Dependability-Performance Trade-off on Multiple Clustered Core Processors

Toshimasa Funaki
Kyushu Institute of Technology
t-funaki@klab.ai.kyutech.ac.jp

Toshinori Sato
Kyushu University
toshinori.sato@computer.org

Abstract

We are currently investigating multiple clustered core processor for embedded applications. It is adaptable by considering trade-off between performance, power, and dependability. It also has variation resilience due to its structured design. This paper focuses on investigating trade-off between dependability and performance. Using MITF (mean instructions to failure), we quantify three dependability-performance trade-offs that the multiple clustered core processor can provide. We find an intuitive but noticeable result that thread-level redundancy is much more dependable than instruction-level redundancy, when those provided by the multiple clustered core processor are compared.

1. Introduction

Advanced semiconductor technologies increase soft error rate (SER) [3]. With the reduction in transistor size, the area per bit scales down. In order to prevent breakdown caused by high electric field, the supply voltage also scales down. Hence, the node charge reduces and the bit cell is easy to flip by cosmic ray and alpha particles. Since each bit cell becomes small so that probability that some particles such as neutrons hit the cell will also become small, resulting in the net effect of almost constant SER per bit. Since the number of transistors per chip has been tremendously increased, SER per chip is also exponentially increasing.

In order to detect (and if possible to correct) faults due to single event upsets (SEU), redundant execution of a single program is proposed [9]. The increase in the popularity of multicore processors is favorable to the redundant execution. A single program is duplicated and its two redundant copies are executed simultaneously in the different cores on a multicore processor. When two outcomes for the single program do not match, a SEU is detected.

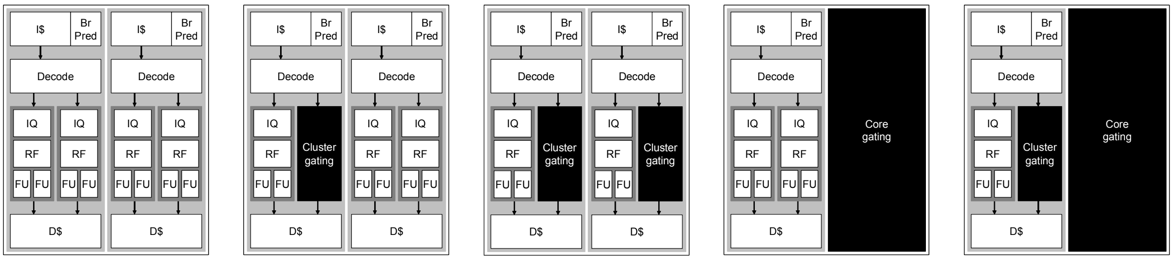
We can exploit redundancy in a single processor [11]. Detecting SEUs is possible by duplicating every instruction in the program rather than the program itself. Two redundant copies of the single instruction are executed simultaneously in the same processor core, and two results for the instruction are compared with each other. If they do not match, a SEU is detected.

There is a trade-off between dependability and performance. For example, up to two times performance gain is expected when we use two processor cores for parallel execution rather than for redundant execution. Hence, it is important to consider the dependability-performance trade-off and this paper investigates how to evaluate the tradeoff.

The rest of this paper is organized as follows. Section 2 introduces the multiple clustered core processor, which is a platform of this study. Section 3 examines the trade-off between dependability and performance. Section 4 presents estimation examples. And Section 5 provides conclusions.

2. Multiple Clustered Core Processor

We proposed the multiple clustered core processor [12] for embedded applications, which require high energy efficiency. As shown in Figure 1, it is a homogeneous multicore processor. The difference from the conventional homogeneous multicore processors is that it consists of multiple clustered cores rather than monolithic ones. Each core is based on the clustered microarchitecture [6]. Figure 1 shows a multiple clustered core processor with two homogeneous clustered cores, each of which has two identical clusters. In the figure, each cluster consists of instruction scheduling queue (IQ), register files (RF), and functional units (FU). Instruction and data caches (I$ and D$), branch predictor (BrPred) and decoder



(a) Dual large core (b) Hetero core (c) Dual small core (d) Single large core (e) Single small core

Figure 2: Reconfigurable Execution Modes

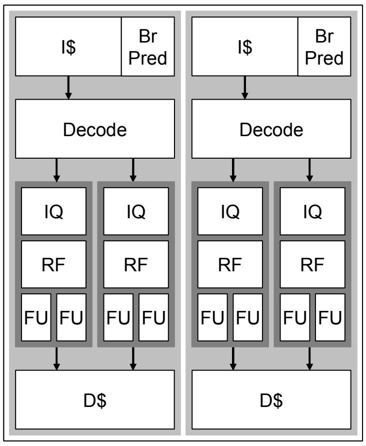


Figure 1: Multiple Clustered Core Processor

(Decode) are shared by the clusters.

2.1. Power-performance trade-off

We exploit the clustered microarchitecture combined with multicore architecture in order to make a trade-off between power and performance. We proposed cluster gating and core gating [12]. Figure 2 explains how the cluster gating and core gating work. Providing several execution mode enables to consume power just enough for the required performance. In our previous study, we found the multiple clustered core processor attains equivalent performance to the conventional homogeneous and heterogeneous multicore processors with less power consumption [12]. The concept of the clustered core is extended into multi-performance processors for low-power embedded applications [8].

2.2. Dependability issue

The multiple clustered core processor has a good characteristic in its dependability. One of the simple implementations for providing dependability is to redundantly execute a single program. Time redundancy or space redundancy can be utilized. The conventional multicore processors are very suitable for exploiting space redundancy for dependability [9]. In order to check errorless, a single thread is redundantly executed on multiple cores. When two outcomes for the single thread (or those for every instruction in the thread) do not match, a fault is detected. Furthermore, the multiple clustered core processor can change its dependability mode according to the importance of the current thread [12], as shown in Figure 3. If the thread is critical, it is duplicated and redundantly executed across multiple cores, as shown in Figure 3(a). If it is less critical, every instruction in the thread is duplicated and redundantly executed across multiple clusters, as shown in Figure 3(b). The trade-off between dependability and performance is the main topic of this paper, and will be deeply investigated later. The trade-off between dependability and power is another interesting topic and is currently under study.

2.3. Resilience in parameter variations

Another good characteristic of the multiple clustered core processor is its resilience in parameter variations [10]. As the complexity of the semiconductor manufacturing process increases, it is likely that process variations will be more difficult to control. The demand for low power leads supply voltage reduction and hence makes voltage variations a serious problem. Higher and higher clock frequency increases temperature variations in a chip. Under these situations [3], the deep submicron semiconductor technologies will make the traditional design impossible, since they can not provide design margins that it requires. Bowman [4] and Hashimoto [5] demonstrate that due to the statistical characteristics of delays (1) the mean delay increases and the standard deviation decreases as the number of independent critical paths increases, and

(2) both the mean delay and the standard deviation decrease as the logic depth increases. When chip area is limited, every core in a multicore processor has smaller critical path delay than a monolithic single-core processor does. In addition, the number of critical paths is larger in the multicore processor than in the single-core processor, since the former one has multiple copies of an identical core. The multiple clustered core processor further decreases its critical path delay and further increase the number of critical path, since it divides each core into smaller clusters. The relationship between the configuration of the multiple clustered core processor and its variation resilience is an interesting topic in the future study.

In the rest of this paper, we focus on the trade-off between dependability and performance.

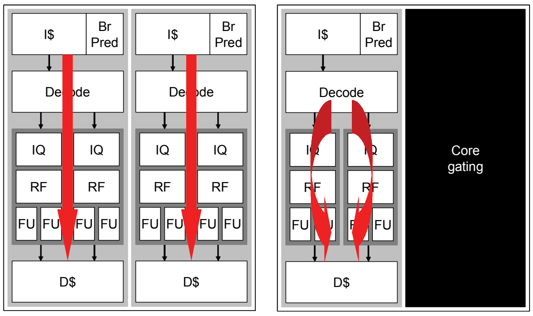
3. Dependability-Performance Trade-off

Next, we consider dependability-performance trade-off. We use mean instructions to failure (MITF) [14] as a metric for evaluating the trade-off. MITF indicates how many instructions a processor executes between two errors. MITF is calculated as follows [14]:

$$MITF = \frac{\#instructions}{\#errors} = \frac{\#instructions}{\left(\dfrac{\#cycles}{f \times MTTF}\right)}$$

$$= IPC \times f \times MTTF = \frac{IPC \times f}{error\_rate \times AVF}$$

where f is clock frequency, MTTF is mean time to failure, IPC is instructions per cycle, *error_rate* is SER, and AVF is architectural vulnerability factor [14], which is the probability that a fault in a processor structure will result in a visible error in a program's final output. Using MITF, we quantify dependability-performance trade-off of a processor core without any redundancy, a multicore chip with thread-level redundancy, and a core with instruction-level redundancy, which are shown in Figure 3.



(a) Thread-level redundancy (b) Instruction-level redundancy

Figure 3: Dependability Modes

MITF of the first one is formulated as follows:

$$MITF_{undependable} = \frac{IPC_L \times f}{N_L \times E_b \times AVF_{core}}$$

where IPC$_L$ is IPC of a large core shown in Figure 2(d), N$_L$ is the total bit of the core, E$_b$ is soft error rate per bit, and AVF$_{core}$ is AVF of the processor core without any redundancy.

MITF of the second model is as follows:

$$MITF_{thread}$$
$$= \frac{IPC_L \times f}{rate\_of\_concurrent\_two\_errors \times AVF_{thread}}$$
$$= \frac{IPC_L \times f}{\left({}_{N_L \times 2}C_2 \times E_b^2\right) \times \left(\dfrac{{}_{N_L}C_1}{{}_{N_L \times 2}C_2} \times AVF_{core}\right)}$$
$$= \frac{IPC_L \times f}{N_L \times E_b^2 \times AVF_{core}}$$

where AVF$_{thread}$ is AVF of the multicore chip with thread-level redundancy. Here, we ignore every overhead regarding redundant execution. In this formulation, we have to consider the case where bit flips occur at the same bits in the different cores. While it is possible that bit flips in different bits cause a same error in both cores, we do not consider the case here. Considering the situation remains to the future study.

And last, MITF of the third is as follows:

$$MITF_{insn} = \frac{IPC_S \times f}{rate\_of\_concurrent\_two\_errors \times AVF_{insn} + N_{shared} \times E_b \times AVF_{shared}}$$
$$= \frac{IPC_S \times f}{N_{cluster} \times E_b^2 \times AVF_{cluster} + N_{shared} \times E_b \times AVF_{shared}}$$

where IPC_S is IPC of a small core shown in Figure 2(e), $N_{cluster}$ and N_{shared} are the total bit of the cluster and that of the structures in the core shared by the clusters, respectively, AVF_{insn} , $AVF_{cluster}$ and AVF_{shared} are AVF of the core with instruction-level redundancy, that without redundancy, and AVF of the shared structures, respectively. Here, we ignore every overhead regarding redundant execution again. In this formulation, we have to consider the case where bit flips occur at the same bits in the different clusters. We do not consider the case where bit flips in different bits cause a same error in both clusters again.

4. Estimations

In order to compare three MITF's, we need E_b, three N's, two IPC's, and three AVF's. We can see E_b is approximately from 0.001 to 0.01 FIT/bit [7] and we use the error rate of 0.01 FIT/bit. We consider two cases where memory structures (caches and TLB) are protected or not. When they are protected by ECC, their error rate is 10^4 times reduced [13].

We assume N is proportional to the area of the structure. We use the area estimates [12] shown in Table 1.

Table 1: Area Estimates (mm^2) [12]

D cache	13.0
I cache	10.4
TLB	4.4
Fetch, BrPred	4.5
Decode	1.7
OOO exec	2x 10.1
RFs	2x 2.9
Func units	2x 6.5
Misc	2.4
Routing	26.4

Next, we estimate IPC's through simulations. We use a multicore processor simulator M5 [2] to estimate IPC's. Processor configuration is that used in [12], which is shown in Table 2. We use 10 programs from SPEC2000. IPC_L and IPC_S are summarized in Table 3.

Table 2: Processor Configuration [12]

IQ entries	2x 16
Int units	2x 2
FP units	2x 2
Ld/St units	2x 1

Table 3: Instructions per cycle (IPC)

	IPC_L	IPC_S
crafty	1.55	1.04
gcc	1.09	0.81
gzip	1.43	1.02
mcf	2.42	1.33
twolf	2.07	1.24
vpr	1.75	1.16
applu	1.79	1.06
equake	2.36	1.36
galgel	2.49	1.36
lucas	2.79	1.67

Based on the assumptions and estimations above, we have Table 4 for processors with non-protected caches. Here, we assume that all AVF's have an identical value and $MITF_{thread}$ and $MITF_{insn}$ are normalized by $MITF_{undependable}$. Surprisingly, $MITF_{insn}$ is smaller than $MITF_{undependable}$ for half of the programs. This is because processor performance (IPC) is seriously degraded due to lack of functional units.

Table 4: Relative MITF's (w/o ECC)

	$MITF_{thread}$	$MITF_{insn}$
crafty	100	1.08
gcc		1.19
gzip		1.14
mcf		0.88
twolf		0.96
vpr		1.06
applu	100	0.95
equake		0.92
galgel		0.88
lucas		0.96

If we consider that memory structures can be protected by ECC, we have Table 5. If we can ignore SEU in memory structures, $MITF_{insn}$ is improved. Actually, three AVF are different and we guess $AVF_{core} > AVF_{cluster} >> AVF_{shared}$. Hence, the differences will be smaller. Nonetheless, we will get $MITF_{thread} >> MITF_{insn} > MITF_{undependable}$. It should be noted that we can get $MITF_{thread}$ at the cost of higher power consumption, since two processor cores are used for redundant execution. This is an intuitive result, however, we quantified the trade-off between dependability and performance.

Table 5: Relative MITF's (with ECC)

	$\text{MITF}_{\text{thread}}$	$\text{MITF}_{\text{insn}}$
crafty	100	1.40
gcc		1.54
gzip		1.48
mcf		1.15
twolf		1.25
vpr		1.38
applu	100	1.24
equake		1.20
galgel		1.14
lucas		1.25

5. Conclusions

In this paper, we quantified dependability-performance trade-off on the multiple clustered core processor. Under several assumptions and estimations, we found that thread-level redundancy has much more dependable than instruction-level redundancy at the cost of higher power consumption. The estimates will be used for the decision which redundancy model is selected for the present thread according to its importance.

The future studies will include AVF estimation via simulations. When we use instruction-level redundancy, we have to consider shared structures, which are vulnerable to errors [1]. We are also interested in dependability-power trade-off on multiple clustered core processor.

Acknowledgement

This work is partially supported by Grant-in-Aid for Scientific Research (KAKENHI) (A) # 19200004 from Japan Society for the Promotion of Science (JSPS), and by the CREST (Core Research for Evolutional Science and Technology) program of Japan Science and Technology Agency (JST).

References

[1] N. Aggarwal, K. K. Saluja, J. E. Smith, P. Ranganathan, N. P. Jouppi, and G. Krejci, "Motivating Commodity Multi-Core Processor Design for System-level Error Protection," 3rd IEEE Workshop on Silicon Errors in Logic - System Effects (2007)

[2] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The M5 Simulator: Modeling Networked Systems," IEEE Micro, Vol. 26, No. 4 (2006)

[3] S. Borker, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation," IEEE Micro, Vol. 25, No. 6 (2005)

[4] K. A. Bowman, S. G. Duvall, and J. M. Meindl, "Impact of Die-to-die and Within-die Parameter Fluctuations on the Maximum Clock Frequency Distribution for Gigascale Integration," IEEE Journal of Solid-State Circuits, Vol. 37, No. 2 (2002)

[5] M. Hashimoto and H. Onodera, "Increase in Delay Uncertainty by Performance Optimization," International Symposium on Circuits and Systems (2001)

[6] R. E. Kessler, "The Alpha 21264 Microprocessor," IEEE Micro, Vol. 19, No. 2 (1999)

[7] E. Normand, "Single Event Upset at Ground Level," IEEE Transactions on Nuclear Science, Vol. 43, No. 6 (1996)

[8] Y. Oyama, T. Ishihara, T. Sato, and H. Yauura, "A Multi-Performance Processor for Low Power Embedded Applications," 10th IEEE Symposium on Low-Power and High-Speed Chips (2007)

[9] E. Rotenberg, "AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors," 29th International Symposium on Fault-Tolerant Computing (1999)

[10] T. Sato, "Ultra Low Power (ULP) Challenge in System Architecture Level -New Architectures for 45-nm, 32-nm Era-," Panel Discussion: Top 10 Design Issues, 12th Asia and South Pacific Design Automation Conference (2007)

[11] T. Sato and I. Arita, "In Search of Efficient Reliable Processor Design," 30th International Conference on Parallel Processing (2001)

[12] T. Sato and A. Chiyonobu, "Multiple Clustered Core Processors," 13th Workshop on Synthesis and System Integration of Mixed Information Technologies, (2006)

[13] M. Sugihara, T. Ishihara, K. Hashimoto, and M. Muroyama, "A Simulation-Based Soft Error Estimation Methodology for Computer Systems," 7th International Symposium on Quality Electronic Design (2006)

[14] C. T. Weaver, J. Emer, S. S. Mukherjee, and S. K. Reinhardt, "Reducing the Soft-Error Rate of a High-Performance Microprocessor," IEEE Micro, Vol. 24, No. 6 (2004)