Energy Efficient CMOS Microprocessor Design

Thomas D. Burd and Robert W. Brodersen

University of California, Berkeley

Abstract

Reduction of power dissipation in microprocessor design is becoming a key design constraint. This is motivated not only by portable electronics, in which battery weight and size is critical, but by heat dissipation issues in larger desktop and parallel machines as well. By identifying the major modes of computation of these processors and by proposing figures of merit for each of these modes, a power analysis methodology is developed. It allows the energy efficiency of various architectures to be quantified, and provides techniques for either individually optimizing or trading off throughput and energy consumption. The methodology is then used to qualify three important design principles for energy efficient microprocessor design.

1: Introduction

Throughput and area have been the main forces driving microprocessor design, but recently the explosive growth in portable electronics has forced a shift in these design optimizations toward more power conscious solutions. Even for desktop units and large computing machines, the cost of removing the generated heat and the drive towards "green" computers are making power reduction a priority.

An energy-efficient design methodology has been developed for signal processing applications, resulting in a strategy to provide orders of magnitude of power reduction [1]. These applications have a fixed throughput requirement due to a real-time constraint given by the application (e.g. video compression, speech recognition). Microprocessors targeted for general purpose computing, however, generally operate in one of two other computing modes. Either they are continuously providing useful computation, in which case maximum throughput is desired, or they are in a user interactive mode, in which case bursts of computation are desired.

A framework for an energy-efficient design methodology more suitable for a microprocessor's two operating modes will be presented. Using simple analytic models for delay and power in CMOS circuits, metrics of energy efficiency for the above modes of operation will be developed and their implications on processor design will be presented. This paper will conclude with the application of these metrics to quantify three important principles of energy-efficient microprocessor design.

2: CMOS Circuit Models

Power dissipation and circuit delays for CMOS circuits can be accurately modelled with simple equations, even for complex microprocessor circuits. These models are dependent upon six variables which an IC designer may control to either individually minimize or trade off power and speed. These models hold only for digital CMOS circuits, and are thus not applicable to bipolar or BiCMOS circuits.

2.1: Power Dissipation

CMOS circuits have both static and dynamic power dissipation. Static power arises from bias and leakage currents. While statically-biased gates are usually found in a few specialized circuits such as PLAs, their use has been dramatically reduced in CMOS design. Furthermore, careful design of these gates generally makes their power contribution negligible in circuits that do use them [2]. Leakage currents from reverse-biased diodes of MOS transistors, and from MOS subthreshold conduction [3] also dissipate static power, but are insignificant in most designs.

The dominant component of power dissipation in CMOS is therefore dynamic, and arises from the charging and discharging of the circuit node capacitances found on the output of every logic gate. This capacitance, C_L, can be expressed as:

$$C_L = C_W + C_{FIX} \qquad \text{(EQ 1)}$$

1060-3425/95 $4.00 © 1995 IEEE

C_W is the product of a technology constant and the device width, W, over which the designer has control. C_W is composed of the subsequent gates' input capacitance and part of the diffusion capacitance on the gate output. C_{FIX} is composed of the remaining part of the diffusion capacitance which is purely technology dependent, and the capacitance of the wires interconnecting these gates which may be minimized by efficient layout.

For every low-to-high logic transition in a digital circuit, C_L incurs a voltage change ΔV, 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 α_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:

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

Aside from the memory bitlines in CMOS circuits, most nodes swing a ΔV from ground to V_{DD}, so that the power equation can be simplified to:

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

where the effective switched capacitance, C_{EFF}, is commonly expressed as the product of the physical capacitance C_L, and the activity weighting factor α, 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 [4] and will be omitted in further analyses.

2.2: Circuit Delay

To fully utilize its hardware, a digital circuit's clock frequency, f_{CLK}, should be operated at the maximum allowable frequency. This maximum frequency is just the inverse of the delay of the processor's critical path. Thus, the circuit's throughput is proportional to 1/delay.

Until recently, the long-channel delay model (in which device current is proportional to the square of the supply voltage) suitably modelled delays in CMOS circuits [5]. However, scaling the minimum device channel length, L_{MIN}, to below 1 micron (which is common in today's process technology), degrades 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 [6].

$$Delay \cong \frac{C_L}{I_{AVE}} \cdot \frac{V_{DD}}{2} \cong \frac{C_L \cdot V_{DD}}{k_V \cdot W \cdot (V_{DD} - V_T - V_{DSAT})} \qquad \text{(EQ 4)}$$

The change in performance can be analytically characterized by what is known as 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 W, the technology constant k_V, and to first-order, V_{DD}. V_T is the threshold voltage (typically 0.5 - 1.0 volts) and is the minimum V_{DD} for which the device can still operate. 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.

2.3: Circuit Design Optimizations

The designer can minimize some of the variables in Equations 3 and 4 to either individually or simultaneously minimize the delay and power dissipation. This can be accomplished by minimizing C_{EFF}, while keeping W constant. The following three methods for minimizing C_{EFF} can be correlated so they may not always be individually optimized.

First, the switching frequency, α, can be minimized to reduce power dissipation without affecting the delay. This optimization shows the best promise for significant power reduction. The α factor can be minimized by a number of techniques such as dynamically gating the clock to unused processor sections and selecting cell/module topologies that minimize switching activity.

Another approach is to minimize the number of nodes, which will minimize the total physical capacitance, and likewise reduce the power. However, this method may come at the expense of computation per cycle, as in the example of reducing a 32-bit datapath to a 16-bit datapath.

The third approach is to reduce C_{FIX} by minimizing the interconnect capacitance, which optimizes both power and delay. Since speed has always been a primary design goal, this is already done as general practice.

Although it beyond the control of the designer, it is worth noting the impact of technology scaling on power and delay. Capacitances scale down linearly with technology parameter L_{MIN} while transistor current stays approximately constant if V_{DD} remains fixed (constant-voltage technology scaling). Thus, delay scales linearly with L_{MIN} for constant power; likewise, power scales linearly with L_{MIN} for constant delay. Essentially, technology scaling is always beneficial.

2.4: Trading off Delay and Power

The remaining three variables under the designer's control can only be used to trade-off delay and power. As the voltage is reduced, the delay increases hyperbolically as the supply voltage approaches V_T. Meanwhile, the power drops due to the product of the squared voltage term and the frequency (inverse of the delay) term. Thus, by operating at various values of supply voltage, a given processor architecture can be made to cover a large range of operating points as demonstrated in Figure 1.

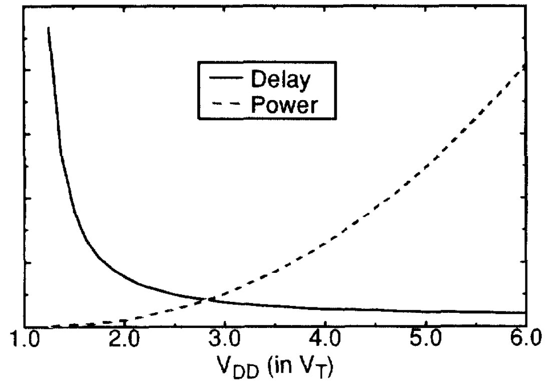


Fig 1: Delay, Power versus V_{DD}

Another method for trading off delay and power is to vary the device width W. However, the trade-off is dependent on how large C_{FIX} is with respect to C_W. Empirical data shows these capacitances are typically on the order of the same size for transistors with minimum width. As the width is increased above the minimum value, the power increases but the delay is decreased. However, C_W soon dominates C_{FIX}, and then power scales linearly with width, while the delay remains approximately constant. Thus, there is a delay-power trade-off, but only over a small range of delay values.

Scaling the clock frequency is a third approach which is most beneficial if it is coupled with voltage scaling. If the clock frequency is reduced, the delay may be increased (keeping it equal to $1/f_{CLK}$) by reducing the supply voltage and thus saving power. If the voltage is kept constant, then power and throughput reduce linearly with clock frequency.

3: Energy Efficiency

No single metric quantifies energy efficiency for all digital systems. The metric is dependent on the type of computation the system performs. 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.

Throughput, T, is 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 proportional to the amount of concurrency per clock cycle (i.e. number of parallel operations) divided by the delay, or equivalently:

$$Throughput \equiv T = \frac{Operations}{Second} \propto \frac{Concurrency}{Delay} \qquad \text{(EQ 5)}$$

Operations are the units of computation at the highest level of the design space. Valid measures of throughput are MIPS (instructions/sec) and SPECint92 (programs/sec) which compare the throughput on implementations of the same instruction set architecture (ISA), and different ISAs, respectively.

3.1: Fixed Throughput Mode

Most real-time systems require a fixed number of operations per second. Any excess throughput cannot be utilized, and therefore needlessly dissipates power. 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).

$$Metric|_{FIX} = \frac{Power}{Throughput} = \frac{Energy}{Operation} \qquad \text{(EQ 6)}$$

Previous work has shown that the metric of energy efficiency in Equation 6 is valid for the fixed throughput mode of computation [5]. 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; equivalently, its power dissipation has been halved. Since throughput is fixed, minimizing the power dissipation is equivalent to minimizing the energy/operation.

3.2: Maximum Throughput Mode

In most multi-user systems, primarily networked desktop computers and supercomputers, the processor is continuously running. The faster the processor can perform computation, the better. This is the defining characteristic of the maximum throughput mode of computation. Thus, this mode's metric of energy efficiency must balance the need for low power and high speed.

$$Metric|_{MAX} = ETR = \frac{E_{MAX}}{T_{MAX}} = \frac{Power}{Throughput^2} \qquad \text{(EQ 7)}$$

A good metric for measuring energy efficiency for this mode is given in Equation 7, henceforth called the Energy to Throughput Ratio, or ETR (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/

Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS'95)
1060-3425/95 $10.00 © 1995 IEEE

operation, satisfying the need to equally optimize throughput and power dissipation. Thus, a lower ETR represents a more energy efficient solution. The Energy-Delay Product [7] is a similar metric, but does not include the effects of architectural parallelism when the delay is taken to be the critical path delay.

In most circuits, however, ETR is not constant for different values of throughput. The throughput can be adjusted with the delay-power trade offs shown in Section 2.4; but, unfortunately, none of the methods perform linear trade-offs between energy/operation and throughput, and only V_{DD} allows the throughput to be adjusted across a reasonable dynamic range.

Because energy/operation is independent of clock frequency, the clock should never be scaled down in maximum throughput mode; only the throughput scales with clock frequency, so the ETR actually increases. Increasing all device widths will only marginally decrease delay while linearly increasing energy. Generally, it is optimal to set all devices to minimum size and only size up those lying in the critical paths.

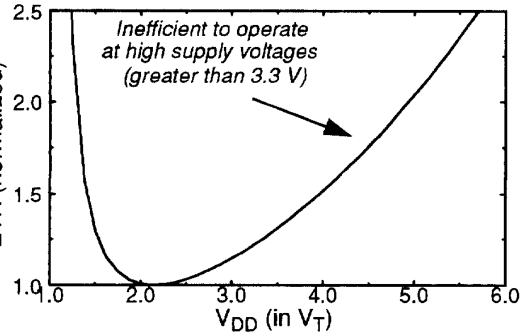


Fig 2: ETR as a function of V_{DD}.

As shown in Figure 2, 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.

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 most digital CMOS designs) to the maximum voltage (3.3V- 5V), while energy/operation and delay are measured. The energy/operation can then be plotted as a function of delay, and the architecture is completely characterized over all possible throughput values.

Using the ETR metric is equivalent to making a linear approximation to the actual energy/operation versus throughput curve. Figure 3 demonstrates the error incurred in using a constant ETR metric. 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.

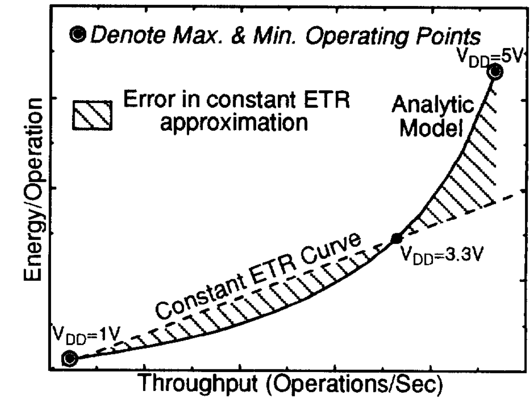


Fig 3: Energy vs. Throughput metric.

3.3: Burst Throughput Mode

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

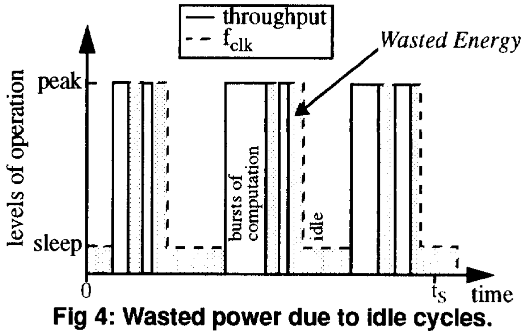


Fig 4: Wasted power due to idle cycles.

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 power dissipated while idling has been eliminated. However, this is not realistic in practice. Many processors do not having a power saving mode and those that do so generally support only simple clock reduction/deactivation modes. The hypothetical example depicted in Figure 4 contains a clock reduction (sleep) mode in which major sections of the processor are powered down. The shaded area indicates the processor's idle cycles in which power is needlessly dissipated, and whose magnitude is dependent upon whether the processor is operating in the "low-power" mode.

$$E_{MAX} = \frac{Total\ Energy\ Consumed\ Computing}{Total\ Operations} \qquad \text{(EQ 8)}$$

$$E_{IDLE} = \frac{Total\ Energy\ Consumed\ Idling}{Total\ Operations} \qquad \text{(EQ 9)}$$

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 3.2), and T_{AVE} is the time-average throughput (total operations / t_S). If the time period t_S characterizes the computing demands of the user and/or target system environment (T_{AVE}), then a good metric of energy efficiency for the burst throughput mode is:

$$Metric|_{BURST} = METR = \frac{E_{MAX} + E_{IDLE}}{T_{MAX}} \qquad \text{(EQ 10)}$$

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 through Equation 8 by t_S•(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 3.2. 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 power dissipation is significant.

Equation 9 can be rewritten as:

$$E_{IDLE} = \frac{[Idle\ Power\ Dissipation] \cdot [Time\ Idling]}{[Average\ Throughput] \cdot [Sample\ Time]} \qquad \text{(EQ 11)}$$

With the Power-Down Efficiency, β, is defined as:

$$\beta = \frac{Power\ dissipation\ while\ idling}{Power\ dissipation\ while\ computing} = \frac{P_{IDLE}}{P_{MAX}} \qquad \text{(EQ 12)}$$

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

$$E_{IDLE} = \frac{[\beta \cdot E_{MAX} \cdot T_{MAX}] \cdot [(1 - T_{AVE}/T_{MAX}) \cdot t_S]}{[T_{AVE}] \cdot [t_S]} \qquad \text{(EQ 13)}$$

Equation 14 shows that idle power dissipation dominates total power dissipation when the fractional time spent computing (T_{AVE}/T_{MAX}) is less than the fractional power dissipation while idling (β).

$$METR = ETR\left[1 + \beta\left(\frac{T}{T_{AVE}} - 1\right)\right], \quad T \geq T_{AVE} \qquad \text{(EQ 14)}$$

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

Idle Energy Consumption is Negligible ($\beta << T_{AVE}/T_{MAX}$): The metric should simplify to that found in the maximum throughput mode, since it is only during the bursts of computation that power is dissipated and operations performed. For negligible power dissipation during idle, the METR metric in Equation 14 degenerates to the ETR, as expected. Likewise, for perfect power-down (β=0) and minimal throughput (T_{MAX}=T_{AVE}), the METR is exactly the ETR.

Idle Energy Consumption Dominates ($\beta >> T_{AVE}/T_{MAX}$): 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/operation constant. While it might be expected that these are independent optimizations, E_{IDLE} may be related back to E_{MAX} and the throughput by β (T_{AVE} is fixed):

$$\frac{E_{IDLE}}{E_{MAX}} \cong \frac{P_{IDLE}/T_{AVE}}{P_{MAX}/T_{MAX}} = \beta \cdot \frac{T_{MAX}}{T_{AVE}} \qquad \text{(EQ 15)}$$

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

$$METR \cong \frac{\beta \cdot E_{MAX}}{T_{AVE}}, \quad \text{(Idle Energy Dominates)} \qquad \text{(EQ 16)}$$

If β remains constant for varying throughput (and E_{MAX} stays constant), then E_{IDLE} scales with throughput as shown in Equation 15. Thus, the METR becomes an energy/operation minimization similar to the fixed throughput mode. However, β may vary with throughput, as will be analyzed further in Section 4.4.

4: Design Optimizations

Many energy efficiency optimization techniques developed for the fixed throughput mode of computation are applicable to the a microprocessor operating in either the maximum or burst throughput modes though not always yielding equal gains. Those techniques that successfully apply to microprocessors are outlined below

If processors are compared without a target system's requirements in mind, then ETR is a reasonable metric of

1060-3425/95 $10.00 © 1995 IEEE

comparison. However, if the targeted system resides in the single-user domain and the required average operations/ second T_{AVE} can be characterized, then METR is a better metric of comparison.

4.1: Fixed Throughput Optimization

Orders of magnitude of power reduction have been achieved in fixed throughput designs by optimizing at all levels of the design hierarchy, including circuit implementation, architecture design, and algorithmic decisions [1].

One such example is a video decompression system in which a decompressed NTSC-standard video stream is displayed at 30 frames/sec on a 4" active matrix color LCD. The entire implementation consists of four custom chips that consume less than 2mW [1].

There were three major design optimizations responsible for the power reduction. First, the algorithm was chosen to be vector quantization which requires fewer computations for decompression than other compression schemes, such as MPEG. Second, a parallel architecture was utilized, enabling the voltage to be dropped from 5V to 1.1V while still maintaining the throughput required for the real-time constraint imposed by the 30 ms display rate of the LCD. This reduced the power dissipation by a factor of 20. The reduction in clock rate compensated for the increased capacitance; thus, there was no power penalty due to the increased capacitance to make the architecture more parallel (though it did consume more silicon area). Third, transistor-level optimizations yielded a significant power reduction.

4.2: Fixed vs. Max Throughput Optimization

Since the fixed throughput mode is a degenerate case of the max throughput mode, the low-power design techniques used in the fixed throughput mode are also applicable to the maximum throughput mode. This is best visualized by mapping the procedure for exploiting parallelism onto the Energy/operation vs. Throughput plot. There is a two-step process to exploit parallelism for the fixed throughput mode as shown in Figure 5.

Step 1: Ideally, doubling the hardware (arch1->arch2), doubles the throughput. Although the capacitance is doubled, the energy per operation remains constant, because two operations are completed per cycle

Step 2: Reduce the voltage to trade-off the excess speed and achieve the original required throughout. With this parallel architecture, the clock frequency is halved, but the throughput remains constant with respect to the original design.

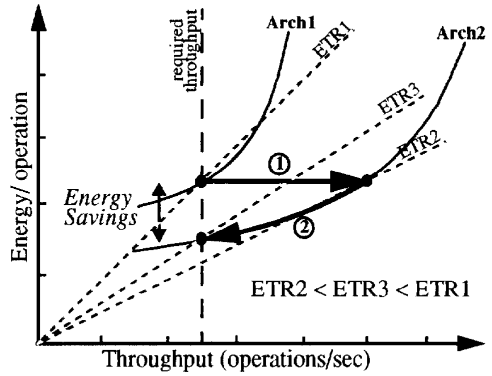


Fig 5: Energy minimization for fixed throughput.

For a processor operating in the maximum throughput mode of computation, it is favorable to reduce the ETR as much as possible, thus, only step 1 is required. In the above example, the final efficiency may even be reduced by decreasing the voltage as was done in step 2.

The energy-throughput trade-off of the maximum throughput operation allows one processor to address separate market segments that have different throughput and power dissipation requirements by simply varying the value of the supply voltage. In essence, a high-speed processor may be the most energy efficient solution for a low throughput application, if the appropriate supply value is chosen. Unfortunately, there are practical bounds that limit the range of operation (e.g. the minimum and maximum supply voltages), preventing one processor from spanning all possible values of throughput.

4.3: Maximum Throughput Optimization

Simply scaling voltage and device sizes are methods of trading off throughput and energy; but they are not energy efficient if used alone since they do not reduce the ETR. However, the architectural modifications that allowed the voltage to be reduced in the fixed throughput mode, do reduce ETR. Conversely, clock frequency reduction, as is common in many portable computers today, does not decrease ETR (it actually increases it) as shown in Section 3.2).

There are three levels in the digital IC design hierarchy. Energy efficient design techniques can drastically increase energy efficiency at all levels as outlined below. Many techniques have corollaries between the fixed and maximum throughput modes, although some differences arise, as noted.

Algorithmic Level: While the algorithm is generally implemented in the software/code domain, which is removed from processor design, it is important to under-stand the efficiency gains achievable. Similar gains are possible in hardware-implemented algorithms found in signal processing applications.

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 [7]. 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. What needs to be minimized is the number of primitive operations: memory operations, ALU operations, etc. In the case of RISC architectures, the machine code closely resembles the primitive operations, making this optimization possible by minimizing path length. However, in CISC architectures, the primitive operations per each machine instruction need to be evaluated, rather than just comparing path length.

The design of the ISA, however, which does impact the hardware, may be optimized for energy efficient operation. Each instruction must be evaluated to determine if it is more efficient to implement it in hardware, or to emulate it in software.

Architectural Level: The predominant technique to increase energy efficiency 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 [5]. This assumes that the algorithm is fully vectorizable, and that N is not excessively large.

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. 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 [8][9]. Therefore, the achievable speedup, S, will be less than the number of simultaneous issuable instructions, and yields diminishing returns as the peak issue rate is increased. S has been shown to be between two and three for practical hardware implementations in current technology [10].

If the code is 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. Even in statically scheduled architectures such as VLIW processors, there will be extra capacitive overhead due to branch prediction, bypassing, 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 unissued instructions is amortized over those instructions that are issued, further increasing C_{EFF}.

The energy efficiency increase can be analytically modelled. Equation 17 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: $C_{EFF}^{C0} << C_{EFF}^{D0,M0}$ [11].

$$ETR|_{RATIO} = \frac{S\left(C_{EFF}^{D0} + C_{EFF}^{M0}\right)}{\left(C_{EFF}^{C1} + C_{EFF}^{D1} + C_{EFF}^{M1}\right)} \qquad \text{(EQ 17)}$$

Whether ILP architecture techniques can yield significant energy efficiency improvement is not inherently clear. The presence of C_{EFF}^{C1} due to control overhead, and increase of C_{EFF}^{M1} (with respect to C_{EFF}^{M0}) due to unissued instructions, may even negate the increase due to S. Current investigation is attempting to quantify these terms and the resulting efficiency increase for a a variety of superscalar implementations.

Other aspects of architecture design can be optimized for improved efficiency. One example is the reduction of extraneous switching activity by gating the clock to various parts of the processor when possible [12]. Another example is the minimization of the lengths of the most active busses.

Circuit level: The design techniques implemented at this level are similar for the fixed and maximum throughput modes. For example, the topologies for the various subcells (e.g. ALU, register file, etc.) should be selected by their ETR, and not solely for speed. Low-swing bus drivers are currently being investigate for high-speed operation; but, these drivers are also applicable to low power design because the energy per transition drops linearly with the voltage swing (ΔV is reduced, not V_{DD}).

The basic transistor sizing methodology is to reduce every transistor not in the critical path to minimum size to minimize C_{EFF}. There are a number of other techniques to minimize effective capacitance which are also viable for optimizing energy efficiency [1].

4.4: Burst Throughput Optimization

If the energy consumed while idling is negligible compared to that consumed during bursts of computation, then the METR metric simplifies to the ETR metric, and all the design optimizations to increase energy efficiency in the maximum throughput mode are equivalently valid in the burst throughput mode.

However, if the energy consumption during idle dominates the total energy consumption, then different optimizations are required. As was shown in Equation 14, this occurs when the fractional time spent computing (T_{AVE}/T_{MAX}) is less than the fractional power dissipation while idling (β). Then the METR optimization is to minimize β and E_{MAX}, as seen from Equation 16. Furthermore, the exact optimization depends on whether β changes as the throughput T is varied as shown below.

β is independent of throughput: This case generally applies to processors with no power-down mode for idle periods; if the clock frequency remains the same (or proportional) during both computation and idle periods, then the idle power dissipation tracks the compute power dissipation. If the throughput is now increased while E_{MAX} is held constant (e.g. using parallelism, as was shown in Figure 5 to decrease the ETR), the METR remains constant because the increase in compute power dissipation causes the idle power dissipation to increase as well.

The METR can be optimized by minimizing E_{MAX}, similar to the fixed throughput mode optimization. By reducing V_{DD} to scale down both throughput and E_{MAX}, the processor's efficiency can be maximized. The efficiency will keep improving as V_{DD} is reduced until $E_{IDLE} < E_{MAX}$, and the energy consumption during idle is not dominant anymore; decreasing V_{DD} any further will have little effect on the energy efficiency.

β varies with throughput: This is common for processors that implement idle power down modes; the power dissipation during idle is not proportional to the power dissipation during computation. So, for constant idle power dissipation, or equivalently, constant E_{IDLE}, it is most efficient to deliver as much throughput as possible, since any increase in computational power dissipation is negligible compared to the total power dissipation. In practice, β will be less than inversely proportional to throughput (e.g. due to latency switching between operating modes) so that E_{IDLE} is not independent of throughput. However, energy efficiency will continue to increase with throughput until idle power dissipation is no longer dominant.

By itself, the processor hardware can only provide a moderate value of β, which is the ratio of the power dissipated executing a nop instruction to the power dissipated executing a typical instruction. While executing nop instructions, the internal state of the controller and datapath is not changing ($\alpha = 0$), but the clock line is still transitioning every cycle. The processor's clock dissipates a sizable fraction of the total power, anywhere from 10% to 50%. Even if the clock is gated to those pipeline sections executing nop instructions, the instruction-memory access per cycle will continue to dissipate power. If the processor is used in a laptop computer, and T_{AVE} is on the order of 1 SPECint92 (high estimate for user's average operations/second) and β is reasonably estimated as 0.2, it is not energy efficient to increase the peak throughput of the processor beyond 5 SPECint92. Thus, to deliver a more tolerable response time to the user, energy efficiency will have to be degraded.

It is imperative that the operating system intervene to provide further reductions in β. In doing so, β will typically become a function of throughput because the operating system can decouple the compute and idle regimes' power dissipation. The hardware can enable software power down modes by providing instructions to halt either parts of the processor or the entire thing, as is becoming common in embedded microprocessors.

Independent of β's relation to throughput, the METR metric indicates poor energy efficiency whenever the energy/operation consumed idling dominates total energy consumption. If β is constant, the energy efficiency is maximized by reducing E_{MAX} through V_{DD} reduction, until the throughput is roughly T_{AVE}/β. If β is inversely dependent on throughput, then the energy efficiency is maximized by increasing the throughput, and possibly V_{DD}, until E_{MAX} is roughly equal to E_{IDLE}.

5: Design Principles

A few examples are presented below to demonstrate how energy efficiency can be properly quantified. In doing so, three design principles follow from the optimization of the previously defined metrics: a high-performance processor is usually an energy-efficient processor; reducing the clock frequency does not increase the energy efficiency; and lastly, idle power dissipation limits the efficiency of increasing deliverable throughput.

5.1: High Performance is Energy Efficient

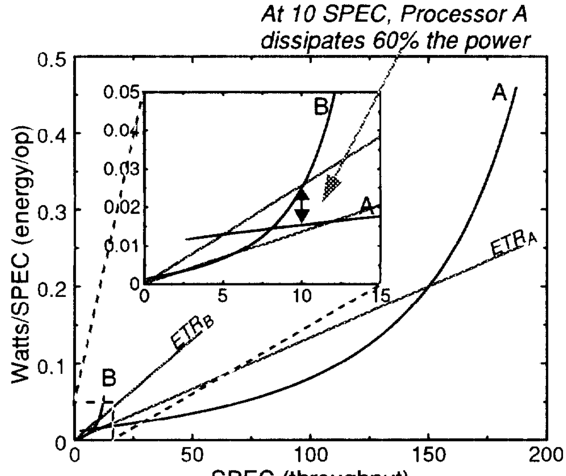
Table 1 lists two hypothetical processors that are similar to ones available today -- B targets the low-power market, and A targets the high-end market; both are fabricated in the same technology, which allows an equal comparison. SPEC is either SPECint92, or if a floating point unit is present, the average of SPECint92 and SPECfp92 [7]. A misused metric for measuring energy efficiency is SPEC/Watt (or Dhrystones/Watt, MIPS/Watt, etc.). Processor B may boast a SPEC/Watt eight times greater than A's, and declare that it is eight times as energy efficient. This metric only compares operations/energy, and does not weight that B has 1/15th the performance.

Table 1: Comparison of two processors

Proc.	SPEC (T_{MAX})	Watts	V_{DD}	SPEC/Watt ($1/E_{MAX}$)	ETR (10^{-3})
A	150	30.0	3.3	5.0	1.33
B	10	0.25	3.3	40.0	2.50

The ETR (Watts/SPEC^2) metric indicates that processor A is actually *more* energy efficient than processor B. To quantify the efficiency increase, the plot of energy/operation versus throughput in Figure 6 is used because it better tracks processor A's energy at the low throughput values. The plot was generated from the delay and power models in Section 2.

According to the plot, processor A would dissipate 0.154 W at 10 SPEC, or 60% of processor B's power, despite the low V_{DD} ($1.31V_T$) for A. Conversely, A can deliver 31 SPEC at 0.25W ($V_{DD}=1.66V_T$), or 310% of B's throughput. This does assume that processor A can be operated at this supply voltage.



At 10 SPEC, Processor A dissipates 60% the power

Fig 6: Energy vs Throughput of Processors A & B.

While the ETR correctly predicted the more energy-efficient processor at 10 SPEC, it is important to note that processor A is not more energy efficient for all values of SPEC, 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 8 SPEC. Below this value, processor B is more energy efficient.

5.2: Clock Reduction is not Energy Efficient

A common fallacy is that reducing the clock frequency f_{CLK} is energy efficient. In maximum throughput mode, it is quite the opposite. At best, it allows an energy-throughput trade-off when idle energy consumption is dominant in burst throughput mode.

In the maximum throughput mode, energy is independent of f_{CLK}, so as the latter is scaled down, the throughput decreases, and the ETR increases, indicating a less efficient design. If f_{CLK} is halved, the power is also halved. However, it takes twice as long to complete any computation, so the energy/operation consumed is constant. Thus, if the energy source is a battery, halving f_{CLK} is equivalent to doubling the computation time, while maintaining constant computation per battery life.

In the burst throughput mode, clock reduction may trade-off throughput and operations per battery life (i.e. energy/operation), but only when E_{IDLE} dominates total energy consumption ($\beta >> T_{AVE}/T_{MAX}$) and β 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 user is engaged in an application where throughput degradation is acceptable, then this is a reasonable trade-off. If either E_{MAX} dominates total energy consumption, or β is inversely proportional to throughput, then reducing f_{CLK} does not affect the total energy consumption, and the energy efficiency drops.

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 3.2) 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 may yield a more energy efficient solution.

5.3: Faster Operation Can Limit 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 10 SPEC, and both processor A and B have a β factor of 0.1. If the processors' V_{DD} is left at 3.3V, B's METR is exactly equal to its ETR value, which is $2.5x10^{-3}$. It remains the same because it never idles. Processor A, on the other hand, spends 14/15ths ($1 - T_{AVE}/T_{MAX}$) of the time idling, and its METR is $3.2x10^{-3}$. Thus, for this scenario, processor B is more energy efficient.

However, if processor A's β can be reduced down to

0.05, then the METR of processor A becomes 2.26×10^{-3}, and it is once again the more energy efficient solution. For this example, the cross-over value of β is 0.063.

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). For the above example, a β for processor A greater than 0.063 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 processor A so that it is more energy efficient for any required throughput. This is true if the dynamic range of processor A is as indicated in Figure 6. However, if some internal logic limited the value that V_{DD} could be dropped, then the lower bound on A's throughput would be located at a much higher value. Thus, finite β can degrade the energy efficiency of high throughput circuits due to excessive idle power dissipation.

6: Conclusions

Metrics for energy efficiency have been defined for three modes of computation in digital circuits. The appropriate metric for the fixed throughput mode, typical of most digital signal processing circuits, is energy/operation. Two other modes which apply to the operation of a microprocessor are maximum throughput and burst throughput modes.

A good energy efficiency metric for the maximum throughput mode is energy/operation versus throughput, which can be approximated with a constant ETR value. Many of the techniques developed for low power design in the fixed throughput mode can be successfully applied to the energy efficient design of a processor in the maximum throughput mode.

However, a better metric to describe more typical processor usage is the Microprocessor ETR, or METR; it includes the energy consumption of the idle mode, which can dominate total energy consumption in user-interactive applications. Decreasing the energy consumption of the idle mode is critical to the design of a 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.

An organized analytical approach to the optimization of power in microprocessor design, based on metrics that include the requirement of both throughput and energy, as well as actual application operation, allow the designer to quantify energy efficiency and provide insights into the design issues of energy efficient processor design.

This research is sponsored by ARPA.

References

[1] A. Chandrakasan, A. Burstein, R.W. Brodersen, "A Low Power Chipset for Portable Multimedia Applications", *Proceedings of the IEEE International Solid-State Circuits Conference*, Feb. 1994, pp. 82-83.

[2] T. Burd, *Low-Power CMOS Cell Library Design Methodology*, M.S. Thesis, University of California, Berkeley, 1994.

[3] S. Sze, *Physics of Semiconductor Devices*, Wiley, New York, 1981.

[4] H. Veendrick, "Short-Circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits", *IEEE Jour. of Solid State Circuits*, Aug 1984, pp. 468-473

[5] A. Chandrakasan, S. Sheng, R.W. Brodersen, "Low-Power CMOS Digital Design", *IEEE Journal of Solid State Circuits*, Apr. 1992, pp. 473-484.

[6] R. Muller, T. Kamins, *Device Electronics for Integrated Circuits*, Wiley, New York, 1986

[7] M. Horowitz, T. Indermaur, R. Gonzalez, "Low-Power Digital Design", *Proceedings of the Symposium on Low Power Electronics*, Oct. 1994.

[8] D. Wall, *Limits of Instruction-Level Parallelism*, DEC WRL Research Report 93/6, Nov. 1993.

[9] M. Johnson, *Superscalar Microprocessor Design*, Prentice Hall, Englewood, NJ, 1990.

[10] M. Smith, M. Johnson, 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

[11] T. Burd, B. Peters, *A Power Analysis of a Microprocessor: A Study of the MIPS R3000 Architecture*. ERL Technical Report, University of California, Berkeley, 1994.

[12] 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. 307-315