RC22362 (W0203-022) March 8, 2002
Computer Science

IBM Research Report

Power-Performance and Power Swing Characterization in Adaptive Microarchitectures

Pradip Bose, David M. Brooks, Viji Srinivasan, Philip G. Emma
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Power-Performance and Power Swing Characterization in Adaptive Microarchitectures

Pradip Bose, Viji Srinivasan, David Brooks, Phil Emma
IBM T. J. Watson Research Center

ABSTRACT
In this paper, we present an analysis of some of the fundamental power-performance tradeoffs in processors that employ adaptive techniques to vary sizes, bandwidths, clock-gating modes and clock frequencies. Initial expectations are set using simple analytical reasoning models. Later, simulation-based data is presented in the context of a simple, low-power super scalar processor prototype (called LPX) that is currently under development as a test vehicle. There are three fundamental issues that we attempt to address in this paper: (a) Does dynamic adaptation - in clocking or microarchitectural resources - help extend the power-performance efficiency range of wider-issue superscalars ? (b) What factors of power and power-density reductions are within practical reach in future adaptive processors ? (c) Does the presence of dynamic adaptation modes cause unacceptably large, worst-case power (or current) swings in affected sub-units ?

Keywords
Power, performance, scalability, adaptive, microarchitecture

1. Introduction:
Power and power-density limits constitute one of the primary design constraints in future high performance processors. In current CMOS technologies, dynamic ("switching") power still dominates; but, increasingly, the static ("leakage") component is threatening to become a major *- or even the dominating* - component in future technologies [1].

Current generation high-end processors like the IBM POWER4^{TM}[2,3] are performance-driven designs where power limits are still comfortably below the 0.5 watts/sq. mm. power density limit afforded by the package/cooling solution of choice in the server markets targeted by such processors. In designing and implementing future processors (or even straight "remaps") the power (and especially the power-density) limits could become a potential "show-stopper" as the areas shrink and the frequencies keep increasing.

As such, techniques like clock-gating (e.g. [4, 5]) and dynamic size adaptation of on-chip resources like cache and queues (e.g. [6-10]) are now being actively examined as candidate power management methods during the early-stage microarchitectural definition of future processor cores. Many of these techniques are relatively new in the server-class processor design world, and aspects like reliability and inductive noise on the power supply rails (Ldi/dt) have not been properly assessed prior to committing a particular gating or adaptation technique to a real design.

In this paper, we present an analysis of the fundamental trade-offs confronting the designer during early-stage microarchitecture definition of a future, high performance, power-efficient processor. At this stage of the design, the exact organization and parameters of the target processor are not known. As such, a custom, cycle-accurate power-performance simulator (e.g. the PowerTimer tool described in [11]) for the full machine is often not available or relevant.

We focus on the following early-stage design issues in this paper:

- *Pipeline Depth*: We wish to understand the fundamental tradeoffs in power and performance when the pipeline depth (and hence the operating frequency) of the target machine is varied.
- *Instruction Issue Width*: We wish to study the basic issues of power-performance scalability within the current generation superscalar paradigm.
- *Adaptive Structures:* A popular approach in proposed new architectures is one where a resource size, latency or bandwidth is dynamically adapted to fit the requirements of the input application (workload). We wish to understand the effect of selected ideas within this approach, on the power-performance scalability criterion. In addition, we wish to quantify the worst-case power (current) swing problem in such architectures.
- *Adaptive Clocking:* Power consumption can be reduced by conditional clocking (i.e. clock-gating) or by reducing clock-frequencies in regions that are not performance-critical. Such mechanisms can be employed as an alternative to (or in addition to) the "adaptive structures" paradigm. We wish to compare the power-performance benefits and the current swing problems afforded by the two adaptive methods.

We present an initial analysis based on a simple model of power and performance for basic pipelines and superscalars. Such a view enables us to understand the fundamental tradeoffs and scalability issues in the power-performance behavior of superscalar pipelines. Subsequently, we validate some of the expectations from analytical reasoning via "simulation in the small" experiments in LPX [19]: a low power issue-execute processor prototype currently under development. The LPX power-performance simulator allows us to run application-based and synthetic test kernels in trying to understand the fundamental benefits and pitfalls of some of the key ideas of interest.

2. Analytical Reasoning Models
In this section, we attempt to set up some simple analytical models to understand the fundamental tradeoffs and scalabilitly limits in power-efficient pipelines and superscalar extensions. Later, in section 3, we report LPX simulation-based

results to validate the expectations derived from such analytical reasoning.

Optimal Pipeline Depth

A fundamental question that is asked in very early-stage definition studies has to do with pipeline depth. Is a deeply pipelined, high frequency ("speed demon") design better than an IPC-centric lower frequency ("braniac") design? For the purposes 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 [16], Chapter 2)

$$t = T/k + D \quad ....(2.1)$$

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) in the logic is W and the corresponding energy spent per level of staging latches is L, then the energy per cycle for the k-stage pipelined version is roughly

$$E = L.k + W \quad .... (2.2)$$

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 consider the effect of clock-gating). 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 a power-delay product ("watt/mips" [18]) based power-performance efficiency, we compute the ratio:

$$\text{Power/Performance} = (L.k + W)(T/k + D)$$
$$= L.T + W.D + (L.D.k^2 + W.T)/k \quad .... (2.3)$$

Figure 1 shows the general shape of this curve as a function of k. Differentiating the right hand side of expression in (2.3) 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 1 can be shown to occur when

$$K(\text{opt.}) = \sqrt{((W.T)/L.D))} \quad .... (2.4)$$

Larson and Davidson (see reference in [16]) first published this kind of analysis, albeit from a *cost/performance* perspective. The 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 a current generation (~0.18 micron technology) floating point unit (or complex multiply-add arithmetic unit) typical (albeit approximate) values are: T = 7.5 ns, D = 0.15 ns and W = 0.15 watts, L = 0.1 watts. This yields a k(opt.) = sqrt(75) = 8.67 ~= 8 (rounded down).

Note that an energy-delay product formulation ("watts/mips^2" [18]) yields an equation that results in a larger k(opt.) for the same pipline; but the basic behavior is roughly the same: i.e., k(opt.) increases with W or T and decreases with L or D. For real super scalar machine pipelined units, however, the number of latches tends to go up much more sharply with k than the linear assumption above. This would have the effect of making k(opt.) smaller. Also, in real pipeline flow with hazards, e.g. in the presence of branch and cache-miss related stalls, performance actually peaks at a certain value of k before

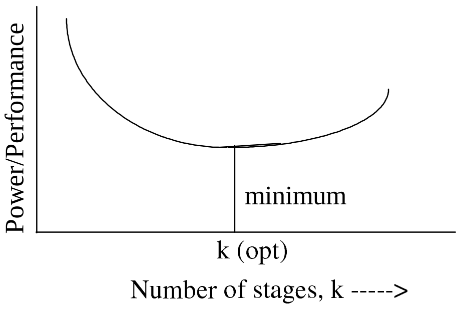


Figure 1. Optimal hazard-free pipeline depth.

decreasing [17], instead of the asymptotically increasing behavior implied by equation 2.1. This causes a further reduction in k(opt.). above. So, for the arithmetic unit pipe example, the k(opt.) value of 8 is the best-case upper bound. Assuming a pipe stall frequency of b, the new form of equation 2.3 would be:

$$\text{Power/Performance}$$
$$= (L.k + W)(T/k + D)(1 + (k-1).b) \quad .... (2.3a)$$

assuming, as in [17] that the effect of a pipe stall has the effect of invalidating k-1 instructions.

Figure 2 shows the variation of k(opt.) with b, for the previously assumed values of T = 7.5 ns, D = 0.15 ns and W = 0.15 watts, L = 0.1 watts. We see that for the pipeline flow and energy model assumed, k(opt.) decreases rather sharply with the stall frequency b. For processors that predict conditional branches, branch-related stalls may themselves contribute as much as 0.04 to the value of b (1 branch per 5 instructions and 80 % prediction accuracy) in certain commercial workloads. So, for the example cited, the real k(opt.) may turn out to be a number around 4. A purely clock-frequency (Ghz)-driven, power-unaware design, therefore, may lead to the choice of a deeply pipelined machine operating at an inherently power-inefficient design point.

Assume now that the pipeline has fine-grain clock-gating support, with a "valid" bit propagated with the

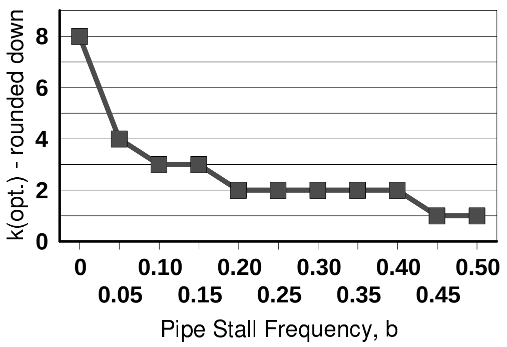


Figure 2. Decrease of k(opt.) with stall frequency

flow of data. The latch-set associated with a given pipeline stage is clocked only if the coresponding valid bit is ON. Then, the average, effective value of L will now be reduced to u*L, where, u < 1, is a fraction that relates directly to the average "valid data" utilization of the pipeline. Thus, presence of fine-grain clock-gating will have the effect of extending k(opt.), i.e. the power-performance scalability, of a simple pipeline. Adapting the microarchitectural resource sizes dynamically also causes a reduction in the average, effective value of L; and, therefore, a similar extension of scalable power-performance.

Super Scalar Processing

Let us model the IPC (or mips) performance of a super scalar processor as follows:

Perf = $K_1 - K_2 * (W)^{-A}$ .... (2.5)

Here, K_1 and K_2 are constants, W is the issue width and A > 0 is a parameter (constant) that controls the manner in which Perf increases towards the asymptotic value of K_1. We will refer to A as the "performance exponent." Figure 3 illustrates the Perf versus W behavior, for $K_1 = 3.4$ and $K_2 = 2.4$ for various values of A < 1. The greater the value of A, the sharper is the rise of the Perf curve towards the upper bound of 3.4. So, the parameter A essentially tries to quantify the degree of complexity (e.g. speculation and out of order execution modes) in the microarchitecture. The general *behavior* predicted by equation 2.5, agrees with experimental evidence and with data produced using other analytical models of IPC performance, e.g. Zyuban et al. [15].

The power consumption, similarly, is modeled as:
Power = $K_3 * (W)^B$ .... (2.6)
where, K_3 is a constant and B is a parameter (constant) that can be called the "power exponent."

Figure 4 shows the variation of Perf/Power with W, for A = 0.5, and across various values of B. (K_3 is, arbitrarily, assumed to have a value of 0.1). This figure shows that only for rather small values of the power exponent, B, the

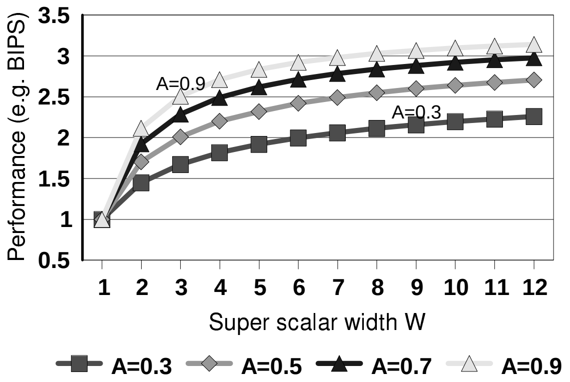


Figure 3. IPC performance versus W (analytical model)

performance-power efficiency scales to large values of W. For values of B greater than 0.35 or so, the efficiency seems to peak at a small value of W, between 2 and 3. This is even more clearly illustrated in Figure 5, where W(opt.) is plotted against B. The equation for W(opt.) is obtained by differentiating the ratio of expressions in 2.5 and 2.6, with respect to W and setting it to 0. W(opt.) is governed by the following equation:

log W(opt.) = (1/A). log $[(A+B).K_2/(B.K_1)]$ .... (2.7)

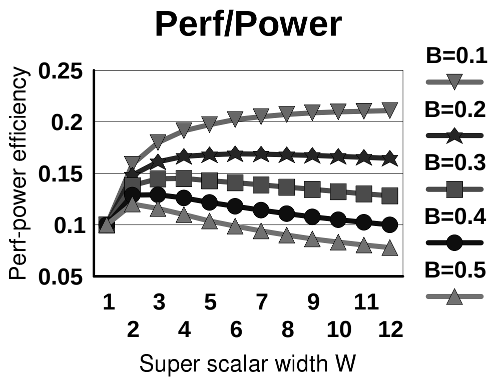


Figure 4. Perf/Power efficiency variation with W

As seen from Figure 5, irrespective of the value of the performance exponent A, W(opt.) seems to settle to a low value (less than 3) beyond B > 0.35 or so.

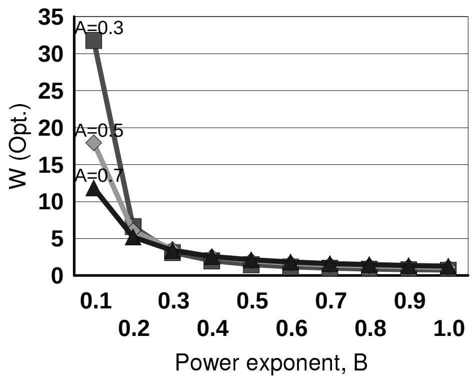


Figure 5. Variation of optimal width W (opt.) with B

Note that without any form of clock-gating, the power exponent B is expected to be high, i.e. greater than 0.5. For such cases, typical of many current generation high-end, server-class processors, super scalar scalability, in terms of Perf/Power efficiency, is limited to small values (like 2 or 3). With the presence of fine-grain, valid-bit based clock-gating, or with the use of adaptive resource sizes, power will grow slower with increase in W; and, so if B is architected to yield a really small value (like < 0.2), then the super scalar paradigm can be made to scale to larger values of W. The more aggressive the microarchitecture (e.g. with greater degrees of out-of-order and speculative execution), the performance exponent A is larger, resulting in diminished scalability.

Power swing characteristics:

As stated at the beginning, one of the potential negative impacts of adaptive clocking and resizing is the problem of current (or power) fluctuations. Such instantaneous current spikes is known to cause inductive noise (Ldi/dt effect) on the power supply rails. Unless kept within control by the effective use of decoupling capacitors, such noise can seriously impact circuit reliability. Hence, it is important to be able to quantify the worst-case swings during the early-stage microarchitecture definition.

For the simplest, linear pipeline flow example, the worst-case current swing occurs when a fully "gated off" pipelined unit gets progressively "gated on" with the transmission of a "valid" bit. The worst-case cycle-to-cycle current increase is the amount drawn by the most power-hungry pipeline stage. Assuming equal power per stage, the worst case power spike would be L, the average power per set of latches driving a pipeline stage. As a fraction of maximum possible power in the pipeline, this power swing is therefore 1/k, where k is the number of pipeline stages. As a fraction of average power, the swing is 1/(u*k), where u is the average utilization due to valid data. Since $u < 1$, for a minimally utilized pipeline, the worst-case power swing may be substantial. For example, if the floating point pipe has a value of $k = 6$, then the worst-case power swing in the pipelined unit could amount to (1/6)th or 16.7 % of maximum power (with $u = 1$); but with respect to average power, assuming $u = 0.3$, the worst-case swing could be a staggering 550 %. In general, such swing characteristics are expected to be smaller with larger pipeline depths (k) and larger utilizations, u.

This analysis points to some interesting tradeoffs in designs that employ fine-grain (i.e., pipeline stage-level) clock-gating. Here, a small pipeline utilization is clearly desirable to ensure a big reduction in average power. However, the smaller the utilization, the larger is the instantaneous power swing as a percentage of the average power consumption. Microarchitectural techniques to improve the pipe utilizations will increase IPC performance, at the cost of increasing the average power, but with the benefit of reduced power swings. Thus, early-stage design definition studies must carefully balance the choices to ensure acceptable performance within the constraints of power consumption and swing limits that are considered feasible.

In the case where adaptive resource resizing is used, the worst-case power swing is determined by the maximum "chunk size" of adaptation. For example, if a 32-entry issue queue is adapted in chunk sizes of 4 entries, then the maximum cycle-to-cycle power swing for that unit is the maximum power consumed by a 4-entry chunk of the issue queue; causing a swing of roughly 4/32 or 12.5 % (in the absence of additional, fine-grain clock-gating).

3. Experimental Results

In this section we present a snap-shot of our simulation-based results to understand the effects of clock-gating and dynamic resizing on super scalar power-performance characteristics. The experimentally derived results are interpreted in the light of earlier "fundamental" tradeoff analysis.

The LPX super scalar processor model

Figure 6 shows a very high-level block diagram of the baseline, parameterized LPX processor model. The "fetch-and-issue" sub-units act together as a *producer* of instructions, which are *consumed* by the "execute" sub-unit. The design attempts to balance the dynamic complexity of the producer-consumer pair with the goal of maximizing performance, while minimizing power consumption. LPX (details described in [19]) is a research prototype under development by a small industry-academia team. LPX serves as a test vehicle for simulation, design and measurement "in the small". The power-performance and power swing characterisics obtained in the LPX project are used to understand the basic tradeoffs in adaptive design. This understanding will help our product-level design teams to be better aware of fundamental tradeoffs, issues and pitfalls during the early-stage design definition of future processors. The hardware measurements will also help validate some of our energy models and the underlying modeling methodology.

For the nominal design point, the issue width is W = 2. Correspondingly, the nominal instruction fetch bandwidth, ifetch_bw is 4. One of the functional units is the scalar FXU (a combined load-store unit and integer unit) and the other is the (optionally vector or SIMD) pipelined arithmetic unit, ARU. In scalar mode, the ARU enables a 32-bit wide datapath; in SIMD mode, the datapath width is 32x4 = 128. The AR register file (not shown in detail) is, for the nominal W=2 scalar case, a 3-read-port, 2-write-port array unit. The number of ports scale as needed for maximal performance support, when W is increased in the LPX simulator. The ARU execution pipe is multi-cycle (nominally 4 cycles). The scalar FXU has a 1-cycle pipe plus (nominally) a 1-cycle (infinite) data cache access for loads and stores. At the end of the final execution stage, the results are latched on to the result bus while the target register tags are broadcast to the instructions pending in the centralized issue queue. The issue queue can operate in *in-order* or full *out-of-order* mode.

Using the LPX simulator[1], experiments on adaptive clocking and resizing can be performed; and within each such adaptive mode experiment, W can be fixed to one of four values: 2, 4, 6 and 8. (*Again, in the actual LPX implementation, W is 2).* As W is adjusted, the maximum resource (e.g. queue/buffer) sizes and bandwidths (e.g. Ifetch bandwidth and issue bandwidth) are also adjusted. The mapping relations used to obtain the maximum resource sizes are:
Instruction fetch bandwidth, *ifetch_bw* = 2*W
Dispatch/decode/rename bandwidth, *disp_bw* = W
Issue bandwidth, *iss_bw* = W
Completion bandwidth, *compl_bw* = W
Instruction buffer size, *ibuf_size* = 2*W
Issue queue size, *issueq_size* = 4*W
Number of scalar LSFX units = W/2
Number of AR units = W/2.

The above mapping relations used in scaling the issue width W (see also, [20]) are based on historical data of super

[1] The simulator has integrated energy models developed from scaled, circuit-simulation based detailed energy data obtained from prior high performance processor projects like POWER4. See description of PowerTimer tool in [11].

scalar RISC processors and/or bandwidth matching arguments and expectations where applicable. We illustrate the use of simple loop-based test cases in understanding the basic power-performance trade-offs of adaptive structures and clocking mechanisms that were chosen for study in LPX. The challenge is to figure out the nominal sizes, adaptation windows and (in each case) a simple "monitor-and-control" mechanism that is appropriate in the context of building a small prototype measurement engine, like LPX. We started with the simplest baseline, where infinite (perfect) cache effects were modeled, by architecting a single-stage LSFX pipe unit; but, we later augmented the specification to include a variable-length LSFX pipe, to simulate data cache miss latency. The choice of what latency to assume, depends on the cache hit/miss scenario. In the absence of real cache hardware (correspondingly, real cache hit/miss code in the simulator), we architect for programmable "miss" scenarios via a user-loadable miss specification register (msr). Details of how this works in the real hardware are not discussed in this initial submission. For brevity, we only show a few example tradeoff analysis examples, mostly limited to the infinite (perfect) cache scenario.

Example loop test-case: vect_add
A simple "vector add" loop trace, formed by execution of the following loop:

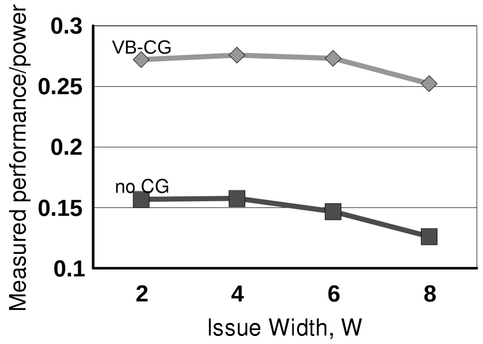
```
 | -> VLD    vr1,   r2 (0x4)   (* vr1 is the target register *)
 |    VADD   vr4,   vr1, vr6   (* vr4 is the target register *)
 |    VLD    vr6,   r2 (0x8)
 |    VADD   vr4,   vr4,  vr6
 |    VST    vr4,   r3 (0x8)   (* vr4 is the register stored *)
 |    DEC    r7                (* decrements count reg r7 *)
---   BRZ    r7,   -0x7        (* conditional branch back *)
```

A 160-iteration vect_add loop trace is run in vector mode, where the VLD instruction loads a 32x4 bit register (vr) using a scalar base address register specifier, following standard PowerPC™ architecture conventions.

Optimal Pipeline Depth:

The general behavior predicted by Figures 1 and 2 were easily validated by varying various pipe lengths in the LPX simulator. However, to conserve space, we omit this particular data, in lieu of the more interesting issues of scalability, adaptation and power swing characterization.

Issue Width Scalability:



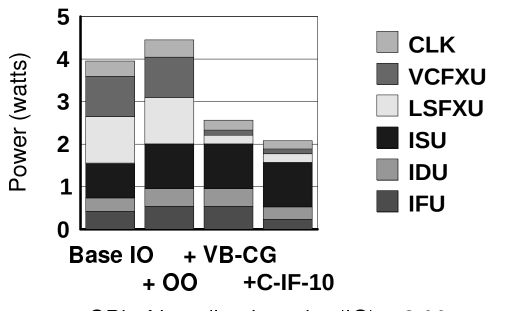
ıg

As W is varied, the measured performance and power characteristics agreed with the general trends modeled by equations 2.5-2.7. An example result is shown in Figure 7 where the effect of valid-bit based clock-gating is shown to extend the scalable range of W from 4 to 6 for the LPX processor running the vect_add loop trace in infinite cache mode. Note that since the vect_add loop trace presents only dependence-related stalls in infinite cache runs, the scalable range of W observed is rather large. With branch and cache miss stalls present in more realistic simulations (i.e. intra-loop branches and cache misses) the scalable range for non-clock-gated operation was barely more than 2 (as expected from Figure 3).

For the baseline (W=2) LPX engine targeted for actual implementation, the following two adaptive resizing mechanisms are studied in this paper:

Conditional ifetch:

Gating off the ifetch process using a hardware heuristic to compute the gating condition, is a viable approach to saving energy [12, 13]. For LPX, we wish to experiment with the simplest of such heuristics, that are easy to implement. The basic method used is to employ the "stall" or "impending stall" signals available from "downstream" consumer units to throttle back the "upstream" producer (ifetch). Such stall signals are easy to generate and are usually available in the logic design anyway. Figures 8 and 9 show results from an illustrative use of conditional ifetch, using the following simple hardware heuristic for determining the ifetch gating scenario. When a "stall" signal is asserted by the instruction buffer (e.g. when the ibuffer is full) the ifetch process is naturally inhibited in most designs; so this is assumed in the baseline model. However, additional power savings can be achieved by retaining the "ifetch-hold" condition for a fetch-gate cycle window, W, beyond the negation of the ibuffer stall signal. Since the ibuffer was full, it would take a while to drain it; hence ifetch could be gated off for W cycles. Depending on the size of the ibuffer, IPC performance would be expected to drop off to unacceptable levels beyond a certain value of W; but increasing W is expected to reduce IFU (instruction fetch unit) power and overall chip power.



CPI of baseline in-order (IO) = 3.00
CPI of all the other out-of-order (OO) = 2.29
Figure 8. LPX power: effect of clock-gating

Adaptive Issue Queue

For LPX, we started with a baseline design of the POWER4 out-of-order integer issue queue [3], which is a latch-based design. The LPX issue queue is structured as a 2-chunk structure, where in adaptive mode, one of the chunks can be shut-off completely. Figure 10 illustrates the benefit of using a simple, LPX-specific adaptive issue queue heuristic that is targeted to reduce power, without appreciable loss of performance. The design and adaptation heuristic illustrated is simpler than proposed in the detailed studies reported earlier [8], for ease of implementation in the LPX context. The control heuristic in LPX is as follows:

```
if (current_cycle_window_issuecount < 0.5 * last_cycle_window_issuecount) then
    increase_size (* if possible*)
else decrease_size (* if possible *);
```

Discussion of results (adaptive microarchitecture experiments)

From Figure 8, we note that adding out-of-order (oo) mode to the baseline in-order (io) machine causes a performance increase (CPI decrease) of 23.6 %, but with a 12.5 % overall power increase. The ISU, which contains the issue queue, increases in power by 27.5 %. So, from an overall power-performance efficiency viewpoint, including the out-of-order (oo) mode does seem to pay off in LPX for this loop trace, in infinite cache mode. However, from a power-density "hot-spot" viewpoint (issue queue region) even this basic enhancement may need to be carefully evaluated with a representative workload suite. Adding the valid-bit-based clock-gating (VB-CG) mode in the instruction buffer, issue queue and the execution unit pipes, causes a sharp (42.4 %) decrease in power from the baseline oo design point. Adding a conditional ifetch mode, (with a cycle window W of 10 cycles over which ifetch is blocked after the ibuffer stall signal goes away) yields an additional 18.8 % power reduction, without loss of IPC performance. As the gating cycle window W is increased, we see a further sharp decrease in net power beyond W=10, but with IPC degradation (see Figure 9). For the adaptive issue queue experiment (Figure 10) shown, we see that a 8 % reduction in net LPX power is possible; but beyond an adaptation cycle window, AW of 1, a 11 % increase in CPI is incurred. Thus, use of fine-grain, valid-bit based clock-gating is the simpler and more effective than adaptive methods. Detailed results, combining VB-CG and adaptation will be reported later.

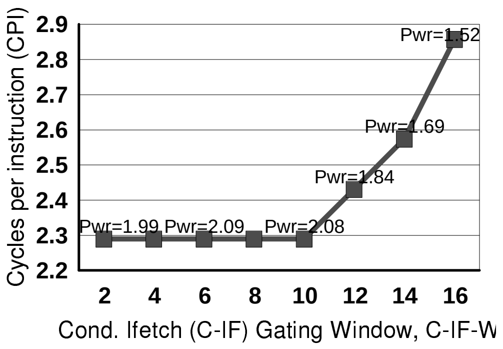


Figure 9. C-IF gating window versus CPI

Power Swing Measurements

Figure 11 shows an example measurement of power swing characteristics using the LPX simulator, for the vect_add loop trace. As seen from the graphs, as the average ARU pipe utilization increases, the worst-case power swing problem diminishes. The general trend is in agreement with earlier analytic expectations. In the graph, we plot the worst-case cycle-to-cycle power swing observed in the issue queue (set to out-of-order mode) and for the full chip. The ARU pipe utilization is controlled by varying the (statistical) cache miss ratio parameter. Although the slope of the "full chip" plot is much greater than that of the "issue queue" plot, it should be noted that the worst-case swing is limited to a couple of cycles at the beginning of the run for the "full chip" case; while, for the issue queue, the worst-case swings occur quite frequently over the simulation run.

4. Conclusions and Future Work

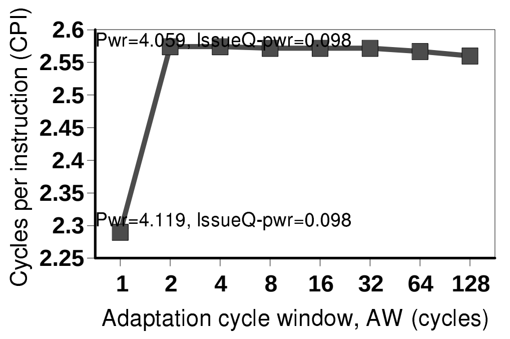
We discussed simple analytical formulations to set up initial, early-stage expectations for power-performance characteristics (including worst-case power swings) in pipelined, super scalar processors. Subsequently, we reported example, loop trace driven simulation results to understand the fundamental trade-offs in more detail and to verify the scalable growth of power-performance efficiency with the introduction of clock-gating and adaptive resizing. The experiments were done in the context of the early-stage design phase of LPX: a low power issue-execute processor prototype under development [19].

REFERENCES

1. S. Borkar, "Design challenges of technology scaling," IEEE Micro, vol. 19, no. 4, July/Aug. 1999, pp. 23-29.
2. C. Anderson et al., "Physical design of a fourth generation POWER Ghz microprocessor, *ISSCC 2001 Digest of Tech. Papers,* Feb. 2001, p. 232.
3. J. M. Tendler, S. Dodson, S. Fields, H. Le and B. Sinharoy, "POWER4 System Microarchitecture," http://www-1.ibm.com/servers/eserver/pseries/hardware/whitepapers/power4.pdf, Oct. 2001.
4. J. M. Rabaey and M. Pedram, ed., Low Power Design Methodologies, pp. 279-281, Kluwer, 1996.
5. M. Gowan, L. Biro and D. Lackson, "Power considerations in the design of the Alpha 21264 microprocessor," *Proc. ACM/IEEE Design Automation Conference (1998),* pp. 726-731.
6. D. H. Albonesi, "The inherent energy efficiency of complexity-effective processors," *Proc. ISCA Workshop on Power-Driven Microarchitecture,* June 1998.
7. R. Balasubramonian, D. H. Albonesi, A. Buyuktosunoglu and S. Dwarkadas, "Memory hierarchy reconfiguration for energy and performance in general purpose architectures," *Proc. 33rd. Int'l. Symp. on Microarchitecture,* pp. 245-257, December 2000.
8. A. Buyuktosunoglu et al., "An adaptive issue queue for reduced power at high performance," *Proc. Workshop on Power-Aware Computer Systems, (PACS'00),* held in conjunction with ASPLOS, November 2000.
9. D. Ponomarev, G. Kucuk and K. Ghose, "Dynamic allocation of datapath resources for low power," *Proc.*

Workshop on Complexity-Effective Design (WCED-01), held in conjunction with ISCA, June/July 2001;

10. D. Folegnani and A. Gonzalez, "Energy-effective issue logic," *Proc. ISCA-01,* pp. 230-239, June 2001.
11. D. Brooks et al., "Power-aware microarchitecture: design and modeling challenges for the next-genration microprocessors," *Proc. IEEE Micro*, vol. 20, no. 6, pp. 26-44, Nov./Dec. 2000.
12. S. Manne, A. Klauser and D. Grunwald, "Pipeline Gating: Speculation Control for Energy Reduction," *Proc. Int'l. Symp. on Computer Architecture (ISCA),* June 1998.
13. T. Karkhanis et al. "Saving energy with just-in-time instruction delivery," under submission for publication.
14. S. Schuster, W. Reohr, P. Cook, D. Heidel, M. Immediato and K. Jenkins, "Asynchronous interlocked pipelined CMOS operating at 3.3-4.5 Ghz.," *Proc. Int'l. Solid State Circuits Conference (ISSCC-2000),* pp. 292-293.
15. V. Zyuban and P. Kogge, "Optimization of high performance super scalar architectures for energy efficiency," in *Proc. IEEE Symp. On Low Power Electronics and Design (ISLPED),* 2000.
16. P. Kogge, The Architecture of Pipelined Computers, Hemisphere Publishing Corp., 1981.
17. M. J. Flynn et al., "Deep-submicron microprocessor design issues," *Proc. IEEE Micro*, pp. 11-22, July-Aug 1999.
18. V. Zyuban, "Unified architecture level energy-efficiency metric," submitted to GLSVLSI-2002 conference.
19. P. Bose et al., "Early-stage definition of LPX, a low-power issue-execute processor prototype," submitted for publication; IBM Research Report, Jan. 2002.
20. D. Brooks et al., "Power-aware microarchitecture: design and modeling challenges for next generation microprocessors," *Proc. IEEE Micro*, vol. 20., no. 6, Nov./Dec. 2000, pp. 26-44.



baseline power (non-adaptive) = 4.46 watts

Figure 10. LPX power-perf: adaptive issue queue

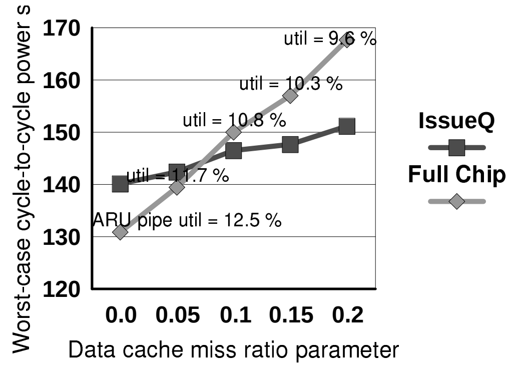


Figure 11. Power swing characteristics

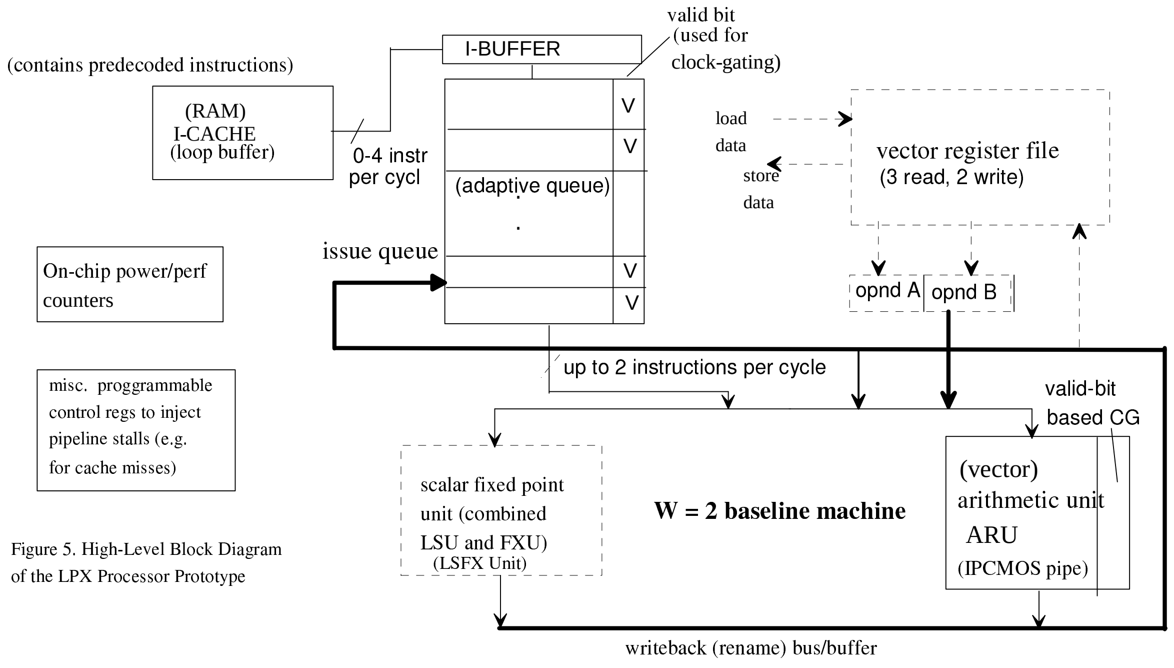


Figure 5. High-Level Block Diagram of the LPX Processor Prototype

Figure 6. LPX high-level block diagram