Riding the Waves Towards Generic Single-Cycle Masking in Hardware

Rishub Nagpal[1,2], Barbara Gigerl[2], Robert Primas[2] and Stefan Mangard[1,2]

[1] Lamarr Security Research GmbH, Graz, Austria, firstname.lastname@lamarr.at
[2] Institute for Applied Information Processing and Communication (IAIK), Graz University of Technology, Austria, firstname.lastname@iaik.tugraz.at

Abstract. Research on the design of masked cryptographic hardware circuits in the past has mostly focused on reducing area and randomness requirements. However, many embedded devices like smart cards and IoT nodes also need to meet certain performance criteria, which is why the latency of masked hardware circuits also represents an important metric for many practical applications.
The root cause of latency in masked hardware circuits is the need for additional register stages that synchronize the propagation of shares. Otherwise, glitches would violate the basic assumptions of the used masking scheme. This issue can be addressed to some extent, e.g., by using lightweight cryptographic algorithms with low-degree S-boxes, however, many applications still require the usage of schemes with higher-degree S-boxes like AES. Several recent works have already proposed solutions that help reduce this latency yet they either come with noticeably increased area/randomness requirements, limitations on masking orders, or specific assumptions on the general architecture of the crypto core.
In this work, we introduce a generic and efficient method for designing single-cycle glitch-resistant (higher-order) masked hardware of cryptographic S-boxes. We refer to this technique as (generic) Self-Synchronized Masking ("SESYM"). The main idea of our approach is to replace register stages with a partial dual-rail encoding of masked signals that ensures synchronization within the circuit. More concretely, we show that WDDL gates and Muller C-elements can be used in combination with standard masking schemes to design single-cycle S-box circuits that, especially in case of higher-degree S-boxes, have noticeably lower requirements in terms of area and online randomness. We apply our method to DOM-based S-boxes of Ascon and AES and compare the resulting circuits to existing latency optimized circuits based on TI, GLM, and LMDPL. The latency of all three designs is reduced to single-cycle operation and are d^{th}-order secure. Compared to GLM-masked Ascon, our approach comes with a 6.4 times reduction in online randomness for all protection orders. Compared to 1st-order LMDPL-masked AES, our approach achieves comparable results, while it is more generic, amongst others, by also supporting higher-order designs. We also underline the practical protection of our constructions against power analysis attacks via empirical and formal verification approaches.

Keywords: Masking · Low-Latency · Dual-Rail Logic, · AES, · Ascon

1 Introduction

Cryptographic devices like smart cards are exposed to active or passive implementation attacks that manipulate or observe the physical properties of the device in order to learn sensitive information like cryptographic keys. Among the class of passive attacks, techniques like differential power analysis [KJJ99] or electromagnetic emanation analysis [QS01] are the most critical threats to physically accessible cryptographic devices. One commonly

used algorithmic countermeasure to protect a cryptographic implementation against these kinds of attacks is the usage of masking, a secret-sharing technique that splits a cryptographic computation into multiple shares such that the observation of an incomplete set of shares does not reveal any information about the underlying value. A lot of research over the last years has focused on reducing the overhead of masked hardware circuits either by making the masking itself more area efficient or by reducing the amount of required online randomness [Bil+13; GMK16; Sug19; Dae+20]. However, besides the additional costs in terms of area and randomness, masking also results in a noticeable increase of latency as additional register stages are usually needed to ensure synchronization of shares within a masked hardware circuit. Otherwise, combinatorial glitching of the shares could violate basic assumptions of the masking scheme. This increased latency is especially worrisome for IoT devices that are physically accessible by potential attackers. One way to address the latency problem is to use lightweight cryptographic schemes that utilize low-degree S-boxes, and thus keep the requirement for additional register stages rather low. However, even if such cryptographic schemes are being used, certain latency critical application like external memory encryption will still be noticeably affected. On top of that, many practical applications require the usage of cryptographic schemes like AES that feature high-degree S-boxes. As a result, the search for efficient and low-latency masked S-box circuits has gained increased attention from the research community in the recent years.

Related Work. Low-latency masking was first explored by Moradi et. al in [MS16], where the authors considered asynchronous design methodologies to reduce the latency of first-order threshold implementations. The first generic approach for designing low-latency (higher-order) masked S-box circuits was presented with GLM by Gross et. al in 2018 [GIB18]. The main idea behind GLM is to skip the share compression step after each nonlinear operation which eliminates the need for register stages at the cost of an increased share count, especially in case of higher-degree S-boxes. Later, Sasdrich et. al [Sas+20] applied the LUT-based masked dual-rail logic (LMDPL) technique introduced by Leiserson et. al [LMW14] to low-latency masking. While this technique can considerably outperform GLM in certain scenarios, it can only offer first-order security and comes with concrete requirements on the architecture of the crypto core, such as explicit precharge cycles [Sas+20]. Most recently, Arribas et. al presented a low-latency masking technique based on threshold implementations called LLTI, that has somewhat comparable area requirements to GLM but can eliminate the need of online randomness [AZN21]. Current approaches lead to unfavorable design trade-offs in terms area, latency, randomness requirement and design complexity. In particular, the circuit designer loses flexibility when having to reconcile with non-ideal register placements or delay incurred by explicit precharging cycles.

Comparison with [MS16]. Moradi et. al utilized asynchronous circuits to realize low-latency first-order TI implementations of PRINCE and Midori. They built their designs using WDDL (and extensions AWDDL, DPLnoEE) with precharging/evaluation cycles toggled through a completion detector circuit. Indeed, there is an overlap between their work and our proposal in the underlying concept. To clearly distinguish SESYM from the prior work, we highlight several key differences. Firstly, we generalize the idea of applying asynchronous design to mask circuits of arbitrary protection order. SESYM is generic in the sense that it need not be tied to any specific masking scheme. Furthermore, we highlight that the entire datapath of a circuit does not need to be expressed in dual-rail logic; only select security critical components, such as S-boxes. In their study, Moradi et. al found that their asynchronous 1^{st}-order TI PRINCE implementation was not competitive in terms of latency and area when compared to its synchronous counterpart. However, in Section 4, we show that asynchronous designs are competitive, when compared to other

low-latency masking techniques and d-order security, which were not available at the time of publication of [MS16]. Finally, SESYM overcomes the design difficulties and detectable leakage issues described in the earlier work by using formally verified secure gadgets.

Our Contributions. In this paper, we address the need for practical, flexible and fast masking in hardware by presenting a generic method for designing (higher-order) masked cryptographic S-box circuits that can be computed securely within a single clock cycle. More concretely we provide the following contributions:

- We present Self-Synchronized Masking - a generalization of previous work on low-latency masking with WDDL to achieve single cycle masked hardware applicable to any circuit. We combine WDDL gates and Muller C-elements to achieve synchronization of signals which would ordinarily require dedicated register stages. Our method is easy to apply on top of existing masking schemes to reduce latency, and can be extended to higher masking orders.

- We apply our method to DOM-based S-box designs of Ascon and AES, as well as AES-128. The resulting circuits are then compared to existing latency-optimized circuits based on TI, GLM, and LMDPL. We design single-cycle d^{th}-order masked implementations of the Ascon permutation, and 1^{st} and 2^{nd}-order masked variants of AES. The 1^{st}-order masked Ascon permutation has a similar area consumption compared to GLM, but comes with a 6.4 times reduction in online randomness. Our 1^{st}-order AES S-box implementation is comparable to the 1^{st}-order LMDPL AES S-box in terms of area and randomness, however, our approach is more generic since it also supports other (higher-order) masking schemes while it does not require an explicit precharge cycle in the crypto core. Our 1^{st}-order masked AES-128 is 40% smaller than its LMDPL counterpart with comparable throughput. To the best of our knowledge, our 2^{nd}-order AES-S-box and AES-128 implementations are the first which can be computed securely in single-cycle and remain practical in both area and randomness requirements.

- We underline practical protection of our constructions, both via test vector leakage assessment and formal verification. In case of formal verification, we explain how an existing formal verification approach for masked synchronized circuits can be adapted to the case where register stages are replaced by self-synchronizing logic and perform a 1st/2nd-order verification that considers effects like glitches and transitions.

Outline. In Section 2, we cover the necessary background for this paper. In Section 3, we discuss the implementation of our scheme using the Keccak χ S-box as an example. In Section 4 we present our Ascon and AES masked implementations. Finally, in Section 5 we experimentally and formally verify our scheme holds in practice using test vector leakage assessment methodologies.

2 Preliminaries

In this section, we review the state-of-the-art concepts required to secure a hardware implementation against side-channel analysis. Additionally, we discuss the primitives and techniques from asynchronous circuit design methodologies we use throughout our implementations.

2.1 Boolean Masking

Masking is a commonly used algorithmic countermeasure against implementation attacks like Differential Power Analysis [KJJ99] that splits intermediate values of a cryptographic computation into $d+1$ uniformly random shares, such that the observation of up to d shares does not leak any information about the underlying value. In classical Boolean masking, the sharing of a native variable s, when split into $d+1$ random shares $s_0 \dots s_d$, must satisfy $s = s_0 \oplus \dots \oplus s_d$ where $s_0 \dots s_{d-1}$ are chosen uniformly at random while $s_d = s_0 \oplus \dots \oplus s_{d-1} \oplus s$. This ensures that each share s_i is uniformly distributed and statistically independent of s. When implementing masked cryptographic algorithms, dealing with linear functions is trivial as they can simply be computed on each share individually. However, implementing masking for non-linear functions requires computations on all shares of a native value, which is more challenging to implement in a secure and correct manner, and thus the main interest in literature [ISW03; GMK16; GC17; Bel+17].

The security of a masked circuit is usually analyzed in a theoretical *probing model* that defines what kind of physical properties a passive attacker can observe. Here, the defacto standard model by Ishai et. al, often also called classical probing model, says that a masked circuit is d^{th}-order secure, if an attacker with the ability to place probes on up to d wires or gate of a circuit (to continuously record the signal transitions over time) is not able to combine the recorded information to reveal any native (unshared) values [ISW03]. This also includes unwanted hardware side-effects, like glitches, that may arise due to different signal delays within combinatorial logic blocks or wire lengths, and are frequently shown to be exploitable if not taken into consideration already during the design phase of masked hardware circuits [MPG05; GMK16; GC17; GIB18; Gig+21].

In the past, many different masking schemes have been proposed to simplify the handling of nonlinear operations within cryptographic S-boxes. Even though our presented approach for achieving low-latency masked hardware implementation is not tied to a specific masking scheme, we will primarily build upon the ideas of the domain-oriented masking (DOM) scheme [GMK16].

2.2 Domain-Oriented Masking

Domain-oriented masking (DOM) [GMK16] is a Boolean masking technique which achieves d^{th}-order security by splitting a security-critical circuit into $d+1$ independent shares. Each subcircuit is associated with a single Boolean share of each variable, referred to as a *domain*. Linear operations can be trivially masked with DOM; each operation is simply duplicated across domains. However, for non-linear operations, cross-domain communication is required, which can potentially leak information [GMK16]. We briefly review the non-linear calculation of the finite field multiplication of $a \cdot b$ as a motivating example for the main idea of this work. Suppose a and b are split across $d+1$ shares, the DOM multiplier computes the product of a and b, given in Equation 1 [GMK16]. Figure 1 shows a first-order protected variant ($d = 1$) with respect to the d-probing model. Note, a and b must be independent in order for this construction to be secure.

$$q_i = a_i b_i \oplus \sum_{j>i}^{d} (a_i b_j \oplus r_{(i+j(j-1)/2)}) \oplus \sum_{j<i}^{d} (a_i b_j \oplus r_{j+i(i-1)/2}) \tag{1}$$

The multiplier circuit is illustrated as a three-step process. In the calculation step, the inner-domain products are computed across the shares of a and b. Note that the terms $a_0 \cdot b_0$ and $a_1 \cdot b_1$ do not violate domain-separation rules. In the resharing step, the cross-domain products $a_0 \cdot b_1$ and $a_1 \cdot b_0$ are "masked" by the XOR with the random bit r, followed by a register. Finally, the compression step reduces the $(d+1)^2$ intermediate terms to $d+1$ shares of the output variable, q. The re-sharing register is crucial to prevent

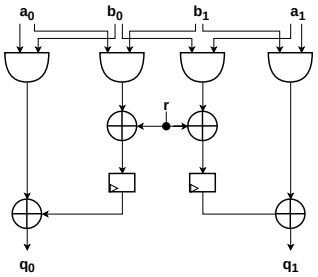


Figure 1: 1st-order protected 1-bit DOM Multiplier

glitches that propagate the unmasked values of a and b to the share compression stage. More complex calculations require more re-sharing registers, having an adverse effect on latency.

2.3 Low-Latency Masking

At ASIACRYPT2016, Moradi et. al investigated asynchronous design methodologies to reduce the latency overhead incurred by masking. Specifically, they compared various asynchronous circuit designs to reduce the cycle latency of first-order threshold implementations (TI) of PRINCE and Midori. They concluded that asynchronous TI designs provided little to no advantage over their synchronous counterparts in terms of both area and latency.

In 2018, Gross et. al introduced a generalized concept to protect latency constrained applications against side-channel analysis by means of Boolean masking [GIB18]. The approach works by skipping the compression step and omitting the re-sharing registers altogether at the price of increased share count. Additionally, the input and linear circuitry is duplicated to avoid variable collisions which would otherwise violate the share independence assumption. This approach for latency reduction however does not come for free as it introduces a significant amount of additional circuitry. For example, a first-order masked ASCON implementation without additional latency requires about 5× the area of an unprotected variant while second-order protection increases the area by 10× [GIB18]. Besides that, the requirement of online randomness is also noticeably increased. For higher degree S-boxes such as AES, the area and online randomness requirements become impractical.

In 2020, Sasdrich et. al introduced generalized low-latency masking using LMDPL gates [Sas+20] and presented a masked AES implementation which computes a full round in one cycle. While the obtained results make this implementation significantly more practical in terms of both area and randomness requirement than the one presented by [GIB18], the LMDPL primitives are somewhat complicated to implement and the LMDPL gadgets are only proven to be first-order secure. Moreover, the entire circuit must be implemented in dual-rail logic and an explicit precharging cycle is necessary prior to evaluation.

Most recently, Arribas et. al presented a low-latency masking technique based on threshold implementations called LLTI [AZN21] and implemented their technique on PRINCE and AES. Compared to the implementation by [Sas+20] the area is worse, however, their construction does not require any online randomness.

2.4 Asynchronous Building Blocks

Following the seminal work on differential power analysis by [KJJ99], various countermeasures were proposed which hoped to mitigate this new class of attacks. For example,

power-hiding countermeasures based on dual-rail with precharging logic (DPL), a central technique from both asynchronous and high-speed circuit design, were developed. DPL uses two wires to transmit a bit: one wire for the bit itself ("true" wire) and the second for the complimentary bit ("false" wire). For a given bit, only one of the two wires is logic 1 and it follows that the same number of bit transitions is exhibited by the circuit for any input. Thus, the general idea behind power-hiding countermeasures is to ensure a circuit consumes the same power for any arbitrary input.

Practically, these countermeasures were difficult to implement correctly and relied on strict physical and timing assumptions. A dual-rail masked circuit must be properly "balanced" to have adequate security. Namely, unmatched wire delays or large physical separation between two rails could cause information leakage [ISU18]. Moreover, the physical circuit area and power consumption increased dramatically due to the duplication of logic cells and the return-to-zero (RTZ) requirement of dual-rail logic. Dual-rail countermeasures eventually fell out of favor for more generic techniques based on Boolean masking such as threshold implementations (TI) and $d + 1$ masking.

Although dual-rail logic was found to be an unsuitable countermeasure on its own, it can be effective when combined with other masking techniques as shown by [Sas+20]. In this work, we also revisit dual-rail logic and apply it in conjunction with DOM. More concretely, we take advantage of the RTZ protocol intrinsic to DPL (referred to as the "handshaking" property) to synchronize data without need for a clock signal.

Dual-rail Encoding. Asynchronous circuit designs employ dual-rail encoding extensively for its handshaking property. The two rails encode a four symbol alphabet which can implicitly indicate to a receiver whether the data on the rails is valid or not via a return-to-zero protocol. In 4-phase dual-rail encoding, the symbol alphabet is divided into two sets: NULL, which is analogous to a "space", and DATA, which contain the data words "0" or "1". The sender transmits a NULL spacer between each DATA word. The NULL spacers allow the receiver to correctly absorb individual DATA words, especially in the case of consecutive DATA0/DATA1 transmission. This implies timing for the receiver without the need for a global clock signal. A simple state diagram for 4-phase dual-rail encoding is given in Figure 2. In asynchronous circuit design, this powerful property enables the construction of (quasi-)delay insensitive circuits: clock-less circuits which make (almost) no assumptions on wire or gate delays.

The RTZ protocol also implies that dual-rail encoding can be used to create monotonic logic gates. The general idea is, when starting from the NULL state, the computation of a DATA0 or DATA1 causes the same number of bit transitions i.e., the power consumed for any input is the same. To ensure all dual-rail buses and gates start from the NULL state, the circuit must be "precharged". During precharging, all dual-rail wires are forced to the NULL state (zeroed). After precharging, the circuit is ready to compute and enters the "evaluation" phase. During evaluation, the circuit is refreshed with the next DATA state on the input wires. Precharging and evaluation typically requires two clock cycles - one for each phase. Some proposed dual-rail logic styles, like WDDL, precharge during the first half of the clock period (clock = 1) and evaluate during the second half (clock = 0).

We emphasize that our proposed masking scheme primarily relies on the handshaking property of dual-rail encoding, and focus on guaranteeing handshakes without the need for a dedicated precharging cycle.

WDDL Logic. "Wave Dynamic Differential Logic" (WDDL) [TV04] is a DPL style initially developed for creating constant power circuits. WDDL follows a 4-phase dual-rail encoding where $\{(0,1),(1,0)\}$ are valid codewords representing logic "0" and logic "1" respectively, and $\{(0,0)\}$ is the NULL spacer (c.f. Figure 2). The $\{(1,1)\}$ codeword is

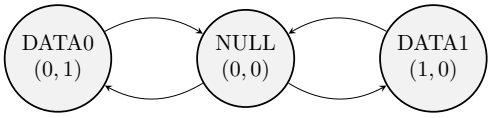
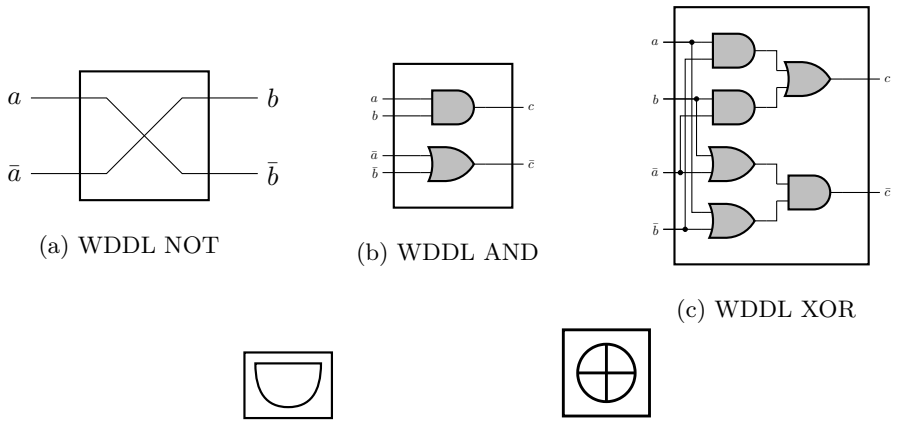


Figure 2: 4-Phase Dual-Rail Code Words



(a) WDDL NOT (b) WDDL AND (c) WDDL XOR

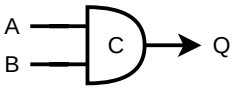
(d) WDDL AND symbol (e) WDDL XOR symbol

Figure 3: Implementation of WDDL logic gates and symbology

considered invalid in WDDL. WDDL logic gates are purely combinatorial and simple to implement with standard cells. Precharging a WDDL gate is equally simple: when all inputs to a WDDL gate are NULL, then the output becomes NULL. For compound WDDL circuits, the NULL spacers are applied to all inputs which propagates as a "precharge wave" across the circuit. Following the precharge wave, DATA on the inputs can be evaluated. Most importantly, when following correct precharge/evaluate procedure, WDDL gates are positive and monotonic. This implies that circuits composed of WDDL gates do not glitch: Consider both the AND gate and the OR gate. By definition, neither gate inverts their inputs i.e., a $0 \rightarrow 1$ (or $1 \rightarrow 0$) transition on the inputs of an AND/OR gate yields a $0 \rightarrow 1$ ($1 \rightarrow 0$) on the output. For different input arrival times, the output either experiences no change, or a transition event in the same direction as the input event. These properties extend for compound AND/OR circuits of arbitrary depth. If the inputs to such a circuit is stable and free of glitches, such as at the output of a register, then no glitch can occur at any point within the circuit. In the DPL setting, the logic transitions are strictly controlled; during precharge, only $1 \rightarrow 0$ transitions can occur. Similarly $0 \rightarrow 1$ transitions only occur during evaluation. We refer the reader to the work by Tiri et. al for a more concrete proof [TV04].

To conclude, Figure 3 shows the standard-cell implementation of WDDL Logic gates we used in this work. For FPGA platforms, careful consideration must be made due to the utilization of lookup tables (LUTs) instead of typical CMOS building blocks to implement logic functions. On Xilinx platforms, WDDL gates can be implemented with a single LUT6_2 primitive [MI14]. Our LUT implementations are given in Appendix A.

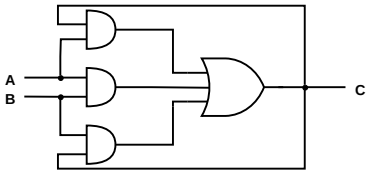
Muller C-Elements. In synchronous circuit design, the rising edge of the clock signal indicates that all signals have reached a valid state. This is guaranteed by setting the clock period to the worst case delay between two registers. During the clock period, signals exhibit transient behavior due to glitching in combinatorial logic. In clock-less circuits,



(a) Symbol

A	B	Q
0	0	0
0	1	-
1	0	-
1	1	1

(b) Truth Table



(c) Standard-cell implementation

Figure 4: Muller C-element

it is impossible to distinguish meaningful signal transitions from transient behavior. To solve this, Muller proposed the "C-Element" [Mul56]: a logic gate which can indicate whether a given set of signals are synchronized. Muller C-elements are state-holding gates akin to SR latches. For a given C-element with n inputs, the output is logic 0 when all inputs are logic 0. Similarly, the output is logic 1 when all inputs are logic 1. For all other input combinations, the output retains its previous value. The gate level implementation of C-Elements have been extensively studied by Moreira et. al in [Mor+12; MC13]. The symbol, truth table and a standard-cell implementation of the fundamental 2-input C-element are given in Figure 4. We note that the implementation of the C-element is purely combinatorial and easy to implement with standard CMOS cells. Moreover, some cell libraries with C-element standard cells are readily available, such as the NCL library by [FB97] and the ASCEnD-FreePDK45 library by [Oli+16].

The C-element solves the fundamental issue of synchronization in a clock-less setting. In asynchronous circuit design, it is most often used to implement valid/ready/acknowledge protocols in hardware (called "micropipelines"). In a dual-rail setting, the C-element can be combined with OR gates to check if every wire-pair has reached the same state.

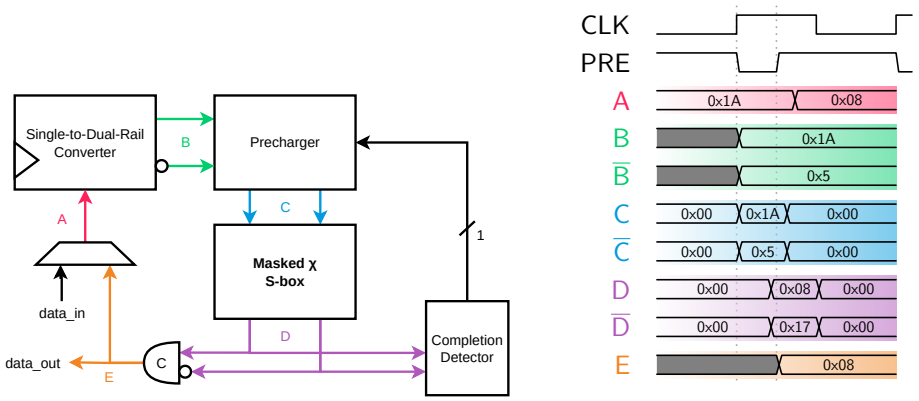
As we discuss in more detail in the next section, we utilize the C-element with an inverted input to convert dual-rail logic to single-rail, and take advantage of the state holding property to preserve the DATA state during precharging.

3 Self-Synchronized Masked Circuits

In this section, we present our approach of combining concepts for signal synchronization in typical asynchronous circuits with (synchronous) masked hardware circuits that significantly reduces the latency of the resulting (synchronous) circuit. We refer to this technique as Self-Synchronized Masking, or SESYM. More concretely, we show how a partial dual-rail encoding of masked signals within masked non-linear components (S-boxes) can be used to eliminate the need of additional register stages as a synchronization measure. Indeed, our approach enables arbitrary masked circuits to compute in a single clock cycle. In contrast to previous works that study the ordinary usage of dual-rail logic alone as a countermeasure to power analysis attacks, our resulting circuits do not have special requirements on the physical layout. To allow for a more apples to apples comparison with prior work, we will apply our approach to hardware circuits using domain-oriented masking (DOM) [GIB18] as a basis, but reiterate that these techniques can be applied to any masking scheme.

3.1 General Concept

As described in Section 2, combinatorial glitching can leak information on secret data. Glitching itself is caused by data races from non-ideal wire and gate delays. Without making any assumptions on physical layout or wire/gate timing, preventing glitches is very difficult. In generic $d + 1$ masking, such as DOM, a register stage must be placed



(a) Circuit Datapath. All signals are 5-bits unless marked otherwise

(b) Timing diagram for the input 0x1A

Figure 5: SESYM-masked χ S-box.

prior to share compression to prevent glitches from potentially leaking the combination of secret shares. The register stage increases the cycle latency of the circuit and its placement in the middle of the non-linear gadget leads a designer to devise awkward pipelining to satisfy security requirements, rather than to maximize performance with equal-length pipeline stages. This problem is exacerbated for higher-degree S-boxes with deeper non-linear logic depth such as the one used in AES. Indeed, in [GIB18], the authors identified the requirement of registers in the resharing and compression steps as a primary cause for latency in DOM which led to them to propose Generic Low-latency Masking (GLM). In GLM, the resharing register and share compression steps are skipped altogether. Although GLM allows for the same cycle latency for an equivalent unmasked circuit, it does not scale well. The number of shares after a non-linear calculation increases by $(d+1)$ times. Furthermore, the online randomness requirement is increased significantly to securely compress $(d+1)^n$ shares back to $(d+1)$ for the next cipher round. For higher protection orders or higher-degree S-boxes, GLM is completely impractical.

The general idea stems from the observation that the only purpose of registers within a masked non-linear circuit is to prevent glitches from propagating to the share compression stage. We can safely remove these registers if we can guarantee glitch-free share compression through other means. This then leads to one complete combinatorial circuit and frees the designer to focus pipelining efforts on optimizing performance.

In the next section, we provide an overview of our implementation goals and present a SESYM-masked variant of the Keccak χ-S-box [Ber+13] as a motivating example. We further discuss implementation details of each SESYM component in a top-down manner.

3.2 Implementation

Figure 5a depicts the full SESYM-masked Keccak χ S-box configured to compute one S-box per cycle. The goal of this circuit is to ensure a continuous, glitch-free handshake from the generation of the dual-rail signals to the end of the S-box computation. We achieve this by adding a single-to-dual-rail converter, a precharger circuit, a completion detector which generates feedback and an array of C-elements as state-holding elements. The state register is configured to act as a single-to-dual-rail converter. The precharger is a combinatorial circuit that passes through data from the converter registers when the precharge control signal is low, otherwise it generates the NULL codeword (zeroes both outputs). The χ S-box is composed of SESYM secure circuits ("gadgets") which

themselves are composed of WDDL logic gates. On the output of the S-box, a completion detector checks whether the S-box has finished computing by testing the dual-rail encoded state of the output wires. If the state is DATA, then the precharge signal is generated. At the same time, the output is forwarded to the state-holding C-elements. The inverted input converts the dual-rail bus back to a single wire. Recall that the C-element's output only switches when both inputs are equal. When the precharge wave propagates to the output, the C-element will preserve the previous value of the wires because it will observe unequal inputs. The C-elements provide the final output which can be forward to outside circuitry or be used in the next calculation. The precharger ensures that the entire circuit is reset to the starting state for the next clock cycle. The combination of the precharger and completion detector circuits enable our design to precharge in a completely self-timed manner i.e., without a clock signal. Figure 5b illustrates an example input to the circuit. Suppose the input 0x1A arrives at the input of the Single-to-Dual-Rail converter (A). At the rising edge of the clock, the data is latched and converted to dual-rail (B). At the same time, the PRE signal is driven low to indicate that the circuit is in the evaluation phase. The data on the dual-rail bus passes through the precharger and into the S-box circuit (C). The output of the S-box is held at 0x00 (NULL) until the computation is completed and transitions to 0x08/0x17 (D). The C-element latches the output of the S-box and converts it back to single rail and forwards the result to the output (E). At the same time, the completion detector drivers the PRE signal high to precharge the S-box circuit. Note, the PRE signal is driven from a flip-flop which is asynchronously set by the completion detector to ensure only a single precharge/evaluation cycle occurs in one clock period. In the next paragraphs, we describe the implementation of each component in detail.

Single-to-Dual-rail Conversion. The single-to-dual-rail component acts as the starting point for our handshake and must be designed with care. To generate the dual-rail bus from a single bit, an inverter must be placed at the origin. However, the delay of the inverter causes a data race between the two wires. Moreover, a high to low transition on the converter's input generates a momentary $(1, 1)$ hazard and invalidates the monotonic property of WDDL.

To prevent this, both rails must have a register immediately following the inverter to re-synchronize the wires. For a cryptographic circuit, the placement of this register can replace the cipher state registers (or any other registers within the datapath). To further reduce overhead, most standard-cell libraries have a register with both the normal and inverted outputs. In principle, it is possible to implement a register-less converter. However, the true and false rails must be delay-matched to compensate for the latency of the false rail inverter. Such a design is only recommended in special cases where timing can be tightly controlled. In this work, we focus on the generic method by implementing the converters with registers.

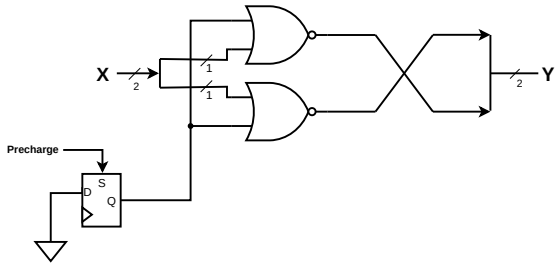


Figure 6: Single cycle precharging/evaluation logic. X and Y are dual-rail buses, where one bit is the "true" wire and the other is the "false" wire.

Single-cycle Precharging and Evaluation. As stated in Subsection 2.4, the handshaking property is crucial for the operation of WDDL and security of SESYM gadgets. This property is only guaranteed when the circuit is precharged. Typically, the precharging and evaluation phases are controlled by an oscillating control signal. In the simplest case this signal is the clock, where the gates are precharged when the clock = 1 and compute when the clock = 0. However, this has a number of disadvantages. Effectively, the clock period for the evaluation of the entire circuit is halved and the forward circuitry is idle during the precharging phase. This can have an adverse effect on the critical path and makes the circuit more susceptible to setup and hold time violations. The clock period could be increased to compensate at the cost of throughput, which is obviously not ideal.

To avoid these issues and to completely decouple from the clock, some additional circuitry can be added to dynamically generate the precharge control signal and still ensure single cycle operation. To achieve this, we place a precharger circuit at the output of the single-to-dual-rail converter. The precharger is composed of a pair of NOR gates and a flip-flop with an asynchronous set (Figure 6). The output of the precharger is connected to the dual-rail input of the χ S-box. At the rising edge of the clock, the flip-flop outputs a logic 0 and the NOR gates propagate x. The circuit then evaluates and generates a "precharge" signal once it has completed. The precharge signal triggers the asynchronous set of the flip-flop, which causes the NOR gates to generate the NULL codeword. The NULL codeword propagates on all inputs and precharges the WDDL gates, until the next clock cycle when the flip-flop clears the stored logic 1.

Completion Detector and C-elements. The precharge signal which triggers the flip-flop is generated from a completion detector circuit at the output of the circuit. The completion detector is implemented as a product-of-sums circuit which tests if every dual-rail bus at the output holds DATA. If all buses hold DATA, the precharge signal is generated and is held high until the NULL wave arrives. In order to preserve the results of the computation, Muller C-elements with one inverted input are placed at the output. Recall that C-elements are state-holding gates. In our case, this means that the C-elements will retain the DATA value of the dual-rail logic, even after all gates are precharged. Conveniently, the C-elements also act as synchronized dual-to-single-rail converters. The C-elements additionally allow us to limit the dual-rail circuitry to only the critical non-linear computations and save on area overhead. One caveat is that the C-element must react quickly enough before the NULL wave reaches its input. For the standard-cell implementation, the logical depth of the C-element is much less than the depth of the SESYM-masked AND gate, so this issue is avoided.

Building Secure Circuits with SESYM. We now shift focus to the implementation of the 1^{st}-order masked χ S-box. In general, secure circuits can easily be constructed with SESYM in a systematic manner. Starting from the AND-XOR representation of the S-box, the first step is to apply domain-oriented masking: create domains by sharing all input variables and duplicating all linear operations to each domain. Next, all AND gates are replaced with the DOM equivalent gadget, in this case the 1-bit DOM multiplier. The naïve implementation of the d-domain DOM AND gate requires $\frac{d(d+1)}{2}$ bits of online randomness. From the DOM circuit, SESYM-masking is as simple as replacing all XOR and DOM AND gates with the respective SESYM-gadgets.

SESYM Gadgets. SESYM linear and non-linear gadgets are derived from WDDL logic gates (Figure 3). For the linear gadget, the WDDL XOR gate (Figure 3c) is sufficient because it is glitch-free and can be applied to compute linear operations independently across shares.

To construct the non-linear gadget, we applied WDDL logic to the DOM multiplier to construct the SESYM variant. Figure 7 shows the 1-bit SESYM multiplier (SESYM AND gate). Structurally, this gate is similar to its DOM with the re-sharing registers removed. It is safe to remove this register because there is no glitch propagation within the entire construction. Furthermore, this property ensures there is no risk of two unmasked shared variables combining during the compression phase of the multiplier. The masked n-bit multiplier is constructed in the exact same fashion as the corresponding DOM multiplier, with the omission of the re-sharing registers. Most importantly, SESYM gadgets propagate the continuous handshake without disruption.

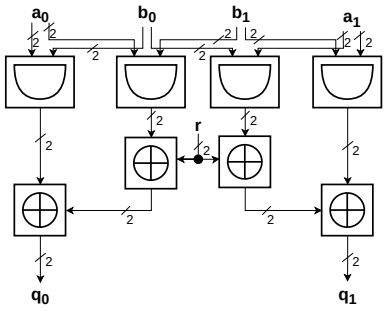


Figure 7: 1^{st}-order SESYM-masked AND gate.

Compositional Security of SESYM Gadgets. Linear operations on shares can introduce dependencies which could cause leakage when a non-linear operation follows. To fix this issue, the dependent shares must be reshared with fresh random bits. As SESYM is generic in the sense that it can be applied on top of any underlying masking scheme, the techniques used by the masking scheme to solve this issue can be translated directly to a SESYM analogue. In the case of DOM, a dependent share is XOR'd with a random bit and registered. This would translate to a WDDL XOR under SESYM. Note, the register is not required if this WDDL XOR gate is precharged correctly. This particular case is discussed and verified in Subsection 4.3 and Section 5

Implementation Caveats. We conclude this section with important points to consider when implementing SESYM-masked circuits. The continuous handshake must be ensured through all components between the single-to-dual-rail converter and the C-elements. If the state registers are selected to act as the converters, every operation up to and including the S-box must be converted to dual-rail and use WDDL gates. However, after the C-elements the handshake is completed and the remainder of the circuit can be (masked) single-rail. We show this in the next section with our SESYM-masked Ascon implementation.

4 Masked Implementations with SESYM

In this section, we discuss the SESYM-masked implementation of the Ascon and AES S-boxes. These ciphers were chosen in particular to demonstrate how SESYM scales for increasing algebraic degree. We provide insights to the design caveats and draw comparisons of each implementation to the state-of-the-art. Each of the implementations were synthesized with Cadence Genus 19.11 for a UMC 65-nm Low-K process with 1.2V supply. Our area results are reported in Gate Equivalents (GE) normalized for the 2-input NAND with size of 1.44 μm^2

4.1 SESYM Masked Ascon Permutation

The Ascon permutation is a lightweight SPN-based round transformation used in the Ascon cipher [Dob+21]. It operates on a 320-bit state that is further divided into five 64-bit lanes. Each permutation round applies three operations: round constant addition, p_C, a 5-bit non-linear substitution across each column of the state, p_S, and a linear permutation of each lane, p_L. Figure 8 shows our implementation of the full permutation. The state registers are built from standard-cell flip-flops which include the inverted output for single-to-dual-rail conversion. Additionally, precharger circuits are required for each S-box and are triggered individually from the completion detector. The round constant addition operation is converted to dual-rail to propagate the continuous handshake that must occur from the dual-rail source to the last non-linear gadget. One possible optimization is to place the state registers between the round constant addition and the S-box at the cost of slightly increased design complexity. We opted not to do this for clarity of illustration.

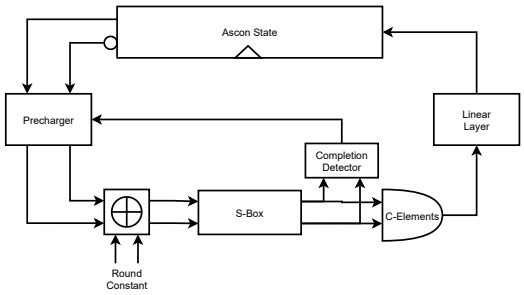


Figure 8: Ascon permutation implementation

Masking the S-Box. The Ascon S-box is a 5-bit S-box with an algebraic degree of 2. It is similar to χ-layer in Keccak, except for two affine transformations at input and output. The similarity allows us to directly insert the SESYM-masked χ implementation defined in the previous section into the Ascon S-box. Our implementation is shown in Figure 9. We omit the completion detector circuitry before the C-elements for readability, but point out that it is necessary for single-cycle operation and is included in our post-synthesis results. Another important point is the need for WDDL XOR gates at the input affine transformation. They are needed to guarantee handshaking from the single-to-dual-rail converter source. There are only linear computations after the χ-layer and therefore no risk of glitching causing the unmasked combination of shared secrets, so we can insert the C-elements directly before the affine output transformation.

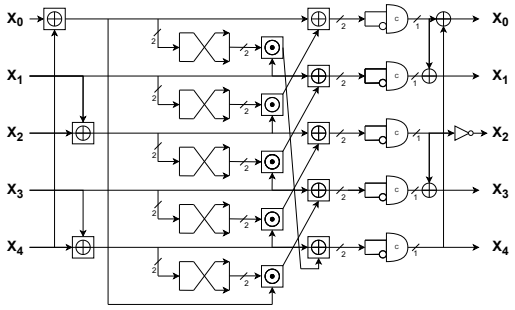


Figure 9: SESYM low-latency S-box

Results and Comparison. Our post-synthesis results are given in Table 1. Compared to the area requirements of GLM [GIB18], our approach results in comparable area for all protection orders. Like GLM, the number of registers grows quadratically with the protection order. The increase in area for our implementation is due to the addition of the precharging logic, C-elements, completion detector and WDDL gadgets. The area cost of the WDDL XOR is particularly expensive since it requires 3 AND and 3 OR gates. Moreover, the completion detector is relatively expensive compared to the other components.

On the other hand, our solution holds a number of other advantages over GLM, the most important of which is the reduction of online randomness requirement by a factor of 6.4. Indeed, SESYM-masking requires the same amount of online randomness as DOM - $\frac{d(d+1)}{2}$ bits per SESYM AND gate. Besides that, since our approach is not strictly tied to a specific masking scheme, a designer might also opt for choosing a different baseline masking scheme that does not require any online randomness, e.g., following the ideas of [Dae17] or [Dae+20]. Another caveat of GLM is the rapid expansion of variable shares after each non-linear gate. In the context of Ascon, the growth from $(d+1) \rightarrow (d+1)^2$ shares is not significant, however as we will show later for AES, the growth for S-boxes with high algebraic degree makes the GLM approach unsuitable for practical applications. The reported maximum clock frequencies are not directly comparable due to the different cell libraries used to generate results.

Table 1: Ascon Permuation Synthesis Results

Protection Order	Area [kGE]	Cycle/round [cycle]	Randomness [bits/cycle]	Max Clock Freq. MHz
This Work				
Unprotected	6.38	1	-	890.5
1	50.40	1	320	408.3
2	102.39	1	960	377.1
3	172.05	1	1 920	358.4
4	257.13	1	3 200	334.2
5	357.65	1	4 800	312.9
GLM [GIB18]				
1	42.59	1	2 048	260.0
2	90.78	1	4 608	-
3	153.76	1	8 192	-
4	238.15	1	12 800	-
5	339.67	1	18 432	-

4.2 SESYM-masked AES S-box

Masking the AES S-box is a notoriously difficult problem due to its high algebraic degree that further complicates its use for applications that are latency sensitive. Many implementations of the AES S-box exist with different optimization targets. We investigated both the Boyar-Peralta [BP12] and the Canright [Can05] designs.

The SESYM-masked implementation follows the same procedure as the Ascon and Keccak designs: a DOM implementation is created first and then all AND and XOR gates are replaced with SESYM counterparts. We placed single-to-dual-rail converters (dual output registers) and precharging circuits at each input, and C-elements at each output. A completion detector prior to the C-elements controls the precharge signal.

Results and Comparison. The post-synthesis results for the 1^{st} and 2^{nd}-order SESYM-masked AES S-boxes are given in Table 2, as well as a summary of related work on the low-latency masked AES S-box. Both designs compute in one clock-cycle and include prechargers, registers for both input and random bits, completion detector and C-elements. The 1^{st}-order protected BP design takes 3.98 kGE and requires 34 online random bits. The 1^{st}-order Canright design is larger at 7.59 kGE but requires half the online randomness as the BP version. This is due to the fact that the Canright design is built with only independent multipliers. In Section 5, we show this construction is secure. For the BP design, the circuitry around the S-box consumes 16% additional area. For typical hardware designs, the Canright design ought to be the most area efficient. Interestingly, this is not the case with our designs. This is due to the difference in the number of WDDL XORs. In fact, the BP design requires 324 WDDL XORs whereas the Canright design requires 565. The large amount of XORs is due to the linear basis transformations at the input and output of the S-box. A single WDDL XOR takes 10.36 GEs, constituting a major component of the total area consumption.

The body of work which report on masked implementations with single-cycle computation is comparatively small. Compared to the best performing work in this category by [Sas+20], the full SESYM-BP implementation requires 12.7% more area and 34 instead of 36 bits of randomness. Compared to GLM, SESYM scales very well both in terms of randomness requirement and in S-box algebraic degree and lends itself to a practical design in the single-cycle category. Our main advantage over the work done by [Sas+20] is the generic approach, flexibility in the baseline masking that allows for various trade-offs, and out-of-the-box support for higher masking orders. In the second-order protection category, SESYM is the only implementation with single-cycle computation. The GLM implementation is able to compute in two cycles, but requires 6.1 times more area and over 4kb of online randomness. Compared to the DOM implementation, we require twice the area and randomness, but improve the latency by a factor of 8. Our implementations could be further optimized to reduce the overall area consumption. Furthermore, SESYM scales to arbitrary protection order and the impact on area, randomness requirement and throughput for d^{th}-order protection remain unclear. Like our Ascon implementation, the online randomness requirement can be further reduced with existing randomness reduction techniques. We leave these areas open for future work.

4.3 SESYM-masked AES-128

To more easily compare our work with LMDPL, we implemented a 1^{st} and 2^{nd}-order SESYM-masked AES-128 (encryption only, masked key-schedule). For these designs, we chose the BP S-box due to the lower area overhead. Our results are given in Table 3. Compared to LMDPL, our implementation uses roughly 34% less area, but has a lower maximum frequency (although this is not directly comparable due to different process technologies). The throughput of both designs is expected to be equal since SESYM does not require explicit precharging cycles. SESYM does impact the critical path of the circuit; all WDDL XORs have double the logical depth than their single rail counterparts. The critical path increases proportionally to the number of XORs in the deepest combinatorial path.. A designer may choose to add one or two register stages in case the desired clock frequency of the overall design is otherwise impacted by the AES S-box. A more direct comparison with LMDPL and SESYM synthesized with the same manufacturing process remains as future work.

Table 2: Summary of related results on 1^{st} and 2^{nd}-order protected AES S-Boxes. Our designs are synthesized for a 100-MHz clock frequency

Implementation	Method	Area [kGE]	Latency [cycles]	Randomness [bits/cycle]
First-Order Implementations				
[Sas+20]	LMDPL	3.48	1	36
This work	**SESYM-BP**	**3.98**	**1**	**34**
This work	**SESYM-Canright**	**7.59**	**1**	**18**
[GIB18]	GLM	60.73	1	2 048
[LMW14]	LMDPL	2.83	2	36
[GIB18]	GLM	6.74	2	416
[AZN21]	4-share LLTI	25.78	2	0
[AZN21]	4-share TI	58.41	2	0
[MRB18]	Multiplicative Masking	1.69	2+3	19
[GC17]	3-share TI Boyar-Peralta	2.91	3	20
[GC17]	3-share TI Canright	2.84	3	32
[Bil+15]	4-share TI	3.71	3	44
[Bil+15]	3-share TI	2.84	3	32
Second-Order Implementations				
This work	**SESYM-BP**	**9.34**	**1**	**102**
This work	**SESYM-Canright**	**14.78**	**1**	**51**
[GIB18]	GLM	57.11	2	4 446
[Cnu+16]	$(d+1)$-share TI	3.66	6	54
[GMK17]	DOM	4.50	8	54

5 Side-Channel Evaluation

To further underline the practical protection of SESYM gadgets, we conduct a side-channel analysis of the Ascon permutation and AES-128 using contemporary Test Vector Leakage Assessment (TVLA) methodologies [Goo+11]. For both algorithms, we analyze the designs under a univariate attack setting. We recorded the first and second-order statistical moments of a 1^{st}-order protected Ascon permutation over 10 million traces. We then extended our measurements for a 2^{nd}-order protected AES and recorded the first, second and third statistical moments over 100 million traces. Moreover, we provide a formal analysis of first and second order SESYM gadgets, targeting the proposed Ascon and AES S-boxes using the verification tool Coco.

Experiment Setup and Measurements. The main purpose of our SESYM approach is to construct ASIC designs of masked S-box circuits that can be securely evaluated within a single clock cycle. For the purpose of an empirical analysis, the fabrication of such ASIC designs is clearly out of scope. However, in order to show that our approach can indeed lead to secure implementations in practice, we map our approach to an FPGA design flow as closely as possible. Several things need to be considered when doing this mapping.

The Ascon and AES implementations were modified such that the precharging and evaluation of the WDDL gates occur over two clock cycles. This was necessary due to the difficulty of implementing C-elements on FPGAs as well as the completion detector loop, which infers a combinatorial loop. On FPGAs, C-elements are typically implemented within a single LUT with the output wire as feedback, or with a latch. When attempting both designs, we found that the synthesizer was unable to reconcile either implementation

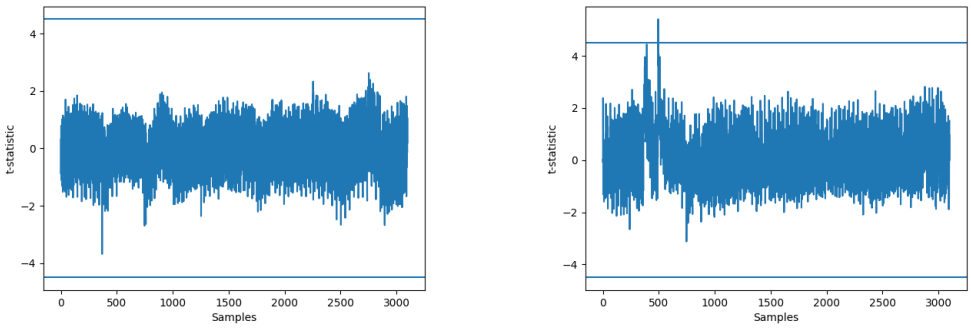
Table 3: AES-128, encryption only with protected key-schedule. Note, the clock frequency numbers are not directly comparable between SESYM and LMDPL due to different process technologies.

Protection Order	Area [kGE]	Cycle/round [cycle]	Randomness [bits/cycle]	Max Clock Freq. [MHz]
This Work				
1	104.86	1	680	192.3
2	203.90	1	2040	169.2
LMDPL [Sas+20]				
1	157.50	1	976	400

and produced circuits with unstable output. In conventional FPGA design, un-clocked state-holding logic is considered undesirable so synthesis tools are not designed with such cases in mind. This is not an issue when considering the ASIC flow used in Section 4. While asynchronous and clock-less circuits have been implemented on FPGAs, the design flow is underdeveloped and beyond the scope of our work. Aside from this, the FPGA implementation is designed to be as close to our ASIC one. These differences do not fundamentally affect our security claims, as the evaluation of the S-box circuits still occur in one clock cycle. The WDDL gates are implemented in single LUT6_2 primitives described in Appendix A. We then analyzed the power side-channel leakage of our Ascon and AES implementations using the CW305 board from NewAE [Tec]. The CW305 hosts an Artix-7 xc7a100t FPGA target and SAM3U microcontroller which provides a software interface to the target via USB. The implementation is placed onto the FPGA within a wrapper which provides an interface for the SAM3U and we included a 1024-bit LFSR to provide the online randomness. The random bits are computed before each measurements to avoid additional noise. The FPGA operates with a 1-MHz clock and we measure the input voltage V_{cc} with PicoScope6000-series oscilloscope which provides 8-bit samples at 2-GS/s. For our evaluation, the input data is generated and split into Boolean shares from a host computer then sent to the device via USB. The input data is randomized between fixed message and random message test batteries. To verify our setup, we employed the technique used by [Sas+20], where a physical switch is tied to the enable pin of the PRNG and an LED. We observed information leakage (as expected) when we applied Welch's t-test to measurements taken with the switch in the "off" position.

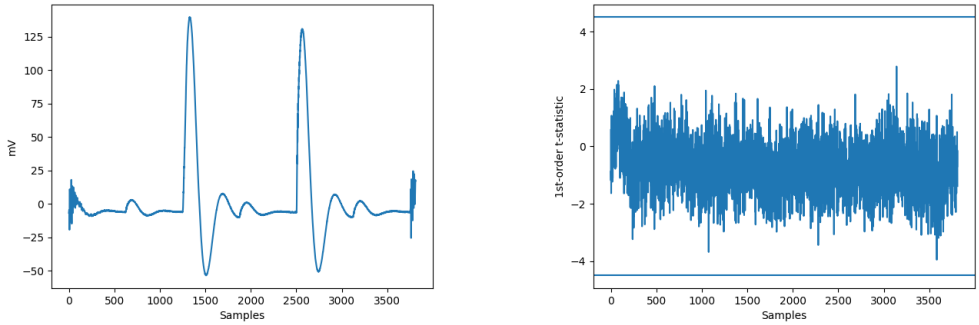
5.1 Results and Discussion

Ascon. We perform Welch's t-test with 10-million traces by encrypting either fixed or random plaintexts and observing the power trace of 12 Ascon permutation rounds. The null hypothesis is that the implementation is considered secure if the means of the power traces are identical. If the absolute t-value exceeds 4.5, we can reject the null hypothesis and claim the implementation leaks information. Figure 10 shows the results of the evaluation. The absolute value of the first-order t-statistic remains within the 4.5 bound and demonstrates that our implementation is first-order secure for up-to 10 million traces. The second-order t-statistic does break the ± 4.5 bound and there is information leakage in the second-order scenario. This is expected considering that the implementation is only first-order masked. Aside from the secure construction of the SESYM gadgets, the dual-rail logic additionally provides some power-hiding benefit that is additive, but not necessary for the security of our constructions. The precharging action adds additional noise to the circuit and tends to a "balanced" power consumption for arbitrary input. We emphasize this phenomenon is not required for the practical security of our constructions, albeit it is a helpful a side-effect of DPL circuits.



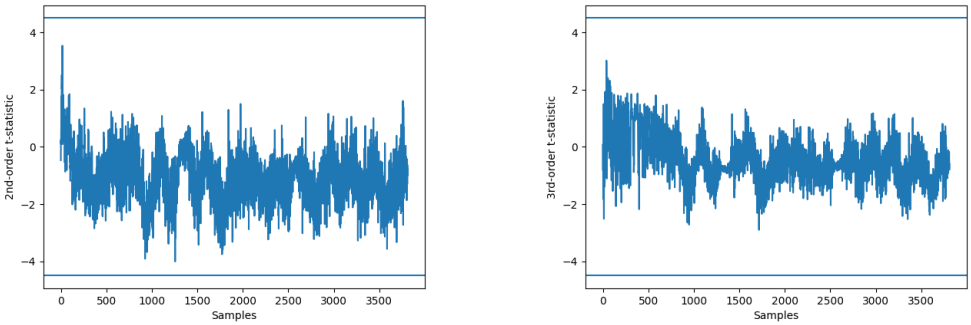
(a) 1^{st} order t-test (b) 2^{nd} order t-test

Figure 10: SCA results for the SESYM-masked 1^{st}-order Ascon FPGA implementation. (10 Million Traces)



(a) Average Power trace. The first peak indicates the transition from precharge to evaluation. The second peak indicates the transition from evaluation to precharge

(b) 1^{st} order t-test



(c) 2^{nd} order t-test (d) 3^{rd} order t-test

Figure 11: SCA results for the SESYM-masked 2^{nd}-order AES FPGA implementation. (100 Million Traces)

AES-128. Following the Ascon experiments, we extended our univariate t-tests to 100 million encryptions. Specifically, we measured one full AES round from precharge →

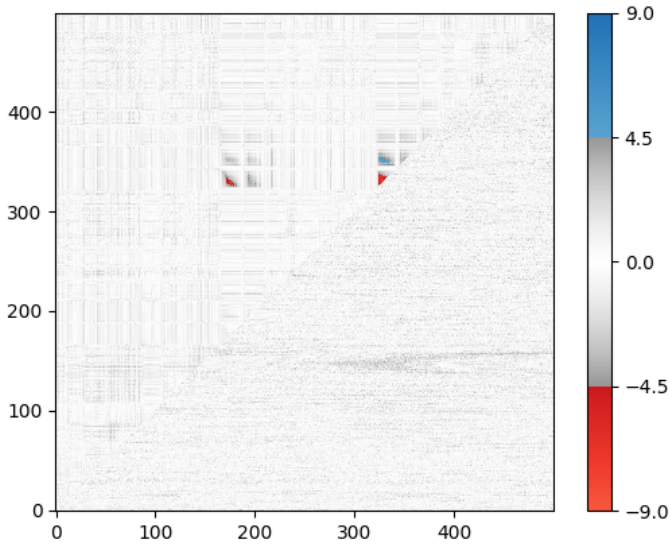


Figure 12: Bivariate analysis of our 2^{nd}-order AES-128 implementation. The lower triangle shows the results of 10 million traces with the RNG enabled. The upper triangle shows the result of 100k traces with the RNG disabled

evaluation → precharge. Our AES implementation is second-order secure and uses the Canright S-box with only DOM-independent multipliers. Figure 11 shows the results of the experiments. For all t-tests, the the t-statistics is with the ±4.5 bound. Interesting, the design does not show any leakage in the 3rd-order. This is most likely due to the dual-rail "hiding" effect. A similar observation was made by Sasdrich et. al in their 1^{st}-order secure LMDPL implementation of AES [Sas+20].

Bivariate analysis. We performed a bivariate analysis of our 2^{nd}-order AES-128 implementation. We reduced the sample count to 500 points to alleviate the computational overhead. We then normalized each trace by the mean and performed a pairwise multiplication of each sample point. Then, we performed a second-order t-test of the larger (fixed vs random) trace sets. The upper triangle shows the results of 100k traces with online randomness disabled i.e., SESYM-masking disabled. The lower triangle shows the result of masking enabled. In the lower triangle, there is no observable leakage, however there is leakage in the upper triangle as expected.

5.2 Formal Verification

In addition to empirical measurements, we apply the formal verification tool Coco [Gig+21] to verify that the addition of WDDL circuitry does not introduce problems in the context of masking. Coco's initial purpose was the verification of masked software executed by a specific CPU [Gig+21; GPM21], but was recently also applied to masked hardware circuits [HB21]. Coco considers the time-constrained probing model, which allows an attacker to distribute d probes in the CPU netlist, in arbitrary execution cycles of the masked software. In this work, we verify first-order single-cycle hardware implementations, for which the time-constrained probing model corresponds to the d-probing model [ISW03]. Therefore, our results are valid in the d-probing model, and we can effectively detect leaks

which are caused by hardware side-effects like glitches and transitions.

As a first step in the verification process, the circuit inputs are labeled in order to express their purpose in the masking scheme. Labels are either *shares* of a native value, *fresh randomness* in case of a fresh independent random variable, or *public*, which includes constants and control signals like the clock signal. Coco operates on the netlist of a masked hardware circuit, consisting of gates connected by wires. During the verification, each gate is assigned a correlation set, which is used to determine whether the gate output correlates with a native value. In a nutshell, the correlation set contains the labels of all values which might be visible to the attacker on the gate output during a clock cycle. The correlation sets of the circuit inputs contain the initially assigned labels, which are then further propagated through the remaining gates, for which the gate type determines the labels in the correlation set. For example, an XOR gate which takes a 1-bit share a, and a 1-bit random value r, will generate the correlation set $\{0, a, r, a \oplus r\}$ on the output. This means, an attacker can either observe an arbitrary independent constant (denoted by 0), the share a (if r is delayed), the randomness r (if a is delayed), or the value $a \oplus r$ once the circuit has stabilized. The whole correlation set assigned to the output of the gate is propagated to the input of the following gate according to the netlist. Coco reports a leak if there exists a correlation set in the circuit which contains a term which directly depends on the unshared native value.

The original version of Coco implements a standard set of ASIC gates. In order to support WDDL gates, we add a new gate type to the verification tool which represents a WDDL XOR gate. We integrate the respective rules into the tool to compute its correlation set and describe its leakage behavior. Tiri et. al [TV04] derive a proof that the WDDL XOR gate is glitch-free, i.e. either outputs 0 or the complete XOR of the inputs, if used correctly. Following the example above, a WDDL XOR gate with the inputs $(a, \neg a)$ and $(r, \neg r)$, will either output 0 (an arbitrary independent constant), or $a \oplus r$. Compared to a standard XOR gate, we can omit the single terms a and r because we know the gate will not glitch. Finally, we derive the complete correlation set of a WDDL XOR gate as $\{0, a \oplus r\}$, which is assigned to the gate's output and will then further propagate through the circuit. We then apply Coco and can successfully verify the correctness of first-order implementation of the SESYM multiplier as well as a SESYM-variant of the Keccak S-box. Additoinally, we successfully verify a first- and second-order implementation of the SESYM-AES-BP S-box. Ultimately, this means that (1) our additional circuitry does not introduce any unintended combinations of shares (assuming WDDL XORs are glitch-free), (2) all components that are not designed to be glitch-free do indeed not cause any glitch-related problems.

6 Conclusions

In this work, we have presented a generic and efficient method for designing single-cycle glitch-resistant (higher-order) masked hardware designs of any cryptographic S-box. We reexamined the role of asynchronous circuit design in the context of low-latency masking and show that it can produce competitive results. More concretely, we showed that WDDL logic gates and Muller C-elements, that are easy to implement with standard CMOS cells, can be used to design cryptographic hardware circuits that, especially in case of higher-degree S-boxes, have noticeably lower requirements in terms of area and online randomness than existing solutions. In the case of low-degree S-boxes our constructions mostly match the best existing solutions but are more flexible in the choice of masking scheme and order, which allows for better trade-off potentials for concrete applications. We have applied our method to DOM-based S-boxes of Ascon and AES and compare the resulting circuits to existing TI, GLM, and LMDPL masking schemes in terms of area, latency, and randomness requirements. For Ascon, we obtained area and latency

results similar to GLM and improved on the randomness requirement by 6.4 times. We also provided results for the 1^{st} and 2^{nd}-order protected AES S-box. Here, the 1^{st}-order variant is similar in terms of area and randomness requirements to the state-of-the-art LMDPL variant. To the best of our knowledge, our higher-order AES S-box and AES-128 designs are the first in literature. We also further underline the practical protection of our constructions against power analysis attacks via empirical and formal verification approaches.

References

[AZN21] Victor Arribas, Zhenda Zhang, and Svetla Nikova. "LLTI: Low-Latency Threshold Implementations". In: *IEEE Trans. Inf. Forensics Secur.* 16 (2021), pp. 5108–5123. DOI: 10.1109/TIFS.2021.3123527. URL: https://doi.org/10.1109/TIFS.2021.3123527.

[Bel+17] Sonia Belaid et al. "Private Multiplication over Finite Fields". In: *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10403. Lecture Notes in Computer Science. Springer, 2017, pp. 397–426. DOI: 10.1007/978-3-319-63697-9_14. URL: https://doi.org/10.1007/978-3-319-63697-9%5C_14.

[Ber+13] Guido Bertoni et al. "Keccak". In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 313–314. DOI: 10.1007/978-3-642-38348-9_19. URL: https://doi.org/10.1007/978-3-642-38348-9%5C_19.

[Bil+13] Begül Bilgin et al. "Efficient and First-Order DPA Resistant Implementations of Keccak". In: *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*. Ed. by Aurélien Francillon and Pankaj Rohatgi. Vol. 8419. Lecture Notes in Computer Science. Springer, 2013, pp. 187–199. DOI: 10.1007/978-3-319-08302-5_13. URL: https://doi.org/10.1007/978-3-319-08302-5%5C_13.

[Bil+15] Begül Bilgin et al. "Trade-Offs for Threshold Implementations Illustrated on AES". In: *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 34.7 (2015), pp. 1188–1200. DOI: 10.1109/TCAD.2015.2419623. URL: https://doi.org/10.1109/TCAD.2015.2419623.

[BP12] Joan Boyar and René Peralta. "A Small Depth-16 Circuit for the AES S-Box". In: *Information Security and Privacy Research - 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012, Heraklion, Crete, Greece, June 4-6, 2012. Proceedings*. Ed. by Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou. Vol. 376. IFIP Advances in Information and Communication Technology. Springer, 2012, pp. 287–298. DOI: 10.1007/978-3-642-30436-1_24. URL: https://doi.org/10.1007/978-3-642-30436-1%5C_24.

[Can05] David Canright. "A Very Compact S-Box for AES". In: *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*. Ed. by Josyula R. Rao and Berk Sunar. Vol. 3659. Lecture Notes in Computer Science. Springer,

2005, pp. 441–455. DOI: 10.1007/11545262_32. URL: https://doi.org/10.1007/11545262%5C_32.

[Cnu+16] Thomas De Cnudde et al. "Masking AES With d+1 Shares in Hardware". In: *Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016*. Ed. by Begül Bilgin, Svetla Nikova, and Vincent Rijmen. ACM, 2016, p. 43. DOI: 10.1145/2996366.2996428. URL: https://doi.org/10.1145/2996366.2996428.

[Dae+20] Joan Daemen et al. "Protecting against Statistical Ineffective Fault Attacks". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.3 (2020), pp. 508–543. DOI: 10.13154/tches.v2020.i3.508-543. URL: https://doi.org/10.13154/tches.v2020.i3.508-543.

[Dae17] Joan Daemen. "Changing of the Guards: A Simple and Efficient Method for Achieving Uniformity in Threshold Sharing". In: *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Ed. by Wieland Fischer and Naofumi Homma. Vol. 10529. Lecture Notes in Computer Science. Springer, 2017, pp. 137–153. DOI: 10.1007/978-3-319-66787-4_7. URL: https://doi.org/10.1007/978-3-319-66787-4%5C_7.

[Dob+21] Christoph Dobraunig et al. "Ascon v1.2: Lightweight Authenticated Encryption and Hashing". In: *J. Cryptol.* 34.3 (2021), p. 33. DOI: 10.1007/s00145-021-09398-9. URL: https://doi.org/10.1007/s00145-021-09398-9.

[FB97] Karl M. Fant and Scott A. Brandt. *NULL Convention Logic*. 1997.

[GC17] Ashrujit Ghoshal and Thomas De Cnudde. "Several Masked Implementations of the Boyar-Peralta AES S-Box". In: *Progress in Cryptology - INDOCRYPT 2017 - 18th International Conference on Cryptology in India, Chennai, India, December 10-13, 2017, Proceedings*. Ed. by Arpita Patra and Nigel P. Smart. Vol. 10698. Lecture Notes in Computer Science. Springer, 2017, pp. 384–402. DOI: 10.1007/978-3-319-71667-1_20. URL: https://doi.org/10.1007/978-3-319-71667-1%5C_20.

[GIB18] Hannes GroSS, Rinat Iusupov, and Roderick Bloem. "Generic Low-Latency Masking in Hardware". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018.2 (2018), pp. 1–21. DOI: 10.13154/tches.v2018.i2.1-21. URL: https://doi.org/10.13154/tches.v2018.i2.1-21.

[Gig+21] Barbara Gigerl et al. "Coco: Co-Design and Co-Verification of Masked Software Implementations on CPUs". In: *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. Ed. by Michael Bailey and Rachel Greenstadt. USENIX Association, 2021, pp. 1469–1468. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/gigerl.

[GMK16] Hannes GroSS, Stefan Mangard, and Thomas Korak. "Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order". In: *IACR Cryptol. ePrint Arch.* (2016), p. 486. URL: http://eprint.iacr.org/2016/486.

[GMK17] Hannes GroSS, Stefan Mangard, and Thomas Korak. "An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order". In: *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*. Ed. by Helena Handschuh. Vol. 10159. Lecture Notes in Computer Science. Springer, 2017, pp. 95–112. DOI: 10.1007/978-3-319-52153-4_6. URL: https://doi.org/10.1007/978-3-319-52153-4%5C_6.

[Goo+11] Gilbert Goodwill et al. “A testing methodology for side channel resistance”. In: 2011.

[GPM21] Barbara Gigerl, Robert Primas, and Stefan Mangard. “Secure and Efficient Software Masking on Superscalar Pipelined Processors”. In: *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13091. Lecture Notes in Computer Science. Springer, 2021, pp. 3–32. DOI: 10.1007/978-3-030-92075-3_1. URL: https://doi.org/10.1007/978-3-030-92075-3%5C_1.

[HB21] Vedad Hadzic and Roderick Bloem. “COCOALMA: A Versatile Masking Verifier”. In: *Formal Methods in Computer Aided Design, FMCAD 2021, New Haven, CT, USA, October 19-22, 2021*. IEEE, 2021, pp. 1–10. DOI: 10.34727/2021/isbn.978-3-85448-046-4_9. URL: https://doi.org/10.34727/2021/isbn.978-3-85448-046-4%5C_9.

[ISU18] Vincent Immler, Robert Specht, and Florian Unterstein. “Your rails cannot hide from localized EM: how dual-rail logic fails on FPGAs - extended version”. In: *J. Cryptogr. Eng.* 8.2 (2018), pp. 125–139. DOI: 10.1007/s13389-018-0185-x. URL: https://doi.org/10.1007/s13389-018-0185-x.

[ISW03] Yuval Ishai, Amit Sahai, and David A. 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*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 463–481. DOI: 10.1007/978-3-540-45146-4_27. URL: https://doi.org/10.1007/978-3-540-45146-4%5C_27.

[KJJ99] Paul C. 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*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 388–397. DOI: 10.1007/3-540-48405-1_25. URL: https://doi.org/10.1007/3-540-48405-1%5C_25.

[LMW14] 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*. Ed. by Lejla Batina and Matthew Robshaw. Vol. 8731. Lecture Notes in Computer Science. Springer, 2014, pp. 580–597. DOI: 10.1007/978-3-662-44709-3_32. URL: https://doi.org/10.1007/978-3-662-44709-3%5C_32.

[MC13] Matheus Trevisan Moreira and Ney Laert Vilar Calazans. “Design of standard-cell libraries for asynchronous circuits with the ASCEnD flow”. In: *IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2013, Natal, Brazil, August 5-7, 2013*. IEEE Computer Socity, 2013, pp. 217–218. DOI: 10.1109/ISVLSI.2013.6654647. URL: https://doi.org/10.1109/ISVLSI.2013.6654647.

[MI14] 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*. Ed. by Lejla Batina and Matthew Robshaw. Vol. 8731. Lecture Notes in Computer Science. Springer, 2014, pp. 598–

615. DOI: 10.1007/978-3-662-44709-3_33. URL: https://doi.org/10.1007/978-3-662-44709-3%5C_33.

[Mor+12] Matheus T. Moreira et al. "Impact of C-elements in asynchronous circuits". In: *Thirteenth International Symposium on Quality Electronic Design, ISQED 2012, Santa Clara, CA, USA, March 19-21, 2012*. Ed. by Keith A. Bowman et al. IEEE, 2012, pp. 437–343. DOI: 10.1109/ISQED.2012.6187530. URL: https://doi.org/10.1109/ISQED.2012.6187530.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. "Side-Channel Leakage of Masked CMOS Gates". In: *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*. Ed. by Alfred Menezes. Vol. 3376. Lecture Notes in Computer Science. Springer, 2005, pp. 351–365. DOI: 10.1007/978-3-540-30574-3_24. URL: https://doi.org/10.1007/978-3-540-30574-3%5C_24.

[MRB18] Lauren De Meyer, Oscar Reparaz, and Begül Bilgin. "Multiplicative Masking for AES in Hardware". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018.3 (2018), pp. 431–468. DOI: 10.13154/tches.v2018.i3.431-468. URL: https://doi.org/10.13154/tches.v2018.i3.431-468.

[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*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. Lecture Notes in Computer Science. 2016, pp. 517–547. DOI: 10.1007/978-3-662-53887-6_19. URL: https://doi.org/10.1007/978-3-662-53887-6%5C_19.

[Mul56] David E. Muller. "A Theory of Asynchronous Circuits". In: *Report 75, University of Illinois* (1956).

[Oli+16] Carlos Henrique Menezes Oliveira et al. "ASCEnD-FreePDK45: An open source standard cell library for asynchronous design". In: *2016 IEEE International Conference on Electronics, Circuits and Systems, ICECS 2016, Monte Carlo, Monaco, December 11-14, 2016*. IEEE, 2016, pp. 652–655. DOI: 10.1109/ICECS.2016.7841286. URL: https://doi.org/10.1109/ICECS.2016.7841286.

[QS01] Jean-Jacques Quisquater and David Samyde. "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards". In: *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*. Ed. by Isabelle Attali and Thomas P. Jensen. Vol. 2140. Lecture Notes in Computer Science. Springer, 2001, pp. 200–210. DOI: 10.1007/3-540-45418-7_17. URL: https://doi.org/10.1007/3-540-45418-7%5C_17.

[Sas+20] Pascal Sasdrich et al. "Low-Latency Hardware Masking with Application to AES". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.2 (2020), pp. 300–326. DOI: 10.13154/tches.v2020.i2.300-326. URL: https://doi.org/10.13154/tches.v2020.i2.300-326.

[Sug19] Takeshi Sugawara. "3-Share Threshold Implementation of AES S-box without Fresh Randomness". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.1 (2019), pp. 123–145. DOI: 10.13154/tches.v2019.i1.123-145. URL: https://doi.org/10.13154/tches.v2019.i1.123-145.

[Tec] New AE Technology. *NAE-CW305*. URL: https://www.newae.com/products/NAE-CW305.

[TV04] Kris Tiri and Ingrid Verbauwhede. "A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation". In: *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*. IEEE Computer Society, 2004, pp. 246–251. DOI: 10.1109/DATE.2004.1268856. URL: https://doi.org/10.1109/DATE.2004.1268856.

A LUT Implementations of WDDL Gates.

Below we present the Verilog instantiations for implementing the WDDL AND and WDDL XOR gates with Xilinx LUT6_2 primitives inspired by similar implementations done by [MI14] for AWDDL gates. The key difference is the LUT .INIT() mask.

Listing 1: WDDL XOR Implementation

```
`define T 0
`define F 1
module WDDL_XOR(
 // Outputs
 c,
 // Inputs
 a, b, pre
 );

 input [1:0] a;
 input [1:0] b;
 input pre; // precharge
 output [1:0] c;

LUT6_2 #(
     .INIT(64'h00300C00000C3000)
 ) LUT6_2_inst (
                .O6(c[`F]),
                .O5(c[`T]),
                .I0(a[`T]),
                .I1(a[`F]),
                .I2(b[`T]),
                .I3(b[`F]),
                .I4(pre),
                .I5(1'b1)
 );
endmodule // WDDL_XOR
```

Listing 2: WDDL AND Implementation

```
`define T 0
`define F 1
module WDDL_AND(
 // Outputs
 c,
 // Inputs
 a, b, pre
 );

 input [1:0] a;
 input [1:0] b;
 input pre; // precharge
 output [1:0] c;

LUT6_2 #(
    .INIT(64'h000C3C0000300000)
 ) LUT6_2_inst (
                .O6(c[`F]),
                .O5(c[`T]),
                .I0(a[`T]),
                .I1(a[`F]),
                .I2(b[`T]),
                .I3(b[`F]),
                .I4(pre),
                .I5(1'b1)
 );

endmodule // WDDL_AND
```