Power-Aware Dynamic Control of Error-Resilience Mechanisms

Wenchao Li
Susmit Kumar Jha
Sanjit A. Seshia

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2011-109
http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-109.html

October 3, 2011

Acknowledgement

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.

Power-Aware Dynamic Control of Error-Resilience Mechanisms

Wenchao Li
UC Berkeley
wenchaol@eecs.berkeley.edu

Susmit Jha
UC Berkeley
jha@eecs.berkeley.edu

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 wearout, 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 the most vulnerable components and then statically hardening them to produce a resilience to overhead tradeoff. 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 that can turn the resilience mechanisms `ON` and `OFF` dynamically and at the right time. These predicates are *mined* using a 0-1 integer linear optimization formulation. An experimental evaluation shows that the proposed approach provides a systematic way to control error-resilience so as to meet reliability targets under a specified power budget. For example, for a chip multiprocessor router, our approach achieves the same amount of soft error resilience with only half of the power overhead compared with the static hardening approach.

I. Introduction

Technology scaling to sub-90nm 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], [2]). Fortunately, in recent years there have been several efforts in this direction, including techniques for hardening circuits against soft error [3], [4], [5] and methods for mitigating the effects of circuit aging [6], [7]. Error detection, recovery and retry mechanisms have also been studied and implemented extensively in modern microprocessors such as in the IBM POWER6 [8], [9].

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. [10] recently showed that for an implementation of the European Space Agency SpaceWire protocol, less than 25% of the latches (flip-flops) needed to be protected against soft errors using Intel's built-in soft error resilience (BISER) [5], 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 latch (flip-flop) versus when to use a standard latch: a latch 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 latches 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 [11], [12], [13], [14], [15], [16] has shown that circuit components exhibit vulnerability non-uniformly. 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, 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*. We show how these conditions can be computed efficiently using a combination of two steps. The first involves *mining a set of useful predicates* from traces obtained in fault injection experiments. In the second step, a *0-1 integer linear program* is solved to find optimal matching of error-resilience logic (e.g., latches hardened to soft errors) to their

controlling predicates. Optimality is defined as maximizing circuit reliability for a given power budget.

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 use predicate mining to generate useful conditions for control synthesis.
- 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 selectively hardening technique for the same overhead.
- An experimental evaluation on a chip multiprocessor router [17] and a hardware electronic voting machine [18] shows that this approach provides a systematic way to control soft error-resilience using BISER [5].

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 have 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. Common fault models are stuck-at and delay, but increasingly transient faults such as soft errors are becoming a concern due to aggressive technology scaling.

In the context of soft errors, statistical fault injection [19], [11], [13], [15], [16] and fault-free simulation (e.g. architectural vulnerability factor estimation [12]) are two common approaches that evaluate vulnerability of different circuit components. Selective hardening of a design for fault tolerance can then be performed using error prevention or correction techniques such as gate sizing [20], [21], redundant circuit for voting [22] or resilient latches [5], [23].

Formal methods have also been used to guide the protection of circuit components. Seshia *et al.* [10] uses a formal verification-guided approach to identify latches that are inherently resilient and therefore do not have to protected. However, this does not provide a way to prioritize protection on the vulnerable latches especially when there is a small power overhead budget. Similar to [11], [13], [15], [16], Holcomb *et al.* [24] uses a simulation-based evaluation to probabilistically estimate the vulnerabilty of each latch with respect to formal specifications. These vulnerability measures are then used to prioritize latch protection using BISER. However, dynamic control of BISER is not investigated in this work.

Additionally, Miskov-Zivanov and Marculescu [21] proposed the use of Markov chains to model sequential system behavior and evaluate system level failure rate as the steady-state probability of output nonequivalence. Selective gate sizing is then used to optimize for soft error resilience. In our empirical studies, we focus on providing resilience against *soft errors* occurring at latches. However, our optimization framework works for other faults and their respective fault-tolerant mechanisms. For a recent survey on resilience techniques, we point the readers to [25].

Our work differs from the above approaches in a few ways. One key difference is we provide *dynamic control* to the resilience mechanisms, where these mechanisms are turned `ON` or `OFF` by their controlling predicates at run time. This allows us to further optimize a fault-tolerant design while meeting the power budget constraint. Also, to the best of our knowledge, our work is the first to *automatically synthesize* the logic (by using predicate mining and solving an optimization problem) for dynamically controlling these configurable error-resilience mechanisms.

III. Overview

Our optimization flow is illustrated in Figure 1.

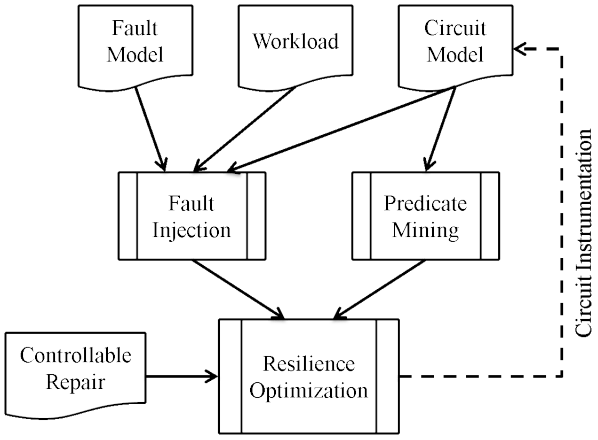

Fig. 1. Dynamic Control Optimization Flow

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 mine 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 mined predicates and the controllable repair, and outputs the allocation of repairs along with the control logic that is optimal for controlling them for maximizing resilience under a given budget constraint. Instrumentation can then be performed to maximize the circuit resilience. The control logic is essentially a mapping from the mined predicates to the repairs. It allows us to take advantage of conditions that can identify critical parts of the computation and hence protect the corresponding components only when it is needed.

Figure 2 illustrates this concept. Suppose the circuit is subject to SEU at the latches, and we are provided with protection mechanisms that can be used to guard the latches against

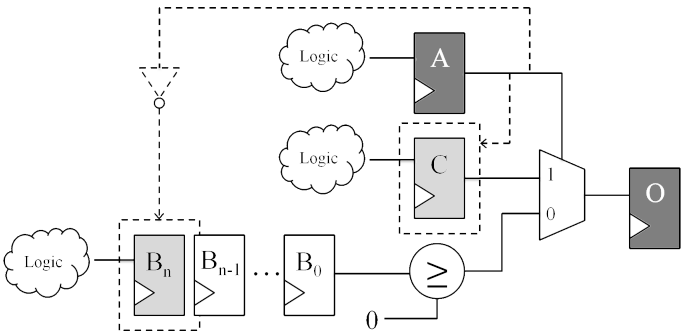

Fig. 2. Illustrative Example

SEU. In addition, 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 latch 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 line shows the control logic that needs to be inserted to turn ON the configurable protection mechanisms (shown as dashed boxes) *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 latches (by 50%) and still achieve protection of all the latches from SEU. In the following sections, we will show that how this logic can be automatically synthesized such that the error-resilience of a circuit is maximized for a given power budget.

IV. Formal Models

We describe in this section the formal model and terminology 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 (latches) 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 latches in L and the inputs.

We denote the value of latch l at cycle j by l^j. Similarly, the vector of values of latches 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 latch to flip. [1] It is parameterized by latch l and cycle j. In other words, when an SEU occurs in cycle j, the value of latch 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 latch 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.

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 the input sequences that we employ 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 latches. For example, for an SEU, the built-in soft error resilience (BISER) [5] technique can be operated in an "economy" mode which roughly halves the dynamic power consumption of a BISER latch, bringing the power consumption almost down to that of a regular non-BISER latch. BISER is based on re-using the scan portion of latches, and thus the "economy" mode is enabled by turning off the scan portions by assigning proper values to the scan clocks.

Thus, consider a controllable, local repair r that negates the effect of an SEU in a latch. The repair r is parameterized by the latch l and a Boolean control signal b. This means that we can set r to ON by setting $b =$ True or OFF by setting $b =$ 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 run of the circuit, i.e. $l^{j+1} := f_l(I^j, L^j)$. Otherwise $l^{j+1} := \overline{f_l(I^j, L^j)}$.

The goal of this paper is to synthesize, for each latch l, the combinational circuit whose output is the control signal b. We describe our approach in the following section.

V. Resilience Control Synthesis

A. Notation

We introduce notation 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 latch l is controlled by predicate p. In other words,

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

if $b_{l,p} = 1$, it means that when p is evaluated to `True`, r of latch l is `ON` at the same cycle, (so that a SEU at l will not be latched at the next cycle). Otherwise r is `OFF`. If r is controlled by `True`, it is then `ON` all the time.

Each repair comes with an associated cost. We use $c_{l,\texttt{ON}}$ denote the power consumed by latch l when it is turned `ON` and $c_{l,\texttt{OFF}}$ denote the power when it is turned `OFF`.

We assume that we are given a set of Boolean formulas, or *predicates* P over L. For each $p \in P$, $p_{\tau,j}$ 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 percentage times that p is evaluated to `True`. Formally, given a set of traces Π, α_p is given by the following expression:

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

where $\beta_{p,\tau,j}$ is 1 when $p_{\tau,j} =$ `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 latch repair to a controlling predicate. Formally, the resilience measure $v_{l,p}$ is a measure of the improvement in error-resilience if latch 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 $v_{l,p}$ metric defined as follows. Suppose we are given a set of fault injection experiments containing `good` and `bad` traces, where a trace is `bad` if the fault results in a system-level failure, and `good` otherwise. We partition these traces into sets E_l for each latch l because only one error at a single latch 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 latch l (l is protected from the SEU by turning r of l `ON`).

B. Problem Formulation

Informally, our goal is to maximize the total gain in resilience subject to a power budget.

More precisely, the problem of resilience control synthesis using local repairs is defined as follows.

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 latches 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:

$$maximize \sum_{p \in P} \sum_{l \in L} b_{l,p} v_{l,p} + C \sum_{p \in P} \sum_{l \in L} b_{l,p} \alpha_p \quad (1)$$

where $C \sum_{p \in P} \sum_{l \in L} b_{l,p} \alpha_p$ is a "regularization" term in which the constant C controls the relative weight of the second term in the objective function. The main reason for including the second term is to synthesize control predicates that generalize well to traces of the system that have not been sampled in fault injection experiments. In the absence of this term, our solution would only be optimal with respect to the specific fault injection experiments performed in generating α_p and $v_{l,p}$ values, and there is a possibility of "over-fitting" these fault injection experiments. The intuition of the regularization term is that we also want to keep the repair r of latch l controlled by p in the `ON` mode as often as possible, so as to maximize error resilience (within the power budget).

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

(A) *Total power overhead is less than Φ:* [2] [3]

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

where d_p is a binary variable denoting whether predicate p is used to control any latch 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.

$$\forall p \in P, \sum_{l \in L} b_{l,p} \le d_p \times |L| \quad (3)$$

$$\forall p \in P, \; d_p \le \sum_{l \in L} b_{l,p} \quad (4)$$

(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.

$$\forall l \in L, \sum_{p \in P} b_{l,p} \le 1 \quad (5)$$

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

[2] 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

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

problem. Hence, P^* together with $\{b^*_{l,p}\}$ give us the control logic for SEU resilience using repairs R.

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 is a 0-1 knapsack problem, which can be solved by dynamic programming [26]. The objective function of the static problem takes the form of (1) with the following constraints

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

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 constraint (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. Mining Useful Predicates

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

Many such tools seek to learn specifications dynamically from an execution trace (or a set of traces). Daikon [27] is one of the earliest tools that mine single-state invariants or pre-/post-conditions in programs. In the hardware context, recent papers [28], [29] have proposed techniques for mining properties (specifications) from circuit simulation or execution traces. 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 mine predicates that are in the form of a Boolean conjunction of two literals, where each literal is either the value of a latch or its negation. Formally, $P = \{p | p = lit_1 \wedge lit_2\}$, where $lit_i = l$ or $lit_i = \neg l$, for $l \in L$. With additional information of a synthesized circuit such as placement and routing, one can construct the predicates only from the nearby signals of a latch so that to minimize wiring. Such a heuristic will reduce the number of variables $b_{l,p}$ in the optimization problem significantly while incurring only a small penalty on the resilience. 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 we are able to achieve significant gain in resilience over the static hardening approach.

VI. Case Studies

To evaluate the effectiveness of our approach, we did two case studies - one on a hardware electronic voting machine and one on a chip multiprocessor router design. We focus on providing resilience to these designs against soft errors using BISER. Simulation results from [5] showed that BISER can reduce SER of a flip-flop by a factor of more than 20 as compared to an unprotected flip-flop. In both experiments, we use the following parameters for the optimization problem:

- $\forall l \in L$, $c_{l,\texttt{ON}} = 1.2$ and $c_{l,\texttt{OFF}} = 0.1$, normalized to the power cost of a latch (which is 1). This means if the BISER latch is set to `ON` all the time, it consumes 2.2 times of the power of a normal latch. On the other hand, if the BISER latch is set to `OFF` all the time, it consumes 1.1 times of the power of a normal latch. These numbers are based on [5].
- $c_p = 0.2$ if the predicate is a conjunction of two distinct literals. We assume that the power cost of a `AND` gate is approximately 20% of that of a latch. $c_p = 0$ if the predicate is a single literal. For simplicity, we assume that the value of a latch and its negation are freely available. We also assume the power cost of additional wiring is 0.
- We use $C = 0$ for the regularization. In a less exhaustive simulation environment, one can choose $C > 0$ to obtain a good solution that is generalizable to unseen traces as well.

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 provides better resilience than the static approach. Moreover, the dynamic approach reaches reliability goal at a much lower cost than the static approach. In the result plots, 100% power overhead of providing resilience corresponds to instrumenting every latch 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 are applied.

A. Voting Machine

The first case study is voting machine design presented by Sturton et al [18]. It is a prototype of a direct recording

electronic voting machine. The user can browse forward or backward through the 8 contests by pressing the navigation buttons on the display. Each contest has 8 candidates and the user can select up to a predefined number of candidates for each contest by pressing the selection buttons on the display. The voting machne records the candidate selection for each contest. The display shows the current contest state and the contest number of the active contest.

We consider a realistic model of user interaction, in which the user can correct an error if it is reflected in the voting machine's display when the user reviews the displayed selections. More specifically, the user browses through different contests and enters his selection. After entering the selections once for all the contests, the user can go back and forth with equal probability across contests to review his selections on the display. So, if a fault happens over selection state of a particular contest, the user can notice this if he browses through that contest again and views the selection state on the display. Such fault gets corrected by the user. But if a fault happens for a contest which is never reviewed by the user, the error cannot be corrected and shows up as an error trace. Similarly, all faults in navigation related state elements can be detected by the user since he would notice the fault on the display. Also, we consider the case of SEU. In our experiments, we assume that the user reviews at least 10 contests moving back and forth through contests with equal probability starting with the last contest after the fault happens.

In the fault injection experiment, we run 4000 simulations with fault injected at a particular contest state. These simulations are injected at random clock cycles during each simulation. All state elements of a single contest are protected by a common predicate condition. The predicate condition is p or $p \wedge q$ where $p, q \in CN = \{contest\_num[2], contest\_num[1], contest\_num[0]\}$. The ILP optimization problem was solved using `feaspump` [30]. It took less than a minute to solve each ILP problem.

In Figure 3, we can see that `Dynamic` always gives a larger resilience gain than `Static`. With our optimization approach, we are able to provide 100% resilience at 57.3% power overhead while static approach takes 83.8% where 100% overhead corresponds to *all latches* being protected. With 46.9% overhead, our optimization approach provides 90% resilience.

B. CMP Router

Our second case study is a simplified version of a chip multiprocessor router designed in Verilog [17]. For this benchmark, our approach achieves significant results – dynamic hardening offers 98% of soft error protection on flip-flops with only a quarter of the power overhead needed in a static hardening approach. The router has two input and two output ports; The router is used to direct incoming packets, called flits, to the correct output port. 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

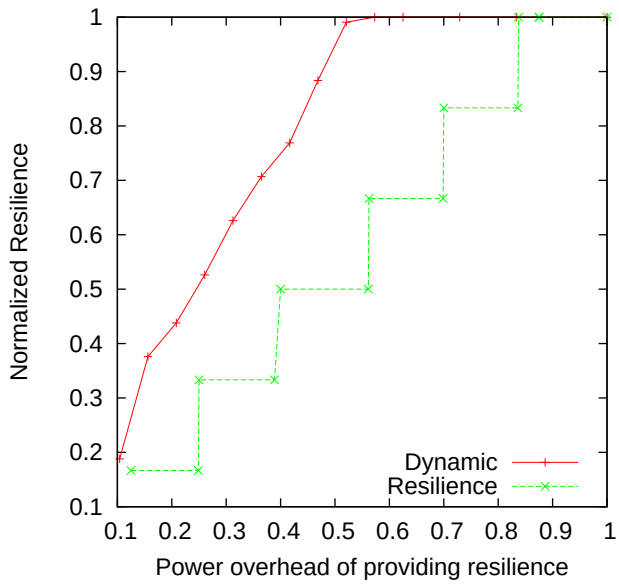

Fig. 3. Voting Machine: Resilience gain vs. Power overhead. In our experiment, we fix a power budget on the x-axis and then optimize for resilience gain

direct each of its input packets to the output port specified by the packet header within a specified number or cycles.

This simplified version has a total of 166 latches. A balanced (1-to-1) workload for the two input ports is used to simulate the router in both the fault injection experiment and the fault-free case. In the fault injection experiment, for each latch, we run 300 simulations, each of about 1000 cycles, and a SEU is injected at a random cycle during each simulation. In the fault-free case, we run a single simulation of about 3000 cycles. This trace is used to estimate α_p for each p. We did not use the faulty traces to esimate α_p because the probability of a SEU is small.

The *0-1 integer linear program* has 964759 binary variables and 11721 linear constraints. We use `feaspump` [30] solver on the NEOS Server [31] to find a good feasible solution quickly. The time that it takes to find a feasible solution using `feaspump` was under a minute for solving each ILP.

In Figure 4, `Dynamic` is a solution to our optimization problem. We can see that `Dynamic` always gives a larger resilience gain than `Static`, as proved in Section V-B. For example, at 5% overhead, `Dynamic` achieves 63.9% resilience, which is 70% better than `Static` which achieves 37.7%. In addition, `Dynamic` achieves 98% resilience at 20% overhead, which is much lower than `Static` at more than 80% overhead.

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

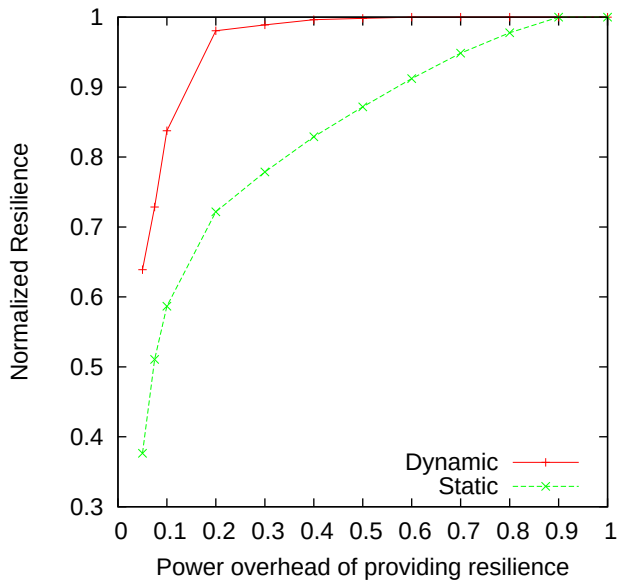

Fig. 4. CMP router: Resilience gain vs. Power overhead. In our experiment, we fix a power budget on the x-axis and then optimize for resilience gain.

any power overhead budget. The control logic is synthesized by first mining simple predicates and then optimally mapping them to the local repairs. Our framework is general. It can be applied to other controllable local repairs and error models. In the future, we would like to explore predicate learning algorithms that can be used in this framework, and the use of controlling predicates to realize 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] S. Nassif et al., "A resilience roadmap," in *DATE*, 2010.

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

[4] Q. Zhou and K. Mohanram, "Gate sizing to radiation harden combinational logic," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 1, pp. 155–166, 2006.

[5] M. Zhang, S. Mitra, T. M. Mak, N. Seifert, Q. Shi, K. Kim, N. Shanbhag, N. Wang, and S. Patel, "Sequential element design with built-in soft error resilience," *VLSI*, Dec. 2006.

[6] M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra, "Circuit failure prediction and its application to transistor aging," in *VTS*, 2007, pp. 277–286.

[7] 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.

[8] 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, 2007.

[9] 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.

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

[11] 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. [Online]. Available: http://portal.acm.org/citation.cfm?id=647883.738395

[12] 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.

[13] 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: IEEE Computer Society, 2004, pp. 61–. [Online]. Available: http://portal.acm.org/citation.cfm?id=1009382.1009722

[14] X. Li, S. V. Adve, P. Bose, and J. A. Rivers, "Softarch: An architectural-level tool for modeling and analyzing soft errors," in *DSN, 2005*, 2005, pp. 496–505.

[15] 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, no. 4, pp. 1986 –1991, 2010.

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

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

[18] C. Sturton, S. Jha, S. A. Seshia, and D. Wagner, "On voting machine design for verification and testability." in *ACM CCS, 2009*. ACM, 2009, pp. 463–476.

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

[20] 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. [Online]. Available: http://doi.acm.org/10.1145/1233501.1233603

[21] 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.

[22] 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, 2008.

[23] 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*, May 2010.

[24] 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.

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

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms, second edition," 2001.

[27] M. D. Ernst, J. H. Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao, "The daikon system for dynamic detection of likely invariants," *Sci. Comput. Program.*, vol. 69, no. 1-3, pp. 35–45, 2007.

[28] S. Vasudevan, D. Sheridan, S. J. Patel, D. Tcheng, W. Tuohy, and D. R. Johnson, "Goldmine: Automatic assertion generation using data mining and static analysis," in *DATE*, 2010, pp. 626–629.

[29] W. Li, A. Forin, and S. A. Seshia, "Scalable specification mining for verification and diagnosis," in *DAC '10*, 2010, pp. 755–760.

[30] "Feaspump2 for solving feasibility of milp," www-neos.mcs.anl.gov/neos/solvers/milp:feaspump/AMPL.html.

[31] "Neos server for optimization," http://neos.mcs.anl.gov/.