ResiLogic: Leveraging Composability and Diversity to Design Fault and Intrusion Resilient Chips

Ahmad T. Sheikh†, *Member, IEEE,* Ali Shoker, Suhaib A. Fahmy, *Senior Member, IEEE,* and Paulo Esteves-Verissimo, *Fellow, IEEE*

***Abstract*—A long-standing challenge is the design of chips resilient to faults and glitches. Both fine-grained gate diversity and coarse-grained modular redundancy have been used in the past. However, these approaches have not been well-studied under other threat models where some stakeholders in the supply chain are untrusted. Increasing digital sovereignty tensions raise concerns regarding the use of foreign off-the-shelf tools and IPs, or off-sourcing fabrication, driving research into the design of resilient chips under this threat model. This paper addresses a threat model considering three pertinent attacks to resilience: distribution, zonal, and compound attacks. To mitigate these attacks, we introduce the *ResiLogic* framework that exploits *Diversity by Composability*: constructing diverse circuits composed of smaller diverse ones by design. This approach enables designers to develop circuits in the early stages of design without the need for additional redundancy in terms of space or cost. To generate diverse circuits, we propose a technique using E-Graphs with new rewrite definitions for diversity. Using this approach at different levels of granularity is shown to improve the resilience of circuit design in *ResiLogic* up to $\times 5$ against the three considered attacks.**

***Index Terms*—Chip Resilience, Hardware Diversity, TMR, Fault and Intrusion Tolerance (FIT), E-Graphs**

I. Introduction

INTEGRATED Circuit (IC) development is a pressing challenge in the modern digital ecosystem. The huge supply chain and toolchain across the development pipeline involves a large number of *known and unknown* stakeholders and dependencies. This has compounded the development complexity and results in higher likelihood of *unintentional faults*, glitches, bugs, etc. [1], [2] and other *intentional intrusions* with malicious intent [3]. While resilience to unintentional benign faults has been well studied and mitigated in the literature, mainly using hardening and redundancy [4]–[9], the case of resilience against malicious actors (e.g., vendors and foundries) has often been ignored. With nation states aligning their priorities to develop local semiconductor manufacturing, technological excellence, and bridge recent global chip shortages [10]–[12], the quest for new resilience methods to withstand untrusted stakeholders is regaining momentum.

Our work is the first that studies chip design resiliency assuming a threat model in which vendors and silicon fabs may be untrusted. Typical chip design can be more productive when the designer relies on available off-the-shelf tools and libraries from different vendors and then sends the design to a silicon fab for manufacturing. The premise is that the vendors and fabs are trusted, both ethically and technically. With the recent digital sovereignty tensions, the former is not guaranteed and hardly measurable, while the latter has been shown to be unrealistic with such a complex development process, incurring too many dependencies on third party tools, libraries, and sometime open source resources [13]. Our analysis shows that a design is highly prone to three potential attacks of relevance to resiliency (as outlined in Section III): *Distribution* attacks (supply chain trojans and backdoors), *Zonal* attacks (using electromagnetic or *Laser* beam interference), and *Compound* attacks (simultaneous *Distribution* and *Zonal* attacks).

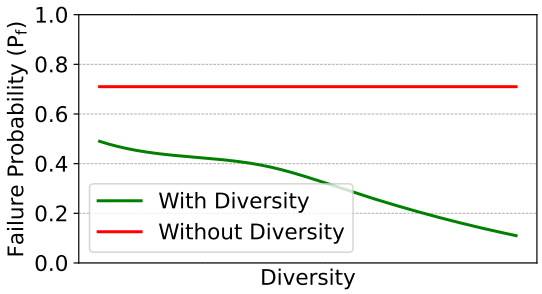


Figure 1: Multi-level diversity (used in *ResiLogic*) has a significant impact on tolerating common mode failures.

Redundancy in space is a fundamental resilience approach that is used to mask faults and intrusions [4], [14]. Two main chip resiliency directions for redundancy at different granularity are studied in state of the art chip design. The first is fine-grained, where redundancy is applied at the logic gate level [15]. This approach aims at diversifying the logic circuit design through adding various basic logic gates, e.g., AND and Inverter. While this approach has been shown to be effective, e.g., increasing design resilience up to 90% under (benign) faults [16], it is designed to be incorporated in the vendor's IP libraries and tools, where the designer has little control. The designer must either create libraries (costly and unproductive) or live under the mercy of the vendor and fab (who may collude).

To facilitate diversification, we provide a new technique to generate diverse circuits using E-Graphs [17]. While E-Graphs have been often used to optimize circuits, we introduce new equality saturation rewrite definitions to produce functionally equivalent but structurally different artifacts at a reasonable overhead for several benchmarks.

The authors are with the CEMSE Division, King Abdullah University of Science and Technology (KAUST) Thuwal 23955-6900, Kingdom of Saudi Arabia. (emails: {ahmad.sheikh, ali.shoker, suhaib.fahmy, paulo.verissimo{@kaust.edu.sa}})

†Correspondence address: ahmad.sheikh@kaust.edu.sa

The second redundancy approach is course-gained, following the *Triple Modular Redundancy* [5], [18] (TMR) style replication that requires three identical copies of a circuit followed by a majority voter to mask the faulty one. However, this works well as long as a minority of circuits are non-faulty and assuming the independence of failures, i.e., the absence of *Common Mode Failures* (CMF, henceforth); this is often considered a hard assumption [19], [20]. In addition, this often incurs high replication cost (i.e., often a factor of three). As shown in Fig. 1, the average failure probability of TMR against CMF (red curve) is around 70%, even when used together with the aforementioned E-Graph approach [16] (see more details in Section VI).

We address this by enabling the concept of *composability* that leverages the properties of higher level composable logic circuits, i.e., circuits whose modular structure is composed of smaller linked circuits. Together with diversification, composability boosts the resilience without incurring extra replication costs, thanks to the implicit modularity of composable circuits. This holds for various circuits that are composable by nature, e.g., *Systolic Array*, *N-bit Multiplier*, *N-bit Adder*, *Vector Unit*, etc. [21], [22], which makes the approach feasible for several modern applications (see Section IV for details).

Our third contribution is introducing *ResiLogic*, a framework for building resilient diversified integrated circuits at a reasonable overhead. *ResiLogic* allows for designing resilient circuits through employing different levels of redundancy and diversity combining diversification and composability. *ResiLogic* provide specifications and tools to cover the process from diverse circuit generation, CMF testing, all the way to different levels of redundancy (by composability and replication). This results in a significant reduction in the probability of failures as shown in Fig. 1. Our evaluation demonstrates that the circuit resilience, like *N-bit Adder* can withstand Distribution, Zonal, and Compound attacks up to a factor of five.

The rest of the paper is organized as follows: related work is discussed in II, Section III introduces the threat and fault models. Section IV introduces the notion of *Diversity by Composability* that is the basis for the *ResiLogic* framework discussed in Section V. *ResiLogic* evaluation strategy and results are presented in Section VI, followed by the concluding remarks in Section VII, respectively.

II. Related Work

The resilience of a system is significantly improved by applying replication followed by a voter for majority consensus. Traditional approaches to avoid CMF in TMR/DMR have been to implement diverse systems at higher levels of abstraction [9], [23]. Faults can be intentional or accidental and have predictable manifestations in digital circuits. State of the art literature classifies faults/vulnerabilities into four categories: 1) hardware trojans or malicious implants, 2) backdoor through test/debug interfaces, 3) accidental/unintentional vulnerabilities [2], [3], and 4) external faults [24], [25]. To combat low level faults, resilience through design diversity must be applied at the gate level. Cartesian Genetic Programming (CGP) [16] had been proposed to generate diverse isofunctional structures to improve resilience at the gate level of a design [15]. Similarly, redundant designs can be spatially separated to provide resilience against external faults in TMR systems [26]. *ResiLogic* attempts to build on this concept of diversity, however, proposes a novel approach to combine small diverse structures to create larger diverse artifacts.

Faults based on external attacks can be due to various sources. Power supplies can be tampered with to alter uniform behavior, resulting in erroneous results. Similarly, power spiking can cause a processor to not only misinterpret an instruction but also induce memory faults [27]. Clock glitches, through a deviated clock signal, can be induced to cause execution of subsequent instructions before retirement of previous instructions, i.e., modification of the program counter (PC) at irregular intervals [1]. This type of attack is considered to be the simplest as low-end FPGAs can be used to inject clock glitches [27], [28]. Electronic devices function properly in a certain temperature range, however, when exposed to extreme temperatures, the data inside memories can be corrupted. These attacks are easier to set up but their effect cannot be localized. Optical attacks are performed by exposing the device to a focused laser beam or strong photo flash. This attack can be targeted on either side of the chip and can be confined to certain area to set or reset memory bits or switch transistor state [29]. Similarly, electromagnetic (EM) interference can corrupt or modify memory contents by inducing eddy currents, resulting in a single or multiple bit faults [30]. Both optical and electromagnetic attack fits our definition of zonal attacks in this paper.

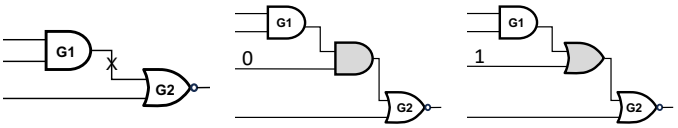
Attacks based on malicious implants can be inserted into a design at various stages, such as by an untrusted CAD designer, by a malicious tool, or at the foundry [31]. Furthermore, hardware verification is a time-consuming and costly task, making it challenging to detect such trojans at scale [32]–[34]. Malicious implants can be activated by an attacker using a carefully crafted strategy [35]. Previous works have discussed hardware trojan activation using a combination of software-hardware [36] and triggers based on time, input sequence and traffic patterns [37]. We model these *Distribution Attacks* by injecting fault(s) at specific location(s) in the design and investigate which input patterns are able to activate them.

III. Threat and Fault Models

A. Actors and Adversaries

We consider a typical design process of three main actors in the ecosystem: *Designer*, *Vendor*, and *Fab*. The designer may use several products from several vendors (e.g., many tools and implementations) to develop a resilient chip, which is sent to the fab for fabrication. The roles and trust models of the actors are as follows:

- *Designer*: the designer of the chip and the main user of *ResiLogic*. They are assumed to be a trusted entity, and a target/victim of acting adversaries.
- *Vendor*: the provider of pre-designed prerequisites like design libraries, implementations, or toolchains. Not all vendors are trusted entities.



(a) Injection (b) `stuck-at-0`. (c) `stuck-at-1`.

Figure 2: `stuck-at` intrusion/fault injection.

- *Fab*: the post-design fabrication foundry that uses silicon technology to implement the design as an integrated circuit. Not all fabs are trusted entities.

B. Intrusions and Faults

During the design and manufacturing process, the design is prone to unintentional faults and intentional intrusions, leading to some functional circuit failure. Faults can be induced at any stage, even by the designer; while intrusions are assumed to be performed through attacking the circuit by the vendor or the fab, who can also collude. Intrusions and faults and their corresponding failures can manifest as `stuck-at` faults [38]. In this model, the injected fault/intrusion on a logic gate results in a logic `0` (`stuck-at-0`) or `1` (`stuck-at-1`). Fig. 2 depicts a schematic of intrusion/fault injection between (a) two gates $G1$ and $G2$, leading to (b) `stuck-at-0` or (c) `stuck-at-1` faults. This can be replicated between any gates in the design.

C. Threats and Attacks

We consider the threat model where the goal of the adversaries, i.e., vendor and fab, is to break the *integrity* of the design or circuit. The aim is to subvert the output either permanently or occasionally. We define three types of potential integrity attacks, i.e., Distribution, Zonal, and Compound attacks. SAT attacks [39], which are devised to extract the functionality of obfuscated circuits are not in the purview of this work.

1) Distribution Attack: It is typically faster and cheaper for the designer to use off-the-shelf tools, pre-designed modules, compilers, synthesizers, etc, to create a new design. Any malicious vendor in this chain can induce intrusions, trojans, or backdoors [3]. The Distribution attack (or fault) is a supply-chain attack where a fault or intrusion can be induced at any step prior to or after design [2], [3], [40]. This attack targets any circuit module, channel, or zone.

2) Zonal Attack: The Zonal attack is a directed attack on the design layout. As shown in Fig. 3, the attack can target a chip in many ways, among them: (1) using an electric bus to overheat an edge [41], (2) using an Electromagnetic field to subvert another edge [30], [35], or (3) using an optical laser beam to target a specific zone after fabrication [41].

3) Compound Attack: The Compound attack is a more complex attack where Distribution and Zonal attacks are preformed simultaneously. In this case, these attacks could be coordinated, e.g., collusion between vendor and fab, or uncoordinated. Although the former is more directed, both scenarios are devastating to the circuit as we show in the simulated experiments.

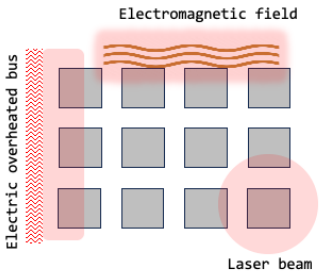


Figure 3: Possible zonal attacks.

IV. Diversity by Composability

In this section, we provide the motivations and design decisions behind *ResiLogic*.

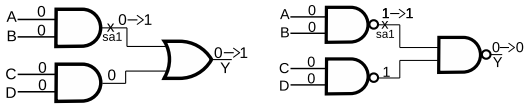
A. The Quest for Resilience

Designing for resilience often requires redundancy in space or time [7]. Redundancy in time entails repeating computation using the same hardware. This is effective against transient failures, but not permanent faults and longer term intrusions. Hence, redundancy in space through replicated hardware, with convergence to a final output is often used. This final output can incorporate the replicated outputs or choose between them, depending on application. Since most applications require determinism, reconciling outputs by majority voting is the most common approach in the literature [14]. This is usually on top of Double- or Triple-Modular-Redundancy (TMR). Majority voting only works if the replicated logic units do not fail together due to Common Mode Failures (CMF) [42]. Consequently, modular redundancy exhibits very limited resilience (e.g, against transient failures) at a high hardware cost. We enable the effectiveness of TMR through techniques to generate and use diverse replicas that are functionally equivalent but structurally different.

On the other hand, the replication cost of TMR may not be justified or affordable. This raises the challenge to find other ways to leverage diversity without including extra replication costs. In our work, we propose composability as an effective approach against untrusted supply chain stakeholders. Importantly, we show that composability preserves the determinism properties of the diversified logic circuits.

B. Diversity Levels

Redundancy and diversity have been studied in the literature across the entire development process, involving all actors: vendors, designers, and fabs. However, the threat model we discussed in Section III may render the redundancy implemented by a malicious vendor or unreliable fab. This necessitates redundancy implemented by the designer to mitigate these potential malicious threats. For integrated circuits, we identify three possible levels of diversity: Gate-level, Module-level, and Artifact-level diversity. We are interested in providing the *designer* with a mechanism to build resilient circuits, following a bottom-up approach, where diversity at the gate level is utilized to create larger diverse artifacts *by composition*.



(a) $Y = AB + CD$ (b) $Y = \overline{\overline{AB} \cdot \overline{CD}}$

Figure 4: Gate-level diversity highlighting different manifestations of the $stuck - at - 1$ fault.

1) ***Gate-level diversity****:* The lowest level of diversity we consider, targets designing diverse primitive logic modules composed of basic logic gates, e.g., AND, OR. The modules (e.g., adders, multipliers, accumulators, etc.) require resilience since they are core to the datapaths of larger designs, and importantly, they are often provided by IP vendors—which we assume to be untrusted. Therefore, an intrusion or fault at this level can be replicated as much as these primitive logic units are used. We require the designer to have access to diverse implementations preferably from different IP vendors or, if possible, constructed by the designer themselves. This minimizes the likelihood of using diverse units from many malicious IP vendors at once, and thus reduces CMFs. Diversity at this level can be achieved using different combinations of logic gates while maintaining the same functionality. This can be done manually, but preferably using an automated algorithm as we do in this work, inspired by [43], [44]. For instance, the equivalent boolean functions $AB + CD$ and $\overline{\overline{AB} \cdot \overline{CD}}$ will have different gate-level implementations that exhibit different behaviors under attacks (manifesting as *stuck-at*). As an example, Fig. 4 shows two diverse implementations of the same boolean function. If all inputs are logic `0`, the true output of both circuits is logic `1`. If a `stuck-at-1` fault is injected at the same point, the two circuits exhibit different behaviors at the final output `Y`.

Definition IV.1 (Diverse Module)**.** *Consider any arithmetic logical circuit $m \in M$ modeled as an underlying DAG of vertices (gates) and edges (gate outputs and inputs). M is a Diverse Module (DM) iff there exist at least two implementations $m_i, m_j \in M$ such that $m_i \neq m_j$ (differ in at least one DAG edge), and $out(m_i) = out(m_j)$ (deterministic) where out is their output function.*

2) ***Module-level diversity****:* This diversity level aims at building a module artifact as a composition of diverse modules with some dependency between them. The artifact's output can be dependent on the position/order of faulty or vulnerable module(s). (This is different from module replication in the TMR sense.) The designer makes use of off-the-shelf modules, diversified at the Gate-level, to create higher order *Composed Module Artifacts (CMAs)* through additional diversity at this layer. We formally define a CMA as follows:

Definition IV.2 (Composable Module Artifact–CMA)**.** *Consider an arithmetic logical artifact L with a deterministic specification S. An implementation A of L is an ordered set of modules M_i defined as:*

$$A = (M_1, M_2, ..., M_N) \in \Pi = \{M_i \times M_j \times ... \times M_N\} \quad (1)$$

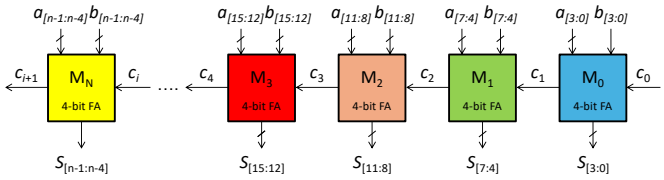


Figure 5: N-bit Ripple-Carry Adder composed of *diverse* 4-bit adder modules.

where Π is a set of logical circuit modules. A is a Composable Module Artifact (CMA) iff it respects the specification S given a defined connectivity matrix Γ.

$$\Gamma_{N \times N} = \begin{bmatrix} V_{0,0} & V_{0,1} & \cdots & V_{0,N-1} \\ V_{1,0} & V_{1,1} & \cdots & V_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ V_{N-1,0} & V_{N-1,1} & \cdots & V_{N-1,N-1} \end{bmatrix} \quad (2)$$

where,

$$V_{ij} = \begin{cases} 1 & \text{if } a_i \text{ connected to } a_j \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

An implementation A uses N modules to conform to the specification S. The connectivity matrix (Γ) of size $N \times N$ is a directed acyclic graph (DAG) that defines how Modules are connected with each other. For example, to construct an $N - bit$ *Ripple-Carry Adder (RCA)*, the connectivity matrix would simply be the Identity Matrix (I) as shown in Eqn. 4. Similarly, an $N - bit$ multiplier would require N^2 AND gates and $2N - 1$ adders (either half adders or full adders).

A Composable Module Artifact has modularity properties that can guarantee deterministic logic over diversified modules. We define and prove this in the next Lemma.

Lemma IV.1. *(A Diverse CMA is deterministic) A CMA $A_i = (M_i, M_j, ..., M_N)$ whose module implementations m_k^t belong to a diverse module M_k is deterministic.*

Proof: Assume the contrary, that there exist at least two diverse CMA implementations $P = (p_1^a, p_2^b, ..., p_N^z)$ and $Q = (q_1^l, q_2^t, ..., q_N^k)$ with at least one tuple location x having $(p_x^e \neq q_x^e)$, $p_x^e, q_x^e \in M_x$ (M_x being a diverse module), and such that $out(P) \neq out(Q)$ (nondeterministic). Since all module implementations p_y^f and q_y^g belong to a diverse module M_y, then it is possible to replace each p_y^f with q_y^g at every index y in CMA P. This generates exactly equivalent tuple implementations $P = Q$. Since we assumed $out(P) \neq out(Q)$, then $out(P) \neq out(P)$. This means having one exact implementation of CMA that is not deterministic, which contradicts the CMA definition in Eq. 1. Therefore, a diverse CMA is deterministic.

■

Such CMAs can be applied to a wide range of designs such as *N-bit Adders/Subtracters, N-bit Multipliers, N-bit Dividers*, or modern accelerator architectures, e.g. *SHA-256* hashing, Vector units, *Systolic Array* architectures [22]. To

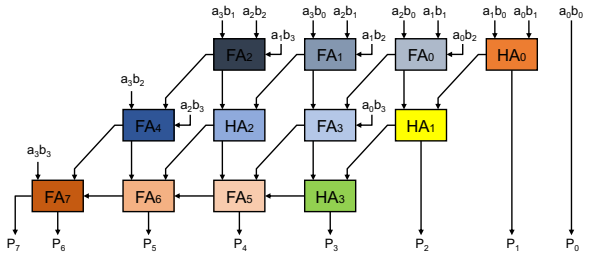


Figure 6: A $4-bit$ Combinational Multiplier composed of *diverse* Half-Adders and Full-Adders.

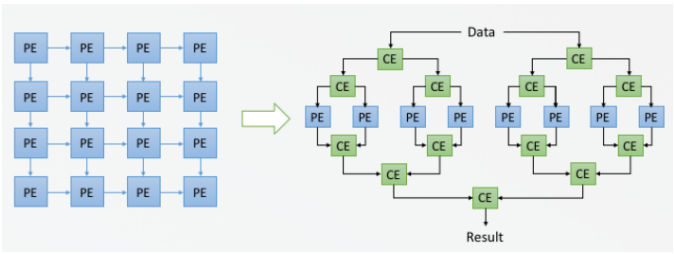


Figure 7: Composable mesh and hierarchical structures of Systolic Arrays [50], composed of (diverse) Processing Elements (PE) and Collection Elements (CE).

demonstrate the concept, we discuss two broadly used designs with different complexity: N-bit Adder and *Systolic Array* architecture which is widely used in *Tensor*-based Deep Neural Networks (DNNs).

$$Adder\ (\Gamma_{N\times N}) = \begin{array}{c} \\ M_0 \\ M_1 \\ \vdots \\ M_{N-1} \end{array} \begin{array}{c} \begin{array}{cccc} M_1 & M_2 & \cdots & M_N \end{array} \\ \begin{bmatrix} \mathbf{1} & 0 & \cdots & 1 \\ 0 & \mathbf{1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{1} \end{bmatrix} \end{array} \quad (4)$$

N-bit Ripple-Carry Adder: Fig. 5 shows an N-bit Ripple-Carry Adder (RCA) composed of smaller modules which can be diverse or homogeneous structures. The adder is composed of $4-bit$ full adder modules. The basic resolution of a module can be decided by the designer. However, smaller modules provide less room for diversity due to the smaller number of gates, while large modules can suffer from worse critical path delays. A $4-bit$ FA module granularity provides a good trade off for exploring diverse combinations. The corresponding connectivity matrix is an Identity matrix that defines how the modules are connected to realize an arithmetic add/subtract operation. The value of 1 denotes which corresponding row module is feeding its output to the column module.

Definition IV.3 (Homogeneous/Heterogeneous CMA)**.** *An artifact implementation $A_i = (M_i, M_j, ..., M_N)$ is a homogeneous CMA iff $M_x = M_y\ \forall\ M_x, M_y$; otherwise, it is heterogeneous.*

4-bit Array Combinational Multiplier: The $4-bit$ combinational multiplier in Fig. 6 is composed of various Half-Adders (HA) and Full-Adders (FAs) which sum partial products. Composability allows use of diverse implementations of HAs and FAs. For brevity, the AND gates that perform bitwise multiplication are not shown. In fact, a library of diverse AND gates can also be created to generate partial products through diverse implementations. The corresponding connectivity matrix is in Eqn. 5, $Mul\ (\Gamma_{N\times N})$:

$$\begin{array}{c} \\ HA_0 \\ HA_1 \\ HA_2 \\ HA_3 \\ FA_0 \\ FA_1 \\ FA_2 \\ FA_3 \\ FA_4 \\ FA_5 \\ FA_6 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} HA_1 & HA_2 & HA_3 & FA_1 & FA_2 & FA_3 & FA_4 & FA_5 & FA_6 & FA_7 \end{array} \\ \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \end{array} \quad (5)$$

The matrix highlights how the modules are connected to realize a multiplication operation. It can also be observed that most of the elements in the matrix are zero i.e., sparse. We foresee it to be a typical structure of a connectivity matrix.

Lemma IV.2. *There exists an N-bit Multiplier implementation that is a heterogeneous CMA. Proof in Appendix IV.2.*

Systolic Arrays: Systolic arrays [45], [46] are a class of parallel computing architectures designed to efficiently execute matrix computations, which are fundamental to many scientific and engineering applications [47], [48] including signal processing [48], image processing, and machine learning [47]. Based on the Systolic architecture, Google proposed Tensor Processing Units (TPU) [22], [49], a custom ASIC to accelerate DNN applications. A typical Systolic array design consists of a 2-Dimensional grid of Processing Elements (PE), a controller and the on-chip memory/buffer. Fig. 7 depicts two examples of composable Systolic Array architectures using Processing Elements (PEs) and Collection Elements (CE) [50]. Leveraging the design of diverse adders and multipliers, diverse PEs can be designed to enhance the resilience of systolic arrays.

Further Use Cases and Applications: Similar to these Adder, Multiplier and Systolic arrays CMA implementations, there is a large variety of other useful arithmetic designs that are composable in this manner [51]. The artifacts we target in this work are basic logic circuits that are widely used, making this approach applicable in many use cases.

Nevertheless, given our threat model, our work may be even more applicable in ASIC or FPGA designs, where the area dedicated to arithmetic operations is significantly higher. More promising use cases may target accelerators for ML algorithms

or cryptographic functions, e.g., *SHA256* or Cryptocurrency miners [52].

Modern cryptographic and Machine Learning (ML) algorithms rely heavily on parallelized datapaths with ample arithmetic modules. *SHA-256* is an integral part of cryptographic algorithms to achieve high level of security [53]. It consists of several rounds of rotate and XOR operations, making it a rich target for applying the composability principle by choosing diverse implementation for each operation.

*3) **Artifact-level diversity**:* This third level of diversity makes use of multiple *replicas* of diverse CMAs running simultaneously, the outputs of which are computed using a majority voter. This is similar to TMR with an important difference: common mode failures are reduced significantly, since diverse CMAs of diverse modules can be used by composition. Visual examples are shown in Fig. 12. Our evaluation in Section VI shows that this level is key to thwart Distribution and Compound attacks in particular.

Furthermore, due to the coarse granularity, the placement of these replicas can be directed to leverage some resilience to Zonal attacks. Combining placement with diversity at different levels boosts the resilience of the system against all considered attacks: Distribution, Zonal, and Compound, as we convey in the evaluations in Section VI.

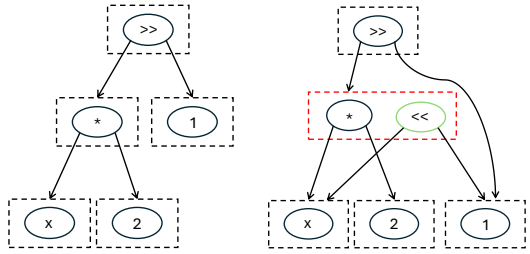
V. The *ResiLogic* Framework

This section presents the *ResiLogic* framework that is used by the designer to implement resilient designs based on the diversity concepts introduced in Section IV. The framework defines a mechanism to construct circuits while: (1) maintaining deterministic behavior across the entire design despite replication and diversity, and (2) ensuring resiliency to the attacks outlined in Section III.

In *ResiLogic*, the ultimate goal is to produce a resilient circuit corresponding to a CMA following Definition IV.2. The process follows three main phases summarized in Algorithm 1, with the details discussed next.

A. Building Diverse Modules via Gate-diversity

This phase aims at identifying the different candidate module implementations m_i for use in a CMA implementation $A = (m_1, m_2, ..., m_N)$; e.g., m_i :=4-bit-adder in our example. The aim is to generate diverse modules of $m_x^l,\ m_x^t \in M_x$, such that $out(m_x^t) = out(m_x^z)$, where out is the output function, but the structure differs in at least one logic gate, i.e., $m_x^t \neq m_x^z$, thus generalizing the diverse CLA to $A_i = (M_i, M_j, ..., M_N)$ in the next phase. Therefore, the designer enumerates off-the-shelf and/or custom implementations of these candidate modules with the criteria defined in the Diverse Module Definition IV.1: all module implementations of M_x are deterministic, i.e., Determinism is, hence, guaranteed by definition. In the case of a homogeneous CMA, a single module type is used across the entire tuple. Heterogeneous CMAs will however be built from diverse modules. To automate this process, we utilize a recent technique for diversifying logic based on E-Graphs [17].



(a) Initial E-graph: $(x \times 2) >> 1$ (b) Apply: $(x \times 2) \rightarrow x << 1$

Figure 8: E-graph rewriting for standard integer arithmetic. Dashed boxes represent e-classes of equivalent expressions. Green nodes represent newly added nodes. Red dashed boxes highlight which e-class has been modified. The arcs are between e-nodes and e-classes.

Example. Fig. 8 highlights the application of the E-Graph in Fig. 8a through a `rewrite` with equivalence operation $(2 \times x)$ is equivalent to $(x << 1)$ to generate an equivalent graph in Fig. 8b, with equivalent operations grouped in the same e-class (red dotted box). Consequently, two mathematically equivalent relations can then be extracted (black dotted box): 1) $(x \times 2) >> 1$ and 2) $(x << 1) >> 1$ with different resiliency profiles.

1) E-graph transformations: An E-Graph or equivalence graph is a data structure that compactly represents equivalence relations in the form of *Abstract Syntax Trees* (AST) [17]. E-Graphs have been mainly used in the optimization of large datapath and arithmetic circuits [54]–[57] and technology aware synthesis [58]. In this work, we use them to generate diverse circuit designs using equivalence *Transformations*. A `rewrite` is a simple AST representing an equivalence operation that is successively applied on an E-Graph to generate an equivalent graph compactly with in the same graph. `Extraction` refers to selecting AST(s) from E-Graph based on a certain cost function. The default cost functions are `ASTSize` and `ASTDepth`, E-Graph tool constructs an optimal AST as a result of minimizing these cost functions.

2) Diversity transformations: We automate the generation of diverse logic circuits using new specifications for *rewrite* and *extraction*. We proposed in Table I a `rewrite` list definition rules for different classes of operations. These rules are used attractively to generate an enormous number of equivalence versions. Afterwards, extraction is defined such that versions have at least one structural difference in its implementation. The *rewrite* and *extraction* transformations ensured versions that are functionally equivalent but structurally different.

3) E-graph diversity implementation: Practically, the implementation of the above process is depicted in the E-Graph flow in Fig. 9 An input circuit represented in the *Verilog* format is synthesized and converted into equation format which represents the circuit in the *AND*, *OR* and *NOT* format. Next, the `rewrite` operations are applied until saturation, and finally a pool of diverse implementations is extracted. Finally, a fault simulation tool, like *HOPE* [59], is utilized

Table I: *ResiLogic* Rewriting Rules

Class	Boolean Rewriting Rules
Complements	$a * 0 \implies (a * \overline{a})$ $a * 1 \implies \overline{a} * (\overline{a} * a) + (\overline{a} * a)$ $a + 1 \implies (a + \overline{a})$ $a + 0 \implies \overline{a} + (\overline{a} * a) + (\overline{a} * a)$ $a * \overline{a} \implies (\overline{a} * a) + (\overline{a} * a)$ $a + \overline{a} \implies (\overline{a} * a) + (\overline{a} * a)$
Covering	$a * (a + b) \implies (a * a) + (a * b)$ $a + (a * b) \implies (a * b) + (a * \overline{b})$
Combining	$(a * b) + (a * \overline{b}) \implies (a * b) + (a * \overline{b})$ $(a + b) * (a + \overline{b}) \implies (a + b) * (a + \overline{b})$
Idempotency	$a * a \implies (a * a) + (a * \overline{a}$ $a + a \implies (a * a) + (a * \overline{a}$
Commutativity	$a * b \implies b * a$ $a + b \implies b + a$
Associativity	$(a * b) * c \implies (a * (b * c))$ $(a + b) + c \implies (a + (b + c))$
Distributivity	$a * (b + c) \implies (a * b) + (a * c)$ $a + (b * c) \implies (a + b) * (a + c)$
Consensus	$(a * b) + (\overline{a} * c) \implies (a * b) + (\overline{a} * c) + (b * c)$ $(a + b) * (\overline{a} + c) \implies (a + b) * (\overline{a} + c) * (b + c)$
De-Morgan	$\overline{(a * b)} \implies \overline{a} + \overline{b}$ $\overline{(a + b)} \implies \overline{a} * \overline{b}$

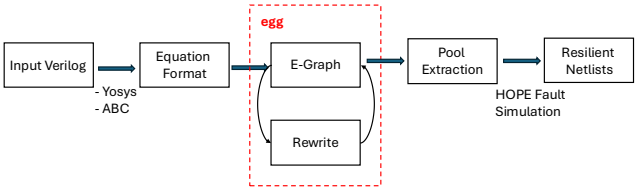


Figure 9: E-graph flow diagram.

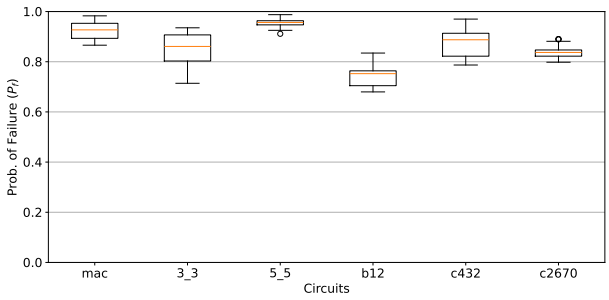
to quantify the fault coverage of pooled *netlists* and perform further pruning to select the candidates that demonstrate low fault coverage i.e., high resiliency.

Transformations in E-Graph consist of two main steps 1) `rewrite` and 2) `extraction`. We use Yosys/ABC logic synthesis [60], [61] flow to synthesize a benchmark circuit into an equivalent `equation` format that could be fed to the E-Graph tool for design space exploration.

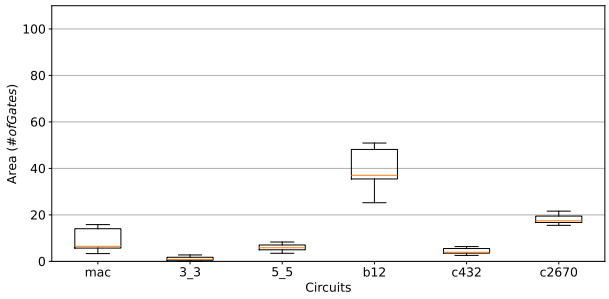
4) E-graph diversity evaluation: The effectiveness of E-Graph based approach to generate diverse and resilient circuits is evaluated on the benchmark circuits shown in Table II. The benchmarks are a mix of circuits from LGSynth [62], [63], ITC [64], and ISCAS85 [65] benchmark suites. Additionally, an open source divider from OpenCores is also considered for evaluation. The considered benchmarks are: `genmul` [66] that generates two multipliers (3x3 and 5x5); `mac` benchmark that corresponds to the Multiply-Accumulate operation unit of

Table II: E-Graph Diversity Generation Benchmarks.

Circuit	I/O	Area (Gates)	Prob. Failure (P_f)
MAC	16/8	121	0.99
3_3 (genmul)	6/6	46	0.96
5_5 (genmul)	10/10	167	0.99
b12 (ITC99)	126/127	1069	0.87
c432 (LGSynth89)	36/7	247	0.95
c2670 (ISCAS85)	233/139	837	0.95



(a) Resilience



(b) Area

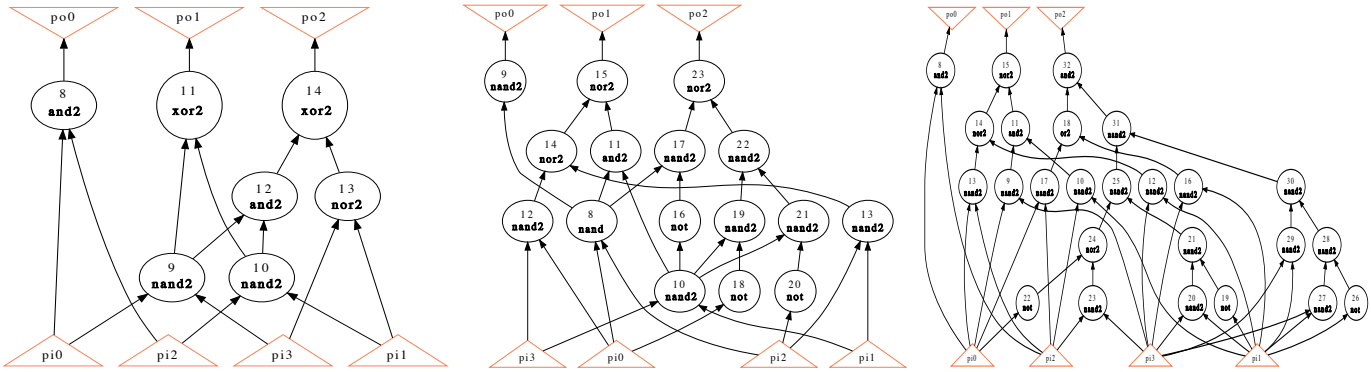
Figure 10: Resilience and area ranges of diverse generated artifacts of benchmarks with Min-Max normalization.

a typical *Systolic* array; LGSynth [62], [63], ITC [64], and ISCAS85 [65] benchmark suites. The `rewrite` (Table I) and `extract` methods of E-Graphs are implemented in `Rust` to interface with the `egg` library.

Fig. 10a and 10b highlight the resilience and area overheads of the above benchmarks. For benchmarks `3_3` and `c432`, the `egg` tool was able to generate diverse replicas with varying resilience (P_f) levels at low area overhead. The area overhead increased in the cases of `mac`, `c2670` and `b12`.

5) Observations and analysis: We observed that the resilience and corresponding area profiles of the egg tool are sensitive to the `rewrite` rules in Table I. New rules added or existing ones modified can lead to different implementations and area/resilience profiles. However, since optimizing this process is orthogonal to this work, the obtained diversity was satisfactory to (1) demonstrate our proof of concept and (2) generate enough diversity for the rest of the evaluation. In addition, one can notice that the probability of failure is not significantly improved with this level of fine-grained diversity. Nevertheless, combined with the other levels of diversity presented in the next sections, we show that the probability of failures is reduced significantly. An interesting research would be to optimize the E-Graph transformations to generate more diverse circuits at a lower area cost.

6) Synthesis Impact on Resilience: The inherent nature of Electronic Design Automation (EDA) synthesis tools is to optimize circuits to meet the area, power and delay constraints. To achieve resilience, additional gates have to be added that doesn't change the function of circuit but increase the area and delay. Since E-Graph works on the ASTs of a given language, therefore, it relies on parser and lexer for text processing. Any



(a) Example circuit (7 Logic Gates) $P_f = 1$

(b) With Optimization (16 Logic Gates) $P_f = 0.84$

(c) Without Optimization (25 Logic Gates) $P_f = 0.74$

Figure 11: Synthesis impact on sample circuit for diversity. P_f is computed using HOPE [59] parallel fault simulation tool.

redundancy in the resulting E-Graphs of diverse replicas will 1) be removed by the parser, or 2) removed by the synthesis tools.To avoid these optimizations, following steps can be followed: 1) rename each redundant gate with a different tag, 2) declare all redundant gates as primary inputs to the circuit, 3) synthesis will not perform any optimizations, and 4) remove redundant gates from the primary inputs list. Following these steps, it is observed that high resilience can be achieved, but at the cost of exorbitant area.

Fig. 11 illustrates the impact of synthesis on digital circuits. Example circuit (Fig. 11a) consists of 7 logical gates with $P_f = 1$. Applying E-Graph rewrite and extract operations with optimizations i.e., not tagging each gate, the resulting area after synthesis is 16 gates with the corresponding $P_f = 0.84$, as shown in Fig. 11b. However, if each gate is tagged i.e., no optimization is applied, synthesis of the circuit will result in 25 logical gates with $P_f = 0.74$, as shown in Fig. 11c. This simple example highlights the area is increased by more than 3x if no optimization is applied. For larger circuits, we observed that the resulting area becomes exorbitantly high (with corresponding significant reduction in P_f) making the final design unfeasible for practical applications. It will be interesting to investigate the impact of E-Graphs to generate diverse replicas at different levels of digital circuits abstraction.

B. Intra-diversity: building a CMA out of Diverse Modules

This phase leverages the generated diverse modules M_i to build a resilient diverse CMA $A_i = (M_i, M_j, ..., M_N)$. The designer starts with the standard implementation of the CMA using any generated module implementations in the previous phase. A conservative option is to use the most resilient CMA implementation $A = (m_1, m_2, ..., m_N)$. In general, this depends on the policy considering the resilience, cost, space, and time tradeoffs. To create diverse CMAs, the designer iterates on the tuple to replace a module implementation m_i^t with another generated one $r_i^t \in M_i$. Diverse CMAs can be generated either by changing a single module implementation at index i or multiple modules in CMA A_i.

Algorithm 1 *ResiLogic* Diversification

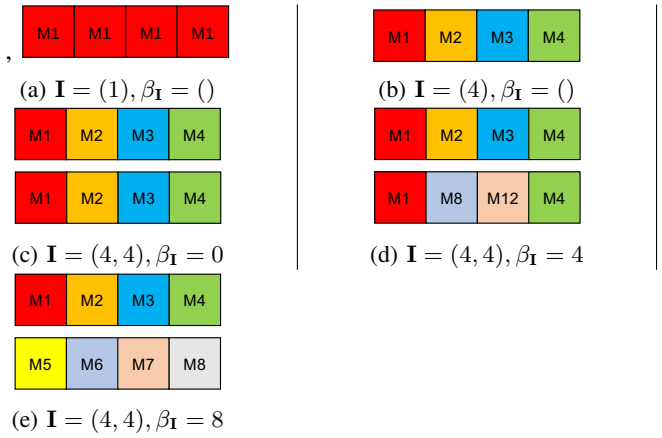
Require: Π: Set of diverse modules M_i
Require: Γ: Connectivity Matrix
Res : Required resilience range of modules
I : Intra-diversity factor
D : Inter-diversity factor
R : Number of fault resilient replicas
N : Required bit-width of artifact
$K \leftarrow FALSE$
while $(K \neq TRUE)$ **do**
  for $(i = 1 \rightarrow R)$ **do** ▷ Create R replicas
    Select modules satisfying N based on I & Res factor
  end for
  if (D condition satisfied among replicas) **then**
    Apply Γ to realize replicas operations
    Connect replicas to the majority voter
    $K \leftarrow TRUE$
  end if
end while

C. Inter-diversity: building a replicated artifact

This phase uses TMR-style replication of the *diverse* CMAs. The designer starts by selecting one CMA implementation and then replicates it into three replicas whose outputs are connected to a majority voter. (One can scale the replication to any numbers.) The designer then iterates over the three tuples to replace the CMA implementations with diverse homogeneous or heterogeneous ones. The latter is preferable as this utilizes Inter-diversity between CMAs to defend against Distribution and compound attacks by reducing common mode failures. To defend against Zonal attacks, we leverage the coarse-grained nature of CMA replication to define a resilient-aware fixed placement of redundant modules in the chip design.

D. ResiLogic Configurations

The above multi-level diversity is expressed through configurations that combine the Intra- and Inter-diversity. We define these configurations as follows. We denote the Intra-diversity of a CMA by α. This refers to the number of different modules used in the CMA composition. Since different CMAs in a replicated artifact can have different Intra-diversity α, we model these as a tuple $\mathbf{I}$ whose dimension $|\mathbf{I}|$ is the replication factor. In this case, each index I_i refers to the Intra-diversity



(a) $\mathbf{I} = (1), \beta_{\mathbf{I}} = ()$ (b) $\mathbf{I} = (4), \beta_{\mathbf{I}} = ()$

(c) $\mathbf{I} = (4,4), \beta_{\mathbf{I}} = 0$ (d) $\mathbf{I} = (4,4), \beta_{\mathbf{I}} = 4$

(e) $\mathbf{I} = (4,4), \beta_{\mathbf{I}} = 8$

Figure 12: Replicas manifestation with varying α & β values.

value for the implementation A_i of replica i. The $\beta_{\mathbf{I}}$ factor for a given tuple $\mathbf{I}$, describes the Inter-diversity among the replica CMAs i.e., how diverse they are with respect to each other. $\beta_{\mathbf{I}}$ is computed by taking the symmetric difference among replicated CMAs, as shown in Eqn. 6. To compute $\beta_{\mathbf{I}}$ for $|\mathbf{I}| > 2$, Eqn. 6 can be applied iteratively.

$$\beta_{\mathbf{I}} = (A_i - A_{i+1}) \cup (A_{i+1} - A_i) \tag{6}$$

Fig. 12 shows few example configurations for different values of $\mathbf{I}$ and $\beta_{\mathbf{I}}$. (The terms CMAs and replicas will be used interchangeably.) In Fig. 12a, a single replica is created with a single module M_1 i.e., $\mathbf{I} = (1)$. Fig. 12b shows a single replica consisting of a set of four different modules $\{M1, M2, M3, M4\}$ i.e., $\mathbf{I} = (4)$ and $|\mathbf{I}| = 1$. Fig. 12c shows a case where $|\mathbf{I}| = 2$ with $\mathbf{I} = (4,4)$ i.e., two replicas are created with four diverse modules, however, they are the same as each other, hence $\beta_{\mathbf{I}} = 0$. Fig. 12d demonstrates a case where four distinct modules are used to create two replicas ($\mathbf{I} = (4,4)$) with two common modules $M1$ and $M4$, the inter-diversity ($\beta_{\mathbf{I}}$), computed using Eqn. 6, is 4. Lastly, Fig. 12e consists of two replicas created with configuration $\mathbf{I} = (4,4), \beta_{\mathbf{I}} = 8$, which implies each replica is composed of four different modules and there are no common modules between them, therefore, they are completely diverse.

E. ResiLogic Scalability

Let's say that the basic unit of computation is defined as a module M and the collection of such modules be denoted by M_i. A replica R_i is a union of modules M_i arranged and connected together in a certain way. The number of modules in a replica is proportional to the size and functionality of an artifact aimed to be realized. A module M_i is a basic unit of function and a set of such modules Π represents diverse implementations of some function or a set of modules with multiple functions. A replica R_x with required bit width x and $|M_i|$ being the bit width of the module M_i, can be defined as:

$$R_x = \bigcup_{j=1}^{x/|M_i|} M_i \qquad \forall M_i \in \Pi = [M_1, ..., M_N] \tag{7}$$

Large SoC designs typically comprise interconnected components with varying functions and levels of design complexity. *ResiLogic* focuses on operating at the component-level abstraction. This approach significantly reduces the complexity of designing large SoCs. However, the time complexity of generating a diverse system with N replicas that meet the I and D factors in Algorithm 1 is directly proportional to these factors.

Design of diverse chips is not a panacea but an attempt to increase the confidence in system resilience against the CMF. With diverse modules at our disposal, design of resilient circuits would then be an amalgamation of these modules along with the connectivity matrix Γ to realize a function. Through the proposed approach, the set of diverse modules can be created leading to the creation of larger diverse artifacts. For fault resilience, mere diversity in a single replica is not enough, a replicated system with Triple Modular Redundancy (TMR) has to be created to: 1) tolerate single fault, 2) detect CMF. TMR ensures to tolerate a single fault, however, a common vulnerability can be exploited to bring the whole system down to its knees. That said, Algorithm 1 describes the steps to create resilient systems in the wake of persistent and dynamic threat. The algorithm creates fault resilient replicas by satisfying the intra and inter diversity conditions.

VI. Results & Discussions

A. Experimental Settings

Our evaluation aims at measuring the resilience of *ResiLogic*'s generated diverse circuits to the Distribution, Zonal, and Compound attacks. Since resilient circuits entail redundancy, we also measure the overhead on time, space, and power.

1) Measuring common mode failures (CMF): Quantifying diversity between two digital circuits is typically done through fault injection at the gate level and measuring the probability of failures. Mitra et. al. [67] used random fault injections which does not fully capture the failure space. To address this, we have built a testbed CMF engine that systematically injects faults into gate locations of two candidate diverse circuits following a fault campaign as discussed in Section III. The output is then compared to a reference golden circuit where faults are injected. Eqn. 8 shows that for the CMF output to be true, all outputs O_k from the replicas must be equal to each other and different from the reference output RO.

$$CMF \equiv O_i = O_j \quad AND \quad O_1 \neq RO \quad \forall\, i \neq j \tag{8}$$

Fig. 13 shows the block diagram of the proposed common mode failure computing engine. It fetches a set of diverse replicas from the library generated by the method discussed in Section V. No fault is injected in RR, which serves as a reference circuit producing the golden output. The number and sites of faults to be injected depends on the selected fault campaign as discussed in Section III. Eqn. 8 shows that for the CMF output to be true, all outputs from the replicas must be equal to each other and different from the reference output RO. The steps to quantify the CMF are inspired from [68].)

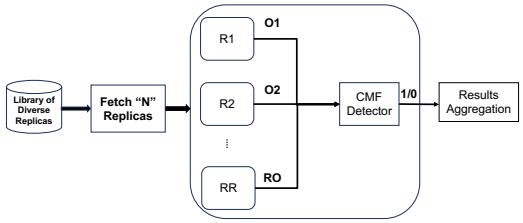


Figure 13: CMF Computing Engine Architecture.

and elaborated in Algorithm 2. The number and sites of faults to be injected depends on the selected fault campaign as discussed in Section III.

Algorithm 2 : Common Mode Failure (CMF) Evaluation

Require: R Replicas
Require: CMF Compute Engine 13
SIM : Simulation Count
F : Failure rate of the circuit
Rel : Reliability (%) of the circuit
$\mathbf{V}$: Randomly generated 100,000 test vectors
CMF_{OUT}: Instantaneous CMF detector output
K : Failure Count
$K \leftarrow 0$
for $(i = 1 \rightarrow SIM)$ **do**
    Inject faults w.r.t. the *Error Campaign*
    Apply $\mathbf{V}$ to the circuit with faults injected
    Compute CMF (8) and save results
    for $(j = 1 \rightarrow \mathbf{V})$ **do**
        if $(CMF_j \neq 0)$ **then** increment K
    end for
end for
$F = \left(\frac{K}{SIM}\right)$
$Rel(\%) = (1 - F) \times 100$

2) Diversity classes: We evaluate (1) various *ResiLogic* configurations (variable α and β) compared to (2) TMR as a state-of-the-art replication method and (3) without replication as a baseline (No-TMR). The focus here is to create diverse N-bit adders. We do not consider other CMAs due to space limitations and given that the N-bit adder is a good representative example. For simulations and evaluations, the $16-bit$ Ripple-Carry adder (RCA) artifact is considered. However, the proposed approach can be extended to any value of N.

A selected set of diverse CMA Adder implementations were generated with different diversity ratios are used across all experiments. The selected versions were enough to generate hundreds of permutations and millions of intrusion/fault injections. The selected modules–functionally equivalent but structurally different–along with the gate count and failure probability (P_f) are shown in Table III. They are also classified into three resilience levels which depends on the respective failure probabilities of the module. The selected modules have varying fault tolerance, area and logic level. Area is measured in terms of the number of logic gates, while the logic depth shows the longest path between an input and an output. P_f denotes the fault coverage against `stuck-at-0` and `stuck-at-1` faults and is computed by performing simulations against a single fault using the HOPE fault simulator [59]. From Table III it is also evident that, as the number of gates increase, P_f (not CMF) decreases due to the logical masking of the modules.

Table III: 4-bit Full Adder (FA) modules (M_i) [15].

Resilience Level	Modules (M_i)	# Gates	Logic Levels	Prob. Failure (P_f)
Low	M_1	20	9	0.90
	M_2	24	9	0.85
	M_3	25	11	0.82
Medium	M_4	27	9	0.77
	M_5	28	9	0.76
	M_6	29	10	0.70
High	M_7	31	10	0.66
	M_8	32	10	0.64
	M_9	34	9	0.57
	M_{10}	37	10	0.56

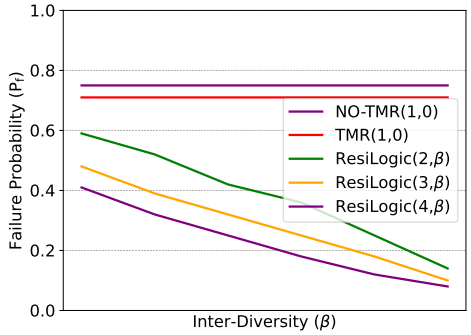


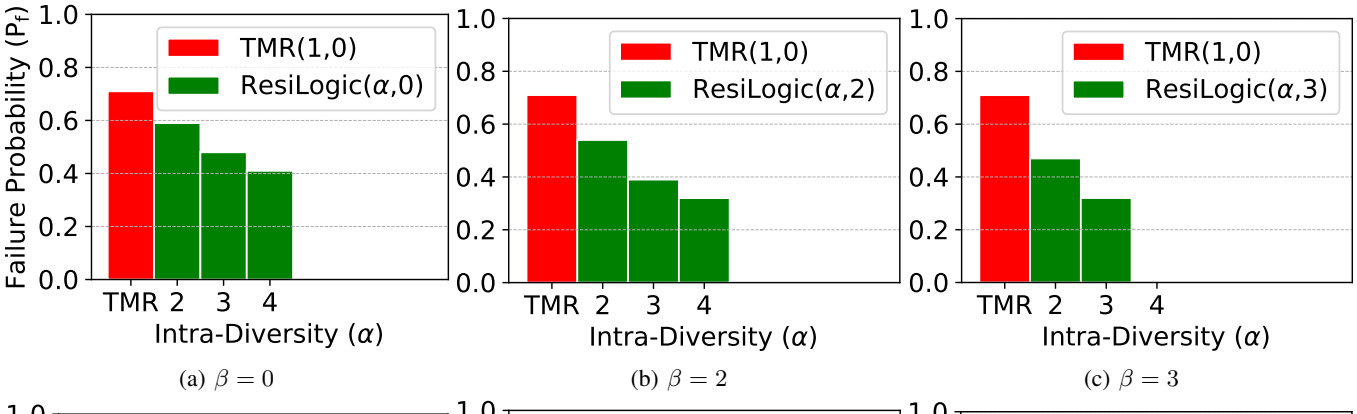
Figure 14: *ResiLogic* (α, β) comparison with TMR and NO-TMR.

3) Simulating Attacks: To simulate Distribution faults, same faults are injected at the same location in the common modules among replicas. Evaluation is done by injecting faults across all possible locations in common modules. For example, consider Fig. 12d, the common modules are $M1$ and $M4$ among the two replicas: faults will be injected at the same locations of $M1$ and $M4$ in these replicas. However, the total fault injection combinations will be the cross product of the number of gates in the respective common modules i.e., $|M1| \times |M4|$.

In the Zonal fault model, faults are injected at the common zones of the redundant replicas. The zone granularity is a module. In Fig. 12, replicas are divided into four zones as each one of them has four modules. This approach emulates the faults that could be manifested due to external attacks on the same spatial zones of the replicas. $\sum_{zones(z)} \prod_{Replicas(i)} |M_{i,z}|$ Where $|M_{i,z}|$ denotes the total number of gates in a module of replica i in zone z. The zonal fault injection campaign is more compute intensive in comparison to the distribution faults as it takes into account combinations of all possible locations in the replicas. The distribution and zonal attacks are combined in the *Compound attack* model by injecting faults in common modules among replicas and then iterating through zones for zonal fault injection. *It should be noted that we only report the detection of CMFs and expect that there is a correction mechanism in place for the rectification of faults if the replicas' outputs are different from each other.*

B. Resilience under Distribution Attack/Fault

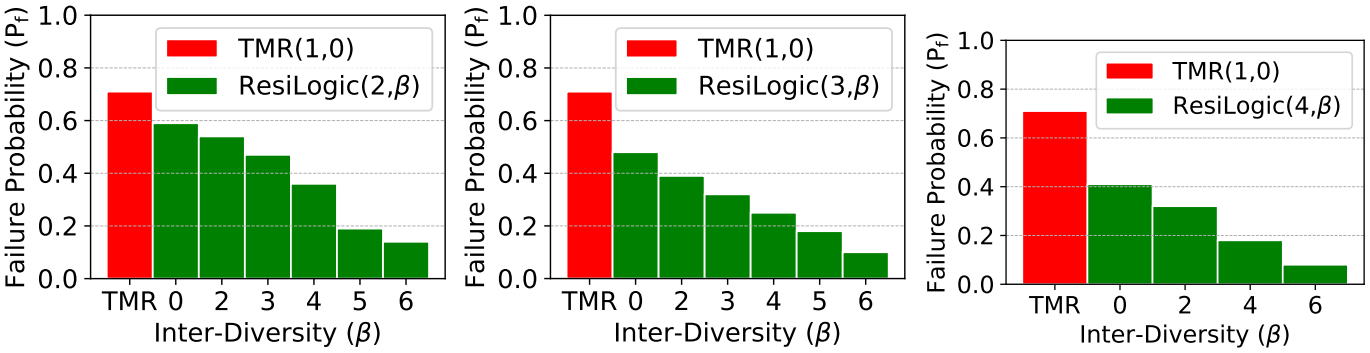
We first discuss the general patterns of failure probability for *ResiLogic* against TMR (without composition diversity)



(a) $\beta = 0$ (b) $\beta = 2$ (c) $\beta = 3$

(d) $\beta = 4$ (e) $\beta = 5$ (f) $\beta = 6$

Figure 15: *ResiLogic*'s (α, β) resilience to Distribution faults/attacks as intra-diversity (α) vary.



(a) $\alpha = 2$ (b) $\alpha = 3$ (c) $\alpha = 4$

Figure 16: *ResiLogic*'s (α, β) resilience to Distribution faults/attacks as inter-diversity (β) varies.

as state-of-the-art solutions and No-TMR (no replication) as baselines. We observe in Fig. 14 that all configurations of *ResiLogic* are significantly more resilient than the baselines. In particular, as more Inter-diversity is induced across replicas *ResiLogic*'s resilience ranges between 0.6 and 0.1, while both TMR and No-TMR (whose lines are separated for visibility) exhibit around 0.7 resilience. This shows that TMR is not effective without diversity as it fails similar to No-TMR due to common mode failures.

We then discuss Intra- and Inter-diversity aside, showing that both are important and complementary. We omit No-TMR for clarity, knowing that its resilience is similar to TMR across all Distribution attack results. Fig. 15 highlights the resilience for fixed β, while intra-diversity α varies between 2 and 4. The results consistently show that diversity by composition improves resilience of *ResiLogic* over TMR as α increases to 10% with each diverse module added. The improvement is significant for higher β. In the best case (f) where $\beta = 6$, i.e., different modules are used in different replicas, the failure probability is lower than 0.1. These results are consistent with the case where α is fixed, whereas the Intra-diversity β varies in Fig. 16. With few diverse modules (a) $\alpha = 2$ per replica, the resilience improves as long as more diverse modules are used in different replicas. The improvement is significant in the most diverse configuration $\alpha = 4$. The interesting observation in varying α and β is that the designer can tune the level of diversity based on the available versions, cost, and sensitivity of the application.

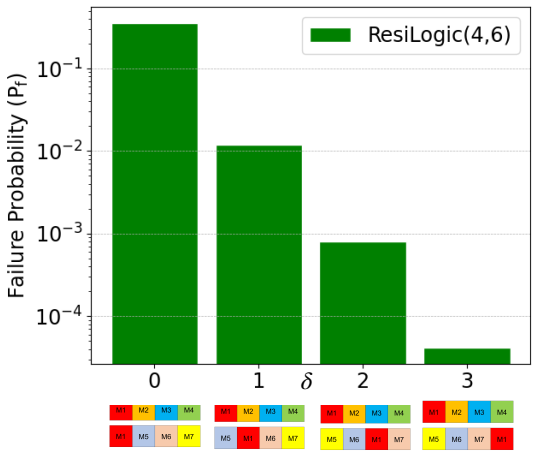


Figure 17: *ResiLogic*'s resilience to Distribution faults/attacks, varying modules' placement in two replicas. Failure probability is lower when common modules are distant.

C. Module placement and alignment across replicas

An impactful property of composition in *ResiLogic* is the position of common modules across replicas that plays an important role in overall resilience. We explain this in Fig. 17, where two replicas $R1$ and $R2$ are constructed in such a way that $R1 = \{M1, M2, M3, M4\}$ and $R2 = \{M5, M6, M7, M1\}$. $M1$ is the common module and the placement distance difference between the location of $M1$ in two replicas is 3 i.e., $M1$ location in $R1$ and $R2$ is 0 and 3, respectively. From Fig. 17, it can be observed that increasing alignment distance between a common module across two replicas improves resilience because varying common module location in replicas has a direct impact on fault controllability and observability conditions, resulting in reduced CMFs.

D. Resilience under Zonal Attacks

We now evaluate *ResiLogic* under Zonal attacks. We pick two simulated scenarios: a horizontal attack (e.g., Electromagnetic field applied Fig. 3) which affects an entire replica and its modules, and a vertical attack that affects one module of each replica. Since No-TMR has no replication, the entire modules of the CMA are attacked. As expected, under the horizontal zonal attack an entire CMA replica is attacked in TMR and *ResiLogic* (4,8) (full diversity). The impact of the attack is 100% masked by the majority voting. The No-TMR case however fails drastically with failure probability close to 0.9. This reinforces the fact that TMR is useful in some cases even without modular diversity under Zonal attacks. The advantage is, however, not present in the case of vertical zonal attack, since all replicas in TMR and *ResiLogic* are hit by the attack, which cannot be masked by the voter. As expected, the failure probability in this case is only affected by the gate-level diversity of the modules used.

E. Resilience under Compound Attacks

We finally consider the case of Compound attack where the Zonal and Distribution attacks are simultaneously applied.

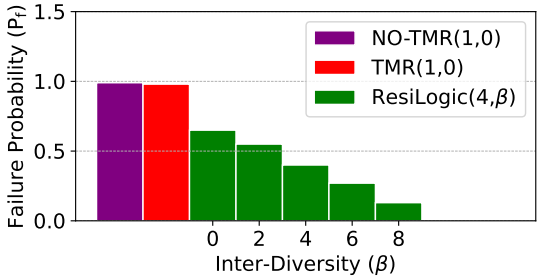


Figure 18: *ResiLogic*'s (α, β) resilience to Compound faults/attacks (simultaneous Distribution & Zonal attacks).

We only choose to evaluate with one *ResiLogic* (best) configuration due to space limitations (and since the trends are similar in others). We fix $\alpha = 4$, while varying the value of β in Fig 18. For the case of $\alpha = 1, \beta = 0$, all modules are victims of the Distribution attack which has a huge impact on TMR and No-TMR as we have seen in Fig. 14. For the case where replicas are composed of diverse modules and there are no common modules among them, a module is picked at random for the injection of the Distribution attack. *ResiLogic* can sustain the Compound attack with down to 0.1 failure probability compared to TMR and No-TMR which fail most of the time.

F. Impact on Area, Power and Delay

To determine the impact of proposed technique on area, power and delay, the replicas are generated with specific resilience according to the classification levels mentioned in Table III. The area, delay, and logic levels are calculated using the SIS synthesis tool [69] for the MCNC library. The MCNC library is slightly modified to include only 2-input AND/OR, NAND/NOR and XOR/XNOR gates along with an Inverter and Buffer. This is done to make sure the library remains highly correlated with the designs mentioned in Table III. The area is computed by adding the fanins of all gates in a circuit. The dynamic power consumption of a digital circuit is determined for the MCNC library operating at a supply voltage of 5V and a frequency of 20 MHz.

Algorithm 1 is applied by fixing the values of α and β to $\{1, 0\}$, $\{1, 8\}$ and $\{4, 8\}$ respectively. For the combination $\{4, 8\}$ 100 CMAs were generated by considering all the modules from $(M_1 ... M_{10})$ in creating CMAs (Eqn. 1) as shown in Table III. Table IV provides valuable insights into the trade-offs between area, power, and delay across different redundancy configurations. The results are averaged, as there are multiple circuits generated against each value of α and β. It can be observed that the area and power overheads are negligible across all profiles, while the delay is more impacted by the depth level of the graph. The case $\{1, 8\}$ is a TMR but with different replicas and it can be observed that the area, power, delay and level steadily increase for each resilience level. This makes sense as higher resilience means more levels, which has a direct impact on other qualitative parameters. From Fig. 18 we observed that the case of $\{4, 8\}$ has the best resilience against the compound attacks, and it is also evident

Table IV: Qualitative profiles highlighting Area, Power and Delay for few diverse cases.

(α, β)	**Modules' Resilience**	**Area (Gates)**	**Power** (μW)	**Delay** (ns)	**Levels**
(1,0)	TMR	1351	3035	65	41
(1,8)	Low	1347	3021	60	39
	Medium	1352	3038	72	43
	High	1357	3051	74	46
(4,8)	$\forall M_i$	1350	3034	68	42

here in Table IV that the overheads are still close to the TMR. This is due to *ResiLogic* fine grained diversity mechanism.

VII. Conclusions

Existing approaches to resilient chip design have mostly been studied under benign fault models where no malicious actors exist [14]. *ResiLogic* addresses that gap, motivated by recent digital sovereignty concerns [10], [11]. Productive chip businesses focus on design, while making use of off-the-shelf tools and libraries from different vendors after which fabrication is outsourced. If vendors and fabs are not trusted, the design can be vulnerable to several attacks, among them three we introduced: Distribution, Zonal, and Compound. To mitigate these attacks, we introduced the notion of *Diversity by Composability* that enables the building of resilient circuits out of smaller diverse modules at a low cost, thus reducing the typical replication factor overhead. We also provided a technique to use E-Graphs to generate diverse replicas at gate-level. The quality of generated diverse replicas is directly influenced by the `rewrite` rules. Together with state of the art gate-level diversity and modular redundancy, we show that resilience to the aforementioned attacks is significantly improved up to a factor of five, and the designer retains some control on diversity.

References

[1] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 370–382, 2006.

[2] A. M. Shuvo, T. Zhang, F. Farahmandi, and M. Tehranipoor, "A comprehensive survey on non-invasive fault injection attacks," Cryptology ePrint Archive, Paper 2023/1769, 2023, https://eprint.iacr.org/2023/1769. [Online]. Available: https://eprint.iacr.org/2023/1769

[3] F. Farahmandi, Y. Huang, and P. Mishra, *System-on-Chip Security: Validation and Verification*. Springer, 2020.

[4] P. E. Veríssimo, N. F. Neves, and M. P. Correia, "Intrusion-tolerant architectures: Concepts and design," in *Architecting dependable systems*. Springer, 2007, pp. 3–36.

[5] A. T. Sheikh and A. H. El-Maleh, "Double modular redundancy (dmr) based fault tolerance technique for combinational circuits," *Journal of Circuits, Systems and Computers*, vol. 27, no. 06, p. 1850097, 2018.

[6] Y.-M. Hsu and E. Swartzlander, "Time redundant error correcting adders and multipliers," in *Proceedings 1992 IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*. IEEE Computer Society, 1992, pp. 247–248.

[7] W. J. Townsend, J. A. Abraham, and E. E. Swartzlander, "Quadruple time redundancy adders [error correcting adder]," in *Proceedings 18th IEEE Symposium on Defect and Fault Tolerance in VLSI Systems*. IEEE, 2003, pp. 250–256.

[8] F. Bas, S. Alcaide, G. Cabo, P. Benedicte, and J. Abella, "Safede: A low-cost hardware solution to enforce diverse redundancy in multicores," *IEEE Transactions on Device and Materials Reliability*, vol. 22, no. 2, pp. 111–119, 2022.

[9] L. A. Tambara, F. L. Kastensmidt, J. R. Azambuja, E. Chielle, F. Almeida, G. Nazar, P. Rech, C. Frost, and M. S. Lubaszewski, "Evaluating the effectiveness of a diversity tmr scheme under neutrons," in *2013 14th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*. IEEE, 2013, pp. 1–5.

[10] U. Congress, "Chips for america act & fabs act," last accessed 23 november 2023. [Online]. Available: https://www.semiconductors.org/chips/

[11] E. Commission, "European chips act: The chips for europe initiative," last accessed 23 november 2023. [Online]. Available: https://digital-strategy.ec.europa.eu/en/factpages/european-chips-act-chips-europe-initiative

[12] I. Spectrum, "How and when the chip shortage will end, in 4 charts," last accessed 23 november 2023. [Online]. Available: https://spectrum.ieee.org/chip-shortage

[13] T. Ajayi and D. Blaauw, "Openroad: Toward a self-driving, open-source digital layout implementation tool chain," in *Proceedings of Government Microcircuit Applications and Critical Technology Conference*, 2019.

[14] J.-i. Inoue, "Gmr, tmr and bmr," in *Nanomagnetism and spintronics*. Elsevier, 2009, pp. 15–92.

[15] U. Afzaal, A. S. Hassan, M. Usman, and J.-A. Lee, "On the evolutionary synthesis of fault-resilient digital circuits," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 2, pp. 281–295, 2022.

[16] J. F. Miller and S. L. Harding, "Cartesian genetic programming," in *Proceedings of the 10th annual conference companion on Genetic and evolutionary computation*, 2008, pp. 2701–2726.

[17] M. Willsey, C. Nandi, Y. R. Wang, O. Flatt, Z. Tatlock, and P. Panchekha, "egg: Fast and extensible equality saturation," *Proc. ACM Program. Lang.*, vol. 5, no. POPL, Jan. 2021. [Online]. Available: https://doi.org/10.1145/3434304

[18] L. Anghel, D. Alexandrescu, and M. Nicolaidis, "Evaluation of a soft error tolerance technique based on time and/or space redundancy," in *Proceedings 13th Symposium on Integrated Circuits and Systems Design (Cat. No. PR00843)*. IEEE, 2000, pp. 237–242.

[19] A. Avizienis, "The methodology of n-version programming," *Software fault tolerance*, vol. 3, pp. 23–46, 1995.

[20] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, "Analysis of operating system diversity for intrusion tolerance," *Software: Practice and Experience*, vol. 44, no. 6, pp. 735–770, 2014.

[21] I. Koren, *Computer arithmetic algorithms*. CRC Press, 2018.

[22] N. Jouppi, C. Young, N. Patil, and D. Patterson, "Motivation for and evaluation of the first tensor processing unit," *ieee Micro*, vol. 38, no. 3, pp. 10–19, 2018.

[23] G. d. M. Borges, L. F. Gonçalves, T. R. Balen, and M. Lubaszewski, "Diversity tmr: Proof of concept in a mixed-signal case," in *2010 11th Latin American Test Workshop*. IEEE, 2010, pp. 1–6.

[24] D. Karaklajić, J.-M. Schmidt, and I. Verbauwhede, "Hardware designer's guide to fault attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 12, pp. 2295–2306, 2013.

[25] L. Fiolhais and L. Sousa, "Transient-execution attacks: A computer architect perspective," *ACM Comput. Surv.*, vol. 56, no. 3, oct 2023. [Online]. Available: https://doi.org/10.1145/3603619

[26] Microchip, "Rt polarfire®: Tmr and spatial separation for higher reliability," last accessed 23 October 2023. [Online]. Available: https://onlinedocs.microchip.com/pr/GUID-34638A73-DA4D-4B97-AAD1-EC033E228DFC-en-US-1/index.html?GUID-26687824-827E-48EC-A5F1-EF0921BE5691

[27] J. Balasch, B. Gierlichs, and I. Verbauwhede, "An in-depth and black-box characterization of the effects of clock glitches on 8-bit mcus," in *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2011, pp. 105–114.

[28] S. Endo, T. Sugawara, N. Homma, T. Aoki, and A. Satoh, "An on-chip glitchy-clock generator for testing fault injection attacks," *Journal of Cryptographic Engineering*, vol. 1, pp. 265–270, 2011.

[29] N. Asadizanjani, M. T. Rahman, M. Tehranipoor, N. Asadizanjani, M. T. Rahman, and M. Tehranipoor, "Optical inspection and attacks," *Physical Assurance: For Electronic Devices and Systems*, pp. 133–153, 2021.

[30] F. Farahmandi, M. S. Rahman, S. R. Rajendran, and M. Tehranipoor, "Cad for electromagnetic fault injection," in *CAD for Hardware Security*. Springer, 2023, pp. 169–185.

[31] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[32] M. Yamashita, C. Otani, K. Kawase, K. Nikawa, and M. Tonouchi, "Noncontact inspection technique for electrical failures in semiconductor devices using a laser terahertz emission microscope," *Applied Physics*

Letters, vol. 93, no. 4, p. 041117, 07 2008. [Online]. Available: https://doi.org/10.1063/1.2965810

[33] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE design & test of computers*, vol. 27, no. 1, pp. 10–25, 2010.

[34] A. P. Shah and P. Girard, "Impact of aging on soft error susceptibility in cmos circuits," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2020, pp. 1–4.

[35] S. Adee, "The hunt for the kill switch," *iEEE SpEctrum*, vol. 45, no. 5, pp. 34–39, 2008.

[36] D. M. Ancajas, K. Chakraborty, and S. Roy, "Fort-nocs: Mitigating the threat of a compromised noc," in *Proceedings of the 51st Annual Design Automation Conference*, 2014, pp. 1–6.

[37] S. Bhunia and M. Tehranipoor, "The hardware trojan war," *Cham,, Switzerland: Springer*, 2018.

[38] K. C. Mei, "Bridging and stuck-at faults," *IEEE Transactions on Computers*, vol. 100, no. 7, pp. 720–727, 1974.

[39] Y. Xie and A. Srivastava, "Mitigating sat attack on logic locking," in *Cryptographic Hardware and Embedded Systems–CHES 2016: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings 18*. Springer, 2016, pp. 127–146.

[40] Y. Xiao, *Security in Sensor Networks*. CRC Press, 2016. [Online]. Available: https://books.google.com.sa/books?id=ISkYgg-tUBoC

[41] M. Nagata, T. Miki, and N. Miura, "Physical attack protection techniques for ic chip level hardware security," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 1, pp. 5–14, 2021.

[42] M. R. Babu, F. N. Taher, A. Balachandran, and B. C. Schafer, "Efficient hardware acceleration for design diversity calculation to mitigate common mode failures," in *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2019, pp. 267–270.

[43] B. C. Schafer and Z. Wang, "High-level synthesis design space exploration: Past, present, and future," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2628–2639, 2019.

[44] Y. Alkabani and F. Koushanfar, "N-variant ic design: Methodology and applications," in *Proceedings of the 45th annual Design Automation Conference*, 2008, pp. 546–551.

[45] H. T. Kung and C. E. Leiserson, "Systolic arrays (for vlsi)," in *Sparse Matrix Proceedings 1978*, vol. 1. Society for industrial and applied mathematics Philadelphia, PA, USA, 1979, pp. 256–282.

[46] P. Quinton, "The systematic design of systolic arrays," Ph.D. dissertation, INRIA, 1983.

[47] R. Xu, S. Ma, Y. Guo, and D. Li, "A survey of design and optimization for systolic array-based dnn accelerators," *ACM Computing Surveys*, vol. 56, no. 1, pp. 1–37, 2023.

[48] H. Snopce, A. Aliu, and A. Luma, "Mapping signal processing algorithms into the systolic arrays," in *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*. IEEE, 2020, pp. 1–5.

[49] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1–12. [Online]. Available: https://doi.org/10.1145/3079856.3080246

[50] C.-P. Lu, "Ai, native supercomputing and the revival of moore's law," *APSIPA Transactions on Signal and Information Processing*, vol. 6, p. e9, 2017.

[51] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, "The aladdin approach to accelerator design and modeling," *IEEE Micro*, vol. 35, no. 3, pp. 58–70, 2015.

[52] F. Calvão, "Crypto-miners: Digital labor and the power of blockchain technology," *Economic Anthropology*, vol. 6, no. 1, pp. 123–134, 2019.

[53] M. Padhi and R. Chaudhari, "An optimized pipelined architecture of sha-256 hash function," in *2017 7th International Symposium on Embedded Computing and System Design (ISED)*. IEEE, 2017, pp. 1–4.

[54] S. Coward, G. A. Constantinides, and T. Drane, "Automatic datapath optimization using e-graphs," in *2022 IEEE 29th Symposium on Computer Arithmetic (ARITH)*. IEEE, 2022, pp. 43–50.

[55] S. Coward, T. Drane, and G. A. Constantinides, "Rover: Rtl optimization via verified e-graph rewriting," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[56] S. Coward, T. Drane, E. Morini, and G. A. Constantinides, "Combining power and arithmetic optimization via datapath rewriting," in *2024 IEEE 31st Symposium on Computer Arithmetic (ARITH)*. IEEE, 2024, pp. 24–31.

[57] E. Ustun, I. San, J. Yin, C. Yu, and Z. Zhang, "Impress: Large integer multiplication expression rewriting for fpga hls," in *2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2022, pp. 1–10.

[58] C. Chen, G. Hu, D. Zuo, C. Yu, Y. Ma, and H. Zhang, "E-syn: E-graph rewriting with technology-aware cost functions for logic synthesis," 2024. [Online]. Available: https://arxiv.org/abs/2403.14242

[59] H. K. Lee and D. S. Ha, "Hope: an efficient parallel fault simulator for synchronous sequential circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1048–1058, 1996.

[60] C. Wolf, J. Glaser, and J. Kepler, "Yosys-a free verilog synthesis suite," in *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*, vol. 97, 2013.

[61] R. Brayton and A. Mishchenko, "Abc: An academic industrial-strength verification tool," in *Computer Aided Verification: 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings 22*. Springer, 2010, pp. 24–40.

[62] B. Lisanke, "Logic synthesis and optimization benchmarks," *Microelectronics Center of North Carolina, Tech. Rep*, 1988.

[63] S. Yang, "Logic synthesis and optimization benchmarks," Version 3.0. Tech. Report. Microelectronics Center of North Carolina, Tech. Rep., 1991.

[64] F. Corno, M. S. Reorda, and G. Squillero, "Rt-level itc'99 benchmarks and first atpg results," *IEEE Design & Test of computers*, vol. 17, no. 3, pp. 44–53, 2000.

[65] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *1989 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1989, pp. 1929–1934.

[66] A. Mahzoon, D. Große, and R. Drechsler, "Genmul: Generating architecturally complex multipliers to challenge formal verification tools," in *Recent Findings in Boolean Techniques: Selected Papers from the 14th International Workshop on Boolean Problems*. Springer, 2021, pp. 177–191.

[67] S. Mitra, N. R. Saxena, and E. J. McCluskey, "Efficient design diversity estimation for combinational circuits," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1483–1492, 2004.

[68] A. T. Sheikh, A. H. El-Maleh, M. E. S. Elrabaa, and S. M. Sait, "A fault tolerance technique for combinational circuits based on selective-transistor redundancy," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 1, pp. 224–237, 2017.

[69] E. M. S. K. J. Singh, L. L. C. M. R. Murgai, and R. K. B. A. Sangiovanni-Vincentelli, "Sis: A system for sequential circuit synthesis," *University of California, Berkeley*, vol. 94720, p. 4, 1992.