EDA Workflow for Optimization of Robust Model Probing-Compliant Masked Hardware Gadgets

David S. Koblah
Florida Institute for National Security
University of Florida
Gainesville, FL, USA
Email: dkoblah@ufl.edu

Dev Mehta
Dept. of Electrical Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
Email: dmmehta2@wpi.edu

Mohammad Hashemi
Dept. of Electrical Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
Email: mhashemi@wpi.edu

Fatemeh Ganji
Dept. of Electrical Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
Email: fganji@wpi.edu

Domenic Forte
Florida Institute for National Security
University of Florida
Gainesville, FL, USA
Email: dforte@ece.ufl.edu

***Abstract*—Side-channel analysis poses a threat to security-critical systems, enabling attackers to exploit the unintended information leakage. To mitigate its effect, masking serves as a provably-secure countermeasure by segmenting computation into random shares. Existing work on optimizing building blocks of modern masked circuits, called gadgets, has primarily focused on latency, with area and power as secondary objectives. To the best of our knowledge, the most up-to-date ASIC-specific masking gadget optimization frameworks require significant manual effort. This paper is the first to reintroduce technology mapping, retiming and functionally-reduced and-inverter graphs (FRAIGs) in the electronic design automation process (EDA) to attempt to optimize and improve existing gadgets and overall designs. In this regard, we aim to enhance gadgets in terms of the power, performance, and area (PPA) metrics. The primary objective is to leverage compatible gates from a technology library to generate an optimized and functional design without compromising security, and latency. Our results show 59% and 57% reduction in power consumption and area respectively when compared to naïve synthesis of masked designs.**

***Keywords*—Side-channel Analysis; Hardware Masking; EDA; Optimization.**

I. Introduction

Side-channel analysis (SCA) has been studied extensively for over two decades since the seminal work introduced its devastating impact on security-critical implementations cf. [1]. The fundamental premise of SCA is that an adversary can observe and measure physical effects to extract sensitive information during execution, where one of the common types of measurements is power consumption and its close counterpart, electromagnetic (EM) emanation [1], [2]. Different classes of countermeasures have also been developed with *masking* being predominantly studied thanks to its formal and sound security foundation [3]. Masking schemes can be likened to techniques of secret sharing, where confidential values are randomly decomposed into *shares*. Operations on these shares are executed to meet a specific security criterion.

Despite the continuous effort to improve the efficiency and security of hardware masking schemes, inaccuracies and design flaws have often resulted in their failure. In this regard, unintentional physical effects, e.g., glitches, transitions, or coupling, and architectural conditions (parallelism and pipelining, for instance) account for the insecurity of physical instantiation of masking scheme regardless of their sound theoretical security proofs [4]. As a response to this, masking of small components, so-called gadgets, has been introduced to ease the burden of such enormous engineering and error-prone tasks [5]. One key aspect is the secure composition of gadgets known to be a non-trivial problem relying on security notions and properties; for instance, *probe-isolating non-interference (PINI)* that enables efficient and secure compositions with respect to multi-input, multi-output gadgets and trivially secure linear operations, i.e., XORing [6]. Taking these important steps toward designing masked hardware circuits has turned the page and allowed for open-source tools that help engineers and hardware easily generate masked hardware circuits, starting from a simple but unprotected design.

As a prime example, AGEMA [4] transforms unprotected cryptographic designs into securely masked circuits using different masked gadgets as fundamental building blocks. It explores different processing techniques to achieve this transformation and supports masked gadgets to offer high flexibility concerning the security level, required randomness, latency, and area overhead. Building upon this, generic hardware private circuits (GHPC) gadgets have been devised to provide a generalizable and automated approach to designing secure, efficient, and trivially composable gadgets for arbitrary Boolean functions [7]. Afterward, [8] has demonstrated improvements by manual optimization in AGEMA. In doing so, the latency in AES S-box implementations, especially for serial implementation, is reduced by $6\times$. An added benefit to their method is reduced area consumption due to changes in the number of registers. This line of research has been followed by proposing masked nonlinear components with improved performance [9] due to latency asymmetry.

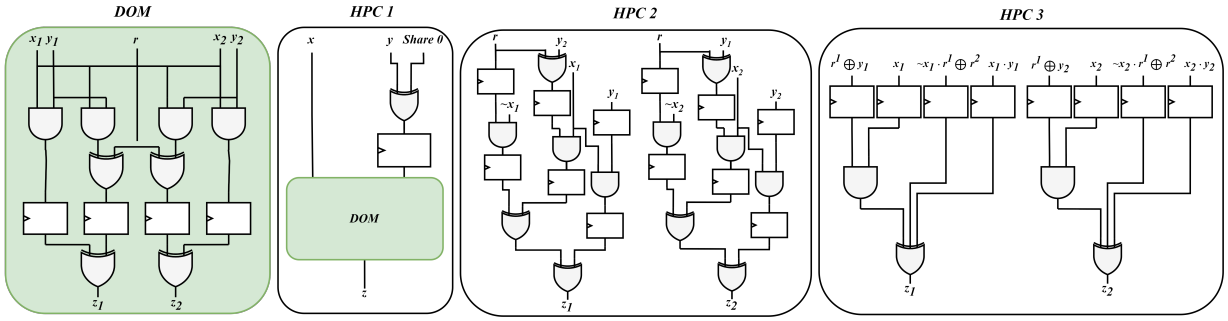

Fig. 1. Diagram of the DOM implementation and its use in the HPC 1 gadget whereas HPC 2 and HPC 3 do not utilize the DOM gadget. Here, x, y, and z represent input and output while r and *share 0* represent the refresh bits. The subscripts represent the shares and the superscripts represent different refresh bits (see Sec. II and IV).

Opportunities for optimization have arisen to improve certain aspects of gadgets without disrupting the secure state. The study [10] has proposed a security-focused design-space exploration framework that utilizes commercial CAD tools; it analyzes the security-versus-PPA design-space across multiple technology nodes with all their voltage threshold cell options. This provides a better understanding of the relationship between standard cells and static power side channel vulnerability. A very recent tool, AGEMA_FPGA, was introduced as an automated framework for transforming unprotected FPGA netlists into optimized masked designs aligning with the PINI notion [11]. Compared to AGEMA, it achieves up to 22% fewer registers, up to 64% fewer LUTs, and up to 59% less power consumption while verifying their side channel resilience.

Our contributions. This paper narrows the gap between the theory and practice of masked gadgets by introducing a completely open-source tool (GitHub link will be added after acceptance) to optimize masked gadgets on ASIC, contrary to AGEMA_FPGA. In fact, while AGEMA is devoted to ASIC applications, our framework highlights the significant room for streamlined optimization of gadgets. Despite the fact that masked gadgets can be optimized using existing ASIC EDA workflow, our paper provides insights into how less careful application of that can result in side-channel leakage in the masked gadgets. In a nutshell, our contributions are: leftmargin=*,nosep

- An EDA flow for efficient and automated design of provably secure composable masked gadgets. Extensive experiments are done on a wide range of existing gadgets to demonstrate a more area-efficient and less power-hungry design without compromising security or latency.
- Completely open-source design and security verification through publicly available tools. In this regard, streamlined masked design synthesis is performed by leveraging the open-source ABC tool and its FRAIG implementation to perform technology mapping.
- Retention of side-channel security confirmed by the state-of-the-art approaches while showing promising applications to different technology nodes. As a result, our framework can be seen as an essential add-on to the classical EDA flow to guarantee the security of masked gadgets.

II. Masking-related Models and Tools

Boolean masking. In essence, the goal of a masking scheme is to ensure security at an established order d, under assumptions concerning the leakage behavior of the targeted device. A prevalent approach to masking is Boolean secret sharing, which uses binary addition to distribute sensitive data. In this context, a secret variable $x \in GF(2^m)$ is split into $d+1$ shares $(x_1, \ldots, x_{d+1})$ with the property $x = \bigoplus_{i=1}^{d+1} x_i$. Uniformity is assured by selecting shares $x_1, \ldots, x_d$ from a uniformly random distribution and determining $x_{d+1} = x \oplus \bigoplus_{i=1}^{d} x_i$ (known as the correctness property). Although a range of Boolean secret-sharing methods has been discussed in previous works, our discussion will center on domain-oriented masking as described in [12].

An example of DOM applications, namely DOM multiplication, is shown in Fig. 1. Initially, cross-product calculations are performed, with randomness introduced to specific products. Precisely, for a *first-order* security level, the following is derived [13]:

$$p_1 = x_1 y_1,\ p_2 = x_1 y_2 \oplus r_1,\ p_3 = x_2 y_1 \oplus r_1,\ p_4 = x_2 y_2. \tag{1}$$

Subsequently, the terms p_i from Eqn (1) are stored in a register and then proceed to a compression phase. Here, the $(d+1)^2$ shares are transformed to $(d+1)$ shares for the output z_i. For example, $z_1 = p_1 \oplus p_2$ and $z_2 = p_3 \oplus p_4$. DOM multiplication requires independent input shares and necessitates an additional clock cycle in the compression layer [12], [14], [15]. Nonetheless, the DOM multiplier remains a frequently adopted masking scheme, with its security rooted in the independent power usage of its constituent functions.

Security models. In *d-probing model*, the security concerns the adversary, granted the ability to observe the distribution over up to d wires of a given circuit. This should not disclose anything about the processed secret value x [16].

Strong non-interference (SNI) and *non-interference (NI)* consider the composition of masked circuits, usually as compositions of gadgets. Concretely, in NI-compliant circuits, security spans across the composed circuit instead of isolated gadgets only cf. [7]. Roughly speaking, the flow of sensitive

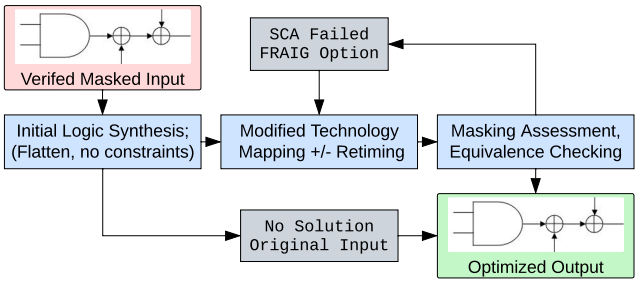

Fig. 2. Summary of framework for mask gadget optimization presented in this paper.

information within the circuit is limited; however, the adversary is still allowed to gain partial information on d internal values and wires. Probes give the adversary access to partial information that could have been solely observed by placing more than d adversarial probes (i.e., probe propagation). In order to tackle this, if SNI is fulfilled, probe propagation is stopped at the primary output of gadgets, which limits the partial information accessible to the adversary. Although SNI reflects the need for secure composition of gadgets, its purpose is limited to single-output gadgets only as it does not scale for multi-output gadgets due to probe propagation cf. [6]. *PINI* has been introduced to meet this need, where share domains, similar to DOM, are introduced, and any probe is restricted to only propagate within its own share domain. In addition to supporting multi-output gadgets (unlike SNI), PINI enables the trivial implementation of linear functions.

Gadgets. The research on designing gadgets has particularly focused on non-linear gates, with the AND gate receiving extensive attention. Among the developed gadgets, the Trichina AND gate [17], the Ishai, Sahai, and Wagner (ISW) AND gate, as well as ones that satisfy PINI, SNI, and NI [6] probing models, are the most noteworthy. The Trichina AND [17] ensures security at the gate level for $\lambda = 1$, indicating security against a single probe. It is specifically designed to manage two 2-share inputs, which is critical for its security assurances. The ISW AND [16] lays the foundation for threshold implementations, providing a robust basis for crafting higher-order protection schemes. This gadget applies to the circuit level, incorporating the gadget composition and scalability for multiplication computation, which is particularly useful in AES implementations.

III. Methodology

Our framework is designed to be seamlessly integrated into the ASIC-focused EDA framework, specifically for masked designs. We automate the process using a shell script where program calls are made sequentially, with each stage producing and reading inputs in a streamlined manner. The commands used for each tool are described throughout this section.

The input benchmark is the result of the high level synthesis stage of the EDA process. The register transfer level (RTL) design has to be verified for side-channel resilience using VERICA [18] (see Fig. 2). Apart from verification capabilities encompassing SCA or fault attacks (FA), VERICA prominently employs the probing models introduced in Sec. II, making it especially suited for verifying masked designs. The verified design initialises the entire framework with logic synthesis via Yosys [19].

A. Yosys for BLIF Conversion

Yosys is designed for RTL design and synthesis tasks within both standard cell mapping and FPGA EDA frameworks. One notable feature of Yosys is its ability to handle the conversion of designs from Verilog to other benchmark representations such as Berkeley Logic Interchange Format (BLIF) and Electronic Design Interchange Format (EDIF). For our optimization task, Yosys converts Verilog to BLIF, aligning the input with the requirements of ABC [20] in the next step. BLIF ensures the retention of ports, input shares, and output shares of the masked gadgets.

B. ABC for Optimization

While ABC provides logic optimization capabilities, its simple design allows different circuit-based applications to be built on its framework [20]. It is the crux of our proposed framework, and the modified technology mapping process shown in Fig. 2 is built around its requirements and capabilities.

1) Technology Mapping: The process entails transforming a technology-independent optimized logic netlist into a representation using predefined gate layouts from a technology library [21]. The primary role of technology mapping is to intelligently choose gates for implementing the logic netlist. The technology mapping process we leverage via ABC is a direct acyclic graph (DAG)-based method that enables the identification of the best-fit standard cells [20]. The default library provided with the ABC package is the NanGate 45nm library. However, we modify the library to utilize only AND2, BUF, CLKBUF, INV, NAND2 NOR2 and OR2 gates. This is essential for compatibility with VERICA.

2) FRAIGs: FRAIG allows for the functional representation of networks via improved technology mapping using multiple structural choices [22]. Overall structures and substructures for a single network can be preserved and retrieved based on design requirements using the FRAIG manager [20]. We target the FRAIGs ability to use structural information from technology-independent synthesis to provide choices during technology mapping. We utilize the FRAIG manager to sweep through possible options and evaluate multiple

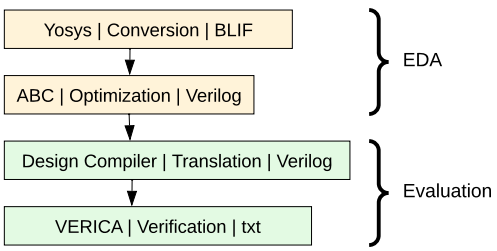

Fig. 3. Details of tools used in order of application, purpose and output. The tools are divided into either EDA-based optimization or security evaluation

designs. The commands we send to ABC are: `fraig, map, fraig_sweep`. Cycling through various choices from the FRAIG, we continue until we identify either the best-optimized and functional candidate or revert to the original, verified masked input.

3) Retiming: Retiming is a technique used in digital circuit design and optimization to improve the performance of synchronous digital circuits by adjusting the placement of registers (flip-flops or latches) within the circuit without changing its functionality [23]. The goal of retiming is to strategically move these registers to different locations within the circuit while maintaining the same logical behavior. Retiming only rearranges the registers to improve timing characteristics [23]. For this, we add the following command to our original ABC set: `retime`. However, the success of retiming depends on the specific characteristics of the circuit, and in some cases, it may not always lead to significant improvements. This is documented in Sec. IV. The design post-ABC is sent to Synopsys Design Compiler for translation into a VERICA-friendly format without further optimization.

C. Synopsys Design Compiler for Translation

Design Compiler [24] is a powerful tool offering constraint-driven synthesis and optimization for various design styles. We utilize Design Compiler for two purposes: Firstly, it "translates" designs obtained in Verilog from the ABC technology mapping process into a useable format for VERICA (Fig. 3). In addition to netlist "translation", Design Compiler offers post-synthesis power, area, and timing reports. These reports are crucial for validating the efficacy of the optimization process.

D. Security Verification using VERICA

The final benchmark is assessed using VERICA [18] again; an ideal result satisfies optimization while retaining its original security characteristics whereas an unsuccessful design flags the tool chain to present the second variant retained in ABC's FRAIG manager. Our entire framework will cycle through all options and verify until a suitable candidate is produced. In the absence of the latter, the original design is simply synthesized, flattened and produced as the optimum result. The FRAIG implementation also allows us to automatically perform formal verification between the input design and its optimized variant.

IV. Experiments and Results

The benchmarks utilized in our experiments were obtained from the open-source AGEMA case studies [4]. Moreover, to assess the framework's viability on larger designs beyond gadgets, we focus on specific implementations of the AES and PRESENT ciphers. For larger AES and PRESENT benchmarks, we found that designs from the PROLEAD [25] example set were easier for both the naïve and optimized tool sets to successfully parse and synthesize. To assess the side channel resilience after the optimization, we employ a locally-installed version of VERICA [18] tool to check for adherence to the PINI probing model. In the execution of our framework, all tools were used within a dedicated Linux-based environment available to collaborators. To streamline the experimental process, we developed a shell script to automatically run each tool when required. Additionally, we wrote Python-based programs to generate annotation and configuration files workflow.

For translation/synthesis with Design Compiler, we relied on the open-source NanGate 45nm Design Library. Although we sought to use other libraries, NanGate emerged as the sole option compatible with VERICA. This point is particularly significant since our overarching goal is to seamlessly integrate the masking process into the EDA pipeline.

A. Benchmarks

HPC. It is a specialized type of private circuit designed for the hardware, which is glitch-resistant and enables trivial composition at arbitrary orders introduced in [26]. They have presented a set of glitch-robust PINI gadgets that improve on the DOM multiplier. For example, the HPC1 gadget is based on the refresh-then-multiply technique where a glitch-robust refresh is added to the DOM to make it a glitch-robust PINI gadget. We have used HPC1 and HPC2 [26], as well as HPC3 [27] for our experiments (see Figure 1).

COMAR. Composable Gadgets with reused fresh masks (COMAR) gadgets are a set of hardware gadgets that enable the construction of arbitrary first-order-secure hardware implementations in the d-probing model under glitches, utilizing only 6 fresh random bits in total [28]. They offer security and free composability in the glitch-extended robust d-probing model for $d = 1$, while minimizing the randomness requirements and design-error susceptibility.

B. Retiming vs Non-retiming

Retiming seemingly yields varied results across all use cases. In the case of HPC1, the difference in optimization values is minimal. However, for HPC3 and COMAR gadgets, the retimed versions generally exhibit higher overhead compared to their non-retimed counterparts (see Tab. I). Conversely, HPC2 gadgets demonstrate a significant difference in levels of optimization, possibly attributed to the distinct methods of construction for each gadget. The key to achieving consistent optimization results from retiming may lie in adopting a more nuanced and controlled approach.

C. Power Optimization

The primary focus of power in this paper is dynamic power, which specifically addresses the consumption during switching activity [29]. While not explicitly documented in this paper, we are also capable of reporting leakage power using Design Compiler for each gadget used in the experimentation process. Fig. 4 highlights observable differences between the naïve and optimized masked designs. The COMAR gadgets reported in Fig. 5 exhibit minimal change in dynamic power dissipation for both retiming and non-retiming methods. This is because COMAR gadgets have a constant number of random bits that already optimizes their randomness requirements, circuit sizes, and power requirements [28]. However, the HPC2 and HPC3

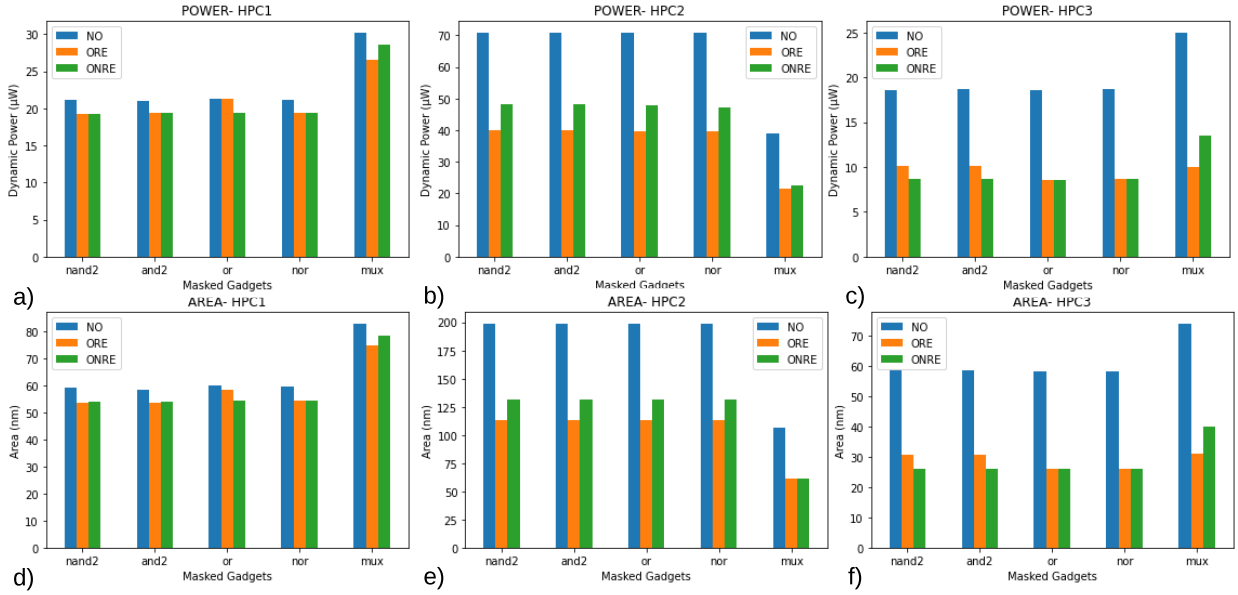

Fig. 4. Comparison of power and area improvements of naïve (NO), Optimized with Retiming (ORE), and Optimized without Retiming (ONRE) methods for HPC1 (a, d), HPC2 (b, e) and HPC3 (c, f) gadgets

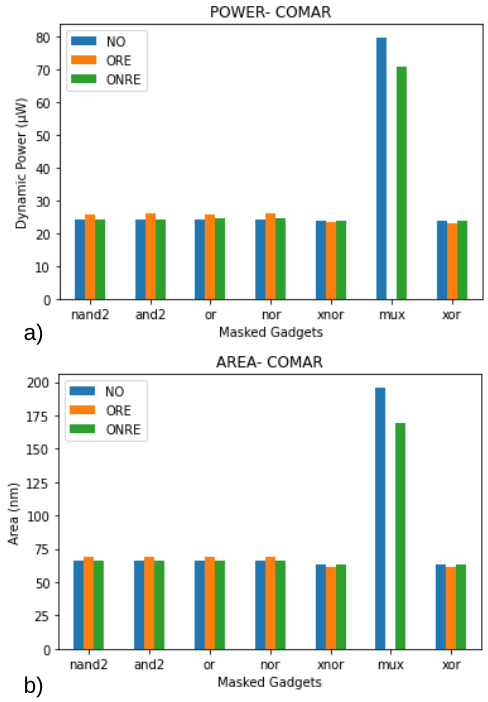

Fig. 5. Comparison of power (a) and area (b) improvements of naïve (NO), Optimized with Retiming (ORE), and Optimized without Retiming (ONRE) methods for COMAR gadgets. The MUX gadget cannot be parsed by Design Compiler after optimization with retiming, hence the empty slots for both graphs.

gadgets consistently show improvement in power consumption. As seen in Table I, HPC3, demonstrates a maximum improvement of over 59%, representing the best-case scenario.

D. Area Optimization

The trend of area optimization aligns with the improved dynamic power values, as illustrated in Fig. 4. The HPC designs demonstrate room for significant optimization opportunities. While area is notably dependent on the standard cell library and its features, it is not unreasonable to expect that the trend would persist with other versions. In the grand scheme of digital design, it may not hold particular relevance. However, optimized area values for masked gadgets help mitigate the inevitable area overhead introduced by the employed masking scheme while guaranteeing side channel resilience.

E. Timing Optimization

In masked gadget optimization frameworks, timing improvements specifically focus on latency [8]. Latency improvements reduce the number of timing cycles required to generate a masked output in the presence of registers and flip-flops. The design reports from the Synopsys Design Compiler show marginal or no changes in delay. The current implementation also strives to maintain latency without compromising security. Latency optimization would significantly improve our framework, and will be explored in future work.

F. PINI-security

PINI security remains reasonably consistent throughout the use of our optimization framework. There are instances in the design when benchmarks cannot be translated by the Design Compiler due to corrupted ABC output. The MUX COMAR gadget is one instance of a design that cannot be translated by the Design Compiler, specifically after optimization with retiming. We attribute this to the repositioning of registers during the retiming process. We also observed that for the naïve experiments, designs from the HPC2 and HPC3 set could not be parsed by VERICA. It is noteworthy that our framework, for both non-retiming and retiming optimization methods of HPC2 and HPC3, are parsed successfully in VERICA, and are PINI secure. This can be may be considered

TABLE I
AVERAGE, MAXIMUM AND MINIMUM PERCENTAGE CHANGES TO DYNAMIC POWER AND AREA CONSUMPTION FOR COMAR AND HPC GADGETS

		COMAR		HPC1		HPC2		HPC3	
		ORE	**ONRE**	**ORE**	**ONRE**	**ORE**	**ONRE**	**ORE**	**ONRE**
Average	**Power**	3.49	-1.67	-7.16	-7.66	-43.84	-34.41	-51.74	-52.14
	Area	2.10	-1.92	-7.74	-7.90	-42.91	-35.63	-52.77	-53.49
Max	**Power**	6.67	0.05	0.25	-5.55	-43.50	-31.95	-45.52	-45.85
	Area	4.42	0.00	-2.65	-5.45	-42.84	-33.73	-47.73	-46.04
Min	**Power**	-3.42	-11.27	-11.98	-8.84	-44.37	-42.38	-59.83	-53.81
	Area	-2.53	-13.43	-9.94	-9.29	-43.03	-43.03	-57.91	-55.45

as an added benefit of our framework. However, the naïve process also reveals a loss of PINI security for AND2 and MUX HPC1 gadgets. Despite these instances, our method overall successfully maintains PINI security. Future work will explore the specific issues causing corrupted ABC output during benchmark translation, as well VERICA's inability to parse certain designs

V. CONCLUSION AND FUTURE WORK

In this article, we present evidence supporting the viability of technology mapping, FRAIGs, and potentially retiming, for the optimization of masked gadgets. Our framework reveals that the ASIC-focused optimization process can be embedded into the EDA pipeline, specifically for masked designs. Although larger implemented designs, such as AES and PRESENT, may not show direct improvements, the success of smaller gadgets suggests the possibility of pursuing a more nuanced and intelligent approach to gadget optimization. Our forthcoming efforts will involve the aforementioned while attempting higher security orders and incorporating improved gadgets into larger benchmarks. Additionally, we highlight its extension to FPGA synthesis and optimization, with reference to the work presented in [11].

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO*. Springer, 1999, pp. 388–397.

[2] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *CHES*. Springer, 2001, pp. 251–261.

[3] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *CRYPTO*, 1999.

[4] D. Knichel, A. Moradi, N. Müller, and P. Sasdrich, "Automated generation of masked hardware," *Cryptology ePrint Archive*, 2021.

[5] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating masking schemes," in *CRYPTO*. Springer, 2015, pp. 764–783.

[6] G. Cassiers and F.-X. Standaert, "Trivially and efficiently composing masked gadgets with probe isolating non-interference," *IEEE TIFS*, vol. 15, pp. 2542–2555, 2020.

[7] D. Knichel, P. Sasdrich, and A. Moradi, "Generic hardware private circuits: Towards automated generation of composable secure gadgets," *IACR TCHES*, pp. 323–344, 2022.

[8] C. Momin, G. Cassiers, and F.-X. Standaert, "Handcrafting: Improving automated masking in hardware with manual optimizations," in *COSADE*. Springer, 2022, pp. 257–275.

[9] L. Wu, Y. Fan, B. Preneel, W. Wang, and M. Wang, "Automated generation of masked nonlinear components: From lookup tables to private circuits," [Online] https://eprint.iacr.org/2023/831 [Accessed: Nov.20, 2023], 2023.

[10] J. Bhandari, L. Mankali, M. Nabeel, O. Sinanoglu, R. Karri, and J. Knechtel, "Beware your standard cells! on their role in static power side-channel attacks," Cryptology ePrint Archive, Paper 2023/920, 2023. [Online]. Available: https://eprint.iacr.org/2023/920

[11] N. Muller, S. Meschkov, D. E. Gnad, M. B. Tahoori, and A. Moradi, "Automated masking of fpga-mapped designs," in *FPL*. Los Alamitos, CA, USA: IEEE Computer Society, sep 2023, pp. 79–85. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/FPL60245.2023.00019

[12] H. Gross, S. Mangard, and T. Korak, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order," in *ACM TIS*. New York, NY, USA: ACM, 2016, p. 3.

[13] T. De Cnudde, M. Ender, and A. Moradi, "Hardware masking, revisited," *IACR TCHES*, pp. 123–148, 2018.

[14] H. Groß, S. Mangard, and T. Korak, "An efficient side-channel protected aes implementation with arbitrary protection order," in *Cryptographers' Track at the RSA Conference*. Springer, 2017, pp. 95–112.

[15] H. Groß and S. Mangard, "Reconciling $d+1$ masking in hardware and software," in *CHES*. Springer, 2017, pp. 115–136.

[16] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *CRYPTO*. Springer, 2003, pp. 463–481.

[17] E. Trichina, "Combinational logic design for aes subbyte transformation on masked data," *Cryptology EPrint Archive*, 2003.

[18] J. Richter-Brockmann, J. Feldtkeller, P. Sasdrich, and T. Güneysu, "Verica - verification of combined attacks: Automated formal verification of security against simultaneous information leakage and tampering," *IACR TCHES*, vol. 2022, no. 4, p. 255–284, Aug. 2022.

[19] C. Wolf, J. Glaser, and J. Kepler, "Yosys-a free verilog synthesis suite," 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:202611483

[20] R. Brayton and A. Mishchenko, "Abc: An academic industrial-strength verification tool," in *Computer Aided Verification*, T. Touili, B. Cook, and P. Jackson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 24–40.

[21] V. Tiwari, P. Ashar, and S. Malik, "Technology mapping for lower power," in *DAC*, 1993, pp. 74–79.

[22] A. Mishchenko, S. Chatterjee, R. Jiang, and R. K. Brayton, "Fraigs: A unifying representation for logic synthesis and verification," 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:9209313

[23] A. Mishchenko, S. Chatterjee, R. Brayton, and P. Pan, "Integrating logic synthesis, technology mapping, and retiming," 01 2006.

[24] Synopsys, "Design compiler concurrent timing, area, power, and test optimization," 2023. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html

[25] N. Müller and A. Moradi, "Prolead: A probing-based hardware leakage detection tool," *IACR TCHES*, vol. 2022, no. 4, p. 311–348, Aug. 2022. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9822

[26] G. Cassiers, B. Grégoire, I. Levi, and F.-X. Standaert, "Hardware private circuits: From trivial composition to full verification," *IEEE TC*, vol. 70, no. 10, pp. 1677–1690, 2020.

[27] D. Knichel and A. Moradi, "Low-latency hardware private circuits," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1799–1812.

[28] ——, "Composable gadgets with reused fresh masks: First-order probing-secure hardware circuits with only 6 fresh masks," *IACR TCHES*, vol. 2022, no. 3, p. 114–140, Jun. 2022. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9696

[29] T. Darwish and M. Bayoumi, "5 - trends in low-power vlsi design," in *The Electrical Engineering Handbook*, W.-K. CHEN, Ed. Burlington: Academic Press, 2005, pp. 263–280. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780121709600500220