SPARSE: Spatially Aware LFI Resilient State Machine Encoding

Muhtadi Choudhury
University of Florida
USA

Shahin Tajik
Worcetser Polytechnic Institute
USA

Domenic Forte
University of Florida
USA

ABSTRACT

As finite state machines (FSMs) control the behavior of sequential circuits, they can be a target for attacks. With laser-based fault injection (LFI), an adversary may attain unauthorized access to sensitive states by altering the values of individual state flip-flops (FFs). Although standard error correction/detection techniques improve FSM resiliency, all states and FFs of an FSM are assumed equally critical to protect, incurring significant overhead. In this paper, we introduce a novel spatial vulnerability metric to aid the security analysis, which precisely manifests the susceptibility of FSM designs to LFI based on state FF sensitivity and placement. A novel encoding and spatially aware physical design framework (SPARSE) are then proposed that co-optimize the FSM encoding and state FF placement to minimize LFI susceptibility. SPARSE's encoding uses the minimum number of FFs by placing security-sensitive FFs a sufficient distance apart from other FFs. SPARSE is demonstrated on 5 benchmarks using commercial CAD tools and outperforms other FSM encoding schemes in terms of security, area, and PDP.

1 INTRODUCTION

The confidentiality and integrity of trusted environments and cryptographic hardware can be threatened by physical attacks, e.g., side-channel, fault injection, and probing attacks [5, 23]. Fault injection attacks are particularly powerful due to their inexpensive and active nature; examples include changing the input voltage level, inducing clock signal irregularities, electromagnetic irradiation, or high temperature. Among different classes of fault injection attacks, laser fault injection (LFI) attacks are the most sophisticated and powerful ones in terms of precision and effectiveness [18, 21, 22].The primary target of LFI attacks are the state elements in the circuit design, such as flip-flops (FFs) and SRAM cells. These indispensable state elements are the essential core of finite state machines (FSMs) that control important functionalities of IPs in an SoC, including different crypto and memory cores. Attackers can use LFI to bypass specific states known as *authorized states*, and gain access to the *protected states* [13]. This can lead to unauthorized access to a service, extracting a secret key, and skipping security-critical instructions.

To mitigate the drawbacks of conventional FSMs against LFI, various classes of countermeasures have been put forward. Physical countermeasures, such as hardware sensors and tamper-proof packaging, can help but their inherent cost and extra manufacturing steps make logical circuit-based countermeasures more attractive. CAD tools can automatically add logical countermeasures to improve FSM resiliency. For example, through FSM redundancy (duplication, time-shifted output, etc.) or security-aware encoding [6, 13], an FSM can detect or prevent LFI. The primary disadvantage of adopting fault resilient coding techniques is their substantial overhead in terms of chip area, power consumption, and performance penalties [3, 7, 12]. The source of high overhead is their inefficacy to distinguish the sensitivity degrees of certain FSM states and state flip-flops (FFs). Despite the possibility of a small number of states or FFs being crucial, the conventional encoding techniques protect all states and FFs *uniformly*. The underlying reason is that these schemes were originally designed for communication systems, where all states (i.e., transmitted messages as electronic signals) and FFs (i.e., individual bits in a message) have to be protected from channel noise. Nevertheless, a substantial number of states and FFs may not be crucial in an FSM and cannot be exploited to efficiently disclose critical information.

Recently, a decision diagram-based approach, PATRON, was shown to provide a more efficient LFI resistant encoding for arbitrary FSM sizes and number of lasers [6]. PATRON consolidates the number of required normal and sensitive states, minimum encoding length, and attacker's fault capability, and explores their interrelationships. Additionally, the *vulnerability metric (VM)* demonstrated FSM susceptibility. However, PATRON as well as the VM metric do not consider the relevant FF placements. Rather it is presumed that any desired number of laser spots can concurrently inflict successful fault(s) at the corresponding locations of a typical chip, which may not be physically possible. In reality, LFI attack is constrained by various factors such as target technology, spot size, laser power, and feasibility of accommodating multiple lasers. Hence, there is no guarantee that any desired number of laser spots can simultaneously effectuate successful fault(s) at critical chip locations.

The key question then arises as to whether there exists a scalable, low overhead scheme that allots protection according to the FSM specification, LFI attack parameters, *and* chip layout. To this end, in this paper we define an improved threat model and introduce a novel *spatial vulnerability metric* (SVM) that captures precise FSM susceptibility to LFI bit flips from state FF placement. Then, we propose *SPARSE*, a design framework that uses binary decision diagrams (BDDs) to generate the high-level encoding bits that are then translated into the corresponding state FFs in the design layout using integer linear programming (ILP). For any arbitrary FSM, encoding and placement are *co-optimized* in terms of overhead and security for a multi-laser equipped adversary. SPARSE is integrated with other CAD tools (see Fig. 1). It is shown that SVM identifies vulnerabilities missed by the VM metric. Comparing our results with traditional and security-aware encoding techniques, SPARSE achieves the highest security while offering comparable, and in most cases, even better power-delay-product (PDP) and area.

Compared to other approaches, SPARSE reduces the number of FFs needed to securely encode an FSM design (and thus overhead) by *exploiting a new dimension* – spatial distance between FFs in the layout. SPARSE is parameterized to securely encode and arrange FFs for any number of lasers and any beam diameter(s). Specifically, as the number of FFs is minimized, a highly-encoded state assignment is manifested, minimizing the number of illegal states [8]. *To the best of our knowledge, this is first paper in the literature to co-optimize a design at RTL and layout levels for security purposes.*

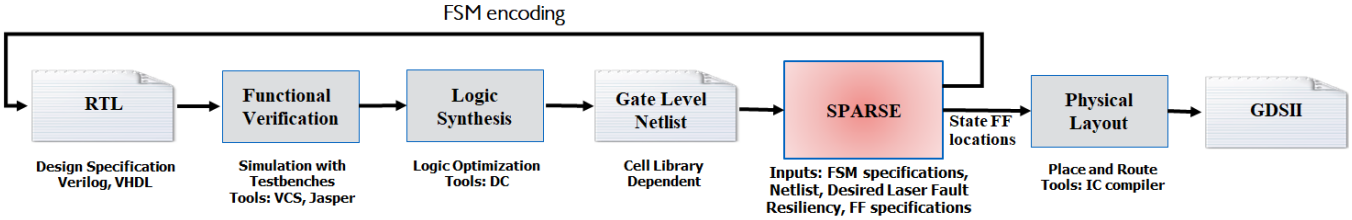


Figure 1: SPARSE integration with traditional ASIC flow and associated tools.

The rest of this paper is organized as follows. In the next section, we discuss basic notation and terms, FSMs, and fault injection. In Section 3, the LFI threat model is introduced and the impact of flip-flop placement is illustrated. Section 4 describes the proposed framework, including our new metric and optimization procedures. Results are presented and discussed in Section 5. Finally, conclusions and future work are given in the last section.

2 BACKGROUND

2.1 Basic Notation and Common Terms

We use blackboard bold fonts with upper case letters, e.g., $\mathbb{S}$, to represent sets. Upper case and italic font with one or no subscripts denote an element of a set, e.g., S_i or S. Vectors are denoted by lower case with arrow, e.g., $\vec{v}$, while elements of a vector are written in lower case with a subscript, e.g., v_i. The shorthand notation for a subset of elements i to j of a vector $\vec{v}$ is $\vec{v}_{i:j}$. A relationship between the ith and jth elements is denoted with a subscript ij. The operator $|\cdot|$ denotes the cardinality of a set or vector. Scalar variables may be upper case or lower case with italic fonts.

2.2 Finite State Machine (FSM) and Encoding

An FSM is defined as a 5-tuple $(\mathbb{S}, \mathbb{I}, \mathbb{O}, \varphi, \lambda)$, where $\mathbb{S}$ is a finite set of states, $\mathbb{I}$ is a finite set of input symbols, $\mathbb{O}$ is a finite set of output symbols, φ is the next-state function, and λ is the output function. $|\mathbb{S}|$ is the number of states in the FSM. Typically, an FSM is depicted as a directed graph $\mathcal{G} = (\mathbb{S}, \mathbb{T})$ where each state $S \in \mathbb{S}$ represents a vertex and each edge $T_{ij} \in \mathbb{T}$ represents a transition or edge from state S_i to the state S_j.

In the FSM, each state should only be accessed from its *accessible set of states*, i.e., $A(S_j) = \{S_i \mid T_{ij} \in \mathbb{T}\}$. In [14], a designer specifies a set $\mathbb{P}$ of *protected states* and a set $\mathbb{AU}$ of *authorized states*. A state $S \in \mathbb{AU}$ is allowed access to $\mathbb{P}$, such that $A(\mathbb{AU}) = \{P \mid P \in \mathbb{P}\}$. If $S \notin \mathbb{AU}$ accesses $\mathbb{P}$, there is a potential vulnerability in the FSM [13]. Thus, one can define two sets of states

$$\mathbb{NS} = \{S \in \mathbb{S} \mid s \notin \mathbb{AU} \cup \mathbb{P}\} \quad (1)$$

$$\mathbb{SS} = \{S \in \mathbb{S} \mid s \in \mathbb{AU} \cup \mathbb{P}\} \quad (2)$$

where $\mathbb{NS}$ and $\mathbb{SS}$ denote the sets of *normal states* and *sensitive states*, respectively.

FSM state encoding assigns a distinct binary pattern $\in \{0,1\}^n$ for each state where n is the number of state flip-flops (FFs). There are two traditional encoding techniques. In *binary encoding*, states are assigned in a binary sequence starting from 0. In *one-hot encoding*, only one bit of the state variable is allowed to be '1' while all others are set at '0' for every state in the FSM. For this scheme, $n = |\mathbb{S}|$, where the total number of FFs is understandably greater than that of binary encoding. Nevertheless, the combinational decoding logic is often simpler than binary. It should be noted here that these traditional FSM encoding schemes are security-unaware and thus unsafe against fault injection.

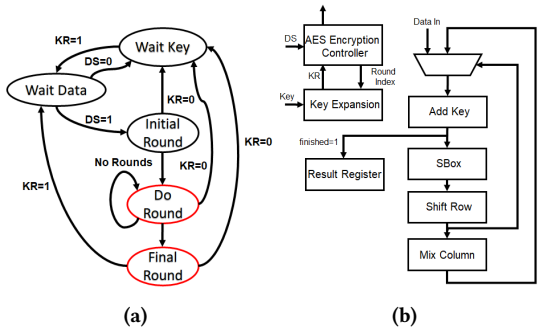


Figure 2: AES encryption module [1]. (a) FSM of AES encryption controller; red nodes represent sensitive FSM states ($S \in \mathbb{SS}$); (b) Encryption data path. KR and DS signals stand for Key Ready and Data Stable, respectively.

2.3 Fault Injection Against FSMs

The potency of fault injection (FI) attacks is illustrated using the controller circuit of an AES encryption module. The data path of the AES encryption algorithm (Fig. 2(a)) is implemented by the state transition diagram of the FSM (Fig. 2(b)). The FSM consists of 5 states: 'Wait Data', 'Wait Key', 'Initial Round', 'Do Round' and 'Final Round'. During the ten rounds of AES encryption, each states modifies the data path. After reaching 'Final Round', the control signal $finished = 1$ triggers the dispatch of the 'Add Key' module result (i.e., final ciphertext) to the 'Result Register'. For this FSM, if an attacker can access the 'Final Round' by bypassing the 'Do Round' state then premature results will be dispatched. For instance, if 'Final Round' state is accessed after 'Initial Round', $key \oplus plaintext$ will be dispatched instead of the ciphertext and the encryption key is easily extracted. Thus, $\mathbb{SS}$ ={'Do Round', 'Final Round'}.

3 THREAT MODEL

We delineate the scope of our threat model using the consolidated fault models and definitions from recent work [16, 17]. Any circuit can be decomposed into combinational logic (CL) gates $\eth_c$ and state elements $\eth_s = \{FF\}$. All the valid gates in one set becomes, $\eth = \eth_c \cap \eth_s$. An attacker can be described by a function $\zeta(f, t, l)$ where

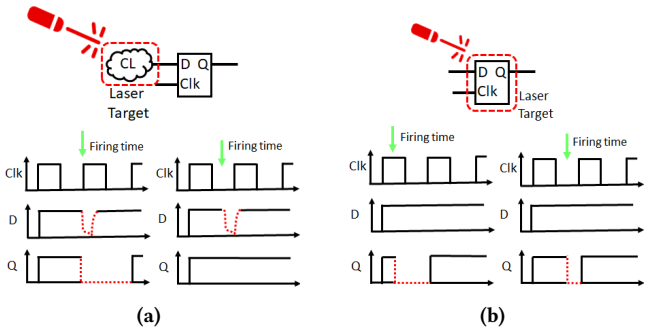


Figure 3: LFI on (a) combinational (CL) and (b) flip flop (FF). Irrespective of the LFI time or inputs, direct target of FF is shown to always flip, unlike targeting CL.

f is the total number of fault events (spatial and temporal components), t represents the fault types (bit flip, reset, and set), and l denotes the fault location(s) in a digital logic circuit. A net in the circuit suffers from a bit set (or bit reset) if it can be changed only from 0 to 1 (or 1 to 0) whereas bit flip model inverts the net's value regardless of its current value. Regarding f, any fault injection might be limited in spatial or temporal dimensions (*univariate* or *multivariate*). Spatially, the attacker may be limited in the number of fault injections that can be caused concurrently in the same clock cycle. Univariate fault injections only consider fault events occurring in the same clock cycle, while for multivariate fault injections, fault events can occur in different clock cycles.

3.1 FSM LFI Threat Model

The threat model for SPARSE and its justification is as follows

(1) *Targets:* In our threat model, we restrict the target gate set to state elements, i.e., $\eth = \eth_s = \{FF\}$ because they are the lowest hanging fruit in LFI attacks against FSMs. In combinational logic, a laser strike will only be successful for a specific input pattern and in a period just prior to the clock transitioning, where the period depends on the amount of charge deposited by the laser as shown in Fig. 3(a). In addition, the propagation time to latch the fault means that the laser synchronization component of the attacker has to be impeccable. On the other hand, overturning a FF state by striking the FF directly does not depend on successful propagation and/or timing of the laser within a clock cycle (see Fig. 3(b)).

(2) *Spatial dimensions:* We assume that the adversary can accurately inject $f(y, z)$ number of faults into the FSM based on the (y, z) coordinates of the laser beam(s), the laser power employed, and the FFs in the layout [2]. For modern technology nodes where the laser beam diameter D is much larger than minimum feature size, it is possible for one laser to flip multiple FFs if they are close to each other. Further, $f(y, z)$ can be increased by incorporating more simultaneous lasers into the attack setup (i.e., increasing x in this paper's notation). *The laser beam diameter D can also be increased, but this lowers the laser power desnity and thus the probability of faults* [20].

(3) *Temporal dimensions:* Following the above notation, $\zeta(f, \tau_{bf}, \eth)$ represent any number of fault(s) in *univariate*, bit-flip model on any FF locations in the circuit design. Bit set and reset models are left for future work.

(4) *Knowledge of FSM and design layout:* The attacker knows the FSM encoding, protected states $\mathbb{P}$, chip layout, etc. through reverse-engineering, an insider, or the design specification.

The above threat model is general and capable of handling LFI in current and future attack setups. Today, the number of lasers is typically restricted to two (i.e., $x \leq 2$). Although we stick to $x \leq 2$ in later experiments, our proposed countermeasure (SPARSE) and its underlying mathematics also apply for $x > 2$. Also, our proposed method is able to handle any beam diameter D, although we keep the laser beam diameter to its minimum for simplicity.

3.2 Importance of State Encoding and Physical Design (PATRON vs. SPARSE)

Traditional state encoding schemes (binary and one-hot) are much more vulnerable to laser fault injection than an LFI-resilient scheme like PATRON [6]. Thus, we mainly compare our threat model and proposed countermeasure to PATRON. However, more schemes will be quantitatively compared in our experiments (see Section 5).

There is one major, salient difference between the above threat model and the one used in PATRON. That is, our threat model incorporates the physical layout in order to determine the number of concurrent faults possible in the FSM. In other words, in [6], the number of faults f equals the number of lasers x whereas in this paper f depends on the FF and laser(s) (y, z) coordinates. A vulnerability in PATRON's threat model to x lasers is captured by the vulnerability metric (VM)

$$VM(x) = \frac{|\mathbb{VS}(x)|}{|\mathbb{S}|}. \tag{3}$$

Intuitively, VM is the percentage of states where $x = f$ faults can lead to a sensitive states. In the numerator, $\mathbb{VS}(x)$ denotes the set of vulnerable states susceptible to $x = f$ laser faults. $\mathbb{VS}(x)$ can be obtained by calculating the Hamming distance (HD) between sensitive state and other states in the FSM. Those with HD $\leq x$ are vulnerable. Since $x \neq f$ in our threat model, VM can miss some vulnerabilities. Below are examples to show how PATRON's model and VM are limited.

Consider an FSM containing $n = 4$ (number of FFs), sensitive state, $SS = \{0000\}$, and an attack capability of $x = 1$. In PATRON, this setting would mean that only one fault can be injected into one FF in any clock cycle. In this case, $\mathbb{VS}(1) = \{1000,\ 0100,\ 0010,\ 0001\}$ since these state encodings are all $HD = 1$ away from $SS = \{0000\}$. Without accounting for layout, PATRON seemingly prevents an LFI attack and $VM = 0$ whenever $\mathbb{VS}(1) \cap \mathbb{NS} = \emptyset$, where $\mathbb{NS}$ is the normal states set.

Fig. 4 provides different layouts for this FSM and illustrates the ineffectiveness of the VM metric to capture LFI vulnerability as well as the inefficiency of PATRON to address certain cell arrangements in design layout. The placement of the above FSM by a place and route tool shown in Fig. 4(a) allows the attacker to *inject 4 faults with just 1 laser* when its (y, z) position is near the center of the FF cluster. Here, even though PATRON's $VM = 0$, the attacker can overturn any combination of all 4 FFs (i.e., $f = 4$ even with $x = 1$), and subsequently SS is not protected. A possible fix, according to

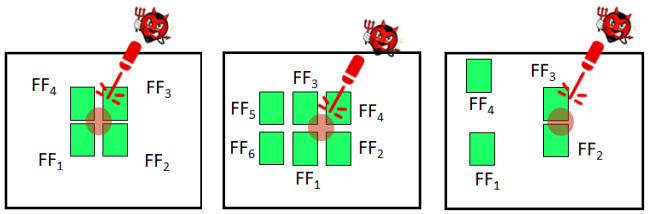


Figure 4: Vulnerability introduced due to FF placement; (a) Example FF layout where $x = 1$ and $VM = 0$ is illusory; (b) Possible countermeasure in PATRON [6] where $x = 4$ and number of FFs is increased, utilizing excessive resources; (c) A more efficient placement that also maintains security.

PATRON's encoding approach, is to increase n (total number of FFs) to 4 to match the worst case number of faults ($f = 4$), and find a secure encoding. An example placement with $n = 6$ is shown in Fig. 4(b). However, both area and power consumption increase with n and the FF decoding logic, which is undesired. More FFs also translate to more don't care states which can result in additional vulnerabilities [14].

Fig. 4(c) illustrates the approach that will be taken by our proposed SPARSE method. Here, a different arrangement of the 4 FFs in the layout is used as compared to (a). By exploiting this *new dimension (placement)*, only two FFs (FF_2 and FF_3) can be overturned individually or together by a single laser. Thus, $\mathbb{VS}(1) = \{1000,\ 0100,\ 0010,\ 0001,\ 0110\}$. As long as $0110 \notin \mathbb{NS}$, transitions to SS is still protected under SPARSE's threat model *without increasing the number of FFs.*

4 SPARSE METHODOLOGY

4.1 Spatial Vulnerability Metric (SVM)

In this section, we introduce the spatial vulnerability metric (SVM). SVM, unlike VM, calculates the percentage of states where x lasers can result in f faults that lead to unauthorized transitions to a sensitive state based on the (y, z) coordinates of the FFs in the physical layout. In realistic designs, there might be area constraints as designers require more flexibility for portability and power efficiency. As concurrent overturning of the required number of FFs by x laser spots may not always be possible due to spatial, temporal or other technical reasons, investigation into the less conservative scenario, where the concept of a few critical FFs out of the total FFs in a design (n) is warranted. To this end, the relative FF inter-distance information must be incorporated into the vulnerability estimation, which is lacking in VM.

We begin by defining some additional notation and with an illustrative example. For a given set of FFs in an FSM, $\mathbb{FF} = \{FF_1, \ldots, FF_n\}$. We denote the laser location matrix $L = [\vec{l}_1, \vec{l}_2, \ldots, \vec{l}_x]$ where location $\vec{l}_i = [y_i, z_i]$ represents the y and z coordinates of the ith laser. The *power set*, $\mathcal{P}(\mathbb{FF})$, is the set of all possible collections of FFs, expressed in sets. We define a *vulnerable FF set* as a set of FFs that can be attacked by x lasers pointed in their vicinities.

Fig. 5 shows an example layout of an FSM with arbitrary arrangement of $n = 4$ FFs. For simplicity, we assume that the power density of a laser spot is uniformly distributed across the diameter,

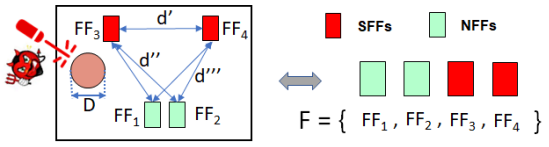


Figure 5: FF arrangement in an example design layout. (left) The laser beam with diameter, D, can only affect FFs in close proximity to each other; (right) Sensitive and Normal FFs (SFFs and NFFs). d' represents the SFF inter-distance whereas d'', and d''' represent the set of inter-distances from each of the SFFs (FF3, and FF4) to the NFFs, respectively.

D. We also assume $SS = \{0000\}$ for this example. In Fig. 5, for $x = 1$, D is smaller than each of the FF inter-distances, d', d'' or d''', meaning that a single laser is unable to flip more than one of their values at once for FF_3, and FF_4. However, FF_1 is close enough to FF_2 such that one laser strike can affect two of them at once. In this case, all the *combinations of vulnerable FF sets*, $\mathbb{E}(x = 1) = \{\{FF_1\}, \{FF_2\}, \{FF_3\}, \{FF_4\}, \{FF_1, FF_2\}\}$. Each set in $\mathbb{E}(x = 1)$ corresponds to a different position of the laser and the number of injected faults f is captured by counting the number of FFs in each set, e.g., $f = 1$ for the first four sets and $f = 2$ for the last set. In general, $\mathbb{E}(x) \subseteq \mathcal{P}(F)$ represents the set of vulnerable FF combinations for any L. If all possible attack scenarios are represented by the function, $\beta\colon L \rightarrow \mathbb{E}(x)$, the set of FF fault combinations in one clock cycle is represented by $\{\mathbb{V}(l_i) \mid l_i \in L\} \subseteq \mathbb{E}(x)$. For the above example, $\mathbb{V} = \{0001, 0010, 0100, 1000, 1100\}$.

Using these definitions, we can provide an expression for $\mathbb{SVS}(x)$, which is an analogue to $\mathbb{VS}(x)$ in Equation (3). $\mathbb{SVS}(x)$ is the spatially vulnerable states set susceptible to x lasers *considering the state FF layout.* Given a set of normal states $\mathbb{NS} \in \mathbb{S}$,

$$\mathbb{SVS}(x) = \mathbb{NS} \cap \left\{ \bigcup_{SS \in \mathbb{SS}} \bigcup_{V \in \mathbb{V}} V \oplus SS \right\}. \tag{4}$$

Then, the SVM metric is defined as

$$SVM(x) = \frac{|\mathbb{SVS}(x)|}{|\mathbb{S}|} \tag{5}$$

The goal for a spatially aware LFI tolerant scheme is $\mathbb{SVS}(x) = \emptyset$ in the numerator resulting in $SVM(x) = 0$. Intuitively, $SVM(x) > VM(x)$ signifies that at least one pair of FFs can be flipped with one laser spot in one clock cycle (i.e., $f > x$) for at least one set of laser coordinates.

4.2 Sensitive FFs and Normal FFs

Given the threat model and above discussion underscoring the importance of the FF spatial distance, we incorporate two additional definitions which will be helpful in our framework later. $\mathbb{SFF}$ and $\mathbb{NFF}$ are the sets of *sensitive FFs* and *normal FFs*, respectively. The spatial distances for each element in the $\mathbb{SFF}$ set is greater than the spot size diameter, D, from all $\mathbb{FF}$ so that a laser beam cannot overturn more than one SFF in one clock cycle. The FFs in $\mathbb{FF}$ with no spatial distance restrictions between them are known as $NFFs$ and depending on the circuit technology, spatial characteristics of the layout and technology library a laser spot may overturn one or more NFFs in one clock cycle. The righthand side of Fig. 5

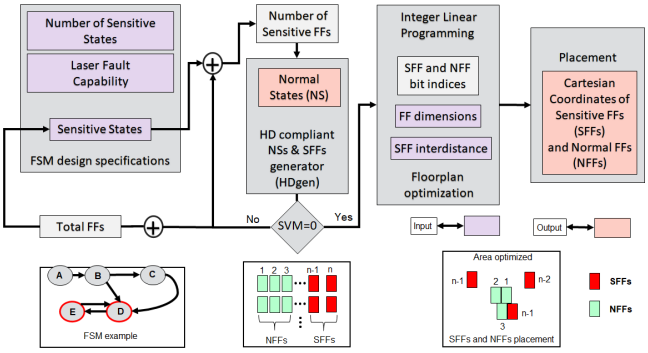


Figure 6: SPARSE block diagram. The bottom portion shows the resulting output of each block: FSM with sensitive states (red), encoding with FFs specified as normal (teal) and sensitive (red), and LFI resistant placement.

depicts the *SFF*s and *NFF*s in red and teal colors. Elaborating on the example in Section 4.1, as the state {1100} can be overturned to access the $SS = \{0000\}$ according to the arrangement of the $\mathbb{NFF}$ in the layout, hence $\{FF_1, FF_2\}$ is included in $\mathbb{E}(x = 1)$. The reason that these two FF types are introduced is to ensure that a maximum of 1 single bit can be flipped for a subset of $\mathbb{FF}$ with 1 laser each. In this way, even if there are multiple lasers, $|\mathbb{SFF}|$ could be adjusted so that the HD between normal and sensitive states is *always kept higher* than the attacker's fault capability; the presence of $\mathbb{NFF}$ facilitates area optimization in the design.

The ratio $r = \frac{|\mathbb{SFF}|}{|\mathbb{NFF}|}$ adjusts the allowable normal states ($|\mathbb{NS}|$) for a specific design. Intuitively, r provides different layout arrangements based on the number of sensitive flip-flops ($|\mathbb{SFF}|$). The $|\mathbb{NS}|$ becomes 0 for certain values of r, in which case the ratio needs to be increased if the FSM contains normal states. Hence, the specific FSM design parameters, $|\mathbb{SFF}|$, $|\mathbb{NFF}|$, $|\mathbb{SS}|$, $|\mathbb{NS}|$, and desired x can be made to appropriately match the required design as only a few transitions may be critical to the design in practice. So at the expense of a higher value of r, more allowable $|\mathbb{NS}|$ can be obtained. In the event that increasing $|\mathbb{SFF}|$ does not meet the security requirements, the number of FFs (n) must be increased.

4.3 SPARSE Encoding Framework

In this section, we describe our framework called SPARSE (SPatially Aware LFI Resilient State Machine Encoding) which makes the FSM inherently tolerant to bit flips from LFI. The goals of SPARSE are to find a flexible encoding (optimal n and $|\mathbb{SFF}|$) for which $SVM(x) = 0$ according to the FSM design specifications, and generate placement information of the $\mathbb{FF}$ in the FSM. Integer Linear Programming (ILP) is employed to minimize the area overhead. The block diagram for SPARSE is shown in Fig. 6.

FSM design specifications consists of the FSM specification (mainly, $|\mathbb{SS}|$ and n are important), the attacker's expected capability (number of lasers x) and sensitive state encoding initialized by the designer. As SPARSE iterates, the sensitive state encoding may be updated in this block through feedback from later blocks.

The adjacent block, *HDgen*, is responsible for generating normal state encoding and identifying the sensitive FF set, $\mathbb{SFF}$ that ensure $SVM(x) = 0$. If this constraint cannot be satisfied, HDGen has two feedback paths – $|\mathbb{SFF}|$ route (bottom to top) and n route (right to left). The first feedback path increases $r = \frac{|\mathbb{SFF}|}{|\mathbb{NFF}|}$ and then re-checks if the FSM specification constraint (i.e., $|\mathbb{NS}| + |\mathbb{SS}| \leq 2^n - 1$) is satisfied. If not, then n is increased and the sensitive state encoding is updated accordingly in the second feedback path. Implicitly, the minimum n and ratio r means that only the required sensitive transitions are catered to by the framework, thereby minimizing $|\mathbb{FF}|$ and overhead needed for LFI resistance. In this way, by utilizing the two feedback paths, the optimum $|\mathbb{SFF}|$, and $|\mathbb{NFF}|$ are generated. The internal procedures of HDgen are elaborated in Section 4.3.1.

In the *Floorplan optimization* block, the $\mathbb{SFF}$ and $\mathbb{NFF}$ encoding information are provided from HDGen. In addition, this block takes the FF dimensions (determined by the process design kit) and minimum *SFF* inter-distance (defined by the laser spot size D) as inputs. Then, integer linear programming (ILP) is used to determine the Cartesian coordinates of all the FFs in the FSM layout. That is, the FFs in $\mathbb{SFF}$ are spread a safe distance apart from each other and $\mathbb{NFF}$, while minimizing the area utilized, courtesy of the $\mathbb{NFF}$ conception. Floorplan optimization is discussed more in Section 4.3.2.

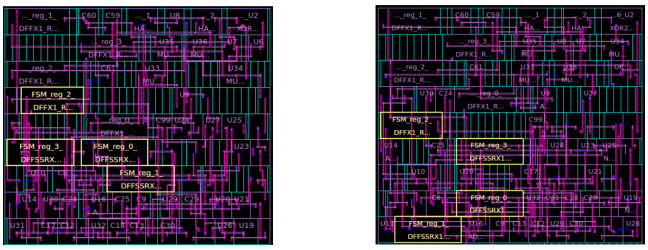
4.3.1 HDgen. The encoding of states in $\mathbb{NS}$ resistant to x lasers is found by solving a combinatorial problem. The encoding of each state NS in $\mathbb{NS}$ is represented as a vector of length n: $\vec{ns} = [ns_1, ns_2, \ldots ns_n], ns_i \in \{0, 1\}\ \forall\ i$. Similarly, the encoding of each state SS in $\mathbb{SS}$ is represented by $\vec{ss} = [ss_1, ss_2, \ldots ss_n], ss_i \in \{0, 1\}\ \forall\ i$. As a convention (without loss of generality), we assume that $m = |\mathbb{SFF}|$ rightmost bit positions of the vectors correspond to $\mathbb{SFF}$. For example, the $\mathbb{SFF}$ correspond to $[ns_{n-m+1}, \ldots, ns_n]$ (shorthand, $\vec{ns}_{n-m+1:n}$) and $[ss_{n-m+1}, \ldots, ss_n]$ (shorthand, $\vec{ss}_{n-m+1:n}$), for states in NS and SS, respectively. This is illustrated at the bottom center of Fig. 6 for $m = 2$. A Boolean function over these n variables should yield logic 1 for states in $\mathbb{NS}$ when HD of state encodings follows these general formulas

$$\mathbb{NS}_p = \{\vec{z} \in (0,1)^n : \bigcap_{SS_j \in \mathbb{SS}} HD(\vec{z}, \vec{ss}_j) > x\} \tag{6}$$

$$\mathbb{NS} = \{NS \in \mathbb{NS}_p : \exists\, SS \in \mathbb{SS}\ (HD(\vec{ns}_{n-m+1:n}, \vec{ss}_{n-m+1:n}) \geq x)\} \tag{7}$$

where $\vec{ss}_j$ is the encoding of state SS_j. $HD(\vec{a}, \vec{b})$ denotes the Hamming distance between binary vectors $\vec{a}$ and $\vec{b}$. Equation (6) is a preliminary step where all the normal state candidates are obtained, regardless of the layout. For example, if $n = 4$, $x = 1$, and $\mathbb{SS} = \{0000, 0101\}$, there can be as many as 8 states in $\mathbb{NS}$. For certain FSMs, exploring relatively higher n might be necessary to meet the constraints in which case solving these Boolean expressions might have scalability issues. Therefore, binary decision diagrams (BDDs) can be used to represent each of the Boolean functions that correspond to HD calculations for each state in $\mathbb{SS}$ in Equation (6). After obtaining a BDD for each state in $\mathbb{SS}$, the BDDs are ANDed together in order to get the intersected sets of combinations. In the above example, the set of $\mathbb{NS}_p$ candidates that are obtained with Equation (6) are the states consisting of $\{0011, 0110, 1001, 1010, 1011, 1100, 1110, 1111\}$ that are $HD > 1$ from states in $\mathbb{SS}$.

Equation (7) represents the subsequent step, where the results from preceding step are processed to subtract the states from $\mathbb{SVS}(x =$



(a) (b)

Figure 7: AES FSM layout for $x = 1$ in Synopsys ICC2 for different schemes; (a) Binary has two pairs of FFs precariously close to each other; (b) SPARSE ensures certain FF arrangements comply with security requirements.

1) in $\mathbb{NS}_p$ that are vulnerable in the layout. With $\mathbb{SFF} = \{FF_3, FF_4\}$, as shown in Fig. 5, the normal state candidates $\{1001, 1100\}$ are subtracted from $\mathbb{NS}_p$ as these states have $HD < 1$ between the $\mathbb{SS}$ bits corresponding to the SFFs. Specifically, for the bit positions corresponding to the $\mathbb{SFF}$, the HD between $SS \in \mathbb{SS}$ and $NS \in \mathbb{NS}$ candidates is compared and the non-compliant states are removed. In this way, secure state encoding is generated that ensures $SVM = 0$ according to the $\mathbb{FF}$ layout. Another example of SPARSE for $n = 4$, $x = 1$, and $\mathbb{SS} = \{0000, 0101, 0011\}$ according to Fig. 5 would have $\mathbb{NS} = \{0110, 1010, 1110\}$. It should be pointed out that SPARSE secures transitions between states in $\mathbb{SS}$, $\mathbb{SS}$ and $\mathbb{NS}$, as well as $\mathbb{NS}$ and $\mathbb{SS}$. The $\mathbb{FF}$ arrangement allows transitions between $\mathbb{NS}$ for area optimization. As mentioned in Section 3.1, although $x = 2$ is adequate to meet the modern LFI attack capacities, the above equations can ensure $SVM = 0$ for any value of x.

4.3.2 Floorplan Optimization (ILP). Floorplan optimization obtains the $\mathbb{SFF}$ and $\mathbb{NFF}$ placement with minimum area. As the circuit and the affiliated parameters, (e.g., n, FF inter-distances, etc.) are real integer numbers and our objective is area minimization, ILP is an appropriate choice [19]. For simplicity, we assume rigid shapes for all FFs, i.e., fixed width (w_i) and height (h_i) for the ith FF (FF_i) out of n FFs. The integer variables, y_i, and z_i represent the lower left vertex coordinates of FF_i. The unknown binary variables, $y_{ij}, z_{ij} \in \{0, 1\}$ denote the relative positional information of FFs i and j as shown in Table 1. W and H are the upper bounds of the floorplan width and height. Since area minimization is inherently nonlinear (width $\times$ height), we utilize an objective function where the floorplan's height (denoted Y) is minimized assuming an initial width that is iteratively reduced to generate the minimum floorplan area in square, i.e., the same minimum floorplan width and height. The floorplan's linear constraints are

$$y_i + w_i \leq W, \quad 1 \leq i \leq n \tag{8}$$

$$z_i + h_i \leq Y, \quad 1 \leq i \leq n \tag{9}$$

$$y_i, z_i \geq 0, \quad 1 \leq i \leq n \tag{10}$$

Equations (8) and (9) ensure that each FF is enclosed within the floorplan bounds while Equation (10) guarantees that the $\mathbb{FF}$ coordinates are non-negative integers. The constraints for ensuring appropriate inter-distances so that the $\mathbb{SFF}$ are a safe distance away

Table 1: Positional notation in ILP constraints.

$y_{i,j}$	$z_{i,j}$	Description
0	0	FF_i is on the left of FF_j
0	1	FF_i is below FF_j
1	0	FF_i is on the right of FF_j
1	1	FF_i is above FF_j

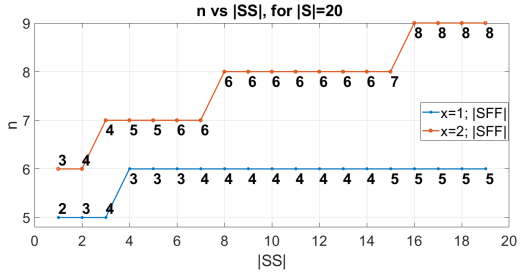


Figure 8: Effect of variation of $|\mathbb{SFF}|$ on the total number of FFs (n) and sensitive states ($|\mathbb{SS}|$) for different total states ($|\mathbb{S}|$). The bold number below each data point is $|\mathbb{SFF}|$.

from each other and $\mathbb{NFF}$ are

$$z_i + w_i \leq z_j + W(y_{ij} + z_{ij}) - D, \quad i \neq j \tag{11}$$

$$z_i - w_j \geq z_j - W(1 - y_{ij} + z_{ij}) + D, \quad i \neq j \tag{12}$$

$$y_i + h_i \leq y_j + H(1 + y_{ij} - z_{ij}) - D, \quad i \neq j \tag{13}$$

$$y_i - h_j \geq y_j - H(2 - y_{ij} - z_{ij}) + D, \quad i \neq j \tag{14}$$

where $n - m + 1 \leq i, j \leq n$, i.e., $FF_i \in \mathbb{SSF}$, $FF_j \in \mathbb{SSF} \bigcup \mathbb{NFF}$. The constant term D represents the laser beam diameter, which we consider a safe distance between FFs in $\mathbb{SFF}$, and $\mathbb{SFF}$ and $\mathbb{NFF}$. There is also an additional set of constraints to make sure that no $\mathbb{NFF}$ overlap with each other. For brevity, these constraints are not shown (the equations are essentially the same as Equations (11) to (14) except with $D = 0$, and $1 \leq i, j \leq n$, $FF_i, FF_j \in \mathbb{NFF}$).

There are numerous commercial and free ILP tools available that can solve this problem. To minimize the area, one can sweep W and solve the ILP problem multiple times, choosing the solution that results in smallest area ($W \times Y$). The resulting Cartesian coordinates of the FFs in $\mathbb{SFF}$ and $\mathbb{NFF}$ and the state encoding from HDGen ensure that $SVM(x) = 0$ and thus the FSM is LFI-resistant.

5 RESULTS AND DISCUSSION

In this section, we evaluate SPARSE and compare it to other FF encoding schemes. First, we examine the flexibility of SPARSE according to $r = \frac{|\mathbb{SFF}|}{|\mathbb{NFF}|}$ as the size of the FSM ($|\mathbb{S}|$ and number of sensitive states $|\mathbb{SS}|$) increase. Second, SPARSE is investigated on five controller benchmark circuits, namely AES, SHA, RSA, MIPS Processor, and Memory Controller in terms of power delay product (PDP), vulnerability metric (VM), spatial vulnerability metric SVM, and area according to LFI capability, x. All benchmark circuits are collected from OpenCores [1] and synthesized using Synopsys Design Compiler with 32-nm library. IC Compiler II (ICC2) is scripted to ensure proper $\mathbb{FF}$ spatial distance; the Cartesian coordinates of the $\mathbb{SFF}$ and $\mathbb{NFF}$ are received using the *get_attribute* command

which is then input to an in-house tool that calculates the corresponding $SVM(x)$. Two example layouts are shown in Fig. 7. For calculation of $SVM(x)$, laser spot diameter, D, is selected to be $1\mu m$ [4]. In our evaluation, laser diameter and technological node are kept close to contemporary values for simplicity.

The controller circuit of AES, SHA, and RSA contain FSMs with less than 8 states. The $|\mathbb{SS}|$ for AES, SHA-256, RSA, MIPS Processor, and Memory Controller are 2, 3, 4, 5 and 8 respectively. For AES, the states "Do Round" and "Final Round" and for SHA-256 the states "Data input", "Block next", and "Valid" are in *SS* [13]. For RSA among the seven states, "Result", "Square", "Multiply" and "Load2" are regarded as in $\mathbb{SS}$. SPARSE has the flexibility to consider all n FFs as sensitive FFs and all states as sensitive states, if intended by the designer. Since most cryptographic algorithms tend to have small number of states, the scalability of SPARSE is demonstrated on larger circuits like memory controller FSM with 66 states.

5.1 Flexibility of SPARSE

Fig. 8 illustrates the effect of increasing number of total states $|\mathbb{S}|$ and sensitive states $|\mathbb{SS}|$ on $r = \frac{|\mathbb{SFF}|}{|\mathbb{NFF}|}$ and n. A smaller r indicates that there are less $|\mathbb{NFF}|$ or more $|\mathbb{SFF}|$, which signifies more freedom in placement and less area overhead. $|\mathbb{S}|$ is varied in the range of 10 to 60, consistent with the size of FSMs in later benchmark results; however only 20 is shown due to space constraints. In Fig. 8, the bold numbers at each data point indicate $|\mathbb{SFF}|$. For all these experiments, $|\mathbb{SS}|$ of up to 20 is explored. For $x = 1$, it is seen that LFI resilience for $|\mathbb{SS}| = 2$ is initially met by increasing $|\mathbb{SFF}|$ *without increasing* n. Only if design requirement is still unmet does n need to be increased. For $x = 1$ and $|\mathbb{SS}| = 10$, $|\mathbb{SFF}|$ needs to be increased up to 4 while $n = 6$. For $x = 2$, as the HD requirement is higher, the $|\mathbb{SFF}|$ and n also need to be higher to meet the same $|\mathbb{SS}|$. Hence, the graph for $x = 2$ is vertically above $x = 1$. The same trends are visible for all $|\mathbb{S}|$. An interesting observation is that n saturates and r is smaller ($|\mathbb{NFF}| > |\mathbb{SFF}|$) for larger FSMs (larger $|\mathbb{S}|$). We utilize this phenomenon to obtain a secure, low overhead encoding as shown below.

5.2 Security, Overhead, and Scalability

Table 2 shows comprehensive results for security (VM and SVM) and overhead (normalized PDP and area) for the following spatially unaware encoding approaches as a comparison with SPARSE.

Binary / One-Hot*: Binary and one-hot encoding are the traditional FSM encoding approaches (described in Section 2), which are used as the baseline to compare overheads across all the other approaches. All the PDP and area values are normalized with PDP and area values of Binary Encoding. However, in [6] it is shown that for large benchmarks like Memory Controller, One-hot Encoding performs better than Binary owing to the simpler decoding logic. So, only for memory controller the PDP and area values are normalized with One-hot Encoding.

Codetable [9]: An $[n, k, d]$ linear code represents k bit messages using n bit codewords where two distinct codewords differ in at least d bits. The codetable represents bounds and construction of the linear code $[n, k, d]$ over Galois Field of q [9]. As we are considering only Boolean values with an objective to calculate the lowest $|\mathbb{FF}|$ achievable by this approach, $q = 2$, i.e., only *binary code*

Table 2: Power-delay product (PDP), vulnerability metric (VM), spatial vulnerability metric (SVM), and area analysis for different encoding schemes. PDP and area values are normalized with PDP and area of Binary Encoding except for asterisked which are normalized with One-Hot. Red and green denote vulnerable and non-vulnerable FSMs, respectively.

	Benchmarks	AES $\|\mathbb{S}\|$=5		SHA-256 $\|\mathbb{S}\|$=7		RSA $\|\mathbb{S}\|$=7		MIPS Processor $\|\mathbb{S}\|$=19		Memory Controller $\|\mathbb{S}\|$=66		Average Results	
	x	1	2	1	2	1	2	1	2	1	2	1	2
Binary / One hot*	PDP	1	1	1	1	1	1	1	1	1*	1*	1	1
	VM	0.6	1	0.4	0.9	0.4	0.9	0.3	0.7	0.1	0.4	0.36	0.78
	SVM	0.6	1	0.4	0.9	0.4	0.9	0.3	0.7	0.1	0.5	0.36	0.8
	Area	1	1	1	1	1	1	1	1	1	1	1	1
Codetable	PDP	1.4	2.3	1.2	1.5	1.6	2.6	1.3	1.9	0.93	1.1	1.29	1.88
	VM	0	0	0	0	0	0	0	0	0	0	0	0
	SVM	0.2	0	0.1	0	0.1	0.1	0.3	0.5	0.05	0	0.15	0.12
	Area	0.82	1.19	1.18	1.44	1.42	1.84	1.23	1.51	0.99	1.1	1.13	1.42
A. Nahiyan et al.	PDP	1.6	1.8	1.2	1.5	2.5	2.5	1.7	1.8	1.7	1.8	1.74	1.88
	VM	0.8	0.4	0.7	0.9	0.9	0.9	0.5	0.8	0.2	0.4	0.62	0.68
	SVM	0.8	0.4	0.7	0.9	0.9	0.9	0.6	0.8	0.5	0.6	0.7	0.72
	Area	0.96	0.97	1.21	1.23	1.81	1.81	1.48	1.48	1.7	1.71	1.43	1.44
PATRON	PDP	1.1	1.7	1.2	1.5	1.4	2.3	1.6	1.5	1	1	1.26	1.6
	VM	0	0	0	0	0	0	0	0	0	0	0	0
	SVM	0.2	0	0	0.1	0.1	0	0.1	0.2	0.02	0.03	0.08	0.07
	Area	0.82	1.01	1.07	1.43	1.42	1.87	1.25	1.35	1.03	1.03	1.12	1.34
SPARSE	PDP	1.1	1.8	1.1	1.4	1.4	2.3	1.4	1.4	1	1	1.2	1.58
	VM	0	0	0	0	0	0	0	0	0	0	0	0
	SVM	0	0	0	0	0	0	0	0	0	0	0	0
	Area	0.81	1	1.11	1.41	1.42	1.84	1.25	1.35	0.99	0.99	1.12	1.32

is examined. For this approach, all FSM states are considered in $\mathbb{SS}$ as there is no procedure to separate states in $\mathbb{NS}$ from $\mathbb{SS}$. Note that with this encoding approach the applicability for all the popular linear encoding such as Hamming (7,4), Extended Hamming, etc. that could be designed for the specific benchmarks parameters are also examined. With regard to nonlinear codes, robust and partially robust codes are seen as promising [10, 15]. However, both these codes have minimum HD of 1. This trait alone makes them inapplicable for LFI where HD flexibility is of utmost importance. Most of these codes also have relatively low code rate (a measure of inefficiency) [11]. Hence, nonlinear codes are disregarded here.

A. Nahiyan et al.: In [13], two fault resilient schemes against set-up time violations attacks, Scheme I and Scheme II, are proposed. Scheme I proposes $\log(|\mathbb{NS}|) + |\mathbb{P}|$ bits for encoding, where $|\mathbb{P}|$ upper bits are dedicated to one-hot scheme while it pads zero for the rest of $\log(|\mathbb{NS}|)$ bits for protected states. This scheme could be potentially useful against LFI according to different laser fault capabilities. Hence, Scheme I is compared in this work. Scheme II is not relevant to LFI as it requires prohibited transitions as input, which is out of the scope of this paper.

PATRON: This encoding scheme is an LFI resilient encoding approach [6] which is most closely related to this work. BDDs are used to generate these encodings for varying x. As discussed in Section 3.2, $x = f$ for PATRON and thus we expect lower security and/or higher overhead as compared to SPARSE.

5.2.1 Security Comparison. For Binary/One-Hot in AES, once the attacker has two lasers ($x = 2$), $VM(2)$ and $SVM(2)$ reach the maximum of 1, i.e., and all the encodings are unsafe. Of the approaches, except Binary/One-Hot and A. Nahiyan et al., all schemes deliver encoding that has $VM = 0$ by design. Although for Codetable and PATRON schemes, it is seen that $VM = 0$ while the corresponding $SVM \neq 0$, which signifies that at least one pair of SFFs are vulnerable in the design layout due to their placement. For example,

Codetable (AES, $x = 1$) cannot generate a safe encoding although all the 5 states have a minimum $HD = 2$ between the codewords. Since Codetable and PATRON do not account for FF placement, $SVM = 0$ likely occurs by chance and can be attributed to the place-and-route optimization flags and heuristics of ICC2 rather than these encodings per se. For the PATRON approach, another contributing factor is in how the designer decides to encode the sensitive states. The fact that $VM \neq SVM$ in many of the benchmarks illustrates why FF placement is so important in estimating FSM vulnerability to LFI.

Another observation is that with more FFs implemented in the FSM (larger n) it is more likely that $SVM > VM$ as the *probability* of a pair of FFs being placed precariously close together rises. At least, for A. Nahiyan et al. and Binary/ One-Hot this phenomenon is observed as inherently more vulnerable state encoding is in $\mathbb{VS}(x)$ set due to low HD resilient capability of these encoding. For A. Nahiyan et al. approach, the first case of $SVM > VM$ occurs when $|\mathbb{S}| = 19$, and for Binary/ One-Hot when $|\mathbb{S}| = 66$. Compared to Binary/One-Hot, A. Nahiyan et al. approach implements higher n which may contribute to more spatially vulnerable states in $|\mathbb{SVS}(x)|$. For Binary/One-Hot, the first case of $SVM > VM$ occurs only when n goes as high as 7 for a large benchmark, denoting additional vulnerable states due to FF placement.

For lower n values, there are instances for Codetable and PATRON where all the FFs are placed sufficiently apart from each other, signified by $VM = SVM$, which indicates no additional states are vulnerable due to FF placement. However, there is no guarantee that the FFs placement will always be sufficiently apart in the layout as evidenced by multiple instances of $SVM > VM$ in Codetable and PATRON columns. Note also that the incremental SVM values (Mem. Controller) for PATRON are not to be taken lightly as only a single state is necessary for a skilled attacker to extract sensitive information from the design. SPARSE does not rely on "luck" in placement – the VM and SVM are both guaranteed to be 0.

5.2.2 Overhead Comparison. As one would expect, Binary/One-Hot encoding always achieve the lowest PDP since they are only guided by traditional metrics, such as power and delay, rather than security. The approach with the next lowest PDP (on average) is SPARSE, which is excellent because it also guarantees security ($SVM = 0$). PATRON is next. The Codetable and A. Nahiyan approaches have the worst PDPs. The reason is that both use redundant FFs. As expected, the best codes provided by these approaches cannot meet the particular FSM parameters, hence the overhead.

In terms of area, SPARSE also has the lowest overall overhead compared to the other approaches, likely because it minimizes number of FFs needed as well as focus on area minimization in its optimization flow. The next is PATRON. Finally, area of Codetable and A. Nahiyan et al. are the worst.

Thus, SPARSE outperforms the other security-aware methods in terms of both security and overhead.

6 CONCLUSION AND FUTURE WORK

In this paper, we took a step towards optimizing the security of chip designs by bridging the gap between RTL and layout. Specifically, we examined the impacts of FSM encoding and FF placement on the security of FSMs to LFI. SPARSE was developed to co-optimize FSM encoding and FF placement according to technology node and LFI attacker's capabilities (number of lasers, spot size, etc.). SPARSE outperformed other FSM encoding schemes in terms of security, PDP, or area and in many cases all three. In future work, we hope to expand this concept to LFI-aware placement of combinational logic and less constrained "bit set" and "bit reset" models.

ACKNOWLEDGMENT

The authors thank Intel for their support of this research.

REFERENCES

[1] [n. d.]. Opencores https://opencores.org/. https://opencores.org/
[2] Michel Agoyan, Jean-Max Dutertre, Amir-Pasha Mirbaha, David Naccache, Anne-Lise Ribotta, and Assia Tria. 2010. How to flip a bit?. In *2010 IEEE 16th International On-Line Testing Symposium*. IEEE, 235–239.
[3] Douglas A Anderson and Gernot Metze. 1973. Design of totally self-checking check circuits for m-out-of-n codes. *IEEE Trans. Comput.* 100, 3 (1973), 263–269.
[4] Christian Boit, Shahin Tajik, Philipp Scholz, Elham Amini, Anne Beyreuther, Heiko Lohrke, and Jean-Pierre Seifert. 2016. From IC debug to hardware security risk: The power of backside access and optical interaction. In *International Symposium on the Physical and Failure Analysis of Integrated Circuits(IPFA)*. 365–369.
[5] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, and Sadeghi Ahmad-Reza Capkun, Srdjan. 2017. Software grand exposure:{SGX} cache attacks are practical. In *11th {USENIX} Workshop on Offensive Technologies*.
[6] Muhtadi Choudhury, Domenic Forte, and Shahin Tajik. 2021. PATRON: A Pragmatic Approach for Encoding Laser Fault Injection Resistant FSMs. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 569–574.
[7] Giovanni De Micheli, Robert K Brayton, and Alberto Sangiovanni-Vincentelli. 1985. Optimal state assignment for finite state machines. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 4, 3 (1985), 269–285.
[8] Steve Golson et al. 1994. State machine design techniques for Verilog and VHDL. *Synopsys Journal of High-Level Design* 9, 1-48 (1994).
[9] Markus Grassl. 2007. Bounds on the minimum distance of linear codes and quantum codes. Online available at http://www.codetables.de.
[10] Taubin Alexander Karpovsky, Mark. 2004. New class of nonlinear systematic error detecting codes. *IEEE Transactions on Information Theory* 50 (2004), 1818–1819.
[11] Zhen Karpovsky Mark G Kulikowski, Konrad Wang. 2008. Comparative analysis of robust fault attack resistant architectures for public and private cryptosystems. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 41–50.
[12] Subhasish Mitra, Ming Zhang, Saad Waqas, Norbert Seifert, Balkaran Gill, and Kee Sup Kim. 2006. Combinational logic soft error correction. In *2006 IEEE International Test Conference*. IEEE, 1–9.
[13] Adib Nahiyan, Farimah Farahmandi, Prabhat Mishra, Domenic Forte, and Mark Tehranipoor. 2018. Security-aware FSM design flow for identifying and mitigating vulnerabilities to fault attacks. *IEEE Transactions on Computer-aided design of integrated circuits and systems* 38, 6 (2018), 1003–1016.
[14] Adib Nahiyan, Kan Xiao, Kun Yang, Yeir Jin, Domenic Forte, and Mark Tehranipoor. 2016. AVFSM: A framework for identifying and mitigating vulnerabilities in FSMs. In *Proceedings of the 53rd Annual Design Automation Conference*. 1–6.
[15] Yaara Neumeier and Osnat Keren. 2012. Punctured Karpovsky-Taubin binary robust error detecting codes for cryptographic devices. In *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*. IEEE, 156–161.
[16] Jan Richter-Brockmann, Pascal Sasdrich, and Tim Güneysu. 2021. Revisiting Fault Adversary Models-Hardware Faults in Theory and Practice. *IACR Cryptol. ePrint Arch.* 2021 (2021), 296.
[17] Jan Richter-Brockmann, Aein Rezaei Shahmirzadi, Pascal Sasdrich, Amir Moradi, and Tim Güneysu. 2021. FIVER–Robust Verification of Countermeasures against Fault Injections. *IACR Cryptographic Hardware and Embedded Systems* (2021).
[18] Cyril Roscian, Alexandre Sarafianos, Jean-Max Dutertre, and Assia Tria. 2013. Fault model analysis of laser-induced faults in sram memory cells. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 89–98.
[19] Sadiq M Sait and Habib Youssef. 1999. *VLSI physical design automation: theory and practice*. Vol. 6. World Scientific.
[20] Falk Schellenberg, Markus Finkeldey, Nils Gerhardt, Martin Hofmann, Amir Moradi, and Christof Paar. 2016. Large laser spots and fault sensitivity analysis. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 203–208.
[21] Bodo Selmke, Stefan Brummer, Johann Heyszl, and Georg Sigl. 2015. Precise laser fault injections into 90 nm and 45 nm sram-cells. In *International Conference on Smart Card Research and Advanced Applications*. Springer, 193–205.
[22] Sergei Skorobogatov and Ross Anderson. 2002. Optical fault induction attacks. In *International workshop on cryptographic hardware systems*. Springer, 2–12.
[23] Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo. 2017. {CLKSCREW}: exposing the perils of security-oblivious energy management. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 1057–1074.