energy savings are then reported as a proportion of *only* the particular functional block under examination, e.g. L1 cache or the branch target buffer. The ED product is generated accordingly by the researchers. This calculated result can be misleading.

Most proposals fail to estimate the additional transistors (and buses) required to implement the hardware assumed. Additional hardware brings additional leakage power and potentially more switching power. Many architectural level low-power techniques propose adding new hardware schemes to exploit power reduction opportunities by filtering, diverting, or monitoring events over a baseline design in order to attain the lower energy profiles while tolerating some, if not zero, performance degradation. As DSM processes become more prevalent, each of these added power components must be addressed in a meticulous manner to avoid incorrect conclusions. Ignoring the growing proportion of leakage energy, incorrectly evaluating the total system energy expenditure, and failing to address the underlying consumer of energy within a module can reduce or even invalidate results.

In this paper, we address the following potential fallacies encountered in low-power architecture research:

1. Which energy (E) are you looking at?
2. Additional hardware can be harmful.

We now explore exactly how these pitfalls can lead to problematic results unless proper care is exercised during analysis.

3. ANALYTICAL ANALYSIS OF ENERGY VS. PERFORMANCE

Calculating an ED product begs a question of which energy is being evaluated. When examining one component of a system, such as a Flash drive, careful design choices can result in respectable energy savings. Yet if the Flash drive consumes only 10% of the total system power, reducing the energy of the drive by 5% overall amounts to a system reduction of only 0.5%. Equivalent savings could be attained by optimizing the access patterns to disk or

other techniques that require no hardware redesign. Including the delay impact of alternate designs exacerbates this situation.

Similarly, optimizations made to subcomponents of a CPU must be balanced against the larger picture of energy consumption within a complete system. To understand whether optimizations make sense at the large-scale vision, the relationship between the consumption of CPU energy and the rest of the system is explored first.

3.1 Which Energy (E) are you looking at?

In this section, we create a very simple equation dubbed the Complete Energy-Delay Product (CEDP). An energy-delay effective design should yield a CEDP value less than 1.0, given that 1.0 represents the baseline or reference system.

Equation (1) describes a simple, necessary condition to be satisfied for attaining a lower Energy-Delay product, i.e. an energy efficient design. In this equation, the original energy dissipated by a particular functional unit u with respect to the CPU, before applying any energy optimization technique, is denoted by the ratio $R_{CPU}(u)$; the energy dissipated by the CPU with respect to the overall platform under the same energy source is denoted by $R_{sys}(u)$; and $R_{saved}(u)$ represents the ratio of energy saved for u by applying a particular low-power microarchitectural technique. The term $\frac{\Delta D}{D}$ represents the additional execution delay, if any, of the overall performance induced by the new design.

$$[1 - R_{sys}(CPU) \cdot R_{CPU}(u) \cdot R_{saved}(u)] \cdot \left(1 + \frac{\Delta D}{D}\right) \leq 1.0 \quad (1)$$

First, assume that our processor uses of a dedicated power source, an exclusive battery reserved for the CPU itself. In other words, no other competing devices share the same power source and hence $R_{sys}(CPU) = 1.0$, leading to the simplification of Equation (2), from which we solve for $\frac{\Delta D}{D}$ to obtain Equation (3).

$$[1 - R_{CPU}(u) \cdot R_{saved}(u)] \cdot \left(1 + \frac{\Delta D}{D}\right) \leq 1.0 \quad (2)$$

$$\frac{\Delta D}{D} \leq \frac{R_{CPU}(u) \cdot R_{saved}(u)}{1 - R_{CPU}(u) \cdot R_{saved}(u)} \quad (3)$$

Using Equation (3), we then plot the curves shown in Figure 1 with a spectrum of $R_{CPU}(u)$ varying from 5% to 99%.

Each curve draws the boundary condition of an effective system, assuming the energy savings of a particular functional unit u is equal to $R_{saved}(u)$. In other words, designs with new $\frac{\Delta D}{D}$ values falling in the *area* above each curve in Figure 1 represents ineffective designs for a given ratio of energy saving — $R_{saved}(u)$ in u . The X-axis is plotted on $R_{CPU}(u)$ given that $R_{sys}(CPU)$ is 1.0. For instance, a value of 0.6 means 60% of the overall original energy consumption is contributed by u . The Y-axis, plotted on a log

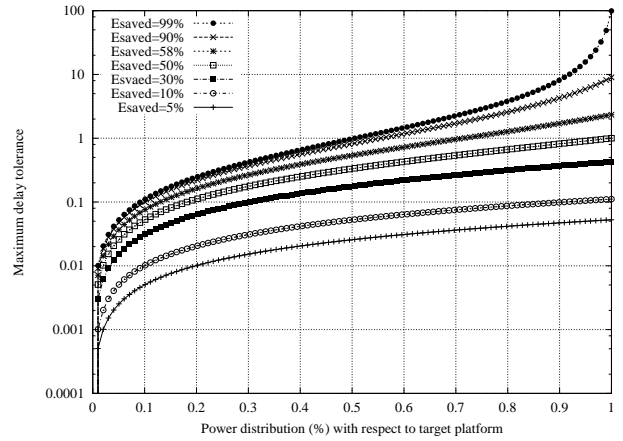


Figure 1: Delay limit vs. Power weighting

Pentium Pro		StrongARM SA-110	
Instruction Fetching	22.2%	ICACHE	27%
Integer Execution Unit	14.3%	Instruction Unit (IBOX)	18%
ReOrder Buffer	11.1%	DCACHE	16%
Data Cache Unit	11.1%	CLOCK	10%
Reservation Station	7.9%	Integer MMU (IMMU)	9%
Floating-Point Unit	7.9%	Integer Exec. Unit (EBOX)	8%
Global Clock	7.9%	Data MMU (DMMU)	8%
Memory Order Buffer	6.3%	Write Buffer (WB)	2%
Register Alias Table	6.3%	Bus Interface Unit (BIU)	2%
Branch Target Buffer	4.7%	Phased Locked Loop (PLL)	< 1%

Table 1: Power distribution for commercial processors

scale, indicates the limit of the extra delay ($\frac{\Delta D}{D}$) allowed in the overall system delay (CPU in this example) for an effective system design. Note that Figure 1 is also applicable when $R_{sys}(CPU)$ is less than 1.0, i.e. when sharing the power source with other devices, by using the value of $R_{sys}(CPU) \cdot R_{CPU}(u)$ on the X-axis.

We examine a simple example to illustrate the applicability of Figure 1: Assume a novel microarchitectural technique is proposed which can reduce the energy consumption by 50% for the branch target buffer (BTB) and its associated branch predictor in a processor. Looking at the curve with $R_{saved}=50\%$, if the original BTB and branch predictor consume 10% of overall platform energy (i.e. the X-axis value is 0.1), then the new design can only afford to lose around 3% performance in delay (i.e. Y-axis value is 0.03) in order to make the new system effective in terms of energy and performance using ED product as the metric. Moreover, if the processor shares the same power source with other devices such as DRAM memory and/or TFT display, the affordable performance loss will be dramatically subdued. For example, when the CPU only consumes 25% power of the battery, i.e. $X=0.025$, from Figure 1, the performance cannot be compromised for more than 1% for an energy efficient design by the definition of the ED product.

In order to get more understanding about the interaction between energy and performance, data from two commercial processors are used for demonstration. Table 1 recapitulates the power distribution on two popular processors used in high performance and embedded/handheld systems, the Pentium Pro and StrongARM SA-100. The data shown are copied from the relevant literature [19, 20]. Similar to Figure 1, we use Equation (3) for each functional

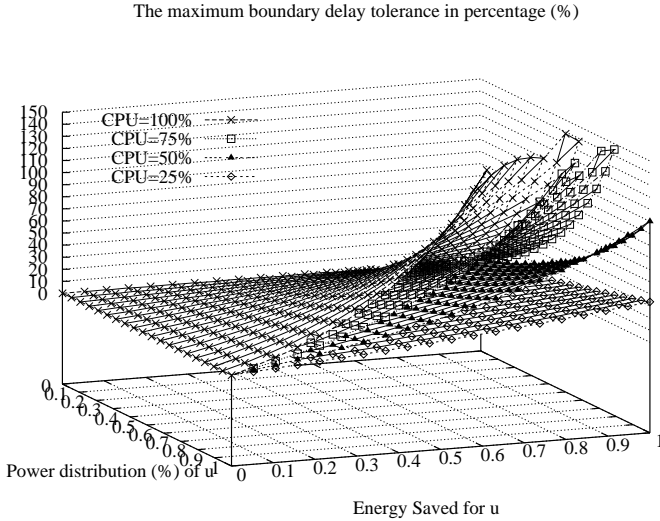


Figure 3: Maximum delay tolerance (Z-axis) in % as a function of energy saved in u (X-axis) vs. power distribution of u (Y-axis)

unit in Table 1 to construct Figure 2. Again, this figure assumes that the CPU has a dedicated battery source.

The X-axis represents the percentage of energy saved by applying some energy optimization technique to a particular functional unit, while the Y-axis represents the allowable performance degradation for keeping the optimized design energy efficient. Using the Pentium Pro processor as an example, if one proposes a technique that can reduce the energy of the ReOrder Buffer by 30%, the new Pentium Pro processor can only afford to lose 3% performance over baseline to be energy efficient. As aforementioned, in a realistic situation when the CPU shares the battery with other peripherals, the allowable loss would be much less than 3%.

To see how the total system energy plays a role in determining energy efficient designs, we now stop using the assumption that the CPU has a dedicated power supply. For the CPU consuming overall platform energy of 25%, 50%, 75% and 100% each, Figure 3 illustrates a general trend of how the energy efficient design space changes as two parameters vary — (1) percentage of energy saved $R_{saved}(u)$ for u , and (2) power distribution $R_{sys}(CPU) \cdot R_{CPU}(u)$ of u with respect to the entire target system. This figure depicts the relationship between the original energy weight of the functional unit u (shown on the Y-axis) being optimized (shown on X-axis) with the total system delay tolerance. The Z-axis plotted in percentage of additional performance degradation represents the maximum delay a modified design, u , tolerates for an energy efficient design. The X-axis plots the energy saved for u from 0%, i.e. no energy optimization, to 100% wherein u is completely eliminated; the Y-axis plots the power distribution of u with respect to the target system prior to optimization. Mostly, the tolerable performance delays are less than 10% in practical cases.

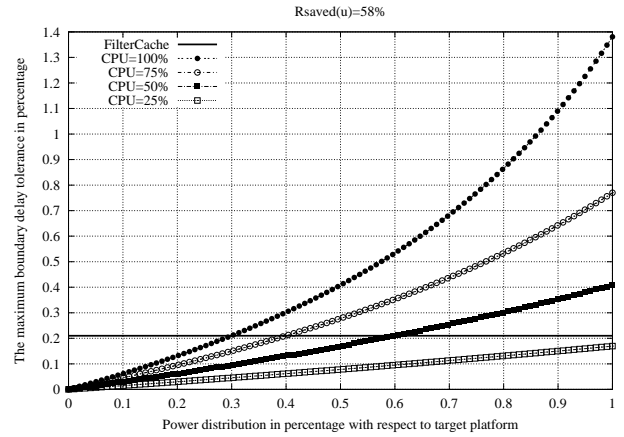


Figure 4: The percent of energy required to be dissipated by the target module, u , over the entire platform.

Examining a more realistic scenario will further illustrate the concept, we use data from the filter cache study reported by Kin et al [13]. In this work, they showed a 58% L1 cache energy reduction by trading off a 21% IPC degradation. Figure 4 illustrates a snapshot of Figure 3 when $R_{saved}(u)$, i.e. energy saved for u , is fixed at 58%. Tracing from the Y-axis by a 21% line (the bold horizontal line in Figure 4), we see four possible outcomes. When the CPU represents 100% of the system power, then the filter cache is an efficient design if the total L1 cache power exceeds 30% of the entire CPU. Similarly, when the CPU consumes 75% of the total power, the L1 caches must consume at least 40% of overall power; for the CPU using 50%, the L1 caches must consume 60% overall; and when the CPU consumes only 25% of the total power, the filter cache cannot be an energy efficient design at all¹.

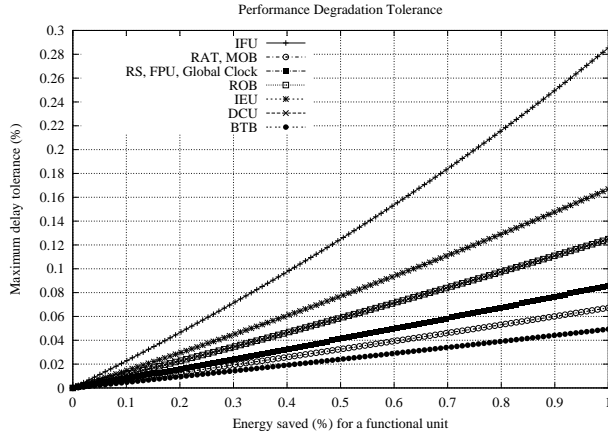
The filter cache study was targeted at the StrongARM SA-110 processor. As can be seen in Table 1, the SA-110 dissipates a combined 43% of CPU energy in the L1 data and instruction caches. When this work is evaluated as a stand-alone system, where the CPU has a dedicated power supply, it is an energy efficient choice. However, as we have demonstrated, if the SA-110 CPU in a system design consumes much less than 75% of total system power then this is not an effective solution at all.

We observe that it is insufficient to examine energy-delay issues from the narrow perspective of just the component being optimized. Some presentation of the total system impact must also be made in order to fully evaluate the design trade-offs.

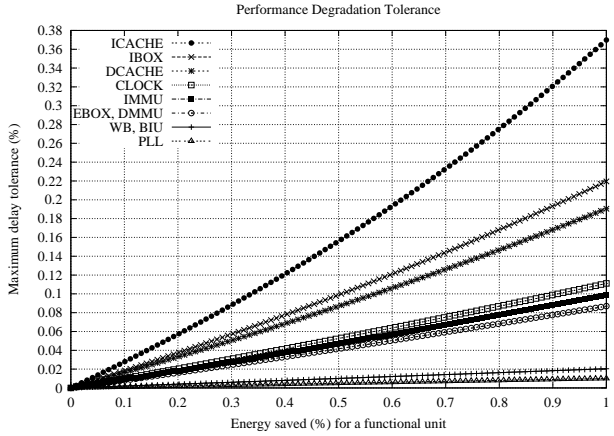
3.2 Additional Hardware can be Harmful

Another pitfall in the R_{saved} term from Equation (1) is the historical insignificance of the leakage energy estimates. In spite a slew of literatures have been raising attention and proposing new techniques for reducing leakage energy, most of them isolated leakage and dynamic switching

¹The static energy consumed by introducing an additional component such as a filter cache is revisited in Section 3.2.2.



(a) Pentium Pro



(b) StrongARM-110

Figure 2: Maximum delay tolerance limits for two commercial processors.

power to two independent subjects in their studies. With the advance of DSM techniques, leakage or static energy dissipation can no longer be ignored or considered independently from dynamic power consumption, especially when they also consume power from the same supply – batteries. The problem is that I_{DDQ} grows exponentially with the threshold voltage [6]. Before examining how the leakage current is impacting contemporary work, we re-examine in a new manner the basic power equations that govern CMOS designs.

3.2.1 Switching Activity

The various factors contributing to the power problem can be represented by a set of simple equations. We distinguish between the *short-circuit*, *dynamic*, and *leakage* power as follows: P_{SC} represents the transient current flow during transition change from the power supply to ground; P_{DYN} is the power consumed by the charging and discharging of capacitive elements during switching; and P_L represents the power consumed when the inputs are stable, also called the leakage power. We use similar notations and conventions as contemporary texts [22, 32, 34].

$$P_L = I_{DDQ} \cdot V_{DD} \quad (4)$$

$$P_{SC} = \overline{I_{SC}} \cdot V_{DD} \quad (5)$$

$$P_{DYN} = a \cdot f \cdot C \cdot V_{DD}^2 \quad (6)$$

$$P = P_L + P_{SC} + P_{DYN} \quad (7)$$

Classically, the power dissipation from P_{DYN} has dominated the total power consumption. P_L and P_{SC} could be discarded. P_L has been historically on the order of micro-Watts, while the total power was on the order of Watts. P_{SC} has been at most 5-10% of P_{DYN} . In low-power architecture research, a large number of work suggests novel designs that reduce P_{DYN} .

In order to evaluate the energy impact when a new design is proposed, some estimate of any change in total transistor count is necessary, ΔT . A second estimate of the delay

(or speedup) caused by the new model on the clock speed, ΔF , is also required. The final impact is the application slowdown, ΔS . If only Equation (6) is considered, then to achieve a “winning” efficiency the activity coefficient, a , must be reduced by some quantity.

Analytically extracting a from the reference model vs. the new model, we solve for a_{new} in Equation (8). The safe assumption is that the *average* short-circuit current drawn, $\overline{I_{SC}}$, will scale linearly so long as ΔT is less than a few % of the total transistor count, T . Sufficient examination of this results in an insight that so long as the *ratio* shown in Equation (9) is satisfied, a more power-efficient solution is present. (This uses the approximation that $C = C_{gateavg} \cdot T$.)

$$a_{new} \leq \frac{P_{ref} - P_{L_{new}} - P_{SC_{new}}}{f_{new} \cdot C_{new} \cdot V_{DD}^2} \quad (8)$$

$$\frac{a_{ref}}{a_{new}} \geq \left(1 + \frac{\Delta T}{T} + \frac{\Delta F}{F} + \frac{\Delta F \cdot \Delta T}{F \cdot T} \right) \quad (9)$$

For example, in a 100M transistor design base, a new module is added at an extra 1% to the total transistor count. This new module also affects the critical path such that a 5% performance delay is incurred. So long as the ratio $a_{ref} : a_{new}$ exceeds 1.06 ($1 + .05 + .01 + .05 \cdot .01$), the new design consumes less energy than the original design. This pattern is true irrespective of ΔT , ΔF , and all other parameters – except I_{DDQ} . If $P_L \leq 0.02 \cdot P$, then these patterns are true to an accuracy of 0.2%, and leakage current can be ignored. In other words, if the target manufacturing process $\geq 0.25\text{-}\mu\text{m}$, then these equations hold true. If the conditions (1) $\Delta F \ll F$ and (2) $\Delta T \ll T$ are also true, then Equation (9) can be reduced to Equation (10).

$$\frac{a_{ref}}{a_{new}} \geq \left(1 + \frac{\Delta T}{T} + \frac{\Delta F}{F} \right) \quad (10)$$

This result is at first obvious, yet also deceptively strong.

The entire viability of any design from a purely energy-efficiency perspective has been reduced to three variables: (1) the new switching probability, a_{new} ; (2) the change in total transistors, ΔT ; and (3) the change in clock speed, ΔF . Given any two of these values, the maximum tolerance for the third is easily calculated.

While this insight is useful, it fails to account for the delay incurred to system users. The results from Equation (9) and Equation (10) do not include any factors to consider the slower execution of programs. (This is not necessarily the same as the clock frequency change, ΔF .) To also include the delay impact, again the ED product is used. By using the reference application delay, D , we then solve for a_{new} from the revised baseline equation of $P_{ref} \cdot D^2 = P_{new} \cdot (D + \Delta D)^2$. Extracting a revised ratio which incorporates ΔD , we obtain a new way to view the ED product.

$$\frac{a_{ref}}{a_{new}} \geq \left(1 + \frac{\Delta T}{T} + \frac{\Delta F}{F}\right) \cdot \left(1 + \frac{\Delta D}{D}\right)^2 \quad (11)$$

The ratio from Equation (11) works as excellent approximations based on three underlying principles: (1) $\Delta T \ll T$, (2) $\Delta F \ll F$, and (3) $P_L \ll P$. While the first two principles generally hold true, the third principle is dependent upon process technology. As earlier work has shown [6], when the target process is $\geq 0.25\mu m$, I_{DDQ} is very small and can generally be ignored. When the process scales from 250 nm to 70 nm, however, P_L is projected equal or surpass P_{DYN} at 70-nm [2]. When this occurs, the errors introduced from the ratios above exceed 15%.

3.2.2 Leakage Impact

The total leakage current has many contributors in the physical design process. The total leakage contributors lie in the physical mediums used in construction of devices, given that the real world fails to match ideal circuit models. The major contributor due to process shrinking is the sub-threshold current. This is the current that flows even though the gate voltage is less than the threshold voltage (e.g., $V_{gs} < V_t$).

Architecture researchers have proposed implementing additional structures to reduce energy consumption, such as Pyreddy and Tyson’s dual speed pipeline [24], Vahid and Gordon-Ross’s loop table [33], or Iyer and Marculescu’s run-time profiling hardware [12], to name a few. In the following discussion, we use the dual speed pipeline as an example design to illustrate the looming issues of these choices in a DSM era. This example will highlight the impact of increasing leakage currents, as well as provide implications in dynamic power consumed by additional hardware modules.

Assume a processor with a high-speed datapath. One way to reduce power consumption suggested in [24] is to introduce a slower but less power hungry datapath in addition to its fast counterpart. Similar to the power reduction idea by exploiting execution slack, instructions are scheduled to different speed datapaths, exploited either statically by compilers or dynamically by the hardware, based on the criticality of each instruction. If the result of the instruc-

Datapath Speed	Higher ($D \uparrow$)	Lower ($D \downarrow$)
Dynamic Energy/Instruction (J)	DE_{\uparrow}	DE_{\downarrow}
Leakage Power (Watt)	LP_{\uparrow}	LP_{\downarrow}

Table 2: Terms used for a dual speed pipeline.

tion is needed immediately as part of the critical path of the program control flow graph, then this instruction is scheduled and dispatched to the fast datapath. The slow datapath will consume less energy than the fast datapath. This may be achieved by using lower supply voltage and/or higher threshold voltage while designing the slower datapath. Similar techniques were investigated recently in [23, 28]. Table 2 illustrates the notation and symbols used in this example.

Table 3 shows the dynamic and static energy characteristics of two different datapath designs, *Single* pipeline and *Dual* pipeline. In the dual speed pipeline, a slow datapath, represented by $D \downarrow$, is introduced. For simplicity, we assume that all the instructions have the same execution latency and the total execution time is proportional to the number of instructions executed through the fast datapath ($D \uparrow$), namely, instructions on the critical path. For each datapath, the same dynamic energy is dissipated for each instruction in that datapath. Moreover, to get the best case energy savings for the dual datapath design, we assume negligible energy consumption of the additional hardware to coordinate the two datapaths.

Given the total number of instructions and the number of instructions dispatched to $D \downarrow$ are N and x , respectively, then the execution time² T_d of the program can be written as $T_s \cdot \frac{N-x}{N}$ where T_s represents the execution time on the *Single* datapath machine. The total energy (TE) consumed by *Single* and *Dual* machines are represented by TE_s and TE_d in Equation (12) and Equation (13).

$$TE_s = LP_{\uparrow} \cdot T_s + DE_{\uparrow} \cdot N \quad (12)$$

$$TE_d = (LP_{\uparrow} + LP_{\downarrow}) \cdot T_s \cdot \frac{N-x}{N} + DE_{\uparrow} \cdot (N-x) + DE_{\downarrow} \cdot x \quad (13)$$

For the *Dual* machine to be more energy efficient, TE_d must be equal to or smaller than TE_s , leading to Equation (14). Assuming the ratios of the slow datapath versus the fast one is r for both of the dynamic energy and leakage power as shown in Equation (15), and by replacing $LP_{\uparrow} \cdot T_s$ with LE_s , we rewrite the new inequality as Equation (16). In reality, the ratios of the dynamic and leakage power can be different due to different V_{dd} and V_t , but we assume they are the same for simplicity.

²Note that the execution time could be slower than $T_s \cdot \frac{N-x}{N}$, i.e. closer to 1, if the non-critical instructions in the *Single* machine model cannot be completely hidden.

Machine model	Single ($D \uparrow$)	Dual ($D \uparrow + D \downarrow$)
Inst Executed	N on $D \uparrow$	x on $D \downarrow$, $(N-x)$ on $D \uparrow$
Execution Time	T_s	$T_d = \frac{(N-x)}{N} \cdot T_s$
Leakage E (LE)	$LP_{\uparrow} \cdot T_s$	$(LP_{\uparrow} + LP_{\downarrow}) \cdot T_d$
Dynamic E (DE)	$DE_{\uparrow} \cdot N$	$DE_{\uparrow} \cdot (N-x) + DE_{\downarrow} \cdot x$
Total Energy	$LP_{\uparrow} \cdot T_s + DE_{\uparrow} \cdot N$	$(LP_{\uparrow} + LP_{\downarrow}) \cdot \frac{N-x}{N} \cdot T_s + DE_{\uparrow} \cdot (N-x) + DE_{\downarrow} \cdot x$

Table 3: Equations used for a dual speed pipeline.

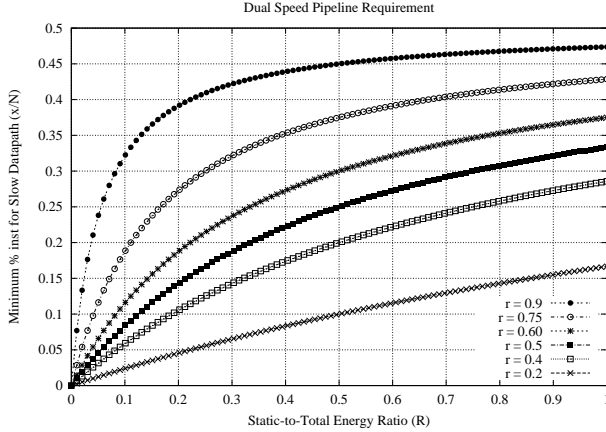


Figure 5: Minimum percentage of instructions to issue to the slow datapath in order to gain energy saving for different initial static to total energy ratios.

$$x \geq \frac{LP_{\downarrow} \cdot T_s}{\frac{LP_{\uparrow} \cdot T_s}{N} + \frac{LP_{\downarrow} \cdot T_s}{N} + DE_{\uparrow} - DE_{\downarrow}} \quad (14)$$

$$r = \frac{DE_{\downarrow}}{DE_{\uparrow}} = \frac{LP_{\downarrow}}{LP_{\uparrow}} \quad (15)$$

$$x \geq \frac{r \cdot LE_s}{\frac{LE_s}{N} + \frac{r \cdot LE_s}{N} + DE_{\uparrow} - DE_{\downarrow}} \quad (16)$$

Since $N \cdot DE_{\downarrow}$ is the total dynamic energy consumed by a program for the *Single* machine, hence the ratio of static-to-total energy can be represented as the R in Equation (17). The final inequality can be expressed as Equation (18). On the left-hand side, it indicates that the minimal ratio of the number of instructions that are needed for the slow datapath in order to make the *Dual* machine more energy efficient than the *Single* one.

$$R = \frac{LE_s}{LE_s + N \cdot DE_{\uparrow}} \quad (17)$$

$$\frac{x}{N} \geq \frac{r \cdot R}{(1+r)R + (1-r)(1-R)} \quad (18)$$

Using Equation (18), Figure 5 plots the minimum percentage of instructions needed to be dispatched to the slow datapath as a function of static-to-total energy ratio in order to save overall energy. As shown in prior projections from the industry [18, 31], the ratio of static energy consumption to the total energy increases dramatically in every technology generation. Thus, the static energy penalty due to the additional hardware becomes more significant.

The active static-to-total energy ratio is in the ballpark of 20% for current process generation. When migrating down to 70nm, the static leakage energy becomes dominant with a ratio more than 50%.

Clearly, the minimum percentage of instructions increases as the ratio of static energy to the total energy increases. For example, for a slow datapath executed at 75% of the fast datapath energy ($r = 0.75$), with a 50% static-to-total energy ratio ($R = 0.5$), we need 37.5% or more of the total instructions to be dispatched to the slow datapath for being energy efficient. In other words, compilers or a specialized hardware have to identify at least 37.5% instructions off the critical path for a given application.

Figure 5 also illustrates a sweep of curves plotted for a slow datapath consuming from 20% to 90% energy of what the fast datapath consumes. Given that lower energy consumption is achieved by lowering the datapath speed, if the speed of the slow datapath is considerably less than that of the fast datapath for energy reduction reason, the overall performance may be further degraded by non-critical instructions becoming critical, leading to an energy inefficient design.

Using Equation (19) derived from prior equations, we plot Figure 6 that shows the energy savings for a variety of different static-to-total energy ratios, R , in which the slow datapath is assumed to consume (a) 75% energy of its fast datapath counterpart ($r = 0.75$) and (b) half the energy of its fast datapath counterpart ($r = 0.5$). For $r = 0.75$, even with a static-to-total energy ratio less than 20%, the energy saving is less than 10% until the machine could find 40% of the instructions to be off the critical path and dispatched to the slower datapath, which could be rather difficult and degrade performance. For $r = 0.5$, although the instruction percentage is reduced to 30%, the potential performance degradation due to the even lower frequency datapath is likely to become a hindrance for it to be feasible. Furthermore, this energy saving is with respect to the datapath only, and according to our discussion in Section 3.1, the overall energy efficiency probably cannot tolerate any tiny performance loss. It also complicates the design complexity and verification efforts.

$$\frac{TE_s - TE_d}{TE_s} = (2Rr - r + 1) \left(\frac{x}{N} \right) - rR \quad (19)$$

Similar issues exist in other techniques and one can apply similar analytical model to evaluate the energy efficiency when the leakage energy consumption of the additional hardware could become an overkill. For example, loop tables described in [33] also obtain energy savings at the expense of additional transistors. An average of 34%

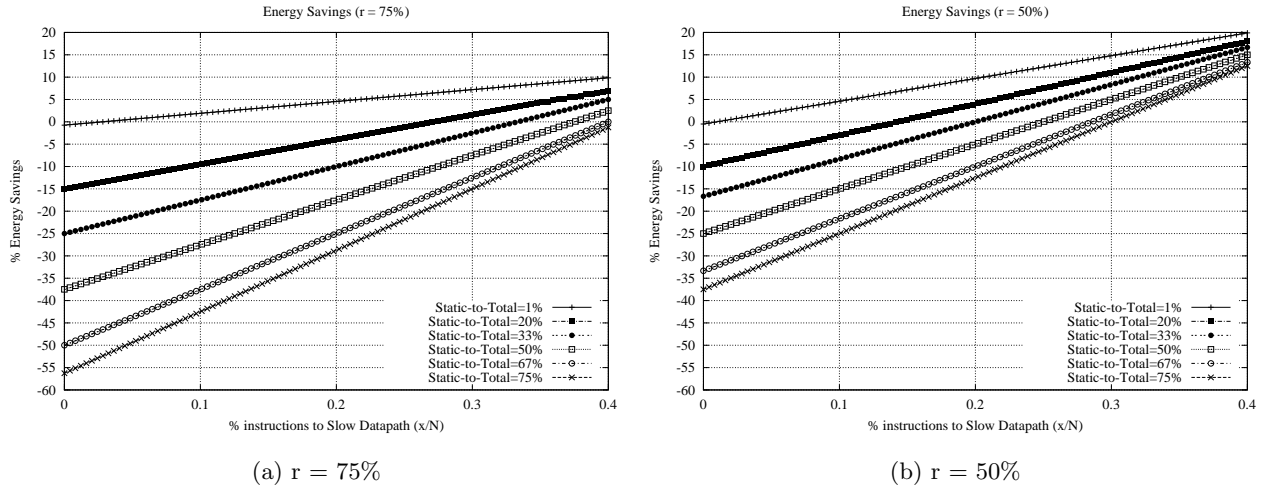


Figure 6: Energy savings versus % instructions issued to the slow datapath.

dynamic energy savings was shown for commercial microprocessors by nearly tripling the transistor count. When the static energy increase is taken into account, the dynamic energy reduction will be equal to the static energy increase for a technology generation where static energy to total energy ratio is 25%. The problem of leakage power is exacerbated when the entire system is put into deep sleep mode, in which only leakage energy is consumed.

Note that there exists another class of research in which additional hardware mechanisms are employed for dynamic thermal detection and management [4, 10, 14, 25, 27]. These temperature-aware architecture designs attempt to avoid overheating issues by dynamically entering the low-power mode when the max-power dissipation budget is reached during execution. In these designs, performance is typically compromised in favor of keeping the processor’s operating temperature under a safe bound. Such a design is aimed at lowering the cost of packaging and cooling systems, thus maximizing energy efficiency is less a first priority.

Additional hardware modules will consume additional leakage power, reducing battery life when idle or in sleep mode. While the extra hardware is consuming no switching activity as discussed in section 3.2.1, it still represents a problem in the total system energy picture as shown in Section 3.1. Any low-power architecture research that introduce new hardware components should take static energy into consideration in localized evaluation, while still portraying the overall energy-delay situation.

4. RELATED WORK

With the increasing importance of power modelling in current architecture research, various tools and methods have been developed. At the device-physics level [2], analytical models are constructed based on the underlying principles of fabrication methods and materials used. These models are very accurate, but can take substantial time to evaluate due to their complexity. The precision and elegance of

an analytical model, however, has led to much work trying to reduce the complexity while still maintaining sufficient accuracy in results.

Macromodeling [1, 21] can be a top-down approach to using large-scale approximations of behavior or probability-based models to generate power signatures. These models have been applied to the power grid within chips [3], as well as “cycle-accurate” simulators on an RTL basis [35]. Various projects have attempted to correlate and compare the errors between macro-models and more traditional designs [1].

The alternative bottom-up approach incorporates the design and/or layout of gates with simulation in detail, generally via SPICE-based tools. The results of these models are then fed up into systems that approximate power on the larger scale. Such methods can be reduced to basic gate designs via compound gate breakdown [7], or analysis of the analog underpinnings for use by higher-level tools [26].

Regardless of whether top-down or bottom-up approaches are used, all macro-models share in common the need for training based on input vectors. While the simulated architecture programs substantially match the characteristics of the training data, for those programs that deviate the reported results can be off by over 20%.

Popular tools such as Wattch [5] and SimplePower [36] include cycle-based power analysis support based on event enumeration for each microarchitectural block. Other efforts are trying to tie models in to the instruction set, such that estimation of power in a processor can be determined by the program to be run [29, 17]. Researchers also combine various methods by hand to derive interesting results [16]. While these tools are powerful and useful, care must be exercised in reporting and interpreting the results to avoid the problems and confusions discussed in this paper.

5. CONCLUSIONS

The objective of this paper is not to invalidate any prior published work in low-power architectural or microarchitectural techniques. Instead, we hope that more meticulousity and preciseness will be paid to the way the energy and performance are evaluated and reported in order to eliminate misconceptions and erroneous conclusions. It is insufficient to assume the CPU is the only power drain in a system that includes DRAM, network, graphics, and many other components. When evaluating the energy-delay impact of an architectural design, it is imperative to be more precise as to how the energy is being measured, and what process is being targeted. As we moved into the DSM era, leakage currents will increasingly dominate the total power dissipation. As new architectural designs are proposed, analysis of whether leakage power offsets any gain in dynamic power should be considered and evaluated in a more precise manner.

6. REFERENCES

- [1] Giuseppe Bernacchia and Marios C. Papaefthymiou. Analytical Macromodeling for High-Level Power Estimation. Technical Report CSE-TR-393-99, University of Michigan – Ann-Arbor, EECS, 1999.
- [2] Azeez J. Bhavnagarwala, Blanca L. Austin, Keith A. Bowman, and James D. Meindl. A Minimum Total Power Methodology for Projecting Limits on CMOS GSI. In *IEEE Transactions on VLSI Systems*, pages 235–251, Vol. 8, No. 3, 2000.
- [3] Srinivas Bodapati and Farid N. Najm. High-Level Current Macro-Model for Power Grid Analysis. In *Proceedings of the 39th Design Automation Conference*, 2002.
- [4] David Brooks and Margaret Martonosi. Dynamic Thermal Management for High-Performance Microprocessors. In *Proceedings of the 7th International Symposium on High Performance Computer Architecture*, 2001.
- [5] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. *Proceedings of the 27th International Symposium on Computer Architecture*, 2000.
- [6] J. Adam Butts and Guri Sohi. A Static Power Model for Architects. In *Proceedings of the 33rd Annual International Symposium on Microarchitecture*, December 2000.
- [7] Alexander Chatzigeorgiou, Spiridon Nikolaidis, and Ioannis Tsoukalas. A Modeling Technique for CMOS Gates. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 557–575, Vol. 18, No. 5, 1999.
- [8] A. Dhodapkar, C. Lim, G. Cai, and W. Daasch. TEM²P²EST: A Thermal Enabled Multi-Model Power/Performance ESTimator. In *Workshop on Power-Aware Computer Systems*, 2000.
- [9] R. Gonzales and M. Horowitz. Energy Dissipation In General Purpose Microprocessors. *IEEE Journal of Solid-State Circuits*, Vol. 31, No. 9, 1996.
- [10] Michael Huang, Jose Renau, Seung-Moon Yoo, and Josep Torrellas. A Framework for Dynamic Energy Efficiency and Temperature Management. In *Proceedings of the 33rd Annual International Symposium on Microarchitecture*, 2000.
- [11] Koji Inoue, Vasily G. Moshnyaga, and Kazuaki Murakami. Omitting Cache Look-Up for High-Performance, Low-Power Microprocessors. In *Asia-Pacific Conference on ASICs*, 2001.
- [12] Anoop Iyer and Diana Marculescu. Power Aware Microarchitecture Resource Scaling. In *Design Automation and Test in Europe*, March 2001.
- [13] J. Kin, M. Gupta, and W. H. Mangione-Smith. Filtering Memory References to Increase Energy Efficiency. *IEEE Transactions on Computers*, Vol. 49, No. 1, 2000.
- [14] Alexander Klaiber. The Technology Behind Crusoe Processors: Low-Power x86-Compatible Processors Implemented with Code Morphing Software. Technical Report 2002-20, Transmeta Corporation, 2000.
- [15] Masaaki Kondo and et al. Reducing Memory System Energy in Data Intensive Computations by Software-Controlled On-Chip Memory. In *International Workshop on Compilers and Operating Systems for Low Power*, September 2002.
- [16] Pinar Korkmaz and et al. Energy Modeling of a Processor Core using Synopsys and of Memory Hierarchy using Kamble and Ghose Model. Technical Report CREST-TR-02-002, Georgia Institute of Technology, February 2002.
- [17] Chandra Krintz, Ye Wen, and Rich Wolski. Predicting Program Power Consumption. Technical Report 2002-20, University of California – Santa Barbara, 2002.
- [18] Ram Krishnamurthy, Atila Alvandpour, Vivek De, and Shekhar Borkar. High-Performance and Low-Power Challenges for Sub-70nm Microprocessor Circuits. In *Custom Integrated Circuits Conference*, 2002.
- [19] Srilatha Manne, Artur Klauser, and Dirk Grunwald. Pipeline Gating: Speculation Control for Energy Reduction. *Proceedings of the 25th Annual International Symposium on Computer Architecture*, 1998.
- [20] J. Montanaro and et al. A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor. *Digital Technical Journal*, Vol.9 No.1, November 1996.
- [21] Farid Najm. A Survey of Power Estimation Techniques in VLSI Circuits. In *IEEE Transactions on VLSI Systems*, pages 446–455, Vol. 2, No. 4, 1994.
- [22] Wolfgang Nebel and Jean Mermet, editors. *Low Power Design in Deep Submicron Electronics*. Kluwer Academic Publishers, 1997.

- [23] Dmitry Ponomarev, Gurhan Kucuk, and Kanad Ghose. Reducing Power Requirements of Instruction Scheduling Through Dynamic Allocation of Multiple Datapath Resources. In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, 2001.
- [24] Ramu Pyreddy and Gary Tyson. Evaluating Design Tradeoffs in Dual Pipelines. In *Workshop on Complexity-Effective Design in conjunction with the 28th Annual Symposium on Computer Architecture*, 2001.
- [25] Erven Rohou and Michael D. Smith. Dynamically Managing Processor Temperature and Power. In *The 2nd Workshop on Feedback Directed Optimization*, 1999.
- [26] J.L. Rossello and Jaume Segura. Power-Delay Modeling of Dynamic CMOS Gates for Circuit Optimization. In *International Conference on Computer-Aided Design*, 2001.
- [27] H. Sanchez, R. Philip, J. Alvarez, and G. Gerosa. A CMOS Temperature Sensor for PowerPC RISC Microprocessors. In *Proceedings of the Symposium on VLSI Circuits*, 1997.
- [28] John S. Seng, Eric S. Tune, and Dean M. Tullsen. Reducing Power with Dynamic Critical Path Information. In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, 2001.
- [29] T.K. Tan, G. Lakshminarayana, and N.K. Jha. High-level Software Energy Macro-modeling. In *Proceedings of the 38th Design Automation Conference*, 2001.
- [30] Weiyu Tang, Rajesh Gupta, and Alexandru Nicolau. Power Savings in Embedded Processors through Decode Filter Cache. In *Design Automation and Test in Europe*, March 2002.
- [31] S. Thompson, P. Packan, and M. Bohr. MOS Scaling: Transistor Challenges for the 21st Century. *Intel Technology Journal*, Q3, 1998.
- [32] John P. Uyemura. *Introduction to VLSI Circuits and Systems*. Wiley and Sons, 2002.
- [33] Frank Vahid and Ann Gordon-Ross. A Self-Optimizing Embedded Microprocessor Using a Loop Table for Low Power. In *Proceedings of the International Symposium on Low Power Electronics and Design*, 2001.
- [34] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design: A System Perspective*. Addison Wesley, second edition, 1993.
- [35] Qing Wu, Qinru Qiu, Massoud Pedram, and Chih-Shun Ding. Cycle-Accurate Macro-Models for RT-Level Power Analysis. In *IEEE Transactions on VLSI Systems*, pages 520–528, Vol. 6, No. 4, 1998.
- [36] W. Ye, Narayanan Vijaykrishnan, Mahmut T. Kandemir, and Mary Jane Irwin. The Design and Use of Simplepower: a Cycle-accurate Energy Estimation Tool. In *Proceedings of the 37th Design Automation Conference*, pages 340–345, 2000.