IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2025, No. 3, pp. 542–582. DOI:10.46586/tches.v2025.i3.542-582

Design and Implementation of a Physically Secure Open-Source FPGA and Toolchain

Sergej Meschkov[1*], Daniel Lammers[2*], Mehdi B. Tahoori[1] and Amir Moradi[3]

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany
{sergej.meschkov,mehdi.tahoori}@kit.edu
[2] Ruhr University Bochum, Horst Görtz Institute for IT Security, Bochum, Germany
daniel.lammers@rub.de
[3] Technische Universität Darmstadt, Darmstadt, Germany
amir.moradi@tu-darmstadt.de

Abstract. The increasing prevalence of security breaches highlights the importance of robust hardware security measures. Among these breaches, physical attacks – such as Side-Channel Analysis (SCA) and Fault Injection (FI) attacks – pose a significant challenge for security-sensitive applications. To ensure robust system security throughout its lifecycle, hardware security updates are indispensable alongside software security patches. Programmable hardware plays a pivotal role in establishing a robust hardware root-of-trust, serving to effectively mitigate various hardware security threats. In this paper, we propose a methodology for the design of a reconfigurable fabric and the corresponding mapping toolchain, specifically tailored to hardware security. This approach offers resistance to various malicious physical attacks, including SCA and FI, addressing each threat individually. As a case study, we propose a resulting fabric that implements a combination of first-order Boolean Masking and hiding countermeasures to provide strong protection against SCA attacks and enables the detection of fault injection attempts. In particular, we present how reconfigurable secure gadgets can be realized employing a reformed variant of the LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL) hardware masking scheme and a modified version of Wave Dynamic Differential Logic (WDDL) to be composed into a fabric. We also show how any basic Hardware Description Language (HDL) design is automatically mapped to the primitives of our fabric, embedding provable hardware security, and bypassing the necessity for hardware security proficiency in this process. It is worth mentioning that our fabric requires approximately 85% less area to map a secure design compared to conventional Field Programmable Gate Arrays (FPGAs). A practical security evaluation of our secure fabric implementation on a real FPGA target board, using Test Vector Leakage Assessment (TVLA), demonstrated no SCA leakage over 100 million traces.

Keywords: Hardware · FPGA · FABulous · Secure Fabric · Side-Channel Analysis · WDDL · Masking · Dual-Rail Pre-charge Logic · LMDPL · Fault Detection · Low-Latency

1 Introduction

In today's rapidly expanding digital world, many aspects of daily life rely solely on information technology infrastructure, making security concerns greater than ever. Although these

*These authors contributed equally to this work.

Received: 2025-01-15 Accepted: 2025-03-15 Published: 2025-06-05

systems include both hardware and software, software security has been studied for a longer time. This is because hardware is assumed to be secure for most users (a root-of-trust), and almost all security attacks target software. However, as software security improves and compromising hardware becomes more studied, hardware security poses a significant vulnerability to the whole system. Evidence of hardware security breaches emphasizes the growing importance of hardware security in the academic, industry, and government sectors.

Physical attacks, such as Side-Channel Analysis (SCA) attacks [Koc96, KJJ99, AARR03], and Fault Injections (FIs) attacks [BDL97, BS97] pose a significant challenge in security-enabled applications, especially for those operating in hostile environments. To grant some level of security of the system as a whole, both software and hardware need to be protected. Although it is impossible to fully prevent any SCA leakage, suitable countermeasures can make SCA attacks practically infeasible or allow detection of fault injection attempts. Such countermeasures usually incorporate masking sensitive information and hiding techniques.

Protection can be added in a dedicated part of an System on Chip (SoC) or in the form of an embedded Application Specific Integrated Circuit (ASIC) that handles sensitive information and executes cryptographic algorithms. However, such secure hardware is generally highly customized and tailored to a specific cryptographic algorithm or a set of functions. Hence, adapting to fast-developing security threats is difficult to impossible. While software updates are possible as a common reaction to new security threats, hardware usually remains unchangeable during its entire life cycle. As a result, secure hardware solutions can become outdated very quickly.

The hardware inflexibility issue can be solved with reconfigurable fabrics, e.g., Field Programmable Gate Arrays (FPGAs), that play an important role due to enabling rapid time-to-market by flexibility and updateability. FPGAs are often used for security applications and allow hardware "security patches" similar to software applications. Nevertheless, commercial FPGAs are not specialized for security applications and many security issues still need to be resolved [SRR+23, SGMT18b, KGT18, ZS18, GKT19, SMTG23].

Integrating known countermeasures to physical attacks in conventional FPGAs is challenging and negatively affects performance significantly compared to solutions in ASICs, e.g., high area, low throughput, high power consumption, increased latency, etc. Implementation and mapping of such security schemes is usually re-done manually for every cryptographic algorithm and fabric architecture, which requires a high level of hardware security proficiency and development effort. In addition, developing designs dedicated to an FPGA platform is limited by the vendor tools and the primitives available on the device. Therefore, countermeasures might not provide the desired security level or lead to a high inefficiency with respect to performance.

For instance, there is a body of work focused on hiding countermeasures for conventional FPGAs, with some efforts exploring improvements through balanced placement and routing techniques [AMM10, AMM13, ABM+14, MI14a], while others propose adjustments to the FPGA architecture or modifications to the implementation of Look-Up Tables (LUTs) [CGH+08, BRF+07, MGP09]. However, these approaches are still not easily applicable to most commercially available FPGAs due to the aforementioned limitations. Moreover, since they rely solely on hiding countermeasures, the provided side-channel resistance is not verifiable through a formal security model and can be considered as insufficient.

1.1 Our Contributions

In this work, we propose a solution to address the challenges elaborated above, which serves as a roadmap to build a foundation enabling efficient realization of reconfigurable SCA-secure devices. The main aspects of our contribution are summarized as follows:

Secure Reconfigurable Fabric Design Methodology. We propose an approach to

designing reconfigurable fabrics and a respective toolchain tailored with hardware security in mind. The concept provides resistance at the hardware level to various malicious physical attacks, such as SCA and FI attacks. A fabric designed by our methodology can act as the root-of-trust in SoCs in order to implement and execute cryptographic algorithms or functions demanding high security. Our technique preserves physical security at the gate level and is based on provable secure gadgets, subcircuits which are composable without violating security assumptions of the resulting circuit. Generally, many known gadget-based hardware security schemes can be utilized in the design process for such a fabric.

Fully Automated HDL Design Flow for Bitstream Generation. In fact, the mapping of a netlist to the fabric is crucial to maintaining security properties. If security assumptions are violated, physical security cannot be maintained. However, we made a customized toolchain to automate mapping as well as place-and-route for specific fabric architectures. As a result, no expertise is required in the hardware security domain, and insecure Hardware Description Language (HDL) designs are automatically mapped to achieve physically secure operation.

Entirely Open-Source Toolchain. We utilize AGEMA [KMMS21], FABulous [KDH+21], Yosys [WGK13], nextpnr [SHW+19], and BitMan [DPHK17, Bit]. These tools pave the way for efficient and customizable open-source development without vendor limitations. Our goal is to enhance security through comprehensive evaluation and expansion within the broader research community.

Secure Gadget-based Fabric Implementation. As a case study, we implemented a first-order SCA-secure reconfigurable fabric based on Wave Dynamic Differential Logic (WDDL) [TV04a] and LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL) [SBHM20a] gadgets with the assistance of the above-mentioned open-source tools. The gadgets are adjusted to enable fault detection on the dual-rail signals according to the concept of Totally Self-Checking Circuits [Lam83]. In this study, we present the complete design flow for secure fabric design. In particular, we explain the design of gadgets and how they are assembled into a reconfigurable fabric. Furthermore, we show the result of fully automated bitstream generation flow for several cryptographic algorithms given their unprotected HDL designs. Moreover, we conduct an area improvement estimation. Compared to a conventional FPGA architecture (FABulous reference implementation), our mapped HDL designs result in reduced area consumption by approximately 85%. Test Vector Leakage Assessment (TVLA) on real power measurements of an emulated FPGA platform confirms no leakage detection over 100 million traces.

2 Background

This section focuses on prior knowledge relevant to this work and provides an overview of SCA and FI attacks, as well as known security concepts to mitigate underlying threats, with an emphasis on FPGA platforms. Additionally, the open-source FPGA design framework utilized in this work is outlined.

2.1 Notation

In the context of this work, lowercase letters such as x denote single-bit (binary) variables in $\mathbb{F}_2$. The subscript indicates the rail of a variable in dual-rail form. The original value is given by x_t, while x_f corresponds to its complement. In condensed format, the two values are written as the tuple $x = (x_\mathrm{t}, x_\mathrm{f})$. In case of a shared variable, the index is explicitly

stated in the superscript, e.g., x^0. Capital letters, such as X, represent a quadruple of variables split in two shares in dual-rail form each, i.e., $X = (x_t^0, x_f^0, x_t^1, x_f^1)$. The $\oplus$ symbol is utilized for additions in $\mathbb{F}_2$.

2.2 Threat Model

In this section, we review physical attacks, such as SCA and FI attacks, as well as exploitable characteristics, particularly in relation to FPGA platforms.

2.2.1 Side-Channel Analysis

SCA attacks have shown that even if the underlying security algorithm is mathematically proven to be secure, the insights gained from the observable physical effects of its implementation during the processing of sensitive data can be used by an attacker to compromise its security. Such observable side channels are, for example, power consumption [KJJ99], electromagnetic emanations [AARR03], runtime behavior [Koc96] and many others.

These attacks often target devices to which an attacker can gain physical access, for example, to measure their power consumption with an oscilloscope. However, some power analysis and electromagnetic attacks do not require dedicated measurement equipment and have been shown to be feasible from within the same chip [ZS18, OD19], the same power domain [SGMT18a], or in close proximity [KP13, CPM+18, ZZL+22]. This enables some remote SCA attacks, such as the attack exploiting signal jitter in a communication link [SMTG23]. Especially, multi-user FPGA environments in the cloud are seen as a well-suited target for such remote SCA attacks, since a malicious FPGA user can implement different types of power sensors, such as Time-to-Digital Converter (TDC) [SGMT18b], Ring Oscillator (RO) [ZH12] or routing based [SGS23], to attack other users or components in the same Power Distribution Network (PDN).

Certain physical defaults are nowadays well known to cause or amplify SCA leakage. This includes glitches [MPO05], transitions [CGP+12], and coupling effects [DCBG+17] during normal operation of the device. In the context of this work, we conduct the security analysis on gate level. Hence, we do not discuss couplings, but cover glitch and transitions.

Glitches. A glitch is a transient physical effect associated with unintended switching activity in Complementary Metal Oxide Semiconductor (CMOS) circuits due to different timings in signal propagation. In each clock cycle, transient changes due to glitches cause significant dynamic power consumption until the internal state and output of a circuit reach a stable state, i.e., ready to be stored in registers. This behavior (pattern of glitches) depends on the processed (secret) data and may cause or amplify exploitable leakage.

Transitions. When the result of a computation overwrites a stored value in a register, the associated power consumption is dependent on both values. This potentially results in secret-dependent power consumption and is referred to as transitional leakage in the literature.

2.2.2 Fault Attacks

Compared to SCA, FI attacks [BDL97, AK96, BS97] are considered as active attacks, in which the attacker actively attempts to make disturbances for the device. Such disturbances include voltage and clock manipulations, laser shots, electromagnetic pulses, or sudden temperature variations. Subsequently, the attacker analyzes gathered information from the correct (non-faulty) and/or incorrect (faulty) outputs to recover some information about the secret, e.g., by Differential Fault Analysis (DFA) in [BS97].

Fault attacks traditionally require access to the target device, while some attacks exploit unintended use of system components to induce voltage drops in the shared PDN. For example, in [KGT18] the proposed attack utilizes ROs to cause faults in a multi-user FPGA. Another work targets a Central Processing Unit (CPU) from an Embedded FPGA (eFPGA) in an SoC system with a power-hammering circuit [GKG+23].

2.3 SCA Countermeasures and Security Proofs

In recent years, a few countermeasures have been developed that address security flaws caused by physical effects, and some approaches have been introduced for proper security evaluation. The techniques on which this work is based are explained in this section.

2.3.1 Hiding

A commonly known category of countermeasures is hiding, which attempts to decrease the Signal to Noise Ratio (SNR), so that the exploitable leakage is hidden below the noise level. This can be achieved by (1) noise amplification (addition of randomized switching activity or instructions) or (2) signal reduction (power equalization for all operations independent of processed data). Such concepts are well suited for hardware implementations. In particular, power-equalization schemes (also called DPA-resistant logic styles) are primarily designed for ASIC platforms deployed at low abstraction levels, such as transistor to gate levels. Examples of such logic styles are Sense Amplifier Based Logic (SABL) [TAV02], Wave Dynamic Differential Logic (WDDL) [TV04a], Masked Dual-Rail Pre-charge Logic (MDPL) [PM05], and Dual-Rail Random Switching Logic (DRSL) [CZ06], which predominantly rely on the Dual-Rail Pre-charge Logic (DRP) concept.

Dual-Rail Pre-charge Logic (DRP). Such schemes replace single wires that drive the signal of a variable, e.g., x, with two wires. They carry the dual-rail representation $(x_\mathrm{t}, x_\mathrm{f})$, where x_t is equivalent to the original signal and x_f to its complement. Consequently, logic functions are replaced by dual-rail modules that process both rails in parallel and generate dual-rail outputs. These logic styles consist of two alternating phases, namely the pre-charge and evaluation phases. The pre-charge phase sets every input and internal signal to a default state (usually 0), followed by the evaluation phase, where actual data is given to and processed by the circuit. Every gate behaves monotonically, i.e., in each phase every gate output toggles in one direction only. This way, the number of toggles in the circuit is constant in both phases and hence becomes data independent. A valid dual-rail encoding is defined as follows:

$$(x_\mathrm{t}, x_\mathrm{f}) = \begin{cases} (0,1) & \Rightarrow x = 0 \\ (1,0) & \Rightarrow x = 1 \\ (0,0) & \Rightarrow x \text{ in pre-charge phase} \end{cases}$$

A proper DRP circuit eliminates the occurrence of glitches and avoids transitions entirely, since the evaluated results are always overwritten by the default state.

2.3.2 Wave Dynamic Differential Logic (WDDL)

WDDL [TV04a] is a well studied DRP scheme that replaces gates with their dual-rail counterpart composed of standard cells. It targets an equal power consumption independent of the processed data. However, multiple follow-up works pointed out flaws in practice. First, it has been shown that its time-of-evaluation is data dependent [SS06]. Subsequently, a modified variant (DPL-noEE: "no early-evaluation") has been introduced for FPGAs in [BGF+10] to avoid its *early propagation* effect. However, the approach does not solve

the same issue during the pre-charge phase. Another work introduced Asynchronous Wave Dynamic Differential Logic (AWDDL) [MI14b] that fully avoids this issue also only for FPGAs.

In short, DRP as a standalone countermeasure does not prevent information leakage in practice. Slight routing imbalances and manufacturing variations lead to different capacitive loads that contribute to total power consumption. Therefore, dual rails should assure balanced routes to minimize (but not prevent) leakage. For example, the fat-wire approach [TV04b] can be employed to minimize imbalanced delays in ASIC designs. However, the wire capacitance of the dual rails still differs slightly in the fabricated designs (due to process variations), potentially causing exploitable leakage. As a side note, the combination of DRP and a suitable masking scheme can provide a considerably high level of resistance in practice [MW15].

2.3.3 Boolean Masking

The concept behind Boolean Masking [CJRR99] is to randomize intermediate values of a cryptographic algorithm to make observable SCA leakages independent of predictable processed data. Based on secret sharing [Sha79], d-th order Boolean Masking splits a sensitive message m into $d+1$ shares. The sharing consists of d individual and independent shares distributed uniformly at random. The last share x^d is computed as $x^d = (\bigoplus_{i=0}^{d-1} x^i) \oplus m$. Due to uniformity, an adversary who has access to at most d shares is unable to recover any information about m. Therefore, a $d+1$ sharing is considered d-order secure. To maintain security during the entire computation, many masking schemes require fresh randomness in every clock cycle.

In this work, we aim for first-order security, i.e., a secret m is concealed with a random mask r, such that $x = x^0 \oplus x^1$, with $x^0 = r$ and $x^1 = m \oplus r$. Our modified masking schemes operate in DRP, such that fresh randomness must be provided every other clock cycle, i.e., updated for every evaluation phase.

2.3.4 Probing Models

In order to ease the evaluation and prove the security of masked implementations, some models have been defined to abstract the leakage behavior of the circuits. In the d-probing model [ISW03] a formal adversary is defined – called the d-probing adversary – that is capable of observing any d circuit wires at a specified clock cycle simultaneously. According to this model, a circuit is considered d-probing secure if and only if a d-probing adversary is unable to recover any information about sensitive variables. As this approach relies on the observation of stable signals only, an extension – called robust probing model [FGP+18] – also covers known physical defaults, namely couplings [DCBG+17], transitions [CGP+12], and glitches [MPO05]. In this model, a transition-extended probe adds a time dimension and records the value of the same wire in consecutive clock cycles. It is noteworthy to highlight that – due to the underlying DRP scheme employed in our methodology – transition-extended probes do not add any additional leakage/information. Hence, the security analysis in this work utilizes glitch-extended probes explained below.

Glitch-extended Probing Model. Since the number of glitches and their pattern depend on all changes occur on the cirtcuit's inputs, a glitch-extended probe placed on a wire propagates backwards to the last synchronization point (registers) or primary inputs of the circuit. By that, a formal adversary simultaneously probes all signals that contribute to the observed wires in the same clock cycle.

2.3.5 Gadget-based Masking and Composability

With increasing circuit complexity, especially in higher security orders, the efficient application of masking becomes increasingly difficult and error-prone. Unfortunately, designing a large provably secure circuit becomes infeasible quickly. A circuit composed of multiple individually-secure subcircuits may violate security assumptions under robust probing model. The probes placed on the primary outputs of a composed circuit is extended up to the last synchronization point, even across multiple subcircuits due to glitch probe propagation. Composability notions generally limit such probe propagations to ensure that security in the model is preserved when the subcircuits are composed. A provably secure subcircuit that fulfills specific conditions to be composable without violation of security assumptions is commonly called a secure (composable) gadget. Such a gadget may implement a function as small as a simple two-input logical AND gate [CGLS20] or as large as an entire S-Box [KSM22, VDB+24]. Notable composable notions in the recent literature are Strong Non-Interference (SNI) [BBD+16], Glitch Strong Non-Interference (GSNI) [DMBR19], and Probe-Isolating Non-Interference (PINI) [CS20]. In this work, we ensure security under the GSNI notion.

d-Glitch Strong Non-Interference (GSNI). A gadget is considered d-GSNI secure if and only if any p_1 glitch-extended probes placed on internal wires and p_2 glitch-extended probes placed on output wires, with $p_1 + p_2 \leq d$, reveal no secret information.

2.3.6 LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL).

The gadget-based LMDPL masking scheme that combines DRP and masking to achieve first-order security has been originally introduced in [LMW14a]. The proposed scheme operates in a glitch-free manner, avoids the *early propagation* effect, and can be implemented on FPGAs or using ASIC standard cell libraries. In a follow-up work [SBHM20a], the authors formally proved its security under the 1-GSNI composability notion and introduced an Advanced Encryption Standard (AES) implementation that needs (in average) a single clock cycle per cipher round. The corresponding experimental evaluations confirmed first-order SCA resistance and even indicates a high level of resilience against higher-order attacks. Further third-party investigations confirmed its security [MLM24].

Conceptually, the scheme is divided into the mask table generation and the operation layer. Considering a two-input non-linear (AND) gate with the inputs being x and y and output $z = xy$, the mask table generation layer processes the first shares x^0 and y^0 in their single-rail representation. It further requires a single fresh random bit r to blind intermediates stored in a register in between the layers while setting the first output share $z^0 = r$. The operation layer employs the DRP protocol and processes dual-rail shares $(x^1_\mathrm{t}, x^1_\mathrm{f})$ and $(y^1_\mathrm{t}, y^1_\mathrm{f})$ as well as the intermediates given by the table generation layer to issue the second output share also in DRP form, i.e., $(z^1_\mathrm{t}, z^1_\mathrm{f}) = (xy \oplus r, \overline{xy} \oplus r)$.

2.4 Automated Generation of Masked Hardware (AGEMA)

The publicly available AGEMA framework [KMMS21] enables mapping of an arbitrary unprotected gate-level netlist of an ASIC design to a gadget-based d-order masked circuit. Therefore, even inexperienced hardware designers with limited time and budget are enabled to create provably secure masked hardware circuits while avoiding security flaws. AGEMA supports various securely composable gadgets such as HPC1/2/3 [CGLS20, KM22], GHPC and $\mathrm{GHPC}_{\mathrm{LL}}$ [KSM22], and others. In addition, this tool is capable of applying the masking on selected parts that are annotated by the designer, i.e., sensitive parts of the circuit. Annotations indicate that secret data is processed, such that uncritical components are left unmasked, e.g., control logic, which can reduce the overheads significantly. Furthermore,

AGEMA can optimize the results according to specific requirements in terms of area utilization, clock cycle count, and randomness.

An extension of the original AGEMA further supports efficient and automated mapping of the given gate-level netlists to masked circuits for FPGA platforms [MMG+23]. This extension is optimized for better usage of FPGA resources like LUTs and registers. For the same annotated given design and the same countermeasures, it achieves results with significantly less resource utilization and power consumption when mapped to an FPGA, compared to the original AGEMA framework.

2.5 FABulous

FABulous [KDH+21] is a framework designed for eFPGA development. It integrates other well-known open-source tools such as Yosys [WGK13], ABC [SG25], VPR [tRVP25] and nextpnr [SHW+19]. By this, it is capable of generating the eFPGA fabric for chip fabrication, matching bitstream generation and testing. The framework organizes its eFPGA fabrics into a grid of tiles. Each tile type is described via three components: (1) primitives, (2) wires, and (3) switch matrices. Primitives are combinational as well as sequential modules that implement the main functionality of the tile, and each tile may consist of one or multiple primitives. Defined wires are oriented in specified directions and, together with switch matrices, are used for configurable tile interconnections, while some wires can also be hard-wired.

The fabric designer implements different customized tile types and organizes them into the layout of a reconfigurable fabric. Usually, the same kind of tiles are placed row- or column-wise. Interconnections to adjacent tiles and global components are also specified for each tile type.

For end-users, FABulous supports the open-source F4PGA toolchain[1], which utilizes Yosys [WGK13] and nextpnr [SHW+19]. This way, mapping of end-user Register Transfer Level (RTL) designs, as well as place-and-route, are fully automated. Eventually, the compatible bitstream generation is performed via BitMan [Bit].

3 Methodology: Secure Fabric Design

In this section, we propose a general design process for reconfigurable fabrics tailored to the prevention of physical attacks. We provide resistance against individual SCA and FI attacks. The underlying security concepts are integrated into the design process, supported by open-source tools. Further, the fully automated flow of mapping an unprotected HDL netlist to secure fabric primitives is explained. In this work, we confine the scope to the primary goals, hence we assume a non-malicious bitstream. Details that are similar and in line with the original FABulous FPGA architecture and minor adjustments that give no further insights into the tailored approach are omitted.

3.1 Security Requirements

Our security requirements are derived from a comprehensive adversary model, which establishes assumptions about the potential attackers' knowledge, capabilities, resources, and limitations. By anticipating the methods that adversaries might use, we ensure that the system's defenses are designed to withstand a wide range of sophisticated attacks. The assumptions derived from this play a crucial role in shaping the security measures to meet the anticipated challenges. A detailed list of these adversarial assumptions, which form the foundation for the system's security requirements, is provided below:

[1] https://f4pga.org/

Adversary Knowledge

- The adversary has knowledge about the general architecture of the reconfigurable device.
- The adversary has knowledge about the programmed bitstream configuration, including the cryptographic algorithms or computational processes it executes.

Adversary Capabilities

- The adversary has physical access to the target device and might depackage the chip to get access to its surface or is in close proximity to the chip.
- The adversary can observe the device inputs and outputs.
- The adversary is able to collect various types of SCA information, such as power consumption, electromagnetic emissions, timing information, and temperature, using appropriate sensors and measurement equipment.
- The adversary has the ability to process and analyze the collected SCA traces to infer sensitive information.
- The adversary possesses the necessary equipment to induce faults, such as voltage and clock manipulation tools, electromagnetic probes, and laser beams.
- The adversary can synchronize fault injections with specific moments during the device operation, based on timing information and observable behavior.
- The adversary can manipulate the device operating environment, e.g., temperature, power supply voltage, additional EM noise, etc.

Adversary Limitations

- Based on the defined security order d (i.e., the order of the applied masking scheme), the adversary is limited to place d probes on the internal signals of the device and monitor the corresponding values at specific clock cycles per encryption/decryption. In this work, we deal with first-order security, i.e., $d = 1$.
- The adversary cannot directly modify the internal state of the device (e.g., memory contents and register values) except through induced faults.
- The adversary does not have the ability to modify the hardware architecture of the device or its configuration, including the bitstream.
- The adversary can either observe SCA leakages or inject faults, but not both simultaneously.

As mentioned before, the system must be fully reconfigurable in the field, capable of implementing an arbitrary circuit while maintaining security properties. The only constraints are available resources, similar to a conventional FPGA. In this way, the following properties, which are essential for maintaining security, are preserved:

- Patches to algorithms implemented in secure hardware.
- Full exchangeability of security-related primitives, algorithms, protocols, etc.

Furthermore, there should be no need for the engineers/developers to have detailed knowledge about the security measures implemented in the fabric. The toolchain should be capable of automatically taking an unprotected target HDL design and map it to secure primitives, as well as handling placement and routing according to the security measures. This ensures robust protection against SCA and FI attacks at the hardware level. Notably, automating the application of security measures to the design in such a manner significantly reduces the risk of human error in the process.

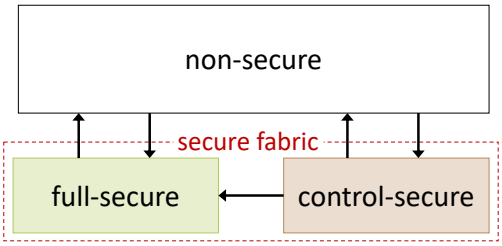
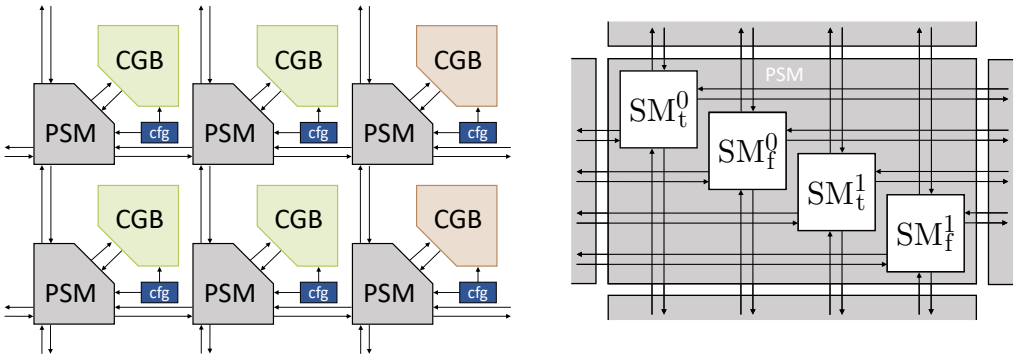


Figure 1: Schematic of regions with different security level.



(a) Secure fabric organization of different CGB types, PSMs and corresponding configuration memory (cfg).

(b) Share-wise domain separation and dual-rail preservation in a PSM.

Figure 2: FPGA-like grid structure of secure fabric.

3.2 Platform Description

The secure reconfigurable fabric is supposed to act as a root-of-trust in an SoC. It should provide protection during the execution of cryptographic algorithms or other functions demanding high security. Fundamentally, this fabric implements a combination of first-order masking and hiding countermeasures to provide strong protection against SCA attacks and enables detection of fault injections. Usually, not every part of an implementation needs SCA protection. Therefore, we consider multiple regions with different security levels (cf. Figure 1), namely full-secure, control-secure, and non-secure. This concept is similar to microprocessors with TrustZone technology [tru]. In this way, the associated total overhead and cost of the fabric are reduced, since building blocks require more area with increasing security.

If protection against physical attacks is not required for a part of the implementation, the non-secure region is the trivial choice. It consists of non-secure logic, such as any kind of processor core or conventional FPGA. The control-secure region is supported by a fault detection mechanism as a countermeasure against FI attacks. This is mandatory to protect parts of the secure design, such as the control logic of a cipher, where secret data are not processed, but faults may directly affect them. Therefore, we propose the application of DRP logic and fault propagation based on invalid DRP states in this region to support a wide range of fault detection. The full-secure region implements a combination of first-order secure masking and DRP. Fault occurrence is intended to propagate from the control- and full-secure regions to a fault detector located at the primary outputs. This will be extensively explained in Section 3.6.

In general, different approaches are possible with either less or more layers of security (including higher-order masking) to suit the desired/required protection level of an actual

SoC. In this paper, we limit our focus to the full-secure and control-secure regions, which fulfill the purpose of an embedded secure reconfigurable fabric. We omit dealing with a non-secure region, as several solutions (like the original FABulous FPGA) can easily handle this. In short, we assume all inputs (and outputs) of our fabric follow a DRP protocol while some of them are additionally masked.

3.3 Secure Reconfigurability

To achieve reconfigurability of the fabric and maintain provable security at the same time, secure composable gadgets are used. The main idea, which is shown Figure 2(a), is to include each gadget in a Configurable Gadget Block (CGB) while the CGBs are organized in an FPGA-like grid structure and interconnected through configurable Parallel Switch Matrices (PSMs) (explained in detail in Section 3.4). CGB and PSM configuration data are stored in dedicated configuration memory (denoted as "cfg" in Figure 2(a)), which is programmed by the bitstream. Since gadgets are provably secure subcircuits that are composable under a specified composability notion without violating the security assumptions, any valid fabric configuration should maintain SCA-secure functionality.

For our purposes, we consider two general gadget types. The linear gadgets implement the XOR function and non-linear gadgets realize the AND functionality. Both of which require the inputs and the output to follow DRP logic. As a dual-rail inversion is implemented by swapping the rails, the gadget reconfiguration is realized by individually configurable multiplexers on the inputs and output. This would allow realizing XOR and XNOR with the linear gadget and AND, NAND, OR, and NOR with the non-linear gadget. The basic structure of a full-secure reconfigurable gadget is shown in Figure 3. The select signal of each multiplexer is set by a particular bit in the programmed bitstream. For the masked gadgets, only one share is invertible, since inverting both shares would result in the same orignal value and cause unnecessary overhead, i.e., $x = x^0 \oplus x^1 = \overline{x^0} \oplus \overline{x^1}$. Multiplexers at the reconfigurable gadget output rails control whether register stages are used. Specifically, two register stages are utilized to support alternating pre-charge/evaluation phases of DRP logic. Therefore, by composing these linear and non-linear gadgets followed by the aforementioned registers, any Boolean function and sequential circuit can be realized. More implementation details on a particular case study gadget designs are given in Section 4.2.

In general, a fabric CGB can consist of multiple gadgets and gadget types, by utilizing some local routing or fixed wiring. However, this is an application-specific optimization that does not affect security. Therefore, we consider only one gadget per CGB to limit the complexity.

3.4 Connectivity Constraints

To maintain provable security and functional correctness for any design, interconnections between the secure reconfigurable gadgets are restricted. The constraints are derived from composability notions and the gadget-based scheme used. For gadgets that rely on a combination of masking and DRP, this can be summarized as follows. Note that other schemes and composability notions may result in different connectivity constraints.

- *Share-wise domain separation.* A signal belonging to one share domain in one gadget should not be connected to another share domain of another gadget.
- *Dual-rail preservation.* It should be assured that dual-rail signals are interconnected in the correct way and no unintended rail swap occurs.

We propose to employ PSMs, illustrated in Figure 2(b), which are in line with the above constraint list. A PSM is a structure of parallel switch matrices in which only

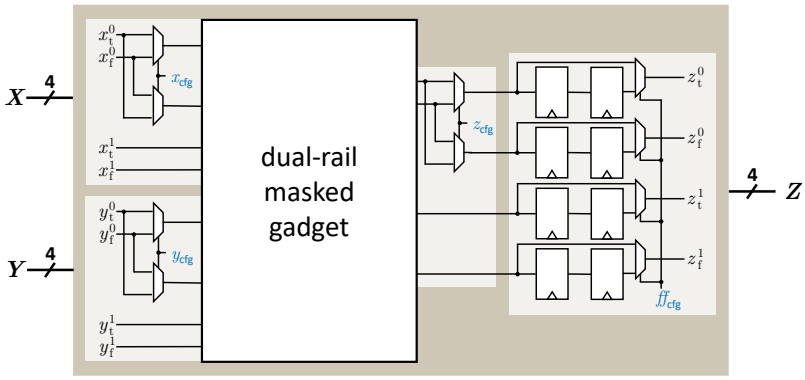


Figure 3: Full-secure CGB with DRP logic and first-order masking. A central gadget implements linear or non-linear functionality. Multiplexers enable inversion of inputs and output. In addition, four multiplexers at the outputs control whether register stages are used. Configuration bits are highlighted in blue.

wires with the same share domain and the same rail indicator are interconnected, e.g, a configuration for $\mathrm{SM}_{\mathrm{t}}^{0}$ determines the true rail connections of the share domain 0. This results in several smaller switch matrices that are interconnected only with other switch matrices or CGB inputs and outputs, each forming an independent grid.

Lemma 1. *The routing of signals through distinct physical wires and dedicated switch matrices, corresponding to their respective share domain and rail, ensures complete physical separation. This assures that both share-domain-based separation and dual-rail preservation are maintained, making any violation physically impossible.*

Proof. Each gadget input or output, corresponding to the share domain $\sigma \in \{0, 1\}$ and the rail $\lambda \in \{\mathrm{t}, \mathrm{f}\}$, is connected exclusively to the wires x_{λ}^{σ} dedicated to that specific share domain σ and rail λ. These wires are further connected only to a switch matrix $\mathrm{SM}_{\lambda}^{\sigma}$ that also corresponds to the same share domain σ and rail λ. In particular, in the control-secure region, each dual-rail signal is carried by two physical wires, i.e., x_{t}^{0} is routed through $\mathrm{SM}_{\mathrm{t}}^{0}$ while x_{f}^{0} is routed through $\mathrm{SM}_{\mathrm{f}}^{0}$. In the full-secure region, the additional wires x_{t}^{1} and x_{f}^{1} are similarly routed through $\mathrm{SM}_{\mathrm{t}}^{1}$ and $\mathrm{SM}_{\mathrm{f}}^{1}$, respectively. In this way, it is physically impossible to violate share-domain-based separation and dual-rail preservation. □

Certain connectivity aspects demand particular attention, such as connections between different security regions. In particular, we consider a unidirectional connection from the control-secure to the full-secure region and no connection in the opposite direction. Processing secret data in the control-secure region is not allowed according to the platform description given in Section 3.2 and may cause SCA leakage.

3.5 Random Number Generator (RNG)

For masking to be applied, random numbers are required. In addition, many masking schemes additionally require fresh masks updated every clock cycle to maintain the claimed security. We consider randomness to be provided by a dedicated module outside the actual reconfigurable fabric. The True Random Number Generator (TRNG) alone seems to be a less efficient solution, as they are either quite slow or require extensive resources. In practice, this results in high costs for each true random bit generation. Since we need a large number of random bits, a combination of TRNG and scalable Pseudo-Random Number Generator (PRNG) is suitable for such a task. It is a common strategy to use more efficient PRNGs to stretch the initial truly random seed into many pseudo-random

bits. It should be noted that the statistical quality of PRNGs (such as passing some National Institute of Standards and Technology (NIST) tests [RSN+01]) is important for masking schemes to keep their security promises.

In our proposed fabric design, externally provided randomness is expected to seed a scaled internal PRNG, which continuously generates fresh randomness during runtime and is directly hardwired to all non-linear gadgets. In addition, all inputs that enter the full-secure region from outside (i.e., primary inputs) should be provided in the masked representation.

3.6 Fault Propagation and Self-Checking

Dual-rail logics provide an opportunity for self-checking. Considering $(x_\mathrm{t}, x_\mathrm{f})$ as the dual-rail representation of x, we define a valid encoding based on the two valid states $(0, 1)$ and $(1, 0)$ as well as the two invalid states $(1, 1)$ and $(0, 0)$. By checking the correctness of dual-rail states in the circuit, faults are detected up to a certain extent. If any dual-rail state in the evaluation phase corresponds to an invalid encoding, a fault is detected.

Since examining the validity of every dual-rail is quite complex, the concept of Totally Self-Checking circuits [Lam83] can be applied to ensure fault propagation. In this way, any invalid state should propagate through the circuit, making the output state of all successor gates also invalid. This would allow instantiating fault detector modules only at the primary outputs to evaluate the validity of the dual-rail states and hence detect faults. Note that neither WDDL nor LMDPL gadgets fulfill this requirement. In order to be in line with these principles, we adjust the secure gadgets to follow this fault propagation scheme. If no fault occurs, for every valid gadget input, we expect a valid gadget output. In case of an invalid input encoding, the output should also be invalid. In a composed circuit, this behavior causes fault propagation through the entire design. Once an invalid state propagates to the outputs and is detected, all embedded fabric IO ports can be disabled. It is important to note that the output evaluation requires an additional clock cycle. Consequently, the final outputs must be delayed by at least one clock cycle to ensure that the disable operation is executed before the outputs are revealed to the IO ports. Additionally, since gadgets represent atomic functions but are not actual atomic gates, any individual atomic gate within a gadget could potentially be faulted. Therefore, it is desirable to design the internal circuit of the gadget such that any fault within either results in a faulty dual-rail state at the output or does not affect the output correctness. Otherwise, this would limit the fault detection capabilities. In total, any effective single fault within a gadget is propagated to the primary output of the composed circuit, due to the fault propagation property of each gadget.

However, a deployed fault detector module may violate the security assumptions and cause SCA leakage if share domain separation is not taken into account. In particular, the XOR operation applied to dual rails of any share always results in a logical '1' if a valid encoding is present during the evaluation phase. However, considering the glitch-extended probing model, a check on all outputs potentially leaks information through combining all shares of different domains.

It should be noted that such a technique cannot detect any faults injected on both rails of the same share simultaneously, e.g., changing a valid dual-rail signal $(0, 1)$ to another valid state $(1, 0)$. On the other hand, this is not always easily possible without affecting other signals. In other words, even if a fault injection causes a valid encoding to toggle to another valid encoding, it is enough to have a single invalid $(0, 0)$ or $(1, 1)$ signal in the entire circuit to detect the injected fault.

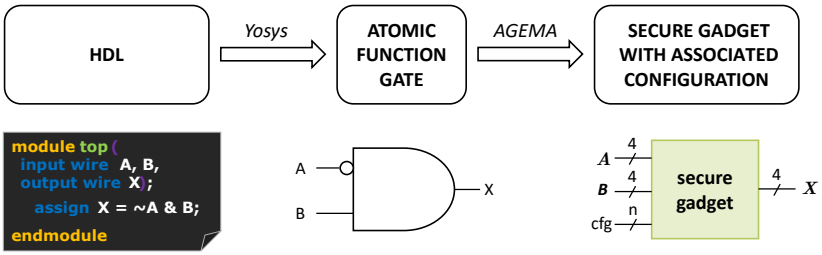


Figure 4: Unprotected HDL design mapping to secure gadgets assisted by Yosys and AGEMA.

3.7 Toolchain and Design Mapping

Since in the proposed approach resistance against physical attacks is provided at the gate level, regardless of the design mapped to the fabric, its implementation provides the desired level of provable security during runtime. During the design mapping step for a provided secure fabric, the developer of the HDL design to be mapped does not require any specific knowledge about hardware security. Only the primary inputs have to be annotated as secret or public in the unprotected HDL design. Then, the tool automatically determines which parts of the given HDL are mapped to which security region based on these annotations. Figure 4 illustrates a flow of how mapping of a given HDL design to secure gadgets is done. As a first step, the HDL design needs to be mapped to specified atomic gates such as AND, NAND, OR, NOR, XOR, XNOR and NOT. This can easily be done with Yosys. In the next step, each of these gates is replaced by a reconfigurable gadget, explained in Section 3.3. This is realized by the instantiation of Verilog primitives according to the gadgets with the corresponding generic parameters in the netlist. The result is later automatically translated into the particular configuration in the further design process. Additionally, every wire is replaced by the corresponding number of wires, which connect gadgets in the same way as gates were connected before. This step can be supported by the open-source framework AGEMA, which we extended for our specific use case, as it is publicly available on GitHub.

3.7.1 Place-and-Route

The result of the aforementioned steps is a netlist of the given design made solely by the primitives available in the CGB. As next, every primitive should find a place in the fabric and the PSMs should realize their interconnections. Similar to mapping, place-and-route is automated by open-source tools, for example, nextpnr or VTR in the FABulous ecosystem. However, in order to enhance SCA security with respect to the underlying DRP, we pursue a balanced routing without different routing delays for dual-rail signals. Even though our architecture enables the same routing for dual-rail signals, it is not assured that parallel routes will be taken by the routing algorithm.

An option to achieve parallel routing is to adopt the fat-wire approach [TV04b]. We consider not only two, but also more associated wires of the same logic variable bundled together for routing and call it *bundle-routing*. First, every dual-rail is bundled and then treated as a single wire. Second, if a logic variable is masked, the dual-rail bundles for all shares are merged together (only as abstraction). Third, the *bundle-routing* routes each bundle as a single wire. Lastly, each bundle is resolved and the determined *bundle-routing* is translated to each contained wire to the actual corresponding physical parallel wire. This leads to parallel routes on the fabric for each rail and share according to the same logic variable.

At the physical level, there may still be some differences dependent on the actual

physical implementation of switch matrices and manufacturing differences. Hence, balanced physical routes (which is relevant when fabricating the chip) should support our proposed bundle-routing technique. Note that – as stated – balanced routing in our approach is not a must as our SCA security relies on the provable secure composable gadgets. Hence, balanced routing may further harden higher-order SCA attacks to succeed in practice. Another problem is partial reconfiguration at run-time, which cannot be allowed, since the compossibility notions may be violated during the reconfiguration process. Therefore, we exclude this option to maintain security.

4 Case Study: Implementation of a Secure Fabric with Open-Source Tools

To evaluate our concept in practice, we implemented a secure reconfigurable fabric with the assistance of the FABulous framework. For this, we implemented custom gadgets and adapted the toolchain to follow our approach explained in Section 3. We map several designs on our custom fabric implementation and the original FABulous fabric as a reference to compare the area consumption in the special use case of physical security. Lastly, we emulate our custom FPGA on a real FPGA device, make use of our entire customized toolchain, and conduct an experimental SCA evaluation of the generated secure designs.

4.1 Design Decisions

First, we would like to highlight that we focus on area requirements and low-latency for the secure platform of our case study. Hence, our solution is based on the LMDPL [SBHM20b] masking scheme that, independent of the circuit size, introduces only one additional cycle of latency for every cycle in the original design, ensuring minimal performance impact. The scheme also offers relatively small area overhead, which makes it an efficient choice for hardware implementations. However, similar to other gadget-based masking schemes, LMDPL requires fresh randomness (updated for every evaluation phase), namely 1 bit per non-linear gadget. For the (non-masked) control-secure region, we adopt WDDL [TV04a] gates, as their area requirement is considerably lower compared to other DRP logics. Additionally, we modified both schemes (based on Section 3.6) to include fault propagation and detection mechanisms with minimal extra area cost, allowing to additionally resist against some FI attacks. As proposed in Section 3.5, the integrated PRNG is part of the reconfigurable fabric as a dedicated (non-reconfigurable) module. This reduces the overall fresh randomness generation overhead significantly compared to any solution that is implemented using reconfigurable primitives, such as LUTs. We rely on an unrolled Trivium stream cipher design, as suggested in [CMM+23]. We constructed the PRNG module also in DRP logic, which provides fresh randomness in every evaluation phase for every non-linear gadget and ensures having its outputs fully nullified during the pre-charge phase. Due to the fault propagation property of the gadgets, a faulty dual-rail state of the fresh randomness propagates (starting at every non-linear gadget input) through the gadget, ensuring detection at the primary outputs of the mapped design, similar to data input faults. For the sake of completeness, we provide a comparison between low-latency masking approaches applied on the AES S-Box in Table 1.

4.2 Gadget Design

Based on the principle of secure reconfigurability, explained in Section 3.3, we matched gadget-based masking to the grid-based approach in the FABulous framework, which leads to the deployed CGBs. Each of our-designed CGB types contains one of the gadgets described below. As described in Section 3.2, we divide the platform into control-secure and

Table 1: Comparison of first-order protected AES S-Box implementations using different low-latency masking schemes.

Reference	Scheme	Latency [cycles]	Area	Randomness [bits/cycle]	Fault Propagation
[SBHM20b]	LMDPL	1	baseline	36	✗
[LMW14b]	LMDPL	2	-19%	36	✗
[GIB18]	GLM	2	+94%	416	✗
this work	LMDPL_{SC}	1	+40%	36	✓

full-secure regions. The control-secure region is based on Self-Checking WDDL (WDDL_{SC}) gadgets which operate on dual-rail signals with a fault-detection facility only. The full-secure region contains LMDPL_{SC} gadgets that operate on masked dual-rail signals and offer first-order SCA security and fault detection. This design decision allows us to implement control logic that does not process secret data, e.g., a counter, based on the primary input annotation of the to be mapped RTL design with significantly less overhead. In the following, the exact gadget implementation and a deep analysis of their fault propagation feature are illustrated, while preserving the SCA security of the underlying scheme.

4.2.1 Self-Checking WDDL (WDDL_{SC})

We constructed a custom WDDL_{SC} gadget, which encompasses the non-linear AND as well as the linear XOR operation to enhance the flexibility of the fabric. As discussed in Section 3.3, the desired gadget functionality is eventually programmed with the bitstream. Certain bits in the bitstream determine whether inputs or outputs are inverted by swapping the rails of their dual-rail representation (cf. Figure 3). In the case of our WDDL_{SC}, one additional configured multiplexer per output rail also selects the non-linear or linear function outputs. As the same configuration must apply for both rails, the same configuration bit is used inside the utilized CGB (cf. Figure 3).

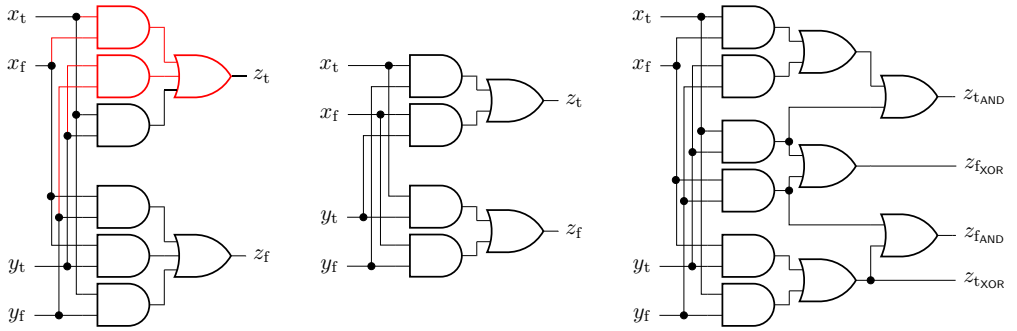
Fault Propagation. The implementation consists of a WDDL_{SC}-AND that is inspired by the DPL-noEE [BGF+10] scheme, which was originally introduced to prevent the *early evaluation* effect. However, the original approach does not propagate all faulty inputs to the gadget outputs.

Lemma 2. *For any variable $x = (x_\text{t}, x_\text{f})$ in a non-faulty DRP circuit, the gate $x_\text{t} \wedge x_\text{f}$ results in 0 and remains unchanged in both pre-charge and evaluation phases. If the function is combined with any other variable y using logical OR gate, the value of y is preserved at the output of the OR gate. In the presence of a fault, where $x = (1, 1)$, the output of the OR gate is effectively tied to 1.*

Following Lemma 2, we added two AND gates in the true rail computation of the WDDL AND gate to cover the cases where any input is faulted to $(1, 1)$ (cf. Figure 5(a)). Each of these gates has both rails of the same variable as their inputs, i.e., for the variable x the added AND gate computes $x_\text{t} \wedge x_\text{f}$ and for the input y the intermediate $y_\text{t} \wedge y_\text{f}$ is computed. Having $x = (1, 1)$ or $y = (1, 1)$ would then assure to have $z = (1, 1)$ at the output. Note that the case $x = y = (1, 1)$ is implicitly covered.

Lemma 3. *Considering any possible combination of a variable among x_t, x_f and another variable among y_t, y_f as two inputs of a logical AND gate, if at least one of the inputs $x = (x_\text{t}, x_\text{f})$ or $y = (y_\text{t}, y_\text{f})$ is $(0, 0)$, the output is assured to be 0.*

As the WDDL_{SC} gadget implements an AND gate 'barrier' that considers every possible combination of input rails, a faulty dual-rail input $(0, 0)$ propagates, leading to the output



(a) WDDL_{SC}-AND. (b) WDDL_{SC}-XOR. (c) WDDL_{SC}-Combined.

Figure 5: WDDL_{SC} gadgets. The original DPL-noEE gates are highlighted in red.

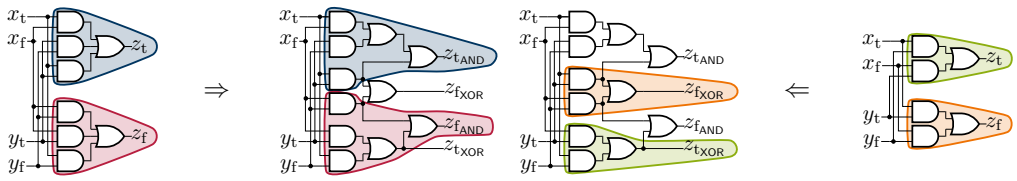
$z = (0, 0)$. The case $x = y = (0, 0)$ is also implicitly covered. The architecture of WDDL_{SC}-XOR is shown in Figure 5(b) which – following the same concept – also assures fault propagation when any of the input is an invalid state $(0, 0)$ or $(1, 1)$. Due to the similar structure of WDDL_{SC}-AND and WDDL_{SC}-XOR, we combine them into the WDDL_{SC}-Combined gadget, illustrated in Figure 5(c), which provides both outputs and can naturally be given to a multiplexer configured to set the gadget's functionality. An analysis of every possible WDDL_{SC}-Combined input fault that is propagated to the outputs, reveals its functional correctness and proper fault propagation for every single faulty input and implicitly, the cases where $x = y$. The results are given in Table 2.

Lemma 4. *Since every WDDL_{SC} gadget itself propagates an invalid input to its output, this fault propagation feature holds true for any arbitrary composition of WDDL_{SC} gadgets.*

In a composition of WDDL_{SC} gadgets, a faulty input encoding of one gadget is either directly faulted or originates from the output of a preceding gadget in the circuit. Consequently, the faulty dual-rail encoding propagates through such a composition. Since WDDL_{SC} gadgets are only representing atomic functions and are not actual atomic gates, it must also be assured that faults at any gate within a gadget are either ineffective or lead to a faulty output. We refer to the general DRP circuit structure, which consists of two redundant parts – one computing the original value and the other its complement – and also applies to the internal circuit of the WDDL_{SC} gadget. Since each gate within one of these parts contributes solely to the single-rail output computed by that part, a single fault can only affect that specific output rail, i.e., a fault within any DRP gadget cannot spread to multiple output rails (cf. Figure 6). As a result, the gadget always produces either a correct or an invalid dual-rail encoding, but never an incorrect yet valid dual-rail output. We verified this via exhaustive logic simulation, testing all possible inputs combined with injecting every possible single fault.

Lemma 5. *A single injected fault within a DRP gadget is either ineffective or leads to a faulty dual-rail encoding at the output.*

It is important to mention that the WDDL_{SC} gadgets are explicitly not designed to be SCA secure. The WDDL_{SC}-Combined gadgets are only utilized in CGBs of the control-secure region of the fabric, i.e., their purpose is to compute values that are not masked. Each gadget must just properly propagate faulty inputs to its outputs and operate in the DRP protocol for compatibility with the rest of the fabric. Since the control-secure region and full-secure region are unidirectionally connected (discussed in detail later in Section 4.3), all modifications support the monotonic behavior in DRP circuits, i.e., we do not introduce any glitches that could propagate to the secure region and violate SCA



(a) WDDL$_{SC}$-AND. (b) WDDL$_{SC}$-Combined. (c) WDDL$_{SC}$-XOR.

Figure 6: Impact of internally injected faults on WDDL$_{SC}$ gadget output rails. Output rail contributing gates are shaded with color in both linear and non-linear gadgets and are transferred to the combined variant.

Table 2: Covered invalid input propagation of WDDL$_{SC}$-Combined.

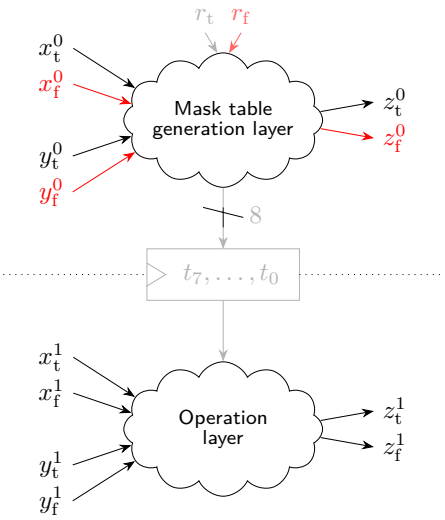
fault type	x_t	x_f	y_t	y_f	$z_{t_{AND}}$	$z_{f_{AND}}$		$z_{t_{XOR}}$	$z_{f_{XOR}}$	
none	0	1	0	1	0	1	✓	0	1	✓
	0	1	1	0	0	1	✓	1	0	✓
	1	0	0	1	0	1	✓	1	0	✓
	1	0	1	0	1	0	✓	0	1	✓
single input to $(0,0)$	0	0	0	1	0	0	✗	0	0	✗
	0	0	1	0	0	0	✗	0	0	✗
	0	1	0	0	0	0	✗	0	0	✗
	1	0	0	0	0	0	✗	0	0	✗
single input to $(1,1)$	0	1	1	1	1	1	✗	1	1	✗
	1	0	1	1	1	1	✗	1	1	✗
	1	1	0	1	1	1	✗	1	1	✗
	1	1	1	0	1	1	✗	1	1	✗
both inputs	0	0	0	0	0	0	✗	0	0	✗
	1	1	1	1	1	1	✗	1	1	✗

security. For the same reason, we stick to the use of AND and OR gates inside the WDDL$_{SC}$ gadgets (instead of NAND and NOR gates) to ensure monotonic behavior for arbitrary gadget compositions.

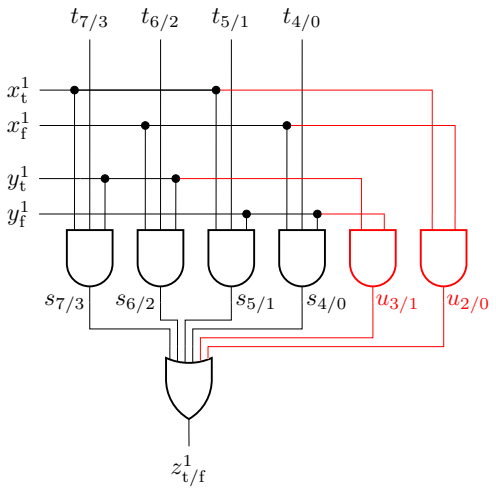
4.2.2 Self-Checking LMDPL (LMDPL$_{SC}$)

The integration of LMDPL gadgets as a basic building block for the full-secure region requires proper fault propagation, while the formal security is preserved. We deploy a linear and a non-linear gadget, while the non-linear one is connected to the global PRNG that provides randomness in its dual-rail representation. An overview of the LMDPL$_{SC}$ which is a modified version of LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL) is shown in Figure 7(a).

Fault Propagation. Signals in the first share domain of the original LMDPL gadget (hence, inputs and output of the mask table generation layer) are not in dual-rail, and thus do not allow fault detection/propagation. Therefore, we replace such signals by their dual-rail counterparts to allow proper fault propagation based on the DRP encoding. This modification also goes hand in hand with our generic wiring approach, described in Section 3.4. Consequently, we apply alternating DRP phases in both layers. Because of the register placed between the layers of the non-linear gadget, the phases are carried out with one clock cycle offset. While intermediate values are evaluated based on x^0 and y^0



(a) LMDPL$_{SC}$ layer overview.



(b) Non-linear LMDPL$_{SC}$ op. layer for true/false rails.

Figure 7: LMDPL$_{SC}$ with modifications compared to the original LMDPL in red.

in the mask table generation layer, the operation layer is pre-charged. The intermediate result is stored in the register and further evaluated with shares x^1 and y^1 in the next clock cycle. During the evaluation phase in the operation layer, simultaneously, the mask table generation layer is pre-charged.

Proper fault propagation in the mask table generation layer of the non-linear variant is implemented by realizing every gate by its corresponding WDDL$_{SC}$ gadget, previously explained in Section 4.2.1. Relying on Lemma 4, fault propagation of the mask table generation layer is assured. More precisely, a fault on x_t^0, x_f^0 or on y_t^0, y_f^0 always propagates to the intermediate values $t_7, \ldots, t_0$, which are also in DRP mode. A fault on the fresh random value (r_t, r_f) additionally propagates to the intermediate values $t_7, \ldots, t_0$ and to (z_t^0, z_f^0).

The operation layer of the original LMDPL gadget propagates one or both invalid inputs $x^1 = (0,0)$ and $y^1 = (0,0)$ by default. Similar to WDDL$_{SC}$, Lemma 3 applies as no combination of input rails enables any AND gate to evaluate to logical '1' (cf. Figure 7(b)), resulting in the output $z^1 = (0,0)$. However, the invalid inputs $x^1 = (1,1)$ and $y^1 = (1,1)$ are not covered. Consider the following gadget inputs as an example, where x^1 is faulty.

$$x^0 = (0,1), \qquad y^0 = (0,1), \qquad x^1 = (1,1), \qquad y^1 = (0,1), \qquad r = (0,1)$$

Since the invalid input is located in the second share domain, the mask table generation layer pre-processes the intermediates based on the first share domain inputs as expected:

$$t_0 = t_1 = t_2 = t_7 = 0, \qquad t_3 = t_4 = t_5 = t_6 = 1.$$

In the mask table generation layer, the intermediate is 'selected' based on x^1 and y^1 values. More specifically, x_f^1 and y_f^1 enable the propagation of t_0 to s_0 and t_4 to s_4:

$$s_0 = s_1 = s_2 = s_3 = s_5 = s_6 = s_7 = 0, \qquad s_4 = 1.$$

Eventually, this leads to a valid dual-rail state at the gadget's output, despite the invalid input x^1:

$$z^1 = (0,1).$$

This example shows that an invalid input given to the LMDPL gadget may not be propagated to any of its outputs, and hence faulty but yet a valid dual-rail output is processed further. To alter this behavior, we extended the operation layer with two additional AND gates, following the same approach as for WDDL_{SC} based on Lemma 2. Our modified design is shown in Figure 7(b). Note that the signals are annotated for both rails allowing to show the circuit of both true and false rails. For example, $u_{3/1}$ denotes the true rail u_3 and the false rail u_1. With our modification, an invalid input $x^1 = (1, 1)$ leads to both $u_2 = 1$ and $u_0 = 1$ while $y^1 = (1, 1)$ leads to $u_3 = 1$ and $u_1 = 1$. Eventually, the occurrence of one or both of these invalid inputs always results in the invalid output $z^1 = (1, 1)$.

Besides the fault propagation of gadget inputs, injected faults inside the non-atomic gadget cause the internal circuit to behave similarly to that of WDDL_{SC} gadgets. In the event of a single injected fault within an LMDPL_{SC} gadget, the output either remains correct or leads to an invalid output encoding. Since the LMDPL_{SC} mask table generation layer is composed of WDDL_{SC} gadgets, Lemma 4 and Lemma 5 apply. As the operation layer is also a DRP circuit, Lemma 5 holds. We verified our claims through exhaustive logic simulation. Building on our fault propagation verification of single internally injected faults within a WDDL_{SC} gadget, we consider WDDL_{SC} gadgets as atomic gates. Our tests covered all possible inputs combined with every possible single fault injected.

Similarly, we follow the same procedure for the linear LMDPL_{SC}-XOR as with the non-linear mask table generation layer and replace every gate with its WDDL_{SC} counterpart. Due to the nature of the linear function, the layers operate independently. Hence, an invalid input in one layer results in an invalid output of that layer.

SCA Security. The authors of [SBHM20b] showed first-order security of LMDPL non-linear gadgets under the SNI composability notion and glitch-extended probing model. In the original scheme, the mask table generation layer only operates on the single-rail representation of the first share of each variable, and the glitch propagation is stopped by the register in between the layers. In the argumentative approach, a contradiction between the glitch-extended probing model and the circuit behavior of the LMDPL operation layer is discussed. Essentially, glitch-extended probes assume leakage by the occurrence of glitches, which is never the case due to the proper execution of the DRP protocol that eliminates glitches entirely. The paper argues that even differences in timings at the second share domain do not reveal sensitive information due to the blinding of the generated intermediates. In the following, we apply and adapt the arguments for our modified variant.

Similar to the original scheme, it still holds that the mask table generation layer processes fresh randomness and the first share of each variable. Under the assumption that share domain separation is kept intact, generation of intermediates is independent of the second share domain and reveals no secret information. In our approach, we guarantee domain separation for arbitrary designs with connectivity constraints (cf. Section 3.4).

In the operation layer, we differentiate between intermediate and output probes to deduce glitch-extended first-order security under the SNI composability notion. Note that we do not cover combined attacks, i.e., SCA attacks on faulty circuits. Hence, considering a fault-free circuit, we focus on the gates/signals which we added to the original LMDPL gadget. If there is no invalid signal, $u_{2/0} = x^1_{\text{t}} \wedge x^1_{\text{f}}$ and $u_{3/1} = y^1_{\text{t}} \wedge y^1_{\text{f}}$ are tied to '0' in both pre-charge and evaluation phases with no toggles or glitches (cf. Figure 7(b)). Therefore, a probe placed on any of these wires does not reveal any information. Moreover, the original security analysis of LMDPL argues that a probe placed on the output z^1_{t} (or z^1_{f}), which observes '1', may reveal information about timing differences. As $u_{2/0}$ and $u_{3/1}$ never toggle in a fault-free circuit, no additional timing information is issued at the output. Thus, the original security argument by the original authors stays valid.

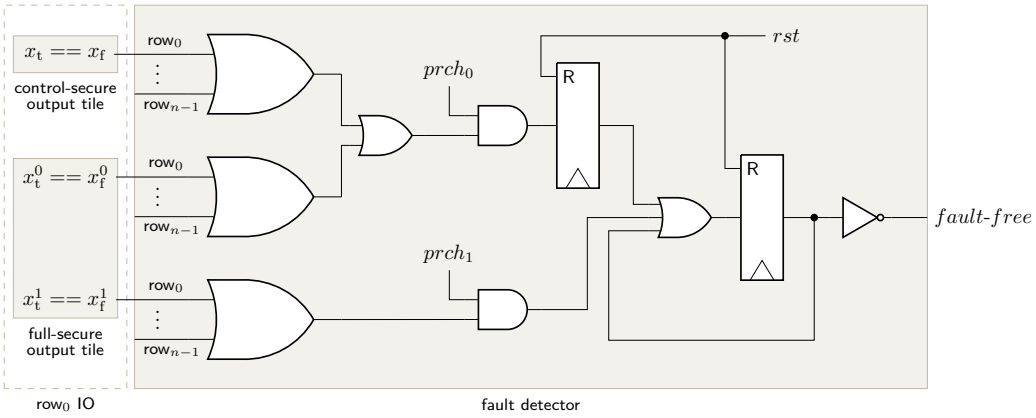


Figure 8: The fault detector takes every XNOR of complementary rail outputs and computes a $fault\text{-}free$ signal. Upon fault detection, the signal toggles to 0 and is maintained in a register until the device is reset. Inside each output tile, the boolean AND function is applied to the output and the $fault\text{-}free$ signal, effectively setting every primary output to logical zero if a fault is detected.

4.2.3 Fault Detector

Building on the fault propagating WDDL$_{SC}$ and LMDPL$_{SC}$ gadgets, it is sufficient to connect a fault detector module to all primary outputs of the secure fabric, including both types of regions: full-secure and control-secure. Since all gadgets integrate self-checking properties for fault propagation, a fault at any gadget input will propagate through the gadget itself and through the consecutive gadgets to at least one primary output, where it can be detected. As shown in Figure 8, the detection circuitry employs an XNOR gate for every dual rail of primary outputs, which identifies an invalid dual-rail state in the evaluation phase. In case one rail is faulty, the XNOR results in logical '1' and is forwarded through an OR-tree that covers the XNOR result of the dual rails of the same share domain. The first share domain also covers the XOR of the control-secure region. Since the share domains are evaluated with one clock cycle offset (mask table generation layer versus operation layer, see Figure 7(a)), the corresponding individual pre-charge signals are utilized to activate the detection only in the respective evaluation phase of the share domain, i.e., $prech_0$ and $prech_1$ signals in Figure 8. By this structure, domain separation is kept intact in the fault detector module to not violate the SCA requirements. As long as no faults are detected, the $fault\text{-}free$ signal is set to 1 and remains stable. If any of the dual-rail primary outputs is in an invalid state, the $fault\text{-}free$ signal toggles to 0 and is maintained in a register until the fabric is reset. Inside each IO tile, including both full-secure and control-secure regions, the $fault\text{-}free$ signal is ANDed with the output signal to effectively avoid issuing any output if a fault is detected until the device is reset.

4.3 Secure Fabric Implementation using FABulous

Since FABulous was not designed for our particular purpose, modifications and extensions to its framework were necessary. The main modifications are the change of basic primitives from LUTs to secure gadgets and configurable wiring adjustments to use only the aforementioned parallel switch matrices. In addition, modules for fault detection, fresh randomness supply, and control signal generation to control DRP phases are integrated. Those modules are automatically scaled to fit the fabric size, including their fixed wiring to the corresponding secure fabric CGBs. After our adjustments, the fabric is easily scalable as in the original FABulous framework. A table specifies the number of rows and columns

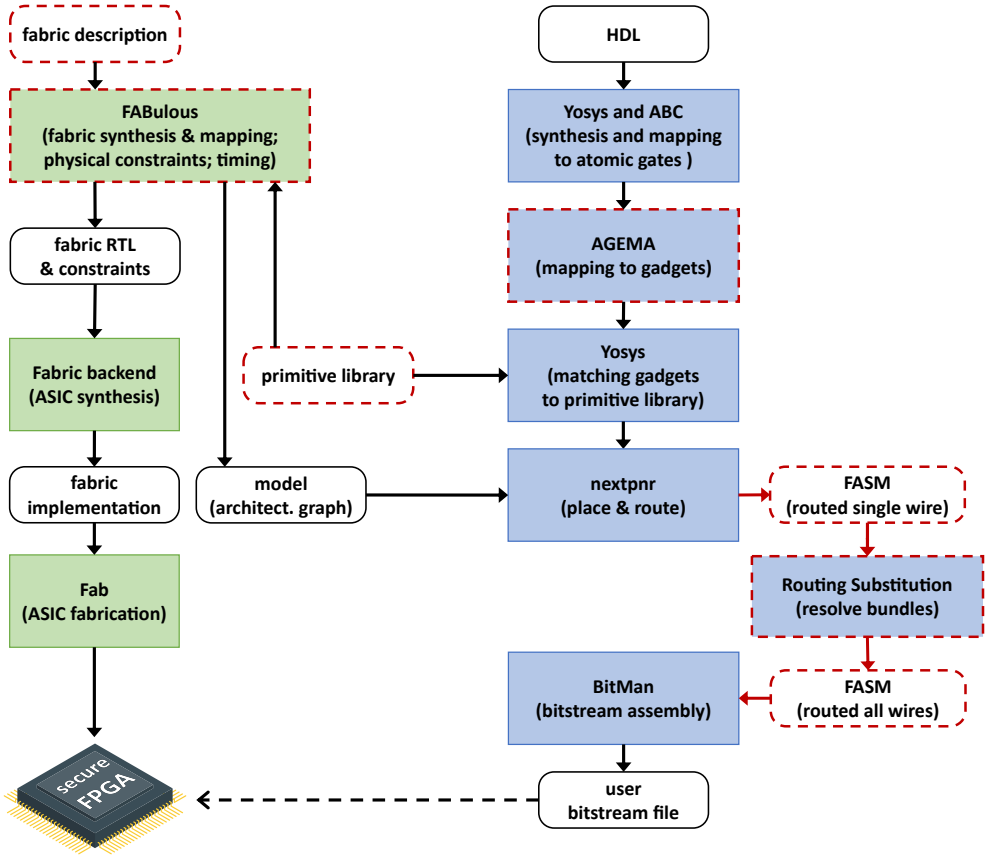
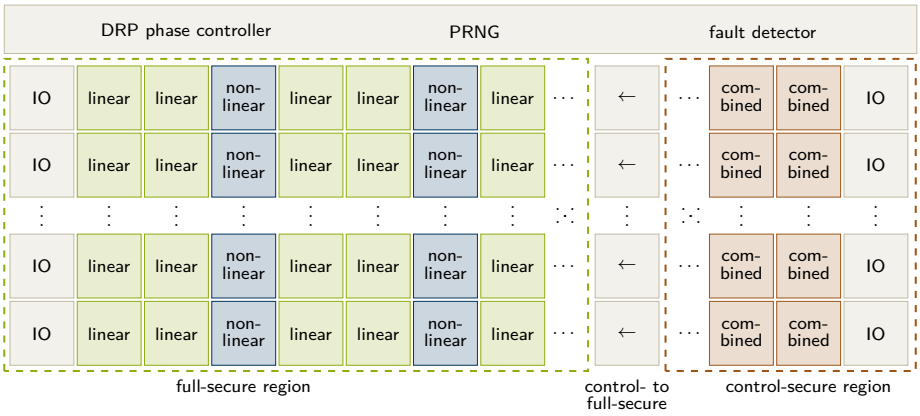


Figure 9: Design flow of a reconfigurable fabric using the FABulous framework. Red highlighted steps show extensions and modified tools.

in the fabric, along with the CGBs types assigned to each cell. In this way, the size of the fabric, the full-secure and control-secure regions as well as the ratio of linear and non-linear CGBs are specified. The general flow for the fabric implementation is illustrated at the left side of Figure 9.

Initially, a description of the fabric is required, which consists of two main parts. The first is a table that defines fabric organization tile by tile, which also contains a detailed description for each tile. This covers utilized primitives, available local wires inside the tile, wires for interconnections to neighbored tiles, and a reference to the switch matrix used. Second, each tile is additionally represented by a set of files. HDL files define the primitives, such as reconfigurable secure gadgets or IO, a switch matrix description, which corresponds to a PSM in our case, and a description of local wiring inside the tile. Therefore, not only primitives may differ from tile to tile, but also switch matrices, allowing for different kinds of routing, such as for dual-rail only (control-secure region) or for masked dual-rail signals (full-secure region).

Since we implement custom primitives and do not implement the basic primitives available in FABulous, the primitive library is also adjusted. All unused primitives are removed and the design abstractions for every deployed gadget type in our CGBs are added. This step is needed to enable the automated flow of mapping HDL designs to the fabric. Using the fabric description and the primitive library, FABulous generates RTL files for the secure fabric, which are then used for synthesis and fabrication. In addition, FABulous generates an architectural graph model, which is needed for place-and-route of



(a) Secure fabric layout.

Region	Type	Gadget	Data	#Wires	#Shares	DRP	Fault Propagation	Provable Side-channel Secure
full-secure	linear	LMDPL_{SC}-XOR	secret	4	2	✓	✓	✓
	non-linear	LMDPL_{SC}-AND						
control-secure	combined	WDDL_{SC}-Combined	public	2	1	✓	✓	✗

(b) CGB type classification overview.

Figure 10: Tile layout in our secure reconfigurable fabric case study. The fabric is partitioned in two security regions with accordingly equipped CGBs, indicating their functionality. Global module extensions are listed at the top.

the designs mapped to the fabric by nextpnr.

LMDPL_{SC}- and WDDL_{SC}-based Secure Fabric. Recalling our defined full-secure and control-secure regions, the non-secure parts of the system are out of the scope, since we assume a secure fabric to be embedded into an SoC. Figure 10 abstracts the structure of our case study implementation. On the left side of the fabric, we placed the full-secure region, which includes full-secure IO ports, full-secure non-linear logic CGBs, and full-secure linear logic CGBs based on LMDPL_{SC} gadgets.

The fresh randomness for non-linear LMDPL_{SC} gadgets is provided by a PRNG every other clock cycle during the evaluation phase of the mask table generation layer. We integrate an unrolled design of the Trivium stream cipher, which is scalable with the fabric size [CMM+23]. Every gate in our Trivium implementation is substituted with its WDDL_{SC} counterpart to also ensure fault propagation during randomness generation. In the scope of this implementation, we expect a seed generated by a TRNG to be loaded into the PRNG only at power-up cycle. The PRNG outputs have a fixed wiring, such that each random bit is uniquely provided to a single non-linear gadget.

On the right side, the control-secure region is placed, which includes control-secure IO ports and WDDL_{SC}-Combined gadget-based CGBs to realize linear and non-linear dual-rail but not masked logic. Between these regions, we place a tile column to interconnect them. The connection is implemented unidirectional, namely from control- to full-secure only. This is done by concatenation of an artificial share tied to '0', i.e., showing x by $(x^0 = x, x^1 = 0)$, more precisely by $(x^0_{\text{t}} = x_{\text{t}}, x^0_{\text{f}} = x_{\text{f}})$ and $(x^1_{\text{t}} = 0, x^1_{\text{f}} = 1)$. Connection in the other direction is not allowed, as it would pose a security breach (see Section 3.4).

To enable fault detection, all IOs perform a check of the valid dual-rail state if configured as output. The resulting signals are hardwired to the fault detector module, which stores a fault detection indicator in a register. A detected fault in the DRP evaluation phase sets a dedicated alarm output signal, and the output ports are simultaneously disabled. The PRNG, the controller of the DRP phases, and the fault detector are adjustable during fabric design and are fixed in the fabricated device, i.e, they are implemented similarly to an ASIC without being reprogrammed after fabrication.

4.4 Open-Source Toolchain for Bitstream Generation

The complete flow to generate the configuration bitstream file for a given HDL design is shown in the right side of Figure 9. The process starts with an unprotected annotated HDL design provided by the user that should be automatically secured with respect to SCA and FI attacks. This design can be any algorithm or function and is mapped to the atomic logic gates by Yosys in the first step.

AGEMA determines the allocation of wires, registers, and gates to their corresponding security region based on primary input annotations by the end-user in the unprotected HDL design, classified as either secret or public. The framework sees the given circuit as a Mealy machine and handles the primary input annotations for secret data by propagating them through the circuit. The remaining parts belong to the control logic that does not depend on secrets and is not masked. In case the control logic operates on secret data, the masking is also applied in that part of the circuit.

Usually, cascaded gadgets of a gadget-based masking scheme contain register stages and lead to extra clock cycles compared to the given original design. LMDPL/LMDPL$_{SC}$ maintains its constant cycle count also in sequential logic, e.g., the state computation of an Finite-State Machine (FSM). This eliminates the need for additional handling or balancing of control signal delays with the rest of the circuit.

Afterward, AGEMA is used to map the atomic logic gates to the deployed fabric primitives, i.e., secure reconfigurable gadgets. To be able to translate the atomic logic gates to the custom gadgets in the proposed fabric, we extended the tool AGEMA to support LMDPL$_{SC}$ and WDDL$_{SC}$ gadgets and generate protected netlists using our primitives only. In the translation process, full-secure non-linear and linear gadgets as well as control-secure gadgets are instantiated. The parameters of the instantiated Verilog modules are set to achieve the desired functionality inherent to the atomic gates they represent. These parameters equal the configuration bits determined by the bitstream that is later generated and programmed onto the device.

The result of AGEMA is a Verilog netlist, which uses parameterized Verilog modules for secure gadgets. At this state, the dual-rail signals as well as masking shares are bundled and represented by a single wire. Any wiring to distribute fresh random bits is also neglected, as randomness is already available to each non-linear masked CGB and does not need to be included in the reconfigurable routing.

In the next steps, we follow the adjusted F4PGA flow with *bundle-routing* without consideration of randomness generation or distribution. Yosys is used again to generate the JavaScript Object Notation (JSON) formatted mapped netlist needed for the nextpnr tool. This step is trivially achieved, since the netlist from the previous synthesis already uses target primitives, and no optimizations are done in this step. Then, the architectural graph model generated by the FABulous framework and the JSON formatted mapped netlist are used for place-and-route with nextpnr. The intermediate result is an FPGA Assembly (FASM) file that describes the placed and routed netlist.

Until this point, the complete workflow treated the dual-rail and masked dual-rail signals as single wires, i.e., bundle-routing. A custom script converts the intermediate netlist to “resolve” the wire bundles by inserting all missing wires directly into the FASM

Table 3: Comparison of area required to map different cryptographic implementations to a secure or conventional fabric with our proposed scheme. The comparison is based on the estimated Gate Equivalent (GE) required to implement the tiles including primitives, switch matrices and configuration memory.

	AES-128 (byte-serial) [DR99]	**AES-128** (round-based) [DR99]	**Midori-64** [BBI+15]	**CRAFT** [BLMR19a]	**Keccak** -f[200] [BDPVA09]
Latency [cycles]	454	22	34	64	1800
Random [bits/cycle]*	389	660	256	192	200

*Numbers stated on average. Double the amount is required during the evaluation phases (every other cycle).

Default FABulous fabric

Cipher [GE]	15453 k	42271 k	13470 k	9364 k	12722 k
#CLBs	2416	6608	2105	1463	1988
#LUT4	19328	52867	16841	11707	15905
#FF	5876	10576	3600	3108	4036
PRNG [GE]	19585 k	32581 k	13201 k	10131 k	10515 k
#CLBs	3062	5094	2064	1584	1644
#LUT4	24492	40752	16512	12672	13152
#FF	1152	1152	1152	1152	1152
Total [GE]	35038 k	74852 k	26671 k	19495 k	23237 k

Ourproposed secure fabric

Cipher [GE]	4648 k	11124 k	3851 k	1963 k	3617 k
difference	*- 70%*	*- 74%*	*- 71%*	*- 79%*	*- 71%*
#linear	998	3088	944	375	1004
#non-linear	778	1320	512	384	400
#control	116	36	143	24	73
PRNG [GE]	158 k	266 k	105 k	79 k	82 k
difference	*- 99%*	*- 99%*	*- 99%*	*- 99%*	*- 99%*
Total [GE]	4806 k	11390 k	3956 k	2042 k	3699 k
difference	*- 86%*	*- 85%*	*- 85%*	*- 89%*	*- 84%*

file. In this way, the parallel routing is achieved with minimal overhead based on share domains and rail indicators.

Eventually, the routed netlist with all wires is used by BitMan to create a user bitstream file in the specified format to program the customized reconfigurable secure fabric.

4.5 Overhead Comparison

Before diving into the conducted overhead comparison, it should be noted that any protection mechanism comes with its own overheads, such as area, energy, and delay. Therefore, a secure reconfigurable fabric is certainly larger and consumes more energy compared to an unprotected reconfigurable platform. However, a fair comparison would be to compare with the same secure design implemented on a conventional fabric.

To demonstrate the efficiency of our proposed method and the gadget-based fabric, we compare it with the LUT-based FABulous reference implementation, synthesized with the same toolchain. This implementation is similar to many commercial FPGA

Table 4: Comparison of ISCAS89 benchmark synthesis results: overhead as Gate Equivalent (GE) between secure and conventional reconfigurable fabrics. The unprotected designs serve as baseline.

	original	**FABulous fabric**			**Our proposed fabric**	
	ASIC	**unprot.**	**WDDL$_{SC}$**	**LMDPL$_{SC}$**	**WDDL$_{SC}$**	**LMDPL$_{SC}$**
s27	19 *-99%*	6396 -	19188 *+200%*	191880 *+2900%*	10048 *+57%*	20834 *+226%*
s382	195 *-99%*	38376 -	198276 *+416%*	2078700 *+5316%*	118064 *+207%*	244286 *+536%*
s420	209 *-99%*	31980 -	185484 *+480%*	1477476 *+4520%*	108016 *+238%*	219660 *+587%*
s641	216 *-99%*	63960 -	236652 *+270%*	2890992 *+4420%*	163280 *+155%*	338908 *+430%*
s713	216 *-99%*	63960 -	243048 *+280%*	2890992 *+4420%*	162024 *+153%*	336442 *+426%*
s1238	722 *-99%*	166296 -	633204 *+281%*	9721920 *+5746%*	536312 *+222%*	1115392 *+571%*
s1423	802 *-99%*	140712 -	761124 *+441%*	7483320 *+5218%*	473512 *+236%*	972658 *+591%*
s1488	776 *-99%*	204672 -	709956 *+247%*	11647116 *+5590%*	630512 *+208%*	1314088 *+542%*
s5378	1847 *-99%*	332592 -	1650168 *+396%*	17736108 *+5233%*	1170592 *+252%*	2406700 *+623%*
s9234	1375 *-99%*	243048 -	1387932 *+471%*	13585104 *+5489%*	1562464 *+543%*	3207376 *+1219%*
s13207	3888 *-99%*	543660 -	3479424 *+540%*	32945796 *+5960%*	2505720 *+361%*	5119856 *+842%*
s15850	4682 *-99%*	812292 -	4816188 *+493%*	45565104 *+5509%*	3158840 *+289%*	6481650 *+698%*
s35932	15936 *-99%*	2315352 -	11256960 *+386%*	98562360 *+4157%*	5935856 *+156%*	12206842 *+427%*
s38417	12888 *-99%*	2296164 -	14512524 *+532%*	135313776 *+5793%*	9771680 *+325%*	19967896 *+769%*
s38584	15230 *-99%*	2417688 -	15369588 *+536%*	163161960 *+6649%*	11677032 *+383%*	24035562 *+894%*
avg.	*-99%*	-	*+398%*	*+5128%*	*+252%*	*+625%*

architectures, since FABulous provides a default template with LUT4 based Configurable Logic Blocks (CLBs), which we use without any modifications. Each tile with such a CLB consists of 8 instances of LUT4, registers, a switch matrix, and configuration memory. Compared to that, in our fabric design one tile consists of a CGB with a configurable gadget, registers, a PSM and configuration memory. For comparison, the same HDL designs are used on both platforms, as follows. In case of the conventional fabric, the HDL is first translated into a protected netlist based on the same security scheme using AGEMA.

Table 5: Area estimate reference for handcrafted first-order secure cryptographic designs synthesized as ASICs and mapped to the FABulous reference fabric.

Implementation	**ASIC** [GE]	**FABulous** [GE]	**RNG** [bit]	**Latency** [cycle]
AES [DCRB+16]	9513	1375140	54	276
AES [GMK16]	9141	1349556	18	246
AES [MPL+11]	15766	2347332	48	266
AES [YCW+24] v1	7437	991380	4	236
AES [YCW+24] v2	12695	1874028	8	148
CRAFT [BLMR19b]	9265	1042548	0	64
Midori-64 [SM21]	10015	1029756	0	32
Midori-64 [MS16]	9881	1432704	0	32

We labeled the primary key and plaintext inputs as secret data and primary control inputs as public, which leads to a mapping of the control logic to the control-secure region, while the logic processing secret data is allocated to the full-secure region. The resulting secure designs are mapped to fabric based on (1) LUT4 or (2) reconfigurable secure gadgets. Estimation of area differences is conducted by the GE required to implement tiles with CLBs or CGBs, respectively, including switch matrices and configuration memory. The results are shown in Table 3.

It should be noted that remaining free resources of a secure fabric can be utilized for some non-secure logic, if necessary. However, due to the overhead of the security measures in the secure fabric, a conventional FPGA would utilize free resources much more efficiently by not implementing any security measures in such a case.

In Table 4 we extend our overhead comparison with non-cryptographic ISCAS89 benchmark circuits. The baseline for each comparison is the unprotected benchmark circuit mapped to the LUT4 CLBs of FABulous fabric. These results are then compared to the original ASIC circuit, and protected versions mapped to the original FABulous fabric and our proposed secure fabric. Since this table is created to show the overheads, we distinguish between two options, where the whole circuit is mapped either to the control-secure gadgets (WDDL_{SC}) or the full-secure gadgets (LMDPL_{SC}). For the LMDPL_{SC} gadgets, we do not consider the overhead required for the PRNG. In the case of the unrolled DRP version of the Trivium cipher (for the PRNG), our proposed secure fabric requires less than 1% of the area needed by the conventional fabric.

To provide a comprehensive evaluation, we include area estimates for handcrafted masked first-order secure implementations of AES, CRAFT and Midori-64 ciphers in Table 5. For fair comparison, we synthesized these designs as ASICs, as well as mapped them to the FABulous reference fabric implementation. For both procedures, we used the same open-source tools, namely Yosys and ABC. Notably, not all handcrafted designs can be directly mapped to an FPGA, such as the FABulous reference fabric and may need some adjustments, e.g., due to the lack of specific registers with asynchronous reset. Due to the nature of reconfigurable fabrics, the overhead compared to ASIC implementations is significantly higher for such mapped designs. Even mapped to a reconfigurable fabric, those handcrafted designs introduce significantly less overhead compared to the proposed gadget-based secure fabric, which can be followed based on the combination of results in Table 3 and Table 5. However, handcrafted implementations are individually designed, while the proposed fabric is capable of protecting arbitrary designs.

4.6 Experimental Evaluation

We verified the correctness of the proposed primitives and architecture by logic simulation with and without fault injection. To illustrate the practical relevance of our approach, we

conducted SCA experiments with a real FPGA device. We emulated a small secure fabric on a real platform and programmed it with a generated bitstream.

4.6.1 Logic Simulation

To evaluate the proposed secure fabric, a series of logic simulations using Icarus Verilog was conducted. Initially, exhaustive simulations on individual gadgets of every type were performed to verify correct functionality and assess the fault propagation property. In this scope, we tested the propagation of faulty inputs and single faults within each gadget for every possible input combination. Subsequently, we simulated a small fabric configured with simple circuits, such as a small MUX tree, which was exhaustively tested for functional correctness. Special emphasis was placed on single-bit fault injection at every possible gadget input. Notably, our results demonstrate that even if a faulty dual-rail state occurs in an inactive circuit path, e.g., at an unselected MUX input, the outputs consistently enter an invalid dual-rail state. Finally, we simulated a fabric of sufficient scale to implement a round-based AES, including the complete fabric configuration process with a given bitstream. This fabric's correct functionality was validated for a variety of ciphers with according bitstream configurations, including LED, Skinny, Midori-64, CRAFT, round-based and serial AES implementations.

4.6.2 Setup

Our platform of choice is a Xilinx Kintex-7 FPGA placed on a SAKURA-X board [SAK16]. We collected power traces by monitoring the voltage drop over a 1 Ω shunt resistor on the Vdd path of the operating FPGA. The platform is clocked by a stable 6 MHz oscillator, while a digital oscilloscope samples the power consumption at a frequency of 500 MS/s. The security analysis relies on the known and common fixed versus random t-tests [SM15].

Due to the high emulation overhead and resulting resource constraints of the physical platform, we deploy a target design that contains a secure fabric with 23×7 full-secure and 23×1 control-secure CGB tiles. This configuration provides a total of 161 full-secure tiles with a 2-to-1 ratio of non-linear and linear gadgets, 23 control-secure tiles, and 23 IO tiles per region. The selected size of the emulated FPGA is sufficient to run an AES S-Box with minimal control logic to execute it in a loop. More precisely, the given input is fed into the AES S-Box, and then its output supplies its input in the further clock cycles.

After the physical device is programmed with the target bitstream, the measurement process is started. In the beginning of that process, the bitstream for the emulated design is serially sent to the target device and serially programmed onto the emulated secure fabric once. In addition, a random seed for the integrated PRNG is provided. For each power trace measurement, the emulated device excluding the PRNG is reset to the default state and restarted. It means that the random seed is only provided and loaded only once right after programming the bitstream. Further, we made sure that the DRP phases of the PRNG are synchronized with the full-secure region.

Utilization. Despite the small emulated fabric size, a 52% LUT utilization of the Kintex-7 is reached by our target design. 99% of which belong to the emulated FPGA. The PRNG takes up a share of about 5.29%, while this ratio should be smaller for larger fabric sizes as the PRNG is also emulated by DRP logic on this commercial FPGA. The resources used by the other global modules, e.g., the fault detector, are comparably negligible. Upon closer inspection of the utilization, the LUT requirements for the emulated switch matrices stand out. As the reconfigurable wiring is realized by multiplexers (FABulous default), the realization with LUTs takes about 85% of the emulated device. For experimental SCA evaluations, we take into account the high limitations that follow from this and program a bitstream that implements a single looped AES S-Box onto the emulated device. In a

manufactured secure FPGA, PSMs would be implemented much more efficiently by logic gates instead of LUTs. However, in our emulation setup, 94% of the full-secure region and 82% of the control-secure region are occupied.

4.6.3 Results

For the security analysis, we supplied the design with either a fixed or random input in a random sequence. In both cases, the input is masked using independent and fresh randomness. After a few idle cycles, as visible in Figure 11(a), the emulated device starts, infinitely the execution of the looped AES S-Box. A first-order fixed vs. random t-test over 100 million traces does not show signs of first-order leakage. The change in the maximum absolute t-value over the number of traces in Figure 11(c) remains clearly below the threshold of 4.5. Figure 11(d) and confirms this for the t-value of every sample point in the power trace after 100 million traces. As a countercheck, we repeated the same experiment without fresh randomness, i.e., we turned the integrated PRNG off. Already after 1 million traces, the t-test shows a significant leakage, plotted in Figure 11(b). Furthermore, we conducted a second-order fixed vs. random t-test as the sanity check to examine the detectability of higher-order leakages. According to our expectations, the experiment shows second-order leakage (cf. Figure 11(e) and Figure 11(f)). The conducted practical experiments reinforce the general approach, theoretical assumptions, and our concrete implementation.

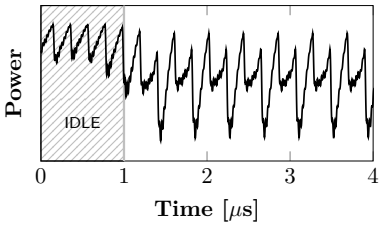
5 Discussions

In this section, we critically discuss our methodology and results. We compare the conventional general-purpose approach with the proposed approach tailored to integrated physical security.

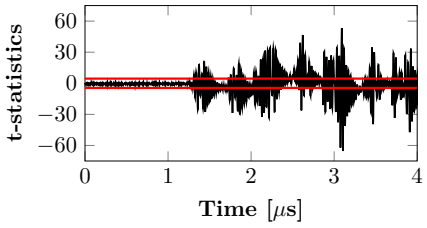
5.1 Secure Fabric Compared to Conventional FPGAs

Our fabric as well as conventional FPGAs both provide the ability to reconfigure the hardware to meet specific requirements of the use case after the manufacturing process. Both of these devices utilize some fixed primitives such as LUTs or secure gadgets, and switch matrices. However, our fabric implements its primitives securely with respect to the selected scheme and the corresponding overhead. Additionally, it implements several submodules such as RNG and fault detector, which are hard-wired. Therefore, it is quite effective in realizing the selected security scheme. However, our approach solely relies on the security assumption of the underlying scheme, which itself remains unchanged after fabrication. A conventional FPGA is more flexible in these terms and is better suited to implement different schemes, but compromises security and efficiency, on the other hand. For example, it would usually require implementing everything with the same reconfigurable primitives, including modules such as an RNG or fault detector. This is not only less efficient compared to ASIC modules but also utilizes non-optimized switch matrices for wiring, which introduces additional overheads.

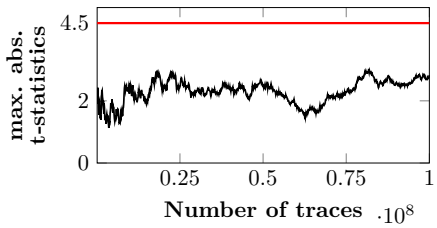
In our case study, the selected gadgets provide protection against first-order SCA attacks and enable fault detection as a countermeasure against some FI attacks. However, it should be highlighted once more that our proposed methodology can be applied to any kind of secure gadgets. For example, higher-order SCA-secure gadgets can be selected to construct CGBs of the secure region. This would naturally lead to higher number of share domains, etc. In the same way, gadgets may implement different approaches for fault detection or other measures against fault attacks like fault correction. It is also possible to implement gadgets with SCA resistance or for fault detection only.



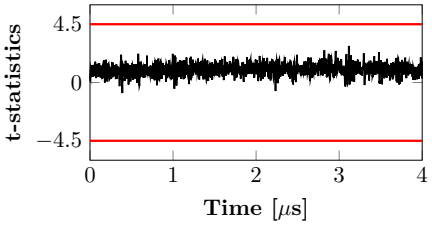
(a) A sample trace.



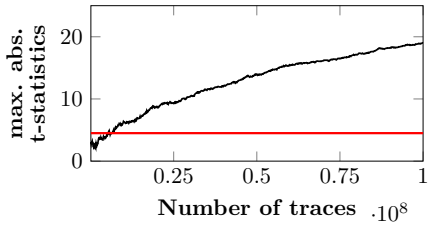
(b) First-order t-test, without fresh randomness, 1 million traces.



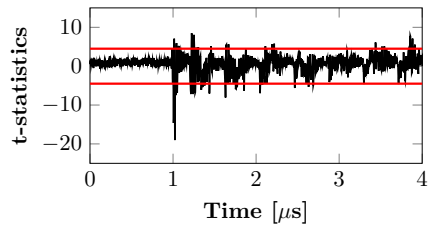
(c) First-order maximum absolute t-statistic, with fresh randomness.



(d) First-order t-test, with fresh randomness, 100 million traces.



(e) Second-order maximum absolute t-statistic, with fresh randomness.



(f) Second-order t-test, with fresh randomness, 100 million traces.

Figure 11: Experimental analysis results, including a sample trace and univariate fixed vs. random t-test results.

Additionally, it should be noted that optimizations can be done to both LUT-based and gadget-based fabric tiles with respect to the target requirements, such as the security schemes employed. However, since our main goal is to show an approach for secure reconfigurable fabric design and to evaluate them from a security perspective, optimizations are considered as lower priority. For example, we use only one secure gadget per CGB, which is sufficient for the evaluation, and makes the analysis process less complex and easier to understand at the same time.

The primary constraint for the proposed approach remains the fabric size, which directly limits the complexity of designs that can be accommodated. According to our synthesis results, we estimate that a round-based AES cipher implementation requires a proposed secure fabric with circuit complexity comparable to a conventional FPGA fabric, such as FABulous, with 15k LUT4. Typically, symmetric crypto implementations (e.g., AES) can be converted to a Boolean masked circuit efficiently and benefit from this approach. Nevertheless, not all algorithms are well-suited for the application of Boolean Masking, e.g., many add–rotate–XOR (ARX) and Post-quantum cryptography (PQC) algorithms, and are at least highly inefficient or even practically unreasonable due to the required fabric sizes.

5.2 Use Cases

In this section, we would like to highlight use case scenarios of our proposed physically secure reconfigurable fabric. In general, it can be used whenever SCA and/or FI attacks pose a risk to the system. Changes and/or security patches can be easily applied via the reconfigurability nature of the fabric. This makes it beneficial for use in new or fast-evolving products, as well as very flexible for integration into the existing systems.

In order to map an HDL design to the secure fabric, no specific knowledge about the details of the secure fabric or the security schemes is required. The tool-chain fully automates this process including mapping, placement, and routing of the target design, according to the implemented security measures. This can greatly reduce the development efforts as well as the risks of human-induced errors when applying security measures.

Our proposed hardware is ideal for systems requiring enhanced security due to the processing of very sensitive information, such as in financial or medical data. It is also crucial for critical infrastructure or safety-critical systems, where any malfunction or interruption could result in significant damage and/or pose substantial risks to human life and property.

5.3 SCA Resistance

LMDPL has already been evaluated multiple times in the literature and its first-order security has been proved both in theory and in practice [SBHM20a, MLM24, LMW14b, MM24]. Estimating higher-order statistics (necessary to conduct higher-order attacks on masked implementations) is susceptible to low SNR, since the required number of SCA traces increases exponentially with the noise level [PRB09]. As a reference, the authors of [SBHM20a] have shown practical (not theoretical) robustness of an LMDPL AES round-based design against second-order attacks. Therefore, we expect a high level of difficulty to successfully conduct higher-order attacks on our fabric due to the realization of every signal of LMDPL_{SC} with a DRP scheme, hence a lower SNR (see the construction of LMDPL_{SC} given in Section 2.3.6). Note that the first share domain of LMDPL gadgets are implemented by single rail logic while this is not the case in LMDPL_{SC} gadgets.

Nevertheless, we should comment on a low number of traces required to detect a second-order leakage in our experiment, i.e., Figure 11(e). This is due to the fact that we used a commercial FPGA to emulate our fabric. It means that several LUTs and switch boxes of the employed Xilinx FPGA are involved in every DRP circuitry which makes them away from being able to reduce the SNR. More precisely, the number of logic elements involved in dual rails are not balanced, resulting in not only imbalanced routing but also potentially imbalanced number of toggles. This is not the case when the fabric is implemented as an ASIC chip, and we expect to see considerably lower amount of second- and higher-order leakages.

5.4 FI Resistance

Preventing fault attacks is very costly, if even possible, and in combination with proper SCA resistance, such as masking, it becomes even more challenging. In our approach, we focus on the self-checking principle and provide a sound detection mechanism for faults. As explained in Section 3.6, fault injection attempts additionally affect some untargeted signals/gates and cause some invalid dual-rail states, which would be sufficient to detect fault injection in our approach. Therefore, our approach can detect faults which may stay undetected by schemes comparing only redundant primary outputs, e.g., majority voting. It should be noted here that faults can be detected in all paths of the entire circuit simultaneously, including those which do not directly affect the primary outputs. This implicitly covers setup time violation attacks such as Fault Sensitivity Analysis (FSA)

[LSG+10] and does not require additional countermeasures as in [SBG+09] due to the propagation of faults to the primary output even if it is occurring in a not activated path. To elaborate on this, it is true that we examine the validity of dual-rail outputs only, but as explained in Section 3.6, any invalid state in the entire circuit would stop the proceeding circuit to generate a valid dual-rail state. Therefore, if a fault is injected and all dual-rail outputs are in a valid state, the fault was not effective or it has been injected after the primary outputs have delivered the circuit output. Note that even in such a case, invalid dual-rail states will be stored in the register, and they reach the primary output in the next clock cycle.

Theoretically, it is possible to inject multiple faults in such a way that the resulting signals are in a valid dual-rail state and bypass our detection facility. However, the rails associated to a logical signal are always placed closely to each other due to our routing constraints, which guarantees that dual-rail pairs are routed together. In addition, the challenge is to cause different faults, one rail from 0 to 1 and the complementary rail from 1 to 0, at the same time, while the targeted wires are placed close to each other. Therefore, we assume that even highly localized fault injections equipments have difficulties to inject undetectable faults in our fabric.

5.5 Other Attack Vectors

In this work, we focused on the security of the designs mapped to the reconfigurable fabric in the first place. However, there are some additional attack vectors for the resulting device as a whole which are out of the scope here, but need to be considered in any final implementation.

A common target of some attacks on any IC is its test circuitry [TM14], i.e., Design-for-Testability (DfT) infrastructure, such as scan-chains or any other kind of Built-In Self-Test (BIST). The test circuitry may reveal secrets directly, if not protected, or may open additional side-channels even if test ports are protected. In addition, DfT may violate security assumptions and weaken the employed security mechanisms.

An attack vector specific to reconfigurable fabric is its bitstream as shown for commercial FPGAs [DRL+19]. In particular, bitstream tampering and insertion of Trojans [MSKGB14] are serious threats. As a countermeasure, it is usually possible to enable encryption and verification of bitstreams loaded onto the FPGA. However, the process of loading an encrypted bitstream can also be attacked. Various SCA attacks show that it is possible to reveal the bitstream encryption key [MBKP11, DMBO+05, TLSB17]. Furthermore, the system components involved in loading, decrypting, and verifying bitstream can be attacked, as shown for First-Stage Boot Loader (FSBL) in [Pet18] or executing some malicious code on AMD/Xilinx 7-series SoCs [EMP20]. In addition, adversaries can try to inject faults into the device configuration [LDS+15, LDF+05, PDF+08] or abuse partial reconfiguration. It might also be possible to execute replay attacks by loading an older version of the bitstream that are not tampered, but with some vulnerabilities that are fixed in the newer versions.

The memory used in the final devices is also a possible attack vector, regardless of its use for configuration or computation. For instance, in [PCMF24] the authors demonstrated a critical device vulnerability over the SDRAM interface of SoC FPGA devices.

5.6 Open-Source

We want to highlight that our entire methodology is realizable with publicly available tools, as shown by our concept and case study. This is very significant, since this topic becomes easily accessible to the entire scientific community, enabling transparency and research on the subject of secure hardware. In addition, this opens opportunities to implement security features and allows fine-grained adjustments to any step of the development

process, such as improving the gadgets with respect to security as stated above, and custom routing algorithms for better balanced wire loads in DRP. Hence, we contribute our implementation by making it publicly available on Github[2].

6 Conclusions

In this work, we presented a novel concept for provable secure reconfigurable fabric design based on secure gadgets realizing first-order Boolean Masking and DRP. Furthermore, we implemented a secure fabric as a case study with the assistance of the open-source framework FABulous, emulated a design on a commercial FPGA and conducted experimental SCA evaluations by means of TVLA using 100 million measurements. In particular, we presented how reconfigurable secure gadgets can be implemented based on WDDL_{SC} and LMDPL_{SC} gadgets and composed to construct a fabric. Based on the underlying security models, our proposed approach offers strong protection against SCA attacks and enables the detection of a wide range of fault injection attempts. The employed countermeasures are abstracted at the gate level, and the entire process of generating a configuration bitstream compatible to the fabric for a given HDL design is fully automated by publicly available tools. This would allow any engineer to apply security countermeasures correctly by primary input annotations in the unprotected HDL design and map it to our fabric. Especially, symmetric crypto implementations that can be converted into Boolean-masked circuits efficiently, greatly benefit from this approach. A configured design can be changed in the field by programming a new bitstream, allowing for updates to programmed (security) algorithms. However, integrated countermeasures at the physical level are fixed at the time of device fabrication and cannot be altered post-production.

Our current approach can be improved from two perspectives. (1) Our case study can detect a reasonably large set of faults, hence should be able to resist DFA attacks. This however does not provide security against some other sophisticated FI attacks like SIFA [DEK+18], where having knowledge about effectivity/ineffectivity of the injected faults is enough to conduct the attack. To cope with this, either fault correction facilities should be considered in the design of gadgets, or some independent sensors for different fault injection techniques should be integrated into the fabric. (2) The gadgets employed in our case study can provide security against first-order SCA and DFA attacks, but not necessary against their combination, i.e., when SCA measurements are collected while faults are also injected. Protection against such combined attacks require a fundamentally new design for the gadgets, where both attack vectors are covered from scratch.

Acknowledgments

The work described in this paper has been supported by the German Research Foundation (DFG) through the project 435264177 (SAUBER).

References

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula R Rao, and Pankaj Rohatgi. The em side—channel (s). In *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*, pages 29–45. Springer, 2003.

[ABM+14] Emna Amouri, Shivam Bhasin, Yves Mathieu, Tarik Graba, Jean-Luc Danger, and Habib Mehrez. Balancing wddl dual-rail logic in a tree-based fpga to

[2] https://github.com/ChairImpSec/SAUBER

enhance physical security. In *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4. IEEE, 2014.

[AK96] Ross Anderson and Markus Kuhn. Tamper resistance-a cautionary note. In *Proceedings of the second Usenix workshop on electronic commerce*, volume 2, pages 1–11, 1996.

[AMM10] Emna Amouri, Zied Marrakchi, and Habib Mehrez. Controlled placement and routing techniques to improve timing balance of wddl designs in mesh-based fpga. In *2010 IEEE Asia Pacific Conference on Circuits and Systems*, pages 296–299. IEEE, 2010.

[AMM13] Emna Amouri, Habib Mehrez, and Zied Marrakchi. Impact of dual placement and routing on wddl netlist security in fpga. *International Journal of Reconfigurable Computing*, 2013(1):802436, 2013.

[BBD+16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong Non-Interference and Type-Directed Higher-Order Masking. In *CCS 2016*, pages 116–129. ACM, 2016.

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A Block Cipher for Low Energy. In *ASIACRYPT*, volume 9453 of *LNCS*, pages 411–436. Springer, 2015.

[BDL97] Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of checking cryptographic protocols for faults. In *International conference on the theory and applications of cryptographic techniques*, pages 37–51. Springer, 1997.

[BDPVA09] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. *Submission to NIST (Round 2)*, 3(30):320–337, 2009.

[BGF+10] Shivam Bhasin, Sylvain Guilley, Florent Flament, Nidhal Selmane, and Jean-Luc Danger. Countering early evaluation: an approach towards robust dual-rail precharge logic. In *WESS*, page 6. ACM, 2010.

[Bit] BitMan GitHub. https://github.com/khoapham/bitman. Accessed: 2024-01-12.

[BLMR19a] Christof Beierle, Gregor Leander, Amir Moradi, and Shahram Rasoolzadeh. CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks. *IACR Trans. Symmetric Cryptol.*, 2019(1):5–45, 2019.

[BLMR19b] Christof Beierle, Gregor Leander, Amir Moradi, and Shahram Rasoolzadeh. Craft: lightweight tweakable block cipher with efficient protection against dfa attacks. *IACR Transactions on Symmetric Cryptology*, 2019(1), 2019.

[BRF+07] Taha Beyrouthy, Alin Razafindraibe, Laurent Fesquet, Marc Renaudin, Sumanta Chaudhuri, Sylvain Guilley, Philippe Hoogvorst, and Jean-Luc Danger. A novel asynchronous e-fpga architecture for security applications. In *2007 International Conference on Field-Programmable Technology*, pages 369–372. IEEE, 2007.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*, pages 513–525. Springer, 1997.

[CGH+08] Sumanta Chaudhuri, Sylvain Guilley, Philippe Hoogvorst, Jean-Luc Danger, Taha Beyrouthy, Alin Razafindraibe, Laurent Fesquet, and Marc Renaudin. Physical design of fpga interconnect to prevent information leakage. In *International Workshop on Applied Reconfigurable Computing*, pages 87–98. Springer, 2008.

[CGLS20] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Transactions on Computers*, 70(10):1677–1690, 2020.

[CGP+12] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 69–81. Springer, 2012.

[CJRR99] Suresh Chari, Charanjit S Jutla, Josyula R Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 398–412. Springer, 1999.

[CMM+23] Gaëtan Cassiers, Loïc Masure, Charles Momin, Thorben Moos, Amir Moradi, and François-Xavier Standaert. Randomness generation for secure hardware masking-unrolled trivium to the rescue. *Cryptology ePrint Archive*, 2023.

[CPM+18] Giovanni Camurati, Sebastian Poeplau, Marius Muench, Tom Hayes, and Aurélien Francillon. Screaming channels: When electromagnetic side channels meet radio transceivers. In *Conference on Computer and Communications Security (CCS)*. ACM, October 2018.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference. *IEEE TIFS*, 15:2542–2555, 2020.

[CZ06] Zhimin Chen and Yujie Zhou. Dual-rail random switching logic: a countermeasure to reduce side channel leakage. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 242–254. Springer, 2006.

[DCBG+17] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? In *Constructive Side-Channel Analysis and Secure Design: 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers 8*, pages 1–18. Springer, 2017.

[DCRB+16] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking aes with shares in hardware. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 194–212. Springer, 2016.

[DEK+18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. Sifa: exploiting ineffective fault inductions on symmetric cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 547–572, 2018.

[DMBO+05] Elke De Mulder, Pieter Buysschaert, SB Ors, Peter Delmotte, Bart Preneel, Guy Vandenbosch, and Ingrid Verbauwhede. Electromagnetic analysis attack on an fpga implementation of an elliptic curve cryptosystem. In *EUROCON 2005-The International Conference on" Computer as a Tool"*, volume 2, pages 1879–1882. IEEE, 2005.

[DMBR19] Lauren De Meyer, Begül Bilgin, and Oscar Reparaz. Consolidating security notions in hardware masking. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 119–147, 2019.

[DPHK17] Khoa Dang Pham, Edson Horta, and Dirk Koch. Bitman: A tool and api for fpga bitstream manipulations. In *Design, Automation, and Test in Europe Conference and Exhibition (DATE), 2017*, pages 894–897, 2017.

[DR99] Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael. 1999.

[DRL+19] Adam Duncan, Fahim Rahman, Andrew Lukefahr, Farimah Farahmandi, and Mark Tehranipoor. Fpga bitstream security: a day in the life. In *2019 IEEE International Test Conference (ITC)*, pages 1–10. IEEE, 2019.

[EMP20] Maik Ender, Amir Moradi, and Christof Paar. The unpatchable silicon: a full break of the bitstream encryption of xilinx 7-series {FPGAs}. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1803–1819, 2020.

[FGP+18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model. *IACR TCHES*, 2018(3):89–120, 2018.

[GIB18] Hannes Groß, Rinat Iusupov, and Roderick Bloem. Generic low-latency masking in hardware. *IACR transactions on cryptographic hardware and embedded systems*, pages 1–21, 2018.

[GKG+23] Mathieu Gross, Jonas Krautter, Dennis Gnad, Michael Gruber, Georg Sigl, and Mehdi Tahoori. Fpganeedle: Precise remote fault attacks from fpga to cpu. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, pages 358–364, 2023.

[GKT19] Dennis R. E. Gnad, Jonas Krautter, and Mehdi B. Tahoori. Leaky noise: New side-channel attack vectors in mixed-signal iot devices. *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, 2019(3):305–339, May 2019.

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. *Cryptology ePrint Archive*, 2016.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*, pages 463–481. Springer, 2003.

[KDH+21] Dirk Koch, Nguyen Dao, Bea Healy, Jing Yu, and Andrew Attwood. Fabulous: An embedded fpga framework. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 45–56, 2021.

[KGT18] Jonas Krautter, Dennis RE Gnad, and Mehdi B Tahoori. Fpgahammer: Remote voltage fault attacks on shared fpgas, suitable for dfa on aes. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 44–68, 2018.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 388–397. Springer, 1999.

[KM22] David Knichel and Amir Moradi. Low-latency hardware private circuits. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1799–1812, 2022.

[KMMS21] David Knichel, Amir Moradi, Nicolai Müller, and Pascal Sasdrich. Automated generation of masked hardware. *Cryptology ePrint Archive*, 2021.

[Koc96] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*, pages 104–113. Springer, 1996.

[KP13] Thomas Korak and Thomas Plos. Applying remote side-channel analysis attacks on a security-enabled nfc tag. In *Cryptographers' Track at the RSA Conference*, pages 207–222. Springer, 2013.

[KSM22] David Knichel, Pascal Sasdrich, and Amir Moradi. Generic hardware private circuits: Towards automated generation of composable secure gadgets. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 323–344, 2022.

[Lam83] Lam. A theory of totally self-checking system design. *IEEE transactions on computers*, 100(9):831–844, 1983.

[LDF+05] Austin Lesea, Saar Drimer, Joseph J Fabula, Carl Carmichael, and Peter Alfke. The rosetta experiment: atmospheric soft error rate testing in differing technology fpgas. *IEEE Transactions on device and materials reliability*, 5(3):317–328, 2005.

[LDS+15] Huiyun Li, Guanghua Du, Cuiping Shao, Liang Dai, Guoqing Xu, and Jinlong Guo. Heavy-ion microbeam fault injection into sram-based fpga implementations of cryptographic circuits. *IEEE Transactions on Nuclear Science*, 62(3):1341–1348, 2015.

[LMW14a] Andrew J. Leiserson, Mark E. Marson, and Megan A. Wachs. Gate-level masking under a path-based leakage metric. In *CHES 2014*, volume 8731 of *LNCS*, pages 580–597. Springer, 2014.

[LMW14b] Andrew J Leiserson, Mark E Marson, and Megan A Wachs. Gate-level masking under a path-based leakage metric. In *Cryptographic Hardware and Embedded Systems–CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings 16*, pages 580–597. Springer, 2014.

[LSG+10] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault sensitivity analysis. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, 2010.

[MBKP11] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. On the vulnerability of fpga bitstream encryption against power analysis attacks: Extracting keys from xilinx virtex-ii fpgas. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 111–124, 2011.

[MGP09] Ali Mokari, Behnam Ghavami, and Hossein Pedram. Scar-fpga: A novel side-channel attack resistant fpga. In *2009 5th Southern Conference on Programmable Logic (SPL)*, pages 177–182. IEEE, 2009.

[MI14a] Amir Moradi and Vincent Immler. Early propagation and imbalanced routing, how to diminish in fpgas. In *Cryptographic Hardware and Embedded Systems–CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings 16*, pages 598–615. Springer, 2014.

[MI14b] Amir Moradi and Vincent Immler. Early propagation and imbalanced routing, how to diminish in fpgas. In *CHES 2014*, volume 8731 of *LNCS*, pages 598–615. Springer, 2014.

[MLM24] Nicolai Müller, Daniel Lammers, and Amir Moradi. A deep analysis of two glitch-free hardware masking schemes SESYM and LMDPL. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(3):76–98, 2024.

[MM24] Nicolai Müller and Amir Moradi. Robust but relaxed probing model. *Cryptology ePrint Archive*, 2024.

[MMG+23] Nicolai Muller, Sergej Meschkov, Dennis RE Gnad, Mehdi B Tahoori, and Amir Moradi. Automated masking of fpga-mapped designs. In *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, pages 79–85. IEEE, 2023.

[MPL+11] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of aes. In *Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30*, pages 69–88. Springer, 2011.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked aes hardware implementations. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 157–171. Springer, 2005.

[MS16] Amir Moradi and Tobias Schneider. Side-channel analysis protection and low-latency in action: –case study of prince and midori–. In *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22*, pages 517–547. Springer, 2016.

[MSKGB14] Sanchita Mal-Sarkar, Aswin Krishna, Anandaroop Ghosh, and Swarup Bhunia. Hardware trojan attacks in fpga devices: threat analysis and effective counter measures. In *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, pages 287–292, 2014.

[MW15] Amir Moradi and Alexander Wild. Assessment of hiding the higher-order leakages in hardware: What are the achievements versus overheads? In *Cryptographic Hardware and Embedded Systems–CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings 17*, pages 453–474. Springer, 2015.

[OD19] Colin O'Flynn and Alex Dewar. On-device power analysis across hardware security domains. *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, pages 126–153, 2019.

[PCMF24] Alexandre Proulx, Jean-Yves Chouinard, Amine Miled, and Paul Fortier. Analyzing the vulnerabilities of external sdram on system-on-chip field programmable gate array devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2024.

[PDF+08] Vincent Pouget, Alexandre Douin, Gilles Foucard, Paul Peronnard, Dean Lewis, Pascal Fouillat, and Raoul Velazco. Dynamic testing of an sram-based fpga by time-resolved laser fault injection. In *2008 14th IEEE International On-Line Testing Symposium*, pages 295–301. IEEE, 2008.

[Pet18] Ed Peterson. Developing tamper-resistant designs with zynq ultrascale+ devices. *Xilinx Application Note*, 2018.

[PM05] Thomas Popp and Stefan Mangard. Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 172–186. Springer, 2005.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Transactions on computers*, 58(6):799–811, 2009.

[RSN+01] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, et al. *A statistical test suite for random and pseudorandom number generators for cryptographic applications*, volume 22. US Department of Commerce, Technology Administration, National Institute of . . . , 2001.

[SAK16] SAKURA. Side-channel Attack User Reference Architecture. http://satoh.cs.uec.ac.jp/SAKURA/index.html, 2016.

[SBG+09] Nidhal Selmane, Shivam Bhasin, Sylvain Guilley, Tarik Graba, and Jean-Luc Danger. Wddl is protected against setup time violation attacks. In *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2009.

[SBHM20a] Pascal Sasdrich, Begül Bilgin, Michael Hutter, and Mark E Marson. Low-latency hardware masking with application to aes. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 300–326, 2020.

[SBHM20b] Pascal Sasdrich, Begül Bilgin, Michael Hutter, and Mark E. Marson. Low-latency hardware masking with application to AES. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):300–326, 2020.

[SG25] Berkeley Logic Synthesis and Verification Group. Abc: A system for sequential synthesis and verification. https://people.eecs.berkeley.edu/~alanmi/abc/, 2025. Accessed: 2025-01-13.

[SGMT18a] Falk Schellenberg, Dennis R. E. Gnad, Amir Moradi, and Mehdi B. Tahoori. Remote inter-chip power analysis side-channel attacks at board-level. In *International Conference on Computer-Aided Design (ICCAD)*, pages 1–7, Nov 2018.

[SGMT18b] Falk Schellenberg, Dennis RE Gnad, Amir Moradi, and Mehdi B Tahoori. An inside job: Remote power analysis attacks on fpgas. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1111–1116. IEEE, 2018.

[SGS23] David Spielmann, Ognjen Glamočanin, and Mirjana Stojilović. Rds: Fpga routing delay sensors for effective remote power analysis attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(2):543–567, 2023.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[SHW+19] David Shah, Eddie Hung, Clifford Wolf, Serge Bazanski, Dan Gisselquist, and Miodrag Milanovic. Yosys+nextpnr: an open source framework from verilog to bitstream for commercial fpgas. *CoRR*, abs/1903.10407, 2019.

[SM15] Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations. In *CHES 2015*, volume 9293 of *LNCS*, pages 495–513. Springer, 2015.

[SM21] Aein Rezaei Shahmirzadi and Amir Moradi. Re-consolidating first-order masking schemes: Nullifying fresh randomness. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 305–342, 2021.

[SMTG23] Kai Schoos, Sergej Meschkov, Mehdi B Tahoori, and Dennis RE Gnad. Jitsca: Jitter-based side-channel analysis in picoscale resolution. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 294–320, 2023.

[SRR+23] Mirjana Stojilović, Kasper Rasmussen, Francesco Regazzoni, Mehdi B. Tahoori, and Russell Tessier. A visionary look at the security of reconfigurable cloud computing. *Proceedings of the IEEE*, 111(12):1548–1571, 2023.

[SS06] Daisuke Suzuki and Minoru Saeki. Security evaluation of dpa countermeasures using dual-rail pre-charge logic style. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 255–269. Springer, 2006.

[TAV02] Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Proceedings of the 28th European solid-state circuits conference*, pages 403–406. IEEE, 2002.

[TLSB17] Shahin Tajik, Heiko Lohrke, Jean-Pierre Seifert, and Christian Boit. On the power of optical contactless probing: Attacking bitstream encryption of fpgas. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1661–1674, 2017.

[TM14] Stephen M Trimberger and Jason J Moore. Fpga security: Motivations, features, and applications. *Proceedings of the IEEE*, 102(8):1248–1265, 2014.

[tru] Layered Security for Your Next SoC. https://www.arm.com/products/silicon-ip-security. Accessed: 2024-01-12.

[tRVP25] Verilog to Routing (VTR) Project. Vpr: Versatile place and route. https://docs.verilogtorouting.org/en/latest/vpr/, 2025. Accessed: 2025-01-13.

[TV04a] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure dpa resistant asic or fpga implementation. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, volume 1, pages 246–251. IEEE, 2004.

[TV04b] Kris Tiri and Ingrid Verbauwhede. Place and route for secure standard cell design. In *Smart Card Research and Advanced Applications VI: IFIP 18th World Computer Congress TC8/WG8. 8 & TC11/WG11. 2 Sixth International Conference on Smart Card Research and Advanced Applications (CARDIS) 22–27 August 2004 Toulouse, France*, pages 143–158. Springer, 2004.

[VDB+24] Dilip Kumar S. V., Siemen Dhooghe, Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. Time sharing - A novel approach to low-latency masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(3):249–272, 2024.

[WGK13] Clifford Wolf, Johann Glaser, and Johannes Kepler. Yosys-a free verilog synthesis suite. 2013.

[YCW+24] Fu Yao, Hua Chen, Yongzhuang Wei, Enes Pasalic, Feng Zhou, and Limin Fan. Optimizing aes threshold implementation under the glitch-extended probing model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[ZH12] Kenneth M Zick and John P Hayes. Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 5(1):1–26, 2012.

[ZS18] Mark Zhao and G. Edward Suh. FPGA-based remote power side-channel attacks. In *Symposium on Security and Privacy (SP)*, pages 805–820. IEEE, May 2018.

[ZZL+22] Zihao Zhan, Zhenkai Zhang, Sisheng Liang, Fan Yao, and Xenofon Koutsoukos. Graphics peeping unit: Exploiting em side-channel information of gpus to eavesdrop on your neighbors. In *Symposium on Security and Privacy (SP)*. IEEE, 2022.