Vol.2014-ARC-209 No.9
2014/3/7

An Asynchronous Commit DMR Architecture for Aggressive Low-Power Fault Toleration

Yuttakon Yuttakonkit[1,a)] Jun Yao[1,b)] Yasuhiko Nakashima[1,c)]

Abstract: Dual modular redundancy (DMR) execution is commonly used in many high-end processor platforms to tolerate the increasing transient faults caused by single event effects (SEEs) along the advances of process technology. As the operations must be performed twice to facilitate the comparison based error detection, the power consumption efficiency is always an important issue for dependable systems. To reduce the power consumption, dynamic voltage scaling (DVS), together with Razor FF, have been proposed and well used to lower the voltage to a balancing point for an optimal energy reduction. However, simply combining a DMR and Razor-FF will easily have performance impact as the synchronous committing logic in the traditional DMR architecture does not work well with the dynamic frequency in a Razor FF processor.
In this research, we propose a globally asynchronous locally synchronous DMR architecture that uses dedicated clocks on each DMR module. FIFOs and delay buffers are additionally added and well controlled to guarantee the data checking inside this asynchronous system for both soft and timing error. Compared to the traditional synchronous DMR system, we can have around 10% performance improvement by this asynchronous committing scheme when a same power reduction ratio is assumed. On the other hand, voltage can be aggressively tuned in either DMR module to achieve 12% better MIPS/W without major down-gradation of the performance.

1. Introduction

In the past decades, the design, manufacturing and utilization of microprocessors have been extensively improved for an optimal performance under a limited power budget, or a minimal energy consumption which can still meet the throughput requirement. The architecture designers work towards this target via optimally arranging microprocessor elements and their working configurations and modes. The working frequency was firstly improved by the state-of-art tunings on the critical paths, and now many-core architecture is well used to increase the per area computation density for higher performance. Meanwhile, material technologies on semiconductor fabrication provide an exponentially scaling of design improvement opportunities. However, so far with all these improvement scenarios being applied, CMOS size is now miniaturized and the working voltage is low enough to be easily effected by particle and radiations, which is then reflected as transient errors and causes system failures [1]. The up-scaled integrity adds to this pressure as exponentially more vulnerable transistors are inside the system. A research, which studies the single event effects (SEEs) on supercomputing machines, pointed out the more than tens of transient errors can be visible in a supercomputing system consisted of many GPGPUs, leading to a mean time to failure (MTTF) in a several hours' order [2]. Also, another example gives that with the large number of cores in K-computer, achieving a peak performance for more than 30 hours' calculation requires that each node can deliver a MTTF longer than 500 years, which is an extremely high pressure to the yielding.

With the increasing correctness challenges from SEEs, dual or triple modular redundancies (DMR/TMR) are largely used to protect critical units in high end platforms, such as IBM g5 microprocessor and its successors, part of circuits in Fujitsu SPARC platforms [3], and so on. In paper [1], a dynamic adaptive redundancy architecture (DARA) that uses mainly DMR to achieve the TMR equivalent reliability is given in detail. However, all these reliable systems require multiple circuit or unit sets to perform redundant executions either per operation or per thread granularity. The low energy efficiency, which comes from the use of at least twice energy and the delivery of an only 1x throughput, is always a big concern in this field. In an environment where the total energy budget is given, such as in the satellite where dependability is highly required while the electricity is powered by the solar panels, or an environment where power utilization is constrained by the utilization wall, the drawback of this low energy efficiency will easily be visible, as additional performance will be traded-off to meet both the power constrains and dependability requirements. Therefore, in this research, we are targeting at the goal to add back some power efficiency for such high dependability requirement systems.

Specifically, we target at both power reduction from the DMR processor floor-planing stage and the low-power execution mode of the baseline DMR processor. Traditional DMR architectures

[1] Computing Architecture Lab, Graduate School of Information Science, Nara Institute of Science and Technology, Takayama-Cho 8916-5, Ikoma, Nara 630-0192, Japan
[a)] yuttakon-y@is.naist.jp
[b)] yaojun@is.naist.jp
[c)] nakashim@is.naist.jp

such as IBM g5 [4] and DARA [1] use synchronous redundant executions and data sanity checks. Many previous studied have pointed out that the clock source is a dominant energy consumer in modern micro processors [5]. This situation will be even worse when the clock tree needs to reach the ends of duplicated modules in a DMR processor. Both the design constraint and working clock skews will increase the power consumption. In our work, we try an alternative way to ensure the use of dedicated clock trees in each redundant module, removing the requirement for a globally synchronous clock tree. Also, we try to extensively lower the supply voltage until the unsafe zone of supply voltage is reached, as our low-power execution mode. Razor FFs are considered to be used to tolerate the possible timing faults from these overly lowered voltage. However, as will be introduced in latter sections, traditional synchronous DMR methods are not well compatible with the bubble cycles in a system with Razor FF during the timing fault recovery. Again, the dedicated clock signals in individual DMR modules can be used and tuned to solve this problem, making Razor FF compatible in the DMR logics.

In summary, the major contributions that this work provides are:

(1) A alternative solution to combine both energy reduction techniques and the transient fault toleration: Due to the voltage-frequency interference, voltage down-scaling technology is hard to be combined with traditional synchronous DMR architectures. With our work, the abilities of extreme dynamic voltage scaling by Razor FF and the dependability in addressing SEEs in DMR architecture can be achieved in one system with very few performance impacts.

(2) The removal of global clock tree in the traditional synchronous DMR architecture. Traditional duplicated DMR logics extends the length of clock tree which further increases the clock delay and manufacturing cost, especially for complicated dependable processors. The schemes in this work retires a strict synchronization between the duplicated modules and allows to use different clock trees in the DMR modules. This lowers the requirement for the tightly coupling between DMR modules, which can expect to contribute to the cost reduction in the processor yielding.

(3) Aggressive supply voltage change to make full use of the asynchronous units: We also go one step further to aggressively adjust the supply voltage to explicitly incur timing faults for better MIPS/W in each DMR module. This supply voltage control can keep the asynchronous module near to the saturate state. The performance interference is still small, while the power can be saved for about 12% from the additional voltage control.

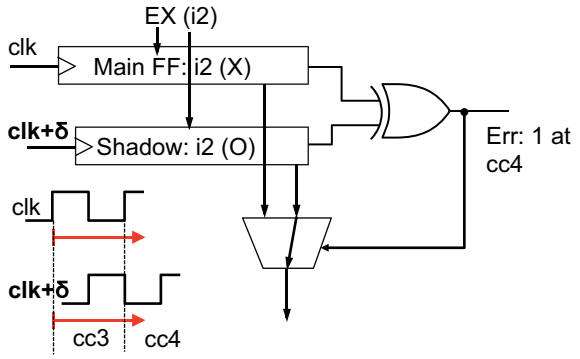
The rest of this paper is organized as follows. Section 2 tries to give the problem demonstration when both transient and timing faults are in consideration. Section 3 describes the proposed architecture mainly focused on how to handle the error detection and recovery in a locally asynchronous system for high dependability. Section 4 shows the experiments, tools and evaluation results. Section 5 concludes the paper.

2. Background: the Incompatibility between Razor FF and Redundant Architecture

A traditional way to lower the power consumption of a microprocessor is via the supply voltage control, which is now widely used in most commercial processors, known as the dynamic voltage scaling (DVS) technique. However, according to the fundamental of value switching in a transistor, a state change of the transistor is firstly triggered by the input electronic signals and then flips the connection between either the power and ground to charge or discharge the capacitance, where the charging/discharging periods are thereby accumulated into the circuit delay. This delay constrains the processor performance via the definition of frequency. During the design and manufacturing phases, processor will be configured in the optimal operation point that has minimum sufficient supply voltage or top working frequency to ensure that the whole circuits can operate correctly. In addition, uncertain environments, such as the ambient temperature and IR-Drops that affect the circuit performance, may make circuits not able to catch up with the clock frequency and then trigger the erroneous states, known as timing errors. These possibilities of timing errors are also taking into account during the manufacturing, which is represented as the voltage definition margin. This voltage margin is a guard-band for tolerating the possible IR-drops during the usage, which is against the energy efficiency but is however necessary where worst cases may occur.

Ernst et. al. [6] has proposed Razor Flip-Flop (FF) for the removal of this voltage guard-band to achieve extreme high power efficiency by intentionally reducing the voltage to below the safe voltage point. A set of main and shadow FF/latch, called as Razor FF, which receives data from the same source at a controlled timing phase shift, is assembled into each stage to detect timing faults, as shown in **Fig. 1**(a). The lowered voltage without considering margins can still hold the delays for most operations, which are referred as normal cases. In the worst cases, however, the unsafe voltage will incur timing faults, which can be detected by a comparison between the main and shadow memory. Under timing faults, the shadow memory can provide the correct data source for the latter stage. However, under this circumstance, a one-cycle bubble will occur, as shown in Fig. 1(b). There are also other recovery scheme of Razor FF which uses a pipeline flush-like scheme to lower the increase of critical path from the error detection signal. Either of these recovery phases can be regarded as a dynamically changed frequency, which can be denoted as f - Δf where the part of Δf is in proportion to the ratio of detected timing faults.

However, this dynamically changing frequency, as f - Δf, will add difficulties to the combination of Razor FF for further voltage down-scaling and the DMR architecture for SEE addressing. For a simple control, most DMR techniques, such as IBM g5 [4], DMR/TMR in chip multiprocessors [7], and DARA [1], use a synchronous execution of multiple threads to ensure strict comparison is performed at the timing that the data are generated. The synchronous comparison and the simultaneous writeback in the duplicated processor modules afterward also make the recovery easier, which only requires to discard all uncommitted data



(a) Razor FF: error detection and recovery.

Inst.	cc0	cc1	cc2	cc3	cc4	cc5	cc6
i1	IF	ID	EX	MEM	WB		
i2		IF	ID	EX*	Bb	MEM	WB

(b) Razor FF: Bubble cycle

Fig. 1 The structure of Razor FF and its error detection/recovery.

Inst./ pipeline	cc0	cc1	cc2	cc3	cc4	cc5	cc6	cc7
i1 (A)	IF	ID	EX	MEM	WB	Stall due to the synchronous WB.		
i1' (B)	IF	ID	EX	MEM	WB			
i2 (A)		IF	ID	EX	MEM	Stall	WB	
i2' (B)		IF	ID	EX*	Bb	MEM	WB	
i3 (A)			IF	ID	EX	MEM		WB
i3' (B)			IF	ID		EX	MEM	WB

Bubble cycle due to timing fault;
Shadow latch in Razor FF provide a correct data.

Fig. 2 The synchronous hazard due to timing fault in a traditional DMR processor.

in both processor modules and recover to the checkpoint states. Applying Razor FF onto this synchronous checking and write-back system will cause a hazard in the duplicated processor modules even when only one module has timing fault and a dynamic frequency shift, as shown in **Fig. 2**. This kind of hazard due to synchronous comparison will be even threatening when the two modules have different manufacturing characteristics, which is common in the current process technologies due to the high processor variation. A simple calculation can be carried out by assuming that each processor core in a DMR architecture has a 30% possibility of timing fault per operation when the voltage is aggressively down-scaled. The chance for simultaneous correct execution and simultaneous incorrect execution due to timing faults is 49% and 9%. The other part of possibility, as 42%, falls into different execution stats in individual processor cores, which will thus cause synchronous comparison hazards in the synchronous DMR architecture.

3. Our Proposal: Asynchronous DMR to Conceal Synchronous Hazard under Timing Faults

The major inconsistency between the DMR architecture and the application of Razor FF for a better DVS utilization is in the synchronous writeback. In addition, the large clock tree used to synchronize the duplicated modules is also a possible power consumer, which is against the purpose of low-power dependable execution. In this research, we try to address the above problems by isolating the two duplicated modules into each individual zones, working under a globally asynchronous locally synchronous (GALS) fashion. Each DMR component module works with its own clock tree, which can thus have different dynamic frequency shifts caused by Razor FF based timing fault recovery. However, under the asynchronous execution, the comparison and recovery become new problems as keeping an exact lock-step fashioned execution [4] is now impossible. Accordingly, we have the following solutions specially designed for this asynchronous DMR architecture.

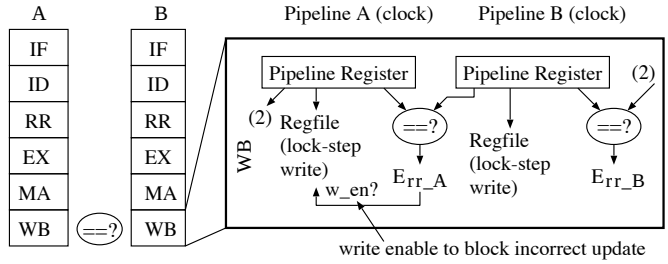
3.1 FIFO based error detection

We used DARA processor as our baseline architecture of a DMR execution processor with full error detection and roll-back based recovery abilities. DARA processor contains two processor pipelines which are driven by a global clock. The instructions are duplicated at the fetch stage by maintaining two copies of program counter (PC). At the beginning of the writeback phase, the committing data, include register file writeback and store address/data, will be compared from the two simultaneous execution. When the values are not identical, a fast recovery will be triggered, following the same scheme of a branch mis-prediction resolution. In this section, we extend the comparison in the write-back stage by using the asynchronous scheme.

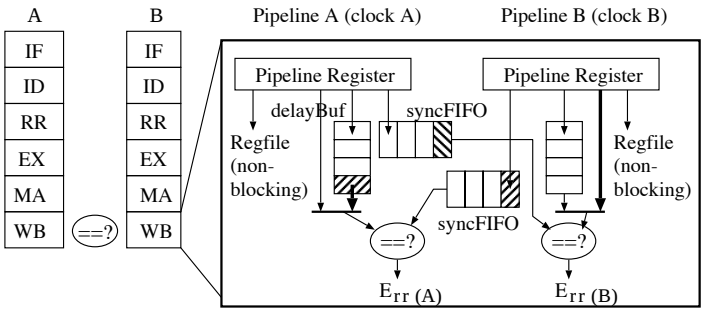
Fig. 3 shows the major change in the error detection logic and the related units between the synchronous and asynchronous DARA processor. In the synchronous mode, the two pipeline registers in the writeback stage in DARA work on the same instructions and work under a same clock. The comparison can be performed directly by using the two outputs from the flip flips before the write back logics, and a lock-step signal will be given when the comparison gives a sign of different data. All the logics in Fig. 3(a) are thus very straightforward and simple.

When the asynchronous mode is introduced, the two pipelines are deliberately designed to be running under their own clock zone. The two clocks can be of a same frequency with some phase shifts between the clock edges. With this asynchronous design, the data to compare for the detection of transient faults are thus from two writeback stages triggered by their own dedicated clocks. There may be some time shift, either within one cycle or several cycles, between the data arrivals of the duplicated instruction set. This time shift may not only comes from the dedicated clocks. As has been introduced in Section 2, the application of Razor-FF can also result into a potential frequency shift. Fig. 2(b) gives a demonstration of this scenario, as in the cc3-th cycle, a timing fault is detected by the Razor-FF in the execution stage and a bubble is therefore added into the pipeline B. In the cc4-th cycle, the recovery takes place by using the data in the shadow latch inside the Razor-FF. By this way, pipeline B looses track with pipeline A, which now respectively makes a slower and a faster cores.

To handle this clock signal mismatch and data verification afterward, a synchronization module to re-stabilize the two copies



(a) Synchronous comparison and committing.



(b) Asynchronous data check, clock A != clock B

Fig. 3 Synchronous data check/committing (previous work) vs. asynchronous data check/committing (proposal).

of identical execution is given in Fig. 3(b). Specifically, we use first in first out (FIFO) memory to store data for comparison between the two cores. Each core will have their own out-bound and in-bound FIFOs. The FIFOs are basically a working module for synchronization, sending values under the clock of data generator pipeline, and reading data at the clock of data receiver pipeline. Note that the out-bound FIFO from pipeline A becomes the in-bound FIFO in pipeline B side, and vice versa. In addition to the data transferring FIFOs, each pipeline will have a delay buffer to store its own values of the processed instructions, which is triggered by its local clock.

As shown in Fig. 3(a) and Fig. 2(b), the synchronous data check scheme will stop the execution of the fast core, to let it wait for the data arrival of the slow core so that simultaneous data commit is guaranteed. Differently, in the asynchronous DMR architecture (Fig. 3(b)), the fast core will commit the processed instruction as soon as possible, while pushing the data values of the committed instruction into its own delay buffer and the out-bound FIFO simultaneously. The slow core, as pipeline B, fetches the oldest instruction result from the in-bound FIFO, comparing with its own execution result for a transient fault detection. The delay buffer in the slow core is not active, as its store depth is zero now under this scenario. Meanwhile, the slow core will also update its execution result with the fast core via another FIFO link. The fast core compares this result with its oldest instruction result in the program order in the delay buffer to detect an execution error. For each FIFO and delay buffer, it requires to have a tail and a head pointer to help write in the local newest data and read out the oldest instruction in the program order. For an out-of-order execution processor, as this FIFO and delay buffer are put inside the WB stage, the order to update these units can still strictly follow the program order, which thus make this asynchronous method applicable even when out-of-order issuing is assumed. When there is an SEE in either of the two executions, both of these two asynchronous pipelines will detect the error if the fault becomes visible.

As can be expected, the depth of the delay buffer and the FIFOs become the major limitation of the timing gap between the two pipelines. If one of the two DMR modules is always working under a lower frequency, the always slow core will easily saturate the FIFOs and the whole system will work under the limited low frequency. However, as has been previously roughly analyzed, in an environment where each operation gets 30% possibility for a timing fault, there is a 42% possibility that only one pipeline is with timing fault. It means that either pipeline A and pipeline B will be dynamically delayed due to timing faults, which can respectively switch the roll of faster and slower cores on-the-fly. This also makes the best expected scenario to apply our asynchronous method. The goal of this research is to find the optimal buffer depth in different applications, which will be fully discussed in the result sections.

3.2 Error recovery

The recovery is also complicated in this asynchronous system, as the fast core may already update its incorrect data into its register file and cache lines before the erroneous execution can be detected. Usually, a checkpoint based recovery will be necessary, as shown in most delay buffer based redundant systems [8]. However, under this situation, the memory and the register file in the slow core, still represent the latest correctly executed instructions, as the erroneous data has not been written back. The register file and the data/instruction cache in the slow core now become the real checkpoint data, which retires the necessity for an additional checkpoint space. **Fig. 4** gives a demonstration of this error recovery triggered by the transient fault detection. Specifically, an control FSM will be used to writeback the D$ content from the slow core to the L2 cache. After that, the FSM will synchronize the data in the register file of the slow core to the fast core. All other memory information can be discarded by invalidating the cache lines then. After these steps, the program can be restarted from the re-fetching of the PC of the latest un-executed instruction, as `i2` in this example. The rollback scheme can be totally the same as in the original DARA system [1]. Note that this method is only possible when no dirty lines in the fast core are updated into the lower caches. Before a dirty line replacement, the fast core is necessary to be stalled to wait for the slow core to cache up. After the memory synchronization, a roll-back based recovery similar to the synchronous DARA can be scheduled to restore the latest correct processor state.

3.3 Active voltage control to achieve more power reduction

The application of this asynchronous DMR system gives another possible use of the voltage control. As the performance of the asynchronous DMR is actually limited by the dynamically slowed down core due to timing faults, the fast core which leads the execution in a short instruction gap can safely take several timing faults before it will influence the DMR performance. In other viewpoints, if the delay buffer and the FIFOs are not satu-

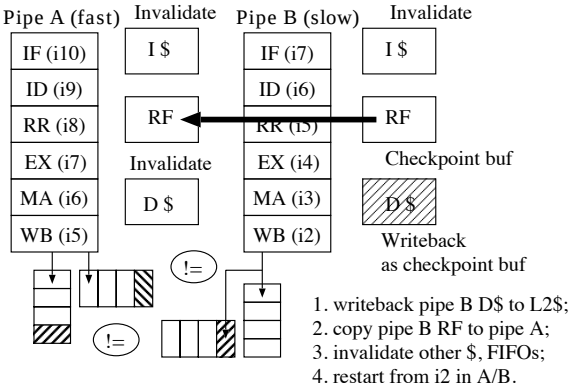


Fig. 4 The recovery scenario in the asynchronous DMR architecture.

rated to trigger synchronization hazards, the performance of the DMR performance will not be hurt by the timing faults in the slow cores.

To make a full use of the FIFO depth, starting from the timing fault free execution, which requires a high voltage in both pipelines, we intensionally increase the error rate inside the system. Using the 30% timing fault possibility per operation in the previous example, only for a 9% possibility, there will be a simultaneous fault pair in both pipelines. In other conditions, the delay caused by timing fault will be buffered inside the asynchronous FIFOs and delay buffers. This buffered delay will be high possibly concealed in some other long pipeline hazards such as branch mis-prediction and L2 cache miss. Especially for the L2 cache miss, when there is no dirty line writeback, the cache miss can be safely triggered by the fast core, providing a fast resolution for the slow core. The possibility of this further voltage reduction by fully using the depth of the FIFOs and delay buffers will be discussed in Section 4.

4. Results

4.1 Methodology

We use a cycle accurate processor simulator to mimic the GALS DMR processor, especially putting emphasis on the asynchronous committing when occasional timing faults are applied onto either DMR module. The ISA that we used in this simulator is SH-2 [9]. The processor parameters are listed in **Table 1**.

To simulate correct timing error, it needs fine-grain data simulation that can access and calculate all of data bit by bit with amounts of environments variable such as voltage drop, temperature dissipating or clock slack with all of these variable precisely concerned This complicated simulation can however be largely accelerated by using VARIUS [10], which provides a statistic model for timing error. VARIUS is based on processor floorplan that is physical blueprint of processor with environment variable such as transistor fabrication technology, voltage threshold in each transistor and heat dissipating rate. It provides precisely voltage that processor circuit can operate under frequency or vice-versa. Also with the statistic model it can provide error probability in varies of configuration without data simulation. The model was verified with real-world test data from Razor data [6] that it's precise enough to use as a reference data. **Fig. 5** shows the voltage and timing fault rate configurations that we used in this work.

Feature	Description
Architecture	32-bit internal data bus
General-register file	32-bit general registers × 16
Instruction set	16-bit fixed length Load-store architecture Delayed branch implemented
ALU execution delay	1 cycle for each instruction
Pipeline	6 stages pipeline IF,ID,RR,EX,MEM,WB
Instruction issuing	2 Issue
Instruction commit	In-Order
Instruction Cache	size/assoc/repl = 64kB/4-way/LRU line size = 16 kB miss penalty = 8
Data I1 Cache	size/assoc/repl = 64kB/4-way/LRU line size = 32 kB miss penalty = 8
I2 Cache	size/repl = 2MB/Direct-map/LRU line size = 64 kB miss penalty = 40

Table 1 SH2 Simulator Features Description

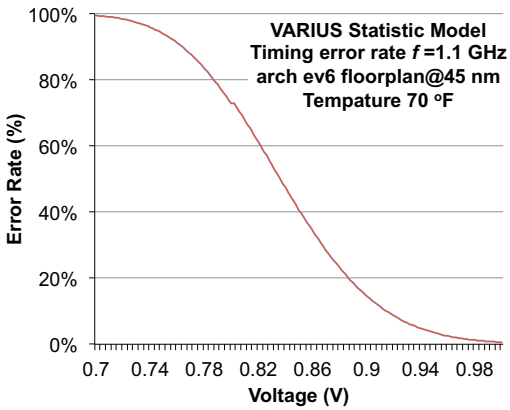


Fig. 5 Supply voltage vs. timing fault rate, from VARIUS model.

4.2 Benchmarks and energy results

We used 10 benchmarks, as Bubble, FFT, Intmm, Mm, Perm, Puzzle, Queens, Quick, Towers, and Trees, from Stanford Benchmark suite to evaluate the performance of our proposed architecture. The timing fault free IPCs of these benchmarks are given in **Table 2**. The timing faults are injected in the execution units in each pipeline individually, following an error rate given by VARIUS system.

Fig. 6 and **Fig. 7** demonstrate the IPC results of our asynchronous DMR architecture, where a 10% and 30% fault rates are respectively assumed. All IPC data are normalized to the IPC of the synchronous committing DMR architecture, which are also listed under the x-axis in the figures. Note that the rate of the transient fault is influenced mainly from the environment and voltage. Our previous research [1] shows that even under a reduced voltage, an transient fault acceleration can get about 2 errors per second, which is a far low fault rate as compared to the timing fault. The system needs to be tested under transient faults to verify a correct error detection and recovery. However, in these experiments, transient faults are not simulated due to its minor impacts on the performance.

From the figure, it can be easily observed that the depth of

bench.	IPC	bench.	IPC
Bubble	1.16	FFT	1.10
Intmm	1.26	Mm	1.14
Perm	1.36	Puzzle	1.08
Queens	0.86	Quick	1.17
Towers	1.27	Tree	0.97

Table 2 IPC of the two-issue processor under timing fault free execution.

.

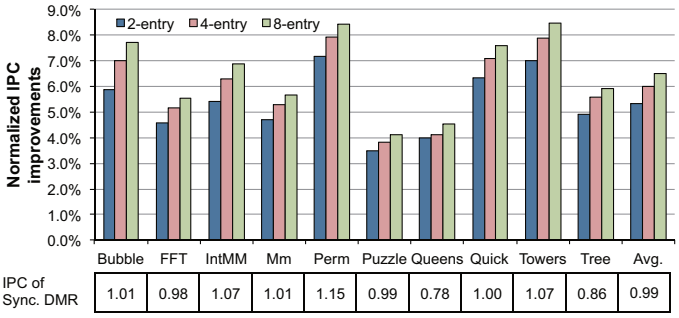


Fig. 6 IPC improvements from sync. to async. under 10% timing faults.

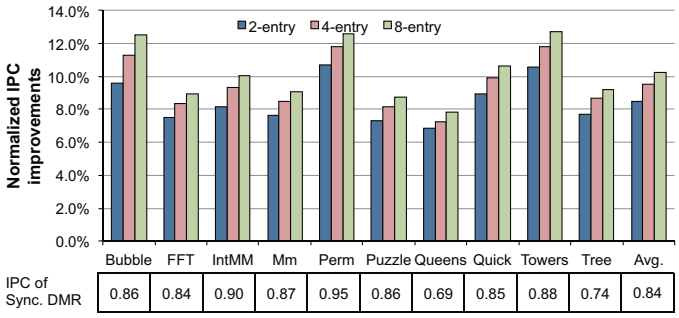


Fig. 7 IPC improvements from sync. to async. under 30% timing faults.

the FIFOs and the delay buffers have a direct effect on the IPC improvement when asynchronous DMR architecture is applied. The reason is straightforward that with more depth, it can hold a larger execution gap between the two asynchronous working pipelines. Eventually, when the fast core gets some timing faults during the period that the slow core is catching up, the FIFOs can hold even longer before a synchronous hazard will be triggered. However, a deep FIFO also requires more energy consumption in the FIFO and buffer memories, and the control logics as well. Meanwhile, it can also be directly observed from these data that from the synchronous execution to the 2-entry FIFO execution, the asynchronous system provides the major IPC improvements. The efficiency of deep in FIFOs and delay buffers goes smaller when the depth is continuously increased. The characteristics of benchmarks also provide a key influence in the selection of the FIFO depth. Other than the findings that different programs will have different IPC improvements under same settings, Bubble and Towers show a tendency to visibly require a deep FIFO depth, while Queens and Puzzles present an image that FIFO depth is easily saturated for IPC improvements. We believe that these differences are mainly caused by the instruction level parallelism (ILP) and the cache miss ratio. Because the delay of the timing fault only causes a one-cycle bubble, the performance impact is highly possible to be hide under other pipeline hazards

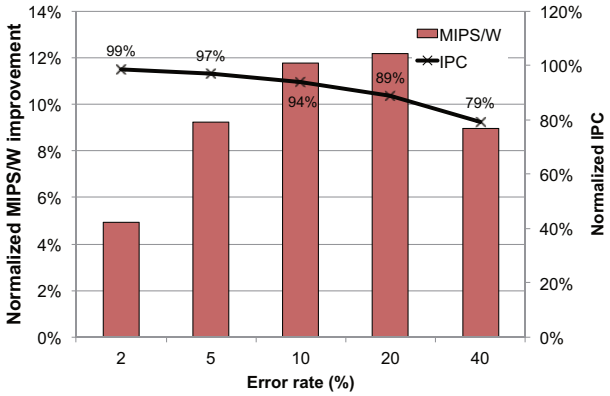


Fig. 8 MIPS/W and IPC of aggressive voltage control in the async. DMR architecture.

before it results visible synchronous hazards. More detailed analysis based on the program characteristics will be one of our major future work. This may also indicate a further possibility for more detailed voltage control according to the understanding of the program characteristics.

In average, the asynchronous execution can achieve around 6% and 10% IPC improvements as compared to the traditional synchronous DMR architecture by using the same voltages and frequencies under these fault rates. The working power is not going to change due to the voltage and frequency but the short execution time will finally make a contribution to the energy or other energy metrics like EDP or EDDP.

In addition, from another viewpoint, we can try to make a full use of the FIFO depth to achieve power reduction by aggressively applying DVS on the proposed asynchronous DMR architecture. we also provided a MIPS/W study by assuming that voltage is aggressively lowered to increase the timing fault rate and use the asynchronous FIFOs and delay buffers to wait for long pipeline hazards to conceal the delay from the timing faults. **Fig. 8** shows the result of these aggressive voltage control results. Without major IPC drop at the 10% fault rate, we can achieve a 12% MIPS/W improvement as compared to the timing fault free execution by aggressively setting the voltage control.

5. Conclusion

In this paper, a GALS DMR based processor architecture has been purposed and described to optimally combine the abilities from Razor-FF to extremely lower the working voltage and the DMR architecture for an SEE addressing. By using dedicated clock source on each core, it can help the DMR processor to add back 6% to 10% performance when the voltage is down-scaled to allow 10% and 30% possibility of timing faults. This also opens a possibility to aggressively lower the supply voltage to make a full use of the asynchronous logics. Our study gives that under an 8-entry FIFO depth, it is possible to allow 20% faults without major performance down-gradation. The MIPS/W of the 20% timing fault execution within our asynchronous architecture is 1.16x of the original execution under a timing error free configuration.

Acknowledgments This work is supported by VLSI Design and Education Center (VDEC), University of Tokyo with the collaboration of Synopsys Corporation, Cadence Design Systems,

and Mentor Graphics. This work is supported by KAKENHI (No. 24240005, No. 24650020, and No. 2370060), and STARC IS program.

References

[1] Yao, J., Okada, S., Masuda, M., Kobayashi, K. and Nakashima, Y.: DARA: A Low-Cost Reliable Architecture Based on Unhardened Devices and Its Case Study of Radiation Stress Test, *Nuclear Science, IEEE Transactions on*, Vol. 59, No. 6, pp. 2852–2858 (online), DOI: 10.1109/TNS.2012.2223715 (2012).

[2] Rech, P., Frost, C. and Carro, L.: Degree of Parallelism Variation Effects on GPUs Reliability, *Radiation Effects on Components and Systems 2013* (2013).

[3] Kan, R., Tanaka, T., Sugizaki, G., Ishizaka, K., Nishiyama, R., Sakabayashi, S., Koyanagi, Y., Iwatsuki, R., Hayasaka, K., Uemura, T., Ito, G., Ozeki, Y., Adachi, H., Furuya, K. and Motokurumada, T.: The 10th Generation 16-Core SPARC64 ™Processor for Mission Critical UNIX Server, *Solid-State Circuits, IEEE Journal of*, Vol. 49, No. 1, pp. 32–40 (online), DOI: 10.1109/JSSC.2013.2284650 (2014).

[4] Slegel, T. J., Averill, R. M., I., Check, M. A., Giamei, B. C., Krumm, B. W., Krygowski, C. A., Li, W. H., Liptay, J. S., MacDougall, J. D., McPherson, T. J., Navarro, J. A., Schwarz, E. M., Shum, K. and Webb, C. F.: IBM's S/390 G5 Microprocessor Design, *Micro, IEEE*, Vol. 19, No. 2, pp. 12–23 (online), DOI: 10.1109/40.755464 (1999).

[5] Tiwari, V., Singh, D., Rajgopal, S., Mehta, G., Patel, R. and Baez, F.: Reducing power in high-performance microprocessors, *Design Automation Conference, 1998. Proceedings*, pp. 732–737 (1998).

[6] Ernst, D., Kim, N. S., Das, S., Pant, S., Rao, R., Pham, T., Ziesler, C., Blaauw, D., Austin, T., Flautner, K. and Mudge, T.: Razor: a low-power pipeline based on circuit-level timing speculation, *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pp. 7–18 (online), DOI: 10.1109/MICRO.2003.1253179 (2003).

[7] Gomaa, M., Scarbrough, C., Vijaykumar, T. N. and Pomeranz, I.: Transient-Fault Recovery for Chip Multiprocessors, *Proceedings of the 30th annual international symposium on Computer architecture*, pp. 98–109 (online), DOI: http://doi.acm.org/10.1145/859618.859631 (2003).

[8] Reinhardt, S. K. and Mukherjee, S. S.: Transient Fault Detection via Simultaneous Multithreading, *Proceedings of the 27th annual international symposium on Computer architecture*, pp. 25–36 (online), DOI: http://doi.acm.org/10.1145/339647.339652 (2000).

[9] Renesas Technology: *SH-1/SH-2/SH-DSP software manual Rev. 5.00* (2004).

[10] Sarangi, S., Greskamp, B., Teodorescu, R., Nakano, J., Tiwari, A. and Torrellas, J.: VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects, *Semiconductor Manufacturing, IEEE Transactions on*, Vol. 21, No. 1, pp. 3–13 (online), DOI: 10.1109/TSM.2007.913186 (2008).