Early-Stage Definition of LPX: A Low Power Issue-Execute Processor

P. Bose[1], D. Brooks[1], A. Buyuktosunoglu[2], P. Cook[1], K. Das[3], P. Emma[1], M. Gschwind[1], H. Jacobson[1], T. Karkhanis[4], P. Kudva[1], S. Schuster[1], J. Smith[5], V. Srinivasan[1], V. Zyuban[1], D. Albonesi[6], and S. Dwarkadas[6]

[1] IBM T. J. Watson Research Center, Yorktown Heights, NY
pbose@us.ibm.com
[2] University of Rochester, NY; summer intern at IBM Watson
[3] University of Michigan, Ann Arbor; summer intern at IBM Watson
[4] University of Wisconsin, Madison; summer intern at IBM Watson
[5] University of Wisconsin, Madison; visiting scientist at IBM Watson
[6] University of Rochester, NY

Abstract. We present the high-level microarchitecture of LPX: a low-power issue-execute processor prototype that is being designed by a joint industry-academia research team. LPX implements a very small subset of a RISC architecture, with a primary focus on a vector (SIMD) multimedia extension. The objective of this project is to validate some key new ideas in power-aware microarchitecture techniques, supported by recent advances in circuit design and clocking.

1 Introduction

Power dissipation limits constitute one of the primary design constraints in future high performance processors. Also, depending on the thermal time constants implied by the chosen packaging/cooling technology, on-chip power-density is a more critical constraint than overall power in many cases. In current CMOS technologies, dynamic ("switching") power still dominates; but, increasingly, the static ("leakage") component is threatening to become a major component in future technologies [6]. In this paper, we focus primarily on the dynamic component of power dissipation.

Current generation high-end processors like the IBM POWER4™ [3, 26], are performance-driven designs. In POWER4, power dissipation is still comfortably below the 0.5 watts/sq. mm. power density limit afforded by the package/cooling solution of choice in target server markets. However, 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 transistors shrink and the frequency keeps increasing.

Techniques like clock-gating (e.g. [21, 13]) and dynamic size adaptation of on-chip resources like caches and queues (e.g. [1, 20, 4, 9, 12, 2, 15, 27]) have been either used or proposed as methods for power management in future processor cores. Many of these techniques, however, have to be used with caution in

B. Falsafi and T.N. Vijaykumar (Eds.): PACS 2002, LNCS 2325, pp. 1–17, 2003.

server-class processors. Aspects like reliability and inductive noise on the power supply rails (Ldi/dt) need to be quantitatively evaluated prior to committing a particular gating or adaptation technique to a real design.

Another issue in the design of next generation, power-aware processors, is the development of accurate power-performance simulators for use in early-stage design. University research simulators like Wattch [7] and industrial research simulators like Tempest [10] and PowerTimer [8] have been described in the recent past; however their use in real design environments is needed to validate the accuracy of the energy models in the context of power-performance tradeoff decisions made in early design.

In the light of the above issues, we decided to design and implement a simple RISC "sub-processor" test chip to validate some of the key new ideas in adaptive and gated architectures. This chip is called: LPX, which stands for low-power issue-execute processor. This is a research project, with a goal of influencing real development groups. LPX is a joint university-industry collaboration project. The design and modeling team is composed of 10-12 part-time researchers spanning the two groups (IBM and University of Rochester) aided by several graduate student interns and visiting scientists recruited from multiple universities to work (part-time) at IBM. LPX is targeted for fabrication in a CMOS 0.1 micron high-end technology. RTL (VHDL) simulation and verification is scheduled for completion in 2002. Intermediate circuit test chips are in plan (mid- to late 2002) for early validation of the circuit and clocking support. LPX chip tapeout is slated for early 2003. In this paper, we present the microarchitecture definition with preliminary simulation-based characterization of the LPX prototype. We summarize the goals of the LPX project as follows:

- To understand and assess the true worth of a few key ideas in power-aware microarchitecture design through simulation and eventually via direct hardware measurement. Based on known needs in real products of the future, we have set a target of average power density reduction by at least a factor of 5, with no more than 5% reduction in architectural performance (i.e. instructions per cycle or IPC).
- To quantify the instantaneous power (current) swings incurred by the use of the adaptive resizing, throttling and clock-gating ideas that are used to achieve the targeted power reduction factors in each unit of the processor.
- To use the hardware-based average and instantaneous power measurements for calibration and validation of energy models used in early-stage, power-performance simulators.

Clearly, what we learn through the "simulation and prototyping in the small" experiments in LPX, will be useful in influencing full-function, power-efficient designs of the future. The calibrated energy models will help us conduct design space exploration studies for high-end machines with greater accuracy. In this paper, we limit our focus to the microarchitectural definition process, with related simulation-based result snapshots, of the LPX prototype. (Note that LPX is a research test chip. It is not intended to be a full-function, production-quality

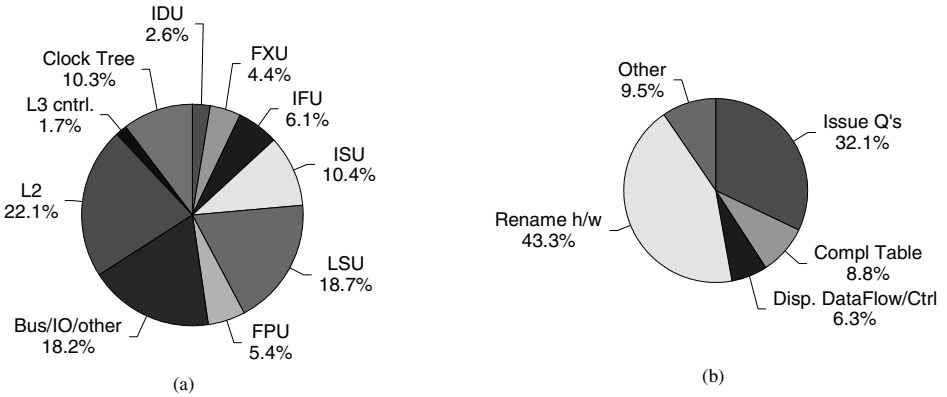

Fig. 1. Power profile: (a) relative unit-wise power; (b) power breakdowns: ISU

microprocessor. At this time, LPX is not directly linked to any real development project).

2 Background: Power-Performance Data

In an out-of-order, speculative super scalar design like each of the two cores in POWER4, a large percentage of the core power in the non-cache execution engine is spent in the instruction window or issue queue unit [26, 9, 20, 12]. Figure 1(a) shows the relative distribution of power across the major units within a single POWER4 core. Figure 1(b) zooms in on the instruction sequencing unit that contains the various out-of-order issue queues and rename buffers.

Figure 2 shows the power density across some of the major units of a single POWER4 core. The power figures are non-validated pre-silicon projections based on unconstrained (i.e. without any clock-gating assumptions) "average/max" power projections using a circuit-level simulation facility called CPAM [19]. (Actual unit-wise power distribution, with available conditional clocking modes enabled, are not shown). This tool allowed us to build up unit-level power characteristics from very detailed, macro-level data. Here, the activity (utilization) factors of all units are assumed to be 100% (i.e. worst case with no clock-gating anywhere); but average, expected input data switching factors (based on representative test cases run at the RTL level, and other heuristics) are assumed for each circuit macro. Such average macro-level input data switching factors typically range from 4-15%. (From Figure 2, we note that although on a unit basis, the power density numbers are under 0.5 watts/sq. mm., there are smaller hotspots, like the integer FX issue queue within the ISU that are above the limit in an unconstrained mode). (Legend for Figs. 1-2: IDU: instruction decode unit; FXU: fixed point unit; IFU: instruction fetch unit; BHT: branch history table;

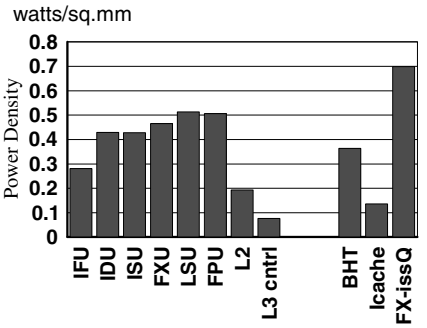

Fig. 2. Unconstrained power density profile

ISU: instruction sequencing unit; LSU: load-store unit: includes L1 data cache; FPU: floating point unit).

Another class of data that we used was the performance and utilization information obtained from pre-silicon performance simulators. Figure 3 shows the relative "active/idle" barchart plot across some of the major units for a POWER4-like pre-silicon simulation model. The data plotted is for a commercial TPC-C trace segment. This figure shows, for example, that the instruction fetch unit (IFU) is idle for approximately 47% of the cycles. Similar data, related to activities within other units, like issue queues and execution unit pipes were collected and analyzed.

3 Areas of Focus in Defining the LPX Processor

Based on microarchitecture level and circuit simulation level utilization, power and power-density projections, as above, we made a decision to focus on the following aspects of a super scalar processing core in our LPX test chip:

Power-efficient, Just-in-Time Instruction Fetch. Here, we wanted to study the relative advantages of conditional gating of the ifetch function, with a goal of saving power without appreciable loss of performance. The motivation for this study was clearly established after reviewing data like that depicted in Figures 1 and 2. In simulation mode, we studied the benefit of various hardware heuristics for determining the "gating condition" [18, 14, 5], before fixing on a particular set of choices (being reported in detail in [17]) to implement in LPX. Our emphasis here is on studying ifetch gating heuristics that are easy to implement and test, with negligible added power for the control mechanism.

Adaptive Issue Queues. The out-of-order issue queue structure inherent in today's high-end super scalar processors is a known "hot-spot" in terms of power dissipation. The data shown in Figures 1, 2, and also corroborative data from

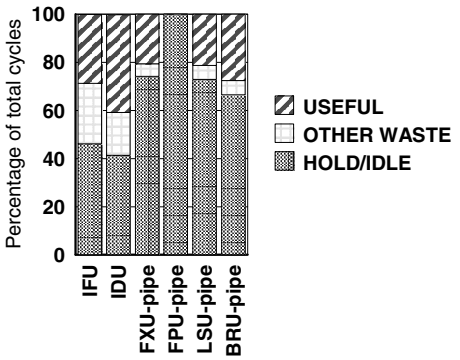

Fig. 3. Unit-wise utilization stack (TPC-C trace)

other processors (e.g. [1]), makes this an obvious area to focus on. In LPX, our goal is also to compare the achieved power savings with a fixed issue queue design, but with fine-grain clock-gating support, where the valid-bit for each issue queue entry is used as a clock-gating control. A basic issue in this context is the extra power that is spent due to the presence of out-of-order execution modes. Is the extra power spent worth the performance gain that is achievable? We wish to understand the fundamental power-performance tradeoffs in the design of issue queues for the next generation processors. Again, simplicity of the adaptive control and monitoring logic is crucial, especially in the context of the LPX prototype test vehicle.

Locally Clocked Execution Pipeline. Based on the data shown in Figures 1 and 2, a typical, multi-stage complex arithmetic pipeline is also a high power-density region within the chip. We wish to study the comparative benefit of alternate conditional clocking methods proposed in ongoing work in advanced circuit design groups ([21, 23, 16]). In particular, we wish to understand: (a) the benefit of simple valid-bit-based clock-gating in a synchronously clocked execution unit; and (b) the added power-savings benefit of using a locally asynchronous arithmetic unit pipeline, within a globally synchronous chip. The asynchronously clocked pipeline structure is based on the IPCMOS circuit technology previously tested in isolation [23] by some in our research team. Such locally clocked methods offer the promise of low power at high performance, with manageable inductive noise (Ldi/dt) characteristics. In LPX, we wish to measure and validate these expectations as the IPCMOS pipe is driven by data in real computational loop kernels.

Power-Efficient Stalling of Synchronous Pipelines. In the synchronous regions of the design, we wish to quantify the amount of power that is consumed by pipeline stall (or "hold/recirculation") conditions. Anticipating (from circuit

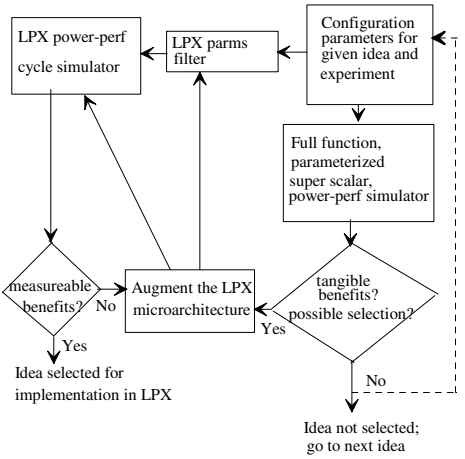

Fig. 4. Methodology for fine-tuning the LPX microarchitecture

simulation coupled with microarchitectural simulation data) such wastage to be significant, we wish to experiment with alternate methods to reduce or eliminate the "stall energy" by using a novel circuit technique called interlocked synchronous pipelines (ISP) that was recently invented by some members of our team [16].

Thus, a basic fetch-issue-execute super scalar processing element (see sections 4 and 5) was decided upon as the study vehicle for implementation by our small research team. The goal is to study the power-performance characteristics of dynamic adaptation: in microarchitectural terms as well as in clocking terms with the target of achieving significant power (and especially, power density) reduction, with acceptable margins of IPC loss.

4 Tuning the Microarchitecture

In this section, we outline the methodology adopted for defining the range of hardware design choices to be studied in the LPX testchip. Since we are constrained by the small size of our design team, and yet the ideas explored are targeted to influence real, full-function processor designs, we adopted the following general method.

Figure 4 shows the iterative method used to decide what coarse-level features to add into the LPX test chip, starting from an initial, baseline "bare-bones" fetch-issue-execute model.

– A given, power-efficient microarchitectural design idea is first simulated in the context of a realistic, current generation super scalar processor model (e.g. POWER4-like microarchitectural parameters) and full workloads (like SPEC and TPC-C) to infer the power-performance benefit. Once a basic

hardware heuristic is found to yield tangible benefit - in other words, a significant power reduction, at small IPC impact - it is selected for possible implementation in LPX.
– A detailed, trace-driven, cycle-by-cycle simulator for the baseline LPX processor is coded to run a set of application-based and synthetic loop test cases designed to test and quantify the LPX-specific power-performance characteristics of the candidate hardware power-saving feature. In order to get a measurable benefit, it may be necessary to further simplify the heuristic, or augment the microarchitecture minimally to create a new baseline. Once the power-performance benefit is deemed significant, we proceed to the next candidate idea.

In this paper we mainly focus on (b) above: i.e. understanding the fundamental power-performance tradeoff characteristics, using a simple, illustrative loop test case. However, we also refer briefly to example, full-model super scalar simulation results to motivate the choice of a particular hardware heuristic.

Energy Models Used. The LPX cycle-by-cycle simulator used to analyze early stage microarchitectural power-performance tradeoffs has integrated energy models, as in the PowerTimer tool [8]. These energy models were derived largely from detailed, macro-level energy data for POWER4, scaled for size and technology to fit the requirements of LPX. The CPAM tool [19] was used to get this data for most of the structures modeled. Additional experiments were performed at the circuit simulation level, to derive power characteristics of newer latch designs (with and without clock- and stall-based gating). The energy model-enabled LPX simulator is systematically validated using specially architected testcases. Analytical bounds modeling is used to generate bounds on IPC and unit-wise utilization (post-processed to form power bounds). These serve as reference "signatures" for validating the power-performance simulator. Since the LPX design and model are still evolving, validation exercises must necessarily continue throughout the high-level design process. Details of the energy model derivation and validation are omitted for brevity.

5 High-Level Microarchitecture of LPX

Figure 5 shows a very high-level block diagram of the baseline LPX processor that we started with before further refinement of the microarchitectural features and parameters through a simulation-based study. The function and storage units shown in dashed edges are ones that are modeled (to the extent required) in the simulation infrastructure, but are not targeted for implementation in the initial LPX design. The primary goal of this design is to experiment with the fetch-issue-execute pipe which processes a basic set of vector integer arithmetic instructions. These instructions are patterned after a standard, 4x32 bit SIMD multimedia extension architecture [11] but, simplified in syntax and semantics. The "fetch-and-issue" sub-units act together as a producer of instructions, which

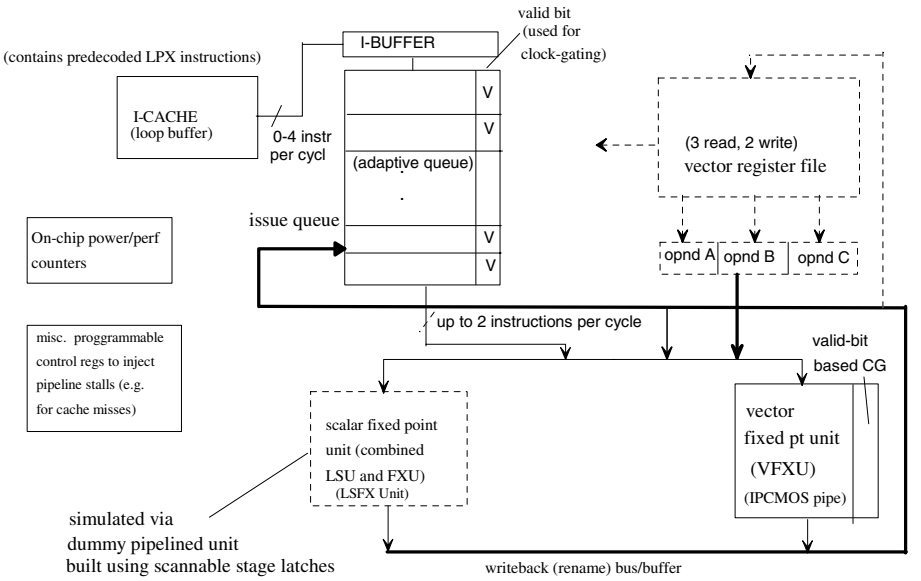

Fig. 5. LPX Processor: High-Level Block Diagram

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.

The basic instruction processing pipeline is illustrated in Figure 6. The decode/dispatch/rename stage, which is shown as a lumped, dummy dispatch unit in Figure 5, is actually modeled in our simulator as an m-stage pipe, where m=1 in the nominal design point. The nominal VFXU execute pipe is n=4 stages deep. The LSFX execute pipe is p=2 stages (in infinite cache mode) and p=12 stages when a data cache miss stall is injected using the stall control registers (Figure 5); in particular, using a miss-control register (MCR).

One of the functional units is the scalar FXU (a combined load-store unit and integer unit, LSFX) and the other is the vector integer unit (VFXU). The VFXU execution pipe is multi-cycle (nominally 4 cycles). The LSFX unit has a 1-cycle pipe plus (nominally) a 1-cycle (infinite) data cache access cycle 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 issue queue.

As a substitute for instruction caching, LPX uses a loop buffer in which a loop (of up to 128 instructions) is pre-loaded prior to processor operation. The loaded program consists of pre decoded instructions, with inline explicit specifiers of pre renamed register operands - in full out-of-order mode of execution. This avoids the task of designing explicit logic for the instruction decode and rename

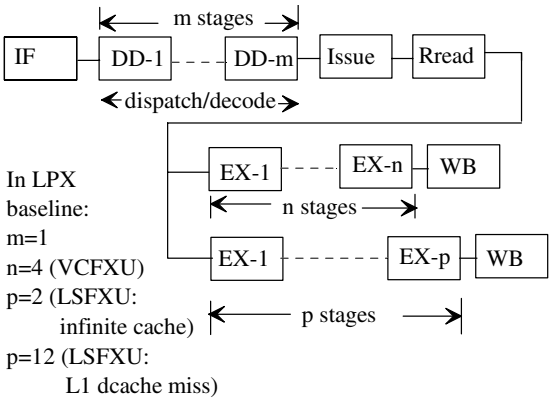

Fig. 6. LPX (simulator) pipeline stages

processes. LPX also supports an "in-order" mode, without register renaming as the lowest performance design point for our tradeoff experiments.

The instructions implemented in LPX are listed below in Table 1. For the most part, these are a set of basic vector (SIMD) mode load, store and arithmetic instructions, following the general semantics of a PowerPC™VMX (vector multimedia extension) architecture [11]. There are a few added scalar RISC (PowerPC-like) instructions to facilitate loading and manipulation of scalar integer registers required in vector load-store instructions. The (vector) load and store instructions have an implied "update" mode in LPX, where the scalar address base register is auto-incremented to hold the address of the next sequential array data in memory.

Table 1. LPX Instruction Set

	Example Syntax	Semantics
Vector Load	VLD vr1, r2, 0x08	Load vr1. Scalar base address register: r2
Vector Store	VST vr1, r2, 0x08	Store vr1.
Vector Add	VADD vr1, vr2, vr3	vr1 <– vr2 + vr3
Vector Sub, Mul, Div instructions: similar to VADD above		
Scalar Load	LD r1, r2, 0x08	Load scalar reg r1
Scalar Inc	INC r1	Increment r1 (scalar)
Scalar Dec	DEC r1	Decrement r1 (scalar)
Cond. Branch	BC +-0x08	Branch conditional (PC relative jump)
Uncond. Branch	BR +-0x08	Branch unconditional

6 Examples: LPX Microarchitecture Analysis

In this section, 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 determine 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 engine, like LPX.

We started with the simplest baseline, where ideal cache effects were modeled, by architecting a single-stage LSFX pipe unit; but, later we had to augment the specification to include a variable-length LSFX pipe, to simulate data cache miss latency. 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 control register (MCR). Details of how this works in the real hardware are not discussed in this paper. For brevity, we only show a few example tradeoff analysis examples limited to the infinite (ideal) cache scenario.

As described before in section 4 (see Figure 4), each candidate power reduction idea is analyzed in the "large" (i.e. using a general out-of-order super scalar simulator) to ensure potential benefit. Then, a simpler hardware heuristic is used for trial and measurement "in the small" within the LPX simulation tool kit.

LPX experiments: an example loop test-case: *vect_add*. We use a simple "vector add" loop trace, formed by execution of the following loop, to illustrate LPX tradeoff experiments:

```
  | -> VLD   vr1,  r2 (0x4)
  |    VADD  vr4,  vr1, vr6
  |    VLD   vr6,  r2 (0x8)
  |    VADD  vr4,  vr4, vr6
  |    VST   vr4,  r3 (0x8)
  |    DEC    r7
 ---   BRZ    r7,  -0x7
```

The baseline LPX model parameters were fixed as follows, after initial experimentation. Instruction fetch (ifetch) bandwidth is up to four instructions/cycle, with no fetch beyond a branch on a given cycle. The instruction fetch buffer size is four instructions; dispatch bandwidth (into the issue queue) is up to two instructions/cycle; issue bandwidth (into the execution pipes) is up to two instructions/cycle; and, completion bandwidth is also two instructions/cycle. Fetch and dispatch is in-order and issue can be in-order or out-of-order (switchable); instructions finish out-of-order. (LPX does not model or implement in-order completion for precise interrupt support using reorder buffers).

Conditional Ifetch. Figures 7(a,b) show a snapshot of analysis data from a typical 4-way, out-of-order super scalar processor model. The data reported is for two benchmarks from the SPECint2000 suite. It shows that the ifetch stage/buffer, the front-end pipe and the issue queue/window can be idle for

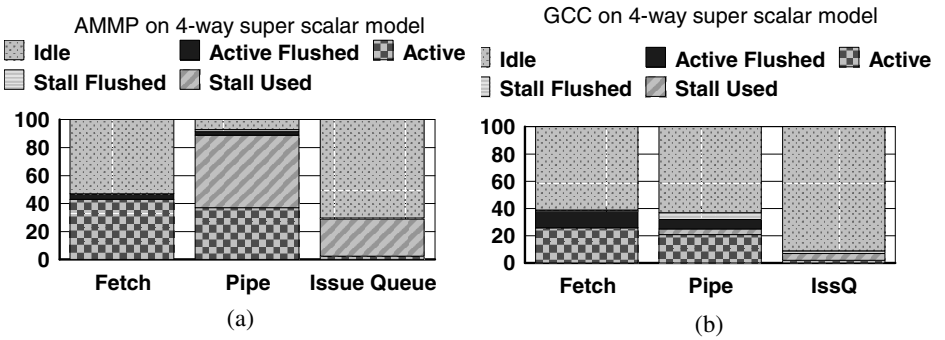

Fig. 7. Idle, speculative and stall waste: (a) AMMP and (b) GCC

significant fractions of the program run. These are cycles where power can be saved by valid-bit-based clock-gating. In addition, the fraction of cycles that are wasted by useful (but stalled) instructions and by incorrectly fetched speculative instructions can also be significant. Gating off the ifetch process using a hardware heuristic to compute the gating condition, is therefore a viable approach to saving energy.

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(a,b) show results from an illustrative use of conditional ifetch while simulating the *vect_add* loop trace.

We use 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, GW, 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 GW cycles. Depending on the size of the ibuffer, IPC would be expected to drop off to unacceptable levels beyond a certain value of GW; but increasing GW is expected to reduce IFU (instruction fetch unit) and overall chip power.

Adaptive Issue Queue. Figure 9 shows a snapshot of our generalized simulation-based power-savings projection for various styles of out-of-order issue queue design. (An 8-issue, super scalar, POWER4-like research simulator was used). These studies showed potential power savings of more than 80% in the issue queue with at most 2-3% hit in IPC on the average. However, the best power reductions were for adaptive and banked CAM/RAM based designs that are not

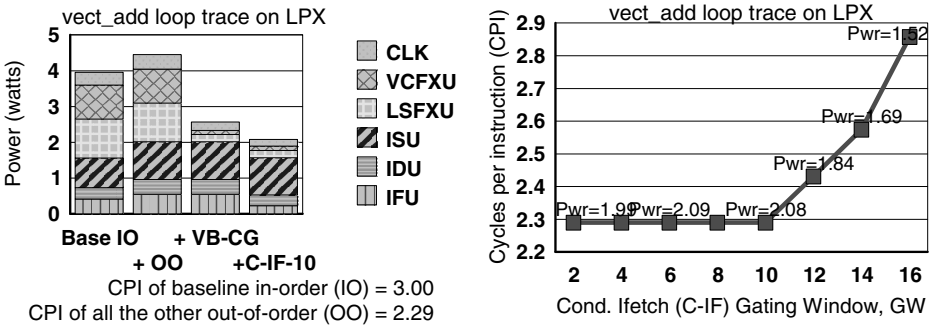

Fig. 8. Conditional ifetch in LPX: (a) power reduction and (b) CPI vs. gating window

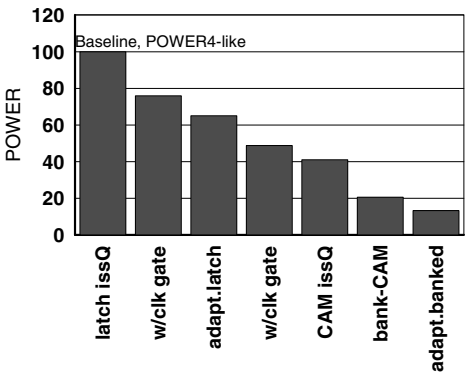

Fig. 9. Adaptive issue queue: power saving (generalized 8-issue super scalar)

easy to design and verify. For LPX, we started with a baseline design of the POWER4 integer issue queue [26], which is a latch-based design. It is structured as a 2-chunk structure, where in adaptive mode, one of the chunks can be shut-off completely (to eliminate dynamic and static power).

Figure 10 illustrates the benefit of using a simple, LPX-specific adaptive issue queue heuristic that is targeted to reduce power, without loss of performance; i.e. the size is adapted downwards only when "safe" to do so from a performance viewpoint; and the size is increased in anticipation of increased demand. (In the example data shown in this paper, we consider only the reduction of dynamic power via such adaptation). The adaptive issue queue control heuristic illustrated is simpler than proposed in the detailed studies reported earlier [9], for ease of implementation in the LPX context. The control heuristic in LPX is as follows:

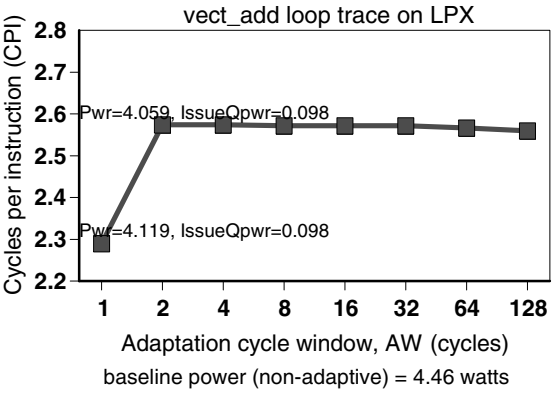

Fig. 10. Adaptive issue queue experiment in LPX

```
if (current-cycle-window-issuecount <
     0.5 *  last-cycle-window-issuecount)
then
        increase-size (* if possible *);
else
        decrease-size (* if possible *);
```

Discussion of Results. From Figure 8(a) we note that adding out-of-order (oo) mode to the baseline in-order (io) machine causes an IPC 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, 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, 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 decrease in power (42.4% 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. For the adaptive issue queue experiment (Fig. 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 (cycles-per-instruction) is incurred. Thus, use of fine-grain, valid-bit based clock-gating is simpler and more effective than adaptive methods. Detailed results, combining VB-CG and adaptation will be reported in follow-up research.

Stall-Based Clock-Gating. As previously alluded to, in addition to valid-bit-based clock-gating in synchronous (an locally asynchronous) pipelines, LPX uses a mode in which an instruction stalling in a buffer or queue for multiple cycles is clock-gated off, instead of a recirculation-based, hold strategy often used in high performance processors. The stall-related energy waste is a significant fraction of queue/buffer power that can be saved if the stall signal is avaliable in time to do the gating. Carefully designed control circuits [16] have enabled us to exploit this feature in LPX. In this version of the paper, we could not include the experimental results that show the additional benefits of such stall-based gating. However, suffice it to say, with the addition of stall-based clock-gating, simulations predict that we are well within the target of achieving a factor of 5 reduction in power and power density, without appreciable loss of IPC performance. The use of a locally asynchronous IPCMOS execution pipe [23] is expected to increase power reduction even further. Detailed LPX-specific simulation results for these circuit-centric features, will be available in subsequent reports.

7 Conclusions and Future Work

We presented the early-stage definition of LPX: a low-power issue-execute processor prototype that is designed to serve as a measurement and evaluation vehicle for a few new ideas in adaptive microarchitecture and conditional clocking. We described the methodology that was used to architect and tune simple hardware heuristics in the prototype test chip, with the goal of drawing meaningful conclusions of use in future products. We presented a couple of simple examples to illustrate the process of definition and to report the expected power-performance benefits of the illustrated adaptive features.

The basic idea of fetch-throttling to conserve power is not new. In addition to work that we have already alluded to [18, 14, 5], Sanchez et al. [22] describe a fetch stage throttling mechanism for the G3 and G4 PowerPC processors. The throttling mode in the prior PowerPC processors was architected to respond to thermal emergencies. The work reported in [18, 14, 5] and the new gating heuristics described in this paper and in [17] are aimed at reducing average power during normal operation. Similarly, the adaptive issue queue control heuristics being developed for LPX are intended to be simpler adaptations of our prior general work [9].

We believe that the constraint of designing a simple test chip with a small design team forces us to experiment with heuristics that are easy to implement with low overhead. If some of these heuristics help create relatively simple power management solutions for a full-function, production-quality processor, then the investment in LPX development will be easily justified.

In addition to the adaptive microarchitecture principles alluded to above, the team is considering the inclusion of other ideas in the simulation toolkit; some of these remain candidates for inclusion in the actual LPX definition: at least for LPX-II, a follow-on design. The following is a partial list of these other ideas:

- Adaptive, power-efficient cache and register file designs: these were not considered for implementation in the initial LPX prototype, due to lack of seasoned SRAM designers in our research team. In particular, as a candidate data cache design for LPX-II, we are exploring ideas that combine prior energy-efficient solutions [1, 4, 2, 15] with recently proposed, high performance split-cache architectures ([24, 25]).
- Exploiting the data sparseness of vector/SIMD-mode execution, through hardware features that minimize clocking waste in processing vector data that contains lots of zeroes.
- Newer features that reduce static (leakage) power waste.
- Adding monitoring hardware to measure current swings in clock-gated and adaptive structures.

Acknowledgement

The authors are grateful to Dan Prener, Jaime Moreno, Steve Kosonocky, Manish Gupta and Lew Terman (all at IBM Watson) for many helpful discussions before the inception of the LPX design project. Special thanks are due to Scott Neely (IBM Watson), Michael Wang and Gricell Co (IBM Austin) for access to CPAM and the detailed energy data used in our power analysis. The support and encouragement received from senior management at IBM - in particular Mike Rosenfield and Eric Kronstadt - are gratefully acknowledged. The authors would like to thank Joel Tendler (IBM Austin) for his comments and suggestions during the preparation and clearance of this paper. Also, the comments provided by the anonymous reviewers were useful in preparing an improved final version; this help is gratefully acknowledged.

The work done in this project at University of Rochester is supported in part by NSF grants CCR–9701915, CCR–9702466, CCR–9705594, CCR–9811929, EIA–9972881, CCR–9988361 and EIA–0080124; by DARPA/ITO under AFRL contract F29601-00-K-0182; and by an IBM Faculty Partnership Award. Additional support, at University of Wisconsin, for continuation of research on power-efficient microarchitectures, is provided by NSF grant CCR-9900610.

References

[1] D. H. Albonesi. The inherent energy efficiency of complexity-effective processors. In *Power-Driven Microarchitecture Workshop at ISCA25*, June 1998. 1, 5, 15

[2] D. H. Albonesi. Selective cache ways: on-demand cache resource allocation. In *Proceedings of the 32nd International Symposium on Microarchitecture (MICRO-32)*, pages 248–259, Nov. 1999. 1, 15

[3] C. Anderson et al. Physical design of a fourth-generation power ghz microprocessor. In *ISSCC Digest of Technical Papers*, page 232, 2001. 1

[4] R. Balasubramonian, D. Albonesi, A. Buyuktosunoglu, and S. Dwarkadas. Memory hierarchy reconfiguration for energy and performance in general purpose architectures. In *Proceedings of the 33rd International Symposium on Microarchitecture (MICRO-33)*, pages 245–257, Dec. 2000. 1, 15

[5] A. Baniasadi and A. Moshovos. Instruction flow-based front-end throttling for power-aware high performance processors. In *Proceedings of International Symposium on Low Power Electronics and Design*, August 2001. 4, 14
[6] S. Borkar. Design Challenges of Technology Scaling. *IEEE Micro*, 19(4):23–29, July-August 1999. 1
[7] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th International Symposium on Computer Architecture (ISCA-27)*, June 2000. 2
[8] D. Brooks, J.-D. Wellman, P. Bose, and M. Martonosi. Power-Performance Modeling and Tradeoff Analysis for a High-End Microprocessor. In *Power Aware Computing Systems Workshop at ASPLOS-IX*, Nov. 2000. 2, 7
[9] A. Buyuktosunoglu et al. An adaptive issue queue for reduced power at high performance. In *Power Aware Computing Systems Workshop at ASPLOS-IX*, Nov. 2000. 1, 3, 12, 14
[10] A. Dhodapkar, C. Lim, and G. Cai. TEM^{2}P^{2}EST: A Thermal Enabled Multi-Model Power/Performance ESTimator. In *Power Aware Computing Systems Workshop at ASPLOS-IX*, Nov. 2000. 2
[11] K. Diefendorff, P. Dubey, R. Hochsprung, and H. Scales. AltiVec extension to PowerPC accelerates media processing. *IEEE Micro*, pages 85–95, April 2000. 7, 9
[12] D. Folegnani and A. Gonzalez. Energy-effective issue logic. In *Proceedings of the 28th International Symposium on Computer Architecture (ISCA-28)*, pages 230–239, June 2001. 1, 3
[13] M. Gowan, L. Biro, and D. Jackson. Power considerations in the design of the Alpha 21264 microprocessor. In *35th Design Automation Conference*, 1998. 1
[14] D. Grunwald, A. Klauser, S. Manne, and A. Pleszkun. Confidence estimation for speculation control. In *Proceedings of the 25th International Symposium on Computer Architecture (ISCA-25)*, pages 122–31, June 1998. 4, 14
[15] K. Inoue et al. Way-predicting set-associative cache for high performance and low energy consumption. In *Proceedings of International Symposium on Low Power Electronics and Design*, pages 273–275, August 1999. 1, 15
[16] H. Jacobson et al. Synchronous interlocked pipelines. IBM Research Report (To appear in ASYNC-2002) RC 22239, IBM T J Watson Research Center, Oct. 2001. 5, 6, 14
[17] T. Karkhanis et al. Saving energy with just-in-time instruction delivery. submitted for publication. 4, 14
[18] S. Manne, A. Klauser, and D. Grunwald. Pipeline gating: Speculation control for energy reduction. In *Proceedings of the 25th International Symposium on Computer Architecture (ISCA-25)*, pages 132–41, June 1998. 4, 14
[19] J. Neely et al. CPAM: A Common Power Analysis Methodology for High Performance Design. In *Proc. 9th Topical Meeting on Electrical Performance of Electronic Packaging*, Oct. 2000. 3, 7
[20] D. Ponomarev, G. Kucuk, and K. Ghose. Dynamic allocation of datapath resources for low power. In *Workshop on Complexity Effective Design 2001 at ISCA28*, June 2001. 1, 3
[21] J. Rabaey and M. Pedram, editors. *Low Power Design Methodologies*. Kluwer Academic Publishers, 1996. Proceedings of the NATO Advanced Study Institute on Hardware/Software Co-Design. 1, 5
[22] H. Sanchez et al. Thermal management system for high performance PowerPC microprocessors. *Digest of Papers - COMPCON - IEEE Computer Society International Conference*, page 325, 1997. 14

[23] S. Schuster et al. Asynchronous interlocked pipelined CMOS operating at 3.3-4.5 GHz. In *ISSCC Digest of Technical Papers*, pages 292–293, February 2000. 5, 14

[24] V. Srinivasan. *Hardware Solutions to Reduce Effective Memory Access Time*. PhD thesis, University of Michigan, Ann Arbor, February 2001. 15

[25] V. Srinivasan et al. Recovering single cycle access of primary caches. submitted for publication. 15

[26] J. M. Tendler, S. Dodson, S. Fields, H. Le, and B. Sinharoy. POWER4 system microarchitecture. *IBM J. of Research and Development*, 46(1):5–26, 2002. 1, 3, 12

[27] S.-H. Yang et al. An energy-efficient high performance deep submicron instruction cache. *IEEE Transactions on VLSI, Special Issue on Low Power Electronics and Design*, 2001. 1