Power-Performance Trade-off of a Dependable Multicore Processor

Sato, Toshinori
System LSI Research Center, Kyushu University

Funaki, Toshimasa
Graduate School of Computer Science and System Engineering, Kyushu Institute of Technology

出版情報：Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing, pp.268-273, 2007-12
バージョン：
権利関係：

Power-Performance Trade-off of a Dependable Multicore Processor

Toshinori Sato
Kyushu University
toshinori.sato@computer.org

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

Abstract

As deep submicron technologies are advanced, new challenges, such as power consumption and soft errors, are emerging. A naïve technique, which utilizes emerging multicore processors and relies upon thread-level redundancy to detect soft errors, is power hungry. Another technique, which relies upon instruction-level redundancy, diminishes computing performance seriously. This paper investigates trade-off between power and performance of a dependable multicore processor, which is named multiple clustered core processor *(MCCP). It is proposed to hybrid thread- and instruction-level redundancy to achieve both large power efficiency and small performance loss. Detailed simulations show that the MCCP exploiting the hybrid technique improves power efficiency in energy-delay product by 13% when it is compared with the one exploiting the naïve thread-level technique.*

1. Introduction

Advanced semiconductor technologies increase soft error rate (SER) [4]. In order to detect (and to correct if possible) faults due to single event upsets (SEU), redundant execution of a single program is proposed [6, 8]. 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, an SEU is detected [8]. In this paper, this technique is called *dependability with thread-level redundancy* (DTR). Unfortunately, the DTR is power consuming. Two cores consume at least two times larger power than a single core does, when power for comparing two results is considered.

Redundancy can be exploited also in a single processor. Detecting SEUs is possible by duplicating every instruction in the program rather than the program itself. Two redundant copies of the instruction are executed simultaneously in the same processor core, and two results for the single instruction are compared with each other. If they do not match, an SEU is detected [6]. In this paper, this technique is called *dependability with instruction-level redundancy* (DIR). While the DIR is less power-hungry than the DTR, it suffers significant performance loss [6].

In order to solve the both problems of power consumption in the DTR and performance loss in the DIR, it is proposed to hybrid the two techniques. The study on an adaptable multicore processor is ongoing [7]. It is called *multiple clustered core processor* (MCCP). Both power efficiency and performance will be improved by hybridizing the DTR and the DIR on the MCCP.

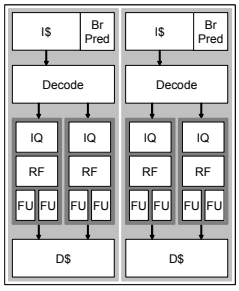

Figure 1. Multiple Clustered Core Processor

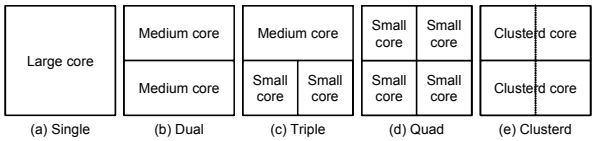

Figure 2. Different Types of Multicore Processors

2. Multiple Clustered Core Processors

MCCP [7] is 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. Figure 1 shows an MCCP with two homogeneous clustered cores, each of which has two identical clusters. In the figure, each cluster consists of instruction scheduling queue (IQ), register file (RF), and functional units (FU). Instruction and data caches (I$ and D$), branch predictor (BrPred) and decoder (Decode) are shared by all clusters in the core.

2.1. Power-Performance Trade-off Issue

Multicore processors are a promising solution that achieves high performance with low power consumption. Figure 2 shows different types of multicore processors. Figure 2a is a uniprocessor. Figures 2b and 2d are homogeneous multicore processors, while Figure 2c is a heterogeneous one. As you can see, the heterogeneous multicore processor consists of several cores with different scales in area and performance. When a thread requires high performance but it does not have large parallelism in it, a large core serves. When the other thread also requires high performance but it has large parallelism in it, it is better in energy efficiency that multiple small cores serve. When high performance is not required by another thread, a small core is utilized. The efficient use of different kinds of cores satisfies requested performance with low power consumption. From the view of energy efficiency, heterogeneous multicore processors consisting of cores with different scales are a

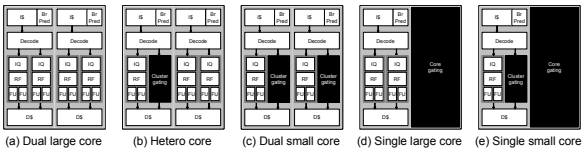

Figure 3. Cluster Gating and Core Gating

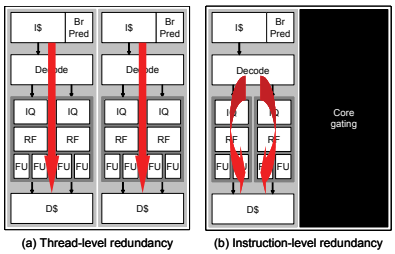

Figure 4. Dependability Modes

good solution [5].

The MCCP exploits the clustered microarchitecture to realize heterogeneity on the homogeneous multicore processor using cluster gating and core gating [7]. Figure 3 explains how they work. This is an MCCP consisting of two dual-cluster cores. Figure 3a shows a homogeneous dual core processor consisting of large cores. When high performance is not required, some clusters or cores are turned off, as shown in Figures 3b-3e. The black box means that the cluster or the core is turned off. Using the gating, only a small number of clusters and cores are active so that requested performance of the allocated thread is satisfied.

2.2. Dependability Issue

The MCCP has a good characteristic in its dependability, as shown in Figure 4. It exploits inter-core redundancy in thread level since it is a multicore processor. A single thread is duplicated and is redundantly executed across multiple cores, as shown in Figure 4a. The MCCP utilizes the DTR. Unfortunately, the DTR is power consuming, since two cores redundantly execute the single thread.

It also exploits intra-core redundancy in instruction level. Every instruction in the thread is duplicated and redundantly executed across multiple clusters in a core, as shown in Figure 4b. The DIR is easily utilized in the MCCP, since each redundant copy of the single instruction is executed in its dedicated cluster. This does not require any complex hardware supports. However, unfortunately, the DIR suffers significant performance loss, since the number of executed instructions is two times increased for the same program.

In order to reduce power consumption of the DTR and performance loss of the DIR, it is proposed to hybrid the two techniques in the MCCP. This hybrid technique is called *dependability with hybrid thread- and instruction-level redundancy* (DHR). The MCCP changes its dependability mode according to required performance. In one case, it is in the dtr mode, and in other case, it is in the dir mode. Exploiting the adaptability in the dependability mode, the MCCP can hybrid the DTR and the DIR. The main topic of the present paper is how to choose the dependability mode. One of the strategies relies upon programmers. A programmer marks how important every thread is and tells it

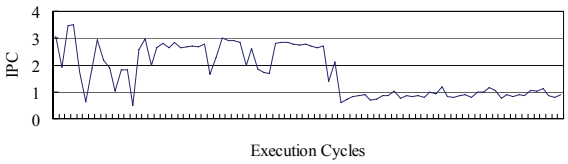

Figure 5. Issue IPC Variation for gcc

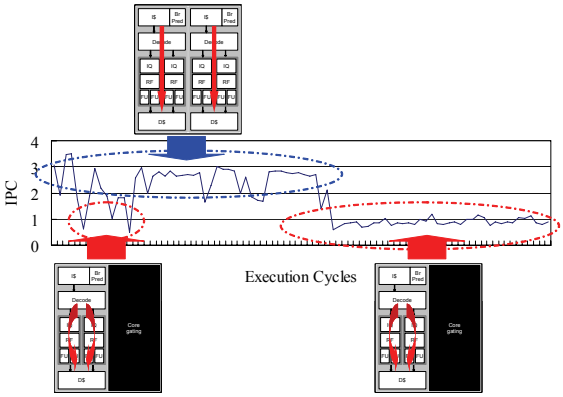

Figure 6. IPC-directed Mode Selection

to hardware (processor) using annotations. Another strategy is OS-based. OS marks the importance of every thread using some metric; for example, deadline time. This paper proposes a fully transparent hardware-based strategy.

The amount of instruction level parallelism (ILP) varies between application programs. If a processor relies upon the DTR, it wastes power consumption when ILP in the application program is small. On the contrary, if a processor relies upon the DIR, performance is severely degraded when ILP is large. Furthermore, the amount of ILP even within a single application program changes by more than a factor of two [2]. Figure 5 shows an example of the issue rate for SPEC2000 CINT benchmark gcc running on a dual-cluster core. The details of the core can be found in Section 3. The horizontal axe indicates the execution cycles and the vertical one represents the average number of instructions issued per cycles (issue IPC) over a window of 10,000 execution cycles. The issue IPC changes by more than a factor of two over a million cycles of execution. If a processor relies upon the DTR, it wastes power during low issue IPC. On the contrary, if a processor relies upon the DIR, performance is severely degraded during high issue IPC. These variations can be exploited to determine the dependability mode.

As explained above, the DTR wastes power consumption during low issue IPC and the DIR degrades performance during high issue IPC. The observations lead us to hybrid the DTR and the DIR. The hybridized technique is called the DHR, as mentioned above. The DHR utilizes the dtr mode only when issue IPC is high and similarly utilizes the dir mode only when issue IPC is low, as sown in Figure 6. When issue IPC is low, there are idle execution resources and thus the dir mode provides dependability without serious performance loss. In addition, since the dtr mode is occasionally turned off, the wasted power consumption is

Table 1. Processor Core Configurations

Fetch width	8 instructions
L1 instruction cache	16K, 2-way, 1 cycle
Branch predictor	1K-gshare + 512-BTB
Dispatch width	4 instructions
Instruction window size	16 entries / cluster
Issue width	2 instructions / cluster
Commit width	4 instructions / cluster
Integer ALUs	2 units / cluster
Integer multiplires	2 units / cluster
Floating ALUs	2 units / cluster
Floating multiplires	2 units / cluster
L1 data cache ports	1 port / cluster
L1 data cache	16K, 2-way, 1 cycle
Unified L2 cache	512K, 2-way, 10 cycles
Memory	Infinite, 100 cycles

eliminated.

It is assumed that past program behavior indicates future behavior. Based on past issue IPC, future issue IPC could be predicted. In order not to use a floating-point divider, the number of instructions issued over a moving sampling window of a fixed width is measured. Future issue IPC is predicted based on the past number of issued instructions rather than on past issue IPC. Predicted issue IPC is used for the mode selection. If it is smaller than a predetermined threshold value (T_{T2I}) in the dtr mode, the DHR switches into the dir mode. Similarly, if it is larger than another predefined threshold value (T_{I2T}) in the dir mode, the DHR switches into the dtr mode.

3. Evaluation Methodology

SimpleScalar/PISA tool set [1] is used for architectural-level simulation. The MCCP consisting two cores, each of which consists of two clusters, is used. Table 1 summarizes the configurations of one core. The front-end and L1 caches are shared by two clusters in a core. L2 cache is shared by two cores. In the case of the dtr mode, the overhead of synchronizing two cores to compare results from them is not included in the evaluations. Six integer programs from SPEC2000 CINT benchmark are used.

Evaluations consist of two phases. First, threshold values for switching the dependability mode are determined. In order to find the potential of the DHR, fine-grain switching is chosen in this phase. The sampling window of 256 cycles is used, and it is assumed that the mode switching does not suffer any performance penalty. In this phase, for each program, 200 million instructions are skipped before actual simulation begins. After that each program is executed for 100 million instructions. After the threshold values are determined, the trade-off between power and performance of the DHR is evaluated. In this phase, some practical assumptions should be considered. When the mode changes from the dir mode to the dtr mode, it is assumed that the overhead of 100 cycles occurs. When such a large overhead is considered, any small sampling windows will not work well and thus the sampling window of 10,000 cycles is used.

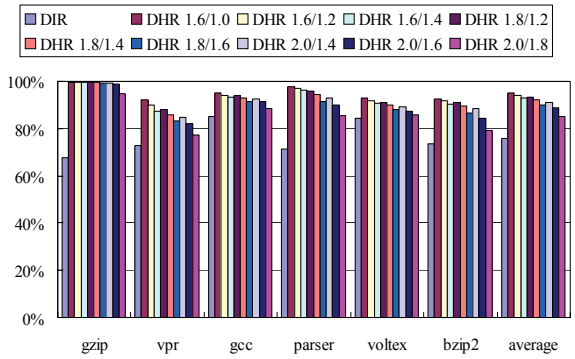

Figure 7. Impact on Performance

In this phase, for each program, 1 billion instructions are skipped before actual simulation begins. After that each program is executed for 2 billion instructions.

As for power consumption, a very simple assumption, where power consumption is proportional to the number of active cores, is used. Hence, the processor consumes two times larger power in the dtr mode than in the dir mode. As for the quality of dependability, it is assumed that both dependable techniques and both dependability modes in the DHR provide equivalent quality. Actually, both the DTR and the dtr in the DHR can provide better quality than the DIR or the dir in the DHR does, since the former ones protect larger portions in processors than the latter ones do [3]. This paper focuses on trade-off between power and performance, and hence this simple assumption is used.

4. Results

Table 1 explains that the issue IPC is up to 2 and 4 for a cluster and for a core, respectively. Therefore, the threshold values for mode switching will be around 2 (x sampling window size, i.e. 256). Values 1.6, 1.8, and 2.0 (x 256) for T_{T2I}, and 1.0, 1.2, 1.4, 1.6, and 1.8 (x 256) for T_{I2T} are considered. Figure 7 presents processor performance when T_{T2I} and T_{I2T} are varied independently. The average number of instructions committed per cycle (commit IPC) is used as a metric. Each value is normalized by the commit IPC of the processor that relies upon the DTR. For each group of 10 bars, the first one indicates the normalized commit IPC of the processor that relies upon the DIR. The rest ones are for the processor that utilizes the DHR. The denotation (X/Y) explains the combination of T_{T2I} and T_{I2T}. For example, 1.6/1.0 explains as follows. The dtr mode switches into the dir mode when the issue IPC in the last sampling window is smaller than 1.6. Similarly, the dir mode switches into the dtr mode when the issue IPC in the last sampling window is larger than 1.0.

It is observed that relying only on the DIR degrades processor performance seriously. On average, performance loss is 25%. The DHR mitigates the loss. For example, 1.6/1.0 reduces the loss to only 5%. Smaller T_{T2I} and smaller T_{I2T} prefer the dtr mode. On the contrary, larger T_{T2I} and larger T_{I2T} prefer the dir mode. It can be seen that combinations that prefer the dtr mode suffer less performance loss. This agrees with the intuitiveness.

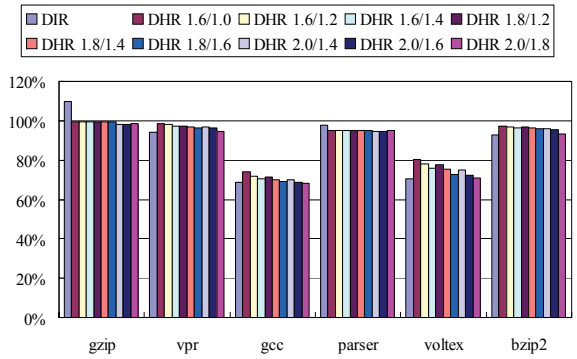

Figure 8. Impact on Energy Efficiency

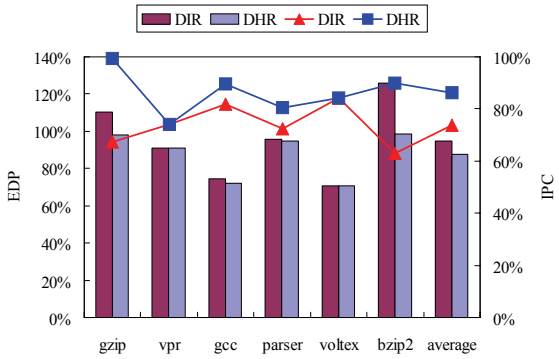

Figure 9. Relative IPC and EDP of Hybrid and I-level

However, it should be noted that the dtr mode consumes larger power than the dir mode does. Next, energy efficiency is investigated.

Figure 8 presents how energy efficiency is affected. Energy-delay product (EDP) is used as a metric. Smaller EDP means better in energy efficiency. The layout of Figure 8 is same to that of Figure 7. Each value is normalized by the EDP of the processor that relies upon the DTR. It is easily observed that energy efficiency is improved as T_{T2I} and T_{I2T} become large. This is because larger T_{T2I} and T_{I2T} prefer the dir mode, which is lower in power consumption than the dtr mode. Due to the same reason, relying only on the DIR improves energy efficiency best for some programs.

Considering commit IPC and EDP above, T_{T2I} and T_{I2T} are determined to be 2.0 and 1.6, respectively. The reason why 1.8 is not chosen for T_{I2T} is that more than 15% performance degradation is undesirable.

Next, the trade-off between power and performance of the DHR is evaluated. Based on the considerations above, the threshold values of 2.0 and 1.6 are used for T_{T2I} and T_{I2T}, respectively. Different from the previous evaluations, here, it is assumed that the overhead of 100 cycles is required when the mode changes from the dir to dtr, and that the sampling window is 10,000 cycles.

Figure 9 shows commit IPC and EDP of the processor that relies on the DHR and those that relies on the DIR. Line graphs present commit IPC and bar graphs present EDP. The line graph with triangle indicates commit IPC of the processor that relies on the DIR and the one with rectangle indicates IPC of that utilizes the DHR. For each group of two bars, the left bar indicates EDP of the processor that relies on the DIR and the right one indicates EDP of that utilizes the DHR. Every value is normalized by the corresponding value of that relies upon the DTR. Comparing with the DTR, the DIR reduces commit IPC by 27% on average, while it improves EDP by only 5% on average. The DHR achieves both mitigation in performance loss and further improvement in EDP. Comparing with the DTR, the DHR reduces commit IPC by 13% on average, while it improves EDP by 13% on average. It should be noted that the DHR improves EDP for all programs, while the DIR degrades it for two programs. The results confirm that the DHR solves the both problems of power consumption in the DTR and performance loss in the DIR.

5. Conclusions

Emerging multicore processors are favorable to attack the soft-error problem, since they can exploit thread level redundancy to detect SEUs. Unfortunately, however, they are power hungry. This paper proposed to hybridizing thread- and instruction-level redundancy and to select one from them according to performance required by the current thread. Thread level redundancy is exploited during high ILP, while instruction level one is exploited during low ILP. Detailed simulations unveil that the hybrid techniques improves EDP by 13% with performance loss of 13% when it is compared with the naïve technique that relies upon only thread level redundancy. In contrast, the technique that relies upon only instruction level redundancy improves EDP by only 5% with severe performance loss of 27%. This confirms that the hybrid technique on the MCCP works well on trade-off between power and performance.

Acknowledgements

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

References

[1] T. Austin et al., SimpleScalar: an infrastructure for computer system modeling, IEEE Computer, Vol. 35, No. 2, 2002.

[2] R. I. Bahar et al., Power and energy reduction via pipeline balancing, 28th Int. Symp. on Computer Architecture, 2001.

[3] T. Funaki et al., Dependability-performance trade-off on multiple clustered core processors, 4th Int. Workshop on Dependable Embedded Systems, 2007.

[4] T. Karnik et al., Characterization of soft errors caused by single event upsets in CMOS processes, IEEE Trans. on Dependable and Secure Computing, Vol. 1, No. 2, 2004.

[5] R. Kumar et al., Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction, 36th Int. Symp. on Microarchitecture, 2003.

[6] T. Sato et al., Evaluating low-cost fault-tolerance mechanism for microprocessors on multimedia applications," 8th Pacific Rim Int. Symp. on Dependable Computing, 2001.

[7] T. Sato et al., Multiple clustered core processors, 13th Workshop on Synthesis and System Integration of Mixed Information Technologies, 2006.

[8] K. Sundaramoorthy et al., Slipstream processors: improving both performance and fault tolerance, ASPLOS-IX, 2000.