RC23026 (W0312-043) December 10, 2003
Electrical Engineering

IBM Research Report

Power Performance Tradeoffs in Design for SoCs

Victor Zyuban, Philip Strenski
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Power Performance Tradeoffs in Design for SoCs

Victor Zyuban and Philip Strenski

INTRODUCTION

The design and implementation of processor cores is characterized by conflicting requirements of the ever increasing demand for higher performance and, usually, stringent power budget. Thus compromises between performance and power need to be made early in the design cycle. In the design of a processor core there is typically a very specific power budget, but there are several ways to trade power for performance.

At the system level, varying power supply is the most straightforward and well understood method for controlling power. One advantage of this method is that power supply can typically be adjusted within a certain range even after the chip has been manufactured. Also, scaling Vdd around the nominal value in ASIC foundry technologies has a known cost which is “typically” 2% in energy per 1% in performance, although it can be anywhere from 0.5% to more than 3% in energy per 1% in performance. A notion of *voltage intensity* has recently been introduced [18] to quantify power-performance tradeoffs through varying the power supply. Although a very powerful technique, scaling Vdd may have a relatively high performance cost for saving power in a processor core that does not meet its power budget, as shown in Section III of this chapter.

Another method for making power-performance tradeoffs is technology scaling, such as shrinking the oxide thickness and effective channel length. Although, such tradeoffs are not generally available to average ASIC customers, some foundry technologies provide libraries and transistors with multiple threshold voltages, and some high end microprocessor designs work with foundries (typically their own) to engineer these parameters effectively.

At the circuit level, power and performance can be traded by changing transistor sizes and power levels of ASIC cells, controlled by changing transistor tuning targets, or by restructuring logic to increase or decrease

the parallelism in circuits, for example, perform more computations in parallel in order to reduce the critical path. These tradeoffs are controlled by either custom or logic designers or by running synthesis tools with different directives. Although more difficult to quantify, this method for power-performance tradeoffs is at least as powerful as scaling the power supply. By scaling circuits, in "typical" designs that we analyzed, one percent in performance could be traded for from 0.5% to 5% in energy, or even higher, if the frequency target is too aggressive. A concept of *hardware intensity* has been introduced in [18] to quantify power-performance tradeoffs through scaling circuits.

Finally, scaling the processor core microarchitecture and, possibly Instruction Set Architecture (ISA), is one more way for trading power and performance. This method involves changing machine organization, such as pipeline depth, issue width, the set of functional units, bypasses, number of ports and entries in queues and register files, the size of branch predictors, etc. Scaling microarchitecture is even a more powerful method for power-performance tradeoffs than voltage and circuit scaling, but it is more difficult to quantify and optimize, and can only be used at early stages of the design. A concept of *architectural complexity* was introduced in [16] to analyze power-performance tradeoffs at the ISA and microarchitectural levels. It has been demonstrated that *architectural complexity* can not only be measured but also set as a design target [8].

It was demonstrated in [18] that in order to develop an energy-efficient processor core, design decisions at all levels must be balanced in such a way that all forms of spending power, described above, have a similar marginal cost. In the following sections we summarize some of the most important formulas for balancing hardware intensity and power supply voltage derived in [18], and give a graphical interpretation of the major result. Then we discuss in detail the formula for balancing architectural complexity with voltage and hardware intensity, and the iterative process of refining the processor core architecture in the power-performance space. Then, since making power-balanced decisions at the architectural level plays such an important role in the development of the energy-efficient processor cores, we give a derivation of a new form of the architectural energy-efficiency criterion that does not require evaluation of the relative changes in processor frequency. In Section IV we discuss other power-performance metrics that are commonly used

in the architectural community and discuss some common mistakes made by architects when applying these metrics. Section V gives some examples of using the architectural energy-efficiency criterion, and Section VI concludes the chapter.

I. Hardware Intensity

The concept of *hardware intensity* η was introduced in [18] as a quantitative measure of how aggressively the circuits in a processor are tuned to meet a target clock frequency. Hardware intensity shows the energy cost (in %) required to improve the delay D of a hardware macro by 1% through restructuring the logic and retuning the circuits, at a fixed power supply v[1],

$$\eta = -\left.\frac{D\partial E}{E\partial D}\right|_{\text{fixed v}} \quad \text{or} \quad \eta = -\left.\frac{\%E}{\%D}\right|_{\text{through retuning}} \tag{1}$$

Alternatively, we can define the hardware intensity η as a parameter in the cost function for optimizing hardware:

$$F_{\text{cost}}(E,D) = (E/E_0)(D/D_0)^{\eta} \qquad \eta \geq 0, \tag{2}$$

where D is the critical path delay through the circuit, E is the average energy dissipated per cycle, D_0 and E_0 are the corresponding lower bounds that can be achieved through tuning and logic restructuring for a fixed supply voltage. Under a very general assumption that the curvature of the energy-delay curve is higher than the curvature of the contour of the cost function (2) at any point the two touch, $\frac{D^2}{E}\frac{\partial^2 E}{\partial D^2} > \eta(\eta+1)$, we can show that for any power supply voltage v, every point on the energy-delay curve corresponds to a certain value of hardware intensity η, $0 \leq \eta < +\infty$. Then, the energy-delay curve in the energy-versus-delay coordinates can viewed as a parameterized curve: $D = D(\eta, v)$, $E = E(\eta, v)$.

Figure 1 gives a graphical interpretation of the hardware intensity. The solid line plots a typical energy-delay curve for some hardware function. Dotted lines show several contours of the cost function (2), for two values of hardware intensity η. Point (D,E) at which the energy-delay curve tangents the lowest of the

[1]It is assumed that the curvature of the energy-delay curve is sufficiently high, so that $\frac{D^2}{E}\frac{\partial^2 E}{\partial D^2} > \eta(\eta+1)$ is satisfied at every point.

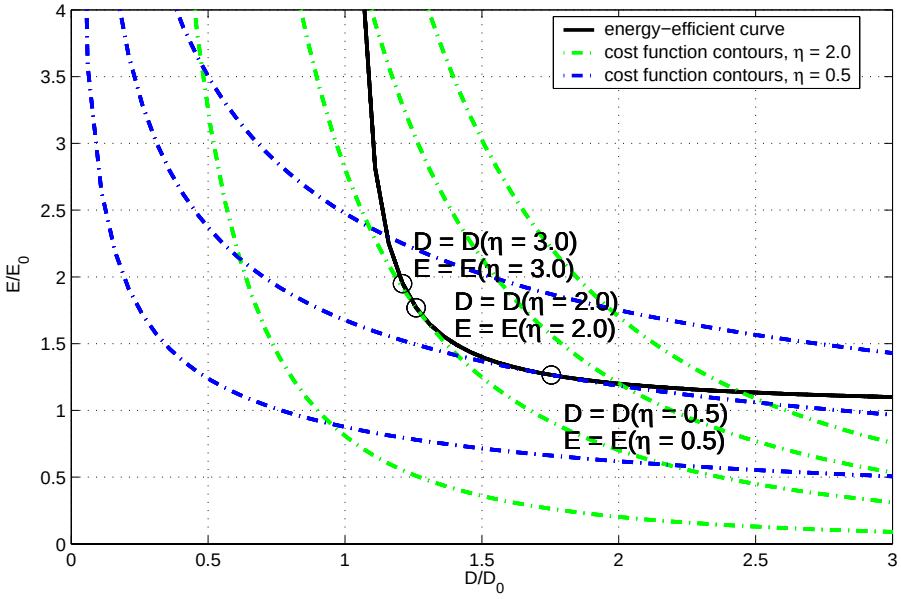

Fig. 1. Typical energy-efficient curve and constant cost function contours for $\eta = 0.5$ and $\eta = 2.0$.

contours ($F_{\text{cost}}(E,D) = A$ with the smallest value of A) corresponds to the implementation for this value of hardware intensity η. Taking advantage of the equivalence of the tangents to the energy-delay curve and the counter of the cost function, we get:

$$\left.\frac{\partial E}{\partial D}\right|_{\text{fixed v}} = \frac{\partial E(\eta,v)}{\partial \eta} \Big/ \frac{\partial D(\eta,v)}{\partial \eta} = -\frac{\partial F_{\text{cost}}}{\partial D} \Big/ \frac{\partial F_{\text{cost}}}{\partial E} = -\eta \frac{E}{D}. \tag{3}$$

This establishes the equivalence of the two definitions of the hardware intensity in (1) and (2).

Then, by formally solving the problem of minimizing the delay as a function of η and v, subject to a constant energy constraint, the following relations were derived in [18] for the optimal balance between hardware intensity and power supply voltage:

$$\text{isolated macro} \qquad \eta = \theta(v) \tag{4}$$

$$\text{composite macro} \qquad \frac{w_j}{u_j}\eta_j = \theta(v) \qquad 1 \le j \le M \tag{5}$$

$$\text{multi-stage pipeline} \qquad \textstyle\sum_i w_i \eta_i = \theta(v) \tag{6}$$

where w_i are energy weights of pipeline stages i in (6), w_j and u_j are energy and delay weights of sub-blocks j in (5), η_j are the hardware intensities in the corresponding sub-blocks, and θ is the *voltage intensity* defined

as

$$\theta = \frac{E_v}{D_v} \qquad E_v = \frac{v}{E}\frac{\partial E}{\partial v} \qquad D_v = -\frac{v}{D}\frac{\partial D}{\partial v}. \tag{7}$$

Relation (4) has a simple interpretation, shown in Fig. 2. The solid curve shows an energy-delay tradeoff curve for variable hardware intensity at a fixed power supply, $v = 1.5V$, $\theta = 2$ in this example (hardware intensity energy-delay curve). The curve was fitted to simulation data [18] for an integer adder, obtained using EinsTuner [5]. The dotted curves show the energy-delay curves for a fixed tuning point (fixed hardware intensity η) of the circuit, but varying power supply, plotted for an ideal $E \sim v^2$ and $f \sim v$ dependence, as commonly assumed in many studies. The dashed curves show *simulated* data (for a set of FUs, running PathMill and PowerMill), with $50mV$ steps in the power supply marked with circles. The point at which the power supply energy-delay curve (dashed curve) tangents the hardware intensity energy-delay curve (solid curve) corresponds to the optimal balance between η and v.

To show this, suppose that the circuit is over-tuned, for example, $\eta = 4$. Then, retuning the circuit for a lower value of η will move the design point down the hardware intensity energy-delay curve. Increasing the power supply to recover the performance will move the design up the power supply energy-delay curve (dashed curve). Since the hardware intensity energy-delay curve (solid curve) is steeper than the power supply energy-delay curve, the same performance will be achieved at a lower energy. Similarly, if the circuit is under-tuned for, say, $\eta = 0.5$, then tuning the circuit for a higher η and then reducing Vdd to achieve the same critical path delay (if the faster operation is not needed) will result in a circuit operating at the same speed, but lower energy. Notice that this reasoning does not require that the curvature of the hardware intensity energy-delay curve be higher than that of the power supply energy-delay curve. Although this property was experimentally verified for a 0.13μ and older CMOS technologies it may or may not hold true in future technologies, depending on the dependence of the gate and subthreshold leakage currents on the power supply. If the curvature of the Vdd energy-delay curve becomes higher, the range in which energy and

performance can be traded through adjusting Vdd will be more limited, but the optimality relation (4) will still hold.

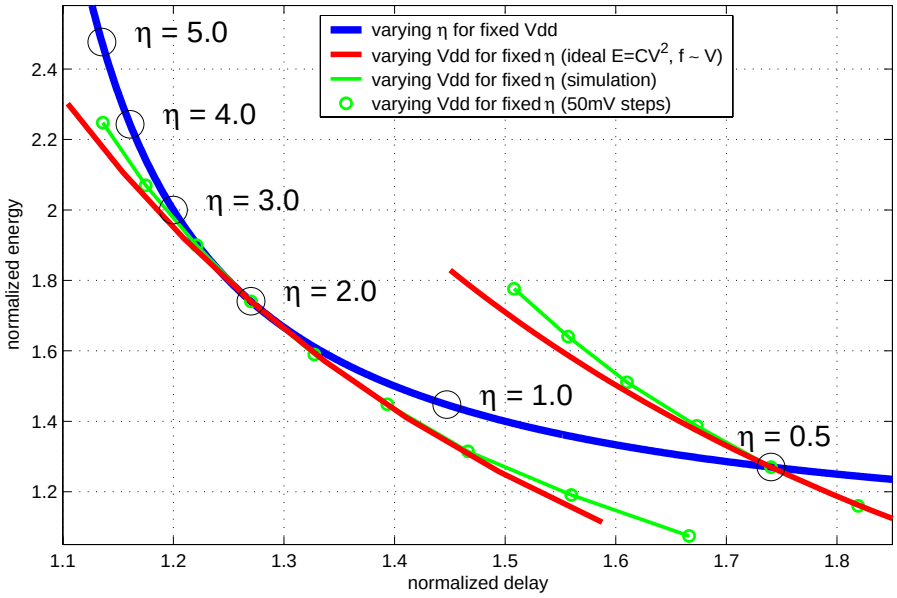

Fig. 2. Graphical interpretation of the optimum hardware intensity balance for an isolated macro.

Although the optimal values of hardware intensities in different pipeline stages, η_i are different, it is useful to abstract for the higher-level microarchitectural analysis of energy-performance tradeoffs a single aggregate quantity for hardware intensity η_{ag} that represents the whole processor, such that $\eta_{ag} = -\frac{D\partial E}{E\partial D}\Big|_v$, where D is the clock period, and E is the total average energy dissipated per cycle in the processor, $E = \sum E_i$. To derive an expression for η_{ag}, notice that increasing the clock cycle time by dD through retuning the circuits in all stages of the pipeline, increases the total energy of the pipeline by $dE = \sum dE_i = -\sum \frac{E_i}{D_i}\eta_i dD$, where the summation is performed over all stages of the pipeline. Since $D_i = D$ (all stages are tuned for the same delay), $\frac{dE}{E} = -\frac{dD}{D}\sum w_i\eta_i$, which means that the aggregate hardware intensity for a multi-stage pipeline is expressed through the hardware intensities of individual stages η_i as

$$\eta_{ag} = \sum_i w_i \eta_i. \tag{8}$$

II. Architectural Complexity

Changing the processor architecture is another way to make tradeoffs between performance and energy. More complex architectures deliver higher architectural performance or IPC, but inevitably dissipate more

energy per every executed instruction. Similar to building the optimal energy-delay curve in the circuit domain, an optimal energy-delay curve can be constructed in the architectural domain, as an envelope in the power-performance space of all feasible architectural alternatives [17]. Similar to (1), the architectural complexity ξ can be defined as[2]

$$\xi = -\left.\frac{D\partial E}{E\partial D}\right|_{\text{fixed v}\eta} \quad \text{or} \quad \xi = -\left.\frac{\%E}{\%D}\right|_{\text{through architecture}} \tag{9}$$

Similar to the optimal balance between η and v in the circuit domain, there exists an optimal balance between architectural complexity ξ and η and v in the unified architectural-circuit domain. By formally solving the problem of minimizing energy as a function of three variables $E = E(\xi, \eta, v)$, subject to a constant delay constraint $D(\xi, \eta, v) = D_0$, we arrive at[3]

$$\frac{\partial D}{\partial \eta}\frac{\partial E}{\partial v} = \frac{\partial D}{\partial v}\frac{\partial E}{\partial \eta} \qquad \frac{\partial D}{\partial \xi}\frac{\partial E}{\partial v} = \frac{\partial D}{\partial v}\frac{\partial E}{\partial \xi} \tag{10}$$

Using definitions (1) and (9), expressions (10) can be re-written as

$$\xi = \eta = \theta \tag{11}$$

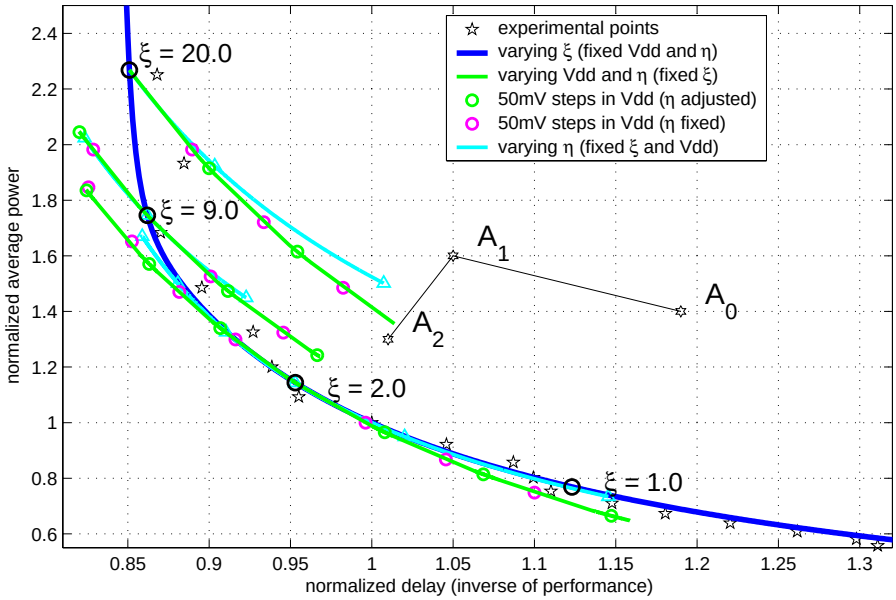

Fig. 3. Graphical interpretation of the optimum architectural complexity balance.

[2]It is assumed that the curvature of the architectural energy-delay curve is such that $\frac{D^2}{E}\frac{\partial^2 E}{\partial D^2} > \xi(\xi+1)$ is satisfied at every point.

[3]The converse problem of minimizing D subject to constant E leads to the same equations.

Figure 3 gives a graphical interpretation of this relation. The solid curve shows the architectural energy-delay curve, plotted by curve-fitting the power-performance data reported in the optimal pipeline depth study [14]. Every data point, plotted as stars in Figure 3, represents a different pipeline depth, and we assume here that the processor microarchitecture was optimally tuned at every pipeline depth. The dotted curves in the figure show the circuit energy-delay curves for fixed ξ, with the power supply and hardware intensity varied simultaneously, so that the optimum balance (4) is observed at each point. Circles mark 50mV steps in Vdd, with the corresponding adjustment in η (6). This data was obtained by simulating a set of representative circuits in a microprocessor [18]. For a reference, triangles show 50mV steps in Vdd without adjusting η. Although the quality of the energy-delay tradeoff of the fixed-η scaling is almost the same as that of the optimal *v*-η scaling, the span is much smaller (larger change in Vdd is needed to achieve the same speed-up or slow-down).

The point at which the architectural energy-delay curve (solid curve) tangents the circuit energy-delay curve (dotted curve) is the point of the optimal balance between ξ, η and *v* in (11). To see this, suppose the architecture is over-designed (say $\xi = 9$). Then by reducing the architectural complexity we can move the design point down the architectural energy-delay curve. Then the performance can be recovered by increasing Vdd (by 100mV in this example) and tuning up circuits for higher η, according to (11), which will move the design point up the circuit energy-delay curve. Since the circuit energy-delay curve is less steep than the architectural energy-delay curve, the same performance will be achieved at a lower power. Similarly, if $\xi < \eta$, say $\xi = 1$, then increasing the architectural complexity to improve the architectural performance (moving up the architectural energy-delay curve) and reducing Vdd and η to save energy (moving down the circuit energy-delay curve) will result in the same performance at a lower power. As with hardware intensity, this relation is not dependent on assumptions about the relative curvatures of the various energy-delay tradeoffs.

Although the nature of the energy-delay tradeoffs at the architectural level is similar to that at the circuit level, one significant difference between them is that with recent advances in the circuit tuning techniques [5] all circuit-level implementations (provided an appropriate circuit topology is chosen) in a properly tuned

processor can be assumed to be on the optimal energy-delay curve, Fig. 2 (designs above the optimal energy-delay curve should be simply discarded), whereas getting the processor architecture that is on the architectural energy-delay curve, Fig. 3, presents a significant challenge. The initial architectural proposal for a new processor is likely to be way off the optimal energy-delay curve. Multiple iterations of optimizing the architecture are required to transfer the design point to the optimal architectural energy-delay curve, and then, to the point of the optimum balance (11), an iterative process illustrated by sequence $A_0 \rightarrow A_1 \rightarrow \ldots A_n$ in Fig. 3. To make the methodology useful for comparing architectural configurations that are not necessarily on the optimal energy-delay curve, the definition of ξ was extended to designs above the optimal energy-delay curve and a more general form of the energy-efficiency criterion was derived in [16], and generalized to include hardware intensity in [18]:

$$\frac{\eta_{ag}}{I}\frac{\triangle I}{\triangle \xi} - \frac{\eta_{ag}+1}{N}\frac{\triangle N}{\triangle \xi} > -\frac{\eta_{ag}}{f}\frac{\triangle f}{\triangle \xi}\bigg|_{\text{fixed } \eta\, v} + \frac{1}{E}\frac{\triangle E}{\triangle \xi}\bigg|_{\text{fixed } \eta\, v} \tag{12}$$

If the inequality holds, then the architectural feature under evaluation is energy-efficient, that is, after adopting it, the processor will deliver higher net performance at the same power budget, after appropriate retuning and, possibly, adjustment in the power supply voltage are done to meet the power budget. In this formula $\frac{\triangle f}{f\triangle \xi}$, $\frac{\triangle I}{I\triangle \xi}$, $\frac{\triangle E}{E\triangle \xi}$, and $\frac{\triangle N}{N\triangle \xi}$ are relative increments in the processor frequency, architectural performance IPC, average energy per instruction and the dynamic instruction count arising from a modification at the architectural or microarchitectural level, evaluated for a *fixed* hardware intensity η_{ag} and power supply v. Thus, all deltas in (12) have the meaning of partial derivatives with respect to the architectural complexity.

The terms $\frac{\triangle I}{I\triangle \xi}$, and $\frac{\triangle N}{N\triangle \xi}$ in (12) can be measured by running the benchmark suite on an architectural simulator. Next we present a methodology for estimating the two remaining terms, $\frac{\triangle f}{f\triangle \xi}$ and $\frac{\triangle E}{E\triangle \xi}$, and derive a new form of the energy-efficiency criterion that does not require estimating the term $\triangle f$.

III. Energy-Efficiency Criterion

FREQUENCY-INVARIANT FORMULATION

The key assumption in deriving the energy-efficiency criterion (12) was that of the optimal tuning of circuits in every pipeline stage (4), (5) and (6) for every architectural alternative, so that the aggregate hardware intensity of the processor η_{ag} (8) is unchanged between designs implementing the architectural alternatives. This assumption imposes special rules on calculating $\frac{\triangle f}{f}$, and $\frac{\triangle E}{E}$, in particular, these relative increments must be calculated assuming that the processor pipeline is re-optimized after every modification to the microarchitecture to satisfy (4), (5) and (6).

Suppose, an architectural feature under evaluation introduces an additional complexity in several (or all) stages of the pipeline, which leads to increments $\triangle D_{\mathrm{i}}|_{\text{no retn}}$ in critical path delays in the corresponding pipeline stages, assuming that no retuning is done to recover the clock frequency. Suppose that the corresponding increments in average energies are $\triangle E_{\mathrm{i}}|_{\text{no retn}}$. The increments $\triangle D_{\mathrm{i}}|_{\text{no retn}}$ and $\triangle E_{\mathrm{i}}|_{\text{no retn}}$ should be evaluated consistently with the initial hardware intensities of the corresponding stages. For example logic added to stage *i* should be tuned (or assumed to be tuned) according to equation (5). Then after adding the logic, the aggregate hardware intensity in pipeline stage *i* will not change. The delay and energy increments may be either positive or negative, and in those pipeline stages that are unaffected by the architectural modification, the delay and energy increments are zero, $\triangle D_{\mathrm{i}}|_{\text{no retn}} = 0$, $\triangle E_{\mathrm{i}}|_{\text{no retn}} = 0$, as shown in Figure 4.

Circuit designers usually have no difficulties estimating the "non-retuned" increments in delay and energy. For example, adding an execution bypass in 10FO4 pipeline results in increments in the critical path delay and average energy of the execution stage of the pipeline which are approximately $\left.\frac{\triangle D_{\mathrm{EX}}}{D}\right|_{\text{no retn}} = 0.2$, and $\left.\frac{\triangle E_{\mathrm{EX}}}{E_{\mathrm{EX}}}\right|_{\text{no retn}} = 0.02$, whereas, adding an extra read port to a multiported register file may result in $\left.\frac{\triangle D_{\mathrm{RF}}}{D}\right|_{\text{no retn}} = 0.1$, and $\left.\frac{\triangle E_{\mathrm{RF}}}{E_{\mathrm{RF}}}\right|_{\text{no retn}} = 0.2$, with no impact in other stages of the pipeline.

In order to recover the clock frequency, circuits in those stages of the pipeline that are negatively affected by the architectural modification need to be tuned up for a higher hardware intensity. To restore the energy optimal balance in the pipeline $\eta_{ag} = \theta$, circuits in all remaining stages need to be tuned down for a lower η,

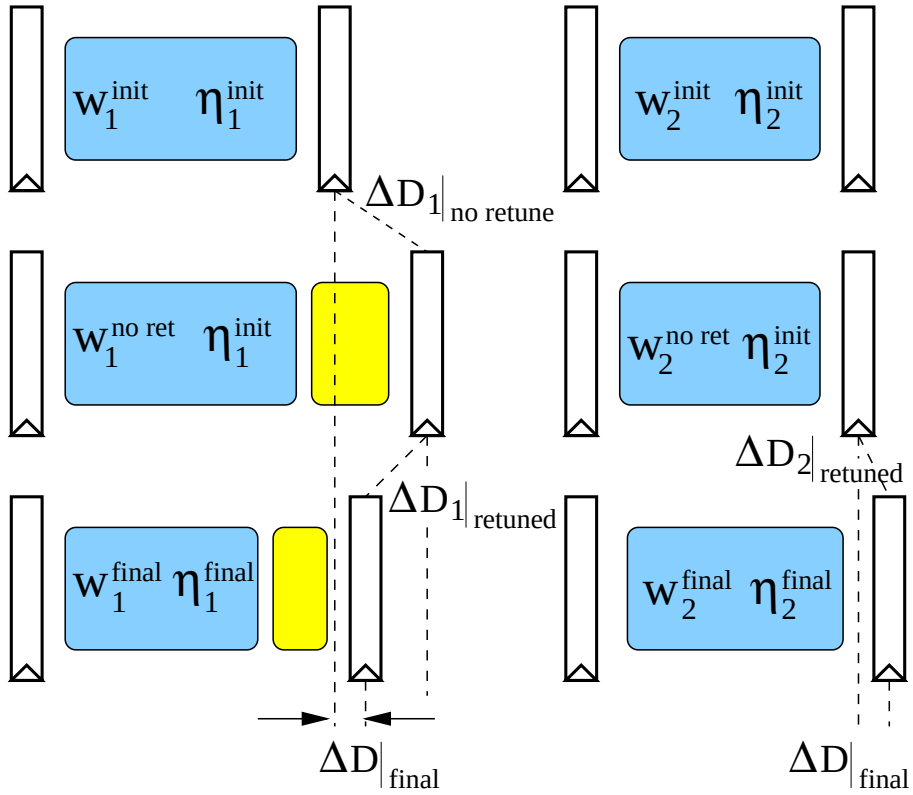

Fig. 4. Retuning pipeline after architectural modification.

so that

$$\triangle\eta_{ag} = \sum_i \eta_i \triangle w_i + \sum_i w_i \triangle\eta_i = 0 \tag{13}$$

where $\triangle\eta_i$ is the increment in the aggregate hardware intensity in stage i as a result of retuning, $\triangle\eta_i = \eta_i^{\text{final}} - \eta_i^{\text{initial}}$, as illustrated in Figure 4, whereas $\triangle w_i$ is the net increment in the corresponding energy weight, as a result of both adding hardware and subsequent retuning, $\triangle w_i = \frac{\triangle E_i}{E} - w_i \frac{\triangle E}{E}$.

We designate by $\triangle D_{\text{i}}|_{\text{retune}}$ and $\triangle E_{\text{i}}|_{\text{retune}}$ the increments in delay and energy in the pipeline stage i as a result of retuning the processor, whereas by $\triangle D_{\text{i}} = \triangle D$ and $\triangle E_{\text{i}}$ we designate the net increment in delay and energy in pipeline stage i as a result of *both* modifying the function and subsequent retuning:

$$\triangle D = \triangle D_{\text{i}}|_{\text{no retn}} + \triangle D_{\text{i}}|_{\text{retune}} \tag{14}$$

$$\triangle E_{\text{i}} = \triangle E_{\text{i}}|_{\text{no retn}} + \triangle E_{\text{i}}|_{\text{retune}} \tag{15}$$

Thus, the net delay and energy increments in every pipeline stage consist of increments due to a change in the functionality resulting from a microarchitectural modification, and additional increments as a result of

retuning the circuits. The net delay increment $\triangle D$ does not need any index because all pipeline stages are assumed to have the same delay before and after the retuning, $D_i = D$. The relative increment in the maximum clock frequency is related to $\triangle D$ as

$$\left.\frac{\triangle f}{f}\right|_{\text{fixed } \eta \text{ v}} = -\frac{\triangle D}{D} \tag{16}$$

Assuming small changes in hardware intensities in all pipeline stages, and neglecting second order terms, the increments in energies $\triangle E_{\text{i}}|_{\text{retune}}$ as a result of the retuning can be expressed through the corresponding increments in delays $\triangle D_{\text{i}}|_{\text{retune}}$ as follows:

$$\triangle E_{\text{i}}|_{\text{retune}} = -\eta_i \frac{E_i}{D} \triangle D_{\text{i}}|_{\text{retune}} \tag{17}$$

Using (14) and (15), the final increments in energies can be expressed as

$$\triangle E_{\text{i}} = \triangle E_{\text{i}}|_{\text{no retn}} - \eta_i \frac{E_i}{D} (\triangle D - \triangle D_{\text{i}}|_{\text{no retn}}) \tag{18}$$

The total increment in energy of the whole pipeline, $\triangle E = \sum \triangle E_i$, is calculated by summing expressions (18) over all pipeline stages and taking advantage of (8) and (16):

$$\left.\frac{\triangle E}{E}\right|_{\text{fixed}\,\eta\text{v}} = \left.\frac{\triangle E}{E}\right|_{\text{no retn}} + \sum_i \eta_i w_i \left.\frac{\triangle D_i}{D}\right|_{\text{no retn}} + \eta_{ag} \left.\frac{\triangle f}{f}\right|_{\text{fixed}\,\eta\text{v}} \tag{19}$$

Substituting this expression into the earlier derived energy-efficiency criterion (12), we notice that the term $\frac{\triangle f}{f}$ cancels out, since in both expressions it has the same meaning of a partial derivative with respect to architectural complexity ξ. Then, dropping $\triangle \xi$ in the denominators of all terms we arrive at the form of the energy-efficiency criterion that does not require estimating the increment in frequency:

$$\eta_{ag} \frac{\triangle I}{I} - (\eta_{ag} + 1) \frac{\triangle N}{N} > \left.\frac{\triangle E}{E}\right|_{\text{no retn}} + \sum_i \eta_i w_i \left.\frac{\triangle D_{\text{i}}}{D}\right|_{\text{no retn}} \tag{20}$$

where $\left.\frac{\triangle E}{E}\right|_{\text{no retn}}$ is the total increase in average energy dissipated per instruction, assuming no retuning, $\left.\frac{\triangle E}{E}\right|_{\text{no retn}} = \sum \left.\frac{\triangle E_i}{E}\right|_{\text{no retn}}$, summation being done over all stages in the pipeline, affected by the architectural modification.

Expression (20) is a more convenient form of the energy-efficiency criterion than (12). According to (20), in order to evaluate the energy-efficiency of some architectural feature, the architects must supply the relative gain (or loss) in the architectural performance $\frac{\triangle I}{I}$ and relative change in the dynamic instruction count $\frac{\triangle N}{N}$ that result from this feature. These estimates can be obtained by running an architectural simulator, or timer, like Turandot [9], [10]. The second term, $\triangle N$ is non-zero if changes to the Instruction Set Architecture (ISA) are considered, or compiler optimizations are analyzed for energy efficiency. It may also be non-zero if microarchitectural changes are considered in a speculative issue processor that impact the average number of instructions executed from mispredicted paths.

The architect needs to consult circuit designers to estimate the impact of the architectural feature under consideration on the average energy dissipated per instruction and the critical path delay through every stage of the pipeline affected by this architectural feature. A significant advantage of the derived formula is that in estimating the relative changes in energy and critical path delays the circuit designer does not need to worry about retuning the circuits to recover the frequency, or reducing the positive timing slack to save power in logic on paths that are no longer critical. Then, the relative increments in critical path delays are summed, multiplied by the appropriate energy weights and hardware intensities. The higher the energy weight w_i and the hardware intensity η_i of a part of the pipeline i affected by the architectural feature the higher the weight of the increase in the critical path delay through this part of the pipeline.

The energy weights w_i in (20) are typically available as part of power budgeting at the early stages of the definition of the processor pipeline. The only additional data that is needed to use the energy-efficiency criterion are hardware intensities η_i in all blocks of the processor. Those quantities can be measured by static tuning tools, like EinsTuner [5], based on the simulations of previous designs, or set as targets at early planning of the microarchitecture, similar to the way the power targets are budgeted.

Then expression (20) is evaluated. If the inequality holds, then the architectural feature under evaluation is energy-efficient, that is, after adopting it, the processor will deliver higher net performance at the same power

budget, after appropriate retuning and, possibly, adjustment in the power supply voltage are done to meet the power budget.

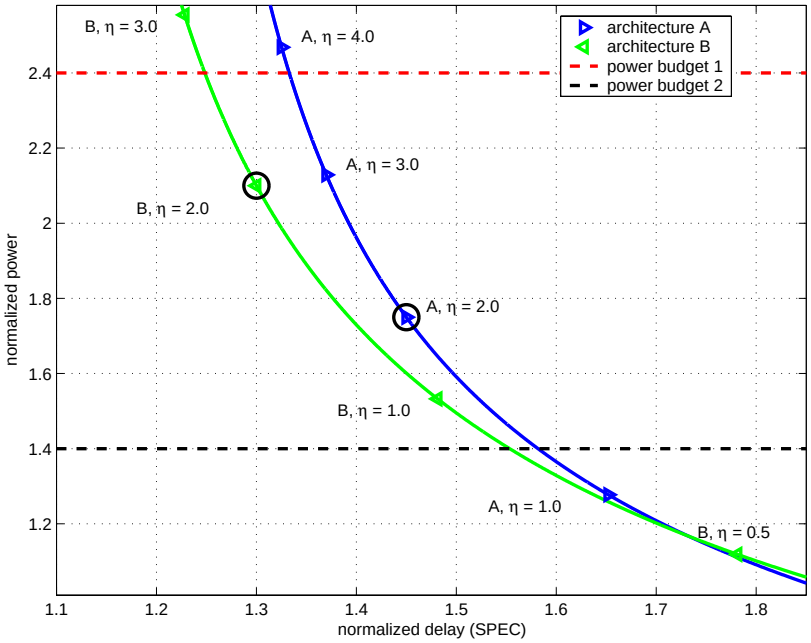

Fig. 5. Graphical interpretation of the energy-efficiency criterion

Figure 5 gives a graphical interpretation of the architectural energy-efficiency criterion (20). Modifying the architecture from an alternative *A* to *B* is evaluated for energy efficiency using (20). The points corresponding to the implementations of both architectural alternatives with the same hardware intensity $\eta = 2$ are marked with circles, and the curves passing through these circles show implementations of architectural alternatives *A* and *B* with different hardware intensities, assuming that the power supply is adjusted accordingly, to keep the optimum balance between the hardware intensity and power supply (6). In other words, these curves show where the corresponding design points will move in the energy-delay space if the same architectures are implemented with more or less aggressive circuits.

If the inequality (20) evaluates as true for the architectural change from alternative *A* to *B*, then in the neighborhood of these points, the circuit energy-delay curve passing through point *B* is below (or left of) the circuit energy-delay curve passing through point *A*, as shown in Figure 5. This means that for any

power budget, within a certain range of the initial points, implementations of architecture *B*, deliver higher performance than implementations of architecture *A*, for the same power budget.

Notice that the curves may intersect, as shown in Figure 5, which means that architectural alternative *B* is more energy efficient than *A* only within some range of the initial design points. This demonstrates the fact that an architecture optimized for a certain range in the power-performance space may not perform well outside of this range. For example, a high-performance core, scaled down to operated in the lower performance space may not be competitive with a core specifically optimized for the low-power low-performance applications. Another conclusion from this analysis is that an accurate estimate of the available power budget for a core is essential for developing an energy-efficient architecture, because only by knowing the power budget can we estimate the maximum value for architectural complexity, hardware intensity and power supply that can be used in the core.

Figure 3 shows a possible outcome of overestimating the power budget available for a processor core at the microarchitecture definition stage, and choosing an overly high value for the architectural complexity. Suppose, at early design stages the power budget is estimated to be at 2.2 and the microarchitecture of the processor core is optimized for the architectural complexity of $\xi = 20$. Suppose that at a circuit phase of the design it is discovered that the actual power budget is only 1.2. Since changing the architecture at this point would result in missing the product release schedule, the only way to bring the processor power under the budget is to re-design all circuits for a lower hardware intensity and reduce the power supply, or just reduce the power supply, if it is too late for redesigning the circuits. This will send the design point down the dashed curve passing through the point $\xi = 20$ in Figure 3, and leading to an almost 15% loss in performance compared to the design originally optimized for the power budget of 1.2 with the architectural complexity of $\xi = 2.0$. Thus, such late changes in the design may lead to a significant performance degradation, and scaling down the power supply to bring an overpowered processor core to the power budget may have a very high performance cost. This demonstrates the importance of accurately estimating the available power budget

at early design stages, and disproves the notion, common in architectural community, that relative power estimates are always sufficient when proposing new architectural features.

IV. Other Power Performance Metrics

Until energy efficiency criterion (20) was introduced the most popular power-performance metric used in the architectural community has been [3], [4], [2], [7], [6], [1], [15], [12], [11]

$$\frac{\text{MIPS}^{\gamma}}{\text{Watt}} \tag{21}$$

with the value of parameter γ ranging from $\gamma = 0$ to $\gamma = 3$, depending on the class of the microprocessor. As was shown in [16], the prior art metric (21) is a special case of the integral form of the derived metric (12), with γ set to $\gamma = \eta_{ag} + 1$.

Another recent work proposed the following metric for evaluating architectural features [13]

$$3\frac{\triangle\text{IPC}}{\text{IPC}} > \frac{\triangle\text{Power}}{\text{Power}} \tag{22}$$

Notice that (22) is also a special case of (12), with $\eta_{ag} = 2$, $\triangle f = 0$ and $\triangle N = 0$, since in clock gated designs Power $\sim E \times$ IPC, and $\frac{\triangle\text{Power}}{\text{Power}} = \frac{\triangle E}{E} + \frac{\triangle\text{IPC}}{\text{IPC}}$.

The main advantage of the derived criterion (20) is that, in addition to being formally derived and being more general than the above metrics (21) and (22), all its terms have a clear meaning and an unambiguous method for estimating them as 'naive' increments in energies and delays. On the other hand, metrics (21) and (22), though correct, may be confusing to use by an architect, because they hide important assumptions about the method for estimating increments in MIPS and Watts. In particular, estimating the term $\triangle$Power may be ambiguous, because it requires knowing both $\triangle f$ and $\triangle E$ which are interrelated and depend on the assumptions about the allowed change in the clock frequency and retuning the pipeline after modifying the architecture. When using these metrics, some architects assume that circuit designers will do whatever is needed to recover the extra delay due to an introduced architectural feature, and set $\triangle f = 0$, neglecting the increase in energy due to redesigning and retuning the circuits. Others calculate the extra delay introduced

due to an added architectural feature and set $\frac{\triangle f}{f} = -\frac{\triangle D}{D}$, assuming that nothing can be done at the circuit level to recover the frequency, and neglecting that circuits in stages not affected by the change will have a timing slack and could be tuned down to save power. In both cases the conclusion of applying metrics (21) and (22) may be incorrect. In the next section we give typical example of incorrectly using metrics (21) and (22).

V. Example: Adding an Execution Bypass

As an example we evaluate the energy-efficiency of implementing an execution bypass in the Integer Unit (IU) of a microprocessor with a target cycle time of 10FO4, and an aggregate hardware intensity target of $\eta_{ag} = 2$. This microarchitectural feature impacts only the Register File (RF) and Execution (EX) stages of the processor. The delay insertion penalty of the bypass multiplexor in front of the latch is approximately 1FO4, and the delay of the bypass wire, including the rebuffering is also approximately 1FO4. Then the relative 'non-retuned' increments in the critical path delays through the register file and execution stages are $\frac{\triangle D_{\mathrm{RF}}}{D}\Big|_{\text{no retn}} = 0.1$ and $\frac{\triangle D_{\mathrm{EX}}}{D}\Big|_{\text{no retn}} = 0.2$.

Adding the bypass multiplexor and the bypass wires also introduces an energy overhead, dissipated whenever the IU is accessed. Based on simulation results, we estimate the relative energy overhead of the bypass wires and multiplexors as 5% of the average energy dissipated in the IU. Suppose the energy budget of the IU is 10% of the total energy dissipated by the microprocessor. Then $\frac{\triangle E}{E}\Big|_{\text{no retn}} = 0.05 \cdot 0.1 = 0.005$

Suppose, the aggregate hardware intensity of the microprocessor is $\eta_{ag} = 2.0$, but since pipelining the register file access and integer functional units has a high cost in IPC degradation, a higher value of hardware intensity is budgeted to them, $\eta_{\mathrm{RF}} = \eta_{\mathrm{EX}} = 3$. Also, suppose, $w_{\mathrm{RF}} = 0.04$ and $w_{\mathrm{EX}} = 0.06$. Then, using the criterion (20), we determine the relative increment in IPC that needs to be demonstrated to justify adding the execution bypass in the IU as $\frac{\triangle I}{I} > 2.7\%$.

Notice that if we used the $\frac{\mathrm{MIPS}^3}{\mathrm{Watt}}$ metric then, depending on the assumption about the change in frequency, the architect could arrive at different conclusions. If the view is taken that the circuit designers can do

nothing to recover the frequency, then metric (21) leads to the answer $\frac{\triangle I}{I} > 20\%$. On the other hand, if the view is taken that circuit designer will do whatever is needed to recover the frequency, and the architect does not need to worry about it, then metric (21) leads to the answer $\frac{\triangle I}{I} > 0.25\%$. Similarly, metric (22) leads to $\frac{\triangle I}{I} > 0.25\%$, assuming the frequency is unchanged. In both cases the conclusions about the energy-efficiency of the IU execution bypass produced by straightforwardly applying metrics (21) and (22) are incorrect.

VI. Conclusions

This chapter analyzed common approaches to trading power and performance in the design of processor cores for SoCs, such as varying the power supply, hardware intensity, and architectural complexity. It was demonstrated that in order to develop an energy-efficient processor core, that is a core that delivers maximum performance at a strictly limited power budget, design decisions at all levels must be balanced in such a way that all forms of spending power have a similar marginal cost. A criterion for optimizing the core architecture was described which is useful for guiding the iterative architectural optimization process that leads to the optimal balance between the architectural complexity, hardware intensity and power supply. It was demonstrated that a single core may not be competitive in both high and low performance domains, and accurate estimates of the available power budget for a core are essential for developing an energy-efficient architecture, as opposed to making decisions based on relative power estimates only. It was also demonstrated that scaling down the power supply to bring an overpowered processor core to under the power budget may have a very high performance cost.

Acknowledgment

The authors would like to thank J. Moreno and K. Warren for the management support.

References

[1] D. Brooks, P. Bose, et al. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE MICRO*, 20(6):26–44, November 2000.

[2] T. Burd and R. Brodersen. Energy efficient CMOS microprocessor design. In *Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, pages 288–297, 1995.

[3] J. Burr and A. Peterson. Energy considerations in multichip module-based multiprocessors. In *Proceedings of ICCD*, pages 593–600, 1991.

[4] A. Chandrakasan, S. Sheng, and R. Brodersen. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, April 1992.

[5] A. Conn et al. Gradient-based optimization of custom circuits using a static-timing formulation. In *Proceedings of Design Automation Conference*, pages 452–459, June 1999.

[6] R. Gonzalez, B. Gordon, and M. Horowitz. Supply and threshold voltage scaling for low power CMOS. *IEEE Journal of Solid-State Circuits*, 32(8):1210–1216, August 1997.

[7] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1283, September 1996.

[8] J. Moreno et al. An innovative low-power high performance programmable signal processor for digital communications. *IBM Journal of Research and Development*, 47(2/3):299–327, 2003.

[9] M. Moudgill, P. Bose, and J.H. Moreno. Validation of Turandot, a fast processor model for microarchitecture exploration. In *Proceedings of the IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 451–457, Feb. 1999.

[10] M. Moudgill, J.D. Wellman, and J.H. Moreno. Environment for PowerPC microarchitecture exploration. *IEEE Micro*, 19(3):9–14, May/June 1999.

[11] K. Nowka, P. Hofstee, and G. Carpenter. Accurate power efficiency metrics and their application to voltage scalable CMOS VLSI design. *to appear in IEEE Transactions on VLSI Systems*, 2003.

[12] P. Penzes and A. Martin. Energy-delay efficiency of VLSI computations. In *Proceedings of the Great Lakes Symposium on VLSI*, pages 104–107, April 2002.

[13] J. Rattner. Making the right hand turn to power efficient computing. In *kynote speech, 35th Annual International Symposium on Microarchitecture*, November 2002.

[14] V. Srinivasan et al. Optimizing pipelines for power and performance. In *Proceedings of the 35th Annual International Symposium on Microarchitecture*, November 2002.

[15] M. Stan. Low-power CMOS with subvolt supply voltages. *IEEE Transactions on VLSI Systems*, 9(2):394–400, April 2001.

[16] V. Zyuban. Unified architecture level energy-efficiency metric. In *Proceedings of the Great Lakes Symposium on VLSI*, pages 24–29, April 2002.

[17] V. Zyuban and P. Kogge. Optimization of high-performance superscalar architectures for energy efficiency. In *IEEE Symposium on Low Power Electronics and Design*, pages 84–89, August 2000.

[18] V. Zyuban and P. Strenski. Unified methodology for resolving power-performance tradeoffs at the microarchitectural and circuit levels. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 166–171, August 2002.