

# Conjoined Processor: A Fault Tolerant High Performance Microprocessor

Viswanathan Subramanian, Naga D. Avirneni, and Arun K. Somani

Dependable Computing and Networking Laboratory

Iowa State University, Ames, IA 50014

Email: {visu,avirneni,arun}@iastate.edu

**Abstract**—Reliability has become a serious concern as systems embrace nanometer technologies. Current reliability enhancement techniques cause slowdown in processor operation. In this work, we propose a novel approach that organizes redundancy in a special way to provide high degree of fault tolerance. Our approach improves performance by reliably adapting the system clock frequency during run time, based on the current running application and environmental conditions. The organization of redundancy in the proposed *conjoined processor* supports overclocking, provides concurrent error detection and recovery capability for soft errors, timing errors, intermittent faults and detects silicon defects. The fast recovery process requires no checkpointing and takes three cycles. Post-layout timing annotated gate level simulations of a conjoined two stage arithmetic pipeline shows that our approach achieves near 100% fault coverage, and a performance improvement of 21%. A five stage in-order conjoined pipeline processor was designed and implemented to verify correctness of the proposed architecture.

**Index Terms**—Microprocessors, Fault tolerance, Redundancy, Adaptive Systems

## I. INTRODUCTION

The impact of soft errors and silicon failures on processor reliability have been steadily rising as we progress towards 32nm technologies and beyond. Reliability issues in combinational logic have become more pronounced and their manifestations result in frequent error occurrence, as we rapidly adopt technological advancements [1]. Modern microprocessors are susceptible to transient faults, induced by high energy radiation and electrical noise, intermittent faults, which occur in bursts at the same location, and permanent faults, resulting from manufacturing imperfections and transistor wear-out [2]. In the past, several fault tolerance techniques have been proposed to tolerate a subset of these faults. The proposed techniques target the entire microprocessor, or particular components, such as the datapath, register file, or control logic. However, these high reliability solutions degrade performance and force processors requiring high performance to sometimes sacrifice reliability.

Duplex systems provide concurrent error detection by means of duplication and comparison [3]. With the advent of Chip Multiprocessors (CMP), fault tolerance techniques that also improve performance have been developed [4], [5], [6]. These approaches utilize two cores to run an application with the goal of executing the application faster than on a single core, while leveraging the redundancy to tolerate faults. The speedup is achieved by exchanging control and data flow information between the two cores. Here, execution is rolled back to a

checkpointed state and instructions are re-executed to recover from a fault.

In this work, we combine duplication and comparison method for fault detection with dynamic reliable overclocking techniques. Current microprocessors, whose operating frequency is fixed based on the worst-case operating conditions and timing paths, allow significant amount of overclocking before encountering timing errors [7]. Having a mechanism that dynamically adapts the system clock frequency based on the current environmental conditions and the current running application, allows significant improvement in processor performance. We propose to overclock until a small tolerable number of timing errors are seen, and overload the fault detection mechanism to tolerate timing errors whenever necessary.

We present a technique that provides tolerance for soft errors, timing errors, silicon defects, intermittent faults, and parameter variations. Our conjoined processor employs a special way to organize redundancy that builds on the better-than-worst-case design methodologies [8] proposed in Razor [9] and SPRIT<sup>3</sup>E [10] microarchitectures. In this processor, both the pipeline registers and the pipeline stage combinational logic are replicated. This processor, when reliably overclocked, executes an application much faster than an unprotected processor, while providing a very high degree of fault coverage.

The organization of redundancy plays a vital role in the concurrent error detection and recovery provided by the conjoined processor. In this processor, both the primary pipeline and the redundant pipeline stages receive inputs from the same primary pipeline register, but store their results in different output registers. The term “conjoined” implies the intertwining of the two pipelines and their constant and continued dependency on each other. Our approach initiates a three cycle recovery process immediately on error detection without requiring the need for any checkpointing. If an error persists after three cycles, the recovery process is triggered repeatedly until the error disappears. After a fixed number of retries, the fault is declared permanent. In that case, the operation is reconfigured to a single-pipeline mode with no fault tolerance and overclocking.

The proposed conjoined processor benefits from a dynamic clock tuning mechanism that is capable of adapting the system clock frequency to the optimal value based on the current executing application and the environmental conditions. The range of frequencies at which the system operates reliably is

estimated based on the implementation of the architecture. The contamination delay, which is the minimum amount of time beginning from when the input to a logic becomes stable and valid to the time that the output of that logic begins to change, of the redundant pipeline is increased to allow a bigger range of possible operating frequencies. The contamination delay of the primary pipeline is not increased, and this allows the operation at much higher frequencies for a set target error rate than Razor or SPRIT<sup>3</sup>E pipeline microarchitectures. This is because, increasing contamination delay increases the longer path delays too, even though the overall worst-case delay is not increased, resulting in more paths to fail for an equivalent increase in clock frequency.

We performed a series of experiments to evaluate the fault tolerance and adaptive clocking capability of the proposed architecture. We designed and implemented a two stage conjoined arithmetic pipeline for this purpose. The first stage performs 64-bit carry look ahead addition, and the second stage performs 32-bit multiplication of the the most significant and least significant words of the adder output. Separate experiments were carried out to verify detection and recovery from soft errors, timing errors, intermittent faults, and permanent fault detection. Our fault injection campaign indicated fault masking in case of soft errors. The output of the pipeline was verified with non fault injection run, and it was made certain that all randomly injected faults were caught. We also designed and implemented a five-stage in-order conjoined pipeline processor. The implemented processor supports operand forwarding to verify correctness in the presence of feedback signals.

The rest of this paper is organized as follows: Section II provides a review of related literature. The conjoined processor architecture and the error detection and recovery methodology is described in Section III. In Section IV, the relevant parameters that affect dynamic frequency scaling are discussed, and the possible range of operating frequencies are derived. Experiments and results are presented in Section V. Section VI concludes the paper.

## II. RELATED WORK

A multitude of fault tolerant architectures have been developed in the past by the research community. Several of these proposed architectures apply fault tolerance with the goal of improving performance past worst-case limits. SSD [11] consists of an integrity checking architecture for superscalar processors that can achieve fault tolerance capability of a duplex system at much less cost than the traditional duplication approach. The REESE architecture [12] takes advantage of spare elements in a superscalar processor to perform redundant execution. DIVA [13] uses spatial redundancy by providing a separate, slower pipeline processor along side the fast processor.

While brute-force overclocking improves performance, it does not guarantee computational correctness. TEAtime [14] scales the frequency of a pipeline using dynamic error avoidance. Our work builds on Razor [9], [15] and SPRIT<sup>3</sup>E [10]

pipeline error detection and recovery mechanisms. Both these techniques use temporal fault tolerance by replicating critical pipeline registers to improve energy efficiency/performance beyond worst-case limits. Of recent interest is the possibility of utilizing two cores to speed up performance and/or improve fault tolerance. Architectures such as, Slipstream [4], Reunion[5], Dual-core Execution [16], and Future Execution [17] exchange control and data flow information between the cores to speed up execution, while leveraging the redundancy to provide partial fault coverage.

## III. CONJOINED PROCESSOR ARCHITECTURE

In a conjoined processor, the entire pipeline is duplicated, and the two pipelines are interlinked in a way so as to provide tolerance for both soft errors and timing errors. The organization of redundancy plays a vital role in the concurrent error detection and recovery provided by this processor architecture. In this architecture, both the primary pipeline and the redundant pipeline stages receive inputs from the same primary pipeline register. However, their results are stored in different output registers. The proposed architecture works for any number of pipeline stages, and the possible limitation on the number of stages arise from the global routing of error detection and recovery signals.

The basic principle behind the conjoined processor architecture is to exploit a combination of spatial and temporal redundancy to detect and recover from both timing errors and soft errors. Timing errors occur as the primary pipeline is overclocked to speed up execution. The replicated pipeline is guaranteed to have sufficient time for execution and is free from timing errors. However, both pipelines are susceptible to soft errors. Since the replicated pipeline cannot be trusted to hold the correct value, the error detection and recovery process is complex and requires certain governing conditions. The following subsections outline the microarchitecture description, the error detection and recovery mechanism, and the extent of fault coverage.

### A. Conjoined Pipeline Datapath Description

The proposed conjoined processor pipeline datapath microarchitecture is illustrated in Figure 1. The figure shows three pipeline stages: P-STAGE N-1, P-STAGE N, and P-STAGE N+1. The conjoined processor concept in its entirety is portrayed in the figure, for P-STAGE N. The primary pipeline is called L-PIPELINE (*Leading Pipeline*) and the redundant pipeline as the S-PIPELINE (*Shadow Pipeline*). As can be inferred from the meaning of the words leading and shadow, the L-PIPELINE runs faster than the S-PIPELINE, and the S-PIPELINE trails the L-PIPELINE with the sole purpose of detecting errors in the L-PIPELINE. Any signal, including feedback signals, that goes as input to the L-PIPELINE also goes as input to the S-PIPELINE.

The L-PIPELINE registers are clocked by the leader clock,  $L_{Clock}$ , while the S-PIPELINE registers are clocked by the shadow clock,  $S_{Clock}$ . The delay between the clocking of the

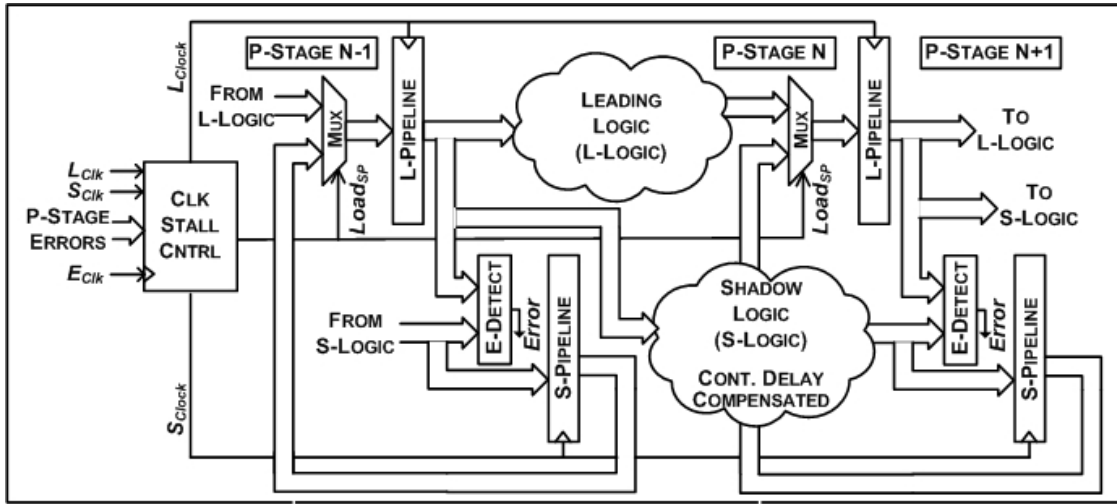


Fig. 1. Conjoined Processor Pipeline Datapath Microarchitecture

L-PIPELINE and the S-PIPELINE registers introduce the necessary temporal redundancy required to detect timing errors.

To detect soft errors, the conjoined processor framework also duplicates the pipeline stage combinational logic between the pipeline registers. The leading logic, L-LOGIC, receives its inputs from the L-PIPELINE registers, and stores its computed results in the next stage L-PIPELINE registers. However, the shadow logic, S-LOGIC, although receives its inputs from the L-PIPELINE registers, stores its outputs in the next stage S-PIPELINE registers.

As mentioned earlier, the output computed by the S-LOGIC is free from timing errors, but susceptible to soft errors. This complicates the error detection and recovery process, as the S-LOGIC outputs cannot be considered as “gold,” and cannot simply be reloaded into the L-PIPELINE registers next cycle on error detection. It is very important to ensure that the next stage S-PIPELINE registers are not corrupted with incorrect result, otherwise recovery will not be possible. As a consequence, error detection is performed before storing results in the S-PIPELINE registers. Only if the results computed by the S-LOGIC matches the values registered in the L-PIPELINE register, then the S-LOGIC outputs are written into the S-PIPELINE register.

The conjoined processor architecture requires three clocks for proper operation. The three input clocks are the leader clock,  $L_{Clk}$ , the shadow clock,  $S_{Clk}$  and the error clock,  $E_{Clk}$ . These three clocks, along with the error signals from all the pipeline stages, control the gating of  $L_{Clock}$  and  $S_{Clock}$ . The  $E_{Clk}$  is required to precisely control when  $L_{Clock}$  and  $S_{Clock}$  need to be stalled to ensure correct operation.  $L_{Clock}$  and  $S_{Clock}$  follow the  $L_{Clk}$  and  $S_{Clk}$  when there are no errors, and on error detection they are stalled during specific cycles to aid the recovery process.

### B. Error Detection and Recovery

Table I shows the possible scenarios where an error can happen in a conjoined processor. Under these circumstances,

values computed by corresponding stages of L-PIPELINE and S-PIPELINE differ resulting in an error. The error detection and recovery mechanism of conjoined processor is so robust that it can handle any number of errors in a single cycle, and all possible combination of errors.

TABLE I  
POSSIBLE ERROR SCENARIOS

Case	L-PIPELINE	S-PIPELINE
1.	Soft Error	No Error
2.	Soft Error	Soft Error
3.	Timing Error	No Error
4.	Timing Error	Soft Error
5.	No Error	Soft Error

The error detection and recovery process does not differentiate between errors occurring in the S-PIPELINE and the L-PIPELINE. Also, there is no possible way to differentiate between soft errors and timing errors. As a result, same recovery process is initiated for any possible error combination occurring in the two pipelines.

The E-DETECT module in Figure 1 compares the value stored in the L-PIPELINE register, and the value computed by corresponding S-LOGIC. This module also incorporates metastability detection, as in [15], for the L-PIPELINE registers, as the L-PIPELINE flip-flops may enter a metastable state when overclocked, or when a soft error reaches the registers during the latching window. The *Error* flag is asserted to indicate an error. The contamination delay of the S-LOGIC needs to be increased to a value more than the delay between the  $L_{Clock}$  and the  $S_{Clock}$ . This is important to ensure that the S-LOGIC outputs are not changed by the values newly registered in the L-PIPELINE registers.

Once an error is detected, a three cycle recovery process is initiated. During the first cycle, the S-PIPELINE register value is loaded into the corresponding L-PIPELINE register. In the second cycle,  $L_{Clock}$  is stalled to ensure that the L-PIPELINE gets two cycles for execution to recover from timing errors. In

the third cycle, normal execution continues. During the entire recovery process, S-PIPELINE is stalled. The recovery process works for any combination of timing errors and soft errors in the two pipelines. If an error persists after three cycles, the recovery process is triggered repeatedly until the error disappears. The proposed solution thus handles intermittent faults. After a fixed number of retries, the fault is declared as permanent. At this point, the operation is reconfigured to a single-pipeline mode with no fault tolerance and overclocking. The reconfiguration can be done using diagnostic test vectors under safe mode of operation on each of the pipelines. Additional circuitry is required to ensure that the processor can run with only one pipeline being operational.

### C. Fault Tolerance Analysis

The possibility of an undetected fault is extremely low in our conjoined processor. One possibility is a timing error happening in the L-PIPELINE and a soft error happening in the S-PIPELINE, and the error flag not being asserted because of identical corruption. This possibility is extremely low since even if a single mismatch happens in the entire system, the error flag is asserted. Timing errors and soft errors affect multiple signals, consequently affecting several flip-flops in the pipeline registers. Another case is when both the S-LOGIC and L-LOGIC are affected by soft errors. The same soft error cannot affect both the logic, if so, then this will be detected by the previous stage E-DETECT module. This is because the L-PIPELINE register outputs feeding the S-LOGIC also goes to the E-DETECT module. Another failure possibility is when a transient pulse occurs right after the error signal is latched and before the S-LOGIC outputs are stored in the S-PIPELINE registers corrupting the S-PIPELINE register values. This duration is extremely small (one NOT and one AND gate, plus global routing delay), and given the distribution of soft errors in time and space, this error possibility is insignificant. The error register is metastability hardened, and any small variation will make the global error signal go high. In essence, the our conjoined processor is capable of providing very high degrees of fault coverage.

## IV. DYNAMIC FREQUENCY SCALING

For proper operation of conjoined processors, it is of paramount importance to respect the timing relationship between the clocks. We need to explore the parameters in a generic way to determine the range of frequencies at which the processor can operate reliably. Figure 2 shows the parameters that control the full range of frequencies,  $F_{Min} \rightleftharpoons F_{Max}$ , that are possible when the conjoined processor is dynamically overclocked beyond the worst-case operating frequency,  $F_{Min}$ .

The following parameters that can be estimated for  $F_{Min}$  settings of any microprocessor are defined below to calculate the dynamic frequency operation range:

Let  $T_{Max}$  represent the worst-case time period required by the microprocessor.

Let  $T_{Err}$  represent the time required for error detection and the assertion of the global error signal.

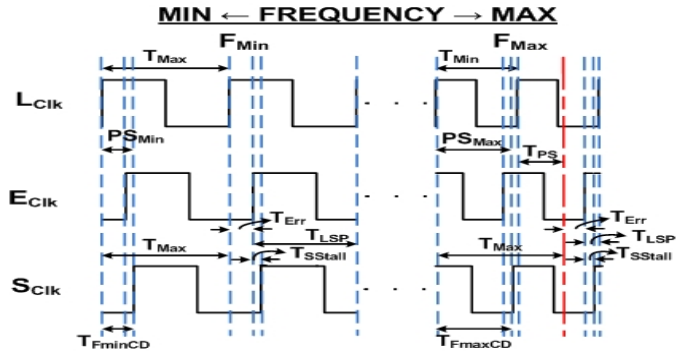


Fig. 2. Dynamic Frequency Scaling

Let  $T_{SStall}$  represent the time required to stall the  $S_{Clk}$  to prevent incorrect values from being loaded into the S-PIPELINE registers.

Let  $T_{LSP}$  represent the time required to assert the  $Load_{SP}$  signal on detection of an error, and the multiplexer delay required to load the S-PIPELINE register values into the L-PIPELINE registers.

Figure 2 shows the time available for the above operations under  $F_{Min}$  and  $F_{Max}$  settings.

$PS_{Min} = T_{Err} + T_{SStall}$  represents the minimum required phase shift to ensure correct operation. Notice in Figure 2 how the above various design parameters affect the relationship between the three clocks. Fixing the phase shift between  $E_{Clk}$  and  $S_{Clk}$  as  $T_{SStall}$  makes dynamic frequency operation easier, since only the phase shift between  $L_{Clk}$  and  $E_{Clk}$  needs to be controlled.

When dynamic overclocking is done to improve performance, the following additional parameters need to be derived for  $F_{Max}$  settings:

Let  $T_{Min}$  represent the minimum clock period at which the conjoined processor is guaranteed to recover.

Let  $PS_{Max}$  represent the maximum phase shift required to ensure correct operation.

Let  $T_{FmaxCD}$  represent the minimum contamination delay of the S-LOGIC of all the pipeline stages.

Depending on the extent of overclocking required, the value  $T_{FmaxCD}$  is fixed at any value within the range given by  $PS_{Min} \leq T_{FmaxCD} \leq T_{Min}$ . If a higher value is chosen, then the contamination delay of the S-LOGIC of the pipeline stages should be increased above this value.

The error detection and if necessary, the recovery should be initiated before the L-PIPELINE registers receive the next set of values and the following relationships must be satisfied.

$$T_{Min} \leq (T_{Max} + T_{Err} + T_{LSP})/2$$

$$PS_{Max} = T_{Max} - T_{Min} + PS_{Min}$$

$$PS_{Max} \leq T_{FmaxCD}$$

Let  $T_{PS}$  represent the adjustable phase shift value. Then  $PS_{Min} \leq T_{PS} + T_{Err} + T_{SStall} \leq PS_{Max}$  and  $0 \leq T_{PS} \leq T_{Max} - T_{Min}$  also must be satisfied.

One important issue is that of clock skew. The effects that lead to variable circuit delays, such as temperature, voltage, and process variations, also cause variations in the clock period, referred to as clock skew. In order to account for this possibility, the worst-case clock skew should be assumed when determining the maximum frequency scaling achievable.

## V. EXPERIMENTS AND RESULTS

To prove the viability of the conjoined processor architecture, we performed the following experimental runs on a two stage arithmetic pipeline. Our designed conjoined system performs 64-bit addition in the first stage, and a 32-bit multiplication in the second stage. The 64-bit carry look ahead adder output is separated into two, and fed to the multiplier as multiplicand and multiplier. The multiplier implementation is based on the high-speed low power design presented in [18].

We synthesized our design in Synopsys design compiler. We used the 0.25um VTVT standard cell library for timing estimation [19]. From static timing analysis reports, we estimated the values of  $T_{Err}$ ,  $T_{SStall}$ , and  $T_{LSP}$ . Then, using the equations derived in Section IV we calculated the values of  $T_{Min}$  and  $PS_{Max}$ . Based on these values, the synthesis was performed again with minimum delay constraints to increase the contamination delay of the S-LOGIC blocks. We used SOC encounter tool to layout the design and to extract standard delay format (SDF) timing information. The post layout timing estimation is provided in Table II. We did VITAL simulations [20] on the SDF annotated post layout design to evaluate fault coverage and performance improvement.

TABLE II  
TIMING INFORMATION

$T_{Max}$	$T_{Min}$	$T_{Err}$	$T_{SStall}$	$T_{LSP}$
18ns	12ns	2.6ns	2.1ns	2.69ns

*Fault Injection:* The conjoined processor architecture is designed to handle all types of faults. Hence, we adopted a fault injection model that causes upsets in the form of transient bit-flips and stuck-at faults. The duration of transient faults is dependent on the environment in which the system under consideration operates and the system clock frequency. Our model considers upsets that last for few hundred pico seconds, and also those that persist for multiple cycles. Faults persisting for multiple cycles model intermittent faults. Stuck-at fault model verifies permanent fault detection functionality. Faults were injected at random locations and time in the design for varying durations.

We performed two different experiments. In the first experiment, for a 10,000 cycle run, we injected 100 faults of one particular type, and evaluated the fault tolerance capability of the design for the particular fault. Table III reports results for the three types of faults injected. The pipeline output is verified for correctness by comparing with a non fault injection run. Random fault injection did not lead to common mode failures. Timing errors may occur as a result of overclocking. In [10], for a multiplier circuit 44% performance improvement was

achieved for an error rate target of 1%. However, because of the limitations imposed on the clock timing requirements, the maximum frequency that is achievable in a conjoined system is limited. Even while running at maximum possible frequency, for randomly generated inputs we observed less than 0.1% timing errors. The timing error recovery process was also verified.

TABLE III  
FAULT INJECTION RESULTS

Fault Type	Masked	Detected	Undetected
Soft Error	74	26	0
Intermittent Fault	0	100	0
Permanent Fault	0	1	0

The dynamic clock tuning mechanism, as in [10], was simulated in VHDL, and this was used to generate the clocks. Figure 3 presents the execution time for three different runs: non fault tolerance mode (NO FT), fault tolerance only mode (FT), and fault tolerance and overclocking mode (*CPipe*). The execution time is for performing 100,000 additions and multiplications. Faults were injected at the rate of 10 per 1000 cycles. The NO FT mode incorrectly completes execution in the presence of faults. The other two modes tolerate all injected faults, and produce results similar to a run with no fault injection. The conjoined Add-Mult system performs better than the non fault tolerance system while offering high reliability. The performance gain is around 21%.

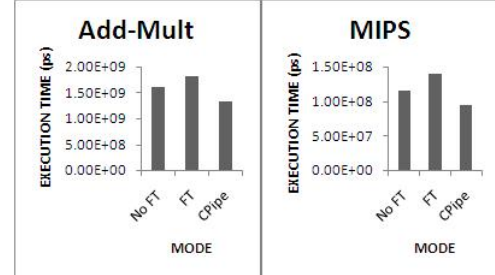


Fig. 3. Execution time

We also designed and simulated a five stage conjoined in-order pipeline processor. The implemented processor supports operand forwarding and is based on the MIPS instruction set architecture [21]. The conjoined processor simulation was carried out differently with delay of the L-PIPELINE stages varied using a random number, and the S-PIPELINE delay values fixed above the contamination delay. We ran three microbenchmarks in succession to evaluate the conjoined processor architecture. The microbenchmarks were written in assembly, and they calculate the first 45 Fibonacci sequence, generate 10000 random numbers, and do 100 element matrix multiplication. The performance of the three modes is shown in Figure 3. The fault injection campaign is similar to the adder-multiplier case. We are currently working on running large scale workload representative benchmarks and back annotated timing simulations for the conjoined MIPS processor.

For our approach, there are no timing overheads on the leading pipeline except for the MUXing-delay in our approach. The error detection is done in parallel with useful computation. Superficially, area overhead is the cost of a second core along with overclocking and error detection overhead. We are working on getting area and power overhead estimates for our approach.

## VI. CONCLUSIONS

Most often than not, system designers need to make difficult tradeoff choice between reliability and high performance. In this paper, we propose a solution that guarantees fault tolerant execution without compromising on the performance of the system. The solution proposed integrates overclocking with redundant execution thereby providing tolerance to soft errors, timing errors, intermittent faults and permanent faults. Conjoined processors rely on the organization of redundancy and adaptive clocking capabilities to improve fault coverage and performance. One of the salient features of our approach lies in the capability to trigger recovery immediately on error detection, without requiring any checkpointing, thereby saving the time and space required to store the current execution status. In essence, conjoined processors present a viable alternative for disjoint cores.

## ACKNOWLEDGMENT

The research reported in this paper is partially supported by NSF grant number 0311061 and the Jerry R. Junkins Endowment at Iowa State University.

## REFERENCES

- [1] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *IEEE/IFIP International conference on Dependable Systems and Networks*, June 2002, pp. 389–398.
- [2] C. Constantinescu, "Trends and challenges in vlsi circuit reliability," *IEEE Micro*, vol. 23, no. 4, pp. 14–19, 2003.
- [3] S. Mitra and E. J. McCluskey, "Which concurrent error detection scheme to choose?" in *ITC*, 2000, pp. 985–994.
- [4] K. Sundaramoorthy, Z. Purser, and E. Rotenburg, "Slipstream processors: improving both performance and fault tolerance," in *ASPLOS-IX*, 2000, pp. 257–268.
- [5] J. C. Smolens, B. T. Gold, B. Falsafi, and J. C. Hoe, "Reunion: Complexity-effective multicore redundancy," in *IEEE Micro*, 2006, pp. 223–234.
- [6] H. Zhou, "A case for fault tolerance and performance enhancement using chip multi-processors," *IEEE Computer Architecture Letters*, vol. 5, no. 1, pp. 22–25, 2006.
- [7] B. Colwell, "The zen of overclocking," *IEEE Computer*, vol. 37, no. 3, pp. 9–12, March 2004.
- [8] T. Austin, V. Bertacco, D. Blaauw, and T. Mudge, "Opportunities and challenges for better than worst-case design," in *Asia South Pacific Design Automation Conference*, 2005, pp. 2–7.
- [9] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *IEEE Micro*, December 2003, pp. 7–18.
- [10] V. Subramanian, M. Bezdek, N. D. Avirneni, and A. K. Somani, "Superscalar processor performance enhancement through reliable dynamic clock frequency tuning," in *IEEE/IFIP International conference on Dependable Systems and Networks*, June 2007, pp. 196–205.
- [11] S. Kim and A. K. Somani, "Ssd: An affordable fault tolerant architecture for superscalar processors," in *Pacific Rim Dependable Computing Conference*, December 2001, pp. 27–34.
- [12] J. B. Nickel and A. K. Somani, "Reese: A method of soft error detection in microprocessors," in *IEEE/IFIP International conference on Dependable Systems and Networks*, 2001, pp. 401–410.
- [13] T. M. Austin, "Diva: a reliable substrate for deep submicron microarchitecture design," in *International Symposium on Microarchitecture*, 1999, pp. 196–207.
- [14] A. K. Uht, "Uniprocessor performance enhancement through adaptive clock frequency control," *IEEE Transactions on Computers*, vol. 54, no. 2, pp. 132–140, February 2005.
- [15] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "A self-tuning dvs processor using delay-error detection and correction," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 4, pp. 792–804, April 2006.
- [16] H. Zhou, "Dual-core execution: Building a highly scalable singlethread instruction window," in *Parallel Architectures and Compilation Techniques*, 2005, pp. 231–242.
- [17] I. Ganusov and M. Burtscher, "Future execution: A prefetching mechanism that uses multiple cores to speed up single threads," *ACM Transactions on Architecture and Code Optimization*, vol. 3, no. 4, pp. 424–449, 2006.
- [18] M. Zheng and A. Albicki, "Low power and high speed multiplication design through mixed number representations," in *IEEE International Conference on Computer Design*, October 1995, pp. 566–570.
- [19] J. B. Sulistyo and D. S. Ha, "A new characterization method for delay and power dissipation of standard library cells," *VLSI Design*, vol. 15, no. 3, pp. 667–678, 2002.
- [20] S. Krolkoski, "Standardizing asic libraries in vhdl using vital: a tutorial," in *Custom Integrated Circuits Conference, 1995., Proceedings of the IEEE 1995*, May 1995, pp. 603–610.
- [21] D. A. Patterson and J. L. Hennessy, *Computer organization and design (2nd ed.): the hardware/software interface*. Morgan Kaufmann Publishers Inc., 1998.