Lightweight Masked Hardware Gadgets through Verification-aware Optimization

David S. Koblah[1], Dev M. Mehta[2], Nathan Poch[2], Mohammad Hashemi[2], Fatemeh Ganji[2], and Domenic Forte[1]

[1] University of Florida, Gainesville, USA, dkoblah@ufl.edu, dforte@ece.ufl.edu
[2] Worcester Polytechnic Institute, Worcester, USA, dmmehta2@wpi.edu, nwpoch@wpi.edu, mhashemi@wpi.edu,fganji@wpi.edu

Abstract. Masked hardware design is a central countermeasure against side-channel analysis (SCA), but its practical deployment remains costly due to the overhead and design complexity of secure implementations. To manage this complexity, modern masked circuits are increasingly built from composable masked building blocks, referred to as gadgets. However, improving the implementation quality of such gadgets is challenging because optimization must not violate formal security requirements. This paper studies the optimization of masked hardware gadgets under formal security by comparing conventional CAD transformations with a security-aware design automation framework based on evolutionary algorithms. The conventional CAD flow explores structural alternatives exposed by technology mapping, FRAIG-based rewriting, and retiming, whereas the evolutionary framework adaptively explores alternative gate-level realizations through circuit-specific mutation, crossover, and multi-objective selection. The resulting implementations are evaluated for power and area while enforcing functional correctness and compliance with the probing model through equivalence checking and pre-silicon side-channel verification. Our results show that conventional CAD techniques can yield substantial improvements in selected cases, up to 60%, but their behavior is not systematic under strict masking constraints. In contrast, the evolutionary framework provides a controlled, adaptive search over alternative gate-level realizations and produces security-preserving implementations with measurable improvements over naïve synthesis, reaching up to 15% reduction in power and area in selected cases. We validate the optimized designs using industry-standard synthesis and pre-silicon side-channel verification tools. We further show, for AES, that additional savings remain possible even relative to a state-of-the-art compressed masked design.

Keywords: Side Channel Analysis · Masking · Evolutionary Algorithms · CAD.

1 Introduction

Side-channel vulnerabilities have been studied extensively since the seminal work of Kocher et al. demonstrated their impact on security-critical implementa-

tions [22]. In side-channel analysis (SCA), an adversary exploits physical effects of computation to extract sensitive information, with power consumption and electromagnetic (EM) emanation being among the most common sources of leakage [22,15]. Among the available countermeasures, *masking* has received particular attention due to its formal security foundation [10]. In Boolean masking, sensitive values are decomposed into randomized *shares*, and computation is carried out on these shares rather than on the secret itself [16].

Despite this strong theoretical basis, the practical deployment of masked hardware remains difficult. In particular, physical effects such as glitches, transitions, and coupling, together with architectural choices such as pipelining and parallelism, can invalidate security arguments that hold only at the abstract level [21]. To address this problem, modern masked hardware design increasingly relies on secure composable building blocks, referred to as *gadgets* [32]. Their secure composition, however, is non-trivial and depends on dedicated security notions. For instance, *probe-isolating non-interference* (PINI) enables efficient composition of multi-input/output gadgets and the trivial security of linear operations such as XOR [9].

This evolution toward gadget-based design raises an important CAD problem. Masked gadgets must be optimized with respect to implementation metrics such as power and area, but such optimization must not violate functional correctness or formal security requirements. Conventional EDA tools are highly effective at optimizing performance, power, and area, yet they do not treat masking security as a first-class design constraint [13]. Standard transformations such as technology mapping and retiming can yield great improvements in selected cases, but their behavior is not systematic under strict masking constraints (see Section 4). This trade-off is particularly relevant for lightweight side-channel-resilient hardware, where power and area must be reduced without sacrificing verifiability or security.

Several works have advanced the automated generation and improvement of masked hardware. AGEMA transforms unprotected cryptographic designs into masked circuits using masked gadgets as fundamental building blocks [21]. Subsequent work demonstrated that hand-crafted optimization can significantly improve selected AGEMA artifacts, for example, by reducing latency in AES S-box implementations [28]. Other efforts proposed improved masked nonlinear components [36] or extended automation to FPGA-oriented flows [29]. Recent work also explored the security-versus-PPA design space of masked implementations across technology libraries and threshold-voltage options [5]. However, these directions still leave an important gap. Prior work has largely focused on Boolean-level transformations, architectural refinements, or manual gadget improvement. In contrast, the problem of automatically optimizing *library-aware gate-level realizations* of masked gadgets under formal security verification remains largely unexplored. This gap is particularly relevant for ASIC-oriented masked design flows, where the final gate-level realization directly affects power and area, yet must continue to satisfy the composability guarantees required by gadget-based masking.

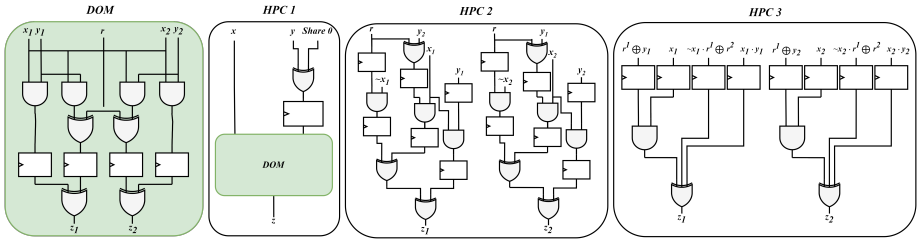

Fig. 1: DOM and representative HPC-style gadgets used as references in this work. Here, x, y, and z denote input and output shares, while r and *share 0* denote refresh bits. The subscripts indicate shares, and the superscripts indicate distinct refresh bits.

Contributions. In this work, we study masked hardware gadget optimization under formal security constraints by comparing conventional CAD transformations with an evolutionary search-based framework. The conventional CAD flow explores structural alternatives exposed by technology mapping, FRAIG-based rewriting, and retiming, whereas the evolutionary framework adaptively explores alternative gate-level realizations through circuit-specific mutation, crossover, and multi-objective selection. Candidate solutions are evaluated with respect to power and area using technology- and library-derived cell characteristics, while functional correctness and PINI security are enforced through equivalence checking and pre-silicon side-channel verification. To ensure the security of each optimized result, we rely on VERICA [33], a pre-silicon verification tool tailored to masking-oriented security notions.

Compared to prior work such as COMPRESS [7], which overlooks gadgets' lightweightness, we focus on power and area optimization under formal security constraints. Our goal is not to redesign the gadget logic at the Boolean or architectural level, but to identify improved gate-level implementations that remain compatible with technology library evaluation and formal verification. The central question is therefore not only which approach yields the largest reduction in isolation, but how conventional CAD transformations and adaptive evolutionary search differ under strict masking constraints. Our main contributions are as follows:

- We present a security-, library-aware gate-level optimization framework based on evolutionary search and evaluate it on representative masked gadget families against conventional CAD optimizations. The comparison characterizes the practical differences between conventional CAD transformations and adaptive circuit-specific exploration under functional and PINI-security constraints.
- We identify the practical limits of both approaches under strict masking constraints: conventional CAD can yield strong reductions in selected cases, but its behavior is not systematic across gadget families, whereas the proposed framework provides controllable and adaptive exploration of alternative gate-level realizations while retaining functional correctness and security.

– We show that the proposed framework can automatically produce functionally correct and PINI-secure gadget implementations with measurable improvements over naïve synthesis, reaching up to 15% reduction in power and area in selected cases, and that these gadget-level gains can translate to selected higher-level cryptographic designs. We further validate this on a larger cryptographic circuit, namely AES, and show that even when starting from a state-of-the-art compressed masked design produced by COMPRESS [7], additional savings are still achievable.

2 Background

2.1 Masking, gadgets, and security notions

Boolean masking. Masking aims to protect a sensitive computation at security order d under an assumed leakage model. In Boolean masking, a sensitive variable $x \in GF(2^m)$ is represented by $d+1$ shares $(x_1, \ldots, x_{d+1})$ such that $x = \bigoplus_{i=1}^{d+1} x_i$.

Computation is then performed on the shares rather than on x itself [18,16].

Probing security and hardware effects. The classical d-probing model formalizes an adversary that can observe up to d internal wires of a circuit [18]. A masked implementation is d-probing secure if these observations reveal no information about the underlying sensitive value. This abstraction is useful, but it is not sufficient for hardware. Physical implementations exhibit glitches, transitions, and coupling, which can recombine shares and invalidate security arguments that hold only at the abstract circuit level [12,21]. To capture these effects, the robust probing model extends the classical model using *extended probes*. In particular, a probe may expand to the stable sources contributing to the observed signal, including primary inputs and register outputs. This allows pre-silicon verification to account for glitch-induced and transition-induced leakage in masked hardware [12].

Gadgets and composability. Modern masked hardware is commonly built from secure masked building blocks, referred to as gadgets [32,21]. This design style reduces engineering complexity, but it introduces an additional requirement: individually secure gadgets must remain secure under composition. Classical notions such as NI and SNI address this issue only partially, especially for multi-output gadgets [3,4]. In this work, we focus on *probe-isolating non-interference* (PINI), which supports multi-output gadgets and enables trivial composition of linear operations such as XOR [9]. Our optimization framework modifies gate-level realizations of existing gadgets; as such, preserving PINI security is a central requirement.

Representative gadget families. To ground the optimization problem, we briefly recall the gadget families considered in our experiments. DOM multiplication is a widely used construction, but under the glitch-robust probing model, its security is reduced to NI, which limits secure composition under probe propagation [11,12]. Subsequent designs such as HPC1, HPC2, and HPC3 improve composability and robustness under stronger hardware-aware security notions [8].

These gadgets provide practically relevant optimization targets because they differ in latency, randomness, and structural complexity while targeting strong composability guarantees.

2.2 VERICA

VERICA is a pre-silicon verification tool for masked hardware that supports probing-based security notions, including NI, SNI, and PINI [33]. In our setting, its main role is to act as the security oracle inside the optimization flow. Given a candidate gadget implementation, VERICA checks whether the optimized netlist still satisfies the required masking notion under the selected probing model. This is essential because improvements in power or area are irrelevant if the resulting implementation is no longer composable or secure. A practical advantage of VERICA is its modular interface. Share annotations are provided externally through configuration files rather than by rewriting the netlist itself. As a result, multiple gate-level realizations of the same gadget can be verified with a consistent verification setup, making VERICA suitable for repeated use within an automated search loop.

2.3 Evolutionary Search

Our framework uses evolutionary search, specifically genetic algorithms (GAs), to optimize masked gadgets under multiple objectives. At a high level, the search starts from an initial population of candidate gate-level realizations, evaluates them using a fitness function, and iteratively applies selection, crossover, and mutation to generate improved candidates. The process terminates when a target threshold or a generation limit is reached. This class of methods is well-suited to our setting for two reasons. First, the search space is discrete, structural, and highly constrained by correctness and security requirements. Second, the optimization objectives, namely power and area, may conflict with one another and must be balanced without violating PINI security. For these reasons, evolutionary search provides a practical mechanism to explore alternative gate-level realizations while retaining explicit control over the optimization objectives.

3 Methodology

Our objective is to optimize masked hardware gadgets at the gate level in terms of power and area while retaining functional correctness and formal security. The input to the framework is an RTL implementation of a masked gadget, along with its share annotation, for pre-silicon verification. The output is an optimized gate-level realization that remains PINI-secure under verification. We further evaluate whether gadget-level improvements translate to improvements in larger cryptographic designs.

A key challenge is that conventional CAD transformations can be effective in selected cases, but their behavior remains highly dependent on the gadget

structure and optimization settings. Techniques such as technology mapping, FRAIG-based rewriting, and retiming can yield strong improvements for some gadget families (see Section 4.1), but they explore structural alternatives exposed by the selected synthesis and mapping transformations. In contrast, the evolutionary framework directly explores alternative gate-level realizations through circuit-specific mutation, crossover, and multi-objective selection. Masked gadgets are small and highly constrained, and security must be preserved throughout the process. As a result, a candidate that improves area or timing may still produce an unusable output or violate the conditions required for secure composition. Our objective is therefore not simply to maximize savings, but to provide a controllable and adaptive search process for exploring usable gate-level realizations under functional and security constraints.

3.1 Overview of the optimization flow

Our framework is designed for ASIC-oriented masked design flows. It starts from a verified masked gadget and represents each candidate implementation as a directed graph, where nodes correspond to logic gates and edges correspond to fan-in/fan-out connections. This representation is used throughout the search because it allows direct structural modification of the circuit while preserving connectivity information.

We use a digraph representation rather than a string-based encoding. In our setting, the graph representation improves search-space diversity and supports more natural crossover and mutation operations on circuit substructures. This is particularly important for masked gadgets, where the number of gates is limited and redundant encodings can quickly reduce the effectiveness of the search.

The optimization loop proceeds as follows. First, an initial population is generated from the input gadget. Next, each candidate is evaluated using a library-aware multi-objective fitness function. Candidates that fail equivalence or security verification are rejected. The remaining candidates are evolved through selection, crossover, and mutation until a stopping criterion is reached. The best valid candidate is then translated into the technology-library-based format required for final synthesis and verification.

Genetic algorithm (GA) and DEAP environment. We instantiate the search using the non-dominated sorting genetic algorithm (NSGA-II), implemented in DEAP [14]. NSGA-II is well-suited to our setting because the optimization targets are multi-objective and potentially conflicting. Rather than collapsing the problem to a single scalar objective, it maintains a diverse set of non-dominated solutions through non-dominated sorting, crowding distance, and elitism. This is important in our discrete, structurally constrained search space, where premature convergence is a practical concern. DEAP provides the underlying evolutionary operators and population management, on top of which we define a circuit-specific representation, a library-aware fitness function, and structural mutation and crossover operators tailored to masked gadgets. Together, these mechanisms allow the search to adaptively explore gate-level realizations

beyond the structural alternatives exposed by a fixed sequence of conventional CAD transformations.

Initialization phase and candidate representation. The optimization process begins by setting up the candidate environment for the search. The input benchmark is read from a file and instantiated as a circuit graph. Sequential elements are explicitly modeled, including a `BlackBox` abstraction for flip-flops with inputs `CK` and `D` and output `Q`. The input design is then converted into a gate-level Verilog netlist, which serves as the basis for candidate evaluation and subsequent translation to the synthesis and verification tools.

Each individual in the population represents a candidate gate-level realization of the input gadget. This representation is intentionally structural. Unlike larger designs, masked gadgets provide only limited room for safe rewriting; therefore, the search must operate directly on the circuit graph rather than through coarse-grained parameter tuning. The population size determines how many such candidates are maintained and evolved throughout the search.

Library-aware multi-objective fitness function. A central requirement of the framework is efficient evaluation of candidate solutions during search. Invoking full synthesis for every individual would be prohibitively expensive. To address this, we define a library-aware fitness function that estimates implementation quality directly from the candidate netlist using cell characteristics derived from the NanGate 45,nm library.

Let N denote the set of cells in a candidate design. We define fitness as

$$F = \left(\sum_{i \in N} ecv_i\right) \alpha_c + \left(\sum_{i \in N} epv_i\right) \alpha_p + \left(\sum_{j \in \text{Cols}} \min_{k \in \text{Rows}} edv_{jk}\right) \alpha_d, \qquad (1)$$

where ecv_i, epv_i, and edv_{jk} denote the complexity, power, and delay contributions associated with the candidate structure, and α_c, α_p, and α_d are weighting coefficients.

To make this fitness function operational, we instantiate ecv_i, epv_i, and edv_{jk} using gate-level values derived from the NanGate 45 nm technology library. The concrete values used in our implementation are listed in Table 1. In this regard, the table is not an auxiliary implementation detail but rather part of the definition of the surrogate objective used during the search.

In the present work, our main optimization targets are power and area. Delay is included in the fitness to discourage structurally poor candidates during search, but it is not the primary optimization objective. This formulation, inspired by [2], enables rapid ranking of candidate gate-level realizations without requiring a full external synthesis run at each generation. Hence, the fitness function serves as a practical surrogate within the evolutionary loop, while the final results are validated through synthesis and formal security verification.

Synopsys Design Compiler vs. multi-objective fitness function. The proposed fitness function is intended to guide the evolutionary search efficiently without invoking full synthesis at each generation. This is important because repeated calls to an external synthesis tool would make the search prohibitively

Table 1: Evaluating complexity value ecv_i, evaluating power value epv_i, and evaluating signal delay value edv_{jk} for the NanGate 45 nm technology [2].

Gate Type	ecv_i	epv_i	edv_j	edv_k
NAND	14	17	13	16
NOR	14	17	13	16
XNOR	18	16	14	14
NOT2	12	18	12	17
WIRE	10	14	16	12
AND	16	15	15	13
OR	16	15	15	13
XOR	18	16	14	14

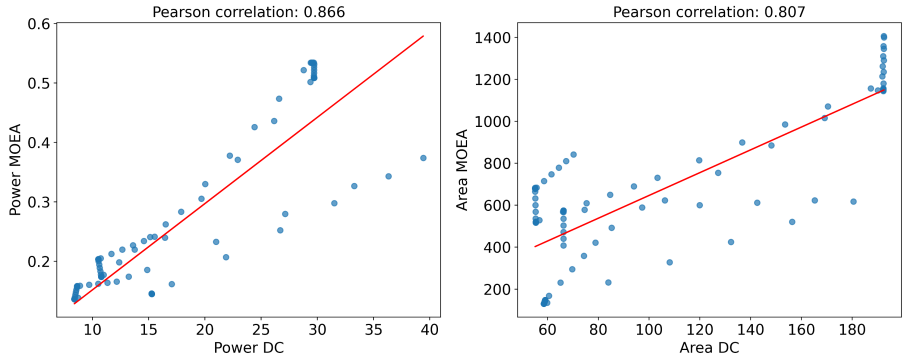

Fig. 2: Pearson's correlation between Synopsys Design Compiler (DC) and our fitness function (MOEA).

expensive. To this end, we instantiate the fitness using gate-level power, area, and delay information derived from the target library.

To assess whether this surrogate is sufficiently faithful, we compare it against Synopsys Design Compiler, which applies NLDM-based timing analysis to the NanGate 45 nm library. We observe Pearson correlation coefficients of 0.866 for power and 0.807 for area between our multi-objective fitness function and Synopsys Design Compiler, as shown in Figure 2. Although Design Compiler provides more precise estimates, this level of agreement indicates that the proposed fitness function is sufficiently aligned with synthesis-based evaluation to guide the search effectively.

Sensitivity analysis. After initialization, the design's sensitivity is analyzed to guide structural variation toward the most relevant parts of the circuit. The circuit is sequentially unrolled using `cg.tx.sequential_unroll`, transforming the sequential design into a combinational form suitable for structural analysis [27]. The number of unrolling steps depends on the number of timing elements and the number of clock cycles. Following prior observations that moderate unrolling yields sufficiently accurate structural estimates with manageable overhead, we use bounded unrolling to avoid exponential growth in complexity [27,26].

Node sensitivity is then computed using `cg.props.sensitivity`. Sensitivity reflects switching activity and highlights the parts of the circuit that are more likely to influence power and timing. The resulting high-sensitivity nodes are stored and used to guide subsequent crossover operations. This allows the search to focus on structural modifications within circuit regions that are most important to the optimization objectives.

Functional equivalence is checked using SAT- and Miter-based methods [23,1]. This ensures that the search explores alternative implementations of the same gadget rather than functionally altered circuits.

Crossover, mutation, and selection. We define a custom multi-objective fitness class, `FitnessMulti`, and evolve individuals through selection, crossover, and mutation. Crossover is implemented as a single-point structural crossover using the `sensitivity_crossover` function. Rather than selecting crossover points uniformly, the operator chooses from the set of sensitive nodes identified during the sensitivity analysis. This biases the search toward structurally important regions of the gadget.

Mutation is implemented through the `mutate_individual` function. This operator supports four mutation types: changing the gate type of a connection, altering the source or target node of a connection, adding a connection, and deleting a connection. The possible nodes and gate types are dynamically extracted from the current individual, ensuring that mutations remain within the valid search space of the circuit representation.

Selection is a critical part of the search because it determines the balance between convergence and diversity. In our framework, offspring are generated from fitter individuals while maintaining enough diversity to avoid early collapse of the search. This is particularly important for masked gadgets, where the structural search space is constrained, and many naïvely attractive candidates are either equivalent to existing ones or fail correctness or security checks.

Termination phase. Fitness evaluation is performed whenever crossover or mutation invalidates an individual's previous score. The population evolves until one of two conditions is met: either a solution below a predefined threshold is found, or a maximum number of generations is reached. If no satisfactory candidate is obtained within the allowed generation budget, the evolutionary process is restarted with a fresh population. The final result of the search is a gate-level realization that satisfies the optimization criteria, passes equivalence checking, and is eligible for synthesis and formal security verification.

Final synthesis and security verification. The final assessment of each optimized candidate is performed outside the surrogate evaluation loop. The resulting Verilog design is translated into the technology-library-based format required by Synopsys Design Compiler and VERICA. Synthesis reports are then used to obtain final power, area, and timing values, while VERICA checks whether the optimized candidate retains the required security property. This final step is essential. An optimization is only meaningful in our setting if the resulting gadget remains functionally correct and PINI-secure. Candidates that fail verification are discarded, even if they improve the optimization metrics.

Table 2: Setting of the genetic operations.

Operation	Type	Value
Population	Direct Acyclic Graph	100
Selection	Tournament	10 indivs [40×]
	Best	1 indiv [10×]
Crossover	Single Point	0.2, 0.4, 0.8
Mutation	Random, Gaussian	0.2, 0.4, 0.8

4 Experiments and Results

The benchmarks used in our experiments are obtained from the open-source AGEMA case studies [21]. To assess side-channel resilience after optimization, we employ VERICA [33] to verify adherence to the PINI probing model. We automate the flow using shell scripts for tool orchestration and Python scripts for generating the annotation and configuration files required by VERICA. For translation and synthesis, we rely on the open-source NanGate 45 nm Design Library [30]. All experiments are executed on a high-performance computing infrastructure with 4 CPU cores, 80 GB of memory, and one GeForce 2080Ti GPU. The parameters used for the evolutionary stages are given in Table 2. We emphasize that the optimization targets in this work are total power and area. Latency and randomness efficiency are not optimization objectives in the present framework. The objective of the GA is not merely to maximize raw reduction, but to adaptively explore alternative gate-level realizations under explicit functional and security constraints.

Gadget implementations under consideration. We consider COMAR [19] and HPC gadgets, particularly HPC1 and HPC2 [8], as well as HPC3 [20], in our experiments; see Figure 1. These gadget families provide meaningful optimization targets because they differ in structural complexity, latency, and randomness requirements while all targeting secure masked design. In particular, HPC gadgets are glitch-robust private circuits designed for composable masked hardware [8], whereas COMAR emphasizes reduced randomness under first-order secure composition [19].

4.1 Conventional CAD Optimizations

We first examine the effectiveness of conventional CAD optimizations for masked gadgets. To this end, we consider technology mapping, FRAIG-based rewriting, and retiming. These techniques expose alternative functionally equivalent gate-level realizations and, in the case of retiming, modify register placement without changing functionality. Our goal is not to propose a new conventional CAD flow, but to assess the optimization opportunities exposed by these standard transformations under functional and security constraints.

For this study, we use ABC [6] because, compared to Synopsys Design Compiler, its FRAIG-based flow exposes multiple structural choices more directly [25], which is useful for exploring alternative gadget realizations before final synthesis and verification. The mapping process is performed using a modified

Table 3: Original gadgets' power and area via Synopsys Design Compiler.

Gadget	Logic Gate	$d+1$	Latency (ina, inb)	Random Bits	Power w/o PRNG (μW)	Area w/o PRNG (μm^2)
COMAR	AND	2	1,2	6	15.265	66.234
	NAND	2	1,2	6	15.266	66.234
	OR	2	1,2	6	15.285	66.234
	NOR	2	1,2	6	15.282	66.234
	MUX	2	1,2	6	40.385	184.338
HPC1	AND	2	1,2	2	8.394	58.52
	NAND	2	1,2	2	8.404	58.52
	OR	2	1,2	2	8.643	59.052
	NOR	2	1,2	2	8.633	59.052
	MUX	2	1,2	2	12.887	82.992
HPC2	AND	2	1,2	1	29.690	192.318
	NAND	2	1,2	1	29.726	192.318
	OR	2	1,2	1	29.717	192.584
	NOR	2	1,2	1	29.354	191.786
	MUX	2	1,2	1	17.962	103.740
HPC3	AND	2	1,1	2	10.790	55.328
	NAND	2	1,1	2	10.786	55.328
	OR	2	1,1	2	10.521	55.062
	NOR	2	1,1	2	10.531	55.062
	MUX	2	1,1	2	15.529	70.224

NanGate 45 nm library chosen for compatibility with VERICA. Starting from the verified benchmark described in Section 3, we convert the design to Berkeley Logic Interchange Format (BLIF) using Yosys [35], apply `fraig`, `map`, and `fraig_sweep`, and evaluate retiming by additionally invoking `retime` [24]. The resulting netlists are passed through Synopsys Design Compiler [34] to obtain a VERICA-compatible format and corresponding synthesis reports, and each candidate is then reassessed using VERICA.

To ensure a fair comparison, we report only conventional-CAD results that remain usable, have been successfully translated, and are PINI-secure. Table 4 lists, for each gadget, the better valid result among optimization with retiming (ORE) and without retiming (ONRE). The strongest reductions are observed for HPC2 and HPC3 gadgets, whereas COMAR shows mostly marginal gains, and HPC1 shows moderate but smaller improvements. Although Table 4 shows that substantial gains are possible, especially for HPC2 and HPC3, the full set of conventional results also contains unusable, untranslated, or excluded variants. The results therefore show that conventional CAD can be highly effective when the selected transformations expose a favorable structural realization, but the outcome depends strongly on the gadget structure and optimization setting. This observation is important for interpreting the GA results. The comparison is not intended to determine a universal winner based only on the largest isolated reduction. Rather, it examines the difference between structural alternatives exposed by conventional CAD transformations and adaptive circuit-specific exploration through evolutionary search. Accordingly, the GA framework should not be interpreted as a universal replacement for conventional CAD optimization, but as an alternative search method that provides direct control over how new gate-level candidates are generated and explored.

Table 4: Best valid improvements achieved through conventional CAD optimization. We evaluated NAND, AND, OR, NOR, XNOR, MUX, and XOR gadgets, and report only those cases for which at least one valid result remained among optimization with retiming (ORE) and optimization without retiming (ONRE). Cases with unusable outputs, translation failures, or non-PINI-secure results are omitted. A negative value indicates reduced overhead.

Gadget	Logic Gate	Best Variant	Reduced Power [%]	Reduced Area [%]
COMAR	AND	ONRE	-0.041	0.000
	NAND	ONRE	-0.041	0.000
	OR	ONRE	0.041	0.000
	NOR	ONRE	0.041	0.000
	MUX	ONRE	-11.266	-13.431
HPC1	AND	ORE	-7.597	-8.185
	NAND	ORE	-8.495	-9.009
	OR	ONRE	-8.804	-9.298
	NOR	ORE	-8.013	-8.929
	MUX	ORE	-11.975	-9.929
HPC2	AND	ORE	-43.505	-42.838
	NAND	ORE	-43.655	-42.838
	OR	ORE	-43.910	-42.916
	NOR	ORE	-43.776	-42.916
	MUX	ORE	-44.359	-43.038
HPC3	AND	ONRE	-53.632	-55.451
	NAND	ONRE	-53.602	-55.451
	OR	ORE/ONRE	-53.772	-55.245
	NOR	ORE/ONRE	-53.776	-55.245
	MUX	ORE	-59.832	-57.918

4.2 Optimization Using Genetic Algorithm

We next evaluate whether adaptive evolutionary search can expose alternative gate-level realizations beyond those obtained through the selected conventional CAD transformations. Unlike the conventional flow, where the explored alternatives are determined by the synthesis and mapping operations, the GA directly modifies the circuit structure through mutation and crossover and uses fitness-based selection to guide subsequent exploration. Here, we evaluate the extent to which this adaptive search can improve masked gadgets while preserving functional correctness and PINI security.

Time vs. complexity. The structural properties of a given gadget directly influence the average time to solution. As shown in Table 3, HPC2 gadgets are the largest, while HPC1 gadgets are the smallest. However, HPC1 and HPC3 gadgets have higher randomness requirements than HPC2 gadgets. Our analysis shows that solutions for HPC2 gadgets are typically easier and more consistent to obtain in terms of time to solution. In contrast, HPC3 and COMAR gadgets exhibit a larger variance across individual logic gates. For example, OR HPC3 and OR COMAR are comparatively tractable, whereas NAND HPC3 and AND COMAR are more difficult to optimize. This trend suggests that randomness and structural constraints both affect search difficulty. Although COMAR uses a fixed randomness value, it still shows greater difficulty in reaching good solutions.

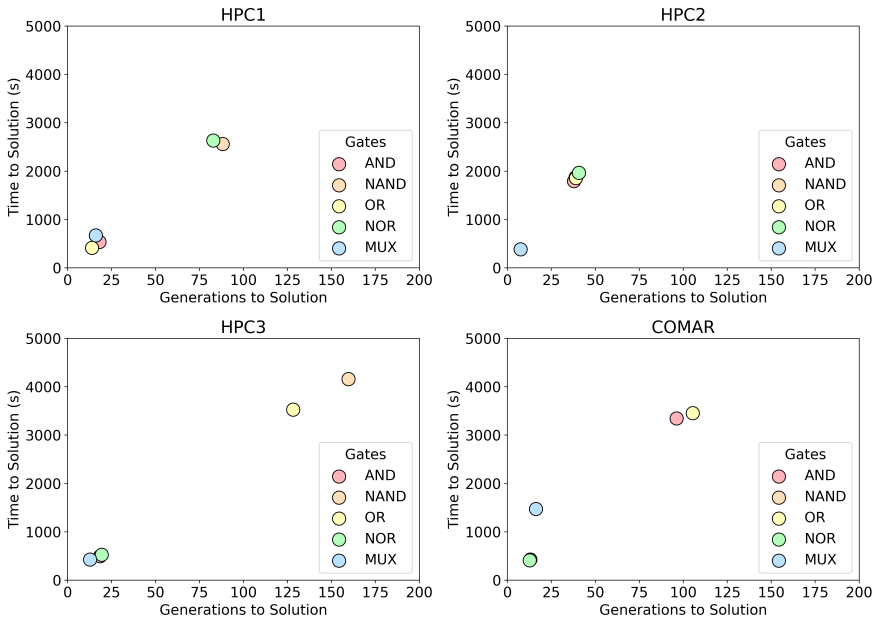

Fig. 3: Average number of generations to first solution against average time to first solution.

Likewise, despite the latency-randomness trade-off of HPC3 relative to HPC2, randomness remains a relevant factor in the search process.

Generation count. Another relevant observation is the relation between generation count and gadget complexity. Larger or structurally more constrained gadgets typically require more generations before converging to a valid solution. Based on the empirical behavior in Figure 3, we selected 200 generations as a practical upper bound for a single run. If no satisfactory candidate was obtained within this budget, the process was repeated. Overall, we executed 10 iterations of 200 generations per gadget.

Best fitness selection. We first examine the best fitness selection as a baseline. This method converges quickly, but its exploration capability is limited. As shown in Figure 4, it tends to converge early and can become trapped in suboptimal regions of the search space. Table 5 further shows that, although best fitness selection yields power improvements for several gadgets, the area improvements are generally modest. This behavior is not surprising in our setting, where the search space is discrete, highly constrained, and prone to premature convergence.

Tournament-based selection. We next evaluate tournament-based selection. The parameters were derived empirically, and the final setting is reported in Table 2. In each round, 10 individuals are randomly selected for each tournament, and the best of them is retained. This process is repeated 40 times per genera-

Table 5: Improvements for best fitness selection method. A negative value indicates reduced overhead.

Gadget Type	Logic Gate	Power Reduction [%]	Area Reduction [%]
COMAR	AND	-2.760	0.402
	NAND	0.003	0.000
	OR	-0.503	0.000
	NOR	-0.778	0.000
	MUX	-0.803	-0.289
HPC1	AND	-4.962	0.000
	NAND	-1.460	0.000
	OR	1.231	0.901
	NOR	-1.180	0.901
	MUX	-1.057	0.000
HPC2	AND	-3.320	0.138
	NAND	-3.253	0.138
	OR	-0.552	0.000
	NOR	0.202	0.139
	MUX	-3.113	-0.513
HPC3	AND	-3.529	-0.962
	NAND	-3.792	-0.962
	OR	3.774	-0.966
	NOR	2.833	-3.382
	MUX	-5.230	0.000

tion, and the resulting offspring, together with elitist carryover, form the next population.

Tournament selection clearly outperforms best fitness selection in our setting. It provides a more expansive exploration of the search space, as shown in Figure 4, and leads to stronger optimization results overall. The increased diversity allows the search to explore a broader set of candidate realizations rather than repeatedly selecting from a narrow region of the search space. This trend is also reflected in Table 6, where randomized mutation under tournament selection produces the strongest overall results among the tested strategies.

We also evaluate Gaussian-based mutation under tournament selection. Its benefit is limited. While it can reduce the number of steps to convergence in some cases, it does not yield a significant improvement over random mutation in the final optimization outcome. This is consistent with our search space, in which the mutated objects are graph topologies and gate-level connections rather than smooth, real-valued parameters.

Summary of results in Sections 4.1-4.2. The strongest overall GA results are obtained with tournament-based selection and randomized mutation. The broader trend is that HPC2 and HPC3 gadgets are more amenable to GA-based optimization than COMAR and HPC1 gadgets. In particular, Table 6 shows that several HPC2 and HPC3 instances achieve clear power and area improvements while retaining PINI security. For the best-performing GA cases, especially the HPC3 AND and MUX gadgets, the reductions are substantial. By contrast, COMAR and HPC1 remain more constrained, which is consistent with their structural limitations and randomness requirements.

These observations also illustrate the adaptive nature of the evolutionary search. Its behavior is not uniform across gadget families because the available

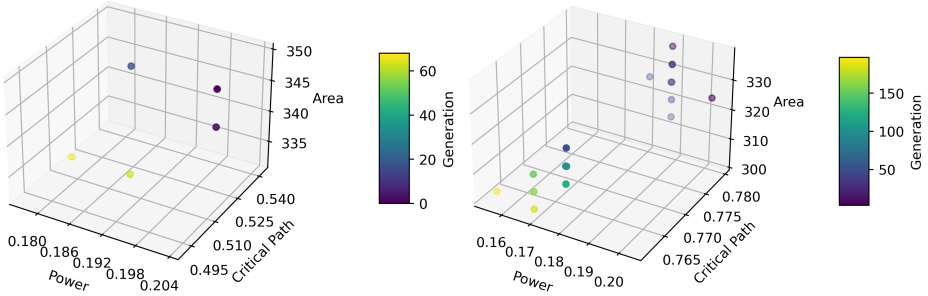

Fig. 4: Search space exploration for best fitness selection (left) and tournament selection (right) on the NOR HPC3.

Table 6: Improvements achieved through randomized mutation under tournament selection. A negative value indicates reduced overhead.

Gadget Type	Logic Gate	Power Reduction [%]	Area Reduction [%]
COMAR	AND	-3.174	1.205
	NAND	0.009	0.000
	OR	-0.012	0.000
	NOR	3.185	1.205
	MUX	-1.019	-0.289
HPC1	AND	-1.271	0.000
	NAND	-1.274	0.000
	OR	-1.260	-1.802
	NOR	-0.961	-0.450
	MUX	-2.442	-0.321
HPC2	AND	-3.179	-1.383
	NAND	-0.495	0.138
	OR	-0.025	0.065
	NOR	-0.590	1.110
	MUX	-2.526	-0.256
HPC3	AND	-15.180	-6.250
	NAND	-2.353	-2.885
	OR	-8.460	-1.932
	NOR	-2.983	-0.966
	MUX	-6.358	-10.227

structural search space depends on the gadget itself. Within that space, however, mutation, crossover, and selection allow the search to generate new candidates and use their fitness to guide subsequent exploration.

All results reported as valid optimized gadgets retain the required security notion under VERICA. Candidates that fail the required verification are not reported as valid optimized results. Hence, the reported power and area improvements are not obtained by relaxing PINI security.

Taken together, the results clarify the different strengths of the two approaches. Conventional CAD delivers the largest reductions in several cases, especially for HPC2 and HPC3, when its predefined transformations expose favorable structural realizations. The GA does not always achieve the largest raw reduction, but it provides a different optimization mechanism by adaptively generating and exploring circuit-specific gate-level alternatives through mutation,

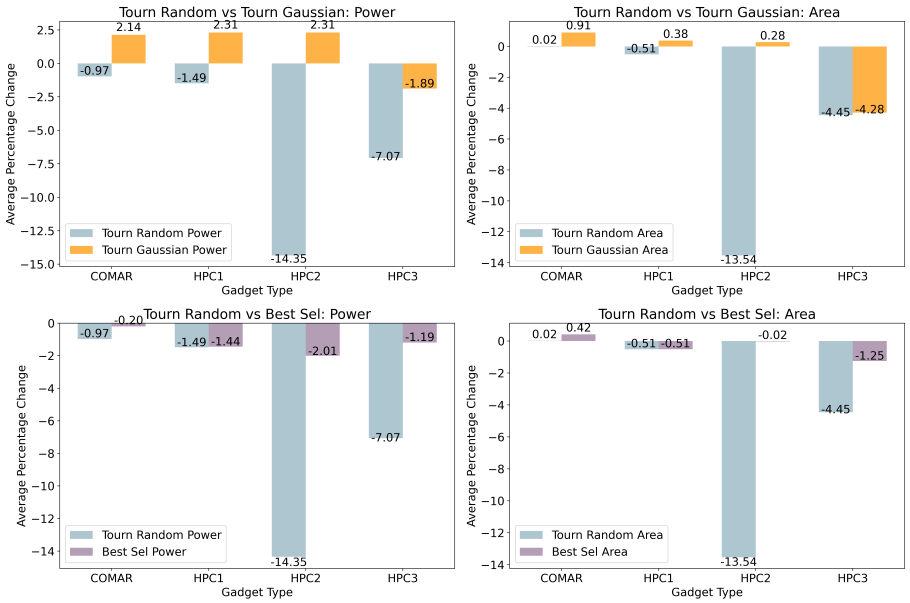

Fig. 5: Average performance of the evolutionary methods.

Table 7: Results of optimizing masked circuits compared to the base implementations as in COMPRESS [7].

Design		d	Power (μW)	Area (μm^2)	Timing (Slack)
Serialized Skinny S-box	Base	2	261.018	684.683	2.39
	Optimized	2	262.468	653.029	2.39
32-bit AES	Base	2	6940.000	17861.367	1.00
	Optimized	2	6110.000	16760.127	1.01

crossover, and selection. The two approaches are therefore complementary rather than direct substitutes: conventional CAD can provide strong reductions through established transformations, whereas evolutionary search provides greater control over how alternative circuit structures are generated and explored.

4.3 Cryptographic Benchmarks

The next question is whether gadget-level improvements translate to improvements in larger cryptographic circuits composed of optimized gadgets. To assess this, we consider a 32-bit AES design and a Skinny S-box design from COMPRESS [7]. Notably, COMPRESS itself already targets small and fast masked implementations, making these benchmarks a relatively strong baseline rather than a non-optimized starting point. Our framework does not alter the higher-level control structure of these designs. Instead, it replaces only the selected gadget implementation with its optimized version.

For conciseness, we report the integration results for the MSKregEn gadget, which shows the most notable improvement among the gadgets used in these benchmark designs. For the other gadgets, our optimized versions are similar to those already obtained in COMPRESS. We emphasize that we did not feed COMPRESS gadgets directly into our tool. Rather, the framework independently generated comparable optimized realizations.

The optimized MSKregEn gadget reduces area by 44% and power by 39%, while also using 3 cells instead of 4. When integrated into the 32-bit AES core, this yields 7.2% area reduction and 12% power reduction. For the Skinny S-box design, the power remains effectively unchanged while the area is reduced by about 5%. These results show that gate-level gadget optimization can provide additional power and area savings even when applied to compact masked implementations produced by an existing optimization framework. The difference between the two designs is due primarily to the relative contribution of the optimized gadget to the overall circuit.

5 Discussion and Future Work

Randomness optimization. In our flow, randomness optimization can, in principle, be inserted either (a) before the GA, by modifying gadget/library choices, or (b) randomness re-wiring and then letting the existing GA optimize power and area, or (c) inside the GA, by extending mutation/crossover to manipulate random nets directly. Since our pipeline relies on fixed module interfaces and JSON share annotations, optimizing randomness inside GA (option c) would substantially enlarge the search space and place even more burden on VERICA to reject insecure candidates. Hence, we examine the possibility of reducing the number of fresh random numbers via options (a) and (b). To evaluate the viability of option (a), we apply reusing to reduce the number of random numbers. The first-order HPC2 constructions already use 1 random bit, which cannot be reduced further. HPC1 uses $(d(d+1)/2) + r(d)$ random field elements, where $(d+1)$ is the share count, and $r(d)$ is the number of random bits needed to share 0 in $\mathbb{F}_2$ [17,8]. For first-order masking with $d = 1$, $r(d) = 1$ as shown in Figure 1. The two random bits in first-order HPC1 should be strictly statistically independent; however, when implementing the gadget, physical reuse (sharing a wire) can be an optimization possibility when the randomness register is updated by the PRNG every clock cycle. Updating the randomness register every clock cycle provides temporal freshness across cycles. To validate the implementation with respect to the independence required by the masking proof, we empirically test it using VERICA. Our experiments indicate only negligible power differences for these HPC1 variants.

Regarding option (b), we investigate whether the randomness-efficient HPC1 gadgets can be further optimized in terms of power and area using the GA. In that setting, we have not observed significant savings in power or area either. Overall, these results suggest that CAD optimization and our current GA-based framework are better suited to power/area optimization under a given

randomness assumption than to discovering new randomness-saving constructions. Meaningful randomness reduction will likely require a dedicated pre-GA pass, especially at the composition level, which is left for future work.

Delay optimization. Delay and latency optimization remain challenging in our framework, mainly because they require timing-aware techniques beyond the current GA formulation. In a few cases, we observed reduced critical-path delay, but typically at the expense of power and/or area. In this regard, effective delay optimization often requires specialized methods such as Pan's Algorithm [31], which explicitly balances timing paths by identifying critical paths and redistributing delays. Our current design flow is also constrained by VERICA, since the use of only single-drive-strength gates in the NanGate 45 nm library restricts access to faster cells that could otherwise improve delay.

Limitations of string-based representation. We also considered a string-based graph representation obtained by converting the digraph benchmark into a string form for evolution. Although this preserves the original circuit structure, it introduces difficulties in correctly implementing crossover and mutation. In particular, repeated identical nodes were frequently generated, which reduced diversity in the search space. For this reason, we did not use this representation in the final framework.

Limitations of VERICA. Two current constraints in VERICA affect our results. First, it supports X1 drive strength by default, which provides a consistent baseline but may affect reported PPA values. Second, the use of the NanGate45 library [30] provides a standardized foundation, but limits the exploration of pre-silicon behavior across a wider range of libraries and cell options, especially for effects such as coupling.

To the best of our knowledge, this is the first application of evolutionary algorithms within a secure design framework for masked hardware. One promising direction for future work is evaluating the framework for implementation on an FPGA. This would require integration with FPGA-specific toolchains and constraints from major vendors such as Xilinx.

6 Conclusion

This work studies the optimization of masked hardware gadgets under formal security constraints by comparing conventional CAD transformations with an evolutionary search-based framework. Our focus is on gate-level power and area optimization under explicit functional and security constraints. The results show that conventional CAD can yield very strong reductions when its predefined transformations expose favorable structural realizations, but the outcome depends strongly on the gadget structure and optimization setting. The evolutionary framework does not always achieve the largest raw reduction, but provides a controllable and adaptive search process that directly generates and explores circuit-specific gate-level alternatives while retaining functional correctness and PINI security. Across the evaluated benchmarks, it achieves measurable improve-

ments over naïve synthesis, reaching up to 15% reduction in power and area in selected cases.

References

1. Andrade, F.V., Silva, L.M., Fernandes, A.O.: Improving sat-based combinational equivalence checking through circuit preprocessing. In: 2008 IEEE International Conference on Computer Design. pp. 40–45. IEEE (2008)
2. Bao, Z., Watanabe, T.: A novel genetic algorithm with cell crossover for circuit design optimization. In: 2009 IEEE International Symposium on Circuits and Systems. pp. 2982–2985. IEEE (2009)
3. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y.: Verified proofs of higher-order masking. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 457–485. Springer (2015)
4. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 116–129 (2016)
5. Bhandari, J., Mankali, L., Nabeel, M., Sinanoglu, O., Karri, R., Knechtel, J.: Beware your standard cells! on their role in static power side-channel attacks. Cryptology ePrint Archive, Paper 2023/920 (2023), https://eprint.iacr.org/2023/920
6. Brayton, R., Mishchenko, A.: Abc: An academic industrial-strength verification tool. In: Touili, T., Cook, B., Jackson, P. (eds.) Computer Aided Verification. pp. 24–40. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
7. Cassiers, G., Gigerl, B., Mangard, S., Momin, C., Nagpal, R.: Compress: Generate small and fast masked pipelined circuits. IACR Transactions on Cryptographic Hardware and Embedded Systems **2024**(3), 500–529 (2024)
8. Cassiers, G., Grégoire, B., Levi, I., Standaert, F.X.: Hardware private circuits: From trivial composition to full verification. IEEE TC **70**(10), 1677–1690 (2020)
9. Cassiers, G., Standaert, F.X.: Trivially and efficiently composing masked gadgets with probe isolating non-interference. IEEE TIFS **15**, 2542–2555 (2020)
10. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: CRYPTO (1999)
11. De Cnudde, T., Ender, M., Moradi, A.: Hardware masking, revisited. IACR TCHES pp. 123–148 (2018)
12. Faust, S., Grosso, V., Del Pozo, S.M., Paglialonga, C., Standaert, F.X.: Composable masking schemes in the presence of physical defaults & the robust probing model. IACR TCHES pp. 89–120 (2018)
13. Feldtkeller, J., Sasdrich, P., Güneysu, T.: Challenges and opportunities of security-aware eda. ACM Transactions on Embedded Computing Systems **22**(3), 1–34 (2023)
14. Fortin, F.A., De Rainville, F.M., Gardner, M.A.G., Parizeau, M., Gagné, C.: Deap: Evolutionary algorithms made easy. The Journal of Machine Learning Research **13**(1), 2171–2175 (2012)
15. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: CHES. pp. 251–261. Springer (2001)

16. Grosso, V., Standaert, F.X., Faust, S.: Masking vs. multiparty computation: how large is the gap for aes? In: Cryptographic Hardware and Embedded Systems-CHES 2013: 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings 15. pp. 400–416. Springer (2013)
17. Hadžic, V., Bloem, R.: Efficient and composable masked aes s-box designs using optimized inverters. IACR Transactions on Cryptographic Hardware and Embedded Systems **2025**(1), 656–683 (2025)
18. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: CRYPTO. pp. 463–481. Springer (2003)
19. Knichel, D., Moradi, A.: Composable gadgets with reused fresh masks: First-order probing-secure hardware circuits with only 6 fresh masks. IACR TCHES **2022**(3), 114–140 (Jun 2022), https://tches.iacr.org/index.php/TCHES/article/view/9696
20. Knichel, D., Moradi, A.: Low-latency hardware private circuits. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 1799–1812 (2022)
21. Knichel, D., Moradi, A., Müller, N., Sasdrich, P.: Automated generation of masked hardware. Cryptology ePrint Archive (2021)
22. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: CRYPTO. pp. 388–397. Springer (1999)
23. Marques-Silva, J.: Practical applications of boolean satisfiability. In: 2008 9th International Workshop on Discrete Event Systems. pp. 74–80. IEEE (2008)
24. Mishchenko, A., Chatterjee, S., Brayton, R., Pan, P.: Integrating logic synthesis, technology mapping, and retiming (01 2006)
25. Mishchenko, A., Chatterjee, S., Jiang, R., Brayton, R.K.: Fraigs: A unifying representation for logic synthesis and verification (2005), https://api.semanticscholar.org/CorpusID:9209313
26. Miskov-Zivanov, N., Marculescu, D.: Soft error rate analysis for sequential circuits. In: 2007 Design, Automation & Test in Europe Conference & Exhibition. pp. 1–6 (2007). https://doi.org/10.1109/DATE.2007.364500
27. Miskov-Zivanov, N., Marculescu, D.: Modeling and optimization for soft-error reliability of sequential circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **27**(5), 803–816 (2008)
28. Momin, C., Cassiers, G., Standaert, F.X.: Handcrafting: Improving automated masking in hardware with manual optimizations. In: COSADE. pp. 257–275. Springer (2022)
29. Muller, N., Meschkov, S., Gnad, D.E., Tahoori, M.B., Moradi, A.: Automated masking of fpga-mapped designs. In: FPL. pp. 79–85. IEEE Computer Society, Los Alamitos, CA, USA (sep 2023). https://doi.org/10.1109/FPL60245.2023.00019, https://doi.ieeecomputersociety.org/10.1109/FPL60245.2023.00019
30. Nangate: 45nm. [Online]https://tilos-ai-institute.github.io/MacroPlacement/Enablements/NanGate45/[Accessed:Apr.7,2025], year=2025
31. Pan, P., Karandikar, A.K., Liu, C.: Optimal clock period clustering for sequential circuits with retiming. IEEE transactions on computer-aided design of integrated circuits and systems **17**(6), 489–498 (1998)
32. Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: CRYPTO. pp. 764–783. Springer (2015)
33. Richter-Brockmann, J., Feldtkeller, J., Sasdrich, P., Güneysu, T.: Verica - verification of combined attacks: Automated formal verification of security against simultaneous information leakage and tampering. IACR TCHES **2022**(4), 255–284 (Aug 2022)

34. Synopsys: Design compiler concurrent timing, area, power, and test optimization (2023), https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html
35. Wolf, C., Glaser, J., Kepler, J.: Yosys-a free verilog synthesis suite (2013), https://api.semanticscholar.org/CorpusID:202611483
36. Wu, L., Fan, Y., Preneel, B., Wang, W., Wang, M.: Automated generation of masked nonlinear components: From lookup tables to private circuits. [Online] https://eprint.iacr.org/2023/831 [Accessed: Nov.20, 2023] (2023)