A Low-cost, Radiation-Hardened Method for Pipeline Protection in Microprocessors

Yang Lin, Mark Zwolinski, *Senior Member, IEEE*, and Basel Halak

***Abstract*—The aggressive scaling of semiconductor technology has significantly increased the radiation-induced soft error rate in modern microprocessors. Meanwhile, due to the increasing complexity of modern processor pipelines and the limited error-tolerance capabilities that previous radiation hardening techniques can provide, the existing pipeline protection mechanisms cannot achieve complete protection. This paper proposes a complete and cost-effective pipeline protection mechanism using a self-checking architecture. The radiation hardened pipeline is achieved by incorporating SETTOFF-based self-checking cells into the sequential cells of the pipeline. A replay recovery mechanism is also developed at the architectural level to recover the detected errors. The proposed pipeline protection technique is implemented in an OpenRISC microprocessor in 65nm technology. A gate-level transient fault injection and analysis technique is used to evaluate the error-tolerance capability of the proposed hardened pipeline design. The results show that compared to techniques such as TMR, the SETTOFF-based self-checking technique requires over 30% less area and 80% less power overheads. Meanwhile, the error-tolerant and self-checking capabilities of the register allow the proposed pipeline protection technique to provide a noticeably higher level of reliability for different parts of the pipeline compared to previous pipeline protection techniques.**

***Index Terms*—Fault-Tolerance, reliability, soft errors, single-event upset, single-event transient, timing error, fault injection.**

I. Introduction

The reliability of modern integrated circuits is severely challenged by strikes from high energy particles. A particle strike can produce soft errors, which can be categorized into Single Event-Upsets (SEU) and Single Event-Transients (SET). SEUs are transient bit-flip errors that invert the state held in memories (DRAMs or SRAMs) or sequential logic cells (flip-flops or latches). SETs are transient voltage pulses occurring in combinational logic. An SET become an SEU if it is sampled by a storage element. Lower operating voltages reduce the energy required to induce soft errors. Increased operating speed also significantly increases the probability that SETs are captured. These trends suggest that a dramatic increase in the soft error rate is inevitable.

Conventionally, microprocessors used in safety-critical applications, such as space, can be protected by Triple Modular Redundancy (TMR), [1]. However, TMR is not a viable solution for less critical applications since its overhead (more than 200% area and power) is far too expensive. There is room, therefore, for compromise techniques that offer a little less protection than TMR, but with significantly lower overheads.

The authors are with Electronics and Computer Science, Faculty of Physical Science and Engineering, University of Southampton, Southampton, UK, SO17 1BJ.
E-mail: {yl5g09,mz, bh9}@ecs.soton.ac.uk

Memory arrays and caches in microprocessors can be protected by conventional Error Correction Codes (ECC), which have acceptable overheads. Protecting the general logic in the pipeline of a microprocessor has always been a challenge as TMR or duplication are too expensive. Previous work has proposed using ECC to protect the Register File (RF) [2], [3], but the ECC bits need to be calculated and read during each operation. The performance and power overheads can be big for the RF [4], and the situation gets worse when there are multiple read ports and thus multiple ECC decoding circuitry is required. In addition, Multiple-Bit-Upsets (MBUs) and the captured SETs from combinational gates have become a serious issue at current technology nodes [5] [6]. ECCs cannot address SETs, and are very expensive for addressing MBUs as they require a larger number of redundant bits.

Other techniques have been proposed to mitigate either the SETs or the SEUs in general logic [7] [8] [9] [10] [11] [12] [13]. However, few of these can efficiently provide full protection of the whole pipeline against both SETs and SEUs. They are either only suitable for protecting the pipeline registers (such as RazorII [13], SEM/STEM techniques [11]), or only applicable in RFs (such as FERST [10]). On the other hand, most of these techniques rely on hardware redundancies, but are not self-checking, since errors occurring in the redundant circuitry can still corrupt data.

In this paper, we propose a novel pipeline protection technique based on a self-checking radiation-hardened register architecture. The technique is capable of providing cost-effective error-tolerance for a microprocessor pipeline. Our first contribution is the design of the self-checking register architecture, which is developed from the SETTOFF (Soft Error and Timing error TOlerant Flip-Flop), combined with a self-checker [4] [14]. The self-checker mitigates any errors occurring in the error-tolerant circuitry of SETTOFF. The overhead of the self-checker is minimized by sharing it between multiple SETTOFFs in a register. Our second contribution is a pipeline protection technique which incorporates the radiation-hardened self-checking registers into the RF, and the registers between each stage of the pipeline (henceforth referred to as the pipeline registers). SEUs occurring during pipeline execution will be detected and corrected on the fly within the register architecture. SETs and Timing Errors (TE) occurring in the combinational gates will be detected if captured by the pipeline registers or the RF. These errors then trigger a pipeline replay which re-executes the operation and corrects the error by re-writing the corrupted registers. In addition, the redundant circuitry added for error-tolerance within the register architecture is, in turn, protected by the self-checker and thus total self-checking is realized. We demonstrate the pipeline protection technique in an OpenRISC 1200 microprocessor. A fault-analysis model is developed to evaluate the reliability of the processor pipeline. The results show that no errors can propagate through the pipeline into the data memory.

This paper is organized as follows: Section II introduces the literature related to the work. Section III presents the self-checking radiation hardened register architecture. The design and implementation of the radiation hardening pipeline protection technique is given in Section IV, and Section V presents the experimental methodology and evaluation results. Finally, the paper is concluded in Section VI.

II. Background

Previous radiation hardening approaches fall into two main categories. Fault avoidance techniques aim to reduce the probability that the system is affected by particle strikes. Fault correction techniques detect and correct faults occurring in the system. Lin, et al proposed a fault avoidance technique using a Schmitt Trigger (ST) to construct radiation-hardened latches [15]. The ST-based latch has 112% higher critical charge than a conventional latch and can be used to reduce the SET error rate. Garg, et al also proposed a fault avoidance technique in which diodes dissipate energy from SEUs [16]. Another approach, [17], aims to re-construct circuit logic based on an error-analysis, to make a combinational circuit less sensitive to soft errors. One fault correction technique, [18], implements a coarse-grained reconfigurable architecture to include redundancy.

In actual system designs, fault avoidance techniques and fault correction techniques can be complementary. This section focuses on fault correction techniques which aim to provide cost-effective protection for microprocessor pipelines.

A. Previous pipeline protection techniques

RazorII is a pipeline protection technique proposed to tolerate both soft errors and timing errors within the pipeline [13]. RazorII protection relies on error-detection latches in the pipeline registers. All the error correction is achieved by using an architectural replay which re-executes the faulty operation and overwrites the erroneous state. As a result of the replay recovery process, RazorII protection may incur large Instruction Per Cycle (IPC) overheads when the error rate is high, and may therefore impact the overall energy efficiency. In addition, RazorII is only suitable for protecting the pipeline registers, but cannot protect the registers that store the architectural state of the processor (such as the RF). Thus, RazorII is not fully SEU-tolerant. The recovery operation can only correct SETs and TEs that occur in the preceding combinational gates, since they are detected during the write cycle of the flip-flop, and the faulty operation can then be re-executed. However, if an SEU is detected during the hold cycle of the flip-flop, the recovery mechanism can no longer find the last operation that wrote to the flip-flop and cannot re-execute it to overwrite the SEU. As a result, the RF is protected by ECC, and suffers from the usual ECC drawbacks.

Soft Error Mitigation (SEM) and Soft and Timing Error Mitigation (STEM), [11], both utilize a variant of TMR to mitigate SETs and SEUs. The STEM cell also adds timing error detection. One novelty of the techniques is that they remove error detection from the critical path, and therefore the delay overhead is reduced, compared to the TMR flip-flop. However, the area and power overhead of the SEM and STEM cells is approximately the same as or slightly bigger than the TMR flip-flop since additional recovery circuitry is added. The large overhead makes these techniques inapplicable for protecting the RF, and therefore they cannot achieve complete pipeline protection.

Error detection and correction flip-flop structures to balance performance, power and reliability of pipeline architectures have been proposed [19]. The four flip-flops provide different levels of error tolerance and overheads. By replacing the four FFs in the best storage locations of a pipeline, this approach can enhance circuit reliability with significantly lower overheads compared to SEM and BISER, [9]. Similar to Razor, this approach focuses on protecting the pipeline registers rather than the RF, as the proposed FF architectures can tolerate SETs, but do not provide full SEU-protection.

SVFD, [20], uses stability violation checking to detect SETs, SEUs and aging effects. The method has been implemented in an OpenRISC core, but the reported area and power overheads are 40% and 43%, respectively, somewhat greater than achieved with SETOFF.

A Confidence-Driven Computing (CDC) model is proposed in [21] for adaptive protection against transient faults. The approach estimates the confidence in a computation and allows repeated computations (in time or in space) to increase confidence. CDC has low overheads compared to conventional error-tolerant techniques, but it requires a controller to trigger duplicate computations. It has a much larger error detection window (multiple clock cycles) than Razor or rollback recovery techniques. CDC is efficient for enhancing the reliability of combinational logic in each pipeline stage, but does not aim to protect dense memory blocks such as the RF.

RF protection based on the FERST hardened cell has been proposed [10] [22]. FERST uses three C-elements to mitigate both SEUs and SETs at the input of the latch. FERST incurs nearly 100% area and power overheads. The main drawback of FERST is that a C-element is added in the signal path, which will induce a delay overhead of around 70% in 65nm technology. This make FERST hard to use to protect the timing-critical pipeline registers. Even in the RF, the delay overhead of FERST can have big impact on the performance.

B. Self-checking capability

Most pipeline protection techniques do not have a self-checking capability and so are vulnerable to soft errors in the redundant, error-tolerance circuitry. The area and geometry of the redundancies determine the probability that the circuit is hit by particles, while the critical charge determines the vulnerability of the circuit to particle strikes. On the other

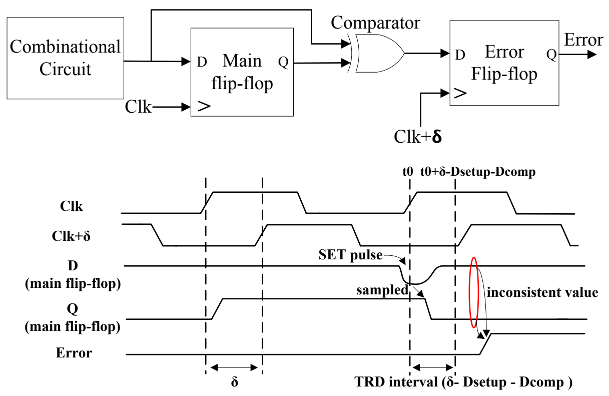


Fig. 1: Time redundancy based error detection flip-flop [12].

hand, if the unprotected redundancies are in combinational logic, such as the ECC or the majority voter in TMR, they may produce SET pulses which can cause errors if captured in a following stage. If the redundancy is in a state holding element, such as the C-elements in FERST, and BISER [9], particle strikes can produce SEUs in a similar manner to a latch, and corrupt the whole cell.

C. Time redundancy-based error detection

Because SETs occurring in the logic blocks only manifest themselves for a limited period of time, and will be recovered automatically, Time Redundancy-based error Detection (TRD) moves duplication into the time-domain [12] [23]. The technique is illustrated in Fig. 1. With no hardware duplication, TRD can detect SETs that are manifest at the input of the flip-flop with a maximum pulse width of $D_{tr} \leq \delta - D_{setup} - D_{comp}$, where D_{setup} is the setup time of the error flip-flop and D_{comp} is the delay of the comparator. Such SETs, if captured by the main flip-flop at t_0, will recover at $t_0 + \delta - D_{setup} - D_{comp}$, while the comparator will assert an error signal due to inconsistent inputs. Similarly, timing errors with a delay no greater than D_{tr} are also detected since the correct result will be presented at the input D when the comparison result is latched. This architecture can also detect SEUs in the main flip-flop from t_0 to $t_0+\delta-D_{setup}-D_{comp}$, which is called the TRD interval. Although TRD is cost-efficient, it cannot correct. Moreover, SEUs occurring in the main flip-flop outside the TRD interval will escape detection. This is dangerous, as SEUs cannot be recovered until the flip-flop is overwritten by the next input.

III. Self-Checking Radiation Hardened Register Architecture

To overcome the drawbacks of the previous techniques, we propose a novel full pipeline protection scheme. The technique is based on a self-checking radiation hardened register architecture that can address both SEUs occurring inside the register, and the SETs and TEs that are captured by the register. It has a self-checking capability that can, complementarily, protect the redundancies added for error-tolerance. This section presents the circuit-level approach of the register architecture, which is developed from the SETTOFF radiation hardened flip-flop [14], combined with a shared self-checker, adapted from that proposed in [4].

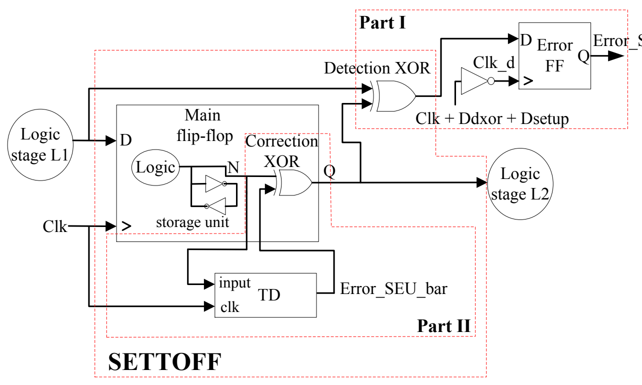


Fig. 2: The architecture of SETTOFF [14].

A. SETTOFF architecture

SETTOFF is a Soft Error and Timing error Tolerant Flip-Flop, which can recover the SEUs occurring inside the flip-flop on the fly, and can detect the captured SETs and TEs that originate from the preceding combinational logic gates.

The architecture of SETTOFF is shown in Fig. 2. The main flip-flop is a conventional flip-flop. For clarity, only the last storage unit (the inverter pair) is shown. Node *N* holds the state of the inverter pair. *Q* is the inverted value of node *N* in normal operation. The error-tolerant circuitry is divided into two parts, which work in turn during two intervals within a clock cycle. Part I is a TRD architecture adapted from [12]. The TRD part works during the TRD interval which is equal to the high clock phase. It detects errors occurring during the write cycle, which include captured SETs and TEs at the input, and the SEUs that flip node *N* during the TRD interval. On detection of an error, Part I asserts the *Error_SET* signal which can be used to trigger a replay mechanism to re-execute the erroneous write operation and overwrite the errors in SETTOFF.

Part II is the Transition Detector (TD) architecture which works during the TD interval (the low clock phase). SEUs that flip node *N* during the TD interval are interpreted as illegal transitions and are detected by the TD. A correction XOR-gate is used to replace the inverter driving the output *Q* of a conventional flip-flop. In normal operation, the *Error_SEU_bar* signal stays high, such that the correction XOR-gate acts as a normal inverter. When an illegal transition (an SEU) is detected by the TD, it assigns 0 to *Error_SEU_bar*. The correction XOR-gate will then propagate *N* to *Q* to correct the SEU on the fly. A correction glitch is generated upon correction of the SEU due to the delay of the TD. The glitch is not a threat because, if captured by the SETTOFF in the following stage, it will be detected by the TRD part as an SET pulse. Notice that only the SEUs that corrupt node *N* are considered, others are masked. We discussed this in more detail previously, [4], [14].

B. Circuit-level Evaluation for SETTOFF

The SETTOFF circuit architecture was implemented in a 65nm technology. The proposed error-tolerant architecture in

SETTOFF was modeled in SPICE. A conventional D-type flip-flop is used for the main flip-flop in SETTOFF. The power consumption and performance (Clock-to-Q delay and setup time) of SETTOFF was then measured by SPICE simulations, with 1.2V supply voltage, and a 185MHz clock. Table I shows the average power and performance overhead averaged over different transition times and load capacitances. The power consumption is measured with 10% activity rate, and the overhead is relative to a conventional flip-flop with the same drive strength.

TABLE I: Circuit-level evaluation results.

Power overhead	Clk-to-Q delay overhead	Setup time of FF
28.0%	15.3%	not changed

The reliability of SETTOFF is also evaluated using fault-injection and simulation in SPICE. Current sources are used to simulate the collected charge induced by particle strikes at circuit nodes. When sufficient charge is injected into a node, it will produce an SET or an SEU, depending on whether the node belongs to a combinational circuit or a storage element. During the simulation, SEUs are injected into both the master and slave latches of the main flip-flop in SETTOFF, and SETs are injected at the input logic. The faults are injected at time instances distributed across the entire clock period. Table II shows the fault simulation results.

C. Register Architecture

The TRD and TD-based parts in SETTOFF provide both SEU- and SET-tolerance for the main flip-flop. However, the added error-tolerant architecture can itself be struck by radiation particles and hence introduces extra vulnerability. For the TRD part, radiation particle strikes can induce SEUs in the error flip-flop, or SETs in the preceding comparator in the TRD part. Such SEUs or SETs, if captured by the error flip-flop, can generate a false $Error\_SET$ signal at the output of the TRD part. The false $Error\_SET$ signal invokes an unnecessary replay execution, but cannot corrupt the system. The unnecessary replay operations only incur an IPC overhead. Similarly, particle strikes in the TD-based part may propagate to its output and hence induce an erroneous $Error\_SEU\_bar$ signal, which can then propagate through the correction XOR-gate and corrupt the output of the flip-flop. This section introduces a self-checking mechanism to address this problem.

The architecture of an n-bit self-checking register is shown in Fig. 3. It is constructed from *n* SETTOFFs and a self-checker adapted from that in [4]. The self-checker can detect soft errors that corrupt the TDs in each SETTOFF, by monitoring the register output during the interval when TD is enabled. One self-checker is used to monitor all the outputs of all the SETTOFFs in the register through a parity checker. The parity checker is constructed as an n-input XOR-tree. Any illegal transitions occurring at the output of any single SETTOFF will change the parity, and therefore will be detected. Upon detection, the parity checker generates a transition at its output.

TABLE II: Transient fault injection simulation results for SETTOFF

Injected SETs	50	Injected SEUs	100
Captured SETs	41	Detected/corrected SEUs	100
Detected SETs	41		

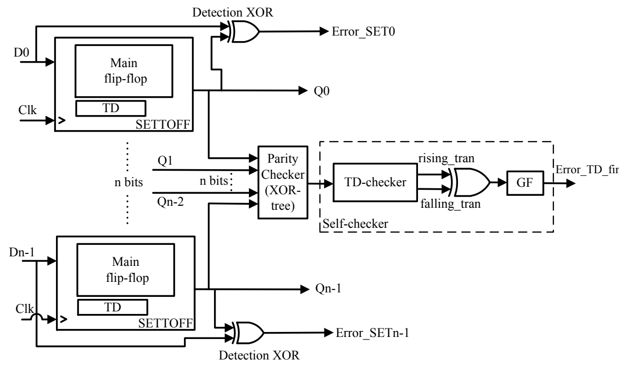


Fig. 3: Architecture of an n-bit self-checking error-tolerant register.

Such transitions are then captured by the self-checker, which asserts the *Error_TD_final* signal.

The transistor-level design of the TD-checker, Fig. 4, is adapted from the transition detector built into SETTOFF [14]. The two delay chains of inverters and transmission gates remain unchanged. The dynamic OR-gate for capturing the implicit pulses generated by the delay chains is separated into two branches, both driven by the system clock. During the TRD interval, when the clock is high, nodes *M1* and *M2* are charged, the TD-checker is disabled, and both of the outputs, *rising_tran* and *falling_tran*, stay low. During the TD interval, when the clock is low, the TD-based part of SETTOFF is vulnerable to soft errors. Therefore, the two branches are both enabled, to capture the pulses generated by the delay chain for the rising and falling transitions, respectively. Any rising transitions at the input of the TD-checker will discharge node *M1* through transistors *d1* and *d3*, and thus will be signaled at the output *rising_tran*. Similarly, any falling transitions will assert *falling_tran* through the respective branch.

The TD-checker can distinguish correction glitches from transitions caused by errors in the TD. A transition can only assert one of the two outputs of the TD-checker. However, a glitch consists of both a falling and a rising transition, and thus will assert both outputs of the TD-checker. The two outputs, *rising_tran* and *falling_tran*, are then XOR-ed to generate a valid error signal, which is asserted when only one of its input is high. The error signal will stay at 0 when both inputs are 0, or when both inputs are asserted due to a correction glitch.

Notice that there is a possibility that correction glitches at the output of the register will propagate through the TD-checker. This can be caused by the rising and falling transitions not asserting the *rising_tran* and *falling_tran* signals at exactly the same time. The time difference may lead to a positive glitch appearing at the output of the XOR-gate. A Glitch Filter (GF), [4], is used to filter out these glitches and generate the final error signal *Error_TD_final*.

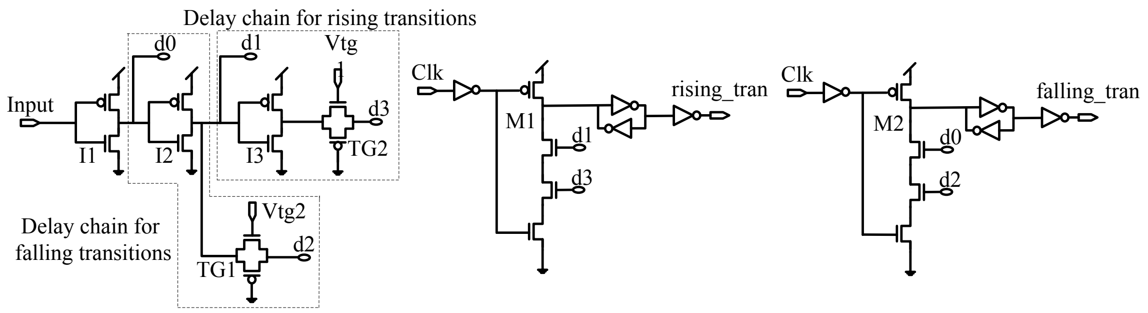


Fig. 4: The transistor level design of the TD-checker.

D. Error-Tolerance Analysis of the Self-Checking Register

Table III summarizes the types of errors that may occur in the proposed register; whether the errors can cause erroneous outputs; whether they can be tolerated; and how they are tolerated. The radiation induced transient faults affecting the register can be categorized into 5 types: (1) captured SETs originating from the preceding combinational logic. (2) SEUs corrupting the output of the main flip-flop; (3) errors corrupting the TRD part; (4) errors corrupting the TD-based part; and (5) errors corrupting the self-checker.

The captured SETs and the SEUs in the main flip-flop can induce erroneous outputs at the register, but they can be tolerated by the TRD and TD-based parts, respectively. As discussed in Section III-C, errors corrupting the TRD part can induce a false $Error\_SET$ signal, which invokes an unnecessary replay operation, but does not corrupt the output of the register. Therefore, these errors do not need to be addressed. The errors corrupting the TD-based part during the TD interval affect the register output through the correction XOR-gate, thus they are mitigated by the self-checker. The self-checker can also be affected by radiation strikes. SEUs can arise at the state-holding nodes, $M1$ and $M2$ of the TD-checker, which will invert either of its outputs, $rising\_tran$ or $falling\_tran$, and generate a false $Error\_TD\_final$ signal (Fig. 4). As with the errors in the TRD part, the false $Error\_TD\_final$ signal does not corrupt the register output since it will invoke a redundant recovery process. Therefore, the self-checker does not need to be protected.

TABLE III: Error-tolerance analysis of the self-checking register.

Error type	Output Corrupt	Tolerated	Means
1. Captured SET	Yes	Yes	TRD arch.
2. SEU in main ff	Yes	Yes	TD-based arch.
3. Errors in TRD arch.	No	No	none
4. Errors in TD-based arch.	Yes	Yes	self-checker
5. Errors in self-checker	No	No	none

The built-in TRD and TD-based parts in each SETTOFF in the register allow the proposed register to tolerate an unlimited number of Multiple-Bit Upsets (MBUs) in multiple main flip-flops. The shared self-checker can also address MBUs corrupting multiple TD-based parts during the same cycle. This is because the first corruption will be detected, and the recovery operation will then reset all the TDs in the register. However, if an even number of upsets simultaneously corrupt multiple TDs, they can escape detection since the parity of the register output will not change.

E. Register Evaluation

To verify and evaluate the method at sub-system level, a 32-bit self-checking register based on SETTOFF2 was synthesized in 65nm technology and simulated in SPICE. The supply voltage was 1.2V. A 185MHz symmetric clock was used to drive both the register (positive edge), and the error flip-flop (negative edge) of the TRD part. The power consumption of the self-checking register was compared to a conventional register with the same operating conditions and drive strength. With a 10% activity rate for a single bit, the average power overhead of the proposed register is 33%, which is only a 5% increase over SETTOFF2 without the self-checking capability (Section III-B). In terms of area, a single SETTOFF2 requires 30 extra transistors. The proposed register only adds one self-checker, shared between bits, thus the area overhead increase is insignificant. Compared to a conventional register constructed from flip-flops with 32 transistors each, the area overhead of the 32-bit self-checking register is 136%. Since the self-checker is not added to the signal path of the register, the delay overhead of the register is comparable to that of a single SETTOFF2, with an average value of 16.5%. Compared to a technique such as TMR which induces over 200% power and area overhead, the self-checking register requires over 30% and 80% less area and power overhead, respectively.

A current source based fault-injection mechanism was used to verify the reliability of the self-checking register. The redundancies in the register are separated into 3 parts: the TRD part, the TD-based part, and the self-checker. As described in Section III-D, the TRD part and the self-checker are not vulnerable to radiation strikes. Therefore, transient fault injection and simulation was only carried out in the TD-based part in each SETTOFF. Because only the errors flipping the state held by the TD during the TD interval can corrupt the output of the TD-based part in SETTOFF, a current source was used to inject SEUs into the TD during the TD interval.

30 SEUs were injected into the TDs in each SETTOFF in the register, thus a total of 940 errors were simulated. The injection times of the SEUs were evenly distributed during the TD interval. The simulation results show that of the 940 SEUs, 884 were detected by the self-checker, and only 56 errors (6%) escaped. The escapes are caused by the delay of the parity checker, which causes the SEUs to reach the self-checker outside the TD interval. The problem can be addressed by using more self-checkers to reduce the number of the SETTOFFs sharing each of them, since this would allow a small parity checker with fewer levels of XOR-gates.

To validate the MBU-tolerant capability of the proposed register, multiple faults were injected into the register to corrupt multiple bits of the register simultaneously. The simulation results demonstrate that all the MBUs are either individually corrected on the fly at the outputs of the corrupted SETTOFFs, or detected by the TRD parts incorporated in each bit.

Table IV compares the implementation overheads and error-tolerance capabilities of several techniques for protecting a 32-bit register in 65nm technology. Notice that ECC is implemented with SEC-DED (Single Error Correction Double Error Detection) coding which requires 7 redundant bits for a 32-bit register. The delay overhead of ECC is big due to the large decoding block, therefore an extra cycle may be required to reload the register for error correction. Due to the C-element added into the signal path, the delay overheads of FERST and BISER are similar to that of TMR. The area overhead of BISER is small since it uses the existing scan flip-flops as duplicates. STEM uses a variant of TMR which removes the error correction from the signal path. The delay overhead of STEM is small[1]. Compared with other techniques, our proposed register incurs smaller overheads. In terms of reliability, only the proposed register and STEM can tolerate both SETs, SEUs, and TEs; and only the proposed register has a self-checking capability. Errors occurring in the redundant circuit of other techniques may either corrupt the cells directly (e.g. FERST and BISER), or produce SET pulses (e.g. TMR, ECC, and STEM).

IV. Self-Checking Hardened Pipeline in OpenRISC

The complete pipeline protection technique was realized in the pipeline of an OpenRISC 1200 microprocessor [24]. This is a 32-bit scalar RISC machine with a Harvard architecture [24]. It has a 5-stage integer pipeline and a dual-port Register File (RF) constructed from 32 general purpose registers. A soft-error injection and analysis model has been developed to evaluate the reliability of the microprocessor through gate-level simulation. In this section, we present the design of the microprocessor pipeline, the implementation procedure, the evaluation methodology, and the results that show the reliability of the protected processor and the error-tolerance overhead.

A. Soft Error Vulnerability Analysis of the OpenRISC Pipeline

As discussed in Section I, the proposed technique aims to provide a cost-effective soft error solution for a micro-

[1]The delay overhead of STEM was not reported by the authors in [11].

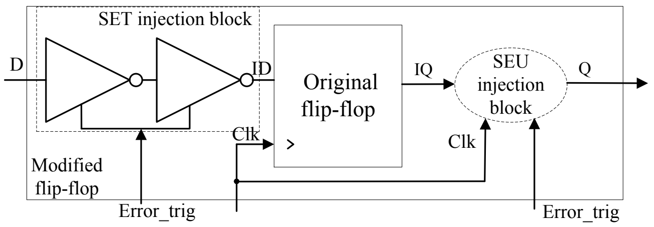


Fig. 5: The modified D flip-flop for error injection.

processor pipeline for use in non-safety-critical ground-level applications, such as mobile devices. It is therefore important to determine the most efficient way to incorporate the technique into the pipeline. This section analyses the soft error vulnerability of the OpenRISC pipeline, on which the reliable pipeline design is based.

The idea of the proposed pipeline protection technique is to protect the registers within the pipeline using the self-checking radiation-hardened register described in Section III. Only the SETs and TEs captured by the registers can propagate through the pipeline, therefore the pipeline protection technique can address both types of soft error within the pipeline. However, the large number of registers in the pipeline could mean big overheads to protect them all. In order to achieve effective protection, a system-level analysis was carried out to identify the soft error vulnerability of different registers inside the pipeline. The registers which are most vulnerable to soft errors can then be selected for protection.

Mehdizadeh, et al presented an analysis of the fault effects in the OpenRISC processor in [25]. They injected different types of faults (such as stuck-at faults and bit-flips) into the processor, and checked whether the execution results were corrupted. Their results show that different units in OR1200 present different vulnerabilities to the injected faults. The levels of vulnerability of different units are independent of software execution. The register file unit, for instance, is always more vulnerable than the WB-multiplexer unit. Ebrahimi, et al also presented an SER analysis of the OpenRISC processor in 45nm technology [26]. Their results show that 75% of the flip-flops within the microprocessor make negligible contributions to the overall system SER, and by protecting the most vulnerable 20% flip-flops, the system SER caused by all the flip-flops can be reduced by 80%. They have also shown that the level of vulnerability of the flip-flops, again, has little dependency on different workloads. Based on these observations, a system-level SER analysis was developed.

The system-level analysis was realized by using a dynamic transient fault injection and analysis model developed from the technique described in [27]. Transient faults were injected into the flip-flops of the pipeline of the processor during gate-level VHDL simulation. The simulation results were then collected to statistically analyze the soft error effects in different parts of the system.

Fig. 5 shows a D flip-flop cell modified for transient fault simulation. Two fault injection blocks, one for SET injection, and one for SEU injection are added. The SET injection block consists of 2 modified inverters at the input of the flip-flop,

TABLE IV: Comparison of error-tolerance capability and overheads for 32-bit registers.

	Self-checking Reg	TMR	ECC	FERST	BISER	STEM	DICE
Area overhead	136%	210%	206%	110%	24%	210%	100%
Power overhead	33%	210%	160%	100%	126%	210%	78%
Delay overhead	16.5%	70%	1 cycle	70%	70%	-	67%
SET-tolerance	YES	NO	NO	YES	NO	YES	NO
SEU-tolerance	YES	YES	YES	YES	YES	YES	YES
TE-tolerance	YES	NO	NO	NO	NO	YES	NO
Self-checking	YES	NO	NO	NO	NO	NO	NO

such that the original input becomes an internal node, *ID*. The actual input of the modified flip-flop, *D*, is the input of the SET-injection block. SETs are modeled as transients at the outputs of each inverter, which will propagate to the input of the flip-flop. The SEU injection block is added to the output of the flip-flop such that the original output becomes an internal signal (*IQ*). The SEU injection block injects SEUs (i.e. bit-flip errors) to the actual output of the modified flip-flop (*Q*), by flipping the state of *IQ*.

The *Error_trig* signal is used to activate pre-determined faults in either of the fault injection blocks. When an SET is activated, the SET injection block will produce an SET pulse with pre-determined width at *ID*. When an SEU is activated, *IQ* is inverted at *Q* to simulate the error, and the bit-flip is not recovered until the flip-flop samples the new input *D* on the rising-edge of the *Clk*, which will overwrite the bit-flip and propagate *IQ* to *Q*.

Using this fault injection technique, an analysis was carried out of the ORPSoc platform, which is the smallest-possible reference system for the OpenRISC processor [24]. During the simulation, a program was loaded into the processor and a fault-free simulation was run to extract the correct program outputs saved in the data memory. Afterwards, the processor was reset and the same program run again with a single fault (either an SET or an SEU) activated at a random time instance. All the pre-determined faults in all the flip-flops within the pipeline were activated one by one. When a fault propagated and corrupted the final outputs stored in the data memory after running the program, it was recorded as a visible soft error.

Fig. 6 shows the soft-error vulnerability results for different registers within the pipeline, based on the execution of two programs: quicksort and tak. Regardless of the workload, the vulnerability levels of different registers is relatively constant. The PC register shows the highest vulnerability. A total of 90 transient faults were injected into the PC register during each program execution. 31 (37.8%) and 34 (34.4%) faults caused corruptions in the execution results for quicksort and tak, respectively. This is because faults occurring in the PC can easily corrupt the pipeline execution by fetching erroneous instructions. The flip-flops that store the control signals generated by the decoders in each stage of the pipeline present the second highest vulnerability. This is because the control signals are also critical for correct pipeline execution. Other registers that present relatively high vulnerabilities are: the registers that store the instruction for the ID (*id_insn*) and IF stages (*if_insn*); the registers that store the operand for the execution stage (op1 and op2); the RF; and the flag registers.

The registers that store the instructions in the EX (execution) stage (*ex_insn* register) and the WB (write-back) stage (*wb_insn* register) had 0% vulnerability despite the injection of 198 transient faults in both benchmark executions. The reason for this is that most of the control signals are decoded in the IF and ID stages. The *ex_insn* register is very rarely used for instruction decoding in the EX stage, hence transient faults occurring in the *ex_insn* hardly affect the pipeline execution. Similarly, *wb_insn* is not used for the decoding process. *wb_reg* also manifested 0% vulnerability for both simulations. This is because *wb_reg* is only used for forwarding execution results in the WB stage back into the pipeline. Most of the time, the execution results written out from the WB stage are stored in the RF, which is then read in the ID stage where the execution results will be used for later instructions. Therefore *wb_reg* has a low vulnerability to radiation hits. It should be noted that the low vulnerability of *wb_reg* does not mean that WB stage of OpenRISC is not susceptible to soft errors, since the main storage in the WB stage, the RF and the flag registers present high vulnerability.

B. The Radiation Hardened OpenRISC Pipeline

Based on the soft error vulnerability results presented in Fig. 6, a pipeline protection technique was developed in the OpenRISC 1200 microprocessor. The sequential cells, *if_insn*, *id_insn*, *if_ctrls*, *id_ctrls*, *ex_ctrls*, RF, flags, *op1*, *op2*, and PC are selected for protection since they manifest the highest vulnerabilities. The radiation hardened pipeline design of the OpenRISC processor is shown in Fig. 7. All the combinational logic blocks are represented by white boxes, while the sequential blocks (including the flip-flops, registers, and the caches) are represented by orange boxes. Notice that the memory access (MEM) stage is optional and is only invoked when a load or store instruction communicates with the data cache. Otherwise, the instruction goes directly to the Write Back (WB) stage after the Execution (EX) stage and writes the execution results into the RF and the flag registers. The blue boxes are the pipeline registers.

The pipeline architecture can be divided into speculative and non-speculative domains. The speculative domain consists of the IF, ID and EX stages of the pipeline. The registers and flip-flops in the speculative domain commit speculative executions, which do not change the architectural state of the pipeline until the results are stored in the non-speculative domain in the WB stage. The non-speculative domain consists of the registers updated in the WB stage of the pipeline. These registers, such as the RF and the flag registers, contain the intermediate execution results and architectural states of the pipeline.

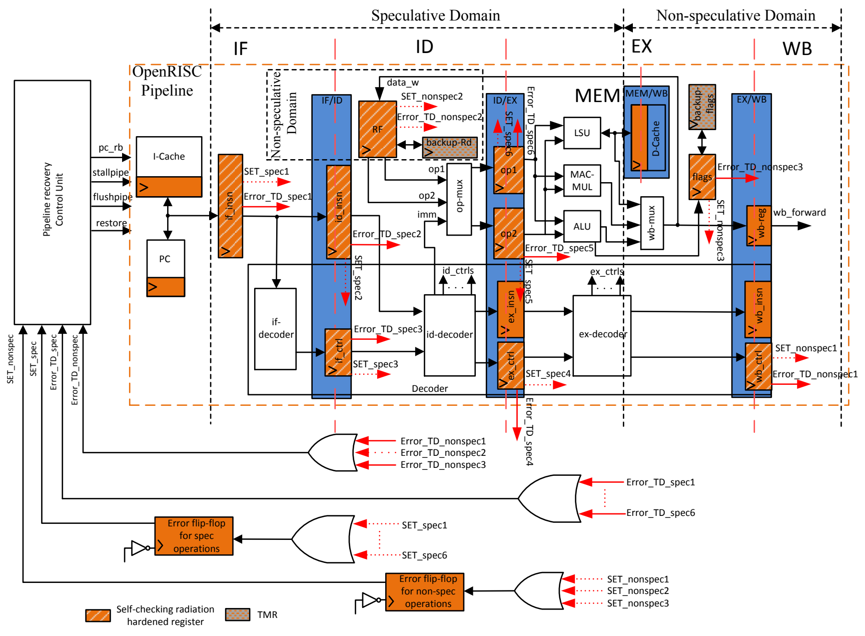


Fig. 7: The robust pipeline design of the OpenRISC processor

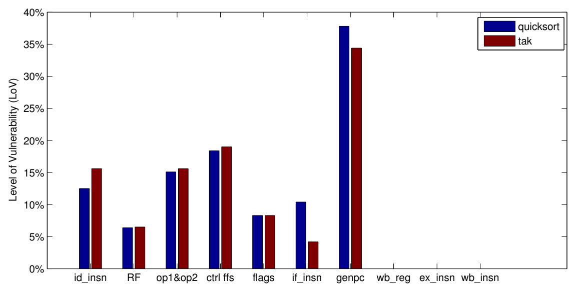


Fig. 6: Soft error vulnerability analysis for OpenRISC processor.

The vulnerable registers, *if_insn*, *id_insn*, *op1*, *op2*, the RF, and the flag registers are all protected by the SETTOFF-based self-checking radiation register architectures proposed in Section III-C. The vulnerable flip-flops storing the control signals between each pipeline stage, *if_ctrls*, *id_ctrls*, and *wb_ctrls*, are replaced by the SETTOFF architecture. In order to realize the self-checking capability in these SETTOFF-protected flip-flops, two self-checkers are added, with one shared by the *if_ctrls* and *id_ctrls* flip-flops, and the other shared by the *wb_ctrls* flip-flops. The PC register is protected by a TMR architecture since it is the most vulnerable part of the processor.

All the *Error_TD* signals from self-checkers for the registers and flip-flops in the speculative domain are OR-ed together. The resulting *Error_TD_spec* signal is fed into the pipeline recovery control unit to trigger corresponding recovery operations. Similarly, the *Error_TD* signals from self-checkers for the registers and flip-flops in the non-speculative domain are also OR-ed together and fed into the control unit.

The *Error_SET* signals generated from the TRD part of the SETTOFFs in each register are OR-ed together. Meanwhile, the *Error_SET* signals generated from the flip-flops storing the control signals in the IF stage, ID stage, and EX stage are also OR-ed together. The resulting signals from these OR-ed *Error_SET* signals are shown by the red dashed arrows. The *Error_SET* signals generated by the registers and flip-flops in the speculative domain (including *SET_spec1* to *SET_spec6*) are further OR-ed together, and the final result is fed into a shared error flip-flop for speculative operations. Similarly, the *Error_SET* signals generated by the registers and flip-flops in the non-speculative domain (including *SET_nonspec1* to *SET_nonspec3*) are further OR-ed together, and the results are fed into a shared error flip-flop for non-speculative operations. Rather than generating a new delayed clock, we simply used the falling edge of the system clock to drive both of the error flip-flops. Certain constraints need to apply to the TRD part to effectively address SETs. The realization of a sufficient TRD interval, and the approach for meeting the constraints required by the TRD part are further considered in Section V-B. Notice that the two error flip-flops only capture SET error signals during the write cycle of the registers when the TRD parts are enabled. The two error signals, *SET_spec* and *SET_nonspec*, are fed into a pipeline recovery control unit which controls the recovery operations within the pipeline. The principle of the recovery operations for the errors detected by the TRD parts are described in Section IV-C.

Two back-up registers, *backup_rd* and *backup_flag*, are added to back up the destination register in the RF and the flag register, which are updated by the instructions in the WB stage. The backed-up data is used for restoring the RF and flag registers during the recovery operations for registers that are updated during the WB stage (see Section IV-C for details of the replay operation). Both of the back-up registers are protected by TMR.

C. Error-Tolerance Operation in the Pipeline Architecture

Soft errors inside the pipeline can be randomly located. However, they can be categorized into 3 types:

1) SEUs that corrupt data stored in the registers.
2) SET pulses that are produced by the combinational logic.
3) Soft errors occurring in the redundant circuitry added for error-tolerance.

As discussed in the previous sections, errors of type (1) can be detected and recovered on the fly at circuit level, therefore no extra recovery operations are required.

For type (2) errors, the SETs occurring in each stage of the pipeline will be detected by the radiation-hardened registers incorporated in the corresponding stage when they are captured. An architectural recovery operation is then required for correcting such SETs. As is suggested by the two SET error signals (*SET_spec* and *SET_nonspec*) generated from the two flip-flops, there are two types of architectural replay recovery operations which re-execute the instruction in either the EX stage or the WB stage, depending on the following three cases:

Case (1): If an SET occurs in the IF or ID stage of the pipeline, it will be detected by the SETTOFF-based sequential cells incorporated in the speculative domain of the pipeline. The detection of such SETs will assert the *SET_spec* error signal, which will trigger a replay operation at the beginning of the following cycle, before the SET contaminates the register file and the flag registers in the non-speculative domain. Since in this case, the SET only corrupts the speculative operations, and none of registers storing the architectural state of the pipeline are affected, the replay operation for case (1) will simply flush the entire pipeline, re-fetch and re-execute the instruction in the execution stage (i.e. the instruction in *ex_insn*) to overwrite the captured SET.

Fig. 8 shows an example timing diagram for the recovery process of an SET captured by the *id_insn* (Case (1)). *PCn* stands for the PC value for instruction *INSNn*. Here the *INSN3* captured by the *id_insn* at the rising clock edge of cycle 3 is corrupted by an SET occurring in the decoder in the IF stage. If the pipeline is unprotected, the ID stage of the pipeline will decode the erroneous *INSN3* stored in *id_insn* during cycle 3, and generate erroneous control signals to corrupt the execution stage. The corrupted execution results will then be forwarded to the registers in the non-speculative domain (RF, flag registers, and *wb_reg*). However, in the protected pipeline, the captured SET is detected by the TRD part in *id_insn* at the falling clock edge of cycle 3, which asserts the *SET_spec* signal in the error flip-flop for speculative operations. The *SET_spec* signal then triggers the recovery operation for Case (1). The recovery operation asserts the *flushpipe* signal, which flushes all the pipeline registers at the rising clock edge of cycle 4 to prevent the erroneous execution from propagating. The *stallpipe* signal is also asserted to stall the pipeline and disables all the registers from updating the results from executing the instruction *INSN3* stored in *ex_insn* in cycle 4. The PC is rolled back to the PC value in the EX stage (i.e. the *PC2*), by the *PC_rb* signal. When the new PC value becomes available at cycle 6, the pipeline starts fetching the new instruction and normal operation resumes (the NOP instruction does not contain any valid executions). The SET is not repeated in the IF stage during the re-execution in cycle 8, so that *id_insn* captures the correct *INSN3* after the replay. The recovery process for Case (1) ensures that the errors in all pipeline registers in the speculative domain are overwritten during re-execution to prevent them corrupting the WB stage.

Case (2): If an SET occurs in the EX stage, it can corrupt the non-speculative registers, such as the RF, the flag register, and the *wb_ctrl* flip-flops during the WB cycle. The recovery operation in this case will re-execute the instruction in the WB stage (i.e. the instruction in *wb_insn*) to re-write the RF, the flag register, and the *wb_ctrl* flip-flops. However, unlike the replay operation in Case (1), the intermediate data and the architectural state of the pipeline might have been corrupted by the SET. These data and pipeline states may be used as operands or other operation conditions during the re-execution. Therefore for Case (2), the RF and the flag register need be restored using their back-up registers before the recovery operation is committed.

Fig. 9 depicts an example timing diagram for the recovery process of an SET captured by the RF (Case (2))[2]. When the execution in the EX stage is about to write its results into the destination register (*Rd*) in RF and the flag register, both *Rd* and the flag register are saved into their back-up registers before they are updated. *resultn* and *flagn* stand for the execution results and the flags generated by the *INSNn* in the EX stage. In cycle 3, the captured *result4* in *Rd* is corrupted by an SET occurring in the EX stage. The *SET_nonspec* signal is asserted upon detection which invokes the recovery process for Case (2). The pipeline is flushed and stalled while the PC is rolled back to the PC value in the WB stage (i.e. *PC4*), to re-fetch and re-execute the instruction in *wb_insn* (i.e. *INSN4*) which corrupts the RF. In addition, *Rd* and the flag register are restored by the back-up registers in cycle 4 to ensure that the system state is consistent with the original execution during the re-execution. In cycle 10, *Rd* is re-updated and recovered by the re-execution. The recovery process for Case (2) ensures that the all SETs captured in the WB cycle can be mitigated.

Case (3): A third circumstance happens when a captured SET is detected while a branch instruction is executing. Fig. 10 shows the timing diagram for Case (3). *PC2(brc)* is the PC value for the branch instruction *brc*. *PC100* is the branching address. In OpenRISC, the branch operation is committed in the ID stage to force the PC to jump to the branching address. Therefore an invalid PC (*PC3(dg)*) is forwarded into the pipeline before the branching PC, to fetch *INSN3*. A dangerous situation can happen if an error is detected when *INSN3* is in the *ex_insn*. In this case, re-fetching the instruction in *ex_insn* only forwards *PC3(dg)* into the pipeline, therefore the branch instruction stored in *PC2(brc)* will not be executed during the replay. This results in the branch operation not being detected during the re-execution, such that the processor fails to jump to the branching address. The problem is solved by monitoring

[2]Control signals (*flushpipe*, *stallpipe*, *PC_rb*) are not shown in Fig. 9 since they operate as shown in Fig. 8.

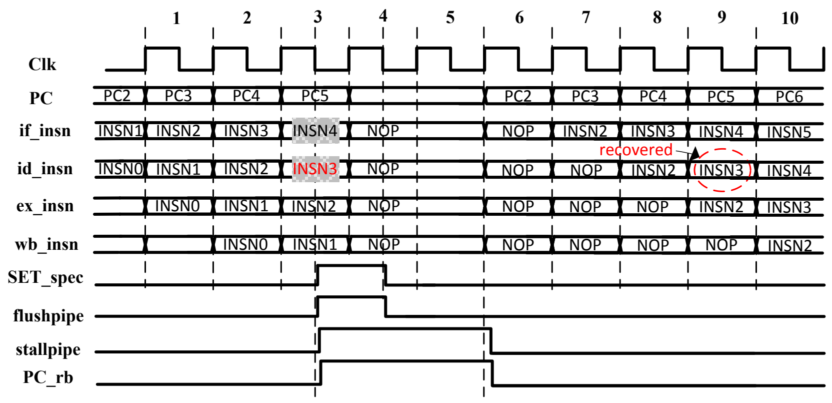


Fig. 8: Timing diagram of the pipeline recovery operation for Case (1)

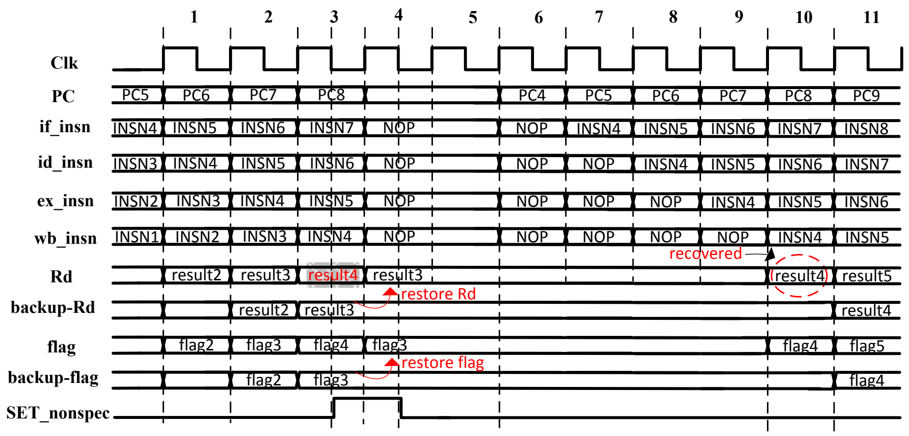


Fig. 9: Timing diagram of the pipeline recovery operation for Case (2)

the branch operation within the pipeline, and when such a situation occurs, the branch PC in the WB stage (*PC2(brc)* in this case) is replayed as is shown in Fig. 10. The error is corrected in cycle 11 in *id_insn*.

Most of the SETs detected by the TRD part are transient errors that will not occur again during the re-execution. In such cases, the recovery process can successfully recovers the error at one time. If other SETs occur during the replay operation, they will be detected and will trigger further replays until no error is detected.

Errors of type (3) include errors that corrupt the TRD part; the TD-based part; and the self-checker.

The errors that corrupt the TD-based part are detected by the self-checkers, which will also be recovered by the replay recovery mechanism for errors of type (2). The errors generated from the speculative domain will assert the *Error_TD_spec* signal (see Fig. 7), which will then trigger a replay execution to flush the pipeline, re-fetch and re-execute the instruction in the EX stage. The errors from the non-speculative domain will assert the *Error_TD_nonspec* signal, which triggers a replay execution that replays the instruction in the WB stage. During both replay operations, all the pipeline registers and flip-flops storing the control signals will be re-written, such that all the TD-based parts in these cells are reset. The errors generated from the TD-based part of these cells will be recovered after the replay, when normal operation can resume. Nevertheless, if an error is detected from the TD-based parts in the RF or the flag register, the *Error_TD* signal will also trigger a reset signal to reset the TD-based parts in the corresponding register which generates the error, besides triggering the replay operation. This is because the RF and the flags might not be updated during the replay, such that the reset of TD may not be triggered automatically. The erroneous TD-based part can stay corrupted unless the reset operation for the TD is explicitly executed.

The errors corrupting the TRD part and the self-checker may produce a faulty error signal, triggering an unnecessary recovery process. However, such errors do not corrupt the pipeline execution results. Therefore, no extra recovery operations are involved.

V. Experimental Setup and Results

A. *Implementation Details and Error-tolerance Overheads*

The OpenRISC processor with the proposed radiation hardened pipeline architecture was synthesized to a 65nm technology, with 1.2V supply voltage and 185MHz clock frequency. The transistor level implementation of SETTOFF and the self-checker which construct the radiation hardened register architecture, were characterized as new cells using Synopsys Liberty NCX. The behavioral model of the OpenRISC 1200 processor was re-designed to incorporate the new pipeline architecture described in Section IV, and was then synthesized

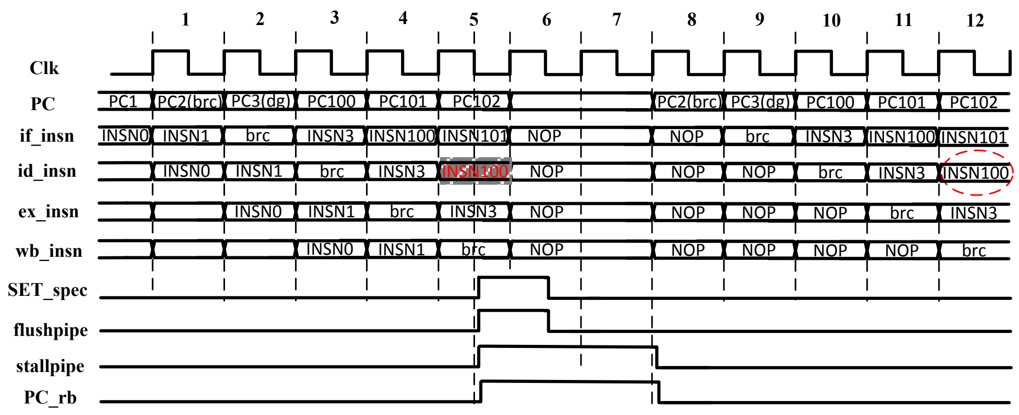


Fig. 10: Timing diagram for pipeline register recovery with branch instruction (Case 3)

to cell level using the characterized technology library. The 32-bit pipeline registers and the registers in RF were replaced by the 32-bit proposed radiation hardened registers. Gate-level simulation was carried out, based on the ORPSoc platform which provides the smallest-possible reference system for testing the processor [28].

The implementation details of the SETTOFF2-based radiation-hardened OpenRISC processor are summarized in Table V. The results come from evaluating only the core of the processor; the power consumption and area of other parts of the processor, such as the caches and memory were not considered. The total error-tolerance area overhead and power overhead for the radiation-hardened OpenRISC processor are 26.7% and 17.8%, respectively, and break down as follows: the incorporated replay recovery architecture and the TRD circuitry of all SETTOFF2s incur 13% more area and 2.4% more power consumption. A total of 956 buffers were inserted into the pipeline to achieve the timing constraint for the TRD parts. The inserted buffers incur 1.5% extra area and 0.5% more power consumption in respect to the whole processor core. A total number of 1219 SETTOFF2s and 39 self-checkers were incorporated. These self-checking architectures incur an area overhead of 12.1% and a power consumption overhead of 14.7%. The system-level implementation overhead is analyzed and compared with other techniques in Section V-D.

B. Setting TRD and TD Intervals and Clock Management

Only one clock is used for driving both the system flip-flops (by using the positive clock edge), and the error flip-flops in the TRD part (by using the negative edge of the clock). The TRD and TD intervals of the incorporated SETTOFF architectures are hence decided by the frequency and the duty cycle of the clock.

The choice of the TRD and TD intervals is based on providing desirable SET and SEU coverage. The two intervals formed by the two clock phases represent a trade-off between SET- and SEU-tolerant capabilities. A bigger TRD interval allows more SETs with bigger widths to be addressed, while a bigger TD interval can detect and correct more SEUs on the fly. Since the SEUs that are not corrected during the TD

TABLE V: Radiation hardened processor implementation details

Technology node	65nm
Frequency	185MHz
Supply Voltage	1.2V
Temperature	25°C
Total Area (μm)	121939
Total Power (mW)	22.4
No. of SETTOFF2s incorporated	1219
No. of self-checkers incorporated	39
Area Overhead of the self-checking arch.	12.1%
Power Overhead of the self-checking arch.	14.7%
Replay Recovery Area Overhead	13%
Replay Recovery Power Overhead	2.4%
Number of Buffers inserted for TRD	956
Buffer Area Overhead	1.5%
Buffer Power Overhead	0.5%
Total Error-tolerance Area Overhead	26.7%
Total Error-tolerance Power Overhead	17.8%

interval will still be addressed during the TRD interval, the aim is then to provide a sufficient TRD interval to efficiently reduce both the SET and SEU failure rate of the SETTOFF. The widths of the potential SET pulses are the main factors for determining the TRD interval. As reported in [29], the mean and the standard deviation of the width of the SET pulses in 65nm technology are $530ps$ and $150ps$, respectively. According to the circuit-level evaluation results for SETTOFF presented in [14], a minimum $800ps$ TRD interval was chosen because it can reduce the SET failure rate of the SETTOFF to 0 in 65nm technology. With an $800ps$ TRD interval, the SEU failure rate of the SETTOFF is also reduced to 0 since it covers all the correction glitches.

Finally, buffers were inserted into the shortest path of the combinational logic in each pipeline stage to satisfy the shortest path constraint of the TRD technique for all the incorporated radiation-hardened cells. The buffers that provide the maximum delay with minimum power consumption and area occupation were chosen to minimize the incurred overhead.

C. Reliability Analysis for the Radiation Hardened Processor

The reliability and the implementation overheads of the proposed radiation-hardened processors were evaluated through gate-level simulations. The transient fault injection technique

TABLE VI: Reliability analysis results for the radiation hardened OpenRISC

Program	Faults injected into the original cells		Faults injected into the redundancies
	4050		1411
	Errors in original OpenRISC	Errors in radiation hardened OpenRISC	Errors in radiation hardened OpenRISC
quicksort	306	0	0
tak	305	0	0
matrix multiplication	354	0	0

and the simulation methodology used for analyzing the error vulnerability of the original OpenRISC described in Section IV-A were used for evaluating the reliability of the radiation-hardened processor. However, besides injecting transient faults into the original cells, the same method was also used to inject transient faults into the redundant circuitry of the processor added for error-tolerance. To be specific, the transient faults injected into the redundant circuitry include:

- SEUs injected into the transition detectors to corrupt the TD-based parts in SETTOFFs.
- SEUs injected into the error flip-flops to corrupt the TRD-based parts in SETTOFFs.
- SEUs injected into the TD-checkers to corrupt the self-checkers in the self-checking registers.
- SEUs injected into the backup registers for the RF and flag register.

Three programs, quicksort, tak and a matrix multiplication program, were used for the fault simulation. The evaluation results are compared with the reliability results of an unprotected OpenRISC processor. Table VI shows the number of transient faults injected into each processor and the visible errors occurring in the data memories after running the programs. The same 4050 faults (including both SETs and SEUs) were injected into the original cells of each of the two processors. In order to carry out a comprehensive fault analysis and reliability evaluation, the locations of these faults cover all the vulnerable registers (the ones identified in Section IV-A) inside the pipeline. Also, the occurrence times of the injected faults are randomly distributed across the entire clock cycle, and both SETs and SEUs were simulated for each bit of the vulnerable registers. All the injected faults incur 306, 305, and 354 errors in the original processor for the quicksort, tak, and matrix multiplication programs, respectively. All these errors however, are mitigated in the protected OpenRISC processor. Specifically, 287 errors of the 306 errors occurring in the quicksort program were recovered on the fly, while the rest (19 errors) were recovered by replay operations. Of the 305 errors occurring in the tak program, 290 were recovered on the fly and 15 errors were recovered through the architectural replay. 344 of the 354 error occurring in the matrix multiplication were recovered on the fly and the other 10 errors were recovered by the replay operations. In addition, the transient faults injected into the redundant circuitry in the protected processor were also tolerated and did not induce any soft errors into the outputs of the programs.

D. Comparison with Razor and SEM/STEM Pipeline Protection

Both the RazorII pipeline protection technique and our proposed pipeline protection technique utilize an architectural replay recovery operation, which costs extra operation cycles and induces an IPC overhead. However, in our pipeline protection technique, most of the SEUs were detected in the TD interval and are recovered on the fly. As the fault-injection results showed in Section V-C, 93.8%, 95.1%, and 97.2% of the injected faults were recovered on the fly for quicksort, tak, and matrix multiplication programs, respectively. The replay operation is only triggered at a fairly low rate, at an average of 4.6% for the three programs. This is because replay is only required for correcting the captured SETs or TEs during the TRD interval. RazorII uses a conventional replay recovery mechanism and all the errors are recovered through re-execution. This means that the RazorII technique would need to recover 100% of the detected errors by using the replay operation. Therefore, the proposed pipeline protection induces a much smaller IPC overhead than the conventional replay recovery mechanism used in RazorII.

On the other hand, RazorII and SEM/STEM pipeline protection techniques cannot protect the WB stage of the pipeline and the registers that store the architectural states of the system during the WB stage, such as the RF and the flag registers. This is because RazorII cannot efficiently address SEUs, and the error-tolerance overhead is too big for SEM/STEM to protect the RF. Instead, they typically use a stabilizer buffer or register before the WB stage to make the WB stage non-timing-critical. Therefore the TEs occurring in the WB stage are not considered. This also guarantees that no errors are forwarded into the RF during the recovery process since the RF does not have SET/TE tolerance capability (see [30] [11] [13]). However, SETs occurring during the WB write cycle can still corrupt the registers that are updated during the cycle. The proposed pipeline protection eliminates this extra stabilizer and shrinks the pipeline depth compared with the previous pipeline protection techniques. Both the SETs occurring during the WB write cycle, and the SEUs occurring in the registers updated during the WB write cycle are tolerated by the incorporated proposed registers.

In terms of implementation overheads, the results in our previous paper show that a single SETTOFF has less or similar area, performance, and power consumption overheads compared to other radiation hardened techniques [14]. Specifically, the power overhead of SETTOFF is 28%, which is comparable to RazorII (28.5% to 30%). The replay recovery

TABLE VII: The SEU-tolerant capabilities of the self-checking register-protected RF and the SEC-DED-protected RF (D: Detection. C: Correction).

No. of SEU-induced MBUs	1 bit	2 bits	more than 2 bits
SEC-DED-protected RF	D + C	D	N/A
Self-checking RF	D + C	D + C	D + C

feature implemented in the SETTOFF-protected pipeline is similar to that used for Razor pipeline protection, therefore the implementation overheads incurred by the two replay features should also be comparable (although the replay feature in SETTOFF technique is much less frequently triggered since most faults are recovered on the fly). Razor used ECC for protecting the RF, therefore the power consumption overhead of the RF will be significantly larger than that of the SETTOFF-based pipeline protection, but the area overhead in Razor RF could be smaller. (See Section V-E for detailed RF overhead numbers.) The overall pipeline protection overheads of the two techniques should be comparable based on these observations. The SEM/STEM pipeline protection technique induces a much bigger overhead, similar to TMR.

E. Comparison with ECC-based RF protection

It has been demonstrated that the our radiation hardened register can efficiently protect the RF. The most commonly used ECC is SEC-DED coding which can correct single bit errors and detect double errors. Although a later SEC-DED coding technique, [31], improved the multiple-bit error-tolerance capability, it requires a large number of parity bits and therefore induces high area and power overheads. Compared with the conventional SEC-DED RF protection technique, the proposed RF protection significantly improves the MBU-tolerance capability since each bit inside the protected RF has its own built-in error-tolerant circuitry. In addition, previous research indicates that the majority of MBUs occurring in the RF are caused by the captured SETs originating in the combinational gates (such as the read and write logic) in the RF [32]. This is because the combinational logic has a high degree of fanout, and the majority of the cell area within the RF is consumed by the read and write logic. As a result, the proposed pipeline technique can provide a noticeably better error-tolerance than ECC, since ECC is not SET-tolerant.

Table VII summaries the error-tolerance capability of the 2 register files. The SETTOFF technique can provide better MBU-tolerance than the SEC-DED coding technique.

TABLE VIII: Implementation overheads of the robust RFs in 65nm.

Overheads (%)	Area	Power	Performance
Proposed RF	78%	46%	15.2%
SEC-DED-protected RF	30%	90%	470%

The implementation overheads of the two techniques for protecting the 32×32-bit RF in the OpenRISC are summarized in Table VIII. The ECC-protected RF is implemented with the SEC-DED code, which can correct single-bit errors and detect double-bit errors. Assuming a 10% activity rate for each bit, the SEC-DED-protected RF incurs an average 90% power overhead during a write cycle, while the proposed register consumes an average of 46% extra power. The self-checking RF incurs a delay overhead of 15.2% due to the extra loads added for error-tolerance. The delay overhead of the SEC-DED-protected RF is big due to the large decoding block at the read port, therefore extra cycles may be required to reload the register for error correction. The area overheads of the two robust RFs are estimated based on the number of transistors. For protecting a conventional 32×32-bit RF that is constructed with 11618 equivalent NAND-gates, the SEC-DED technique requires 3442 extra equivalent NAND-gates whereas the SETTOFF technique requires 9106 extra equivalent NAND-gates. It should be noted that these results are derived for an RF with only 1 read port. The area and power overhead of the SEC-DED technique will increase significantly for protecting an RF with multiple read ports.

VI. Conclusions and Future Work

In this paper, we present a complete pipeline protection mechanism realized on an OpenRISC microprocessor. The pipeline protection is achieved by incorporating SETTOFF-based self-checking cells into the most vulnerable sequential cells of the pipeline. A pipeline replay recovery mechanism is also developed at the architectural level to recover the errors detected by the hardened cells. The entire pipeline is protected, both SETs and SEUs occurring in each stage of the pipeline are detected by fault-tolerant cells in the corresponding stages. The proposed robust OpenRISC microprocessor was implemented in 65nm technology for evaluation. A cell-level transient fault injection and simulation technique was used to automatically inject SETs and SEUs into different parts of the pipeline and to record errors caused by the injected faults. The fault simulation results show that the proposed processor pipeline is robust to both SEUs and SETs occurring in different pipeline stages. The overheads of proposed technique for protecting the pipeline registers are smaller than or comparable to the previous low-cost techniques, while the power and performance overhead for protecting the RF is noticeably smaller than conventional ECC. Future work will focus on developing reliable systems which can satisfy both aggressive power and performance requirements.

References

[1] M. Favalli and C. Metra, "TMR voting in the presence of crosstalk faults at the voter inputs," *Reliability, IEEE Transactions on*, vol. 53, no. 3, pp. 342 – 348, Sept. 2004.

[2] P. Montesinos, W. Liu, and J. Torrellas, "Using register lifetime predictions to protect register files against soft errors," in *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, 2007, pp. 286–296.

[3] T. Slegel, I. Averill, R.M., M. Check, B. Giamei, B. Krumm, C. Krygowski, W. Li, J. Liptay, J. MacDougall, T. McPherson, J. Navarro, E. Schwarz, K. Shum, and C. Webb, "IBM's S/390 G5 microprocessor design," *Micro, IEEE*, vol. 19, no. 2, pp. 12–23, 1999.

[4] Y. Lin and M. Zwolinski, "A Cost-Efficient Self-Checking Register Architecture of Radiation Hardened Designs," in *International Symposium on Circuits and Systems (ISCAS)*, 2014.

[5] N. Seifert, B. Gill, V. Zia, M. Zhang, and V. Ambrose, "On the Scalability of Redundancy based SER Mitigation Schemes," in *Integrated Circuit Design and Technology, 2007. ICICDT '07. IEEE International Conference on*, 2007, pp. 1–9.

[6] S. Buchner, M. Baze, D. Brown, D. McMorrow, and J. Melinger, “Comparison of error rates in combinational and sequential logic,” *Nuclear Science, IEEE Transactions on*, vol. 44, no. 6, pp. 2209 –2216, Dec. 1997.

[7] T. Calin, M. Nicolaidis, and R. Velazco, “Upset hardened memory design for submicron cmos technology,” *Nuclear Science, IEEE Transactions on*, vol. 43, no. 6, pp. 2874 –2878, Dec. 1996.

[8] S. Whitaker, J. Canaris, and K. Liu, “Seu hardened memory cells for a ccsds reed-solomon encoder,” *Nuclear Science, IEEE Transactions on*, vol. 38, no. 6, pp. 1471–1477, 1991.

[9] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, “Robust system design with built-in soft-error resilience,” *Computer*, vol. 38, no. 2, pp. 43–52, 2005.

[10] M. Fazeli, S.-G. Miremadi, A. Ejlali, and A. Patooghy, “Low energy single event upset/single event transient-tolerant latch for deep submicron technologies,” *Computers Digital Techniques, IET*, vol. 3, no. 3, pp. 289–303, 2009.

[11] N. Avirneni and A. Somani, “Low overhead soft error mitigation techniques for high-performance and aggressive designs,” *Computers, IEEE Transactions on*, vol. 61, no. 4, pp. 488 –501, Apr. 2012.

[12] L. Anghel and M. Nicolaidis, “Cost reduction and evaluation of a temporary faults detecting technique,” in *Design, Automation and Test in Europe Conference and Exhibition. Proceedings*, 2000, pp. 591 –598.

[13] S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D. Bull, and D. Blaauw, “RazorII: In situ error detection and correction for PVT and SER tolerance,” *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 1, pp. 32 –48, Jan. 2009.

[14] Y. Lin, M. Zwolinski, and B. Halak, “A Low-Cost Radiation Hardened Flip-Flop,” in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014.

[15] S. Lin, Y.-B. Kim, and F. Lombardi, “Design and performance evaluation of radiation hardened latches for nanoscale cmos,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 7, pp. 1315–1319, July 2011.

[16] R. Garg, N. Jayakumar, S. Khatri, and G. Choi, “Circuit-level design approaches for radiation-hard digital electronics,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 6, pp. 781–792, June 2009.

[17] K.-C. Wu and D. Marculescu, “A low-cost, systematic methodology for soft error robustness of logic circuits,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 2, pp. 367–379, Feb 2013.

[18] D. Alnajjar, H. Konoura, Y. Ko, Y. Mitsuyama, M. Hashimoto, and T. Onoye, “Implementing flexible reliability in a coarse-grained reconfigurable architecture,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 12, pp. 2165–2178, Dec 2013.

[19] H.-M. Chou, M.-Y. Hsiao, Y.-C. Chen, K.-H. Yang, J. Tsao, C.-L. Lung, S.-C. Chang, W.-B. Jone, and T.-F. Chen, “Soft-error-tolerant design methodology for balancing performance, power, and reliability,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[20] G. Yan, Y. Han, and X. Li, “Svfd: A versatile online fault detection scheme via checking of stability violation,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 9, pp. 1627–1640, Sept 2011.

[21] C.-H. Chen, D. Blaauw, D. Sylvester, and Z. Zhang, “Design and evaluation of confidence-driven error-resilient systems,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 22, no. 8, pp. 1727–1737, Aug 2014.

[22] M. Fazeli, A. Namazi, and S. Miremadi, “Robust register caching: An energy-efficient circuit-level technique to combat soft errors in embedded processors,” *Device and Materials Reliability, IEEE Transactions on*, vol. 10, no. 2, pp. 208 –221, June 2010.

[23] M. Nicolaidis, “Time redundancy based soft-error tolerance to rescue nanometer technologies,” in *VLSI Test Symposium, 1999. Proceedings. 17th IEEE*, 1999, pp. 86 –94.

[24] OpenCores, “OpenRISC 1000 architecture manual,” Apr. 2006. [Online]. Available: http://opencores.org/openrisc,overview

[25] N. Mehdizadeh, M. Shokrolah-Shirazi, and S. Miremadi, “Analyzing fault effects in the 32-bit openrisc 1200 microprocessor,” in *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, Mar. 2008, pp. 648 –652.

[26] M. Ebrahimi, A. Evans, M. Tahoori, R. Seyyedi, E. Costenaro, and D. Alexandrescu, “Comprehensive analysis of alpha and neutron particle-induced soft errors in an embedded processor at nanoscales,” in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, March 2014, pp. 1–6.

[27] M. Zwolinski, “A technique for transparent fault injection and simulation,” *Microelectronics Reliability*, pp. 797–804, 2000.

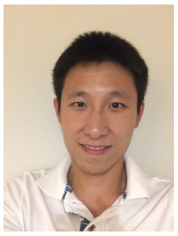
[28] J. Baxter, “ORPSoC - OpenRISC Reference Platform SoC.” [Online]. Available: http://opencores.org/openrisc,orpsocv2

[29] R. Harada, Y. Mitsuyama, M. Hashimoto, and T. Onoye, “Measurement circuits for acquiring set pulsewidth distribution with sub-fo1-inverter-delay resolution,” in *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, 2010, pp. 839–844.

[30] 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 *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, Dec. 2003, pp. 7 – 18.

[31] P. Reviriego, S. Pontarelli, A. Evans, and J. Maestro, “A class of sec-ded-daec codes derived from orthogonal latin square codes,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 23, no. 5, pp. 968–972, May 2015.

[32] J. A. Blome, S. Gupta, S. Feng, and S. Mahlke, “Cost-efficient soft error protection for embedded microprocessors,” in *Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, ser. CASES ’06. ACM, 2006, pp. 421–431.



Yang Lin received his B.S. degree in electronic information science and technology from Jilin University, China, in 2009, and MSc degree with distinction in System on Chip in 2010 from the University of Southampton, UK, where he has been working towards a Ph.D. degree since October 2010. His research interests include fault-tolerant design techniques and methodologies.

Mark Zwolinski received the B.Sc. degree in electronic engineering and the Ph.D. degree in electronics from University of Southampton, Southampton, U.K., in 1982 and 1986, respectively. He is currently a Professor in the School of Electronics and Computer Science, University of Southampton. He has authored two textbooks and has co-authored a third. He has written over 180 papers in the areas of EDA and test. His current research interests include high-level synthesis, fault tolerance, and behavioral modeling and simulation. Dr. Zwolinski is a Fellow of IET and BCS and Senior Member of IEEE and ACM.

Basel Halak received his B.S. degree from the school of Electronics Engineering in Damascus University, Syria in 2001, and his MSc and PhD degrees in Microelectronics system design from Newcastle University, UK in 2005 and 2009, respectively. In 2011, he joined the University of Southampton, UK, where he is currently pursuing his research interests in dependable systems on a chip, fault tolerance techniques, and VLSI circuits for communications. He has published a monograph and more than 25 refereed papers.