Analyzing the Soft Error Resilience of the CRIB Microarchitecture

Vignyan Reddy Kothinti Naresh, David J. Palframan, and Mikko H. Lipasti
Department of Electrical and Computer Engineering
University of Wisconsin–Madison
{kothintinare, palframan}@wisc.edu, mikko@engr.wisc.edu

Abstract

As processor designs become increasingly complex, soft errors in the execution core are becoming more common. Modern out-of-order processors require large storage structures such as the physical register file, reorder buffer, issue queue, and load-store queue, all of which may be vulnerable to bit upsets. This vulnerability is exacerbated in higher-performing designs that require these structures to be larger.

In this work, we analyze the soft error resilience of the CRIB architecture, which does not require many of these vulnerable storage structures [2]. Despite the lack of traditional superscalar units, a CRIB processor also executes out of order and can achieve performance on par with an area-matched conventional design. To quantify the reliability benefits of the CRIB architecture, we develop a simple analytical model to estimate the soft error rate. Our results show a 70% reduction in the soft error rate from bit upsets for a CRIB processor compared to a conventional baseline. We also perform analysis estimating the relative combinational logic soft error rate in both processors.

1. Introduction

With the scaling of technology, processor reliability is becoming a chief concern. Of particular note is the trend of increasing susceptibility to soft errors resulting from particle strikes. Though the per-device soft error rate (SER) is expected to remain relatively constant, the processor failure rate will increase dramatically as designs incorporate an increasing number of transistors [6]. As we create more complicated designs to take advantage of this larger pool of available resources, it is therefore important to combat the soft error problem by using resilient design techniques or resilient architectures.

In modern processors, storage elements such as SRAM and latches are particularly vulnerable to soft errors in the form of single event upsets (SEUs) [5]. An SEU occurs when the deposited charge from an energetic particle exceeds a storage cell's critical charge, causing the stored value to flip. If undetected, SEUs can lead to silent data corruption, wrong path execution, or other erroneous behavior. Historically, it has been sufficient to protect only cache data with error correcting codes (ECC), since caches generally consume the majority of die space. However, designers must now begin to consider the resilience of the execution core across a range of designs.

Modern high-performance processor cores contain a number of storage structures that are vulnerable to faults. Vulnerable SRAM structures include the register file, issue queue, reorder buffer, and load-store queue. This work analyzes the soft error rate for the CRIB (Consolidated Rename, Issue, and Bypass) architecture, which is a dramatic departure from traditional superscalar design [2]. In particular, CRIB processors are much less vulnerable to bit flips in storage elements, since a significant portion of storage structures are replaced with combinational logic.

The CRIB microarchitecture consolidates the rename, issue and bypass stages of a conventional out-of-order processor into a single structure called a CRIB partition. This modification makes CRIB conceptually similar to a dataflow architecture—instead of reusing a small number of ALUs for all computations, CRIB relies on many ALUs with routing logic to pass data between them according to the dataflow graph. For instance, our analysis models a CRIB processor with 4 ALUs per partition and 7 partitions, giving us a total of 28 ALUs, while our superscalar baseline has only 3 ALUs. Because CRIB relies on execution localized scheduling, it consumes much less power than an area matched conventional OoO processor while achieving similar performance. In addition to significantly lower power consumption, the CRIB architecture greatly simplifies recovery from precise exceptions and misspeculation. In this work we present one additional benefit of CRIB: enhanced resilience to soft errors.

As previously mentioned, a number of features of the CRIB architecture make it more resilient to soft errors than equivalent superscalar designs. Typical structures dominated by vulnerable SRAM such as the rename table, reservation station, physical register file, and re-order buffer are replaced with combinational logic. This large reduction in the number of storage elements is the primary contributor to CRIB's robustness. Additionally, CRIB uses a substantially smaller banked load-store queue, enabled by its low overhead speculative load recovery. To quantify the impact of these changes on the soft error rate, our analysis considers the architectural vulnerability (AVF) of the OoO structures in a CRIB processor and a conventional baseline OoO processor. The concept of AVF acknowledges that due to invalid data and other masking effects, not all bits in a microarchitectural structure can impact correct execution. Thus, we must derate the raw soft error rate of a structure to capture this effect.

In addition to calculating the contribution of storage elements to the overall soft error rate, we also analyze the role of combinational logic soft errors. It is widely held that the overall contribution of logic soft errors is lower than that of bit upsets due to a number of masking effects, including logical masking, timing window masking, and electrical masking [12]. Despite the small contribution of logic soft errors to the overall SER, we include this analysis because CRIB requires significantly more combinational logic. This paper includes the following contributions:

1. Discussion of the inherent resilience of the CRIB architecture to soft errors;
2. Presentation of our simplified methodology to estimate the SER contribution from bit upsets (SEUs) and transient pulses in logic (SETs);
3. Comparison of the SER for a traditional OoO processor and a CRIB processor.

The remainder of this paper is organized as follows. Section 2 describes the operation of the CRIB architecture. Section 3 discusses our methodology for analyzing the soft error rate from bit upsets and transient pulses. Section 4 presents our results comparing the soft error rate of a CRIB processor to that of a traditional baseline. Finally, Section 5 concludes the paper.

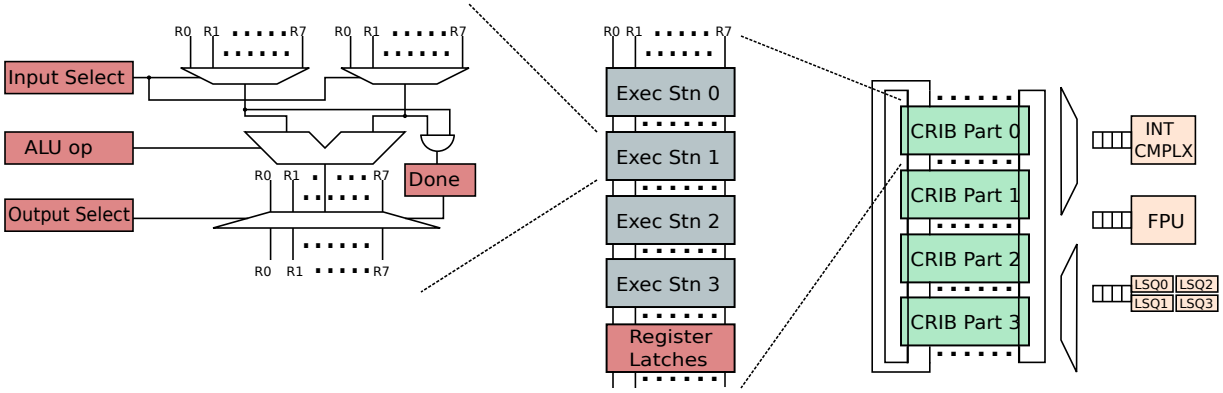

Figure 1: The CRIB architecture. Each CRIB partition consists of four execution stations. Each execution station contains an integer ALU and sequential elements to store the instruction. The done bit is set after a station completes.

2. The CRIB Architecture

Figure 1 shows the components of the CRIB (Consolidated Rename, Issue, and Bypass) architecture [2]. The execution core consists of multiple CRIB partitions connected in a loop. Each CRIB partition consists of four execution stations. Wires carrying register values run the length of each CRIB partition and across all four execution stations. Each execution station has an ALU, two input multiplexers, and an output demultiplexer connected to the register wires. Storage elements are provisioned for each execution station to select the input register wires, the output register, and the ALU operation. A done bit indicates when the execution station is complete. At the output of each CRIB partition is a set of register latches. These register latches are transparent unless the partition is being committed. Thus, at any time only the register latches from the last committed partition will be opaque.

Complex structures like multipliers, floating point units, banked load store queues and special accelerators are too expensive to be replicated across all the CRIB partitions. They are implemented as standard pipeline designs and are shared across all partitions. Request queues allow data transfer between these shared units and the execution stations requesting the shared service.

When a program is executing, instructions are filled into the execution stations in program order. When an instruction is placed into an execution station, it immediately sets the select signals for its input multiplexers to receive the source registers. The done bit, which drives the output valid signal, is also cleared when execution station is initialized. All execution stations receive the current register state, which is passed down from the prior execution station. Because each register is associated with a valid bit, the execution station waits until both input operands are ready. When both inputs are ready, the operation completes in one cycle and the station's done bit is set. Once the instruction has executed, the output select logic drives the result to the appropriate register column. The valid bit of the output register is now set can wake up younger instructions in downstream execution stations. Since independent instructions may have their sources ready at the same time, they start and finish at the same time. This results in out-of-order execution of the program, just as in a traditional OoO processor.

Because CRIB processors rely on in-place execution, a number of structures present in traditional superscalar designs are not needed. For instance, traditional processors achieve instruction-level parallelism through register renaming, which requires a large physical register file. CRIB, on the other hand, relies on what is essentially an architected register file stored in the latches between each partition. The register wires connecting each execution station entry carry positionally correct values, such that each execution station can have a different view of the register state and renaming is not needed. Similarly, typical architectures contain an issue queue, which stores instructions and tracks which are ready to execute based on operand availability. In CRIB, a valid bit propagates alongside each register indicating if the incarnation at the input of the execution station is valid or not. These valid bits eliminate the need for an issue queue in the CRIB architecture, since instruction words are stored in the CRIB entry to which they are assigned.

Traditional superscalar designs also include a reorder buffer so that instructions are committed in order and precise exceptions can be implemented. In CRIB machines, precise interrupts can be easily implemented by broadcasting a re-execute or a invalidate signal to downstream execution stations with younger instructions. Because register values at the output of the last partition to commit are always valid, precise state is maintained. Only when all instructions in a partition complete non-speculatively are output registers latched and the commit pointer advanced. These latched registers then hold the current architectural register state. Since instructions are always committed in program order and precise exceptions handled locally, a reorder buffer is not required in a CRIB processor.

In a standard superscalar processor, load-store queue entries are allocated at the same time as issue queue entries. This requires a load-store queue that is large enough to accommodate instructions that are not currently executing. CRIB processors, in contrast, allocate LSQ entries at execution time, allowing for a smaller load-store queue.

CRIB's lack of a standard issue queue and reorder buffer along with an architected register file in place of a larger physical register file and a smaller load-store queue means that CRIB processors store far fewer state bits than their traditional superscalar counterparts. Having fewer state bits that are susceptible to upsets makes the CRIB architecture more resilient to soft errors. Based on this observation, we perform a more detailed analysis to estimate CRIB's robustness.

3. Soft Error Rate Analysis

This section introduces our methodology for estimating the SER for the CRIB and standard OoO processor architectures. We consider two different components of the overall SER: the SER resulting from upsets in storage elements and the contribution from transients in combinational logic. Although single event upsets are generally considered the largest contributor to the overall soft error rate, we

Unit	Bits per entry	# entries	Total bits
Issue Queue	95	36	3420
PRF	64	96	6144
ROB	100	128	12800
Load Queue	68	48	3264
Store Queue	132	32	4224

Table 1: Storage structures in the baseline superscalar processor.

Unit	Bits per entry	# entries	Total bits
ARF	64	20	1280
Execution Station	147	28	4116
Load-Store Queue	132	12	1584

Table 2: Storage structures in a CRIB processor.

consider logic soft errors as well since the CRIB execution core relies more heavily on combinational logic than traditional architectures.

3.1. Single Event Upsets

Single-event upsets (SEUs) occur when a particle strike flips a bit held in a storage element. It is important to note that when such a flip occurs, this does not necessarily lead to incorrect program execution. For instance, in a physical register file, physical registers that are on the free list contain no useful data and cannot affect program execution. Therefore, when examining a complex microarchitecture, only the bits that are required for architecturally correct execution (ACE) should be considered vulnerable to SEUs.

Prior work by Mukherjee et al. proposes a methodology for identifying these so-called ACE bits [7]. By default, they consider all bits to be ACE bits unless proven otherwise. The work enumerates a number of different categories of un-ACE bits that can be removed from the total vulnerable bit pool. Most obviously, idle or invalid entries in a structure do not contain ACE bits. This includes unallocated physical registers and vacant entries in the issue queue, load-store queue, and reorder buffer. Similarly, speculative state in these units from instructions executed along an incorrectly predicted path cannot impact correct execution. Other state that cannot affect correctness includes prefetch instructions, dead instructions, and bits that will otherwise be logically masked.

The primary difference between a CRIB processor and a more traditional processor is in the execution core, since CRIB accomplishes out-of-order execution differently. The front end components including fetch and decode are similar between the two architectures, and thus we consider only the OoO execution structures in our comparison. For the out-of-order processor, we consider the issue queue, register file, load-store queue, and reorder buffer. For each of these units, we compute the average number of vulnerable entries while running different SPEC2006 benchmarks. To simplify our analysis, we consider only the effects of invalid data and speculative state, and assume that all other state can impact correct execution. We can then compute the total number of vulnerable bits in the processor on average, which we assume to be proportional to the SER. Table 1 shows the relevant storage elements in our baseline processor and the bit breakdown for each.

For the CRIB processor, we perform a similar analysis considering the instructions stored in each CRIB execution station and the load-store queue. We also consider the latched architected state to be always vulnerable in the CRIB processor. The architected state contains 16 ISA visible registers, a flags register and three temporary registers, making a total of 20 registers in a x86-64 design. Note that only one set of register latches will be opaque and hold valid data at any given time. Thus, in our analysis, the SER contribution from

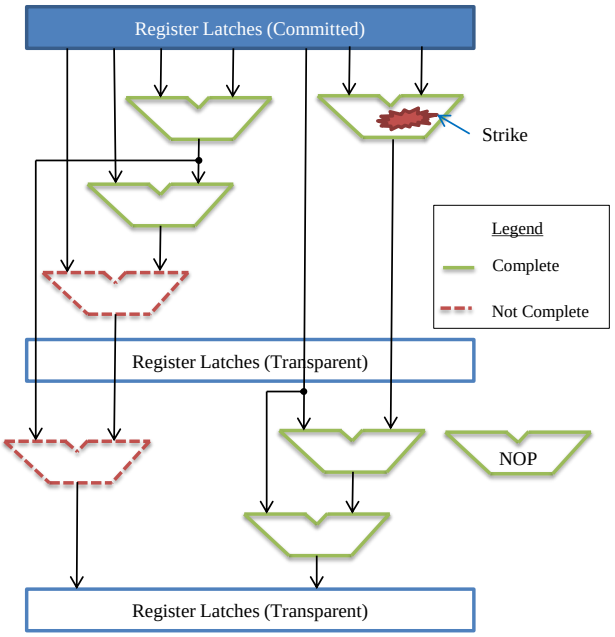

Figure 2: A transient pulse originating in one CRIB partition can be latched at the output of the next partition if the commit pointer moves faster than the height of the dataflow graph.

CRIB's architected register file remains constant across benchmarks. Table 2 shows the relevant storage elements in the CRIB processor.

In this analysis, we are considering bits stored in both latches and SRAM. Traditionally, latches are considered more resilient to soft errors than SRAM. Recent work, however, suggests that the soft error rate for latches may soon be on par with that of SRAM [1]. We therefore, assume the soft error rate per bit to be equal for latches and SRAM during our analysis. Additionally, we recognize that although the performance of the CRIB processor is similar to that of the baseline processor, the IPC can differ. To guarantee a fair comparison, we normalize the SER of each processor to the benchmark IPC.

3.2. Single Event Transients

Single-event transients (SETs) occur when a particle strikes combinational logic and creates a transient pulse that can propagate to the logic output, where it may be latched. The CRIB architecture exploits the area savings from having fewer traditional OoO structures by incorporating many more ALUs, thus increasing the combinational area of the design. We therefore analyze the vulnerability of CRIB and our baseline to SETs. We perform a high-level analysis to compare the combinational SER of the two architectures. As with SEUs, we focus solely on the execution core, since this is what differs between the two architectures.

The raw soft error rate for a combinational logic block is proportional to the total vulnerable area, typically considered to be the gate drain area [3]. If we assume that logic density is approximately uniform, this means that the raw SER is proportional to the area of the logic block. This raw SER is then further derated by different masking factors. One of the most significant masking factors is timing window masking, in which a transient pulse is masked because it does not overlap with the latching window of the sequential element at the logic output. Prior work has identified the probability of latching a transient pulse as dependant on the pulse duration and clock period, as shown in Equation 1 [12]. Here, d is the pulse duration, w is the latching window size, and c is the clock period. Based on

these factors, we consider the logic soft error rate to be proportional to logic area divided by the clock period. For both the CRIB and baseline architectures, we include an additional derating factor to consider only the average fraction of ALUs in use, corresponding average active area. This relationship is shown in Equation 2, which we use to compute the SER for the baseline OoO processor. In this equation, α is the average fraction of ALUs in use, A is the execution core area (3 ALUs for the superscalar design), and c is the clock period. Note that we do not model logical masking, as this will be roughly equivalent since both architectures are executing the same instructions. Additionally, we ignore electrical masking, since this is not easily modeled at a high level.

$$P(latched) = \frac{d-w}{c} \tag{1}$$

$$SER_{OoO} \propto \frac{\alpha \cdot A}{c} \tag{2}$$

The logic SER analysis for CRIB is a somewhat more complicated than it is for a conventional processor. In the CRIB architecture, the latches between partitions are only made opaque when the preceding partition is being committed and transparent otherwise. Because the number of cycles to commit each CRIB partition varies depending on the available instruction-level parallelism, a subtle issue is introduced. Because latches are used, a transient pulse can originate in one partition and propagate through the transparent latches to the next partition. If the commit pointer corresponding to an opaque rank of latches was moved forward at a constant rate, intuition dictates that the pulse would always be ahead of the commit pointer and could never be latched. Since the commit pointer moves at a nonuniform rate, however, there is a chance of latching the pulse that slips through transparent latches to the next partition. Specifically, this can occur if the commit pointer moves faster than the height of the dataflow graph in a partition.

Consider the scenario illustrated in Figure 2. The figure shows a snapshot in time of the dataflow graph mapped to execution stations in two CRIB partitions. The top partition has only one execution station left to complete, and so will commit after one cycle. Likewise, the bottom partition will commit two cycles after the snapshot shown in the figure, since it is waiting on data from the top partition. Now consider a particle strike that occurs in the indicated execution station at some time before the top partition commits. The transient pulse can propagate to the second partition before the first commits. The pulse can take up to two cycles to propagate through the second partition due to the dataflow graph configuration. Despite this, the second partition will only take one cycle to commit after the first, since most execution stations have already completed. This illustrates that it is possible for the commit pointer to overtake or even latch the pulse that originated in the previous partition. Due to this possibility, the average effective area that feeds each rank of latches and through which transients can propagate is larger than the area of a single partition.

To obtain the effective raw SER for a partition, we consider three possible scenarios after a particle strike occurs:

1. The pulse is latched at the output of the partition.
2. The pulse reaches the output after the latches are opaque. The pulse is masked.
3. The pulse propagates through the transparent latches at the partition output. The pulse may be latched by the next partition.

Assuming that the strike occurs during the time in which the commit pointer (opaque latches) moves from the input of the partition to the output, the probability of scenario 1 above corresponds to Equation 1. Because the strike occurs randomly during this time frame, the probability of scenario 2 is equal to that of scenario 3. Conservatively assuming that the probability of the pulse being latched is small, the probability of it slipping through to the next partition is 50%. We treat a pulse that slips through from a previous partition as a strike in the current partition. Thus, the probability of a transient slipping through two ranks of latches is 25%. To capture this effect, we add a multiplier to the SER of a single partition to include transients from preceding partitions. This multiplier m is computed by summing the probability of a pulse originating from all previous partitions, as shown in Equation 3.

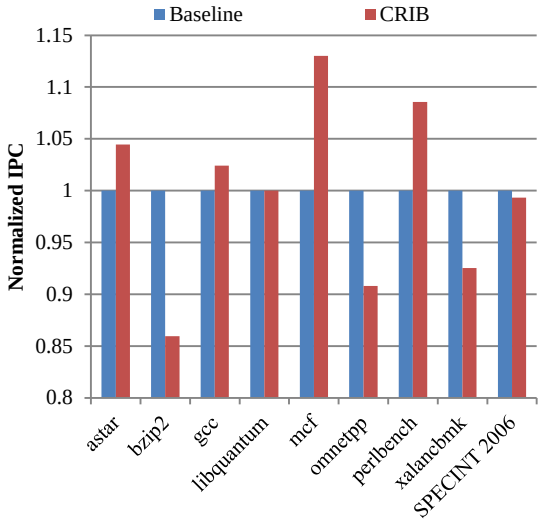

Figure 3: IPC of representative SPEC 2006 integer benchmarks for the simulated baseline and CRIB processors.

$$m = \lim_{N\to\infty} \sum_{n=0}^{N} 0.5^n = 2 \tag{3}$$

$$SER_{CRIB} \propto \frac{2\alpha \cdot A_{partition}}{c_{eff}} \tag{4}$$

As previously mentioned, each CRIB partition can take a different number of cycles to commit. Since this is a dynamic attribute, we compute the effective cycle time c_{eff} as the average time it takes to move the commit pointer and measure this for different benchmarks. Augmenting Equation 2 with the multiplier m and c_{eff} give us Equation 4, which we use to compute the combinational SER for the CRIB architecture. Note that here, the α factor measures the average number of active ALUs per CRIB partition. Just as with SEU analysis, we compensate for the slightly differing performance between architectures by normalizing the computed SER to the benchmark IPC.

4. Results

Table 3 shows the configuration details of the baseline conventional OoO processor and the CRIB processor. The baseline processor is configured to be similar to the Intel Nehalem architecture. The CRIB processor is configured as suggested in [2] such that it is area matched with the baseline processor.

For architectural modeling, we use an x86 execution-driven simulator derived from Bochs [4]. To simulate the CRIB processor, we

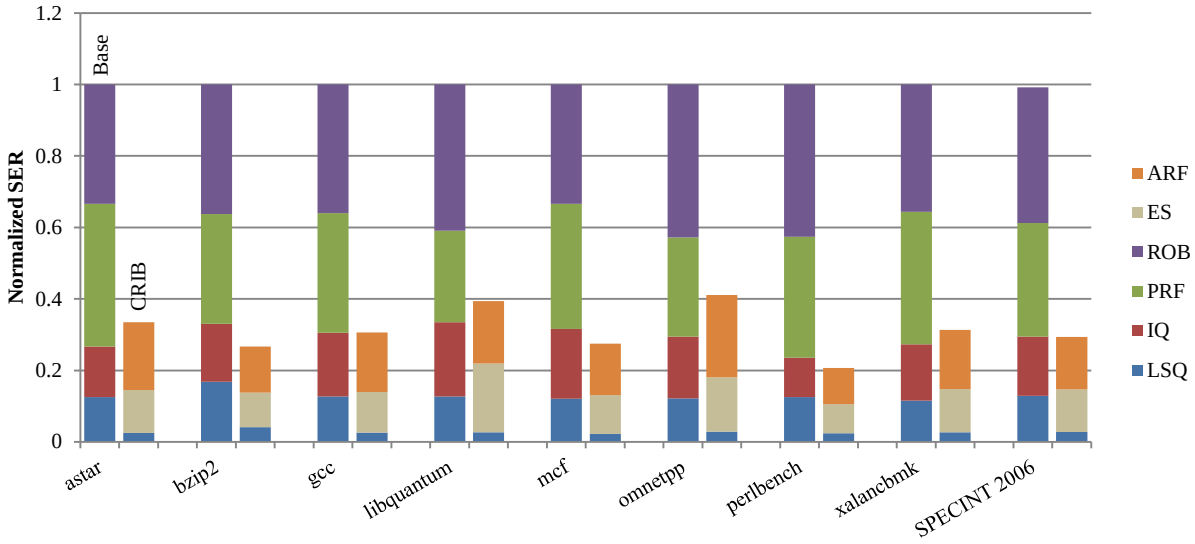

Figure 4: Breakdown of the SER due to single event upsets for the baseline and CRIB processors. Computed SERs are normalized to the IPC of each benchmark.

	Conventional Baseline	**CRIB**
OoO structures	4-wide fetch/commit, 6-wide issue, 128 ROB, 36 IQ, 48 LQ, 32 SQ, 96 Int-PRF, 96 FP-PRF, 11-stage pipeline, speculative scheduling with squashing recovery, aggressive memory reordering with store set predictor (4k ssit, 128 lfst), flush recovery and runahead on L2 miss	7 x 4-entry integer CRIB, 7 x 2-entry FP CRIB, aggressive memory reordering without predictor, light CRIB recovery and runahead on L2 miss, 8 LQ, 4 SQ
Branch Predictor	Combined bimodal (16k entry) / gshare (16k entry) with selector (16k), 32 entry RAS, 4 way 2k-entry BTB	
Integer ALUS	3 (1 cycle)	1 per integer CRIB entry - 28 total (1 cycle)
FP ALUs	2 FP add (4 cycles) and 2 FP multipliers (3 cycles)	2 FP add (4 cycles) and 2 FP multipliers (4 cycles)
Common units	1 integer multiplier (2 cycles), 1 divider (4 cycles), 1 Load (1+2 cycles), 1 Store address (1 cycle), 1 Store data (1 cycle), 1 FP divider/square-root (12-cycles)	
Memory System (Latency)	L1 I-cache: 64KB, 2-way, 64B line size (2 cycles); L1 D-cache: 32KB, 4-way, 64B line size (2 cycles); L2 Unified: 2MB, 8-way, 128B line size (12 cycle); Off-chip memory: (168 cycles); 32-entry prefetch buffer, stream prefectching on L1D miss	

Table 3: Processor configurations used in our evaluation.

also use a modified version of the same simulator. For our vulnerability and performance evaluation, we use a representative set of SPECINT2006 benchmarks [10]. The Pinpoint tool is used to get *simpoints* for each of these benchmarks [9, 11]. Each benchmark is fast-forwarded to the first *simpoint* during which the branch predictor, L1 I-Cache and L2 caches are warmed up. Timing analysis is performed on the 100 million instructions following the first *simpoint*. The simulator records the various activity counts during this window that are used in this analysis.

Figure 3 compares the benchmark IPC of the CRIB processor with that of the baseline processor. As the figure shows, the IPC per benchmark is similar with the CRIB processor both outperforming and underperforming the baseline. The worst-case IPC deviation is 15%, and considering a geometric mean for the SPECINT2006 benchmarks, the IPC differs by 1%. We use the individual benchmark IPCs to normalize our computed soft error rates for each architecture.

The soft error rate from single event upsets is computed for each processor and benchmark as detailed in Section 3.1 and normalized to the benchmark IPC. Figure 4 shows the computed SER from bit upsets for the baseline and CRIB processors. As shown, the overall processor SER is obtained by adding up the SERs of the individual OoO structures during program execution. The issue queue (IQ), physical register file (PRF), and reorder buffer (ROB) are exclusive to baseline processor. The execution stations (ES) which store instruction words and architectural register file state (ARF) are exclusive to the CRIB processor. Although both designs include a load-store queue (LSQ), the CRIB implementation has fewer entries.

As shown, the CRIB processor is 70% less vulnerable to SEUs on average than the baseline processor. In the baseline processor, the physical register file and reorder buffer have the largest SER contribution, and the CRIB design benefits significantly by not relying on these units. As expected, the LSQ contributes less to the SER in CRIB because of its reduced size. The instructions stored in execution stations are analogous to the instruction queue in the baseline and contribute similarly to the SER.

The combinational logic SER for the baseline processor and CRIB processor is presented in Figure 5. These results were obtained as described in Section 3.2, based on the areas published in [2]. As the figure shows, our analysis computes a 1.5x to 2.5x higher error rate from SETs for the CRIB design. While there has been much speculation about if and when combinational logic soft errors will become a significant threat, recent work reveals the SER contribution of SETs relative to SEUs remains trivial [5]. We also note that our SET analysis is conservative. For instance, the baseline superscalar processor requires additional logic for its large OoO structures such as CAM logic, that we have omitted from this analysis. Also, we

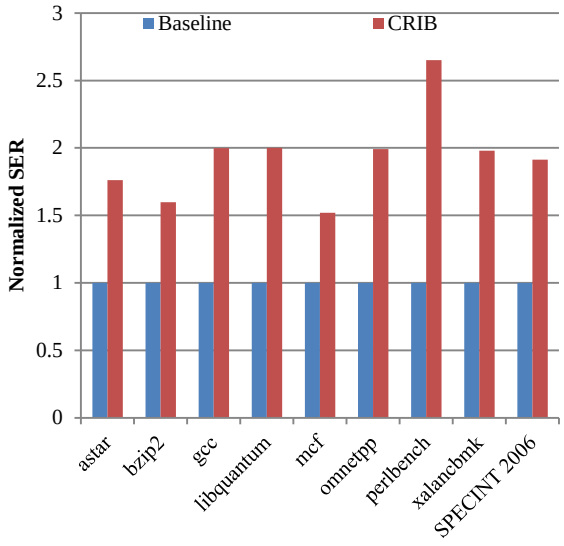

Figure 5: SER due to logic soft errors (SETs) for the baseline and CRIB processors. Computed SERs are normalized to the IPC of each benchmark.

have not modeled electrical masking, which would reduce the SER of the CRIB processor more than the baseline, since the CRIB design employs deeper logic.

5. Conclusion

In this paper, we have evaluated the impact of soft errors on the CRIB architecture. A CRIB processor provides a number of benefits over a traditional architecture, including reduced power consumption. In this paper, we have identified an additional benefit of CRIB processors: resiliency to soft errors. Our analysis shows a CRIB processor to be on average 70% less vulnerable to single-event upsets than an equivalent traditional design. Much of this benefit is derived from CRIB's lack of large structures to store state for out-of-order execution.

Our results also indicate that the CRIB architecture may be more vulnerable to soft errors in combinational logic, though such SETs have a much smaller contribution to the SER than SEUs. Future work will explore techniques to make the CRIB architecture even more robust, particularly considering SETs. For instance, transient errors can be detected at partition boundaries using previously-proposed low-cost detection techniques [8]. When an error is detected, the low overhead re-execution inherent in the CRIB design can be used to recover with a negligible performance penalty.

References

[1] R. Baumann, "Soft errors in advanced computer systems," *IEEE Design & Test of Computers*, vol. 22, no. 3, pp. 258–266, May-June 2005.

[2] E. Gunadi and M. Lipasti, "CRIB: Consolidated rename, issue, and bypass," in *Proc. 38th Annual Int. Symp. on Computer Architecture (ISCA)*, June 2011, pp. 23–32.

[3] K. J. Hass and J. W. Ambles, "Single event transients in deep submicron CMOS," in *Proc. 42nd Midwest Symp. on Circuits and Systems, 1999*, vol. 1, 1999, pp. 122–125.

[4] K. Lawton, B. Denney, N. Guarneri, V. Ruppert, and C. Bothamy, "Bochs: The cross-platform ia-32 emulator," *Sourceforge (January 2006). Available online at URL http://bochs.sourceforge.net.*

[5] N. N. Mahatme, I. Chatterjee, B. L. Bhuva, J. Ahlbin, L. W. Massengill, and R. Shuler, "Analysis of soft error rates in combinational and sequential logic and implications of hardening for advanced technologies," in *Proc. IEEE Int. Reliability Physics Symp. (IRPS)*, May 2010, pp. 1031–1035.

[6] S. Mukherjee, J. Emer, and S. Reinhardt, "The soft error problem: an architectural perspective," in *Proc. Int. Symp. High-Performance Computer Architecture*, 2005, pp. 243–247.

[7] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proc. 36th Annual IEEE/ACM Int. Symp. on Microarchitecture*, December 2003, pp. 29–40.

[8] D. J. Palframan, N. S. Kim, and M. H. Lipasti, "Time redundant parity for low-cost transient error detection," in *Proc. Design, Automation and Test in Europe (DATE)*, March 2011, pp. 1–6.

[9] H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi, "Pinpointing representative portions of large Intel Itanium programs with dynamic instrumentation," in *Microarchitecture, 2004. MICRO-37 2004. 37th International Symposium on*.

[10] A. Phansalkar, A. Joshi, and L. K. John, "Analysis of redundancy and application balance in the SPEC CPU2006 benchmark suite," in *Proceedings of the 34th annual international symposium on Computer architecture*, ser. ISCA '07, 2007.

[11] T. Sherwood, E. Perelman, and B. Calder, "Basic block distribution analysis to find periodic behavior and simulation points in applications," in *Proc. Parallel Architectures and Compilation Techniques*, 2001.

[12] P. Shivakumar *et al.*, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proc. Dependable Systems and Networks*, 2002, pp. 389–398.