**RC 22234 (99183) 09, November 2001** **Computer Science**

# IBM Research Report

## Architectures for Low Power

**Pradip Bose**
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**

**ACKNOWLEDGMENTS**

The author is indebted to his fellow researchers (including summer interns) who either have been or still are contributing to the power-aware microprocessor project at IBM Watson. In particular, thanks are due to: Victor Zyuban, David Brooks, Alper Buyuktosunoglu, Hans Jacobson, Stanley Schuster, Peter Cook, Jude Rivers, Daniel Citron, J-D Wellman, Prabhakar Kudva and Phil Emma. The author is also thankful to other colleagues in his organization's management chain, including Mike Rosenfield and Eric Kronstadt for their constant support and encouragement. In addition, the author would like to express his gratitude to Prof. Margaret Martonosi at Princeton University and Prof. David Albonesi at University of Rochester for their active support and help in enabling the research work in this area within IBM Research.

## Abstract

We present a brief review of *some* of the most promising ideas in power-aware design at the (micro)architecture level. This review is based primarily on the most recently published work in relevant architecture and design conferences or workshops. We also refer to prior fundamental work on analytical models of pipelined and parallel machine performance and recast the results to fit the modern framework of joint power-performance metrics. The second part of the paper is an attempt at comparing the power-performance scalability of selected microarchitecture paradigms of interest: e.g. wide-issue out-of-order super scalar, multi-cluster superscalars, SMT and CMP. In conclusion, we touch on future areas of research on the topic of power-aware architectures.

## 1. Introduction:

Power dissipation limits have emerged as a key constraint in the design of microprocessors, even for those targeted for the high end server product space. At the low end of the performance spectrum, power has always dominated over performance as the primary design constraint. However, while battery life expectancies have shown modest increases, the larger demand for increased functionality and speed has increased the severity of the power constraint in the world of handheld and mobile systems. At the high end, where performance was always the primary driver, we are witnessing a trend where increasingly, energy and power limits are dictating the high-level processing paradigms, as well as the lower level issues related to clocking and circuit design.

Figure 1 shows the expected maximum chip power (for high performance processors) through the year 2014. The data plotted is based on the updated 2000 projections made by the International Technology Roadmap for Semiconductors [http://public.itrs.net]. The projection indicates that beyond the linear growth period (through year 2005), there will be a saturation in the maximum chip power. This is ostensibly due to thermal/packaging and die size limits that are expected to kick in during that time frame. Beyond a certain power regime, air cooling is not sufficient to dissipate the heat generated; and, use of liquid cooling and refrigeration causes a sharp increase of the cost-performance ratio. Thus, power-aware design techniques, methodologies and tools are of the essence at all levels of design.

In this paper, we first present a survey of some of the most promising ideas in power-aware design at the (micro)architecture level. We base this review on the currently available literature, with special emphasis on relevant work presented at recent architecture conferences and workshops. Where useful, we also refer to prior papers that deal with fundamental issues related to the performance, cost and scalability of concurrent machine architectures. We show how the fundamentals of machine performance relate to the modern problem of architecting processors in a way that allows them to scale well (over time) in terms of joint power-performance metrics. In this context, we comment on and compare the viability and future promise of several microarchitectural paradigms that seem to be catching on in industry: e.g. clustered super scalars, various flavors of multithreading (e.g. SMT) and chip multiprocessors (CMP).

In section 2, we review the fundamentals of pipelined and vector/parallel computation as they relate to performance and energy characteristics. We also dwell briefly on the topic of defining a suitable set of metrics to measure power-performance efficiency at the microarchitecture level. In section 3, we provide a survey of the most promising ideas and approaches in power-aware design at the microarchitecture level, with references to circuit design and clocking issues that are relevant in that discussion. This review is presented in the context of workloads and benchmarks that represent different markets. In section 4, we compare the power-performance outlook of three emerging microarchitectural paradigms in the general purpose processor space: multicluster superscalars, multithreaded processors and chip multiprocessors. We conclude, in section 5, by summarizing the main issues addressed in this survey paper. We also point to a list of future research items that the architecture community needs to pursue in collaboration with the circuit design community in order to achieve the targets dictated by future cost and performance pressures. In passing, we refer briefly to LPX: a research

processor prototype being designed at IBM Watson, to validate some key ideas in power-aware design.

## 2. Fundamentals of Performance and Power: an Architect's View

*Performance Fundamentals* [1-2]

The most straightforward metric for measuring performance is the execution time of a given program mix on the target processor. The execution time can be written as:

$$T = PL * CPI * CT = PL * CPI * (1/f) \quad \text{..... (2.1)}$$

where PL is the dynamic path length of the program mix, measured as the number of machine instructions executed; CPI is the average processor cycles per instruction incurred in executing the program mix; and CT is the processor cycle time (measured in seconds per cycle) whose inverse determines the clock frequency f. Since performance increases with decreasing T, one may formulate performance, Perf as:

$$Perf_{chip} \sim= K_{pf} . f \sim= K_{pv} . V \quad \text{..... (2.2)}$$

where, the K's are constants for a given microarchitecture-compiler implementation and for a specific workload mix. (This is often referred to as IPC, the inverse of CPI). The $P_{PF}$ value stands for the average number of machine instructions that are executed per cycle on the machine being measured. Performance, $Perf_{chip}$, in this case is measured in units like (millions of) instructions per second, or mips.

Selecting a suite of publicly available benchmark programs that everybody accepts as being "representative" of real-world workloads is difficult to begin with. Adopting a non-controversial weighted mix is also not easy. For the commonly used SPEC benchmark suite (see http://www.specbench.org) the SPECmarks rating (for each class: e.g. integer or floating point) is derived as a geometric mean of execution time ratios for the programs within that class. Each ratio is calculated as the speedup with respect to execution time on a specified reference machine. This method has the advantage that different machines can be ranked unambiguously from a performance viewpoint, if one believes in the particular benchmark suite. That is, the ranking can be shown to be independent of the reference machine used in such a formulation.

Let us now discuss the basics of power dissipation in a processor chip.

*Power Fundamentals* [2-5]:

At the elementary transistor gate (e.g. an inverter) level, total power dissipation can be formulated as the sum of three major components: switching loss, leakage and short-circuit loss.

$$Power_{device} = (1/2)C.V_{dd}.V_{swing}.a.f + I_{leakage} . V_{dd} + Isc . V_{dd} \quad \text{..... (2.3)}$$

where, C is the output capacitance, $V_{dd}$ is the supply voltage, f is the chip clock frequency and a is the activity factor ($0 < a \leq 1$) which determines the device switching frequency; $V_{swing}$ is the maximum voltage swing across the output capacitor, which in general is less than $V_{dd}$; $I_{leakage}$ is the leakage current and $I_{sc}$ is the short-circuit current. In the literature, $V_{swing}$ is often approximated to be equal to $V_{dd}$ (or simply V for short) making the switching loss ~ $(1/2)C.V^2.a.f$. Also, as discussed in [3], for today's range of $V_{dd}$ (say 1 V to 3 V) switching loss: $(1/2)CV^2af$ remains the dominant component. So, as a first-order approximation, for the whole chip, we may formulate the power dissipation to be:

$$Power_{chip} = (1/2)[\sum_i C_i . V_i^2 . a_i . f_i] \quad \ldots\ldots\ldots\ldots\ldots (2.4)$$

where, $C_i$, $V_i$, $a_i$ and $f_i$ are unit- or block-specific average values in the most general case; the summation is taken over all blocks or units i, at the microarchitecture level (e.g. icache, dcache, integer unit, floating point unit, load-store unit, register files and buses [if not included in individual units], etc). Also, for the voltage range considered, the operating frequency is roughly proportional to the supply voltage; and the capacitance C remains roughly the same if we keep the same design but scale the voltage. If a single voltage and clock frequency are used for the whole chip, the above reduces to:

$$Power_{chip} = V^3 . (\sum_i K_i^v . a_i) = f3 . (\sum_i K_i^f . a_i) \quad \ldots\ldots (2.5)$$

If we consider the worst-case activity factor for each unit i, i.e. if $a_i = 1$ for all i, then,

$$Power_{chip} = K_V . V^3 = K_F . f^3 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.6)$$

where $K_V$ and $K_F$ are design-specific constants.

The last equation, 2.6 is what leads to the so-called "cube-root" rule [3], where redesigning a chip to operate at ½ the voltage (and frequency), results in the power dissipation being lowered to $(1/2)^3$ or 1/8 of the original. This implies the single-most efficient method for reducing power dissipation for a processor that has already been designed to operate at high frequency: namely, reduce the voltage (and hence the frequency). It is believed that this is the primary mechanism of power control in the Transmeta chip (see http://www.transmeta.com). There is a limit, however, of how low $V_{dd}$ can be reduced (for a given technology), which has to do with manufacturability and circuit reliability issues. Thus, a combination of microarchitecture and circuit techniques to reduce power consumption, without necessarily employing multiple or variable supply voltages is of special relevance in the design of robust systems.

*Power-Performance Efficiency Metrics:*

The most common (and perhaps obvious) metric to characterize the power-performance efficiency of a microprocessor is a simple ratio, like mips/watt. This attempts to quantify the efficiency by projecting the performance achieved or gained (measured in millions of instructions per second) for every watt of power consumed. Clearly, the higher the number, the "better" the

machine is. While this seems a reasonable choice for some purposes, there are strong arguments against it in many cases, especially when it comes to characterizing higher end processors. Performance has typically been the key driver of such server-class designs and cost or efficiency issues have been of secondary importance. Specifically, a design team may well choose a higher frequency design point (which meets maximum power budget constraints) even if it operates at a much lower mips/watt efficiency compared to one that operates at better efficiency but at a lower performance level. As such, $(mips)^2$/watt or even $(mips)^3$/watt may be the metric of choice at the high end. On the other hand, at the lowest end, where battery-life (or energy consumption) is the primary driver, one may want to put an even greater weight on the power aspect than the simplest mips/watt metric; i.e. one may just be interested in minimizing the watts for a given workload run, irrespective of the execution time performance, provided the latter does not exceed some specified upper limit.

The "mips" metric for performance and the "watts" value for power may refer to average or peak values, derived from the chip specifications. For example, for a 1 gigahertz (= $10^9$ cycles/sec) processor which can complete up to 4 instructions per cycle, the theoretical peak performance is 4000 mips). If the average completion rate for a given workload mix is p instructions per cycle, then the average mips would equal 1000 times p. However, when it comes to workload-driven evaluation and characterization of processors, metrics are often controversial. Apart from the problem of deciding on a "representative" set of benchmark applications, there are fundamental questions which persist about how to boil down "performance" into a single ("average") rating that is meaningful in comparing a set of machines. Since power consumption varies, depending on the program being executed, the issue of benchmarking is also relevant in assigning an average power rating. In measuring power and performance together for a given program execution, one may use a fused metric like power-delay product (PDP) or energy-delay product (EDP) [5,6]. In general, the PDP-based formulations are more appropriate for low-power, portable systems, where battery-life is the primary index of energy efficiency. The mips/watt metric is an inverse PDP formulation, where delay refers to average execution time per instruction. The power-delay product, being dimensionally equal to energy, is the natural metric for such systems. For higher end systems (e.g. workstations) the EDP-based formulations are deemed to be more appropriate, since the extra delay factor ensures a greater emphasis on performance. The $(mips)^2$/watt metric is an inverse EDP formulation. For the highest performance, server-class machines, it may be appropriate to weight the "delay" part even more. This would point to the use of $(mips)^3$/watt, which is an inverse $ED^2P$ formulation. Alternatively, one may use $(cpi)^3$.watt as a direct $ED^2P$ metric, applicable on a "per instruction" basis (see [2]).

The energy*$(delay)^2$ metric, or $perf^3$/power formula is analogous to the cube-root rule [3] which follows from constant voltage scaling arguments (see previous discussion, equation 2.6). Clearly, to formulate a voltage-invariant power-performance characterization metric, we need to think in terms of $perf^3$/(power). When we are dealing with the SPEC benchmarks, one may therefore evaluate efficiency as $(SPECrating)^x$/watt, or $(SPEC)^x$/watt for short; where the exponent value x (= 1, 2, or 3) may depend on the class of processors being compared.

Figure 1 shows the power-performance efficiency data [sources used are: http://www.bwrc.eecs.berkeley.edu/CIC/, http://www.specbench.org, Microprocessor Report,

August 2000 and individual vendor web sites] for a range of commercial processors of approximately the same generation. In each chart, the latest available processor is plotted on the left and the oldest one on the right. We have used SPEC/watt, $SPEC^2$/watt and $SPEC^3$/watt as the alternative metrics, where SPEC stands for the processor's SPEC95 rating (see definition principles, earlier in this section or in [1]). For each category, like $SPEC^2$/watt, the best performer is normalized to 1, and the other processor values are plotted as relative fractions of the normalized maximum. The data validates our assertion that depending on the metric of choice, and the target market (determined by workload class and/or the power/cost) the conclusion drawn about efficiency can be quite different. For performance-optimized, high-end processors, the $SPEC^3$/watt metric seems to be fairest, with the very latest Intel Pentium-III and AMD Athlon offerings (at 1 GHz) at the top for integer workloads; and, the older HP-PA 8600 (552 MHz) and IBM Power3 (450 Mhz) still dominating in the floating point class. For "power-first" processors targeted towards integer workloads (like Intel's mobile Celeron-333) spec/watt seems to be the fairest.

Table 1 and 2 below show the explicit ranking of the processors considered in Figure 1, from a power-performance efficiency viewpoint based on specint and specfp benchmarks. The only intent here is to illustrate the point that we tried to make earlier: that depending on the intended market (e.g. general purpose: server-class, workstation or low-power mobile, etc.) and application class (e.g. integer-intensive or floating point intensive) different efficiency metrics may be suitable. Note that we have relied on published performance and "max power" numbers; and, because of differences in the methodologies used in quoting the maximum power ratings, the derived rankings may not be completely accurate or fair. As an example, the 33 W maximum power rating for the Intel PIII-1000 processor that we computed from the maximum current and nominal voltage ratings specified for this part in the vendor's web page [http://www.intel.com/design/pentiumiii/datashts/245264.htm], is higher than that reported in the Microprocessor Report source cited before. Actually, this points to the need of standardization of methods used in reporting maximum and average power ratings for future processors. It should be possible, in future for customers to compare power-performance efficiencies across competing products in a given market segment, i.e. for a given benchmark suite.

## 3. A Review of Key Ideas in Power-Aware Microarchitectures

In this paper, we limit our attention to dynamic ("switching") power governed by the $CV^2af$ formula. Recall that C refers to the switching capacitance, V is the supply voltage, a is the activity factor ($0 < a < 1$) and f is the operating clock frequency. Power reduction ideas must therefore focus on one or more of these basic parameters. In this section, we examine the key ideas that have been proposed in terms of (micro)architectural support for power-efficiency.

The effective (average) value of C can be reduced by using: (a) area-efficient designs for various macros; (b) adaptive structures, that change in effective size, latency or communication bandwidth depending on the needs of the input workload; (c) selectively "powering off" unused or idle units, based on special "nap/doze" and "sleep" instructions generated by the compiler or detected via hardware mechanisms; (d) reducing or eliminating "speculative waste" resulting from executing instructions in mis-speculated branch paths or prefetching useless instructions and data into caches, based on wrong guesses.

The average value of V can be reduced via dynamic voltage scaling, i.e. by reducing the voltage as and when required or possible (e.g. see the description of the Transmeta chip: http://www.transmeta.com). Microarchitectural support, in this case, is not required, unless the mechanisms to detect "idle" periods or temperature overruns are detected using counter-based "proxies", specially architected for this purpose. Hence, in this paper, we do not dwell on dynamic voltage scaling methods. (Note again, however, that since reducing V also results in reduction of the operating frequency, f, net power reduction has a cubic effect; thus, dynamic voltage scaling, though not a microarchitectural technique *per se*, is the most effective way of power reduction).

The average value of the activity factor, a, can be reduced by: (a) the use of clock-gating, where the normally free-running, synchronous clock is disabled in selected units or sub-units within the system based on information or predictions about current or future activity in those regions; (b) the use of data representations and instruction schedules that result in reduced switching. Microarchitectural support is provided in the form of added mechanisms to: (a) detect, predict and control the generation of the applied gating signals or (b) aid in power-efficient data and instruction encodings. Compiler support for generating power-efficient instruction scheduling and data partitioning or special instructions for "nap/doze/sleep" control, if applicable, must also be considered under this category.

Lastly, the average value of the frequency, f, can be controlled or reduced by using: (a) variable, multiple or locally asynchronous (self-timed) clocks; (b) reduced pipeline depths.

We consider power-aware microarchitectural constructs that use C, a or f as the primary power-reduction lever. In any such proposed processor architecture, the efficacy of the particular power reduction method that is used must be assessed by understanding the net performance impact. Here, depending on the application domain (or market), a PDP, EDP or $ED^2P$ metric for evaluating and comparing power-performance efficiciencies must be used. (See earlier discussion in Section 2).

*Optimal Pipeline Depth*

A fundamental question that is asked has to do with pipeline depth. Is a deeply pipelined, high frequency ("speed demon") design better than an IPC-centric lower frequency ("braniac") design? In the context of this paper, "better" must be judged in terms of power-performance efficiency.

Let us consider, first, a simple, hazard-free, linear pipeline flow process, with k stages. Let the time for the total logic (without latches) to compute one answer be T. Assuming that the k stages into which the logic is partitioned are of equal delay, the time per stage and thus the time per computation becomes (see [7], Chapter 2):

$$t = T/k + D \quad \text{.....(3.1)}$$

where D is the delay added due to the staging latch. The inverse of t determines the clocking rate or frequency of operation. Similarly, if the energy spent (per cycle, per second or over the duration of the program run) in the logic is W and the corresponding energy spent per level of staging latches is L, then the total energy equation for the k-stage pipelined version is roughly,

$$E = L.k + W \quad \text{.....(3.2)}$$

The energy equation assumes that the clock is free-running, i.e., on every cycle, each level of staging latches is clocked to enable the advancement of operations along the pipeline. (Later, we shall consider the effect of clock-gating). Equations (3.1) and (3.2), when plotted as a function of k, are depicted in Figures 2(a) and 2(b) respectively.

As the number of stages increases, the energy or power consumed increases linearly; while, the performance also increases, but not as fast. In order to consider the PDP-based power-performance efficiency, we compute the ratio:

$$\frac{\text{Power}}{\text{Performance}} = (L.k + W)(T/k + D) = L.T + W.D + (L.D.k^2 + W.T)/k \quad \ldots\ldots\ldots \quad (3.3)$$

Figure 3 shows the general shape of this curve as a function of k. Differentiating the right hand side expression in (3.3) and setting it to zero, one can solve for the optimum value of k for which the power-performance efficiency is maximized; i.e., the minimum of the curve in Figure 2(b) can be shown to occur when

$$k\,(\text{opt.}) = \sqrt{(W.T)/(L.D)} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (3.4)$$

Larson [8] first published the above analysis, albeit from a cost/performance perspective. This analysis shows that, at least for the simplest, hazard-free pipeline flow, the highest frequency operating point achievable in a given technology may not be the most energy-efficient! Rather, the optimal number of stages (and hence operating frequency) is expected to be at a point which increases for greater W or T and decreases for greater L or D.

For real super scalar machines, the number of latches in a design tends to go up much more sharply with k than the linear assumption in the above model. This tends to make k (opt) even smaller. Also, in real pipeline flow with hazards, e.g., in the presence of branch-related stalls and disruptions, performance actually peaks at a certain value of k before decreasing [3, 9] (instead of the asymptotically increasing behavior shown in Figure 2(b)). This effect would also lead to decreasing the effective value of k (opt). (However, k(opt) increases if we use EDP or $ED^2P$ metrics instead of the PDP metric used.)

*Vector/SIMD Processing Support*

Vector/SIMD modes of parallelism present in current architectures afford a power-efficient method of extending performance for vectorizable codes. Fundamentally, this is because: for doing the work of fetching and processing a single (vector) instruction, a large amount of data is processed in a parallel or pipelined manner. If we consider a SIMD machine, with p k-stage functional pipelines (see Figure 4) then looking at the pipelines alone, one sees a p-fold increase of performance, with a p-fold increase in power, assuming full utilization and hazard-free flow, as before. Thus, a SIMD pipeline unit offers the potential of scalable growth in performance, with commensurate growth in power; i.e. at constant power-performance efficiency. If, however, one includes the instruction cache and fetch/dispatch unit that are shared across the p SIMD pipelines, then power-performance efficiency can actually grow with p. This

is because, the energy behavior of the instruction cache (memory) and the fetch/decode path remains essentially invariant with p, while net performance grows linearly with p.

In a super scalar machine with a vector/SIMD extension, the overall power-efficiency increase is limited by the fraction of code that runs in SIMD-mode (Amdahl's Law).

*Clock-Gating: Power Reduction Potential and Microarchitectural Support*

Clock-gating refers to circuit-level control (e.g. see [10, 17]) for disabling the clock to a given set of latches, a macro, a bus or an entire unit, on a particular machine cycle. Microarchitecture-level analysis points to opportunities of power savings in a processor, since idle periods of a particular resource can be identified and quantified. Figure 5 depicts the execution pipe utilization of the various functional units (e.g. fixed point or integer unit, floating point unit, load-store unit, branch unit, condition register unit) within a current generation, out of order superscalar processor. This data is based on *simulation-based* data of a hypothetical processor, similar in complexity to that of current generation designs like the Power4™ [11]. We show the results for selected SPEC95 benchmarks and a large sample from a commercial TPC-C trace.The data shows that the pipe utilization attains a maximum of slightly over 50 % only for the FXU; in many cases, the utilization is quite low, often 10 % or less.

Figures 6(a) and 6(b) show the opportunities available within several units (and in particular, the instruction fetch unit, IFU) of the same example processor in the context of the TPC-C trace segment referred to above. Figure 6(a) depicts the instruction frequency mix of the trace segment used. This shows that the floating point unit (FPU) operations are a very tiny fraction of the total number of instructions in the trace. Therefore, with proper detection and control mechanisms architected in hardware, the FPU unit could essentially be "gated off" in terms of the clock delivery for the most part of such an execution. Figure 6(b) shows the fraction of total cycles spent in various modes within the instruction fetch unit (IFU). I-fetch was on hold for about 48 % of the cycles; and the fraction of useful fetch cycles was only 28 %. Again, this points to great opportunities: either in terms of clock-gating or dynamic ifetch throttling (see *Dynamic Throttling of Communication Bandwidths* below).

(Micro)architectural support for clock-gating can be provided in at least three ways: (a) dynamic detection of idle modes in various clocked units or regions within a processor or system; (b) static or dynamic prediction of such idle modes; (c) using "data valid" bits within a pipeline flow path to selectively enable/disable the clock applied to the pipeline stage latches. If static prediction is used, the compiler inserts special "nap/doze/sleep/wake" type instructions where appropriate, to aid the hardware in generating the necessary gating signals. Methods (a) and (b) result in coarse-grain clock-gating, where entire units, macros or regions can be gated off to save power; while, method (c) results in fine-grain clock-gating, where unutilized pipe segments can be gated off during normal execution within a particular unit, like the FPU. The detailed circuit-level implementation of gated-clocks, the potential performance degradation, inductive noise problems, etc. are not discussed in this paper. However, these are very important issues that must be dealt with adequately in an actual design.

Referring back to Figures 2 and 3, note that since (fine-grain) clock-gating effectively causes a fraction of the latches to be "gated off", we may model this by assuming that the

effective value of L decreases when such clock-gating is applied. This has the effect of increasing k (opt.); i.e., the operating frequency for the most power-efficient pipeline operation can be increased in the presence of clock-gating. This is an added benefit.

*Variable Bit-Width Operands*

One of the techniques proposed for reducing dynamic power consists of exploiting the behavior of data in programs, which is characterized by the frequent presence of small values. Such values can be represented as and operated upon as short bit-vectors. Thus, by using only a small part of the processing datapath, power can be reduced without loss of performance. Brooks and Martonosi [12] analyzed the potential of this approach in the context of 64-bit processor implementations (e.g. the Compaq Alpha™ architecture). Their results show that roughly 50 % of the instructions executed had both operands whose length was less than or equal to 16 bits. Brooks and Martonosi proposed an implementation that exploits this by dynamically detecting the presence of narrow-width operands on a cycle-by-cycle basis. (Subsequent work by Jude Rivers et al. at IBM has documented an approach to exploit this in PowerPC architectures, using a different implementation. This work is still not available for external publication).

*Adaptive Microarchitectures*

Another method of reducing power is to adjust the size of various storage resources within a processor or system, with changing needs of the workload. Albonesi [13] proposed a dynamically reconfigurable caching mechanism, that reduces the cache size (and hence power) when the workload is in a phase that exhibits reduced cache footprint. Such downsizing also results in improved latency, which can be exploited (from a performance viewpoint) by increasing the cache cycling frequency on a local clocking or self-timed basis. Maro et al. [14] have suggested the use of adapting the functional unit configuration within a processor in tune with changing workload requirements. Reconfiguration is limited to "shutting down" certain functional pipes or clusters, based on utilization history or IPC performance. In that sense, the work by Maro et al. is not too different from coarse-grain clock-gating support, as discussed earlier. In recent work done at IBM Watson, Buyuktosunoglu et al. [15] designed an adaptive issue queue that can result in (up to) 75 % power reduction when the queue is sized down to its minimum size. This is achieved with a very small IPC performance hit. Another example is the idea of adaptive register files (e.g., see [16]) where the size and configuration of the active size of the storage is changed via a banked design, or through hierarchical partitioning techniques.

*Dynamic Thermal Management*

Most clock-gating techniques are geared towards the goal of reducing *average* chip power. As such, these methods do not guarantee that the worst-case (maximum) power consumption will not exceed safe limits. The processor's maximum power consumption dictates the choice of its packaging and cooling solution. In fact, as discussed in [17], the net cooling solution cost increases in a piecewise linear manner with respect to the maximum power; and the cost gradient increases rather sharply in the higher power regimes. This necessitates the use of mechanisms to limit the maximum power to a controllable ceiling, one defined by the cost profile of the market for which the processor is targeted. Most recently, in the high performance world, Intel's Pentium 4 processor is reported to use an elaborate on-chip thermal management system

to ensure reliable operation [17]. At the lower end, the G3 and G4 PowerPC microprocessors [18, 19] include a Thermal Assist Unit (TAU) to provide dynamic thermal management. In recently reported academic work, Brooks and Martonosi [20] discuss and analyze the potential reduction in "maximum power" ratings without significant loss of performance, by the use of specific dynamic thermal management (DTM) schemes. The use of DTM requires the inclusion of on-chip sensors to monitor actual temperature; or proxies of temperature [20] estimated from on-chip counters of various events and rates.

*Dynamic Throttling of Communication Bandwidths*

This idea has to do with reducing the width of a communication bus dynamically, in response to reduced needs or in response to temperature overruns. Examples of on-chip buses that can be throttled are: instruction fetch bandwidth, instruction dispatch/issue bandwidths, register renaming bandwidth, instruction completion bandwidths, memory address bandwidth, etc. In the G3 and G4 PowerPC microprocessors [18, 19], the TAU invokes a form of instruction cache throttling as a means to lower the temperature when a thermal emergency is encountered.

*Speculation Control*

In current generation, high performance microprocessors, branch mispredictions and mis-speculative prefetches end up wasting a lots of power. Manne et al. [21, 22] have described means of detecting or anticipating an impending mispredict and using that information to prevent mis-speculated instructions from entering the pipeline. These methods have been shown to reduce power by up to 38 % with less than a 1 % performance loss.

## 4. Power-Efficient Microarchitecture Paradigms

Now that we have examined specific microarchitectural constructs that aid power-efficient design, let us examine the inherent power-performance scalability and efficiency of selected paradigms that are currently emerging in the high-end processor roadmap. In particular, we consider: (a) wide-issue, speculative super scalar processors; (b) multi-cluster superscalars; (c) chip multiprocessors (CMP) - especially those that use single program speculative multithreading (e.g. multiscalar); (d) simultaneously multithreaded (SMT) processors.

In illustrating the efficiency advantages or deficiencies, we use the following running example. It shows one iteration of a loop trace that we consider in simulating the performance and power characteristics across the above computing platforms.

Let us consider the following floating-point loop kernel, shown below (coded using the PowerPC™ instruction set architecture):

*Example Loop Test Case*

```
[P] [A]  fadd   fp3,  fp1,  fp0
[Q] [B]  lfdu   fp5,  8(r1)
[R] [C]  lfdu   fp4,  8(r3)
[S] [D] fadd  fp4, fp5, fp4
[T] [E]  fadd  fp1, fp4, fp3
[U] [F]  stfdu fp1, 8(r2)
[V] [G]  bc  loop_top
```

The loop body consists of 7 instructions, the final one being a conditional branch that causes control to loop back to the top of the loop body. The instructions are labeled A through G. (The labels P through V are used to tag the corresponding instructions for a parallel thread - when we consider SMT and CMP). The lfdu/stfdu instructions are load/store instructions with update, where the base address register (e.g. r1, r2 or r3) is updated after execution by holding the newly computed address.

*Single-core superscalar processor paradigm*

One school of thought anticipates a continued progression along the path of wider, aggressively superscalar paradigms. Researchers continue to innovate in an attempt to extract the last "ounce" of IPC-level performance from a single-thread instruction-level parallelism (ILP) model. Value prediction advances (pioneered by Lipasti et al. [23]) promise to break the limits imposed by true data dependencies. Trace caches (Smith et al. [23]) ease the fetch bandwidth bottleneck, which can otherwise impede scalability. However, increasing the superscalar width beyond a certain limit tends to yield diminishing gains in *net* performance (i.e. the inverse of CPI * CT; see equation 2.1). At the same time, the power-performance efficiency metric (e.g. performance per watt or $(\text{performance})^2$/watt, etc) tends to degrade beyond a certain complexity point in the single-core superscalar design paradigm. This is illustrated below in the context of our example loop test case.

Let us consider a base machine that is a 4-wide superscalar, with two load-store units supporting two floating point pipes (see Figure 7). The data cache has two load ports and a separate store port. Two load-store store unit pipes (LSU0 and LSU1) are fed by a single issue queue, LSQ; similarly, the two floating point unit pipes (FPU0 and FPU1) are fed by a single issue queue, FPQ. In the context of the loop above, we essentially focus on the LSU-FPU sub-engine of the whole processor.

Let us assume the following high-level parameters (latency and bandwidth) characterizing the base super scalar machine model of width W = 4.

* Instruction fetch bandwidth, fetch_bw = 2*W = 8 instructions/cycle.
* Dispatch/decode/rename bandwidth, disp_bw = W = 4 instructions/cycle; dispatch is assumed to stall beyond the first branch scanned in the instruction fetch buffer.
* Issue_bandwidth from LSQ (reservation station), lsu_bw = W/2 = 2 instructions/cycle
* Issue_bandwidth from FPQ, fpu_bw = W/2 = 2 instructions/cycle.
* Completion bandwidth, compl_bw = W = 4 instructions/cycle
* Back-to-back dependent floating point operation issue delay, fp_delay = 1 cycle.
* The best-case load latency, from fetch to writeback is: 5 cycles
* The best-case store latency, from fetch to writing in the pending store queue is: 4 cycles; (a store is eligible to complete the cycle after the address-data pair is valid in the store queue).
* The best-case floating point operation latency, from fetch to writeback is: 7 cycles (when the issue queue, FPQ is bypassed, because it is empty).

Loads and floating point operations are eligible for completion (retirement) the cycle after writeback into rename buffers. For simplicity of analysis let us assume that the processor uses

in-order issue from the issue queues (LSQ and FPQ). In our simulation model, the superscalar width W is a ganged parameter, defined as follows:

W = (fetch_bw/2) = disp_bw = compl_bw.
The number of LSU units, ls_units, FPU units, fp_units, data cache load ports, l_ports and data cache store ports are varied as follows as W is changed:
ls_units = fp_units = l_ports = max [floor(W/2), 1].
s_ports = max [floor(l_ports/2), 1].

For illustrative purposes, a simple (and decidedly naive) analytical energy model is assumed, where the power consumed is a function of the following parameters: W, ls_units, fp_units, l_ports and s_ports. In particular, the power, PW, in watts is computed as: $PW = K * [(W)^y + ls\_units + fp\_units + l\_ports + s\_ports]$, where y ($0 < y < 1$) is an exponent that may be varied to see the effect on power-performance efficiency; K is a constant. Figure 8 shows the performance and performance/power ratio variation with superscalar width, W; for this graph, y has been set to 0.5 and the scaling constant K is 2. The BIPS (billions of instructions per second) values are computed from the IPC (instruction per cycle) values, assuming a clock frequency of 1 GHz.

The graph in Figure 8(a) shows that a maximum issue width of W = 4 could be used to achieve the best (idealized) BIPS performance. This idealized plot is obtained using a tool called eliot [24]. This is a parameterized, PowerPC super scalar model, that can operate either in cycle-by-cycle simulation mode, or, it can generate idealized bounds, based on static analysis of a loop code segment. The eliot model has now been updated to include parameterized, analytical energy models for each unit or storage resource within the processor. This new tool, called elpaso can be used to generate power-performance efficiency data for loop test cases. As shown in Figure 8 (b), from a power-performance efficiency viewpoint (measured as a performance over power ratio), the best-case design is achieved for $W < 4$. Depending on the sophistication and accuracy of the energy model (i.e. how power varies with microarchitectural complexity), and the exact choice of the power-performance efficiency metric, the inflexion point in the curve in Figure 8(b) changes; however, it should be obvious that beyond a certain superscalar width, the power-performance efficiency diminishes continuously. Fundamentally, this is due to the single-thread ILP limit of the loop trace being considered (as apparent from Figure 8 (a) ).

Note, by the way that the resource sizes are assumed to be large enough, so that they are effectively infinite for the purposes of our running example above. Some of the actual sizes assumed for the base case (W=4) are:
Completion (reorder) buffer size, cbuf_size = 32; load-store queue size, lsq_size = 6; floating point queue size, fpq_size = 8; pending store queue size, psq_size = 16;

The microarchitectural trends beyond the current superscalar regime, are effectively targeted towards the goal of extending the power-performance efficiency factors. That is, the complexity growth must ideally scale at a slower rate than the growth in performance. Power consumption is one index of complexity; it also determines packaging and cooling costs. (Verification cost and effort is another important index). In that sense, a microarchitecture paradigm that ensures that the power-performance efficiency measure of choice is a non-decreasing function of time: is the ideal, complexity-effective design paradigm for the future. Of course, it is hard to keep scaling a given paradigm beyond a few processor generations.

Whenever we reach a maximum in the power-performance efficiency curve, it is time to invoke the next paradigm shift.

Next, we examine some of the promising new trends in microarchitecture that can serve as the next platform for designing power-performance scalable machines.

*Multicluster superscalar processors*

As described in our earlier article [2], Zyuban et al. [25, 26] studied the class of multicluster superscalar processors as a means of extending the power-efficient growth of the basic super scalar paradigm. One way to address the energy growth problem at the microarchitectural level is to replace a classical superscalar CPU with a set of clusters, so that all key energy consumers are split among clusters. Then, instead of accessing centralized structures in the traditional superscalar design, instructions scheduled to an individual cluster would access local structures most of the time. The main advantage of accessing a collection of local structures instead of a centralized one is that the number of ports and entries in each local structure is much smaller. This reduces the latency and energy per access. If the non-accessed sub-structures of a resource can be "gated off" (e.g. in terms of the clock), then, the net energy savings can be substantial.

According to the results obtained in Zyuban's work, the energy dissipated per cycle in every unit or sub-unit within a superscalar processor can be modeled to vary (approximately) as $IPC_{unit} * (IW)^g$, where IW is the issue width, $IPC_{unit}$ is the average IPC performance at the level of the unit or structure under consideration; and, g is the energy growth parameter for that unit. Then, the energy-delay product (EDP) for the particular unit would vary as:

$$EDP_{unit} = \frac{IPC_{unit} * (IW)^g}{IPC_{overall}} \quad \text{..... (4.1)}$$

Zyuban shows that for real machines, where the overall IPC always increases with issue width in a sub-linear manner, the overall EDP of the processor can be bounded as:

$$(IPC)^{2g-1} \leq EDP \leq (IPC)^{2g} \quad \text{..... (4.2)}$$

where g is the energy-growth factor of a given unit and IPC refers to the overall IPC of the processor; and, IPC is assumed to vary as $(IW)^{0.5}$. Thus, according to this formulation, superscalar implementations that minimize g for each unit or structure will result in energy-efficient designs. The eliot/elpaso tool does not model the effects of multi-clustering in detail yet; however, from Zyuban's work, we can infer that a carefully designed multicluster architecture has the potential of extending the power-performance efficiency scaling beyond what is possible using the classical superscalar paradigm. Of course, such extended scalability is achieved at the expense of reduced IPC performance for a given superscalar machine width. This IPC degradation is caused by the added inter-cluster communication delays and other power management overhead in a real design. Some of the IPC loss (if not all) can be offset by a clock frequency boost which may be possible in such a design, due to the reduced resource latencies and bandwidths.

Current high-performance processors (for example, the Compaq Alpha 21264 and the IBM Power4) certainly have elements of multi-clustering, especially in terms of duplicated register files and distributed issue queues. Zyuban proposed and modeled a specific multicluster organization in his work. This simulation-based study determined the optimal number of clusters and their configurations, for the EDP metric.

*Simultaneous Multithreading (SMT):*

Let us examine the SMT paradigm [27] to understand how this may affect our notion of power-performance efficiency. The data in Table 3 shows the steady-state utilization of some of the resources in our base super scalar machine in response to the input loop test case discussed earlier. Since, due to fundamental ILP limits, the IPC will not increase beyond W=4, it is clear why power-performance efficiency will be on a downward trend beyond a certain width of the machine. (Of course, here we assume maximum processor power numbers, without any clock gating or dynamic adaptation to bring down power).

With SMT, assume that we can fetch from two threads (simultaneously, if the icache is dual-ported, or in alternate cycles if the icache remains single-ported). Suppose two copies of the same loop program (see example at the beginning of this section, i.e. Section 4) are executing as two different threads. So, thread-1 instructions A-B-C-D-E-F-G and thread-2 instructions P-Q-R-S-T-U-V are simultaneously available for dispatch and subsequent execution on the machine. This facility allows the utilization factors, and the net throughput performance to go up, without a significant increase in the maximum clocked power. This is because, the issue width W is not increased, but the execution and resource stages or slots can be filled up simultaneously from both threads. The added complexity in the front-end, of maintaining two program counters (fetch streams) and the global register space increase alluded to before adds to the power a bit. On the other hand, the core execution complexity can be relaxed a bit without a performance hit. For example, the fp_delay parameter can be increased, to reduce core complexity, without any performance degradation. Figure 9 shows the expected performance and power-performance variation with W for the 2-thread SMT processor. The power model assumed for the SMT machine is the same as that of the underlying superscalar, except that a fixed fraction of the net power is added to account for the SMT overhead. (The fraction added is assumed to be linear in the number of threads, in an n-thread SMT). Figure 9 shows that under the assumed model, the performance-power efficiency scales better with W, compared with the base superscalar (Figure 8).

Seng and Tullsen [28] presented analysis to show that using a suitably architected SMT processor, the per-thread speculative waste can be reduced, while increasing the utilization of the machine resources by executing simultaneously from multiple threads. This was shown to reduce the average energy per instruction by 22 %.

*Chip Multiprocessing*

In a multiscalar-like chip-multiprocessor (CMP) machine [29] , different iterations of a single loop program could be initiated as separate tasks or threads on different core processors on the same chip. Thus, the threads A-B-C-D-E-F-G and P-Q-R-S-T-U-V, derived from the same user program would be issued in sequence by a global task sequencer to two cores, in a 2-way

multiscalar CMP. Register values set in one task are forwarded in sequence to dependent instructions in subsequent tasks. For example, the register value in fp1 set by instruction E in task 1 must be communicated to instruction T in task 2; so instruction T must stall in the second processor until the value communication has occurred from task 1. Execution on each processor proceeds speculatively, assuming the absence of load-store address conflicts between tasks; dynamic memory address disambiguation hardware is required to detect violations and restart task executions as needed. In this paradigm also, if the performance can be shown to scale well with the number of tasks, and if each processor is designed as a limited-issue, limited-speculation (low complexity) core, it is possible to achieve better overall scalability of performance-power efficiency.

Another trend in high-end microprocessors is true chip multiprocessing (CMP), where multiple (distinct) user programs execute separately on different processors on the same chip. A commonly used paradigm in this case is that of (shared memory) symmetric multiprocessing (SMP) on a chip (see Hammond et al. in [23]). Larger SMP server nodes can be built from such chips. Server product groups such as IBM's high-end PowerPC division have relied on such CMP paradigms as the scalable paradigm for the immediate future. The Power4 design [11] is the first example of this trend. Such CMP designs offer the potential of convenient coarse-grain clock-gating and "power-down" modes, where one or more processors on the chip may be "turned off" or "slowed down" to save power when needed.

## 5. Conclusions and Future Research

In this paper, we first discussed issues related to power-performance efficiency and metrics from an architect's viewpoint. We limited the discussion to dynamic power consumption. We showed that depending on the application domain (or market), it may be necessary to adopt one metric over another in comparing processors within that segment. Next, we described some of the promising new ideas in power-aware microarchitecture design. This discussion included circuit-centric solutions like clock-gating, where microarchitectural support is needed to make the right decisions at run time. Later, we used a simple loop test case to illustrate the limits of power-performance scalability in some popular paradigms that are being developed by various vendors within the high-end processor domain. In particular, we show that scalability of current generation super scalars may be extended effectively through multi-clustering, SMT and CMP. Our experience in simulating these structures points to the need of keeping a single core (or the uni-threaded core) simple enough to ensure scalability in the power, performance and verification cost of future systems. Detailed simulation results with benchmarks to support these conclusions were not provided in this tutorial-style paper. Future research papers from our group will present such data for specific microarchitectural paradigms of interest.

We limited our focus to a few key ideas and paradigms of interest in future power-aware processor design. Also, we did not consider methods to reduce static (leakage) power: a component of net processor power that is expected to grow significantly in future technologies. Many new ideas to address various aspects of power reduction have been presented in recent workshops (e.g. [30-32]). All of these could not be discussed in this paper; but the interested reader should certainly refer to the cited references for further detailed study.

At IBM T. J. Watson Research Center, a project on power-aware microprocessor design, led by the author of this article, is currently engaged in designing a research processor prototype,

called LPX (low-power issue-execute processor). This processor has many of the elements touched upon in this paper among other new innovations. By incorporating on-chip monitoring hardware, we shall attempt to measure power reduction benefits and performance degradations (if any) resulting from the various ideas that are being tried. This hardware prototype will also enable us to validate our power-performance simulation methodologies, e.g. the *PowerTimer* toolkit referred to in [2]. Details of the LPX design will be described in later publications.

## REFERENCES


1. Hennessey, J.L. and Patterson, D. A. *Computer Architecture: a Quantitative Approach*, 2nd. edition, Morgan Kaufmann Publishers, Inc. 1996.
2. Brooks, D. M., Bose, P, Schuster, S.E., Jacobson, H., Kudva, P. N., Buyuktosunoglu, A., Wellman, J-D., Zyuban, V., Gupta, M and Cook, P.W., "Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors," *Proc. IEEE Micro*, vol. 20, no. 6, pp. 26-44, November/December 2000.
3. Flynn, M.J., Hung, P and Rudd, K. "Deep-submicron microprocessor design issues," *Proc. IEEE Micro*, vol. 19, no. 4, pp. 11-22, July/August 1999.
4. Borkar, S, "Design challenges of technology scaling," *Proc. IEEE Micro*, vol. 19, no. 4, pp. 23-29, July/August 1999.
5. Gonzalez, R. and Horowitz, M., "Energy dissipation in general purpose microprocessors," *IEEE J. Solid-State Circuits*, vol. 31, no. 9, Sept. 1996, pp. 1277-1284.
6. Oklobdzija, V. G., "Architectural tradeoffs for low power," in *Proc. ISCA Workshop on Power-Driven Microarchitectures*, Barcelona, Spain, June 1998.
7. Kogge, P.M., *The Architecture of Pipelined Computers*, Hemisphere Publishing Corporation, 1981.
8. Larson, A.G., "Cost-effective processor design with an application to Fast Fourier Transform Computers," Digital Systems Laboratory Report SU-SEL-73-037, Stanford University, Stanford, Calif., August 1973; see also, Larson and Davidson, "Cost-effective design of special purpse processors: a Fast Fourier Transform Case Study," *Proc. 11th Ann. Allerton Conference on Circuits and System Theory*, University of Illinois, Champaihn-Urbana, pp. 547-557, 1973.
9. Dubey, P. K. and Flynn, M.J., "Optimal pipelining," *J. Parallel and Distributed Computing*, vol. 8, no. 1, Jan. 1990, pp. 10-19.
10. Tiwari, V. et al., "Reducing power in high-performance microprocessors," in *Proc. IEEE/ACM Design Automation Conference*, ACM, New York, 1998, pp. 732-737.
11. Diefendorff, K., "Power4 focuses on memory bandwidth," *Microprocessor Report*, Oct. 6, 1999, pp. 11-17.
12. Brooks, D and Martonosi, M. "Dynamically exploiting narrow width operands to improve processor power and performance," in *Proc. 5th Int'l. Symp. on High-Performance Computer Architecture (HPCA-5),* January 1999.
13. Albonesi,D., "Dynamic IPC/Clock Rate Optimization," in *Proc. 25th. Ann. Int'l. Symp. on Computer Architecture (ISCA),* pp. 282-292, Barcelona, 1998.

14. Maro, R., Bai, Y. and Bahar, R. I., "Dynamically reconfiguring processor resources to reduce power consumption in high-performance processors," in *Proc. Power Aware Computer Systems (PACS) Workshop*, held in conjunction with ASPLOS, Cambridge, MA, Nov. 2000.
15. Buyuktosunoglu, A. et al. "An adaptive issue queue for reduced power at high performance," in *Proc. ISCA Workshop on Complexity-Effective Design (WCED),* Vancouver, Canada, June 2000.
16. Cruz, J-L., Gonzalez, A., Valero, M. and Topham, N.P., "Multiple-banked register file architectures," in *Proc. Int'l. Symp. On Computer Architecture (ISCA),* pp. 316-325, Vancouver, June 2000.
17. Gunther, S. H., Binns, F., Carmean, D.M., Hall, J.C., "Managing the impact of increasing microprocessor power consumption," in *Proc. Intel Technology Journal*, March 2000.
18. Reed, P. et al. "250 MHz 5 W RISC microprocessor with on-chip L2 cache controller," in *Digest of Technical Papers, IEEE Int'l. Solid State Circuits Conference*, pp. 40:412, 1997.
19. Sanchez, H. et al. "Thermal management system for high performance PowerPC microprocessors," in *Digest of papers, IEEE COMPCON*, p. 325, 1997.
20. Brooks, D. and Martonosi, M., "Dynamic thermal management for high-performance microprocessors," in *Proc. 7th Int'l. Symp. On High Performance Computer Architecture*, pp. 20-24, January 2001.
21. Manne, S., Klauser, A., and Grunwald, D., "Pipeline gating: speculation control for energy reduction," in *Proc. 25th. Ann. Int'l. Symp. on Computer Architecture (ISCA),* pp. 132-141, Barcelona, 1998.
22. Grunwald, D., Klauser, A., Manne, S., and Pleszkun, A. "Confidence estimation for speculation control," in *Proc. 25th. Ann. Int'l. Symp. on Computer Architecture (ISCA),* pp. 122-131, Barcelona, 1998.
23. Theme issue, "The future of processors," *IEEE Computer*, vol. 30, no. 9, September 1997, pp. 97-93.
24. Bose, P., Kim, S., O'Connell, F. P. and Ciarfella, W. A., "Bounds modeling and compiler optimizations for superscalar performance tuning," *Journ. of Systems Architecture*, vol. 45, pp. 1111-1137, 1999; Elsevier Press.
25. Zyuban, V., "Inherently lower-power high performance super scalar architectures," Ph.D Thesis, Dept. of Computer Science and Engineering, University of Notre Dame, 2000.
26. Zyuban, V. and Kogge, P., "Optimization of high-performance superscalar architectures for energy efficiency," in *Proc. IEEE Symp. on Low Power Electronics and Design*, ACM, New York, 2000.
27. Tullsen, D. M., Eggers, S.J., Levy, H. M., "Simultaneous Multithreading: Maximizing On-Chip Parallelism," in *Proc. 22nd. Ann. Int'l. Symp. on Computer Architecture*, 1995, pp. 292-403.
28. Seng, J. S., Tullsen, D. M., and Cai, G., "The power efficiency of multithreaded architectures," invited talk presented at: ISCA Workshop on Complexity-Effective Design (WCED), Vancouver, Canada, June 2000.
29. Sohi, G., Breach, S. E., and Vijaykumar, T. N., "Multiscalar Processors," in *Proc. 22nd Ann. Int'l. Symp. on Computer Architecture,* IEEE CS Press, Los Alamitos, Calif. 1995, pp. 414-425.
30. Talks presented at the 1998 ISCA Workshop on Power-Driven Microarchitecture: http://www.cs.colorado.edu/~grunwald/LowPowerWorkshop.

31. Talks presented at the 2000 ISCA Workshop on Complexity Effective Design (WCED-00), Vancouver, Canada, June 2000: http://www.ece.rochester.edu/~albonesi/WCED00/.
32. Talks presented at the Power Aware Computer Systems (PACS) Workshop, held in conjunction with ASPLOS, Cambridge, MA, November 2000.

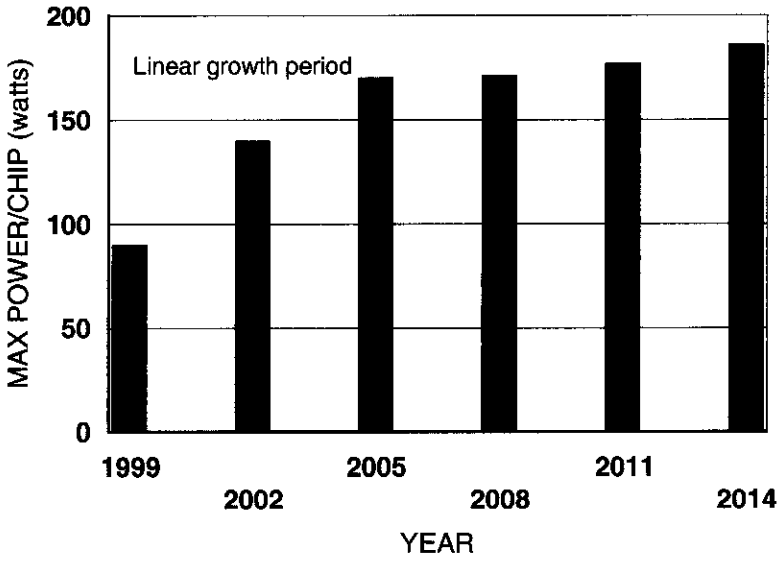



Figure 1. Maximum chip power projection (ITRS roadmap)

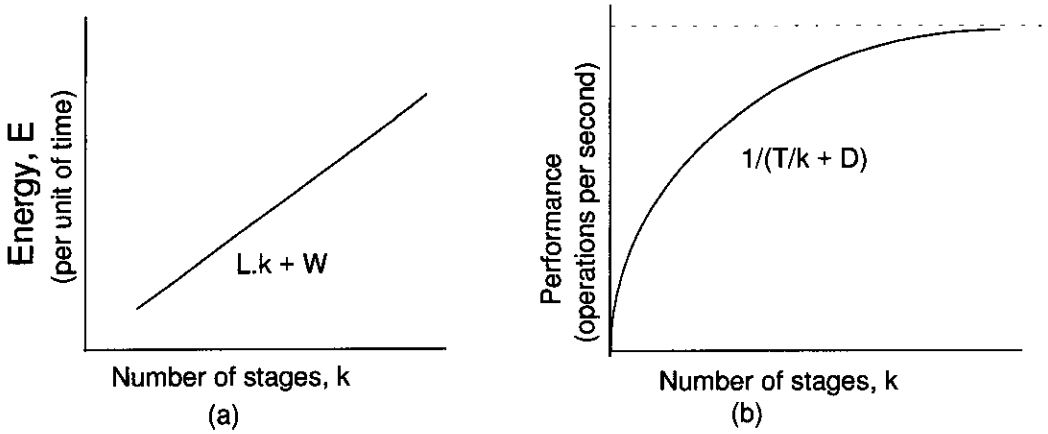
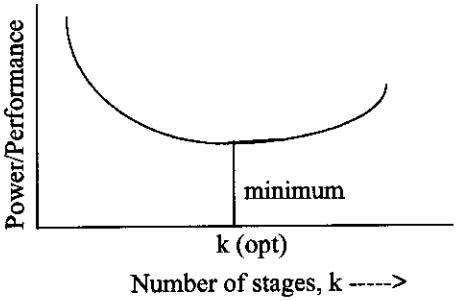



Figure 2. Power and performance curves for idealized pipeline flow


Power/Performance
minimum
k (opt)
Number of stages, k ----->
Figure 3. Power-performance ratio curve for idealized pipeline flow

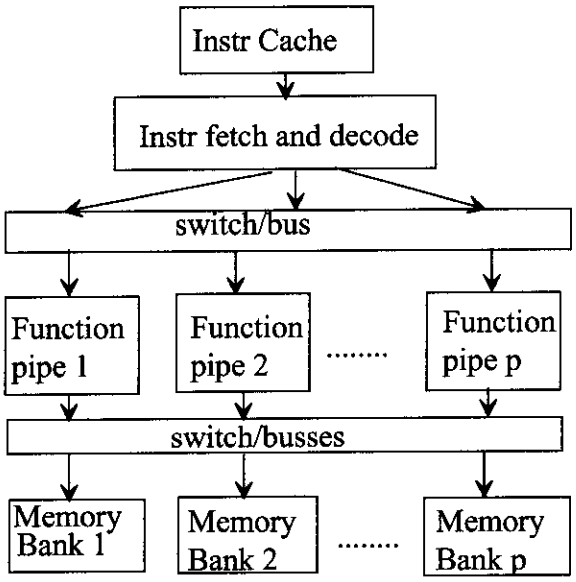



Figure 4. Parallel SIMD architecture

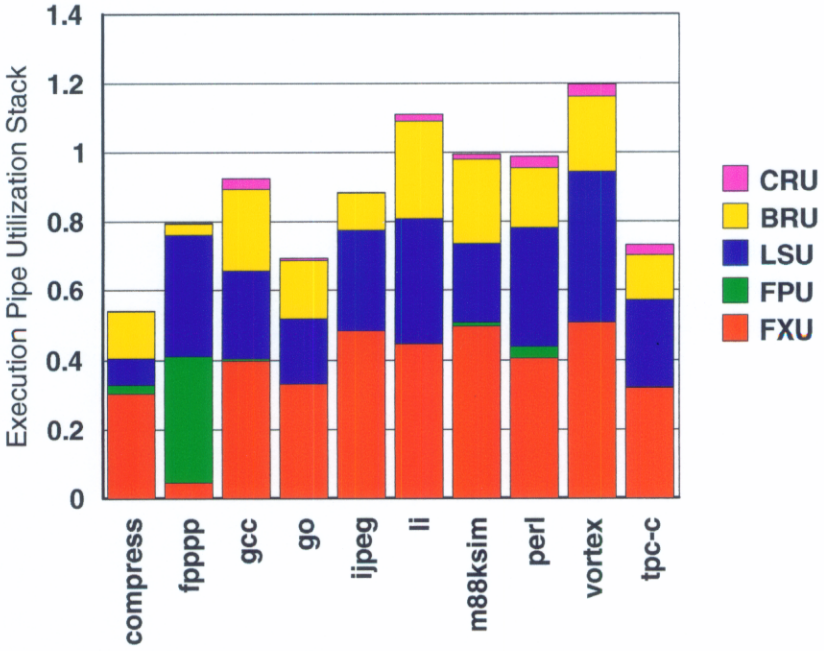



Figure 5. Execution Pipe Utilization Component Stack Across Workloads

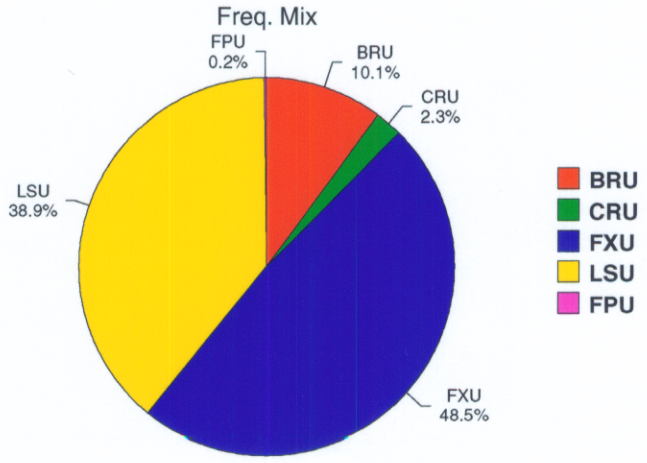



Figure 6(a) Instruction frequency mix for a typical commercial workload trace segment

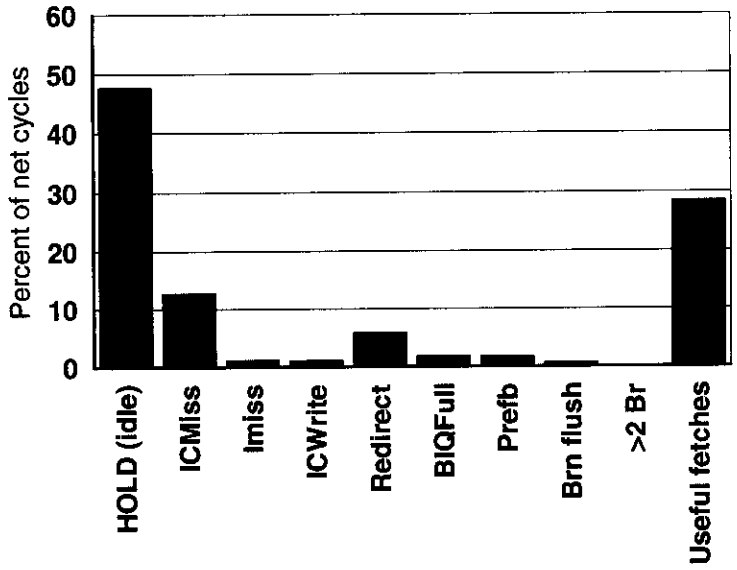



Figure 6(b). Stall profile in the instruction fetch unit (IFU) for the commercial workload

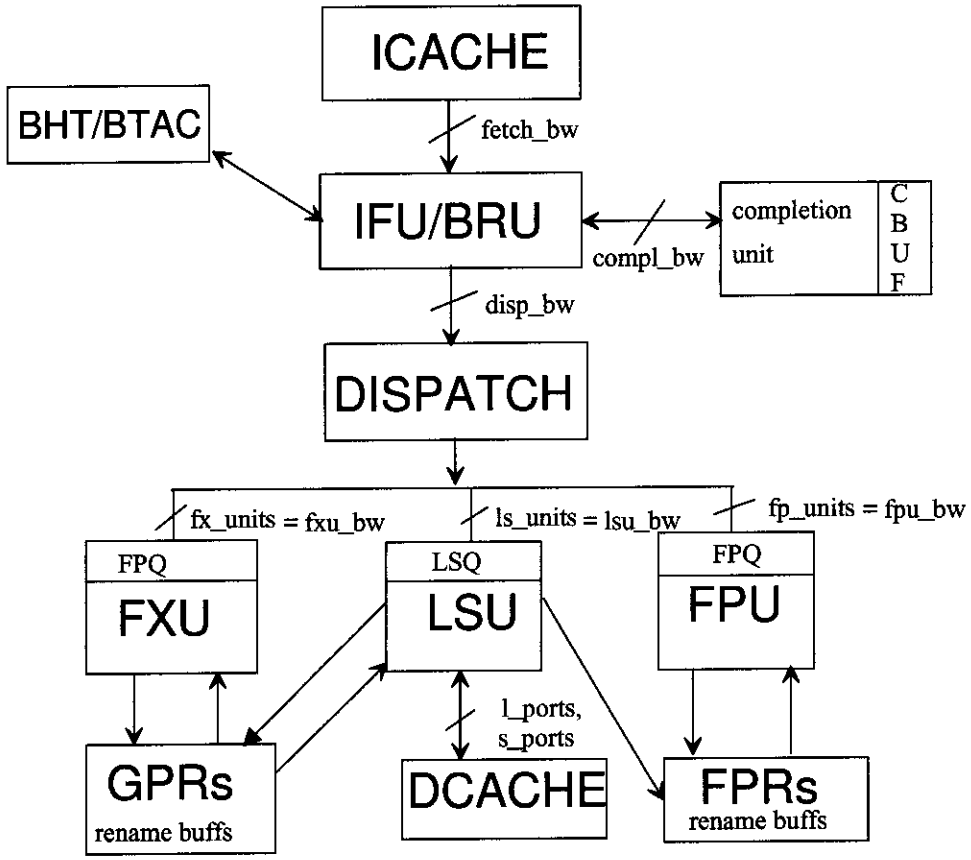
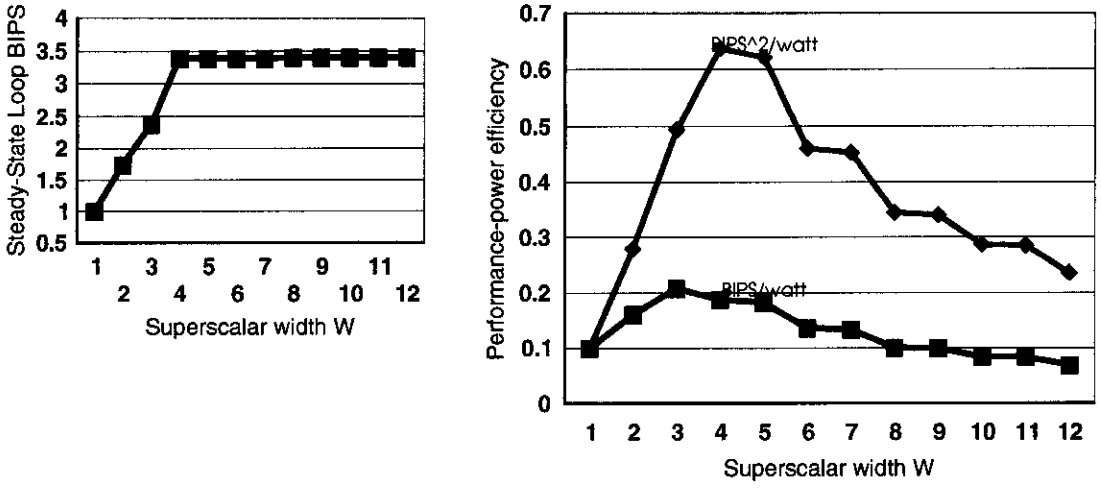
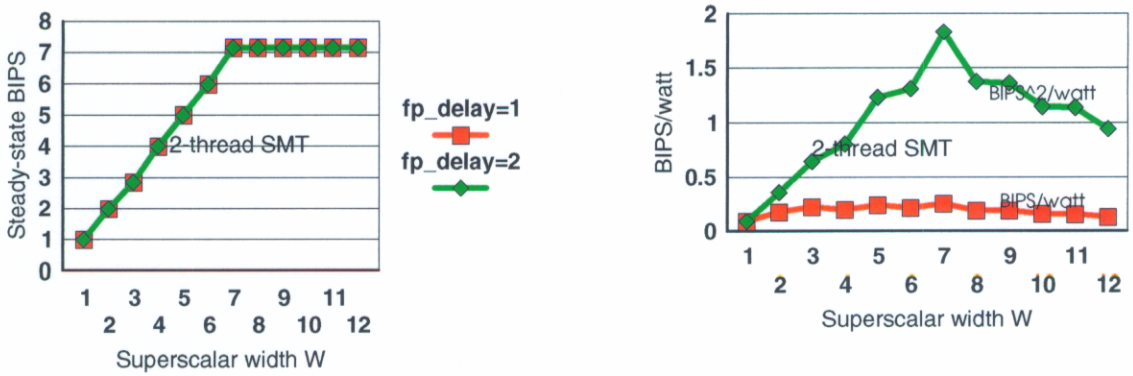



Figure 7. High-level block-diagram of machine organization modeled in the eliot/elpaso tool.




(a) (b)

Figure 8. Loop performance and performance/power variation with issue width




(a) (b)

Figure 9. Performance and power-performance variation with W for 2-thread SMT