Low Overhead Soft Error Mitigation Techniques for High-Performance and Aggressive Designs

Naga Durga Prasad Avirneni, *Student Member*, *IEEE*, and Arun K. Somani, *Fellow*, *IEEE*

Abstract—The threat of soft error induced system failure in computing systems has become more prominent, as we adopt ultradeep submicron process technologies. In this paper, we propose two efficient soft error mitigation schemes, namely, Soft Error Mitigation (SEM) and Soft and Timing Error Mitigation (STEM), using the approach of multiple clocking of data for protecting combinational logic blocks from soft errors. Our first technique, SEM, based on distributed and temporal voting of three registers, unloads the soft error detection overhead from the critical path of the systems. SEM is also capable of ignoring false errors and recovers from soft errors using *in-situ* fast recovery avoiding recomputation. Our second technique, STEM, while tolerating soft errors, adds timing error detection capability to guarantee reliable execution in aggressively clocked designs that enhance system performance by operating beyond worst-case clock frequency. We also present a specialized low overhead clock phase management scheme that ably supports our proposed techniques. Timing-annotated gate-level simulations, using 45 nm libraries, of a pipelined adder-multiplier and DLX processor show that both our techniques achieve near 100 percent fault coverage. For DLX processor, even under severe fault injection campaigns, SEM achieves an average performance improvement of 26.58 percent over a conventional triple modular redundancy voter-based soft error mitigation scheme, while STEM outperforms SEM by 27.42 percent.

Index Terms—Soft errors, adaptive systems, overclocking, error detection, reliability, performance.

1 Introduction

Nano-sized transistors, coupled with deployment in hazardous environments, have magnified the reliability concerns plaguing modern computing systems. Rapid enhancements in VLSI technology have fueled the increasing apprehension of system hardware being susceptible to myriad of faults. Many fault tolerance techniques are proposed at different levels of design hierarchy, starting from the design of hardened latches to system-level fault tolerant architectures [1], [2], [3], [4]. All these techniques strive to provide high degrees of fault coverage by providing redundancy in either information, spatial, or temporal domains. For example, on-chip memories, which are regularly structured arrays, are protected by Error Correcting Codes (ECC) against transient bit flips, also known as, soft errors [5]. ECC applies information redundancy to mitigate soft errors, while incurring resource overhead, in terms of area and power.

In the past, single event upsets (SEUs) were a major concern in space applications creating hard threats like loss of control, which often lead to catastrophic system failures. An SEU is caused when a high-energy particle, either from cosmic radiation or decaying radioactive material, strikes the silicon substrate. If enough charge is deposited by the strike, it causes a bit flip in the memory cell or a transient pulse in the combinational logic. The latter is referred to as a Single Event Transient (SET). A report by NASA, in September 2009, indicates that cosmic ray intensities have increased 19 percent above the previous space-age highs [6]. Study conducted by Normand in [7], provides recent evidence of upsets at ground level, which implies, terrestrial applications also require fault tolerant techniques to ensure their dependability.

With shrinking transistor feature sizes, supply voltages and node capacitance's of the circuits are getting smaller and smaller. However, this lowers the energy threshold needed by high-energy particles to induce errors. It results in rapid increase in number of particles in the flux that can induce a soft error. As indicated in [8], the problem of soft errors in combinational circuits is becoming comparable to that of unprotected memory elements in current and future technologies. Radiation induced SET pulses have widths in the range of 500 ps to 900 ps in the 90 nm process, as compared to 400 to 700 ps in the 130 nm process [9].

Providing fault tolerance capabilities for random and complex logic is expensive, both in terms of area and power. Techniques such as, duplication and comparison, and temporal triple modular redundancy (TMR) and majority voting have been proposed to mitigate soft error rate in logic circuits [10]. These approaches pay penalty for false errors and incur performance overhead even during error-free operation. Also at this juncture, when static power is comparable to dynamic power, logic replication is not a viable alternative.

Increasing system wide integration forces designers to adopt worst-case design methodologies, while designing

• *The authors are with the Department of Electrical and Computer Engineering, Iowa State University, 2215 Coover Hall, Ames, IA 50011. E-mail: {avirneni, arun}@iastate.edu.*

Manuscript received 8 Feb. 2010; revised 10 Sept. 2010; accepted 5 Jan. 2011; published online 4 Feb. 2011.
Recommended for acceptance by C. Metra.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2010-02-0097.
Digital Object Identifier no. 10.1109/TC.2011.31.

individual system components. With such design practices, safety margins are added to address parameter variations, which include intradie and interdie process variations, and environmental variations, which include temperature and voltage variations [11], [12]. These additional guard bands are becoming non-negligible in nanometer technologies. Designers conservatively add these safety margins to salvage chips from timing failures and shortened lifetime. Most systems are characterized to operate safely below a particular vendor specified frequency. When they are operated beyond this rated frequency, timing errors and system failure may happen.

Overclocking as a means to improve performance is a popular technique among enthusiasts [13]. Microprocessor vendors are even introducing capabilities in the chipset, examples being AMD's Overdrive and Advanced Clock Calibration techniques, to support overclocking. The latest 45 nm AMD Phenom II processor has been overclocked to upward of 5 GHz from its rated frequency of 2.8 GHz, using liquid nitrogen cooling [14]. Frequency binning and discreteness in artifact grading are contributing to this improvement in performance through overclocking. Also, circuits exhibit worst-case delay only when their longest delay paths are sensitized by the inputs. However, these worst-case delay inducing inputs and operating conditions are rare, leading to room for performance improvement that overclockers exploit [15]. The problem is that timing errors may occur at overclocked speeds, so in order to reliably take advantage of this improvement in execution time, the processor must be made in some way tolerant to errors.

Overclocking without guaranteeing functional correctness leads to unpredictable system behavior and loss of data. Aggressive, but reliable, design methodologies employ relevant timing error detection and recovery schemes to prevent erroneous data from being used [16], [17]. In [18] and [19], it has been shown that operating frequency can be increased beyond worst-case limit, allowing systems to operate at optimal clock frequency, by adapting to the current set of instructions and environmental conditions. Performance gains as high as 50 percent were achieved for a small recoverable error rate of one percent.

Safety critical systems with hard real-time constraints require wide fault coverage with no compromise in performance. An interesting capability in nanometer design space, we believe, is to provide soft error tolerant reliable execution for high-performance aggressive designs. In this paper, we propose new ways of designing fault tolerant and reliably overclocked register cells that enable systems to improve both their performance and dependability.

1.1 Our Contribution

In this paper, we address the issue of soft errors in random logic and develop solutions that provide fault tolerance capabilities. For this, we consider the approach of multiple clocking of data for detecting and correcting soft errors in combinational logic, an approach widely used in [2], [10], [20], [21], [22], and [23]. Our first technique, Soft Error Mitigation (SEM), replaces register elements in a circuit with **Soft Error Mitigation** register cells. It allows systems to operate without the overhead of soft error detection circuitry. Unlike earlier approaches proposed using dual modular redundancy (DMR) in space/time and triple modular redundancy in time, SEM technique differs in error detection and recovery process and doesn't incur any performance penalty during error-free operation. SEM completes error detection process with two comparisons and also pays no penalty for false errors. Also on error detection, SEM avoids recomputation and recovers using local recovery. Our second technique, Soft and Timing Error Mitigation (STEM), concurrently detects and corrects soft and timing errors using **Soft and Timing Error Mitigation** register cells. STEM cells have soft error mitigation capabilities comparable to those of SEM cells, and they also support reliable overclocking. Both of our techniques employ a distributed and temporal voting scheme that enables *in-situ* error detection and fast recovery. We support circuit level speculation in both SEM and STEM techniques. We allow data to move forward speculatively, and when an error happens, we void the computation and perform recomputation. Since systems are protected from timing errors that may happen when systems are clocked aggressively, STEM approach allows systems to operate beyond design margins by providing support for reliable dynamic overclocking (DYNOC). Developed from better-than-worst-case design methodologies, dynamic overclocking exploits input data dependencies permitting the clock frequency to increase beyond the critical path delays and allowing systems to operate at optimal clock frequency [18], [19].

For error detection and correction, our temporal data sampling mechanisms, sample data at three different time intervals and thus require three clocks for proper operation. Clock distribution and routing are significant challenges in nanoscale technologies. Clock distribution network (CDN) consumes a significant portion of the power, area, and metal resources in an integrated circuit. As a consequence, a specialized clock generation, distribution, and routing scheme that minimizes the clock distribution overhead incurred by our fault mitigation techniques is important. Also, to support reliable dynamic overclocking, as discussed in [18] and [19], it is important to precisely control the relative phase shifts of clock signals at high frequencies. Therefore, we focus on developing an efficient local clock manager (LCM), which helps in generating the required clock signals, with the desired phase shifts, locally. The clocks, so generated and distributed, satisfy the timing constraints required for proper working of our techniques. We also analyze the area overhead incurred for developing such LCMs.

For our experimental study, we integrated our data sampling mechanisms into a two-stage pipeline consisting of an adder and a multiplier. Our results show that, with STEM cells, performance of this system can be increased by 55.93 percent over conventional TMR schemes, while providing near 100 percent fault coverage. For the experiments conducted on a DLX processor, we observed that SEM technique achieves an average performance improvement of 26.58 percent over the TMR scheme and STEM outperforms SEM by 27.42 percent, while providing near 100 percent fault coverage.

The preliminary idea of SEM and STEM registers cells was reported in [24]. In this paper, we extend our past work by explaining why false errors are of a concern. We also provide the timing relationship that needs to be maintained between the control signals of a pipeline employing STEM cells for all error cases. We also carried out the characterization of temperature effects on our LCM implementation

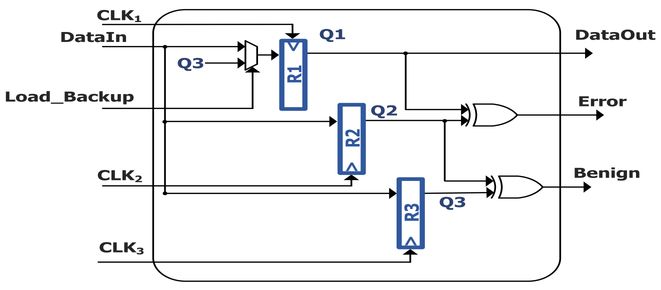

Fig. 1. SEM cell to mitigate soft errors.

and report the new results associated with that study in Section 5. Finally, we incorporated two versions of dynamic clock control schemes designed to support adaptive reliable overclocking and new results are presented in Section 6

1.2 Paper Organization

The remainder of this paper is organized as follows: in Section 2, we describe our soft error mitigation technique and recovery mechanism. Section 3 describes how both timing error and soft error are concurrently detected and corrected. In Section 4, we discuss the issues in designing a pipeline system with our proposed soft/timing error mitigation techniques. Section 5 discusses the design aspects and area overheads of implementing a local clock manager. We present our results in Section 6. Section 7 presents the related work and Section 8 concludes the paper.

2 Soft Error Mitigation for High Performance

2.1 Problem of False Errors

To combat the problem of soft errors, spatial redundancy or time redundancy or both have been used in previously proposed solutions [10]. Dual Modular Redundant (DMR) systems employ two components to detect soft errors and the error recovery process is triggered whenever the comparison between these two components fails. There is an error in the system only if the primary component has an error and not in the redundant component. If we assume, probability for any component (out of the two components) to fail is equally likely, then the false error rate is as high as 50 percent. This can make systems to pay a huge penalty in the form of performance and power during the error recovery process. This penalty is aggravated in TMR systems as the false error rate can shoot up to 66 percent. Moreover, voting process degrades system performance. So, an ideal situation would be to reduce power and performance penalty by lowering the false error rate or completely avoiding them. With SEM cell design, we explain how the problem of false errors or false positives is avoided with the redundancy organization associated with systems employing them.

2.2 SEM Cell Design

Prior soft error mitigation techniques at the circuit level are either based on temporal redundancy, spatial redundancy, or a combination of both. These techniques achieve high degree of fault coverage, while degrading or trading

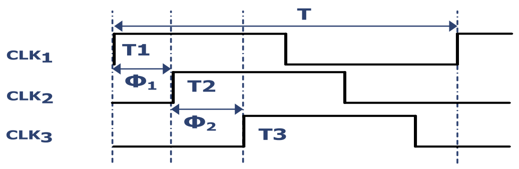

Fig. 2. Timing relationship between clocks for data sampling.

performance, silicon area, and other resources. For example, in [10], a specific design of a voting mechanism based on temporal triple modular redundancy is discussed, which mitigates all single event upsets. However, the overhead incurred is very high, as the operating frequency of a system built with such fault mitigation scheme must include the delays of combinational logic blocks, phase shifts of the clocks, and the delay incurred by the voter. In this section, we present a variant of this scheme, and show that with a combination of local and global recovery, we can remove the additional overhead imposed, by the fault mitigation scheme, on the system operating frequency. The intent of our scheme is to make systems operate at frequencies same as that of nonfault tolerant designs, by unloading the error detection overhead from the circuit worst-case timing delay estimation. To keep the overhead of error detection off the critical path, we present the following redundancy organization using our *Soft Error Mitigation* cells. Fig. 1a shows a gate-level embodiment of a SEM cell. It consists of three registers R_1, R_2, and R_3, clocked by clock signals CLK$_1$, CLK$_2$, and CLK$_3$, respectively. $DataIn$ represents data coming from the combinational logic and $DataOut$ represents the latched data that are sent to the next stages of computation. $Error$ and $Benign$ represent the control signals generated by the voting process that helps in soft error detection and recovery. $Load\_Backup$ represents the control signal that is used to recover from soft error scenarios. Data are sampled at three different time intervals T_1, T_2, and T_3, and are stored in registers R_1, R_2, and R_3, respectively.

2.3 Timing Constraints

Fig. 2 shows the timing relationship between the clock signals and the data sampling intervals. Clock signals, CLK$_1$, CLK$_2$, and CLK$_3$, have the same frequency, but they are out of phase by an amount governed by the timing constraints, explained below. Data are stored in registers at the rising edge of the clock signals, and strict timing constraints are required for efficient mitigation of soft errors. Notations that are used to explain the timing requirements are listed below. Contamination delay is the minimum amount of time beginning from when the input to logic becomes stable and valid to the time that the output of that logic begins to change. Propagation delay refers to the maximum delay of the circuit, under worst-case conditions.

- $T_{CD} =$ Contamination delay of the logic circuit.
- $T_{PD} =$ Propagation delay of the logic circuit.
- $T_{PW} =$ Expected soft error/noise pulse width.
- $\Phi_1 =$ Phase shift between CLK$_1$ and CLK$_2$.
- $\Phi_2 =$ Phase shift between CLK$_2$ and CLK$_3$.
- $T =$ Clock period.

TABLE 1
Possible Soft Error Scenarios of SEM Cell ($\surd =$ No Error; $\times =$ Soft Error)

Case	$\mathbf{R_1}$	$\mathbf{R_2}$	$\mathbf{R_3}$	Error	Benign	Error Recovery
I	$\surd$	$\surd$	$\surd$	0	0	No recovery required
II	$\times$	$\surd$	$\surd$	1	0	Load R_2 or R_3 into R_1
III	$\surd$	$\times$	$\surd$	1	1	No recovery required
IV	$\surd$	$\surd$	$\times$	0	1	No recovery required

Equations (1) and (2) ensure that registers R_1, R_2, and R_3 are not corrupted by the same soft error. Since the system is running at CLK_1 frequency, data are forwarded speculatively to subsequent stages after latching in register R_1, and subsequent stages start their computation immediately.

$$\Phi_1 = T_2 - T_1 \geq T_{PW}, \tag{1}$$

$$\Phi_2 = T_3 - T_2 \geq T_{PW}. \tag{2}$$

Short paths present in the combinational circuit may corrupt the data before they get latched in registers R_2 and R_3. Consequently, it is required to constrain short paths so that the same data registered in R_1 are also latched in registers R_2 and R_3, during error-free operation. Equation (3) ensures that these constraints are met, enabling timing speculation, by increasing the contamination delay above the desired combined phase shift values, given by Φ_1 and Φ_2. Equation (4) makes sure that temporal sampling happens only after the computation by the combinational logic is done. Our technique is capable of detecting all SEUs happening on registers, and all SETs having pulse duration less than T_{PW}.

$$T_{CD} \geq \Phi_1 + \Phi_2, \tag{3}$$

$$T \geq T_{PD}. \tag{4}$$

2.4 Soft Error Detection and Recovery

Table 1 presents the possible soft error scenarios that an SEM technique is capable of detecting and recovering from. The table also lists the corresponding recovery mechanisms used. Once the data are latched in registers R_1, R_2, and R_3, they are compared with each other as shown in Fig. 1a to produce ERROR and BENIGN signals. This comparison operation completes the voting process required to detect soft errors. On error detection, a single cycle system stall is all that is required for complete recovery. Below, we explain the different possible scenarios and the recovery mechanism used when an error happens.

- CASE I. No soft error occurs. Data latched in all three registers are correct. Both ERROR and BENIGN signals stay low, and no recovery mechanism is triggered. System operation continues without any interruption.
- CASE II. A soft error corrupts the data latched in register R_1. ERROR signal goes high after the data are latched in R_2. Since the next stage speculatively uses the data forwarded from R_1, recomputation is required next cycle to ensure functional correctness. The data stored in registers R_2 and R_3 are unaffected by the soft error. During the next cycle, value stored in R_2 or R_3 is loaded back into register R_1 with the help of the control signal LBKUP, completing the local recovery process. Fig. 1a shows R_3 being loaded into R_1. Global recovery, in the form of a stall signal sent to all other SEM cells that are unaffected by the soft error, is initiated and completed in one cycle.
- CASE III. A soft error corrupts the data latched in register R_2. Both the signals, ERROR and BENIGN, go high once temporal data sampling is completed. This is a false positive scenario. No recovery is required as data forwarded to the next stage are correct. System operation is not interrupted.
- CASE IV. This represents a case where register R_3 is corrupted with a soft error. In this case, ERROR signal stays low, while BENIGN signal is asserted high. No recovery and interruption is required in this case too, as BENIGN signal is high.

As can be seen, SEM does not trigger error recovery for false positive scenarios. Also, since the data latched in R_1 are speculatively used by the succeeding stages, as soon as they are available, the error detection overhead is not incurred during normal system operation. This is also a low overhead solution, as it shuns the need for check pointing at regular time intervals. Thus, we enable systems to mitigate soft errors, using SEM cells, without any loss of performance, compared to a nonfault tolerant design.

2.5 Fault Tolerance Analysis

The SEM technique detects and recovers from all possible soft error scenarios involving both SEUs and SETs. This scheme is well suited for fast transient pulses. Since fast transients typically correspond to soft errors with high strike rate probabilities, SEM cells have near 100 percent transient fault mitigation capability. Our scheme offers protection for pulses of widths less than the phase shifts provided between the clock signals. Any noise signal, whose pulse width exceeds this limit, cannot be detected by our scheme.

3 Soft Error Mitigation in Aggressive Designs

Aggressive designs are based on the philosophy that it is possible to go beyond worst-case limits to achieve best performance by not avoiding, but detecting and correcting a modest number of timing errors. Adaptive reliable overclocking technique proposed for synchronous implementation of logic blocks acts like an asynchronous system design technique exploiting data-dependent variance in delay. In this section, we further investigate the solution presented in previous section for soft error mitigation, and explain how it can be modified for soft error mitigation in aggressive designs, which uses adaptive reliable overclocking technique for improving system performance. Such a technique enables systems to operate beyond safety margins, while preserving data integrity.

With a conventional voter design, to detect and correct n errors simultaneously, we need to have up to $2n+1$ data samples. In our case, we have $n = 2$, since we need to detect

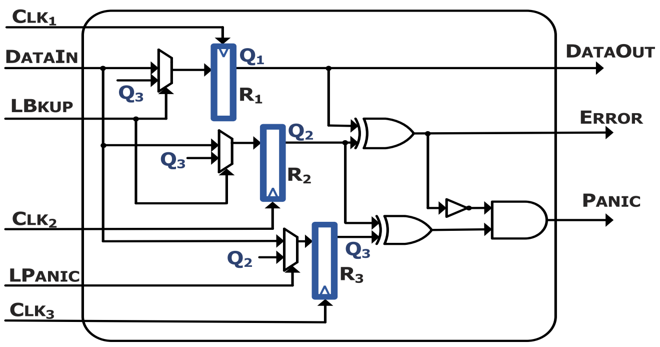

Fig. 3. STEM cell to mitigate soft and timing errors.

TABLE 2
Possible Error Scenarios in Using STEM Cell
(NE = No Error; SE = Soft Error; TE = Timing Error)

Case	R_1	R_2	R_3	ERROR	PANIC	RECOVERY
I	NE	NE	NE	0	0	No recovery required
II	SE	NE	NE	1	0	Load R_3 into R_1, R_2
III	NE	SE	NE	1	0	Load R_3 into R_1, R_2
IV	NE	NE	SE	0	1	Load R_2 into R_3
V	TE	NE	NE	1	0	Load R_3 into R_1, R_2
VI	TE	SE	NE	1	0	Load R_3 into R_1, R_2

and correct both soft and timing errors. For this analysis, we consider soft errors to be of only type SET. A traditional fault tolerance technique requires five different data values for guaranteeing both soft error and timing error detection and correction. The overhead incurred by this approach is very high as it increases the number of registers by four times, and requires five different clocks to sample data at five different times. Our goal is to develop a soft and timing error mitigation scheme that incurs minimal overhead. The proposed *Soft and Timing Error Mitigation* cell is similar to the SEM cell in area complexity. However, the error detection and recovery mechanism is significantly different to address the requirements of concurrent soft and timing error mitigation.

3.1 Error Detection

Fig. 3 shows a gate-level embodiment of a STEM cell, which acts as an online-fault monitor for soft and timing error mitigation. It also consists of three registers R_1, R_2, and R_3, clocked by clock signals CLK_1, CLK_2, and CLK_3, respectively. DATAIN represents data coming from the combinational logic and DATAOUT represents the latched data that are sent to the next stages of computation. ERROR and PANIC represent the control signals generated by the voting process that helps in soft error and timing error detection and recovery. LBKUP and LPANIC represent the control signals that are used to recover from error scenarios as presented in Table 2. Under no error scenarios, incoming data are sampled at three different time intervals T_1, T_2, and T_3, and are stored in registers R_1, R_2, and R_3, respectively.

The working of a STEM cell is as follows: once the data are latched in registers R_1 and R_2, they are compared with each other. This comparison operation completes the timing error detection process, since R_2 is timing safe [17], [18], [19]. But in the presence of soft errors, this comparison operation presents an ambiguous situation, as it is not possible to distinguish which one of these two registers is corrupted by an erroneous value. Also, value in R_2 is not to be trusted during the error recovery process.

If the comparison between R_1 and R_2 flags a mismatch, register R_3 is shielded from the incoming data value, and its content is used to recover the system state. This is done because any soft error that happens after comparing R_1 and R_2 has the potential to corrupt R_3 and push the system into an unrecoverable state. Only when there is no mismatch between registers R_1 and R_2, register R_3 is allowed to latch the data safely. However, we have not yet ascertained whether R_3 is free from soft error. Therefore, we perform another comparison operation to complete the error detection process. After register R_3 is updated, we compare it with register R_2, to detect any error happening in register R_3. If there is no mismatch, register R_3 is trusted for error recovery purposes. If they mismatch, then that represents a case where register R_3 is corrupted by a soft error. At this point, it is possible to say that data latched in registers R_1 and R_2 are uncorrupted. The system is stalled for one cycle for flushing out the erroneous value from R_3, and loading either R_1 or R_2 value into R_3.

3.2 Timing Constraints

As is the case with SEM cells, STEM cells also require strict timing constraints, to detect and correct soft and timing errors. STEM cells must satisfy (1), (2), and (3). Equation (4) is modified as shown in (5) for STEM cells. Equations (1) and (2) ensure that registers present in a STEM cell are not corrupted by the same SET. Equations (3) and (5) ensure that data latched in registers R_2 and R_3 are timing correct, i.e., free from timing errors. The timing relationships shown in Fig. 2 still hold, with the caveat that Φ_1 also includes the extent of overclocking that is possible every cycle.

$$T + \phi_1 \geq T_{PD}. \tag{5}$$

3.3 Error Recovery

Table 2 lists all possible error scenarios with corresponding recovery mechanisms. In the following discussion, we explain the various possible events that take place in the STEM cell, and the associated recovery mechanism that is used in case of an error. It employs either a single cycle or three cycle fast local recovery based on the values of ERROR and PANIC signals, shown in Fig. 3.

- CASE I. No error case. Both signals, ERROR and PANIC, stay low. System operation is not interrupted.
- CASES II, III, V, VI. This represents a case where one of the registers R_1 or R_2 is corrupted. In this case, $\mathrm{ERROR} = 1$ and $\mathrm{PANIC} = 0$. In this scenario, R_3 is not updated, and the system recovers by loading R_3 into R_1 and R_2 triggering recomputation. A three cycle global recovery process is initiated, which includes one cycle stall for loading data back into the registers R_1 and R_2, using LBKUP signal, and two cycles for recomputation. This two cycle recomputation is required, as the error might have occurred because of overclocking, and this error will repeat in

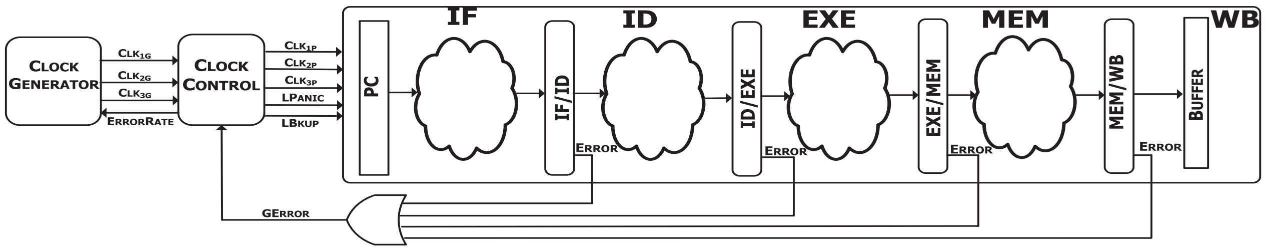

Fig. 4. Pipeline design with STEM cells.

R_1, if sufficient time is not given for recomputation. This prevents recurrent system failures.

- CASE IV. Only R_3 is corrupted. In this case, $\text{ERROR} = 0$ and $\text{PANIC} = 1$. No recomputation is required. However, it is necessary to flush the erroneous data from R_3, to facilitate error recovery in subsequent cycles. As data in only R_3 are corrupted, "golden" data present in R_2 are loaded into R_3. This requires a single cycle system stall, during which all STEM cells perform a local correction, using LPANIC signal.

3.4 Fault Tolerance Analysis

As is seen, the STEM technique detects and recovers from all possible soft and timing error scenarios, wherein the soft error is only of type SET. Also, the case where $\text{ERROR} = 1$ and $\text{PANIC} = 1$ never happens by design. Our technique leads to silent data corruption, if an SEU happens in R_3. However, since register R_3 is only used as a checkpointing register, a corrupted R_3 value may lead to failure, only if an error occurs in R_1 or R_2 in the next cycle. Consequently, the possibility of a system failure because of an SEU in R_3 is heavily mitigated. For Case VI, we expect that a TE or SE affects several STEM cells, and the possibility of all cells having a TE in R_1 and SE in R_2 is insignificant. Hence, we hope one of the STEM cells will have the error signal triggered, preventing R_3 of all STEM cells from being loaded. If $\text{ERROR} = 1$, then we do not look at PANIC signal. The fault coverage is similar to that of the SEM technique, except that in the case of false positives, we still need to take appropriate corrective action. In the case of the SEM scheme, this value will be overwritten, as R_3 is used only for error detection. However, the STEM technique allows reliable overclocking, achieving higher performance than those systems incorporated with SEM cells.

4 Pipeline Design

The basic step in using SEM or STEM cells in a pipeline is to replace all pipeline registers with either one of them. Input clocks are to be constrained in a way, so as to provide fault tolerance capabilities to the pipeline from soft error, as well as, timing error when STEM is the cell of choice. In this section, our discussion is based on the use of STEM cells in place of pipeline registers. Using SEM cells follows straightforward.

Fig. 4 illustrates how STEM cells are integrated into a processor pipeline. The figure depicts the data and control flow for a five-stage pipeline processor. To the last stage of the pipeline, which is writeback (WB), an extra write buffer is added. This is to ensure that data written to the register file or memory are always free from timing errors. Every pipeline stage register is replaced with STEM cells, except for the write buffer registers. All error signals from a pipeline stage are logically OR-ed to generate the stage error for that pipeline stage. Global error signal, GERROR, is generated from all pipeline stage error signals, by combining them using another "OR" function. Similarly, global LPANIC signal is generated from individual PANIC signal from all STEM cells. Timing errors may occur once the operating frequency exceeds the worst-case frequency estimate. As explained in the previous section, our data latching scheme of STEM cell guarantees sufficient time before latching values in registers R_2 and R_3. However, data latched in all three registers are susceptible to soft errors that are uniformly distributed in time and space. Here, we explain the pipeline operation for $\text{ERROR} = 1$ and $\text{PANIC} = 0$ (Cases II, III, V, VI), as this is the most complicated case. Once an error is detected in any one of the pipeline stages, the global error signal is asserted, and in every stage of the pipeline, registers R_3 of the STEM cells are not updated with the incoming data. In the next clock cycle, the load backup signal, LBKUP, is asserted, and in each STEM cell, the content of register R_3 is loaded into corresponding R_1 and R_2 registers. After this, the clock to the pipeline is stalled for two cycles, completing the error recovery process.

4.1 Clock Control

Clock control monitors the error rate of the pipeline and communicates this error rate information with the clock generator for frequency tuning. Clock generator is connected to the pipeline in a feedback loop. It checks the pipeline error rate with a set target rate, which is programmable. Process of generating new frequency takes up to $10\ \text{us}$, depending upon the speed at which the phase locked loop (PLL) generates the new stable clock signal. Depending on the clock control scheme and error rate sampling scheme chosen [18], [19], clock frequency is adjusted to allow the pipeline to operate below a set target error rate. Once the new clock is generated, the main clock signal, CLK, is switched to that frequency and other clock signals CLK_{1G}, CLK_{2G}, and CLK_{3G} are generated by providing the necessary phase shifts to CLK.

4.2 Dynamic Frequency Scaling

In the following discussion, we derive the limits of frequency scaling within which a system integrated with STEM cells operates reliably. Pipeline starts execution with

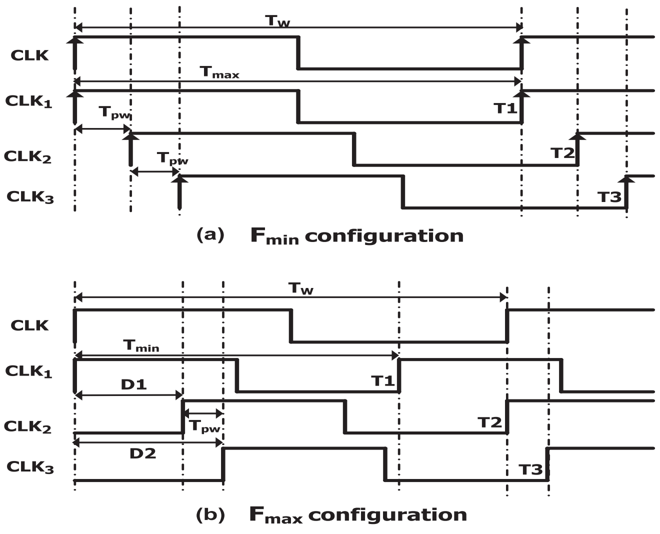

Fig. 5. Dynamic frequency scaling.

a minimal phase shift required between the clocks, and the clock frequency is gradually increased, while satisfying the error rate constraint, as shown in Fig. 5a. To support reliable dynamic overclocking, certain governing conditions need to be met at all times, during pipeline operation. Let us assume that the pipeline operates reliably between the clock frequencies, F_{MIN} and F_{MAX}, governed by time periods, T_{MAX} and T_{MIN}, respectively. T_{MAX} is estimated by the worst-case design settings, and is equal to worst-case clock period, T_{WC}. The following clocking constraints decide T_{MIN}. Under overclocking conditions, the following constraints must be satisfied for proper error detection and recovery.

Let D_1 represent the phase shift that needs to be provided for CLK$_2$, with respect to CLK$_1$, for soft and timing error mitigation, when the system is clocked with clock period T_{MIN}, as shown in Fig. 5b. Let D_2 represent the phase shift that needs to be provided for CLK$_3$, with respect to CLK$_1$, for proper error recovery, when the system is clocked with clock period T_{MIN}. Value of T_{MIN}, satisfying (6), corresponds to the maximum frequency at which a system can possibly recover, after a timing error occurs.

$$T_{MIN} + D_1 \geq T_{PD}, \tag{6}$$

$$D_2 - D_1 \geq T_{PW}, \tag{7}$$

$$T_{CD} \geq D_2. \tag{8}$$

4.3 Fixed Frequency Operation

For operating without any runtime optimizations, the frequency of the main clock can be fixed at the desirable operating frequency satisfying the conditions described in Section 3. This operating frequency can also be set above the worst-case estimate. Under these conditions, systems using STEM cells will see varying error rates based on the executing application. This configuration offers performance better than the system integrated with SEM cells and cannot achieve the best performance as the dynamic frequency adjustment is not allowed. If the frequency is fixed at the worst-case operating frequency, systems integrated with SEM and STEM cells are expected to offer quite high levels of reliability having identical performance levels, when compared to an unprotected system, as there is no overhead incorporated by error detection and correction circuitry on system operating frequency.

4.4 Pipeline Error Recovery

In this section, we present the error recovery scheme in detail for a pipeline using STEM cells. Various events involved in the recovery process are illustrated with the help of a timing diagram. Fig. 6 shows how our global error recovery scheme rescues a system when an error happens in the pipeline. The figure also depicts the timing relationship between various control signals required for the recovery process. The following description explains the events that take place during the recovery process for the cases: $\text{ERROR} = 1$, $\text{PANIC} = 0$ and $\text{ERROR} = 0$, $\text{PANIC} = 1$.

Fig. 6 shows a set of clock signals, CLK$_{1G}$, CLK$_{2G}$, and CLK$_{3G}$, that are generated from the main clock signal, CLK, using an LCM. Next, it shows a set of clock signals, CLK$_{1P}$, CLK$_{2P}$, and CLK$_{3P}$, that are routed to the pipeline. These clock signals, which are gated versions of CLK$_{1G}$, CLK$_{2G}$, and CLK$_{3G}$, respectively, are stalled in a manner that enables the pipeline to recover from different error scenarios. Signal ERROR$_N$ indicates an error happening in the pipeline stage N. ERROR signals from all the pipeline stages are OR-ed together to generate the global error signal, GERROR, which is latched in the clock control unit. Similarly, to initiate global recovery when $\text{ERROR} = 0$, $\text{PANIC} = 1$, i.e., for panic cases, all stage PANIC signals are used.

For case, $\text{ERROR} = 1$, $\text{PANIC} = 0$, once an error is detected, the very next clock edge of clock signal CLK$_{3G}$ is gated and in the next cycle, LBKUP signal is asserted high for one clock cycle. In the same clock cycle, using CLK$_{1P}$ and CLK$_{2P}$, recovery data from register R_3 are loaded back into registers R_1 and R_2. During the next cycle, all clock signals, CLK$_{1G}$, CLK$_{2G}$, and CLK$_{3G}$, are clock gated to give the pipeline sufficient time for recomputation. Clock gating is achieved through control signals CLKSTALL$_{12}$ and CLKSTALL$_3$, which are generated by the clock control unit.

For case, $\text{ERROR} = 0$, $\text{PANIC} = 1$, once an error is detected, the very next clock edge of clock signals, CLK$_{1G}$ and CLK$_{2G}$, is gated and in the same cycle, LPANIC signal is asserted high for one clock cycle. Also, recovery data present in register R_2 or R_1 are loaded back into register R_3 during this cycle using CLK$_{3P}$. Clock gating of clock signals, CLK$_{1G}$ and CLK$_{2G}$ is achieved through control signal CLKSTALL$_{12}$, which is generated by the clock control unit.

To illustrate our error recovery mechanism, error occurrences corresponding to cases, $\text{ERROR} = 1$, $\text{PANIC} = 0$ and $\text{ERROR} = 0$, $\text{PANIC} = 1$, are highlighted in Cycles 3 and 8, respectively. First error occurs during the execution of INST one of pipeline stage N. This event triggers the error recovery mechanism that spans from Cycle 4 to Cycle 6. During Cycle 4, data are loaded into registers R_1 and R_2 from the corresponding stage golden register R_3. Pipeline is allowed to perform the computation during Cycles 5 and 6. Results are again checked at the end of Cycle 6. Since no error is detected in this cycle, normal pipeline operation resumes.

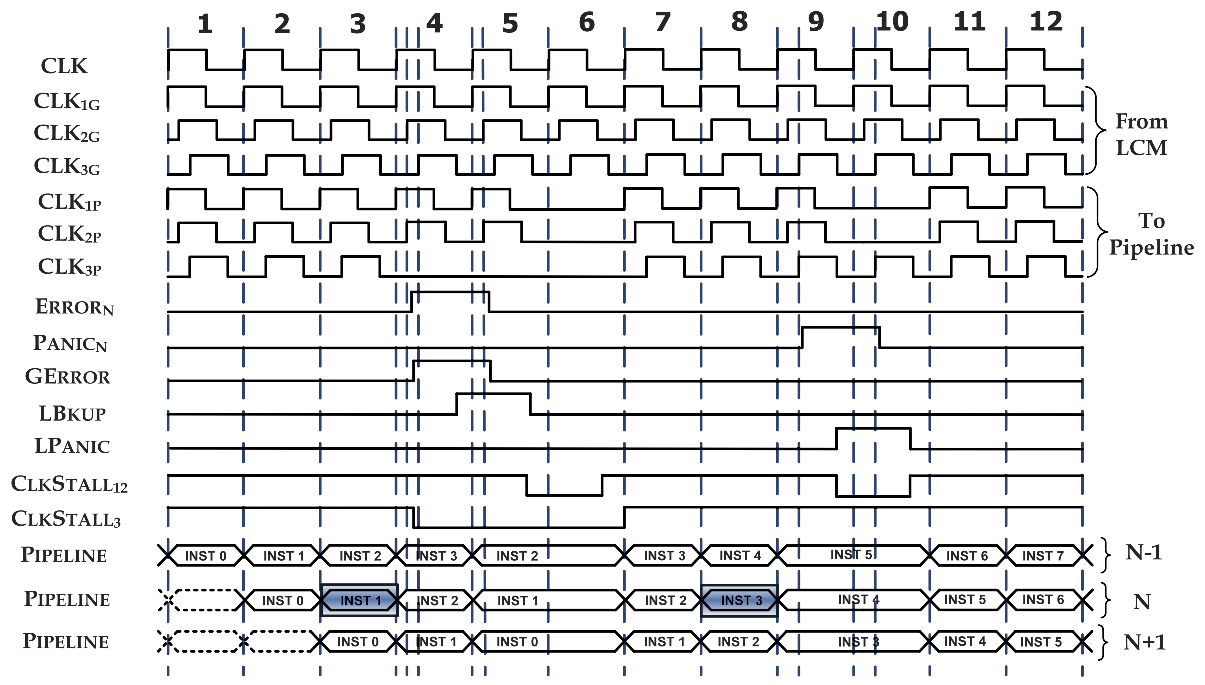

Fig. 6. Timing diagram illustrating STEM error recovery process.

From the waveforms, we can see that on error detection, the entire pipeline goes back by one instruction. Second error occurs during the execution of INST 3 of pipeline stage N. This event triggers the error recovery mechanism that spans from Cycle 9 to Cycle 10. During Cycle 9, data are loaded into register R_3 from the corresponding stage golden registers, R_1 or R_2. At the end of Cycle 9, no new results are stored in all stage registers R_1 and R_2. New results are latched and checked at the end of Cycle 10. Since no error is detected in this cycle, normal pipeline operation resumes. In this case, the pipeline does not roll back and just the corresponding stage R_3 register is updated. As we can see from Fig. 6, for 12 clock cycles, pipeline computes only eight instructions as four clock cycles are accounted for the error recovery process.

4.5 Performance Analysis

A key factor that limits frequency scaling is error rate. As frequency is scaled higher, the number of input combinations that result in delays greater than the new clock period also increases. The impact of error rate on frequency scaling is analyzed as follows:

Let t_{wc} denote the worst-case clock period. Let t_{ov} denote the clock period after overclocking the circuit. Let n be the number of cycles needed to recover from an error. Let us assume that a particular application takes N clock cycles to execute, under normal conditions. Let t_{diff} be the time difference between the original clock period and the new clock period. Then, the total execution time is reduced by $t_{diff} \times N$, if there is no error. Let us assume that the application runs at the overclocked frequency of period t_{ov} with an error rate of k percent. To achieve any performance improvement at this frequency, (9) must be satisfied. It states that even after accounting for error recovery penalty, execution time required is still less than that required for worst-case frequency operation.

$$N \times t_{ov} + n \times N \times k \times t_{ov} < N \times t_{wc}, \tag{9}$$

$$k < \frac{(t_{wc} - t_{ov})}{n \times t_{ov}}. \tag{10}$$

For the STEM technique, an error can happen in five different scenarios, as mentioned in Table 2, and also the error recovery penalty paid is not the same for all the cases. If we assume that all these error scenarios are equally likely, then the average error penalty in cycles is $n = \frac{4\times3+1\times1}{5} = 2.6$. According to (10), for a frequency increase of 15 percent, the error rate must not be higher than 5.76 percent, for the STEM technique to yield no performance improvement. For error rates less than *one percent*, a frequency increase of *2.6 percent* is enough for the STEM scheme to have a performance improvement over nonfault tolerant designs.

4.6 Impact of Process Variation

Process variation can alter gate delays of a combinational circuit and hence can alter the distribution of path delays. Both our techniques, SEM and STEM, sample data at three different times, namely, T_1, T_2, and T_3, after the computation is completed. Paths that can affect this sampling window are short paths that are present in the circuit. These paths are already padded with buffers to allow sufficient time for data sampling. In presence of process variation, in particular, only sampling interval T_3 can be affected. Since, these paths are present locally, interdie variation has no impact on these paths. To mitigate intradie variation, conservative padding of buffers needs to be done on the short paths to make sure that same data are sampled. This will only result in slight increase of the circuit area.

4.7 Overheads

One of the main overheads incurred by our schemes is fixing the circuit contamination delay to a required value. Increasing this delay involves rapid increase in silicon area,

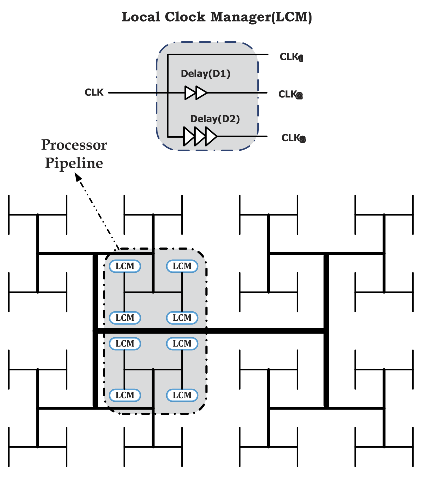

Fig. 7. Local clock phase management with single clock routing.

as buffers need to be inserted in the short circuit delay paths. This problem has to be addressed from different design perspectives that include developing new synthesis algorithms and delay buffer design with minimal area consumption. Both SEM and STEM cells require metastability mitigation circuits, as flip-flops may enter a metastable state when overclocked, or when a soft error reaches the registers during the latching window. We envisage the incorporation of a metastability detection circuit, similar to the one developed in [17].

5 Local Clock Management

Reliable dynamic overclocking technique has been proposed earlier, in [18] and [19], to improve system performance by tuning the clock frequency beyond the conservative worst-case clock period. It requires a dynamic phase shift (DPS) between the clock signals to support aggressive dynamic clock tuning. This implies the system has to be stalled for a few cycles whenever the clock frequency is changed. This is done for two purposes: 1) the time needed to switch to a new frequency and 2) to appropriately phase shift for the respective clock signals, as mandated by the timing constraints. At higher frequencies, controlling the phase shift precisely is a challenge and often the phase shift step size can restrict the possible system operating frequencies. Also, SEM and STEM cells would require three clock distribution networks, thus increasing clock routing complexity and resources needed for it. The increase in clock resources would in turn increase system energy consumption. To avoid dynamic phase shift between the clock signals, we incorporate a constant phase shift (CPS) between the clocks that are configured to run between frequencies corresponding to the time periods, T_{MAX} and T_{MIN}.

Let us consider a case where $T_{MAX} = 10$ ns, $T_{MIN} = 6$ ns, and $T_{CD} = 4$ ns. Considering a dynamic phase shift between the clock signals, when we scale the system clock period down to 8 ns, we need to provide a phase shift of 2 ns. Similarly, a phase shift of 3 ns is required for a 7 ns clock period. Since the circuit contamination delay is increased to 4 ns to aggressively clock the system, computed data will remain stable for $(T + 4)$ ns, where T is the current operating frequency of the system. Instead of requiring a dynamic phase shift along with frequency scaling, we provide a constant phase shift of at most T_{CD} at all times.

TABLE 3
Number of Buffers versus Delay

Delay(ns)	Buffers	Delay(ns)	Buffers
1.0	6	1.5	8
2	11	2.5	14

With a CPS scheme, aggressive systems can be built with a single CDN, thus tremendously reducing the amount of clock resources required. Using a constant phase shift that is valid for all frequencies between F_{MIN} to F_{MAX}, allows phase management to be decoupled from frequency switching. This completely eliminates the constraint imposed by the phase shift step size in DPS, allowing CPS to support more operating configurations than DPS. Local clock managers handle the phase shift management in CPS. Fig. 7 shows a conceptual implementation of an LCM that can distribute CLK_1, CLK_2, and CLK_3. Employing CPS, delay values D_1 and D_2, are set to constant values to satisfy the timing constraint explained earlier in Section 4. Fig. 7 also depicts how these LCMs could be integrated into an H-tree clock distribution network.

5.1 Case Study: Local Clock Management Using Buffers

For generating clock signals required by SEM and STEM schemes locally, we present a possible implementation using buffers. We perform this study using 45 nm spice models distributed by Nangate Technologies [25]. Post layout spice models containing parasitic information are used. Area overhead, incurred for generating constant phase shift clocks, is analyzed by applying a load of 128 STEM cells. From this study, we observe that, even for a 2.5 ns phase shift, only 14 clock buffers are needed. This overhead is much lower than having a second and third clock tree networks. Study results are summarized in Table 3. Fig. 8 shows the transient response of clock signals generated with an LCM designed with delay buffers. A load of 32 STEM cells is applied at room temperature (25°C). From this, we notice that, sufficient phase shift between the clock signals is provided across all the cycles. Also, the rise time and the fall time of CLK_2 and CLK_3 signals are around 569 and 222 ps, respectively, as

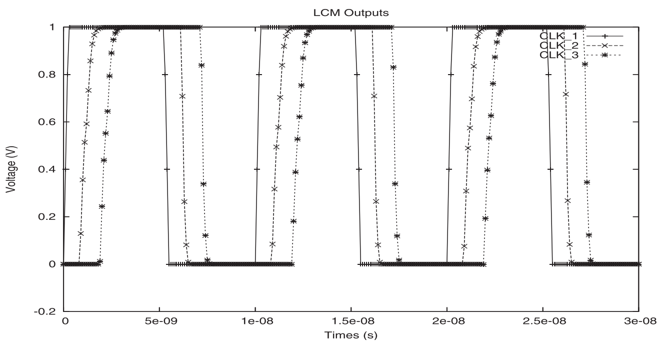

Fig. 8. Transient response of LCM outputs.

TABLE 4
Temperature Effects on LCM Outputs

OUTPUT	70°C		75°C		80°C	
	risetime	falltime	risetime	falltime	risetime	falltime
CLK_1	$200ps$	$200ps$	$200ps$	$200ps$	$200ps$	$200ps$
CLK_2	$875ps$	$285ps$	$913ps$	$292ps$	$956ps$	$300ps$
CLK_3	$861ps$	$278ps$	$898ps$	$286ps$	$935ps$	$293ps$

illustrated in Fig. 8. Table 4 shows the rise time and the fall time for LCM outputs at different temperatures. From these values, we notice that, all the clock signals track each other very well in terms of fall time even at very high temperatures. They can also be made to have similar characteristics in terms of rise time at the expense of designing sophisticated buffers.

Impact of process variations. In this section, we presented a simple approach, using delay buffers for managing phase relationship between clock signals that are required for SEM and STEM cells. It is presented to give a conceptual picture of our CPS scheme. In presence of process variations, delay of these buffer cells can vary and hence the phase difference between the clocks can also be altered. In a CPS scheme, it is important to realize the difference in phase difference requirements between CLK_1 and CLK_2, and CLK_2 and CLK_3. In CPS, the phase difference between CLK_1 and CLK_2 at max needs to be D1 (where D1 $\gg T_{PW}$) and for CLK_2 and CLK_3, it has to be at least T_{PW}. Because of process variations, any variation in D1 will alter only the frequency scaling limits and not the soft error mitigation capabilities of our techniques. For CLK_2 and CLK_3 phase relationship, post fabrication delay tuning approaches [26] or using tunable/programmable delay elements [27] can be used to mitigate the effect of process variation (intradie only) on our techniques.

6 EXPERIMENTS AND RESULTS

In this section, we present our results based on the experiments conducted on a two-stage arithmetic pipeline and a five-stage DLX in-order pipeline processor, wherein pipeline registers are augmented with our fault detection and correction circuitry. It should be pointed out that soft error rate depends on electrical masking and logical masking which inherently mask transient faults. Across process technology nodes, logical masking remains the same but the electrical masking differs. This is because electrical properties of gates differ between technology nodes. Electrical masking could diminish significantly as feature size decreases. In order to model electrical masking that happens in nanometer technologies, we have implemented system in 45 nm technology. Also, fault mitigation capabilities exhibited for same T/T_{PW} ratio but for different clock frequencies and pulse widths are expected to be same. For example, fault mitigation capabilities exhibited for 100 ps pulse at 1 GHz are expected to be the same as those offered for 1,000 ps pulse at 100 MHz. As presented in Section 1, soft error pulses have width in the range of 500-900 ps in the 90 nm process. In our experiments, we have considered this *extreme-case* pulse width for 45 nm process to demonstrate transient fault detection and correction capabilities of our techniques. Without lose of generality,

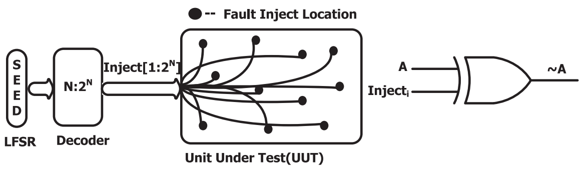

Fig. 9. Fault injector framework.

our results can be improved by relaxing the soft error pulse width (T_{PW}) constraint and insights derived from our experiments will still hold.

6.1 Experimental Methodology

To estimate the performance gains and fault tolerant capabilities offered by SEM and STEM techniques, simulations are carried out on a two-stage arithmetic pipeline. This circuit performs a 64-bit addition in the first stage and a 32-bit multiplication in the second stage. Adder output is fed to the multiplier as multiplicand and multiplier. RTL level models are developed for both the circuits, and are synthesized using the 45 nm OSU standard cell library [28]. Timing-annotated gate-level simulations are then carried out by extracting timing information in standard delay format (SDF), and back annotating them on the design.

Fig. 9 illustrates our fault injection methodology. The working of our fault injector is as follows: a total of 2^N (N being seven in our experiments) fault injection test nodes that are spread uniformly across the area of the logic circuit are selected. To make sure that our injected fault has indeed produced a SET, we modified the circuit netlist to insert XOR gates at all selected nodes, as shown in Fig. 9. If a location i is chosen for fault injection, $Inject_i$ is made high to invert the signal A driven by the fault injection node i. Out of 2^N locations, one location is chosen randomly for fault injection at a time, by using the output of an N-bit random number generator. For our experiments, we used a linear feedback shift register (LFSR) for generating the N-bit random number. Final fault location is then selected with the help of an N:2^N decoder.

6.2 Results for Arithmetic Pipeline

For the arithmetic pipeline, from static timing analysis reports, we estimated the value of T_{MAX} to be 9 ns. For aggressively clocking the design, we increased the contamination delay to 3 ns. Combinational area of the arithmetic pipeline is increased by 38 percent, for fixing the contamination delay to 3 ns. This area overhead is in addition to the area consumed by SEM/STEM cells. Pulses of varying widths ranging from 500 to 900 ps are injected in the unit under test (UUT). In each cycle, results are checked for correctness after the computation is over to ensure that the recovery mechanism works. Whenever recovery is triggered, we logged the occurrence of an error.

For evaluating STEM technique, we performed our experiments for a set error rate target of one percent over 10,000 cycles. During runtime, the number of errors that happened during a sampling interval is communicated to the clock controlling unit at the end of each interval. The clock controlling unit makes a decision based on the error rate, during the previous sampling interval, and the set

TABLE 5
Fault Injection Results for the Arithmetic Pipeline

	STEM (MAXOC)			STEM (DYNOC)			STEM (NOOC)		SEM		TMR	
	TE	Transient Faults		TE	Transient Faults		Transient Faults		Transient Faults		Transient Faults	
		Injected	Detected		Injected	Detected	Injected	Detected	Injected	Detected	Injected	Detected
RUN1	14	2031	432	14	2033	421	2030	325	2031	334	2026	256
RUN2	14	2031	450	12	2033	414	2025	315	2028	323	2026	268
RUN3	14	2031	449	15	2032	424	2025	307	2030	311	2034	273

target error rate. We considered a linear control scheme for switching clock frequency between the worst-case clock frequency, F_{MIN} and the overclocked frequency, F_{MAX}. For our design, T_{MIN} is set at 7 ns. This range is divided into 32 steps, and if the error rate is less than one percent, clock frequency is increased by one step size; otherwise, it is decreased. Our fault injection results for the arithmetic pipeline are presented in Table 5. We initialized the LFSR with different seeds, and the fault injection results are presented for three different runs.

We configured the arithmetic pipeline designed with STEM cells to operate in three different modes. They are no overclocking (NOOC), wherein $T_{MAX} = T_{MIN} = 9$ ns, maximum overclocking (MAXOC), wherein $T_{MAX} = T_{MIN} = 7$ ns, and dynamic overclocking, wherein $T_{MAX} = 9$ ns and $T_{MIN} = 7$ ns. For DYNOC mode, we started with a low frequency setting. For TMR system, worst-case frequency, T_{MAX}, is set at 11 ns. We evaluate SEM scheme at a constant clock period of 9 ns. Performance improvements offered by both SEM scheme and different modes of STEM are shown in Fig. 10. From this, we can see that DYNOC mode offers 49 percent improvement over TMR, while MAXOC mode offers 55 percent improvement. Performance of NOOC mode is comparable to that of SEM and SEM offers 23 percent performance improvement over TMR. From Table 5, we can see that fault masking rate is high in TMR design when compared with SEM and STEM designs. This is because, its operating frequency includes the phase shifts of the clocks and voter delay. Hence, TMR operates with a longer clock period compared to SEM and STEM, resulting in more SET pulses attenuating before reaching the latching window.

Fig. 11 shows the transient response of our arithmetic pipeline configured with STEM cells in DYNOC mode. It shows how the frequency of the system changes dynamically for high to low frequency switching mode and low to high frequency switching mode. From the figure, we see that the system tunes its optimal clock period to 7.2 ns. For high to low frequency switching mode, the optimal value is reached after four sampling intervals; for low to high mode, optimal value is reached after 22 such intervals. High to low frequency mode ran more operations than the other mode, and the performance difference was about three percent. Even though contamination of the system is increased to 3 ns, the available dynamic frequency range is restricted to 2 ns due to pulse width considerations.

6.3 Results for DLX Processor

We also simulated three different microbenchmarks to evaluate the performance improvement and fault coverage of both SEM and STEM (DYNOC mode) schemes on a five-stage in-order pipelined processor. This processor, implemented in 45 nm technology, is based on the DLX instruction set architecture. First application, RandGen, calculates a simple random number generation to give a number between 0 and 255. The MatrixMult application multiplies two 50×50 integer matrices and the BubbleSort program implements bubble sort algorithm on 5,000 half-word variables. Here, we followed the same fault injection strategy and clock control used for two-stage arithmetic pipeline. For each benchmark, processor state is checked to verify the correctness of the computed results after simulation. From timing reports, the worst-case clock period, T_{MAX}, is estimated as 6 ns. Contamination delay is increased by 3 ns and the system operates at an optimal clock period of 4 ns. At chip level, area overhead incurred is less than 15 percent for the processor because significant area consumption of the system comes from the memory system. The results for the three different benchmarks are presented in Fig. 12, showing relative execution times for conventional TMR, SEM, and STEM schemes. From this, we found that SEM offers 26.58 percent performance improvement over TMR and STEM offers 27.42 percent over SEM.

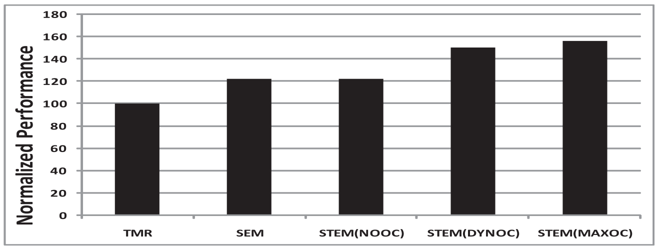

Fig. 10. Normalized arithmetic pipeline execution time.

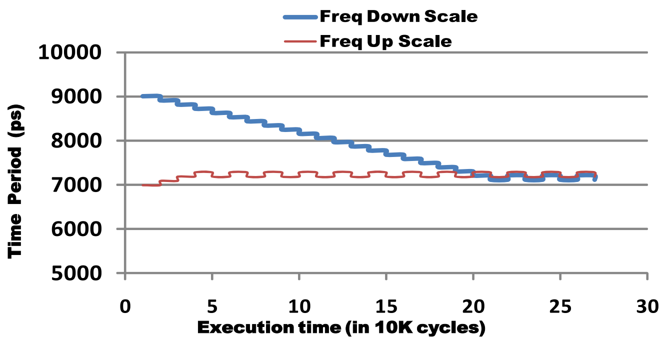

Fig. 11. Dynamic frequency scaling.

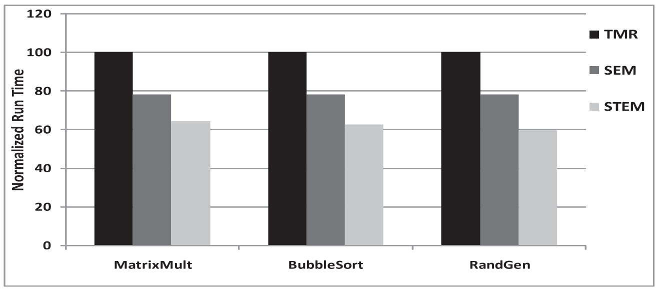

Fig. 12. DLX execution rime for various benchmarks.

7 Related Work

A multitude of fault tolerant architectures for tolerating soft errors has been developed in the past by the research community. Many schemes incur performance overhead even during error-free operation and do not support timing speculation. LEON-FT processor [3] uses TMR approach and triplicates every flip-flop in the processor and incurs a 100 percent area overhead. The REESE architecture [29] takes advantage of spare elements in a superscalar processor to perform redundant execution. DIVA [30] uses spatial redundancy by providing a separate, slower pipeline processor alongside the fast processor. Reunion [31] exchanges control and data flow information between the cores to speed up execution, while leveraging the redundancy to provide partial fault coverage. These approaches trade performance and power for achieving soft error fault tolerant capabilities and none of these approaches support timing speculation. Our technique, SEM, is the first technique in this class, that effectively exploits timing speculation and enhances performance without any logic duplication (LD).

Multiple data latching schemes. Multiple clocking of data has been widely used to combat the issue of soft errors in combinational logic [2], [10], [20], [21], [22], [23]. These approaches involve latching data at different times or latching data and its delayed version using a single clock. Authors in [22] optimize a temporal TMR framework that is presented in [10]. Even with this approach, the clock frequency at which the circuit can be clocked must include the delays of combinational logic, double the pulse width under consideration and also the delay incurred for the majority voting. Moreover, this technique doesn't detect false errors.

More recently, BISER, a DMR framework, using C-elements to tolerate soft errors has been proposed [21]. It offers two techniques—1) using logic duplication (DMR in space) and 2) time shifted data (DMR in time). Also, using C-elements, authors in [23] propose soft error hardened flip-flops to tolerate wide error pulses or hard errors and apply DMR in both time and space domains. As we explained, SEM requires only two comparison operations for error detection, as opposed to full voting of multiple samples. With our distributed and temporal voting, SEM completely eliminates the error recovery penalty paid for false errors. This is a big contribution when compared to previously proposed techniques, which use multiple data latching, by employing either DMR in space/time or temporal TMR. Moreover, the fault detection overhead is taken off the critical path, which enables systems to have performance levels comparable to a nonfault tolerant system in error-free operation. Previously proposed schemes, using multiple data latching, employ either DMR or TMR [10], [21], [22], [23] and pay penalty for false errors. SEM effectively combines timing speculation with temporal TMR framework and completely removes the overhead paid for false errors. Also, when an error is detected, unlike previous DMR solutions, we need not perform recomputation to recover the system state. This is a salient feature of our approach, which has *in-situ* fast error recovery. Hence, SEM does not require any checkpointing, thereby saving the time and space required to store the system state. Also, it is estimated that static power is comparable to dynamic power with nano-sized transistors. Hence, providing fault tolerance for energy constrained systems like embedded systems, with logic replication is not a viable option. Systems designed with SEM cells improve reliability without logic replication.

As the performance margin between designing circuits to meet worst-case constraints versus typically case constraints has become increasing significant, a new area of research emerged that aimed to use fault tolerance to allow synchronous circuits to perform reliably at the highest possible clock frequencies. A good summary of the principles of this new area is presented in [15]. The methods reviewed in [15] are good initial attempts at pushing circuits close to safe limits for increasing performance; however, most approaches are not feasible in practice. For example, in [32], a method for increasing clock frequency is proposed by adding redundancy and clocking each stage of a pipeline at a phase shift of the original clock. However, the overheads associated with the redundancy and clocking resources of this approach were not addressed for deployment into real-world designs. A more recent approach is presented in [33], where the clock frequency of a simple in-order processor was tuned dynamically. An extra delay chain with a worst-case propagation delay greater than the critical path of the processor was used to guide frequency tuning. The delay chain was monitored constantly and detection of errors was used to adjust the processor's frequency to avoid errors within the processor as operating conditions changed. A benefit of this approach is that it avoided errors within the processor, thus eliminating the performance penalties often associated with error recovery circuitry. However, this approach can adapt to changes in operating conditions, but can't exploit data-dependent circuit delay. This attribute of circuit delay has been exploited in SPRIT^{3}E to further extend the operation boundaries often set by worst-case estimates [18], [19].

Razor [16], [17] and SPRIT^{3}E [18], [19] architectures employ timing error tolerance techniques to operate beyond worst-case limits. Both these architectures use temporal fault tolerance by replicating critical pipeline registers to improve performance beyond worst-case limits. While Razor focuses on achieving lower energy consumption by reducing supply voltage in each pipeline stage, SPRIT^{3}E improves performance of a superscalar processor by reliably overclocking the pipeline. Other closely related work is Paceline [34], which employs leader-checker configuration in a chip multiprocessor system and tolerates

TABLE 6
Comparison with Other Schemes in Terms of Logic Duplication, Soft Error Protection (SEP), Aggressive Clocking (AC) and Energy Savings (ES)

Design	LD	SEP	AC	ES
Razor	×	×	×	√
SPRIT^{3}E	×	×	√	×
Paceline	√	√	×	×
SEM	×	√	×	×
STEM	×	√	√	×

both timing and soft errors. Systems designed with STEM cells look to improve the reliability and performance by enabling reliable overclocking without logic duplication. Table 6 summarizes how our schemes differ from previously proposed techniques.

8 Conclusions

In this work, we have developed two efficient soft error mitigation schemes, SEM and STEM, considering the approach of multiple clocking of data for tolerating soft errors in combinational logic. Both these techniques remove the error detection overhead from the circuit critical path. These specialized register cells provide near 100 percent fault tolerance against transient faults. Our schemes tolerate fast transient noise pulses, which is the principal characteristic of SETs. Both our schemes have no significant performance overhead during error-free operation. SEM cells are capable of ignoring false positives and recover from errors using *in-situ* fast recovery avoiding recomputation. STEM cells have soft error mitigation characteristics similar to SEM. STEM cells allow reliable dynamic overclocking and are capable of tolerating timing errors as well. We also propose a scheme to manage phase shifts between clocks locally with constant delay values. Such an approach increases the possible frequency settings for aggressively clocked designs and also minimizes the clock routing resources.

Acknowledgments

The research reported in this paper is funded in part by Information Infrastructure Institute and the Jerry R. Junkins Endowment at Iowa State University. The authors would like to thank the anonymous reviewers for their comments that helped in improving the quality of our manuscript.

References

[1] M. Alam, "Reliability- and Process-Variation Aware Design of Integrated Circuits," *Microelectronics Reliability,* vol. 48, nos. 8/9, pp. 1114-1122, 2008.

[2] L. Anghel and M. Nicolaidis, "Cost Reduction and Evaluation of a Temporary Faults Detecting Technique," *Proc. Design, Automation and Test in Europe Conf. and Exhibition,* pp. 591-598, 2000.

[3] J. Gaisler, "A Portable and Fault-Tolerant Microprocessor Based on the SPARC v8 Architecture," *Proc. Int'l Conf. Dependable Systems and Networks (DSN '02),* pp. 409-415, 2002.

[4] A. Meixner, M.E. Bauer, and D. Sorin, "Argus: Low-Cost, Comprehensive Error Detection in Simple Cores," *Proc. 40th Ann. IEEE/ACM Int'l Symp. Microarchitecture,* pp. 210-222, 2007.

[5] C.L. Chen, "Symbol Error-Correcting Codes for Computer Memory Systems," *IEEE Trans. Computers,* vol. 41, no. 2, pp. 252-256, Feb. 1992.

[6] "NASA News Report," http://science.nasa.gov/headlines/y2009/29sep_cosmicrays.htm, 2011.

[7] E. Normand, "Single Event Upset at Ground Level," *IEEE Trans. Nuclear Science,* vol. 43, no. 6, pp. 2742-2750, Dec. 1996.

[8] 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," *Proc. Int'l Conf. Dependable Systems and Networks,* pp. 389-398, 2002.

[9] B. Narasimham et al., "Characterization of Digital Single Event Transient Pulse-Widths in 130-nm and 90-nm Cmos Technologies," *IEEE Trans. Nuclear Science,* vol. 54, no. 6, pp. 2506-2511, Dec. 2007.

[10] D.G. Mavis and P.H. Eaton, "Soft Error Rate Mitigation Techniques for Modern Microcircuits," *Proc. 40th Ann. Reliability Physics Symp.,* pp. 216-225, 2002.

[11] S. Borkar et al., "Parameter Variations and Impact on Circuits and Microarchitecture," *DAC '03: Proc. 40th Design Automation Conf.,* pp. 338-342, 2003.

[12] K.A. Bowman et al., "Impact of Die-to-Die and Within-Die Parameter Variations on the Throughput Distribution of Multi-Core Processors," *Proc. Int'l Symp. Low Power Electronics and Design,* pp. 50-55, 2007.

[13] B. Colwell, "The Zen of Overclocking," *Computer,* vol. 37, no. 3, pp. 9-12, Mar. 2004.

[14] Channel Wire, "Amd's 45nm Phenom ii Chips Overclocked to 6+ghz," http://www.crn.com/hardware/212101254, 2008.

[15] T. Austin, V. Bertacco, D. Blaauw, and T. Mudge, "Opportunities and Challenges for Better than Worst-Case Design," *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC),* vol. 1, pp. 2-7, Jan. 2005.

[16] 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 J. Solid-State Circuits,* vol. 41, no. 4, pp. 792-804, Apr. 2006.

[17] D. Ernst et al., "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," *Proc. 36th Ann. IEEE/ACM Int'l Symp. Microarchitecture,* 2003.

[18] M. Bezdek and A. Somani, "Utilizing Timing Error Detection and Recovery to Dynamically Improve Superscalar Processor Performance," MS thesis, Iowa State Univ., 2006.

[19] V. Subramanian, M. Bezdek, N.D. Avirneni, and A. Somani, "Superscalar Processor Performance Enhancement through Reliable Dynamic Clock Frequency Tuning," *Proc. Ann. IEEE/IFIP Int'l Conf. Dependable Systems and Networks (DSN),* pp. 196-205, June 2007.

[20] M. Nicolaidis, "Design for Soft Error Mitigation," *IEEE Trans. Device and Materials Reliability,* vol. 5, no. 3, pp. 405-418, Sept. 2005.

[21] S. Mitra, M. Zhang, N. Seifert, T. Mak, and K. Kim, "Soft Error Resilient System Design through Error Correction," *VLSI-SoC: Proc. Int'l Conf. Very Large Scale Integration of System-on-Chip,* pp. 332-337, Oct. 2006.

[22] M. Chen and A. Orailoglu, "Improving Circuit Robustness with Cost-Effective Soft-Error-Tolerant Sequential Elements," *Proc. 16th Asian Test Symp. (ATS '07),* pp. 307-312, 2007.

[23] S. Ruan, K. Namba, and H. Ito, "Soft Error Hardened FF Capable of Detecting Wide Error Pulse," *Proc. IEEE Int'l Symp. Defect and Fault Tolerance of VLSI Systems,* pp. 272-280, 2008.

[24] N.D. Avirneni, V. Subramanian, and A. Somani, "Low Overhead Soft Error Mitigation Techniques for High-Performance and Aggressive Systems," *Proc. IEEE/IFIP Int'l Conf. Dependable Systems and Networks (DSN '09),* vol. 29, pp. 185-194, July 2009.

[25] Nangate, http://www.nangate.com, 2011.

[26] S. Narendra, A. Keshavarzi, B.A. Bloechel, S. Borkar, and V. De, "Forward Body Bias for Microprocessors in 130-nm Technology Generation and beyond," *IEEE J. Solid-State Circuits,* vol. 38, no. 5, pp. 696-701, May 2003.

[27] S.B. Kobenge and H. Yang, "Power Optimized Digitally Programmable Delay Element," *Proc. Ninth WSEAS Int'l Conf. Robotics, Control and Manufacturing Technology,* pp. 21-24, 2009.

[28] J.E. Stine et al., "FreePDK: An Open-Source Variation-Aware Design Kit," *Proc. IEEE Int'l Conf. Microelectronic Systems Education,* pp. 173-174, 2007.

[29] J.B. Nickel and A. Somani, "REESE: A Method of Soft Error Detection in Microprocessors," *Proc. Int'l Conf. Dependable Systems and Networks (DSN '01),* pp. 401-410, July 2001.
[30] T.M. Austin, "DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design," *Proc. Ann. Int'l Symp. Microarchitecture,* pp. 196-207, 1999.
[31] J.C. Smolens, B.T. Gold, B. Falsafi, and J.C. Hoe, "Reunion: Complexity-Effective Multicore Redundancy," *Proc. Ann. IEEE/ACM Int'l Symp. Microarchitecture,* pp. 223-234, 2006.
[32] A.K. Uht, "Achieving Typical Delays in Synchronous Systems via Timing Error Toleration," Technical Report 032000-0100, Univ. of Rhode Island, Mar. 2000.
[33] A.K. Uht, "Uniprocessor Performance Enhancement through Adaptive Clock Frequency Control," *IEEE Trans. Computers,* vol. 54, no. 2, pp. 132-140, Feb. 2005.
[34] B. Greskamp and J. Torrellas, "Paceline: Improving Single-Thread Performance in Nanoscale cmps through Core Overclocking," *Proc. Int'l Conf. Parallel Architecture and Compilation Techniques (PACT),* pp. 213-224, Sept. 2007.

Naga Durga Prasad Avirneni received the BE degree with honors in electrical and electronics engineering and the MSc degree with honors in physics from Birla Institute of Technology and Science, Pilani, Rajasthan, India, in 2007. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering, Iowa State University. His research interests include variability aware circuit design, computer system design and architecture, fault tolerant computing, and security computing. He is a student member of the IEEE.

Arun K. Somani received the MSEE and PhD degrees in electrical engineering from the McGill University, Montreal, Canada, in 1983 and 1985, respectively. He is serving as the Anson Marston distinguished professor and Jerry R. Junkins endowed chair professor of electrical and computer engineering at Iowa State University. He also served as a scientific officer for the Government of India, New Delhi, from 1974 to 1982, and as a faculty member at the University of Washington, Seattle, from 1985 to 1997. His research interests are in the areas of computer system design and architecture, fault tolerant computing, computer interconnection networks, WDM-based optical networking, and reconfigurable and parallel computer systems. He has also served as the IEEE distinguished visitor and the IEEE distinguished tutorial speaker and has delivered several key note speeches, tutorials, and distinguished and invited talks all over the world. He has been elected a fellow of the IEEE for his contributions to theory and applications of computer networks. He has been awarded a Distinguished Scientist/Engineer member grade of the ACM in 2006.