Generating Control Logic for Optimized Soft Error Resilience

Wenchao Li
UC Berkeley
liwenchaoapollo@gmail.com

Susmit Jha
Intel
susmit.kumar.jha@gmail.com

Sanjit A. Seshia
UC Berkeley
sseshia@eecs.berkeley.edu

***Abstract*—Aggressive technology scaling has necessitated the development of techniques to ensure resilience to device faults, including soft errors, circuit wear-out, variability, and environmental effects. All error resilience techniques employ some form of redundancy, resulting in added cost such as area or power overhead. Existing selective hardening techniques have been focused on identifying on the most vulnerable components and then statically harden them. This paper proposes a new technique that can further reduce this overhead for error resilience mechanisms that are controllable. The key idea is to generate control predicates[1] that can turn the resilience mechanisms ON and OFF dynamically, at the right time. These predicates are mined using an optimization formulation that leverages fault injection simulation information. Our experimental results demonstrate that our approach significantly outperforms the static hardening approach for optimizing soft error resilience.**

I. Introduction

Technology scaling to 65nm and below has caused reliability problems to become a dominant design challenge. There is a need to make circuits resilient to a wide range of physical defects ranging from soft (transient) errors, aging and wearout, environmental and device parameter variations, and aggressive deployment to reduce power and increase performance (see, e.g., [1]). Fortunately, in recent years there have been several efforts in this direction, including techniques for hardening circuits against soft error (e.g., [2][3]) and methods for mitigating the effects of circuit aging (e.g., [4][5]). Error detection, recovery and retry mechanisms have also been studied and implemented extensively in modern microprocessors such as in the IBM POWER6 [6][7].

Every error resilience mechanism employs some *redundancy*, incurring cost in the form of increased area and power, and possibly reduced performance. Thus, design today is a process of achieving a trade-off between performance, power, area, and reliability. With power becoming an extremely important design consideration, error resilience circuitry must be inserted or enabled judiciously.

For some types of faults, such as soft errors, it is possible to selectively insert or enable error-resilience mechanisms. Verification techniques such as model checking can then be used to identify only those components that must be protected for the circuit to satisfy its specification. As an example, Seshia et al. [8] showed that for an implementation of the European Space Agency SpaceWire protocol, less than 25% of the flip-flops needed to be protected against soft errors using Intel's built-in soft error resilience (BISER) [3], reducing the power overhead of employing the error resilient mechanism by a factor of 4.35. Their approach provides a binary classification of when to use a BISER flip-flop versus when to use a standard flip-flop: a flip-flop is protected if there is a single input sequence and a single cycle at which the fault occurs that causes the specification to fail. However, for many designs, such a binary classification might indicate that most of the flip-flops must be protected, and it also ignores the fact that faults at certain cycles lead to failures whereas at other cycles they might fail to propagate.

Simulation-based techniques on the other hand aim to identify components that are the most vulnerable statistically. Previous work [9][10][11][12][13] has shown that circuit components exhibit vulnerability non-uniformly. One example is that soft errors in the branch prediction unit of a microprocessor may impact latency but do not affect correctness. Hence, one can selectively harden the circuit components by prioritizing the protection on components that have the highest resilience-gain (e.g. soft error rate (SER) reduction) to cost ratio. In this paper, we refer to such techniques as *static hardening*.

Some error-resilience circuitry, such as BISER [3], can be turned `ON` and `OFF` at runtime, thus providing a dynamic mechanism for reducing power overhead. Hence, if we can identify conditions under which the circuit components are vulnerable, we can turn on the resilience circuitry *only when it is needed*. For example, in a multi-core design, we can save power by turning off the recovery unit of a core (such as the one in the IBM POWER6) if we know it is idle or performing non-critical computation. In general, the difficulty is knowing *when* to protect a circuit component.

The present paper addresses this problem. We give a mathematical framework to synthesize low-cost error-resilient circuits, where the error-resilience mechanisms are controlled dynamically. In particular, we synthesize control logic (*predicates*) that are used to turn error resilience mechanisms `ON` and `OFF` *online* in an optimal way. In Section V, we show how these predicates can be generated from simulation traces using an optimization formulation as a *0-1 integer linear program*. The traces are based on performing fault-injection experiments for the type of fault (e.g., soft errors) that we target.

The second author started this work while at UC Berkeley.

[1]In this paper, predicates are Boolean formulas over the set of signals in a circuit. Hence, the predicates can be synthesized together with the original circuit into a gate-level netlist.

We make the following novel contributions in this paper:

- We formally define the resilience control synthesis problem for digital circuits, and propose an optimization framework that maximizes circuit resilience for a given power budget.
- We prove that if we can control the error-resilience mechanisms online, our dynamic approach is guaranteed to provide resilience that is at least as good as that can be provided by the static hardening technique for the same power overhead.
- An experimental evaluation on a chip multiprocessor router [14] shows that this approach provides a systematic way to control soft error-resilience using BISER [3].

The rest of this paper is organized as follows. We discuss related work in Section II. Section III gives an overview of the proposed method. Our formal model is presented in Section IV. The problem definition and our approach are described in Section V. Two case studies are presented in Section VI, and conclusions in Section VII.

II. Related Work

The problem of providing error-resilience either to a circuit or to an entire system has been studied extensively in the literature. Resilience is typically measured for a specific fault model under a specific correctness criterion, such as output correctness or with respect to a formal specification (e.g. an assertion expressed as a logical formula). Increasingly transient faults such as soft errors are becoming a concern due to aggressive technology scaling; for example, researchers have shown that soft errors can contribute significantly to system-level silent data corruption [15].

Related work on automated insertion of error-resilience mechanisms can be classified into three dimensions depending on the nature of the mechanism used for error resilience, the technique used for vulnerability analysis; and the method used to select locations for inserting error-resilience mechanisms.

The mechanism used for error resilience can be static or dynamic. Static mechanisms such as gate sizing [16][17], redundant circuit for voting [18] or resilient flip-flops [19] can not be switched on and off at runtime. The only choice is to either harden a component or not, at design time. In contrast, dynamic mechanisms such as BISER [3] and recovery unit in IBM POWER6 [7] are controllable and can be switched on and off at runtime. The dynamic mechanisms provide a more fine-grained control over resilience mechanisms. While costs such as area do not depend on whether the resilience mechanism is selectively switched on or always stays on, power cost can be lowered by using dynamic mechanisms and a suitable control logic to switch them on and off. We point the readers to a recent survey [20] for more details on relience techniques. Our approach is targeted towards *dynamic* error mechanisms.

Vulnerability analysis is the first step in making a design resilient. Traditionally, statistical fault injection [21][9][11][12][13] and fault-free simulation [10] have been used to evaluate vulnerability of different circuit components to soft errors. Miskov-Zivanov and Marculescu [17] proposed the use of Markov chains to model sequential system behavior and evaluate system level failure rate as the steady-state probability of output non-equivalence. Formal methods have also been used to identify components that are vulnerable to soft errors with respect to some high-level formal specifications [8]. More recently, a technique based on combination of simulations and formal methods was proposed to estimate a probabilistic measure of the vulnerability of each flip-flop with respect to a formal specifications [22]. In our work, we adopt this approach to estimate the vulnerability factors of different circuit components with respect to a fault model.

Once the vulnerable components in the design have been identified, the next step is to choose the components to be protected using error resilience. This choice is guided by some resource constraint such as area and power which prohibit protecting all vulnerable components. For approaches that *statically* select a resilience mechanism [9][11][12][13], the selection of components can be done greedily to fit the resource budget. Hence, the goal of these approaches is to come up with a vulnerability ranking which can guide the greedy selection for any budget. Since we consider dynamic error resilience mechanisms, we also need to choose *when* to protect a component in addition to *what* to protect. Our approach uses a novel integer programming based formulation to select (a) the components to be protected, and (b) the corresponding control logic for each component to switch the resilience mechanism on and off.

There has also been work on leveraging validation test data to generate control logic to ensure the functional correctness of a digital IC. Notably, a signature-based approach is proposed in [23]. In this framework called "Semantic Guardian", a subset of important control signals are monitored at runtime, and when an untrusted configuration of these signals is encountered, a recovery controller switches the circuit to a safe mode. The safe mode is a stripped down and lower performance mode of operation which has been formally verified. Since both their technique and ours use validation tests to extract conditions for the control logic, we face similar challenges such as the exhaustiveness of tests and the large number of conditions to select from. There are also several aspects that our work differs from theirs. First, soft errors are transient physical upsets instead of functional errors. Hence, to imitate the "Semantic Guardian" approach in the soft error context, an extra soft error hardened copy of the circuit is required, which incurs significant area and power overhead. Second, the control conditions we identify are the ones that correspond to latch soft error vulnerability, as opposed to untrusted executions. Lastly, our approach works for generic digital ICs where the control conditions can be arbitrary and are less understood than those in microprocessors.

In our empirical studies, we focus on providing resilience against *soft errors* occurring at flip-flops by using BISER. However, we note that our optimization framework applies to other faults and their respective fault-tolerant mechanisms as long as the mechanisms can be turned `ON` and `OFF` at runtime.

III. Overview

Our optimization flow is illustrated in Figure 1.

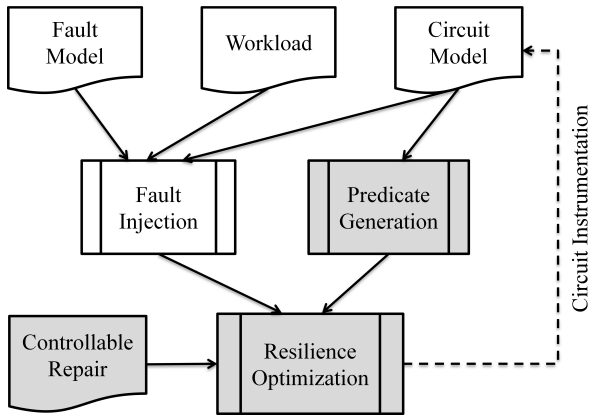

Fig. 1. Dynamic Control Optimization Flow

Given a fault model, a circuit and its workload, we first perform fault injection experiments to obtain a set of simulation traces. From the circuit model, we also generate a set of predicates that are potentially useful for controlling the repair mechanisms. The resilience optimization takes in as input the simulation traces, the set of generated predicates and the controllable repair, and outputs a mapping of the repairs to circuit components along with the additional logic for controlling them. The output is optimal in the sense that the error resilience is maximized for a given budget constraint (e.g. power overhead). Circuit instrumentation can then be performed for final synthesis. The added control logic allows us to take advantage of conditions at which the computation is non-critical and hence protect the corresponding components only when needed.

Figure 2 illustrates this concept. Suppose the circuit is

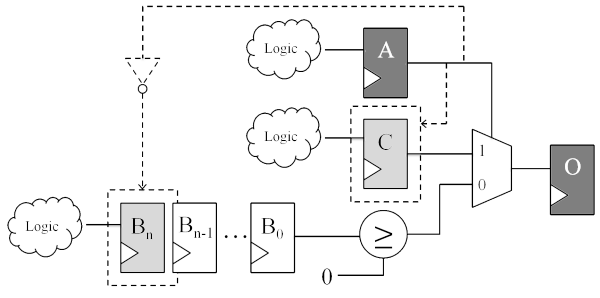

Fig. 2. An Example of Using Predicate to Control Error Resilient Mechanism

subject to a single event upset (SEU) at the flip-flops, and we are provided with protection mechanisms that can be used to guard the flip-flops against SEU. The mechanism can be turned `ON` online by setting a controlling input to `True` and `OFF` otherwise. For correctness, we only care about the value of flip-flop O. B_n is the sign bit of the bit vector B. To protect the circuit from SEU (which is a transient error), we have to protect A and O at all cycles. On the other hand, we do not have to protect the data bits of B $(B_{n-1}, \ldots, B_0)$ since we only need B_n to determine if $B \geq 0$. Whether we need to turn `ON` the protection on C and B_n depends on the current value of A. The dashed lines and the "NOT" gate represent the extra wires and control logic that need to be inserted so that the configurable protection mechanisms (shown as dashed boxes) are turn `ON`*only when it is needed*. Hence, with the addition of an inverter, we are able to reduce the dynamic power consumption of the protection mechanism deployed at these two flip-flops (by 50%) and still achieve protection of all the flip-flops from SEU. Our key proposition in this paper is that, even though the additional logic gates consume power, the cost saving of turning `OFF` the protection mechanism when they are not needed can offset the extra cost and achieve an *overall* power reduction. In the following sections, we will show how this logic can be automatically generated for arbitrary circuits via an optimization formulation.

IV. Formal Models

We describe in this section the formal model and the terminologies used in the remainder of the paper.

A. Circuit Model

A sequential circuit C_s is formally modeled as a tuple $\langle I, O, L, \delta, \rho, \theta \rangle$, where I is the set of input signals, O is the set of output signals, L is the set of state variables (flip-flops) that induce the state space 2^L, $\delta : 2^I \times 2^L \rightarrow 2^L$ describes the transition relation of C_s, ρ is the output function, and θ describes the initial state of the circuit. For each $l \in L$, there is a next-state assignment $next(l) := f_l(I, L)$, where f_l is a function determining the next-state value of l in terms of the current-state values of all flip-flops in L and the inputs.

We denote the value of flip-flop l at cycle j by l^j. Similarly, the vector of values of flip-flops in set L at cycle j is denoted by L^j, and the vector of values of inputs is denoted by I^j.

B. Fault Model

For concreteness, we focus in this paper on transient bit flips in L. However, the methodology proposed herein can also be applied to other faults that have controllable error-resilience mechanisms.

A single event upset (SEU) is a transient fault that causes the value of a flip-flop to flip. [2] It is parameterized by flip-flop l and cycle j. In other words, when an SEU occurs in cycle j, the value of flip-flop l in cycle $j+1$, l^{j+1}, is the opposite of what it is supposed to be. Formally, $l^{j+1} := \overline{f_l(I^j, L^j)}$.

A fault injection experiment is performed by choosing a set of flip-flop and cycle pairs $\{(l, j)\}$ to inject bit flips, and then simulate the modified circuit with an input sequence. We assume that we can label the resulting trace with `good` or `bad`, with respect to a pre-defined correctness criterion. This typically can be done by checking sequential equivalence of the faulty circuit with the fault-free circuit, or using end-to-end monitors to verify correctness.

[2] In this work, we consider only single bit upsets (SBUs) that are caused by SEUs.

We also assume that we are provided with a set of input sequences (workload) which defines all input behavior that the designer is concerned with. Correctness of the circuit is defined as one that maintains the correctness criterion (as discussed above) for these input sequences. These are also the input sequences that we use in fault injection experiments.

C. Repair Model

We consider fault-tolerance (error-resilience) techniques that are *controllable*, meaning that they can be turned ON and OFF dynamically, during circuit operation. Moreover, in this paper, we focus on *local repairs*, which can be applied to individual circuit components, such as individual flip-flops. For example, for an SEU, a built-in soft error resilience (BISER) [3] flip-flop can operate in an "economy" mode which roughly halves the power consumption of a BISER flip-flop, bringing the power consumption down to that of a regular non-BISER flip-flop. The technique is based on re-using the scan portion of a flip-flop. Details of the design can be found in [3]. The "economy" mode can be enabled by turning off the scan portion with appropriate values of the control signals.

Formally, consider a controllable, local repair r that negates the effect of an SEU in a flip-flop. The repair r is parameterized by the flip-flop l and a Boolean control signal b. This means that we can set r to ON by setting $b = \texttt{True}$ or OFF by setting $b = \texttt{False}$ at any cycle. If r is ON at cycle j, then a SEU occurring also at cycle j will not have any effect on the functional behavior of the circuit. That is, a bad trace will be relabelled as good if this repair is applied.

The goal of this paper is to synthesize, for each flip-flop l, whether to allocate a local repair r to l as well as the combinational circuit whose output is the control signal b of r. We describe our approach in detail in the following section.

V. Resilience Control Synthesis

A. Notation

We introduce notations to model key aspects of control predicates and repairs that will be central to our formulation of the overall optimization problem.

Let the Boolean variable $b_{l,p} = 1$ if and only if the local repair r of flip-flop l is controlled by predicate p. In other words, if $b_{l,p} = 1$, it means that when p is evaluated to True, r of flip-flop l is ON at the same cycle, (so that a SEU at l will not be flip-floped at the next cycle). Otherwise r is OFF. If r is controlled by True, it is then ON all the time.

We denote the data activity factor (the average number of output transitions per clock cycle) of a flip-flop l as ρ_l. Each repair comes with an associated cost. We use $c_{l,\texttt{ON}}$ to denote the power consumed by flip-flop l when the repair at l is turned ON, assuming $\rho_l = 1$. Similarly, $c_{l,\texttt{OFF}}$ to used to denote the power consumption when the repair at l is turned OFF assuming $\rho_l = 1$. Hence, we approximate the power consumption of a protected flip-flop by $c_{l,\texttt{ON}} \times \rho_l$ and that of an unprotected flip-flop by $c_{l,\texttt{OFF}} \times \rho_l$ (given that the resilience mechanism is already deployed at l).

The *predicates* P is a set of Boolean formulas over L. For each predicate $p \in P$, $p_{\tau,j}$ denotes the valuation of p on trace τ (of length $|\tau|$) at cycle j. We further assume that each $p \in P$ can be characterized by α_p, which is the probability of p being True. Formally, given a set of traces Π, α_p is given by the following expression:

$$\alpha_p = \frac{1}{|\Pi|} \sum_{\tau \in \Pi} \left(\frac{1}{|\tau|} \sum_{1 \le j \le |\tau|} \beta_{p,\tau,j} \right)$$

where $\beta_{p,\tau,j}$ is 1 when $p_{\tau,j} = \texttt{True}$, and 0 otherwise. Section V-C discusses the generation of the predicate set P in more detail.

Each predicate has to be synthesized as a combinational circuit with inputs drawn from L. Therefore, each control predicate also has an associated cost c_p, which is the power drawn by the circuit that computes p. We include the predicate True with cost 0 in the set P (corresponding to always turning the repair ON).

Finally, we associate a *resilience measure* with each assignment of a flip-flop repair to a controlling predicate. Formally, the resilience measure $v_{l,p}$ is a measure of the improvement in error-resilience if flip-flop l is protected with controlling predicate p, i.e., if $b_{l,p} = 1$.

Several resilience measures are possible. In this paper, we use a specific measure $v_{l,p}$ defined as follows. Given a set of fault injection simulation containing good and bad traces, where a trace is bad if the fault results in falsifying the correctness criterion (e.g. failing a monitor), and good otherwise. We partition these traces into sets E_l for each flip-flop l because only one error at a single flip-flop is injected at each run according to the SEU model. In general, a set E_l will contain both good and bad traces. For each set E_l, we define $v_{l,p}$ to be the fraction of traces that will be relabeled to good from bad if $b_{l,p} = 1$. A bad trace is relabeled to good if p is True at the same cycle as the injection of SEU at flip-flop l (l is protected from the SEU by turning r of l ON). It is important to note that our optimization formulation is compatible with any resilience measure that depends on the value of predicate p at component l.

B. Problem Formulation

Our goal is to maximize the total gain in resilience subject to a power overhead budget. With this goal, the problem of resilience control synthesis using local repairs is described in detail below.

Resilience Control Synthesis Problem: Given the following inputs:

- a set of labeled fault injection runs Ω;
- a set of predicates P with associated power costs;
- a set of repairs R with associated power costs, and
- a power overhead budget Φ,

Generate a mapping M from the set of flip-flops L to the set of control predicates P, such that if $P^* \subseteq P$ is the range of the mapping M (the subset of chosen predicates), then:

(i) the total power overhead of P^* and R is less than Φ, and

(ii) the total resilience measure is maximized.

We formulate this optimization problem as a 0-1 integer linear program. The *objective function* is as given below:

$$maximize \sum_{p\in P}\sum_{l\in L} b_{l,p}v_{l,p} \tag{1}$$

The above objective function is optimized subject to the following linear *constraints*:

(A) *Total power overhead is less than* Φ*:* [3] [4]

$$\sum_{p\in P} d_p c_p + \sum_{l\in L}\sum_{p\in P} b_{l,p}(\alpha_p c_{l,\texttt{ON}}\rho_l + (1-\alpha_p)c_{l,\texttt{OFF}}\rho_l) \leq \Phi \tag{2}$$

where d_p is a binary variable denoting whether predicate p is used to control any flip-flop at all. For predicate p, $\sum_{l\in L} b_{l,p} > 0$ iff $d_p = 1$. This can be encoded as the following linear constraints.

$$\forall p \in P, \sum_{l\in L} b_{l,p} \leq d_p \times |L| \tag{3}$$

$$\forall p \in P, \; d_p \leq \sum_{l\in L} b_{l,p} \tag{4}$$

(B) *Each local repair is controlled by at most one predicate:* If a local repair is not controlled by a predicate, then it incurs no cost, i.e. we do not instrument the circuit with this repair. However, one predicate may control multiple repairs.

$$\forall l \in L, \sum_{p\in P} b_{l,p} \leq 1 \tag{5}$$

P^* is then the set $\{p \in P | b^*_{l,p} = 1\}$, where $\{b^*_{l,p}\}$ is the optimal assignment of $\{b_{l,p}\}$ to the above optimization problem. Hence, P^* together with $\{b^*_{l,p}\}$ give us the control logic for SEU resilience using repairs R.

The number of binary variables in the 0-1 integer linear program is $|P|\times|L|+|L|$. We can assume that we have already identified the set of flip-flops that are inherently resilient (i.e. no need to protect these flip-flops), using techniques such as the ones described in [8] and [7], and hence reduce the size of L in the optimization problem. We discuss the generation of P in more detail in Section V-C.

Existing static hardening techniques work by ranking components according to their vulnerability estimates and then prioritizing repair for the more vulnerable ones until the cost constraint cannot be met. In fact, if the cost of repair is non-uniform, it becomes a 0-1 knapsack problem (if the regularization term is 0 in Equation (1)). The objective function of the static problem takes the form of Equation (1) with the following constraints

$$\sum_{l\in L} b_{l,\texttt{True}} c_{l,\texttt{ON}} \rho_l \leq \Phi \tag{6}$$

[3]Logic sharing between predicates and the original circuit is not explored in this work. A re-synthesis step can be taken after this optimization to further lower the power overhead

[4]Similarly , one can add an area overhead constraint. However, this is less interesting since area is a fixed cost.

As formalized in the theorem below, our proposed technique is provably better than a corresponding static hardening approach.

Theorem 1: For the same power budget, the resilience measure produced by our optimization framework (1) is at least as large as that produced by the static hardening approach with the constraint given by Equation (6).

Proof: The result follows from the fact that the optimal solution of the static problem is a feasible solution to the dynamic problem (1) with constraints (2)-(5). Let $\{b^*_{l,\texttt{True}}\}$ be the optimal solution of the static problem. In the dynamic problem, Since the predicate `True` has corresponding $c_p = 0$ and $\alpha_p = 1$, constraint (2) can be reduced to $\sum_{l\in L} b_{l,\texttt{True}} c_{l,\texttt{ON}} \leq \Phi$, which is clearly satisfied by $\{b^*_{l,\texttt{True}}\}$ due to (6). For any $p \in P \setminus \{\texttt{True}\}$, let $b_{l,p} = 0$. Then constraint (5) is satisfied. If $\sum_{l\in L} b_{l,\texttt{True}} > 0$, let $d_p = 1$ if p is `True` and $d_p = 0$ otherwise. Then constraints (3) and (4) are satisifed. If $\sum_{l\in L} b_{l,\texttt{True}} = 0$, let $d_p = 0$ for all $p \in P$. Then constraints (3) and (4) are also satisfied. Hence, $\{b^*_{l,\texttt{True}}\}$ is also a feasible solution to the dynamic problem. ■

C. Predicate Generation

An important piece of this optimization is the set of predicates P from which we can synthesize control logic from. In general, P can be either manually written using designer's insight or automatically generated by predicate mining tools.

While the effectiveness of our framework depends on the quality of the predicates that we use to control the local repairs, generating good predicates is orthogonal to the optimization problem. One consideration when generating P is that c_p has to be reasonably small since otherwise the additional cost of p will exceed the power saving of using p to set some repairs to `OFF`. For example, one can use existing logic in the circuit as candidates for the set P since their cost is essential 0 if additional wiring is small. In this paper, we use predicates that are in the form of a Boolean conjunction of two literals, where each literal is either the value of a flip-flop on the control path or its negation. Formally, $P = \{p | p = lit_1 \wedge lit_2\}$, where $lit_i = l$ or $lit_i = \neg l$, for $l \in L^*$. L^* is the set of flip-flops on the control path. With additional information of a synthesized circuit such as placement and routing, one can construct the predicates only from the nearby signals of a flip-flop so that additional wiring is negligible. Such a heuristic can reduce the number of variables $b_{l,p}$ in the optimization problem significantly while incurring only a small penalty on the objective. We choose to use simple predicates in this paper because the power consumption of an `AND` gate is comparable to that of the local repair which we use in our experiments. The optimization only makes sense if the additional control logic consumes relatively a small amount of power as compared with the resilience logic. In Section VI, we show that even with these simple predicates, our approach still significantly outperforms the static hardening approach.

VI. Case Study

To evaluate the effectiveness of our approach, we did a thorough case study on a chip multiprocessor router design. We focus on providing resilience to the design against soft errors using BISER.

We compare our dynamic approach with the static hardening technique described in V-B. We show that for the same power budget, the dynamic approach always provided better resilience than the static approach. Moreover, the dynamic approach reached reliability goals at much lower costs than the static approach. In the result plot, 100% power overhead of providing resilience corresponds to instrumenting every flip-flop with BISER and each BISER is set to `ON` all the time. By 100% resilience gain, we mean that every `bad` trace in the set of fault injection experiments gets re-labelled to `good` when the error-resilience mechanisms were applied.

CMP Router: The overview of the chip multiprocessor (CMP) router is illustrated in Figure 3. It is a simplified version of the design presented in [14]. It is a composition of four high-level modules. The *input controller* comprises a set of FIFOs buffering incoming *flits* and interacting with the *arbiter*. A *flit* is a flow control unit. A data packet is composed of multiple flits. Each *input controller* contains a circular FIFO buffer.When the arbiter grants access to a particular output port, a signal is sent to the input controller to release the flits from the buffers, and at the same time, an allocation signal is sent to the *encoder* which in turn configures the *crossbar* to route the flits to the appropriate output port. The synthesized router has a total of 166 flip-flops and 944 gates.

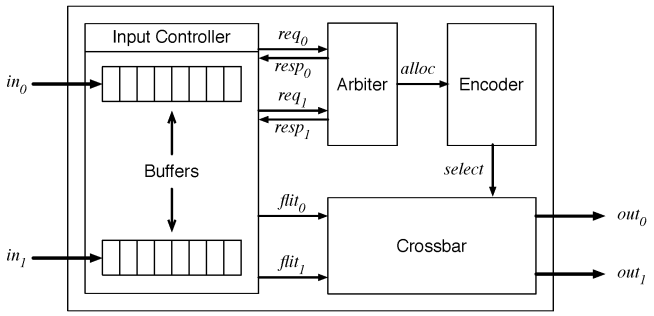

Fig. 3. CMP Router Comprising Four High-Level Modules

The CMP router was chosen because it is representative of on-chip interconnection networks with readily available system-level specifications. The functionality of the router is easy to state: it must correctly direct each of its input packets to the output port specified by the packet header within a specified number or cycles.

A balanced workload (same flit generation rate) for the two input ports was used to simulate the router in both the fault injection experiment and the fault-free case. In the fault injection experiment, for each flip-flop, we ran 300 simulations, each of 1000 cycles, and a SEU was injected at a random cycle during each simulation. In the fault-free case, we ran a single simulation of 3000 cycles. This trace was used to estimate α_p for each p and ρ_l for each l.

We used these parameters in the optimization problem:

- $\forall l \in L$, $c_{l,\texttt{ON}} = 5.72$ and $c_{l,\texttt{OFF}} = 0.68$, normalized to the power cost of a flip-flop (which is 4 at an activity factor of 1). These numbers were estimated from Table II in [3] where the power consumption was calculated assuming a data activity factor of 0.25.
- $c_p = 0.2$ if the predicate was a conjunction of two distinct literals. We assumed that the power cost of a `AND` gate was approximately 20% of that of a flip-flop. $c_p = 0$ if the predicate was a single literal. For simplication, We also assumed the power cost of additional wiring was 0.

IBM ILOG CPLEX Optimization Studio 12.3 [24] was used to solve the 0-1 integer linear programs (for both static and dynamic hardening). A summary of the runtime for solving these programs (i.e. with a particular power budget) is given in Table I. On average, solving the 0-1 ILP for dynamic hardening takes less than 8 minutes.

TABLE I
Solver runtime in seconds

	Longest	Shortest	Average
Static	0.16	0.01	0.1
Dynamic	1112.3	168.3	452.8

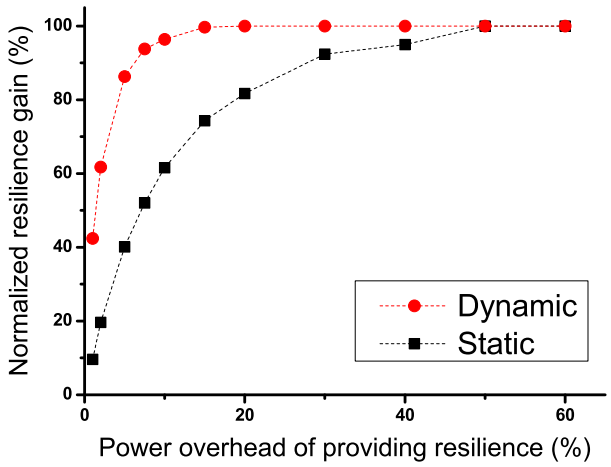

Fig. 4. CMP router: Resilience gain vs. Power overhead. In our experiment, we fix a power budget on the x-axis and then optimize for resilience gain.

In Figure 4, `Dynamic` is the solution to our optimization problem. We can see that `Dynamic` always gives a larger resilience gain than `Static`, as proven in Section V-B. For example, at 1% overhead, `Dynamic` achieves 42.4% resilience (a normalized measure on protection of the latches from SEU), which is a 4.4X improvement over `Static` which achieves only 9.6% resilience. In addition, `Dynamic` achieves 100% resilience at 15% power overhead, which is much lower than `Static` at 50%.

We also studied the resulting mapping of predicates to latches and highlight some of our findings below.

- 12 of the flip-flops were inherently resilient to soft errors in this router (no BISER was allocated to these flip-flops). These include the priority state of the arbiter. This is because a soft error at the priority bit only impacts latency but not correctness.
- Flip-flops in the FIFOs: BISERs for the data bits were controlled by predicates over the header bits. This is because a special header is used in the design to determine whether the data in the buffer is actually a flit. If the data is not a flit (e.g. initialized value in the buffer), then it does have to be protected. Our optimization approach was able to figure this out automatically.
- BISERs for the important control signals such as the header field in the buffer and pointers to the FIFO were controlled by the predicate `True` (always protected).
- BISER for the "full" signal in the FIFO was controlled by the value of the "dequeue" signal in the same FIFO. This was a subtle finding because the "full" indicator matters only when the FIFO is not being simultaneously dequeued.

In short, our approach generated predicates that were as good as what an user with expert knowledge of the design could produce. On the other hand, solving 0-1 ILP can be expensive for larger circuits. We envisage our techinque to be applied at the modular level, not at the full-chip level. This will sacrifice some potential power saving by ignoring predicates from other modules. However, using control predicates far away will incur additional wiring cost, which may cancel out the benefit of power-gating the resilience mechanisms. Our experiment, while preliminary, shows and confirms the theoretical result that the dynamic power-gating approach proposed in the paper significantly outperforms a static one.

VII. Conclusion

We have proposed a novel optimization framework that synthesizes logic to dynamically control local repairs such that the error resilience of the target circuit is maximized for any power overhead budget. The control logic is synthesized by first generating simple predicates and then mapping them to the local repairs via an optimization procedure. Our framework is general. It can be applied to other controllable local repairs and error models. In the future, we plan to evaluate the scalability of our approach on industrial-scale circuits and explore the applicability of our technique to cross-layer resilience.

Acknowldgement The authors acknowledge the support of the Gigascale Systems Research Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity. This work was also supported in part by a Hellman Family Faculty Fund Award, and by the NSF grant CNS-0644436.

References

[1] S. Y. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, Nov-Dec 2005.

[2] M. Nicolaidis, "Design for soft error mitigation," *IEEE Trans. Device and Matl. Reliability*, vol. 5, no. 3, pp. 405–418, Sept. 2005.

[3] M. Zhang, S. Mitra, T. M. Mak, N. Seifert, N. J. Wang, Q. Shi, K. S. Kim, N. R. Shanbhag, and S. J. Patel, "Sequential element design with built-in soft error resilience," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 14, no. 12, pp. 1368–1378, Dec. 2006.

[4] M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra, "Circuit failure prediction and its application to transistor aging," in *Proceedings of the 25th IEEE VLSI Test Symmposium*, ser. VTS '07, Washington, DC, USA, 2007, pp. 277–286.

[5] Y. Li, Y. M. Kim, E. Mintarno, D. S. Gardner, and S. Mitra, "Overcoming early-life failure and aging for robust systems," *IEEE Design & Test of Computers*, vol. 26, no. 6, pp. 28–39, 2009.

[6] M. J. Mack, W. M. Sauer, S. B. Swaney, and B. G. Mealey, "Ibm power6 reliability," *IBM Journal of Research and Development*, vol. 51, no. 6, pp. 763 –774, Nov. 2007.

[7] P. N. Sanda, J. W. Kellington, P. Kudva, R. Kalla, R. B. McBeth, J. Ackaret, R. Lockwood, J. Schumann, and C. R. Jones, "Soft-error resilience of the ibm power6 processor," *IBM Journal of Research and Development*, vol. 52, no. 3, pp. 275 –284, May 2008.

[8] S. A. Seshia, W. Li, and S. Mitra, "Verification-guided soft error resilience," in *DATE*, 2007, pp. 1442–1447.

[9] S. Kim and A. K. Somani, "Soft error sensitivity characterization for microprocessor dependability enhancement strategy," in *DSN, 2002*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 416–428.

[10] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *MICRO 36*. Washington, DC, USA: IEEE Computer Society, 2003, p. 29.

[11] N. J. Wang, J. Quek, T. M. Rafacz, and S. J. patel, "Characterizing the effects of transient faults on a high-performance processor pipeline," in *DSN, 2004*, Washington, DC, USA, 2004, p. 61.

[12] M. Valderas, M. Garci anda, C. Lo andpez, and L. Entrena, "Extensive seu impact analysis of a pic microprocessor for selective hardening," *Nuclear Science, IEEE Transactions on*, vol. 57, pp. 1986 –1991, Aug. 2010.

[13] M. Maniatakos and Y. Makris, "Workload-driven selective hardening of control state elements in modern microprocessors," in *VTS, 2010 28th*, April 2010, pp. 159 –164.

[14] L.-S. Peh, "Flow control and micro-architectural mechanisms for extending the performance of interconnection networks," Ph.D. dissertation, Stanford University, August 2001.

[15] R. Baumann, "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," pp. 329–332, 2002.

[16] R. R. Rao, D. Blaauw, and D. Sylvester, "Soft error reduction in combinational logic using gate resizing and flipflop selection," in *ICCAD, 2006*. New York, NY, USA: ACM, 2006, pp. 502–509.

[17] N. Miskov-Zivanov and D. Marculescu, "Modeling and optimization for soft-error reliability of sequential circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 27, no. 5, pp. 803–816, 2008.

[18] J. Teifel, "Self-voting dual-modular-redundancy circuits for single-event-transient mitigation," *Nuclear Science, IEEE Transactions on*, vol. 55, no. 6, pp. 3435 –3439, Dec. 2008.

[19] T. Uemura, Y. Tosaka, H. Matsuyama, K. Shono, C. Uchibori, K. Takahisa, M. Fukuda, and K. Hatanaka, "Seila: Soft error immune latch for mitigating multi-node-seu and local-clock-set," in *IRPS 2010*, 2010.

[20] S. Mitra, K. Brelsford, and P. Sanda, "Cross-layer resilience challenges: Metrics and optimization," in *DATE, 2010*, mar. 2010, pp. 1029 –1034.

[21] M.-C. Hsueh, T. Tsai, and R. Iyer, "Fault injection techniques and tools," *Computer*, vol. 30, no. 4, pp. 75 –82, apr. 1997.

[22] D. E. Holcomb, W. Li, and S. A. Seshia, "Design as you see FIT: System-level soft error analysis of sequential circuits," in *DATE*, April 2009, pp. 785–790.

[23] I. Wagner and V. Bertacco, "Engineering trust with semantic guardians," in *Proceedings of the conference on Design, automation and test in Europe*, ser. DATE '07. San Jose, CA, USA: EDA Consortium, 2007, pp. 743–748.

[24] "IBM ILOG CPLEX Optimization Studio 12.3," http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.