Comparing Fault Recovery Mechanisms for Superscalar Processors

Toshinori Sato
PRESTO, Japan Science and Technology Agency
toshinori.sato@computer.org

1. Introduction

This paper investigates the performance cost due to recovering hardware transient faults and compares two fault recovery mechanisms. There are two driving forces to study fault tolerant techniques for microprocessors. One is deep submicron fabrication technologies. Smaller and smaller transistors, higher and higher clock frequency, and lower and lower power supply voltage reduce reliability of microprocessors. The other is increasing popularity of mobile platforms. Microprocessors are used in systems which require high dependability, such as e-commerce businesses. Based on these trends, it is expected that the quality with respect to reliability will become important as well as performance and cost for future microprocessors. To meet the demand, a lot of fault detection mechanisms are recently proposed and evaluated (see Section 2 in [1]). However, fault recovery mechanism is remained to be not investigated with the expectation of rare fault occurrence [2]. This paper focuses on the recovery mechanism. From detailed simulations, we evaluate the cost of transient fault recovery and compare two recovery mechanisms. To the best of our knowledge, this is the first paper evaluating the recovery mechanism based on instruction reissue.

2. Fault detection

In order to detect transient faults, we propose to duplicate committed instructions and compare two results of a single instruction as shown in Figure 1 [1]. It is assumed the comparator is fault-free. This is possible by using large strong cells or triple modular redundancy. We use the instruction reissue to execute each committed instruction twice.

1. First, every instruction is dispatched from the RUU to a functional unit, where it is executed.
2. Second, its execution outcome which is generated for the first time (denoted as F in the figure) is written in the RUU.
3. When the instruction becomes ready for commitment, it is reissued in the RUU and dispatched into a functional unit again.
4. After that, its execution outcome which is generated for the second time (denoted as S in the figure) is delivered to the RUU again.
5. And last, the two results of the single instruction will be compared with each other. If they do not match, a transient fault is detected.

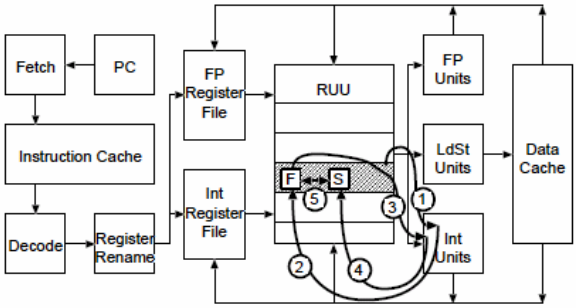

Figure 1: Fault-tolerance mechanism

3. Fault recovery

Even when fault detection is successful, a system provided with no scheme for recovery will usually hang, causing application down. Recovery can be handled by either OS or hardware. We can use two transparent hardware-based recovery schemes. One uses the existing speculation recovery mechanism for mispredicted branches [1, 2], and the other is based on the instruction reissue mechanism for incorrect data speculation [2]. In this paper, we assume the failure is transient, and thus instruction retry is successful. That is, when a fault is detected by the instruction reissue mechanism, re-executing the fault instruction is sufficient.

If we utilize the recovery mechanism for mispredicted branches, the microprocessor flushes its pipeline and then restarts at the corresponding instruction, when a transient fault is detected. All instructions following the faulty instruction are squashed and thus the penalty is very large. On the other hand, the fault recovery mechanism utilizing the instruction reissue exploits its ability of selective squashing. As the mechanism reissues only misspeculated instructions when a data dependence

misprediction occurs, we reissue only instructions dependent upon faulty instructions. That is, each fault is regarded as a misspeculation. Different from the recovery from misspeculation, the recovery from a fault requires the additional two executions of each instruction dependent upon the faulty instruction. This is necessary to detect if a transient fault occurs with the dependent instruction whose operands are corrected. This might explode the number of running instructions, and hence processor might suffer high performance penalty.

4. Simulation results

We implemented a timing simulator using SimpleScalar/Alpha tool set (ver.3.0a) [3]. We evaluate 4- and 8-way out-of-order execution superscalar processors. We use a 1K-entry reuse buffer in order to mitigate performance penalty due to redundant execution. The SPEC2000 benchmark suite is used for this study. The details of the processor model and benchmark programs can be found in [1].

The fault injection module in our simulator randomly corrupts some instructions. Figure 2 shows the percentage increase in execution cycles when faults are injected. The horizontal line indicates the average fault frequency per one million cycles, while the vertical line indicates the corresponding percentage increase. There are four graphs in the figures. Flush- and Selective- mean that their corresponding fault recovery mechanisms are based on the recovery mechanism for control and data misspeculation, respectively. -4 and -8 mean that their corresponding processor models are 4- and 8-way superscalars, respectively. For both 4- and 8-way superscalar models, no significant difference between Flush and Selective models is found until the number of faults reaches about 1000 per million cycles. In addition, the performance loss is increased over the point where approximately 1000 faults occur per million cycles, but the fault frequency is much higher than the actual fault frequency we intend. In other words, it is possible to ignore the cost due to fault recovery when evaluating fault-tolerance microarchitecture, because the overall performance is dominated by fault-free behavior.

5. Conclusions

The current trend of deep submicron technology will reduce reliability of microprocessors, while it is a key technology to increase their performance. Considering the background, a lot of microarchitectures to attack the problem are recently proposed. However, the main interest of researchers is fault detection and is not fault recovery. This paper investigates the cost of fault recovery using two recovery mechanisms; one is the extension of the recovery mechanism for control speculation and the other is that for data speculation. To the best of our knowledge, this was the first evaluation of the latter recovery mechanism on fault recovery. From the detailed simulations, we did not observe considerable difference between two mechanisms for the processor models we evaluated. In addition, through the fault frequency we intended, we found that the overall processor performance is determined by the fault-free behavior.

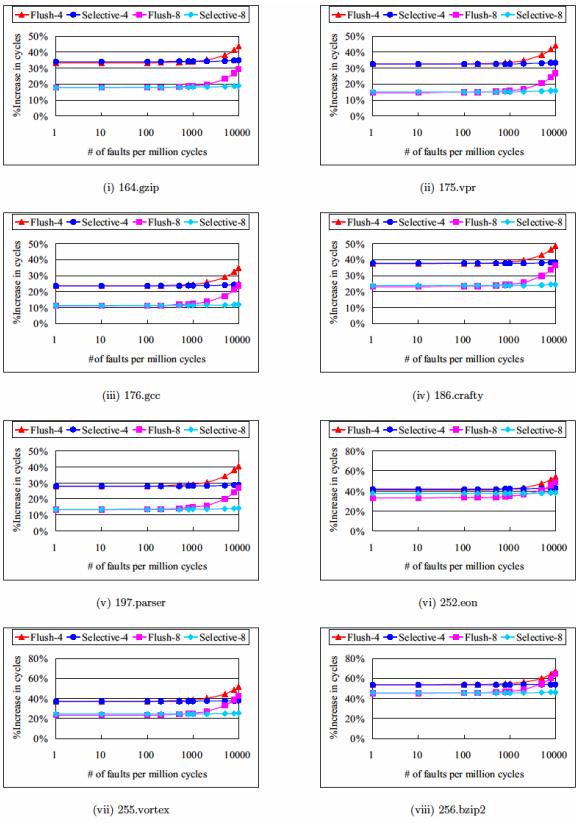

Figure 2: Fault frequency vs. performance penalty

6. References

[1] T. Sato, “Exploiting instruction redundancy for transient fault tolerance,” 18th International Symposium on Defect and Fault Tolerance in VLSI Systems, 2003.

[2] J. Ray, J. C. Hoe, and B. Falsafi, “Dual use of superscalar datapath for transient-fault detection and recovery,” 34th International Symposium on Microarchitecture, 2001.

[3] D. Burger and T. M. Austin, “The SimpleScalar tool set, version 2.0,” ACM SIGARCH Computer Architecture News, vol.25, no.3, 1997.