Power and Thermal Effects of SRAM vs. Latch-Mux Design Styles and Clock Gating Choices

Yingmin Li[†], Mark Hempstead[‡], Patrick Mauro[‡], David Brooks[‡], Zhigang Hu[††], Kevin Skadron[†]
[†] Dept. of Computer Science, University of Virginia [††] IBM T.J. Watson Research Center
[‡] Division of Engineering and Applied Sciences, Harvard University

Abstract

This paper studies the impact on energy efficiency and thermal behavior of design style and clock-gating style in queue and array structures. These structures are major sources of power dissipation, and both design styles and various clock gating schemes can be found in modern, high-performance processors. Although some work in the circuits domain has explored these issues from a power perspective, thermal treatments are less common, and we are not aware of any work in the architecture domain.

We study both SRAM and latch and multiplexer ("latch-mux") designs and their associated clock-gating options. Using circuit-level simulations of both design styles, we derive power-dissipation ratios which are then used in cycle-level power/performance/thermal simulations. We find that even though the "unconstrained" power of SRAM designs is always better than latch-mux designs, latch-mux designs dissipate less power in practice when a structure's average occupancy is low but access rate is high, especially when "stall gating" is used to minimize switching power. We also find that latch-mux designs with stall gating are especially promising from a thermal perspective, because they exhibit lower power density than SRAM designs. Overall, when combined with implementation and verification challenges for SRAMs, latch-mux designs with stall gating appear especially promising for designs with thermal constraints. This paper also shows the importance of considering the interaction between architectural and circuit-design choices when performing early-stage design exploration.

Categories and Subject Descriptors

B.6 [**Hardware**]: Logic Design

General Terms

Design

Keywords

Clock gating, power, temperature, architecture

1. INTRODUCTION

As modern CPU designs face power and thermal bottlenecks, designers typically adopt *clock gating*—gating off the clock signal to unneeded units, thus reducing dynamic power dissipation. Although there has been quite a lot of circuit-level work on clock gating [5, 11], we see very little work from the perspective of architecture. Brooks et al. [3] describe how to model clock gating in an architecture level power simulator, and Li et al. [8] propose a deterministic clock gating scheme. Neither work compares the efficiency of different clock gating schemes, nor explores the thermal effects of clock gating. In this paper, by comparing the power and thermal efficiency of three different clock gating schemes, we show it is important to take architectural factors into consideration when the clock gating decision is made.

We focus on clock gating techniques applicable to queue/array structures in CPUs. Queue/array structures, like register files, TLBs, and every kind of decoupling queue in the processor, consume a large portion of the chip area and power budget. The chip's hotspot is typically in one of these structures. Power and thermal effects of different clock gating schemes for queue/array structures are therefore an important area of investigation.

We investigate two design styles and three clock gating schemes for queue/array structures. Two clock gating schemes apply to latch-mux design: valid-bit clock gating, in which only valid entries are clocked; and "stall" gating, in which even valid entries are not clocked when not in use. The third clock gating style applies to SRAM designs, and simply gates ports not in use. The effectiveness of valid-bit gating is determined by the queue occupancy, SRAM port gating by access rate, and latch-mux stall gating by both. The ratio of queue occupancy versus array access rate depends on architectural factors and varies from unit to unit on the chip and benchmark to benchmark.

While there are many considerations as to what design style each queue/array structure should adopt, in this paper we focus on their architectural characteristics. More specifically, we investigate two architectural aspects of each structure: occupancy and access rate. If a structure has high occupancy but relatively low access rate, an SRAM-based design will be power-efficient because most of the time the structure can be clock-gated (due to its low access rate). On the other hand, if a structure usually has very few valid entries, which are accessed very frequently, then a latch-mux design makes more sense since most of the entries can be gated-off most of the time.

This paper presents results of circuit simulations for several implementations of array structures and architectural analysis of the utilization of these structures. Despite the power and area benefits of SRAM-based array structures,

ISLPED'05, August 8–10, 2005, San Diego, California, USA
Copyright 2005 ACM 1-59593-137-6/05/0008 ...$5.00.

there are several reasons why designers may favor latch-mux designs for relatively small array structures such as queues and buffers within a microprocessor. SRAM designs typically require a full-custom design methodology and can require additional design attention due to increased SER-susceptibility and complications with SOI process technologies. For example, array design effects with SOI technology include parasitic bipolar currents during writes and bitline leakage during read operation [1]. Latch-based design structures may also be favored as they fit in more easily with standard scan-chain based testing strategies.

2. MODELING METHODOLOGY

2.1 Performance Model

We use Turandot/PowerTimer [2, 6, 10] to model a single-threaded out-of-order, superscalar processor with resource configuration similar to current generation microprocessors. Table 1 describes the configuration of our baseline processor for the single-threaded design point.

Processor Core	
Dispatch Rate	5 instructions per cycle
Retirement Rate	9 instructions per cycle
Reservation stations	mem/fix queue (2x20), fpq (2x5)
Functional Units	2 FXU, 2 FPU, 2 LSU, 1 BRU
Physical registers	80 GPR, 72 FPR
Branch predictor	16K-entry bimodal, 16K-entry gshare, 16K-entry selector, all with 1-bit entries
Memory Hierarchy	
L1 Dcache Size	32KB, 2-way, 128B blocks
L1 Icache Size	64KB, 2-way, 128B blocks
L2 I/D	1MB, 4-way LRU, 128B blocks
L1/L2/L3 latency	1/9/77 cycles

Table 1: Configuration of simulated processor.

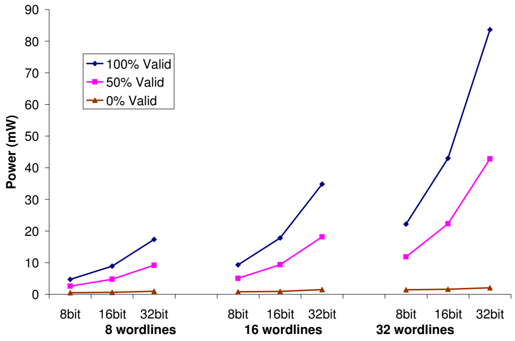


Figure 1: Latch-Mux Power.

The baseline single-threaded performance model has been extensively validated against a pre-RTL, latch-accurate processor model for a current generation microprocessor [9].

2.2 Benchmarks

To keep our data collection and presentation tractable, we use eight SPEC2000 integer benchmarks and seven SPEC2000 floating point benchmarks. They were selected to provide a mix of programs with a range of compute-intensive vs. memory-bound behaviors. They are compiled by the *xlc* compiler with the -O3 option.

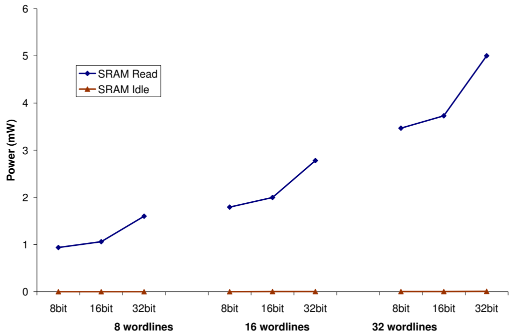


Figure 2: SRAM Power.

2.3 Power Model

PowerTimer differs from existing academic microarchitectural power-performance simulators primarily in energy-model formation. The base energy-models are derived from circuit-level power analysis performed on structures in a current, high-performance PowerPC processor. This analysis has been performed at the macro level, and in general, multiple macros will combine to form microarchitectural level structures corresponding to units within our performance model. PowerTimer models over 60 microarchitectural structures which are defined by over 400 macro-level power equations.

These energy models are tightly coupled with Turandot. PowerTimer defines the “unconstrained” power estimates—baseline power assuming no clock gating of each structure, and these are scaled by microarchitectural utilization and clock gating style to estimate clock-gated power dissipation in each clock cycle.

For this paper, we studied the fixed point register file (FX_REG), fixed point issue queue (FXQ), fixed point register mapping unit (FX_MAPPER), floating point register file (FP_REG), floating point issue queue (FPQ), floating point register mapping unit (FP_MAPPER), load reorder queue (LSU_LRQ), store queue (LSU_SDQ) and store reorder queue (LSU_SRQ). The number of ports modeled for these structures appears in Table 2. For these structures, we developed detailed models to compare the unconstrained power for SRAM and latch-mux implementations.

For the specific structures we studied, the SRAM designs were adapted from low-power memory designs. The design utilizes minimum sized transistors and does not include sense amps because we are primarily looking at relatively small queues and buffers. The latch-mux designs were developed specifically for this paper to be as comparable as possible to the SRAM designs. The decoders and input latches were actually reused from the SRAM designs, and the latch-mux designs followed similar sizing and fanout methodology. Simulations of the latch-mux and SRAM register files were completed using Nanosim with accuracy equivalent to HSPICE. Each register file size was designed at the schematic level, for a total of eighteen designs. Designs were simulated using 130nm process technology models, at 1.2V, and 1GHz. Additionally, for the latch-mux design, the valid bits were generated externally to facilitate rapid testing. During simulation each netlist was paired with three different vector files, corresponding to the three different measurements: read, write, and idle powers. The simulation vector

	FXQ	FPQ	FX MAPPER	FP MAPPER	FX REG	FP REG	LRQ	SRQ	SDQ
Read/Write ports number	2/2	2/2	5/2	2/1	5/2	2/1	2/2	2/2	1/1

Table 2: Number of ports modeled. For clustered structures, we only report ports for a single instance.

files allowed Nanosim to verify the functionality of a register file while collecting power consumption data. To ensure measurement consistency, the same vector files were used to simulate SRAM and latch-mux designs of equal dimensions, based on word size and number of wordlines. Furthermore, some care was taken to ensure that different sized register files had similar input vectors.

For each design style, we simulated 9 configurations: 8, 16, and 32 bits wide for each of 8, 16, and 32 wordlines/entries. For the latch-mux designs, we repeated these simulations for scenarios with all, half, and zero entries valid. Figures 1 and 2 compares latch-mux and SRAM read and idle power dissipation. From this data we can determine the relative power dissipation for the two design styles, e.g. the ratio of read and idle power was 0.06 and 0.008 for the 32-bit, 32-entry structure, assuming all the entries are valid.

We interpolated/extrapolated to find the correct power for each structure of interest. For multi-ported structures, we then scaled these values proportionally–see Table 2. For the 80-entry register files, we assume these consist of two 40-entry banks.

2.4 Clock Gating Methodology

PowerTimer uses microarchitectural activity information from the Turandot model to scale down the unconstrained power under a variety of clock gating assumptions. In this study, we apply clock gating on a per-macro basis to explore how power depends on microarchitectural activity. We determine which macros can be clock gated in a fine-grained manner (per-entry or per-stage clock gating) and which can be clock gated in a coarse-grained manner (the entire unit must be idle to be clock gated). For some macros (in particular control logic), we do not apply any clock gating; this corresponds to about 20-25% of the unconstrained power dissipation. Typically, the overall savings due to clock gating relative to the unconstrained power are roughly 40-50%.

There are several styles of clock gating that we can apply. These include valid and stall gating for latch-based structures and read and write port gating for array structures.

Figure 3(a) conceptually diagrams valid-bit based clock gating. This type of clock gating is commonly used in pipeline latches and relatively small memory structures that are designed using latch-mux schemes (e.g. issue queues, instruction buffers, etc). In this style of gating, a valid bit is associated with every bank of latches and the local clock buffer of the latch bank is gated when the valid-bit is not set. Figure 3(b) diagrams stall gating, a more aggressive version of valid-bit gating, that can also clock gate a bank of latches if it is encountering a stall condition. In this case, if a bank of latches contains valid data, but the pipeline is stalled (or when a queue entry is not being accessed), the clock feeding the latch can still be gated, holding the data. While the second style of clock gating does save additional power, it requires additional timing and verification efforts; for example, the gate signal must be glitch-free. These efforts must be justified by the potential power savings quantified by architectural simulations.

Figure 3(c) conceptually diagrams the clock gating methodology that we apply to SRAM-based array structures. In this case, the array structure utilization is proportional to the number of read and write accesses to the structure. We call this read-write port gating.

To model clock gating, we assume that the SRAM array and read-write circuitry can be gated, while the D-latch, precharge, and decoder circuitry cannot; and the latch-mux array can be gated but the D-latch and decoder circuitry cannot.

2.5 Temperature Model

To model operating temperature, we use HotSpot 2.0 (http://lava.cs.virginia.edu/HotSpot) [7]. HotSpot models temperature using a circuit of thermal resistances and capacitances that are derived from the layout of microarchitecture units. The thermal package that is modeled consists of the die-to-spreader TIM (thickness 0.05mm), the heat spreader (thickness 1mm), another TIM and the heat sink (combined thickness 6.9mm), and a fan. Removal of heat from the package via airflow takes place by convection and is modeled using a single, equivalent thermal resistance of 0.8K/W. This assumes the fan speed and the ambient temperature inside the computer “box” (40°C) are constant, both of which are true for the time scales over which our benchmarks are simulated.

Due to lateral heat spreading, thermal behavior is sensitive to the layout of the microarchitecture units. We use the floorplan shown in Figure 4, derived by inspection from the die photo of the POWER5 in [4], but with only a single core on chip. This floorplan is based on the assumption that the same size latch-mux structure consumes equal area as SRAM structure; when we account for the area differential, the structures in question are scaled according to the area ratio. (We do not show a second floorplan for the reduced SRAM areas because the changes from our base floorplan are too small to show.)

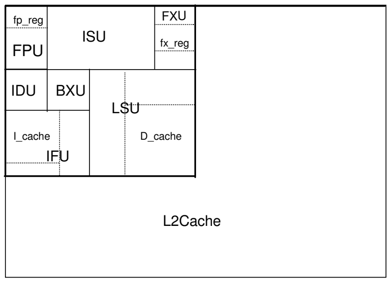
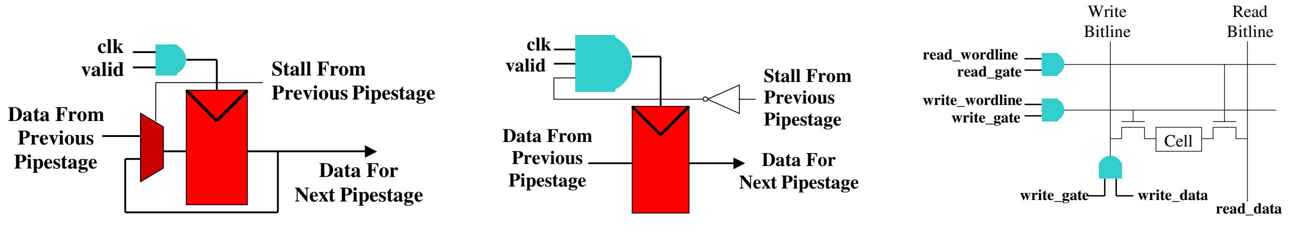


Figure 4: Floorplan used for thermal simulation.

3. RESULTS

We simulate three clock gating styles (valid-bit gating and stall gating for latch-mux designs and read-write port gating for the SRAM design) for the units introduced in section 2.3.

These units can likely be implemented with either design style, but the SRAM implementation is considered more difficult to design and verify.



(a) Valid-bit gating

(b) Stall Gating

(c) Array Gating

Figure 3: Abstract diagrams of clock-gating styles.

First we compare the impact of the different schemes on power, then temperature. We round out the discussion by explaining the architectural behavior that favors one or the other implementation.

3.1 Power

Figure 5 compares the power dissipation of our CPU structures with different clock gating choices. These data are averaged across the integer benchmarks and the floating point benchmarks separately. (Note that even in the integer benchmarks, the floating-point mapper and register file must hold at least 32 active registers, corresponding to the 32 FP registers in the instruction set.) Because the unconstrained power of an SRAM design is much lower than that for the corresponding latch-mux designs, the SRAM design is almost always superior, regardless of clock gating choice.

There are some important exceptions, however. The most striking exception is the fixed-point issue queue, where the latch-mux designs, even with mere valid-bit gating, are superior. The reason for this is that queues with sufficiently low occupancy favor latch-mux designs in which only active entries are clocked. As we can see from Figure 6, unlike other units, the utilization of FXQ with latch-mux design and valid-bit gating is lower than that with SRAM design. (Note that the occupancy is the same across all designs; since we do not consider dynamic thermal management here, the different design choices do not affect execution. What matters is how the power and temperature for different design styles depend on occupancy and activity factors.)

If we compare the fixed point issue queue and the fixed point register file, entries in the register file typically must stay active much longer than in the issue queue. A fixed point instruction is put into the issue queue after renaming and is pulled out of that queue as soon as all its data dependencies are resolved. However, the entry of a physical register file can only be freed after the corresponding instruction commits. Branch mispredictions also play an important role in regularly clearing the queue and keeping average occupancy low, whereas at least 32 registers must remain active even after a misprediction flush. These factors are less true for FP programs, where mispredictions are much less frequent and FP execution latencies increase issue-queue waiting times. Because of its low occupancy, the fixed-point issue queue favors latch-mux design for many benchmarks, despite its large unconstrained power consumption. The FXQ favors latch-mux even more with stall gating. Indeed, stall gating is always vastly superior than valid-bit gating, because stall gating can gate more entries. Even structures with high occupancies will fare well with stall gating if access rates are low.

3.2 Temperature

As figures in the left columns of Figure 7 (integer workloads) and 8 (floating point workloads) show, if we assume that the SRAM and latch-mux designs have equal area, then the temperature follows approximately from its power. The unit temperature with SRAM design is consistently cooler than that with the latch-mux design, regardless of its clock gating styles. Even for the fixed-point issue queue, although the power consumption of this structure with SRAM design is higher than with the latch-mux design, its temperature is lower due to thermal coupling with neighboring units, which all have consistently higher power consumption and higher temperatures with the latch-mux design. Considering the thermal profile of each possible combination is beyond the scope of this work but necessary to fully consider the interaction of design style and thermal coupling.

Of course, the SRAM design is likely smaller than the latch-mux design. This increases its power density. From our circuit design, we estimate that the same frequency SRAM design is roughly 3.3 times smaller than the corresponding latch-mux design. If we include this area effect, we will have the units temperature figures in the right column of Figure 7 and 8. As we can see from these figures, the increased power density of the SRAM design versus the lower power density of the latch-mux design increase the temperature of the units with the SRAM design and decrease the temperature of the units with the latch-mux design. Now for the latch-mux design with stall gating, temperature is consistently lower than for the SRAM design. Even for the latch-mux design with valid bit gating, we find that the FXQ, FX_MAP, and FX_REG have lower temperatures than the SRAM design. We can reduce the temperature of the SRAM design by enlarging its area, however, this will lead to extra latency. It is our future work to quantify this temperature/performance tradeoff with area scaling for the SRAM design.

3.3 Per-Benchmark Differences

The relative power and thermal efficiency of different clock gating styles not only changes from unit to unit, but also changes from benchmark to benchmark.

Figure 9 illustrates this trend for the fixed-point issue queue. As we can see from this figure, we can classify the four benchmarks into four categories: mcf has high occupancy, low access rate; crafty has low occupancy, high access rate; gcc has high occupancy, high access rate; and art has low occupancy, low access rate. Corresponding to these different occupancy-access rate ratios, for the latch-mux design with valid bit gating, mcf and gcc have relatively high temperatures while crafty and art have relatively low tem-

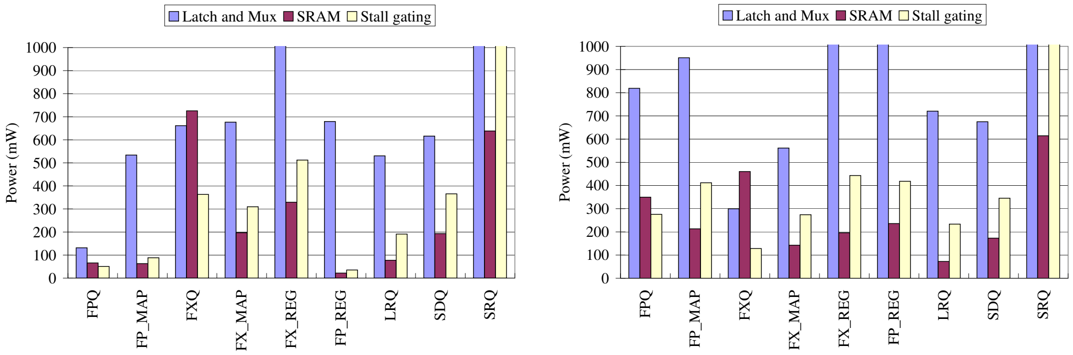


Figure 5: The average unit power of integer benchmarks (left) and floating point benchmarks (right)

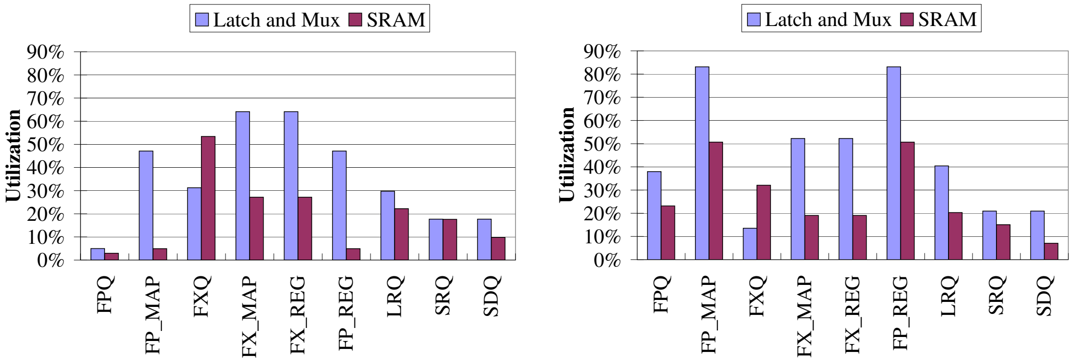


Figure 6: The average unit utilization of integer benchmarks (left) and floating point benchmarks (right)

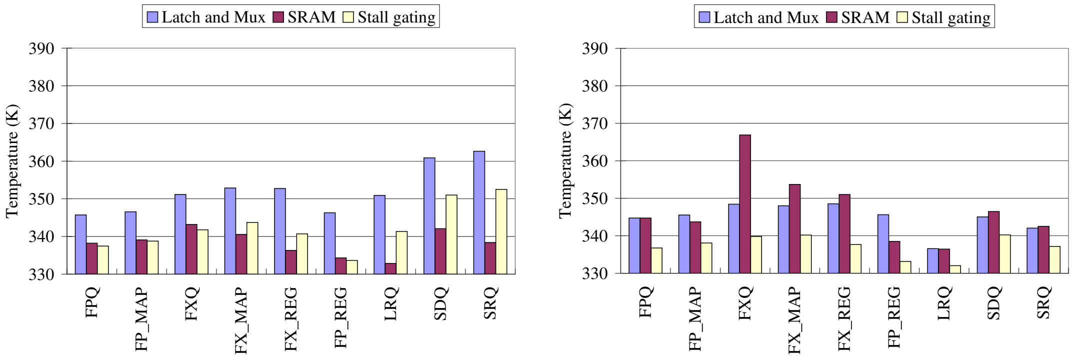


Figure 7: The temperature of the units for integer benchmarks with the ratio of the area of the Latch-Mux design versus the SRAM design at 1 (left) and at 3.3 (right)

peratures; while for the SRAM design, crafty and cc1 have relatively high temperatures and mcf and art have relatively low temperatures.

4. FUTURE WORK AND CONCLUSIONS

This paper investigates energy and thermal effects of different design styles and their associated clock gating choices for queue and array structures in a high-performance, superscalar, out-of-order CPU. We designed and simulated SRAM and latch-mux structures to determine their power dissipation as well as their scaling properties. We then used this data in architectural cycle-accurate performance/power/thermal simulations.

Our SRAM and latch-mux designs only represent one possible set of designs, when in fact this design space is huge. While the specific implementations, areas, and resultant hotspots may vary with different designs, this paper illustrates intrinsic differences between SRAM and latch-mux designs. Specifically, we find that even though SRAM designs have a huge advantage according to their unconstrained power, results can be different when architecture-level effects are modeled. Even latch-mux designs with valid-bit gating, the worst of our three designs, outperforms SRAM for a queue with low occupancy but high access rate, namely the integer issue queue. Furthermore, even though SRAM designs do yield the lowest power dissipation for most structures, their smaller area leads to higher power density. Assuming a 3X area ratio, this causes latch-mux designs with stall gating to consistently give better thermal performance for most structures and most benchmarks.

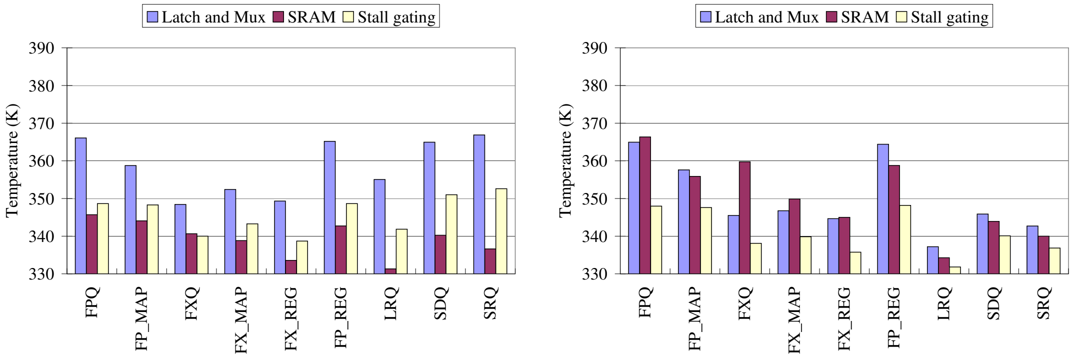


Figure 8: The temperature of the units for floating point benchmarks with the ratio of the area of the Latch-Mux design versus the SRAM design at 1 (left) and at 3.3 (right)

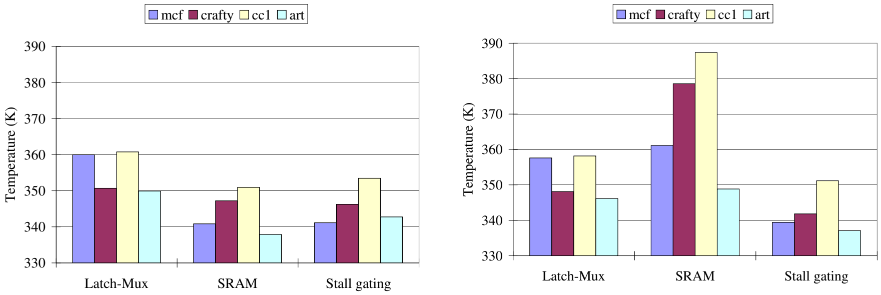


Figure 9: The temperature of FXQ for four benchmarks with the ratio of the area of the Latch-Mux design versus the SRAM design at 1 (left) and at 3.3 (right)

These results show that circuit-level simulations are insufficient for making design-style and clock-gating choices. The behavior of these structures also depends on architecture-level and thermal behavior. Especially in an era of thermally limited design, latch-mux designs with stall gating are an attractive choice, despite their apparent disadvantage when viewed purely from the perspective of raw switching power. SRAMs also have other implementation and testing drawbacks.

Finally, this paper shows the importance of considering design style and clock gating for thermal simulation, as they substantially change operating temperatures and the distribution of hot spots.

Our current results apply to relatively small queue/buffer structures. Scaling to larger structures, exploring designs of different densities (to trade off performance for reduced power density), and a more detailed exploration of how thermal coupling affects these design decisions are all interesting areas for future work.

Acknowledgments

This work was funded in part by the National Science Foundation under grant nos. CCR-0103364, CCF-0429765, CCF-0448313, CNS-0330244, a research grant from Intel MRL, a Faculty Partnership Award from IBM T.J.Watson, and an Excellence Award from the University of Virginia Fund for Excellence in Science and Technology.

5. REFERENCES

[1] A. G. Aipperspach, D. H. Allen, D. T. Cox, N. V. Phan, and S. N. Storino. 1.8-V, SOI, 550-MHz, 64-b PowerPC microprocessor with copper interconnects. *IEEE Journal of Solid-State Circuits*, 34, 1999.

[2] D. Brooks, P. Bose, V. Srinivasan, M. Gschwind, P. G. Emma, and M. G. Rosenfield. New methodology for early-stage, microarchitecture-level power-performance analysis of microprocessors. *IBM Journal of Research and Development*, 47(5/6), 2003.

[3] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 83–94, June 2000.

[4] J. Clabes et al. Design and implementation of the POWER5 microprocessor. In *ISSCC Digest of Technical Papers*, pages 56–57, Feb. 2004.

[5] M. Gowan, L. Biro, and D. Jackson. Power considerations in the design of the Alpha microprocessor. In *Design Automation Conference*, 1998.

[6] Z. Hu, D. Brooks, V. Zyuban, and P. Bose. Microarchitecture-level power-performance simulators: Modeling, validation, and impact on design, Dec. 2003. Tutorial at the 36th Annual IEEE/ACM International Symposium on Microarchitecture.

[7] W. Huang, M. R. Stan, K. Skadron, S. Ghosh, K. Sankaranarayanan, and S. Velusamy. Compact thermal modeling for temperature-aware design. In *Proceedings of the 41st Design Automation Conference*, June 2004.

[8] H. Li, S. Bhunia, Y. Chen, T. N. Vijaykumar, and K. Roy. Deterministic clock gating to reduce microprocessor power. In *Proc. of the Ninth International Symposium on High-Performance Computer Architecture*, February 2003.

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

[10] M. Moudgill, J.-D. Wellman, and J. H. Moreno. Environment for PowerPC microarchitecture exploration. *IEEE Micro*, 19(3):15–25, 1999.

[11] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez. Reducing power in high-performance microprocessors. In *Design Automation Conference*, 1998.