

# Processor Design for Portable Systems

**Thomas D. Burd and Robert W. Brodersen**

Department of EECS, University of California at Berkeley

*Abstract: Processors used in portable systems must provide highly energy-efficient operation, due to the importance of battery weight and size, without compromising high performance when the user requires it. The user-dependent modes of operation of a processor in portable systems are described and separate metrics for energy efficiency for each of them are found to be required. A variety of well known low-power techniques are re-evaluated against these metrics and in some cases are not found to be appropriate leading to a set of energy-efficient design principles. Also, the importance of idle energy reduction and the joint optimization of hardware and software will be examined for achieving the ultimate in low-energy, high-performance design.*

## **1. Introduction**

The recent explosive growth in portable electronics requires energy conscious design, without sacrificing performance. Simply increasing the battery capacity is not sufficient because the battery has become a significant fraction of the total device volume and weight [1][2][3]. Thus, it has become imperative to minimize the load on the battery while simultaneously increasing the speed of computation to handle ever more demanding tasks.

One successful approach is to move the data processing (e.g. handwriting and speech recognition, data encoding and decoding, etc.) to specialized DSP integrated circuits, which can achieve orders of magnitude improvement in energy efficiency [4]. However, there exists a large amount of control processing (e.g. operating system, network control, peripheral control, etc.) that cannot be suitably implemented on dedicated architectures. New approaches to energy-efficient design are required for reducing the energy consumption of programmable processors that are needed for this kind of processing.

There are three main elements to energy-efficient processor design. First, the best available technology should be used, since energy efficiency improves quadratically with technology [5]. Next, design for energy efficiency should be considered from the outset, rather than retrofitting an existing design. In doing so, the portable device must be optimized as a complete system, rather than optimizing individual components. Finally, energy-efficient design techniques should be aggressively utilized in the processor implementation.

A framework for an energy-efficient design methodology suitable for a processor used in a portable environment will be presented. First, the unique demands on a processor in this environment will be examined. Starting from simple analytic models for delay and power in CMOS circuits, metrics of energy efficiency for the processor will be quantified. These metrics will then be applied to develop four important principles of energy-efficient processor design. This paper will conclude with a survey of both hardware and software techniques, and their relative impact on processor energy efficiency.

## 2. Operation in a Portable Environment

Since portable devices (e.g., notebook computers, PDAs, cellular phones, etc.) are single-user systems, their usage is typically bursty; that is, the useful computation is interleaved with periods of idle time. Also, portable devices demand minimum energy consumption to maximize battery life, whereas desktop computers require minimum power dissipation to minimize heat dissipation. The growing single-user “green” PC movement, however, has the same demands on performance and energy consumption as portable devices since it is total energy which is being limited.

### 2.1 Processor Usage Model

Understanding a processor’s usage pattern in portable applications is essential to its optimization. Processor utilization can be evaluated as the amount of processing required and the allowable latency before its completion. These two parameters can be merged into a single measure. Throughput,  $T$ , is defined as the number of operations that can be performed in a given time:

$$\text{Throughput} \equiv T = \frac{\text{Operations}}{\text{Second}} \quad (1)$$

Operations are defined as the basic unit of computation and can be as fine-grained as instructions or more coarse-grained as programs. This leads to measures of throughput of MIPS (instructions/sec) and SPECint92 (programs/sec) which compare the throughput on implementations of the same instruction set architecture (ISA), or different ISAs, respectively.

Figure 1 plots a sample usage pattern which shows that the desired throughput falls into one of three categories. Compute-intensive and minimum-latency processes (e.g., spreadsheet update, spell-check, scientific computation, etc.) desire maximum performance, which is limited by the peak throughput of the processor,  $T_{MAX}$ . Background and relatively high-latency processes (e.g., screen update, low-bandwidth I/O control, data entry, etc.) do not desire the full throughput of the processor, but just a fraction of it as there is no intrinsic benefit to exceeding their latency requirements. When there are no active processes, the processor idles and has zero desired throughput.

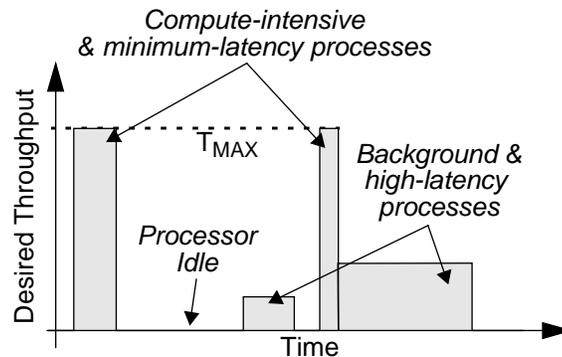


Figure 1. Processor Usage Model.

### 2.2 What Should Be Optimized?

Any increase in  $T_{MAX}$  can be readily employed by compute-intensive and minimum-latency processes. In contrast, the background and high-latency processes do not benefit from any increase in  $T_{MAX}$  above and beyond their average desired throughput since the extra throughput cannot be utilized. Peak throughput is the variable to be maximized since the average throughput is determined by the user.

To maximize the amount of computation delivered per battery life, the energy consumed per operation should be minimized. Only the time it takes to completely discharge the battery is of interest, and not the incremental rate of energy consumption (i.e. power). Thus, the average energy consumed per operation should be minimized, rather than the instantaneous energy consumption.

These are the optimizations to target in energy-efficient processor design: maximize peak deliverable throughput, and minimize average energy consumed per operation. Metrics will be developed in Section 4 to quantitatively measure energy efficiency.

### **2.3 InfoPad: A Portable System Case Study**

The InfoPad is a wireless, multimedia terminal that fits a compact, low-power package in which as much as possible of the processing has been moved onto the backbone network [4]. An RF modem sends/receives data to/from five I/O ports: video input, text/graphics input, pen output, audio input, and audio output. Each I/O port consists of specialized digital ICs, and the I/O device (e.g. LCD, speaker, etc.). In addition, there is an embedded processor subsystem used for data flow and network control. InfoPad is an interesting case study because it contains large amounts of data processing and control processing, which require different optimizations for energy efficiency.

The specialized ICs include a video decompression chip-set which decodes 128 x 240 pixel frames in real-time, at 30 frames/second. The collection of four chips takes in vector quantized data and outputs analog RGB directly to the LCD and dissipates less than 2 mW. Implementing the same decompression in a general purpose processor would require a throughput of around 10 MIPS with hand-optimized code. A processor subsystem designed with the best available parts would dissipate at least 200 mW. This provides a prime example of how dedicated architectures can radically exploit the inherent parallelism of signal processing functions to achieve orders of magnitude reduction of power dissipation over equivalent general-purpose processor-based systems.

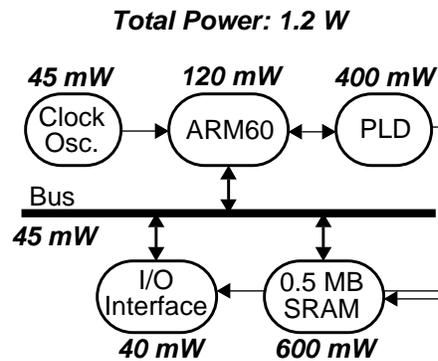
The control processing, which has little parallelism to exploit, is much better suited towards a general purpose processor. An embedded processor system was designed around the ARM60 processor, which combined with SRAM and external glue logic dissipates 1.2 W, while delivering a peak throughput of 10 MIPS. It is this discrepancy of almost 3 orders of magnitude in power dissipation, which can result in the processor sub-system dominating the total energy consumption, that leads to the current objective of substantially reducing the processor energy consumption.

### **2.4 The System Perspective**

In an embedded processor system such as that found in InfoPad, there are essential digital ICs external to the processor required for a functional system: main memory, clock oscillator, I/O interface(s), and system control logic (e.g., PLD). Integrated solutions have been developed for embedded applications that move the system control logic, the oscillator, and even the I/O interface(s) onto the processor chip leaving only the main memory external.

Figure 2 shows a schematic of the InfoPad processor subsystem, which contains the essential system components described above. Interestingly, the processor does not dominate the system's power dissipation; rather, the SRAM memory dissipates half the power. For aggressive low-power design, it is imperative to optimize the entire system and not just a single component; optimizing just the processor in the InfoPad system can yield at most a 10% reduction in power.

While the work presented here focuses just on processor design, in the implementation of a complete system it is important to optimize all components as necessary.



**Figure 2.** InfoPad Processor Subsystem.

High-level processor and system simulation is generally used to verify the functionality of an implementation and find potential performance bottlenecks. Unfortunately, such high-level simulation tools do not exist for energy consumption with the result that simulations to extract energy consumption are typically not made until the design has reached the logic design level. At this time it is very expensive to make significant changes, with the result that it is difficult to make system optimizations for energy consumption through redesign or repartitioning. Only recently has this issue been addressed [6].

It is important to understand how design optimizations in one part of a system may have detrimental effects elsewhere. An example is the effect a processor's on-chip cache on the external memory system. Because smaller memories have lower energy consumption, the designer may try to minimize the on-chip cache size to minimize the energy consumption of the processor at the expense of a small decrease in throughput (due to increased miss rates of the cache). The increased miss rates affect not only the performance, however, but may increase the system energy consumption as well because high-energy main memory accesses are now made more frequently. So, even though the processor's energy consumption was decreased, the total system's energy consumption has increased.

### 3. CMOS Circuit Models

Power dissipation and circuit delays for CMOS circuits can be accurately modeled with simple equations, even for complex processor circuits. These models, along with knowledge about the system architecture, can be used to derive analytical models for energy consumed per operation and peak throughput.

These models will be presented in this section and then used in Section 4 to derive metrics that quantify energy efficiency. With these metrics, the circuit and system design can be optimized for maximum energy efficiency. These models hold only for digital CMOS circuits, and are not applicable to bipolar or BiCMOS circuits. However, this is not very limiting since CMOS is the common technology of choice for low power systems, due to its minimal static power dissipation, and high level of integration.

#### 3.1 Power Dissipation

CMOS circuits have both static and dynamic power dissipation. Static power arises from bias and leakage currents. While static gate loads are usually found in a few specialized circuits such as PLAs, their use has been dramatically

reduced in CMOS designs focussed on low power. Furthermore, careful design of these gates can make their power contribution negligible in circuits that do use them [7]. Leakage currents from reverse-biased diodes of MOS transistors and from MOS subthreshold conduction [8] also dissipate static power but are also insignificant in most designs that dissipate more than 1mW.

The dominant component of power dissipation in CMOS is therefore dynamic. For every low-to-high logic transition in a digital circuit, the capacitance on that node,  $C_L$ , incurs a voltage change  $\Delta V$ , drawing an energy  $C_L \Delta V V_{DD}$  from the supply voltage at potential  $V_{DD}$ . For each node  $n \in N$ , these transitions occur at a fraction  $\alpha_n$  of the clock frequency,  $f_{CLK}$ , so that the total dynamic switching power may be found by summing over all  $N$  nodes in the circuit:

$$Power = V_{DD} \cdot f_{CLK} \cdot \sum_{i=1}^N \alpha_i \cdot C_{L_i} \cdot \Delta V_i \quad (2)$$

Aside from memory bit-lines and low-swing logic, most nodes swing a  $\Delta V$  from ground to  $V_{DD}$ , so that the power equation can be simplified to:

$$Power \cong V_{DD}^2 \cdot f_{CLK} \cdot C_{EFF} \quad (3)$$

where the effective switched capacitance,  $C_{EFF}$ , is commonly expressed as the product of the physical capacitance  $C_L$  and the activity weighting factor  $\alpha$ , each averaged over the  $N$  nodes.

During a transition on the input of a CMOS gate both p and n channel devices may conduct simultaneously, briefly establishing a short from  $V_{DD}$  to ground. In properly designed circuits, however, this short-circuit current typically dissipates a small fraction (5-10%) of the dynamic power [9] and will be omitted in further analyses.

### 3.2 Circuit Delay

To fully utilize its hardware, a digital circuit should be operated at the maximum possible frequency. This maximum frequency is just the inverse of the delay of the processor's critical path.

Until recently, the long-channel delay model suitably modeled delays in CMOS circuits [8]. However, minimum device channel lengths,  $L_{MIN}$ , have scaled below 1 micron, degrading the performance of the device due to velocity saturation of the channel electrons. This phenomenon occurs when the electric field ( $V_{DD}/L_{MIN}$ ) in the channel exceeds 1V/um [10].

$$Delay \cong \frac{C_L}{I_{AVE}} \cdot \frac{\Delta V}{2} \cong \frac{C_L \cdot V_{DD}}{k_V \cdot W \cdot (V_{DD} - V_T - V_{DSAT})} \quad (4)$$

The change in performance can be characterized by the short-channel or velocity-saturated delay model shown in Equation 4.  $I_{AVE}$  is the average current being driven onto  $C_L$ , and is proportional to device width  $W$ , technology constant  $k_V$  and to first-order,  $V_{DD}$ .  $V_T$  is the threshold voltage. For large  $V_{DD}$ ,  $V_{DSAT}$  is constant, with typical magnitude on order of  $V_T$ . For  $V_{DD}$  values less than  $2V_T$ ,  $V_{DSAT}$  asymptotically approaches  $V_{DD} - V_T$ . The important difference between the two delay models is that in the latter, current is roughly linear, and not quadratic, with  $V_{DD}$ .

### 3.3 Throughput

Throughput was previously defined as the number of operations that can be performed in a given time. When clock rate is inversely equal to the critical path delay, throughput is equal to the amount of computational concurrency (i.e. operations completed per clock cycle) divided by the critical path delay:

$$T = \frac{\text{Operations}}{\text{Second}} = \frac{\text{Operations per Clock Cycle}}{\text{Critical Path Delay}} \quad (5)$$

The critical path delay can be related back to the velocity-saturated delay model by summing up the delay over all  $M$  gates in the critical path:

$$\text{Critical Path Delay} \cong \frac{V_{DD}}{k_V \cdot (V_{DD} - V_T - V_{DSAT})} \cdot \sum_{i=1}^M \frac{C_{Li}}{W_i} \quad (6)$$

Making the approximation that all gate delays are equal, Equation 6 can be simplified if  $N_{gates}$  is used to indicate the length of the critical path (i.e. number of gates), and average values for  $C_L$  and  $W$  are used. Throughput can now be expressed as a function of technology parameters, supply voltage, critical path length, and operations per clock cycle:

$$T \cong \frac{k_V \cdot W \cdot (V_{DD} - V_T - V_{DSAT})}{N_{gates} \cdot C_L \cdot V_{DD}} \cdot \frac{\text{Operations}}{\text{Clock Cycle}} \quad (7)$$

As mentioned earlier, typical units for operations per clock cycle are MIPS / Mhz, and SPECint92 / MHz when operations are respectively defined as instructions and benchmark programs.

### 3.4 Energy/Operation

A common measure of energy consumption is the power-delay product (PDP) [11]. This delay is often defined as the critical path delay, so PDP is equivalent to the energy consumed per clock cycle (Power /  $f_{CLK}$ ). However, the measure of interest is the energy consumed per operation which can be derived by dividing the PDP by the operations per clock cycle. The energy consumed per operation can now be expressed as a function of effective switched capacitance, supply voltage, and operations per clock cycle:

$$\frac{\text{Energy}}{\text{Operation}} \cong \frac{V_{DD}^2 \cdot C_{EFF}}{\text{Operations} / \text{Clock Cycle}} \quad (8)$$

### 3.5 Technology Scaling

Although it is usually beyond the control of the IC designer, it is worth noting the impact of technology scaling on throughput and energy/operation. Capacitances and device width,  $W$ , scale down linearly with minimum channel length  $L_{MIN}$ ; transistor current is approximately independent of  $L_{MIN}$  if  $V_{DD}$  remains fixed; technology constant  $k_V$  scales approximately inversely with  $L_{MIN}$ . So, as  $L_{MIN}$  is scaled down, the throughput scales up and the energy/operation is reduced, thus yielding the conclusion that technology scaling is an important strategy for improving energy efficiency.

## 4. Energy Efficiency

While the energy consumed per operation should always be minimized, no single metric quantifies energy efficiency for all digital systems. The metric is dependent on the system's throughput constraint. We will investigate the three main

modes of computation: fixed throughput, maximum throughput, and burst throughput. Each of these modes has a clearly defined metric for measuring energy efficiency, as detailed in the following three sections. While portable devices typically operate in the burst throughput mode, the other two modes are equally important since they are degenerate forms of the burst throughput mode in which the portable device may operate.

#### 4.1 Fixed Throughput Mode

Most real-time systems require a fixed number of operations per second. Any excess throughput cannot be utilized, and therefore needlessly consumes energy. This property defines the fixed throughput mode of computation. Systems operating in this mode are predominantly found in digital signal processing applications in which the throughput is fixed by the rate of an incoming or outgoing real-time signal (e.g., speech, video, handwriting).

$$Metric|_{FIX} = \frac{Power}{Throughput} = \frac{Energy}{Operation} \quad (9)$$

Previous work has shown that the metric of energy efficiency in Equation 9 is valid for the fixed throughput mode of computation [11]. A lower value implies a more energy-efficient solution. If a design can be made twice as energy efficient (i.e. reduce the energy/operation by a factor of two), then its sustainable battery life has been doubled and since the throughput is constant its power dissipation has been halved. For the case of fixed throughput, minimizing the power dissipation is equivalent to minimizing the energy/operation.

#### 4.2 Maximum Throughput Mode

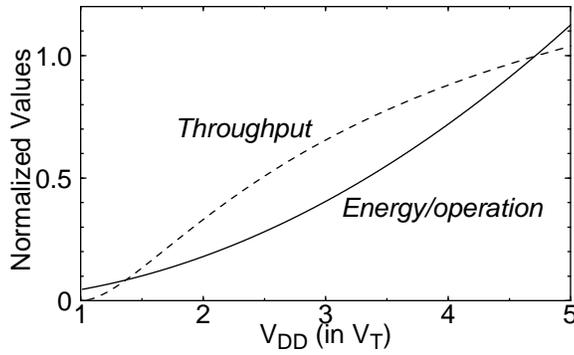
In most multi-user systems, primarily networked desktop computers and mainframes, the processor is continuously running. The faster the processor can perform computation, the better, yielding the defining characteristic of the maximum throughput mode of computation. Thus, this mode's metric of energy efficiency must balance the need for low energy/operation and high throughput, which is accomplished through the use of the Energy to Throughput Ratio, or ETR given in Equation 10,

$$Metric|_{MAX} = ETR = \frac{E_{MAX}}{T_{MAX}} = \frac{Power}{Throughput^2} \quad (10)$$

where  $E_{MAX}$  is the energy/operation, or equivalently power/throughput, and  $T_{MAX}$  is the throughput in this mode

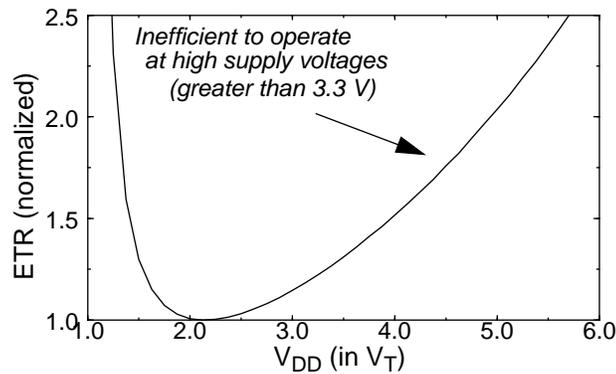
A lower ETR indicates lower energy/operation for equal throughput or equivalently indicates greater throughput for a fixed amount of energy/operation, satisfying the need to equally optimize throughput and energy/operation. Thus, a lower ETR represents a more energy-efficient solution. The Energy-Delay Product [5] is a similar metric, but does not include the effects of architectural parallelism when the delay is taken to be the critical path delay.

Throughput and energy/operation can be scaled with supply voltage, as shown in Figure 3 (the data for Figures 3-5 is derived from Equations 7 and 8, and suitably models sub-micron processes); but, unfortunately, they do not scale proportionally. So while throughput and energy/operation can be varied by well over an order of magnitude to cover a wide dynamic range of operating points, the ETR is not constant for different values of supply voltage.



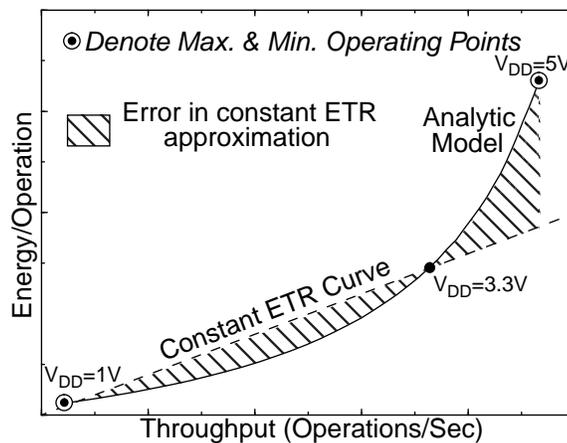
**Figure 3.** Energy/operation, Throughput vs.  $V_{DD}$ .

As shown in Figure 4,  $V_{DD}$  can be adjusted by a factor of almost three ( $1.4V_T$  to  $4V_T$ ) and the ETR only varies within 50% of the minimum at  $2V_T$ . However, outside this range, the ETR rapidly increases. Clearly, for supply voltages greater than 3.3V there is a rapid degradation in energy efficiency, as well as for supply voltages that approach the device threshold voltage.



**Figure 4.** ETR as a function of  $V_{DD}$ .

To compare designs over a larger range of operation for the maximum throughput mode, a better metric is a plot of the energy/operation versus throughput. To make this plot, the supply voltage is varied from the minimum operating voltage (near  $V_T$  in many digital CMOS designs) to the maximum voltage (2.5V- 5V, depending on the technology), while energy/operation and throughput are measured. The energy/operation can then be plotted as a function of throughput, and the architecture is completely characterized over all possible throughput values.



**Figure 5.** Energy vs. Throughput metric.

Using the ETR metric is equivalent to making a linear approximation to the actual energy/operation versus throughput curve. Figure 5 demonstrates the error incurred in using a constant ETR metric, which is calculated at a nominal supply voltage of 3.3V for this example. For architectures with similar throughput, a single ETR value is a reasonable metric for energy efficiency; however, for designs optimized for vastly different values of throughput, a plot may be more useful, as Section 5.1 demonstrates.

### 4.3 Burst Throughput Mode

Most single-user systems (e.g., stand-alone desktop computers, notebook computers, PDAs, etc.) spend a fraction of the time performing useful computation. The rest of the time is spent idling between processes. However, when bursts of computation are demanded, the faster the throughput (or equivalently, response time), the better. This characterizes the burst throughput mode of computation in which most portable devices operate. The metric of energy efficiency used for this mode must balance the desire to minimize energy consumption, while both idling and computing, and to maximize peak throughput when computing.

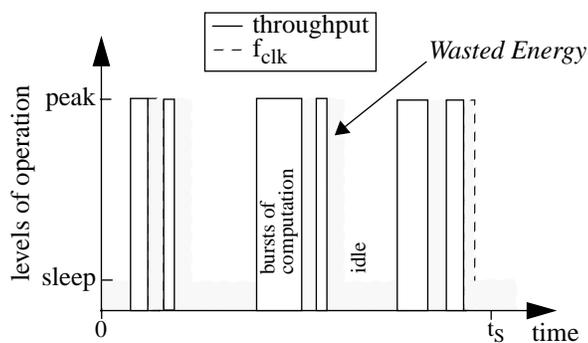


Figure 6. Wasted energy due to idle cycles.

Ideally, the processor’s clock should track the periods of computation in this mode so that when an idle period is entered, the clock is immediately shut off. Then a good metric of energy efficiency is just ETR, as the energy consumed while idling has been eliminated. However, this is not realistic in practice. Many processors do not having an energy saving mode and those that do so generally support only simple clock reduction/deactivation modes. The hypothetical example depicted in Figure 6 contains a clock reduction (sleep) mode in which major sections of the processor are shut down. The shaded area indicates the processor’s idle cycles in which energy is needlessly consumed, and whose magnitude is dependent upon whether the processor is operating in the “low-power” mode.

$$E_{MAX} = \frac{\text{Total Energy Consumed Computing}}{\text{Total Operations}} \quad (11)$$

$$E_{IDLE} = \frac{\text{Total Energy Consumed Idling}}{\text{Total Operations}} \quad (12)$$

Total energy and total operations can be calculated over a large sample time period,  $t_s$ .  $T_{MAX}$  is the peak throughput during the bursts of computation (similar to that defined in Section 4.2), and  $T_{AVE}$  is the time-averaged throughput (total operations /  $t_s$ ). If the time period  $t_s$  is sufficiently long that the operation characterizes the “average” computing demands of the user and/or target system environment yielding the average throughput ( $T_{AVE}$ ), then a good metric of energy efficiency for the burst throughput mode is:

$$Metric|_{BURST} = METR = \frac{E_{MAX} + E_{IDLE}}{T_{MAX}} \quad (13)$$

This metric will be called the Microprocessor ETR (METR); it is similar to ETR, but also accounts for energy consumed while idling. A lower METR represents a more energy-efficient solution.

Multiplying Equation 11 by the actual time computing =  $[t_S \cdot (\text{fraction of time computing})]$ , shows that  $E_{MAX}$  is the ratio of compute power dissipation to peak throughput  $T_{MAX}$ , as previously defined in Section 4.2. Thus,  $E_{MAX}$  is only a function of the hardware and can be measured by operating the processor at full utilization.

$E_{IDLE}$ , however, is a function of  $t_S$  and  $T_{AVE}$ . The power consumed idling must be measured while the processor is operating under typical conditions, and  $T_{AVE}$  must be known to then calculate  $E_{IDLE}$ . However, expressing  $E_{IDLE}$  as a function of  $E_{MAX}$  better illustrates the conditions when idle energy consumption is significant. In doing so,  $E_{IDLE}$  will also be expressed as a function of the idle power dissipation, which is readily calculated and measured, as well as independent of  $t_S$  and  $T_{AVE}$ .

Equation 12 can be rewritten as:

$$E_{IDLE} = \frac{[Idle\ Power\ Dissipation][Time\ Idling]}{[Average\ Throughput][Sample\ Time]} \quad (14)$$

With the Power-Down Efficiency,  $\beta$ , defined as:

$$\beta = \frac{Power\ dissipation\ while\ idling}{Power\ dissipation\ while\ computing} = \frac{P_{IDLE}}{P_{MAX}} \quad (15)$$

$E_{IDLE}$  can now be expressed as a function of  $E_{MAX}$ :

$$E_{IDLE} = \frac{[\beta \cdot E_{MAX} \cdot T_{MAX}] [(1 - T_{AVE}/T_{MAX}) t_S]}{[T_{AVE}] \cdot [t_S]} \quad (16)$$

Equation 17 shows that idle energy consumption dominates total energy consumption when the fractional time spent computing ( $T_{AVE}/T_{MAX}$ ) is less than the fractional power dissipation while idling ( $\beta$ ).

$$METR = ETR \left[ 1 + \beta \left( \frac{T_{MAX}}{T_{AVE}} - 1 \right) \right], \quad T_{MAX} \geq T_{AVE} \quad (17)$$

The METR is a good metric of energy efficiency for all values of  $T_{AVE}$ ,  $T_{MAX}$ , and  $\beta$  as illustrated below by analyzing the two limits of the METR metric.

*Idle Energy Consumption is Negligible* ( $\beta \ll T_{AVE}/T_{MAX}$ ): The metric should simplify to that found in the maximum throughput mode, since it is only during the bursts of computation that energy is consumed and operations performed. For negligible power dissipation during idle, the METR metric in Equation 17 degenerates to the ETR, as expected. For perfect power-down ( $\beta=0$ ) and minimal throughput ( $T_{MAX}=T_{AVE}$ ), the METR is exactly the ETR.

*Idle Energy Consumption Dominates* ( $\beta \gg T_{AVE}/T_{MAX}$ ): The energy efficiency should increase by either reducing the idle energy/operation while maintaining constant throughput, or by increasing the throughput while keeping idle energy/oper-

ation constant. While it might be expected that these are independent optimizations,  $E_{IDLE}$  may be related back to  $E_{MAX}$  and the throughput by  $\beta$  since  $T_{AVE}$  is fixed:

$$\frac{E_{IDLE}}{E_{MAX}} \cong \frac{P_{IDLE}/T_{AVE}}{P_{MAX}/T_{MAX}} = \beta \cdot \frac{T_{MAX}}{T_{AVE}} \quad (18)$$

Expressing  $E_{IDLE}$  as a function of  $E_{MAX}$  yields:

$$METR \cong \frac{\beta \cdot E_{MAX}}{T_{AVE}}, \quad (\text{Idle Energy Dominates}) \quad (19)$$

If  $\beta$  remains constant for varying throughput (and  $E_{MAX}$  stays constant), then  $E_{IDLE}$  scales with throughput as shown in Equation 18. Thus, the METR becomes an energy/operation minimization similar to the fixed throughput mode. However,  $\beta$  may vary with throughput, as will be analyzed further in Section 7.

#### 4.4 Energy Efficiency for Portable Systems

As mentioned earlier, the METR metric measures the energy efficiency of portable systems. Unfortunately, information on the system's average throughput ( $T_{AVE}$ ) is required to utilize this metric and is very application specific. Thus, the METR metric cannot be used to describe the energy efficiency of a processor in general terms. It is only useful when a target application (or class of related applications) has been specified. An example application is the InfoPad, as described in Section 2.3. The processor system is responsible for packet-level network control on the pad and has an average throughput requirement of 0.8 MIPS. If the video decompression was implemented by the processor rather than the custom chip-set, then the average throughput would increase to approximately 11 MIPS.

So that energy-efficient design techniques can be discussed independent of the final application, the METR metric's sub-components, ETR and  $E_{IDLE}$ , will be discussed individually. Section 6 discusses design techniques for optimizing ETR, and Section 7 discusses techniques for minimizing idle energy consumption.

### 5. Design Principles

Four examples are presented below to demonstrate how energy efficiency can be properly quantified. In the process, four design principles follow from the optimization of the previously defined metrics: a high-performance processor can be an energy-efficient processor; idle energy consumption limits the energy efficiency for high-throughput operation; reducing the clock frequency is not energy efficient; and dynamic voltage scaling is energy efficient.

#### 5.1 High Performance is Energy Efficient

Table 1 lists two processors that are available today -- the ARM710 targets the low-power market, and the R4700 targets the mid-range workstation market, and both are fabricated in similar 0.6um technologies, facilitating an equal comparison. The measure of throughput used is SPECint92. The typical metric for measuring energy efficiency is SPECint92/Watt (or SPECfp92/Watt, Dhrystones/Watt, MIPS/Watt, etc.). The ARM710 processor has a SPECint92/Watt five times greater than the R4700's, and the claim then follows that it is "five times as energy efficient". However, this metric only compares operations/energy, and does not weight the fact that the ARM710 has only 15% of the performance as measured by SPECint92.

Table 1. Comparison of two processors [12][13].

Processor	SPECint92 ( $T_{MAX}$ )	Power (Watts)	Supply voltage, $V_{DD}$ (volts)	SPECint92/Watt ( $1/E_{MAX}$ )	ETR ( $10^{-3}$ )
R4700	130	4.0	3.3	33	0.24
ARM710	20	0.12	3.3	167	0.30

The ETR (Watts/SPECint92<sup>2</sup>) metric indicates that the R4700 is actually *more* energy efficient than the ARM710. To quantify the efficiency increase, the plot of energy/operation versus throughput in Figure 7 is used because it better tracks the R4700's energy at the low throughput values. The plot was generated from the throughput and energy/operation models in Section 3.

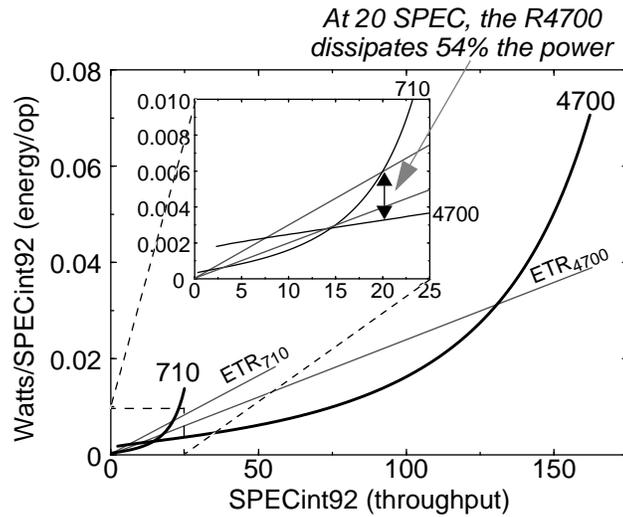


Figure 7. Energy vs. Throughput of R4700 and ARM710.

According to the plot, the R4700 would dissipate 65 mW at 20 SPECint92, or about 1/2 of the ARM710's power, despite the low  $V_{DD}$  ( $1.5 V_T$ ) for the R4700. Conversely, the R4700 can deliver 30 SPECint92 at 120 mW ( $V_{DD}=1.7 V_T$ ), or 150% of the ARM710's throughput.

This does assume that the R4700 processor has been designed so that it can operate at these low supply voltages. If the lower bound on operating voltage is greater than  $1.7 V_T$ , then the ARM710 would be more energy efficient in delivering the 20 SPECint92 than the R4700. Typically, a processor is rated for a fixed standard supply voltage (3.3V or 5.0V) with a +/- 10% tolerance. However, many processors can operate over a much larger range of supply voltages (e.g., 2.7-5.5V for the ARM710 [12], 2.0-3.3V for the Intel486GX [14]). The processor can operate at a non-standard supply voltage by using a high-efficiency, low-voltage DC-DC converter to generate the appropriate supply voltage [15].

While the ETR correctly predicted the more energy-efficient processor at 20 SPECint92, it is important to note that the R4700 is not more energy efficient for all values of SPECint92, as the ETR metric would indicate. Because the nominal throughput of the processors is vastly different, the Energy/Operation versus Throughput metric better tracks the efficiency, and indicates a cross-over throughput of 14.5 SPECint92. Below this value, the ARM710 is more energy efficient.

## 5.2 Fast Operation Can Decrease Energy Efficiency

If the user demands a fast response time, rather than reducing the voltage, as was done in Section 5.1, the processor can be left at the nominal supply voltage, and shut down when it is not needed.

For example, assume the target application has a  $T_{AVE}$  of 20 SPEC, and both the ARM710 and R4700 have a  $\beta$  factor of 0.2. If the processors'  $V_{DD}$  is left at 3.3V, The ARM710's METR is exactly equal to its ETR value, which is  $3.0 \times 10^{-4}$ . It remains the same because it never idles. The R4700, on the other hand, spends 85% ( $1 - T_{AVE}/T_{MAX}$ ) of the time idling, and its METR is  $5.0 \times 10^{-4}$ . Thus, for this scenario, the ARM710 is nearly twice as energy efficient.

However, if the R4700's  $\beta$  can be reduced down to 0.02, then the METR of the R4700 becomes  $2.66 \times 10^{-4}$ , and it is once again the more energy-efficient solution. For this example, the cross-over value of  $\beta$  is 0.045.

This example demonstrates how important it is to use the METR metric instead of the ETR metric if the target application's idle time is significant (i.e.,  $T_{AVE}$  can be characterized and is significantly below  $T_{MAX}$ ). For the above example, a  $\beta$  for the R4700 greater than 0.045 leads the metrics to disagree on which is the more energy-efficient solution. One might argue that the supply voltage can always be reduced on the R4700 so that it is more energy efficient for any required throughput. This is true if the dynamic range of the R4700 is as indicated in Figure 7. However, if some internal logic limited the value that  $V_{DD}$  could be dropped, then the lower bound on the R4700's throughput would be located at a much higher value. Thus, finite  $\beta$  can degrade the energy efficiency of high throughput circuits due to excessive idle power dissipation.

## 5.3 Clock Frequency Reduction is NOT Energy Efficient

A common fallacy is that reducing the clock frequency  $f_{CLK}$  is energy efficient. When compute energy consumption dominates idle energy consumption, it is quite the opposite. At best, it allows an energy-throughput trade-off when idle energy consumption is dominant. The relative amount of time in idle versus maximum throughput is an important consideration in determining the effect of clock frequency reduction.

*Compute energy consumption dominates ( $E_{MAX} \gg E_{IDLE}$ ):* Since compute energy consumption is independent of  $f_{CLK}$ , and throughput scales proportionally with  $f_{CLK}$ , decreasing the clock frequency increases the ETR, indicating a drop in energy efficiency. Halving  $f_{CLK}$  is equivalent to doubling the computation time, while maintaining constant computation per battery life, which is clearly not optimal.

*Idle energy consumption dominates ( $E_{IDLE} \gg E_{MAX}$ ):* Clock reduction may trade-off throughput and energy/operation, but only when the power-down efficiency,  $\beta$ , is independent of throughput such that  $E_{IDLE}$  scales with throughput. When this is so, halving  $f_{CLK}$  will double the computation time, but will also double the amount of computation per battery life. If the currently executing process can accept throughput degradation, then this may be a reasonable trade-off. If  $\beta$  is inversely proportional to throughput, however, then reducing  $f_{CLK}$  does not affect the total energy consumption, and the energy efficiency drops.

Table 2. Impact of Clock Frequency Reduction on Energy Efficiency.

Operating Conditions:	Compute Energy Consumption Dominates	Idle Energy Consumption Dominates	
		$\beta$ independent of throughput	$\beta$ inversely proportional to throughput
Throughput	decreases	decreases	decreases
Energy	unchanged	decreases	unchanged
Energy Efficiency (METR)	decreases	unchanged	<i>decreases</i>

#### 5.4 Dynamic Voltage Scaling is Energy Efficient

If  $V_{DD}$  were to track  $f_{CLK}$ , however, so that the critical path delay remains inversely equal to the clock frequency, then constant energy efficiency could be maintained as  $f_{CLK}$  is varied. This is equivalent to  $V_{DD}$  scaling (Section 4.2) except that it is done dynamically during processor operation. If  $E_{IDLE}$  is present and dominates the total energy consumption, then simultaneous  $f_{CLK}$ ,  $V_{DD}$  reduction during periods of idle will yield a more energy-efficient solution.

Even when idle energy consumption is negligible, dynamic voltage scaling can still provide significant wins. Figure 8 plots a sample usage pattern of desired throughput, with the delivered throughput super-imposed on top. For background and high-latency tasks, the supply voltage can be reduced so that just enough throughput is delivered, which minimizes energy consumption.

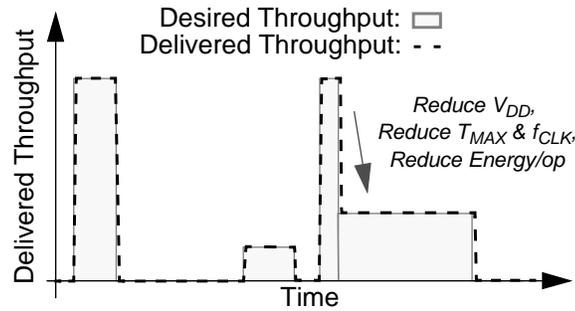


Figure 8. Dynamic Voltage Scaling.

For applications that require maximum deliverable throughput only a small fraction of the time, dynamic voltage scaling has a significant win. For the R4700 processor, the peak throughput is 130 SPECint92. Given a target application where the desired throughput is either a fast 130 SPECint92 or a slow 13 SPECint92, Table 3 lists the peak throughput, average energy/operation, and effective ETR depending on the fraction of time spent in the fast mode. For each category of throughput the total number of operations completed are the same so that the relative changes in battery life can be evenly compared. When that fraction becomes small, the processor's peak throughput is still set by the fast mode, while the average energy consumed per operation is set by the slower mode. Thus, the best of both extremes can be achieved. For simplicity, this examples assumes that idle energy consumption is always negligible

Table 3. Benefits of Dynamic Voltage Scaling.

Throughput:	Time spent operating in:			$T_{MAX}$ (SPECint92)	$E_{AVE}$ (W/SPECint92)	ETR ( $10^{-6}$ )	Normalized Battery Life
	Fast Mode	Slow mode	Idle Mode				
Always full-speed	10%	0%	90%	130	0.031	237	1 hr.
Sometimes full-speed	1%	90%	9%	130	0.006	45.0	5.3 hrs.
Rarely full-speed	0.1%	99%	0.9%	130	0.003	25.8	9.2 hrs.

## 6. Energy-Efficient VLSI Design

When idle energy consumption of the processor is negligible, ETR is a valid energy-efficiency metric. A variety of low-power and low-energy design techniques have been published but are not energy efficient in the ETR sense, if the power/energy savings came at the expense of throughput. Various design techniques drawn from the literature and original research are discussed below as well as their impact on ETR.

The techniques can be categorized into one of three areas of processor design: instruction set architecture, microarchitecture, and circuit design. However, these are not entirely independent; design decisions in one area may impact design decisions made in the other areas.

### 6.1 Instruction Set Architecture

Typically, instruction set architectures (ISA) are designed solely with performance in mind. High-level performance simulators allow the architect to explore the ISA design space with reasonable efficiency. Energy is not a consideration, nor are there high-level simulators available to even let the architect estimate energy consumption. Simulation tools exist, but require a detailed description of the microarchitecture so that they are not useful until the ISA has been completely specified. Processor's targeted towards portable systems should have their ISA designed for energy efficiency, and not just performance.

Many processors have 32-bit instruction words and registers. Register width generally depends on the required memory address space, and cannot be reduced; in fact, more recent microprocessors have moved to 64-bits. For low-energy processors, 16-bit instruction widths have been proposed. Static code density can be reduced by 30-35%, while increasing the dynamic run length by only 15-20% over an equivalent 32-bit processor [16][17]. Using 16-bit instructions reduces the energy cost of an instruction fetch by up to 50% because the size of the memory read has been halved [18]. In system's with 16-bit external busses, the advantage of 16-bit instructions is further widened [19][17]. Since instruction fetch consumes about a third of the processor's energy [20][21], total energy consumption is reduced by 15-20%, which is cancelled out by the 15-20% reduction in performance, giving approximately equal energy efficiency. The available data indicates that this technique is significantly energy efficient only if the external memory's energy consumption dominates the processor's energy consumption, or if the external bus is 16 bits.

The number of registers can be optimized for energy efficiency. The register file consumes a sizable fraction of total energy consumption since it is typically accessed multiple times per cycle (10% of the total energy in [20]). In a register-memory architecture, the number of general purpose registers is kept small and many operands are fetched from memory. Since the energy cost of a cache access surpasses that of a moderately sized (32) register file, this is not energy efficient.

The other extreme is to implement register windows which is essentially a very large (100+) register file. The energy consumed by the register file increases dramatically increasing total processor energy consumption 10-20%. Unless this increase in energy is compensated by an equivalent increase in performance, register windows is not energy efficient. One study compared register files of size 16 and 32 for a given ISA, and found that for 16 registers, the dynamic run length is 8% larger [22]. The corresponding decrease in processor energy due to a smaller register file is on the order of 5-10%. There appears to be a broad optimum on the number of registers since the energy efficiency is near equal for 16 and 32-entry register file.

The issue of supported operation types and addressing modes has been a main philosophical division between the RISC and CISC proponents. While this issue has been debated solely in the context of performance, it can also have an impact on energy consumption. Complex ISAs have higher code density, which reduces the energy consumed fetching instructions and reduces the total number of instructions executed. Simple ISAs typically have simpler data and control paths, which reduces the energy consumed per instruction, but there are more instructions. These trade-offs need to be analyzed when creating an ISA.

The amount of hardware exposed (e.g., branch delay slot, load delay slot, etc.) is another main consideration in ISA design. This is typically done to improve performance by simplifying the hardware implementation. Since the scheduling complexity resides in the compiler, it consumes zero run-time energy while the simplified hardware consumes less energy per operation. Thus, both the performance is increased and the energy/operation is decreased, giving a two-fold increase in energy efficiency. A good example of radically exposing the hardware architecture are very long instruction word (VLIW) architectures, which will be discussed in more detail in the next section.

## 6.2 Microarchitecture

The predominant technique to increase energy efficiency in custom DSP ICs (fixed throughput) is architectural concurrency; with regards to processors, this is generally known as instruction-level parallelism (ILP). Previous work on fixed throughput applications demonstrated an energy efficiency improvement of approximately  $N$  on an  $N$ -way parallel/pipelined architecture [11]. This assumes that the instructions being executed are fully vectorizable, that  $N$  is not excessively large, and that the extra delay and energy overhead for multiplexing and demultiplexing the data is insignificant.

Moderate pipelining (4 or 5 stages), while originally implemented purely for speed, also increases energy efficiency, particularly in RISC processors that operate near one cycle-per-instruction. Energy efficiency can be improved by a factor of two or more [23], and is essential in an energy-efficient processor.

*Superscalar Architectures:* More recent processor designs have implemented superscalar architectures, either with parallel execution units or extended pipelines, in the hope of further increasing the processor concurrency. However, an  $N$ -way superscalar machine will not yield a speedup of  $N$ , due to the limited ILP found in typical code [24][25]. Therefore, the achievable speedup will be less than the number of simultaneous issuable instructions and yields diminishing returns as the peak issue rate is increased. Speedup has been shown to be between two and three for practical hardware implementations in current technology [26].

If the instructions are dynamically scheduled in employing superscalar operation, as is currently common to enable backwards binary compatibility, the  $C_{EFF}$  of the processor will increase due to the implementation of the hardware scheduler. Also, there will be extra capacitive overhead due to branch prediction, operand bypassing, bus arbitration, etc. There will be additional capacitance increase because the  $N$  instructions are fetched simultaneously from the cache and may not all be issuable if a branch is present. The capacitance switched for un-issued instructions is amortized over those instructions that are issued, further increasing  $C_{EFF}$ .

The energy efficiency increase can be analytically modeled. Equation 20 gives the ETR ratio of a superscalar architecture versus a simple scalar processor; a value larger than one indicates that the superscalar design is more energy efficient. The  $S$  term is the ratio of the throughputs, and the  $C_{EFF}$  terms are from the ratio of the energies (architectures are compared at constant supply voltage). The individual terms represent the contribution of the datapaths,  $C_{EFF}^{Dx}$ , the memory sub-system,  $C_{EFF}^{Mx}$ , and the dynamic scheduler and other control overhead,  $C_{EFF}^{Cx}$ . The 0 suffix denotes the scalar implementation, while the 1 suffix denotes the superscalar implementation. The quantity  $C_{EFF}^{C0}$  has been omitted, because it has been observed that the control overhead of the scalar processor is minimal:  $C_{EFF}^{C0} \ll C_{EFF}^{D0, M0}$  [20].

$$ETR|_{RATIO} = \frac{S(C_{EFF}^{D0} + C_{EFF}^{M0})}{(C_{EFF}^{C1} + C_{EFF}^{D1} + C_{EFF}^{M1})} \quad (20)$$

Simulation results show that  $C_{EFF}^{C1}$  is significant due to control overhead and that  $C_{EFF}^{M1}$  is greater than  $C_{EFF}^{M0}$  due to un-issued instructions negating the increase due to  $S$ . Since  $C_{EFF}^{C1}$  increases quadratically as the number of parallel functional units is increased, the largest improvement in energy efficiency would be expected for moderate amounts of parallelism. In this best case, however, the superscalar architecture yields no improvement in energy efficiency [23].

*Superpipelined Architectures:* These architectures also exploit ILP and offer speedups similar to those found in superscalar architectures [27], but their performance is lower because the number of stall cycles increases with the depth of the pipeline due to data dependencies. While these architectures do not need as complex hardware for the dynamic scheduler ( $C_{EFF}^{Cx}$  is lower), they do need extra hardware for more complex operand bypassing ( $C_{EFF}^{Dx}$  is higher). The net differences in speedup and capacitance should give superpipelined architectures an energy efficiency similar to superscalar architectures.

*VLIW Architectures:* These architectures best exploit ILP by exposing the underlying parallelism of the hardware to the compiler's scheduler which minimizes the complexity of the hardware. A good compiler is necessary to fully utilize the hardware. One such implementation from Multiflow gives a speedup factor,  $S$ , between 2 and 6 [28]. Because the parallelism is visible, VLIW processors do not require aggressive branch prediction, dynamic schedulers, and complex bus arbitration, so that the energy consumed per operation is roughly the same as for the scalar processor. The main additional energy cost is for the communication network that connects the autonomous functional units that comprise the VLIW processor, and executing the instructions that shuffle data between them. Even assuming a worst case energy per operation increase of 50%, the VLIW processor's energy efficiency increases anywhere from 33% to 300%.

On-chip caches reduce off-chip communication that is both slow and energy consuming. Caches consume around a third of the processor's energy consumption (50% in [20]). Designing the cache in a sectored (or sub-banked) manner, such that

only one part of the SRAM array is activated per memory access reduces energy/access and increases throughput [4]; this is recommended technique for any memory larger than one kilobyte.

For split caches, the instruction cache consumes up to four times the data cache's energy consumption since loads and stores do not occur every instruction. For the instruction cache, an instruction buffer (or Level 0 cache) can dramatically increase energy efficiency by exploiting the spatial locality of instructions [18]. When a cache line is accessed, it is placed into a buffer, and the instruction cache is not accessed again until the instruction buffer takes a miss. For a 32-byte wide buffer, the hit rate is around 80% [18]; this reduces the instruction cache energy consumption up to 80%. If the buffer is designed to have no penalty on a miss so that performance is unchanged, the processor energy efficiency can be improved by 15 - 25%. Further techniques have been proposed to reduce the accesses to the instruction cache's tag array by exploiting this same spatial locality, increasing processor energy efficiency of 5-10% [29].

The processor control typically knows which pipeline stages are being used each cycle. Those pipeline stages not used in a given cycle should have their clock disabled for that cycle. This is particularly important to do in superscalar architectures that typically have only a fraction of the entire processor being utilized in any given cycle. With only a small overhead cost, this technique increases processor energy efficiency by 15-25% (estimated that 40-50% of the processor is disabled 40-50% of the time) [30]. To maximize the benefit of clock-gating, NOP instructions should be suppressed. In many microarchitectures, NOP instructions are mapped to real instructions. Although NOPs write to a null register, they consume more than half the energy of a normal instruction, as demonstrated by empirical measurements described in [42]. Instead, NOPs should be detected by a comparator in the instruction decode stage, and later stages executing on the NOP should be disabled. Similarly, pipeline stalls and/or bubbles should not inject NOP instructions into the pipeline but should instead cause subsequent pipeline stages to be disabled during the appropriate cycle.

Correlation of data is often exploited for energy efficiency in signal processing circuits. While processors do not exhibit the same level of correlation as found in DSP circuits, high amounts of correlation can be found in calculating the effective address which is typically offset from a high-valued stack pointer. In most scalar processors, a single ALU calculates the effective addresses and all integer additions. By partitioning these two types of additions onto separate adders, the signal correlation increases by 16%, decreasing the adder's energy consumed per addition by an equivalent 16%. Total processor energy efficiency is then increased by 3-7%.

The ETR metric can be used to evaluate other microarchitectural design decisions for their relative energy efficiency. For those decisions with more than one feasible approach, the relative ETRs can be compared to select the most energy-efficient alternative.

### **6.3 Circuit Design**

A variety of energy-efficient design techniques exist at the circuit design level. Many were developed in earlier research targeted towards custom DSP (fixed throughput) design, and some remain applicable to general-purpose processor design [31]. The ETR metric can be used to determine which of these "low power" techniques are also energy-efficient design choices. Of the three levels of the processor design hierarchy, the circuit design level has by far the largest amount of previously results from which to draw.

For example, the topologies for the various macrocells (e.g. adder, register file, etc.) should be selected by their ETR, and neither solely for speed nor solely for energy. A variety of studies have been made to study the relative energy consumption and speed of various macrocells, which can be used to aid in making design decisions [32][33]. Similar studies have also been with respect to various logic design styles [34].

Transistor level optimizations can be made, such as minimizing all devices not in the critical path(s). This typically requires have a fast and slow versions of the same cell, and the cell selection is based on whether it is in the critical path(s) or not [30]. Low-voltage swing circuits for large capacitive nodes, such as those found in memories and global busses, can significantly drop energy consumption, while improving speed at the same time [7].

## 7. Minimizing Idle Energy Consumption

As demonstrated in Section 2.1, when the processor is not actively computing on user or background tasks, the desired throughput is zero. Any throughput delivered by the processor in this idle mode needlessly consumes energy. The METR metric is revisited to understand when idle energy consumption is important followed by a survey of design techniques to minimize this energy.

### 7.1 Optimizing METR.

Equation 17 shows that when the fractional time spent computing ( $T_{AVE}/T_{MAX}$ ) is less than the fractional power dissipation while idling ( $\beta$ ), idle energy consumption dominates total energy consumption. Then the METR optimization is to minimize  $\beta$  and  $E_{MAX}$  as shown in Equation 19. Furthermore, the exact optimization depends on whether  $\beta$  changes as the throughput is varied as shown below.

*$\beta$  is independent of throughput:* This is the case when the processor has no power-down mode. If the clock frequency remains the same, or proportional, during both the computation and idle periods, then idle power dissipation tracks compute power dissipation. Idle energy consumption cannot be optimized independent of throughput and compute energy consumption. If throughput increases, the compute power dissipation increases, and the idle power dissipation and energy consumption increases proportionally. Minimizing the compute energy consumption will have a proportional decrease in the idle energy consumption.

*$\beta$  varies with throughput:* This case occurs for processors that implement idle power down modes in which idle power dissipation is independent of compute power dissipation. It is energy efficient to maximize throughput, since idle energy consumption will remain constant and dominate compute energy consumption. In practice,  $\beta$  will be less than inversely proportional to throughput (e.g. due to latency switching between operating modes) so that idle energy consumption is not entirely independent of throughput. However, energy efficiency will continue to increase with throughput until idle energy consumption is no longer dominant.

### 7.2 Power Down Modes

Unless there is specific hardware support to externally disable a processor's clock to turn off the processor when it is not being utilized, the processor typically executes a busy wait loop, which consists of NOP instructions. The processor hardware has an intrinsic, moderately-valued  $\beta$  which can be estimated or measured as the ratio of the power dissipated exe-

cutting a NOP instruction to the power dissipated executing a typical instruction. Even if the clock is gated to those pipeline sections executing NOP instructions, the instruction-memory access per cycle will continue to consume energy. For a laptop computer in which average throughput is on the order of 1 SPECint92 (high estimate for user's average operations/second) and  $\beta$  is reasonably estimated as 0.2, increasing the peak throughput of the processor beyond 5 SPECint92 reduces the processor energy efficiency. This is equivalent to a 386-class processor. To deliver a more tolerable response time to the user, energy efficiency will have to be degraded.

An alternative to degrading energy efficiency is to implement power down modes. To achieve their full benefit requires an energy-conscious operating system that utilizes them. Then,  $\beta$  can be decreased by one or more orders of magnitude.  $\beta$  will typically become a function of throughput since the operating system can decouple the compute and idle regimes' power dissipation. There may also be multiple values of  $\beta$ , one for each power down modes.

The design of the PowerPC 603 processor provides a good demonstration of useful power down modes to include [30]. A doze mode stops the processor from fetching instructions, but keeps alive snoop logic for cache coherency and the clock generation and timer circuits, giving a  $\beta$  of 0.16 for this mode. A nap mode disables the snoop logic, only keeping alive the timer logic, dropping the  $\beta$  to 0.06. Lastly, there is a sleep mode which only keeps alive the PLL and clock. The  $\beta$  for this mode is 0.05, while the processor can be up and running at full speed within 10 clock cycles, and a cache flush. Further power reduction can be achieved by disabling the PLL in the sleep mode, which drops the  $\beta$  down to 0.002, but at the cost of several thousand cycles (up to 200 usec) to return to full speed.

It is important to notice how much the PLL, which is found on most microprocessors, limits the reduction of idle energy consumption. Frequently turning off the PLL is not a viable approach due to the large overhead of restarting it. Techniques for improving the energy efficiency of PLLs in power down modes are needed.

While most microcontrollers and some embedded processors have power down modes, only a few microprocessors have them. It is an important technique to include in energy-efficient processors. The actual energy savings, though, depends more on how well the operating system can utilize these modes.

### 7.3 Transition Time

There is an energy cost associated with entering and leaving power down modes. When entering a mode, the processor will continue to operate at full throughput and energy consumption for a number of cycles while cleaning up state in the processor. This creates an energy penalty. When leaving a mode, there is usually a latency incurred to restart the processor, which creates an effective throughput penalty. Restarting a PLL can cost several thousand clock cycles.

A metric has been proposed to measure this cost, the Cycles Per Stop Ratio, or CPSR, which includes the entry, exit, and processing overhead of entering a power down mode [35]. This is useful for first-order comparisons of various power down methods, but does not accurately measure the energy consumed per operation and performance.

To activate the power down modes for the PowerPC 603, the processor must handshake with the system logic via external control lines. In many microcontrollers and embedded microprocessors, instructions have been added to the ISA to directly activate power down modes. The benefit of this is to reduce the amount of time it takes to transition between

modes, and in the best case, an instruction added to the ISA to shutdown the processor can take effect with one cycle of latency.

Included in the turn-off or restart time is the number of cycles it takes to save the internal processor state to memory. To save state in a consistent manner, it is best to allow the operating system to invoke the processor shut down instructions.

The cost of restarting the processor includes a one or two cycle delay for synchronously un-gating the clock, and a number of cycles delay equal to the pipeline depth to restart the pipeline. The PowerPC 603 is reported to have a start-up time of under 10 cycles for all the power down modes that leave the PLL running. The biggest latency cost is to restart the PLL, which typically takes 10-100 usec. to lock, and in the case of the 603 processor, can be as high as 200 usec. [30]. However, this is for an analog PLL. A digital PLL has been implemented with a reported lock time of under 2 usec., drastically reducing the start-up cost when in a fully powered-down mode (i.e. clock generation disabled) [36].

## **8. Energy-Efficient Software**

Power down modes and halt instructions provide no benefit unless they are effectively used by the software running on the processor. Thus, an energy-efficiency minded operating system is crucial in portable systems. Other energy reductions can be achieved through variable-performance schedulers and optimizing compilers.

### **8.1 Operating System**

The processor should be completely disabled during idle periods to minimize idle energy consumption. Only the operating system has knowledge when there are no more pending events to process, and can invoke processor halt instructions to disable the processor. The operating system is central to system power management for a portable system.

Since the operating system is also aware of the peripheral hardware components' usage (e.g. disk drive, LCD, network controller, etc.), it should be given the ability to switch the power on and off to these devices as well. This is common practice in most notebook computers today and can reduce energy consumption by up to 50% [37]. With more aggressive design, such as a proposed technique for predictive shutdown of system components [38], this should be able to be reduced further.

Intel and Microsoft has put forth a specification called Advanced Power Management (APM) [39]. This specification defines an interface between hardware-specific power management software, which resides in the BIOS, and a hardware-independent operating system power management driver. This driver can manage APM-aware applications, by notifying them of impending processor state changes, and it provides an API that allows applications to directly employ power management. In a multi-tasking operating system, the driver will also negotiate conflicting power management requests. This vertical approach to power management shows great promise for further reductions in energy consumption.

### **8.2 Variable Performance Scheduling**

Software processes have different performance and latency demands as shown in Section 2.1. Not every process needs the peak throughput of the processor. The supply voltage, along with the clock frequency, can be reduced to just meet the required throughput for those processes with lower performance demands, yielding a reduction in energy consumption, as described in Section 5.4.

The operating system can set the performance level at the time of a process context switch, with the level proportional to the priority level of the process. Most operating systems have the concept of process priority levels and the granularity of the performance settings increases with number of priority levels.

Another approach is to use predictive scheduling in which CPU performance is incrementally changed over finite intervals [40][41]. The amount of performance delivered in the current time interval is set by evaluating CPU activity in previous intervals, using a variety of averaging algorithms. This technique dynamically trades off throughput and energy with no knowledge of what process is being executed.

### **8.3 Algorithms and Compilers**

Algorithms have always been tuned and optimized for maximum performance. These same techniques have a large impact on energy efficiency, as well. By using an algorithm implementation that requires fewer operations, both the throughput is increased, and less energy is consumed because the total amount of switched capacitance to execute the program has been reduced. A quadratic improvement in ETR can be achieved [5]. This same improvement holds for optimizing compilers which also try to minimize a program's dynamic run length, as demonstrated empirically in [42].

This does not always imply that the program with the smallest dynamic instruction count (path length) is the most energy efficient, since the switching activity per instruction must be evaluated. The work presented in [42] demonstrates through empirical measurements that the energy consumed per cycle is roughly constant, so that by minimizing execution time of the program, the energy consumption will be minimized. This implies that when considering the energy overhead for each cycle (e.g. clocks, instruction fetch, etc.), the key parameter to minimize is cycle count, and not instruction count.

## **9. Conclusions**

Processors used in portable systems have a usage pattern in which the desirable throughput varies. Compute-intensive processes desire maximum throughput and high-latency processes desire less than maximum throughput to sufficiently complete. When no processes are pending, the processor idles and yields zero throughput. The important optimizations for these processors are to maximize throughput, which minimizes the response latency of the system, and minimize average energy consumed per operation, which maximizes the computation delivered over the life of the battery.

Metrics for energy efficiency have been defined for the three modes of computation that characterize typical processor operation. In particular, an energy efficiency metric, called is the Microprocessor Energy Throughput Ratio, or METR, was defined which describes typical processor usage in a portable system. In addition to the energy consumed while computing it includes the energy consumed while idling which can dominate total energy consumption in user-interactive applications. When the idle energy consumption is negligible, METR degenerates to the Energy Throughput Ratio or ETR. Because of the variation of ETR with supply voltage, a better metric though less convenient, is the complete curve of energy/operation versus throughput for which ETR is just a linear approximation.

Four important design principles were developed to aid in energy-efficient design. High performance design was shown to be similar to energy-efficient design. Actually operating at high speeds, however, may not be energy efficient if idle energy consumption becomes dominant. Clock frequency reduction which is generally believed to be a method of improv-

ing battery life can actually be detrimental in some circumstances. However, if this reduction is coupled with an equivalent reduction in supply voltage and is performed dynamically depending on the performance requirements, then it becomes energy efficient.

A variety of new and existing design techniques were evaluated for energy efficiency. Some techniques that are low-energy such as 16-bit instructions were shown to not be energy efficient since the reduction in energy came at the expense of too much a reduction in throughput. Other techniques such as pipelining and cache sectoring were shown to be indispensable for energy-efficient design.

Decreasing the idle energy consumption is critical to the design of an energy-efficient processor and complete shut down of the clock while idling is optimal. If this cannot be accomplished, then it is imperative that the operating system implement a power down mode so that the idle power dissipation becomes independent of the computing power dissipation. Then the METR optimization will maximize the throughput delivered to the user in an energy-efficient manner. Otherwise, if idle power dissipation is proportional to the compute power dissipation, achieving energy-efficient operation requires the throughput to be minimized.

### **Acknowledgments**

This research is sponsored by ARPA. We would like to thank Arthur Abnous, Andy Burstein, Dave Lidsky, Trevor Pering, Tony Stratakos, and the reviewers of this paper for their invaluable input.

## **10. References**

1. S. Kunii, "Means of Realizing Long Battery Life in Portable PCs," *Proceedings of the IEEE Symposium on Low Power Electronics*, Oct. 1995, pp. 12-3.
2. M. Culbert, "Low Power Hardware for a High Performance PDA," *Proceedings of the Thirty-Ninth IEEE Computer Society International Conference*, Mar. 1994, pp. 144-7.
3. T. Ikeda, "ThinkPad Low-Power Evolution," *Proceedings of the IEEE Symposium on Low Power Electronics*, Oct. 1995, pp. 6-7.
4. A. Chandrakasan, A. Burstein, and R.W. Brodersen, "A Low Power Chipset for Portable Multimedia Applications," *IEEE Journal of Solid State Circuits*, Vol. 29, Dec. 1994, pp. 1415-28.
5. M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-Power Digital Design," *Proceedings of the IEEE Symposium on Low Power Electronics*, Oct. 1994, pp. 8-11.
6. D. Lidsky and J. Rabaey, "Early Power Exploration - A World Wide Web Application," *Proceedings of the Thirty-Third Design Automation Conference*, June 1996.
7. T. Burd, *Low-Power CMOS Cell Library Design Methodology*, M.S. Thesis, University of California, Berkeley, UCB/ERL M94/89, 1994.
8. S. Sze, *Physics of Semiconductor Devices*, New York: Wiley, 1981.
9. H. J. M. Veendrick, "Short-Circuit Dissipation of Static CMOS Circuitry and its Impact on the Design of Buffer Circuits," *IEEE Journal of Solid-State Circuits*, Aug. 1984, pp. 468-73.
10. R. Muller and T. Kamins, *Device Electronics for Integrated Circuits*, New York: Wiley, 1986.
11. A. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid State Circuits*, Apr. 1992, pp. 473-84.
12. Advanced RISC Machines, Ltd., *ARM710 Data Sheet*, Technical Document, Dec. 1994.
13. Integrated Device Technology, Inc., *Enhanced Orion 64-Bit RISC Microprocessor*, Data Sheet, Sep. 1995.

14. Intel Corp., *Embedded Ultra-Low Power Intel486™ GX Processor*, SmartDie™ Product Specification, Dec. 1995.
15. A. Stratakos, S. Sanders, and R.W. Brodersen, "A Low-voltage CMOS DC-DC Converter for Portable Battery-operated Systems," *Proceedings of the Twenty-Fifth IEEE Power Electronics Specialist Conference*, June 1994, pp. 619-26.
16. J. Bunda, et. al., "16-Bit vs. 32-Bit Instructions for Pipelined Architectures," *Proceedings of the 20th International Symposium on Computer Architecture*, May 1993, pp. 237-46.
17. Advanced RISC Machines, Ltd., *Introduction to Thumb*, Developer Technical Document, Mar. 1995.
18. J. Bunda, W.C. Athas, and D. Fussell, "Evaluating Power Implications of CMOS Microprocessor Design Decisions," *Proceedings of the International Workshop on Low Power Design*, Apr. 1994, pp. 147-52.
19. P. Freet, "The SH Microprocessor: 16-Bit Fixed Length Instruction Set Provides Better Power and Die Size," *Proceedings of the Thirty-Ninth IEEE Computer Society International Conference*, Mar. 1994, pp. 486-8.
20. T. Burd, B. Peters, *A Power Analysis of a Microprocessor: A Study of an Implementation of the MIPS R3000 Architecture*, ERL Technical Report, University of California, Berkeley, 1994.
21. J. Montanaro, et. al., "A 160MHz 32b 0.5W CMOS RISC Microprocessor," *Proceedings of the Thirty-Ninth IEEE International Solid-State Circuits Conference - Slide Supplement*, Feb. 1996, pp. 170-1.
22. J. Bunda, *Instruction-Processing Optimization Techniques for VLSI Microprocessors*, Ph.D. Thesis, The University of Texas at Austin, 1993.
23. R. Gonzalez and M. Horowitz, "Energy Dissipation in General Purpose Processors," *Proceedings of the IEEE Symposium on Low Power Electronics*, Oct. 1995, pp. 12-3.
24. D. Wall, *Limits of Instruction-Level Parallelism*, DEC WRL Research Report 93/6, Nov. 1993.
25. M. Johnson, *Superscalar Microprocessor Design*, Englewood, NJ: Prentice Hall, 1990.
26. M. Smith, M. Johnson, and M. Horowitz, "Limits on Multiple Issue Instruction," *Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems*, Apr. 1989, pp 290-302.
27. N. Jouppi and D. Wall, "Available Instruction-Level Parallelism for Superscalar and Superpipelined Machines," *Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems*, Apr. 1989, pp 272-82.
28. P. Lowney, et. al., "The Multiflow Trace Scheduling Compiler," *The Journal of Supercomputing*, Vol. 7, Boston: Kluwer Academic Publishers, 1993, pp. 51-142.
29. R. Panwar and D. Rennels, "Reducing the Frequency of Tag Compares for Low Power I-Cache Design," *Proceedings of the International Symposium on Low Power Design*, Apr. 1995, pp. 57-62.
30. S. Gary, et. al., "The PowerPC 603 Microprocessor: A Low-Power Design for Portable Applications," *Proceedings of the Thirty-Ninth IEEE Computer Society International Conference*, Mar. 1994, pp. 307-15.
31. A. Chandrakasan, *Low Power Digital CMOS Design*, Boston: Kluwer Academic Publishers, 1995.
32. C. Nagendra, et.al., "A Comparison of the Power-Delay Characteristics of CMOS Adders," *Proceedings of the International Workshop on Low Power Design*, Apr. 1994, pp. 231-6.
33. T. Callaway and E. Swartzlander, "Optimizing Arithmetic Elements for Signal Processing," *VLSI Signal Processing*, Vol. 5, New York: IEEE Special Publications, 1992, pp. 91-100.
34. K. Chu and D. Pulfrey, "A Comparison of CMOS Circuit Techniques: Differential Cascode Voltage Switch Logic Versus Conventional Logic," *IEEE Journal of Solid State Circuits*, Aug. 1987, pp. 528-32.
35. T. Biggs, et. al., "A 1 Watt 68040-Compatible Microprocessor," *Proceedings of the IEEE Symposium on Low Power Electronics*, Oct. 1994, pp. 8-11.
36. J. Lundberg, et. al., "A 15-150 MHz All-Digital Phase-Locked Loop with 50-Cycle Lock Time for High-Performance Low-Power Microprocessors," *Proceedings of the Symposium on VLSI Circuits*, June 1994, pp. 35-6.
37. J. Lorch, *A Complete Picture of the Energy Consumption of a Portable Computer*, M.S. Thesis, University of California, Berkeley, 1995.
38. A. Chandrakasan, M. Srivastava, and R.W. Brodersen, "Energy Efficient Programmable Computation," *Proceedings of the Seventh International Conference on VLSI Design*, Jan. 1994, pp. 261-64.
39. Intel Corp. and Microsoft Corp., *Advanced Power Management (APM): BIOS Interface Specification*, Technical Document, Feb. 1996.

40. M. Wieser, et. al., "Scheduling for Reduced CPU Energy," *Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation*, Nov. 1994, pp. 13-23.
41. K. Govil, E. Chan, and H. Wasserman, "Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU," *Proceedings 1st ACM International Conference on Mobile Computing and Networking*, Nov. 1995, pp. 13-25.
42. V. Tiwari, et. al., "Instruction Level Power Analysis and Optimization of Software," *Journal of VLSI Signal Processing*, this issue.