Low-Latency Masking with Arbitrary Protection Order Based on Click Elements

Mateus Simões, Lilian Bossuet, Nicolas Bruneau, Vincent Grosso, Patrick Haddad, Thomas Sarno

Low-Latency Masking with Arbitrary Protection Order Based on Click Elements

Mateus Simões[†‡], Lilian Bossuet[‡], Nicolas Bruneau[†], Vincent Grosso[‡], Patrick Haddad[†], Thomas Sarno[†]

[†]*System Research and Applications*
STMicroelectronics
Rousset, France
{mateus.simoes, nicolas.bruneau, patrick.haddad, thomas.sarno}@st.com

[‡]*Laboratoire Hubert Curien — CNRS UMR 5516*
Université de Lyon
Saint-Étienne, France
{lilian.bossuet, vincent.grosso}@univ-st-etienne.fr

***Abstract*—Masking is the main countermeasure against side-channel attacks due to its sound formal proof of security and the scalability of its protection parameters. However, effective masking increases the implementation complexity by requiring additional silicon area, random number generators and higher latency. Thus, reducing the masking implementation costs while conserving its robustness under side-channel attacks is a relevant branch of research in hardware security applications.**

Relying on the two-phase bundled-data protocol, this work presents a low-latency masking implementation with arbitrary protection order. In particular, we base our approach on the click elements to control the handshake logic, allowing us to implement asynchronous circuits using conventional synthesis tools. In this manner, we are able to obtain an effective single-cycle and protected implementation of the AES S-box requiring smaller silicon area and potentially lower power consumption compared to the state-of-the-art. Additionally, we detail the asynchronous design methodology that can be applied in different scenarios to improve the latency of secure hardware designs. Finally, we assess leakages to evaluate the robustness of our approach against side-channel attacks.

***Index Terms*—Side-channel attacks, hardware masking, asynchronous circuits, low-latency, leakage assessment**

I. Introduction

Side-channel attacks [11] represent a threat to electronic systems designed to manipulate encrypted data. This class of security exploit allows an adversary to obtain sensitive information by observing the physical properties of a cryptographic device. In this manner, side-channel traces, such as power consumption and electromagnetic emanation, can be statistically analyzed to reveal secret data — e.g., the cipher key.

To avoid side-channel attacks, various countermeasures exist. Masking [3], [7], the most relevant among these solutions, splits secret data into several uniformly distributed shares, rendering more complex to predict the side-channel behavior of a cryptographic device. Despite its sound formal proof of security [3], implementing a secure masking scheme is not a straightforward task.

Indeed, to satisfy different design and security properties, an effective masking scheme involves significant implementation resources. For instance, to avoid exploitable leakages due to physical hazards such as glitches [15], [16], hardware designers tend to add several register barriers in the circuit [4], which raises the latency of masked modules, i.e., the number of clock cycles needed to finish processing the data. In addition, the protected design requires higher area overhead due to the increase of the implementation complexity.

With the increasing proliferation of IoT devices, secure low-latency and area-efficient cryptographic modules become therefore necessary to satisfy commercial demands. In this manner, many techniques have been proposed to balance the masking implementation costs, with recent efforts aiming at the design of low-latency schemes based on different architectural approaches [1], [8], [20], [22], [29], [31]. Nevertheless, the reduction in the overall clock cycle score is accompanied by higher implementation costs and, in some cases, lower throughput. In fact, many of these solutions rely on asynchronous primitives and dual-rail encoding to obtain low-latency implementations. As a consequence, the overall silicon area is increased due to the chosen logic wiring. Thus, maintaining a single-rail channel may be a better option to achieve superior area efficiency.

Furthermore, designing asynchronous circuits is a challenging task, as most conventional EDA tools — e.g., Synopsys Design Compiler — are not suited to this hardware design approach. For this reason, clockless circuits are not widely adopted as a solution, despite its potential advantages in low power consumption and high performance [33]. Nevertheless, recent works have addressed different methodologies to use established tools, commonly used in synchronous design flow, to ease the implementation of asynchronous circuits [5], [14], [17], [40]. Based on these methodologies, this work proposes a generic approach to design low-latency and area-efficient higher-order secure masking built upon the two-phase bundled-data communication protocol.

We rely on handshake control circuits made of edge-triggered flip-flops to implement asynchronous masking with the aid of conventional synthesis tools. To illustrate this design approach, we present a case study of an asynchronous domain-oriented masked AES S-box, a single-cycle implementation that achieves d-glitch-extended security [4] with lower area requirements compared to related solutions. We also compare the proposed AES S-box with its equivalent synchronous design to illustrate the potential advantages in terms of lower power consumption and higher throughput.

The paper is structured as follows: Sections II and III present the background and the related works, respectively. Section IV discusses the asynchronous design methodology. Then, Section V shows the implementation of the single-cycle AES S-box relying on the domain-oriented masking. Section V also examines the implementation results while Section VI reviews the security robustness of our designs against univariate and bivariate side-channel analysis. Finally, Section VII concludes the paper presenting some perspectives on the future work.

II. Background

A. Masking

The masking countermeasure is based on secret sharing: an algorithm splits the secret data x into several shares x_i in such a way that $x = x_0 \circ x_1 \circ \cdots \circ x_d$, with the symbol $\circ$ denoting a mathematical operation.

Knowledge of all shares $\mathcal{S} = (x_0, x_1, \dots x_d)$ is required to recover the secret. Thus, a subset of $\mathcal{S}$ cannot reveal the unshared data x. In this context, the masking of a linear operation is straightforward, as the shares can be manipulated separately, that is, without recombining them back together. However, the same does not stand for non-linear functions, since recombining the shares may break their statistical independence. In fact, masking non-linear functions effectively is a critical aspect of hardware security.

To illustrate, let us take the domain-oriented masking (DOM) scheme [9] — which is based on Boolean sharing, i.e., $\circ = \oplus$, the XOR operator — to d^{th}-order mask with $d + 1$ shares the multiplication $f(a, b) = a \otimes b = z$ in $GF(2^n)$, expressed as $\mathcal{Z} = \mathcal{A} \otimes \mathcal{B}$. For a first-order masking, the sets $\mathcal{A} = (a_0, a_1)$ and $\mathcal{B} = (b_0, b_1)$ represent the input shares and $\mathcal{Z} = (z_0, z_1)$ the output sharing. Assuming that the input shares are statistically independent, we want to solve $(z_0 \oplus z_1) = (a_0 \oplus a_1) \otimes (b_0 \oplus b_1)$. A non-linear layer computes the product terms $a_0 \otimes b_0$, $a_0 \otimes b_1$, $a_1 \otimes b_0$, $a_1 \otimes b_1$ and adds a fresh random mask r to the cross-domain products, that is, $a_0 \otimes b_1$ and $a_1 \otimes b_0$. Then, to ensure resistance against glitches [4], registers ($\longrightarrow$) store the resulting shares (x_0, x_1, x_2, x_3), as we can see in (1).

$$\begin{aligned} f_0(a_0, b_0) &= a_0 \otimes b_0 && \longrightarrow x_0 \\ f_1(a_0, b_1) &= (a_0 \otimes b_1) \oplus r && \longrightarrow x_1 \\ f_2(a_1, b_0) &= (a_1 \otimes b_0) \oplus r && \longrightarrow x_2 \\ f_3(a_1, b_1) &= a_1 \otimes b_1 && \longrightarrow x_3 \end{aligned} \tag{1}$$

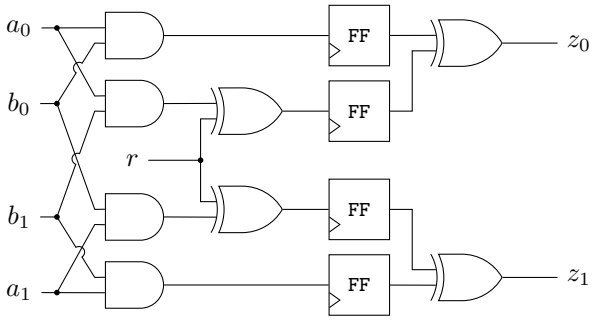

Fig. 1. The first-order DOM multiplier.

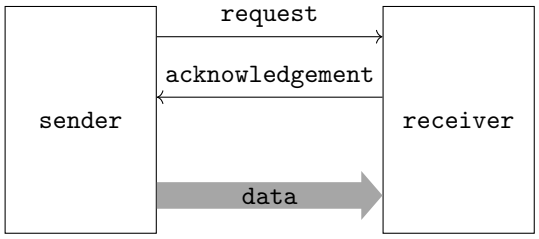

Fig. 2. A bundled-data block diagram.

The non-linear layer produces $(d + 1)^2$ shares and is followed by a sharing compression layer — as shown in (2) for a first-order masking — to reduce the number of shares back to $(d+1)$, preventing a quadratic growth of the number of shares through the computation [28].

$$\begin{aligned} z_0 &= x_0 \oplus x_1 \\ z_1 &= x_2 \oplus x_3 \end{aligned} \tag{2}$$

Thanks to the register barrier between both layers and the ISW random refreshing method [10] to mask the cross-domain products, d-glitch-extended probing security is satisfied [4]. A gate-level design of the DOM multiplier is shown in Fig. 1.

B. Two-Phase Bundled-Data Circuits and the Click Elements

The two-phase bundled-data is an asynchronous circuit implementation style introduced by Ivan Sutherland in [35]. In this design approach, a bundled-data message carries the single-rail data signal alongside with the handshake logic, the `request` and `acknowledgement` signals, as shown in Fig. 2. The sender indicates data availability with the request signal. Then, the receiver uses the acknowledgement channel to signal the computation of the corresponding data. To meet data arrival timing requirements, delay elements are inserted in the request channel between the sender and the receiver. This delay is necessary to obtain a positive slack on the data channel, resulting in an effective data propagation.

In this work, the handshake logic triggers the local control pulse when the correspondent request or acknowledgement signal transitions, which is known as two-phase protocol. The local control pulses are generated by click elements, an asynchronous control circuit introduced in [24] to implement two-phase bundled-data handshake logic.

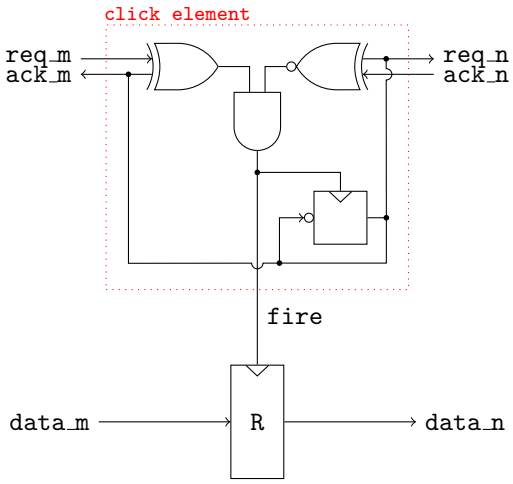

Fig. 3. A single stage bundled-data pipeline with a click element.

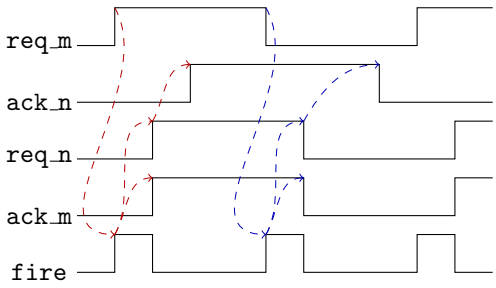

Fig. 4. The handshake logic.

The click element relies on edge-triggered flip-flops and combinatorial gates available in any standard cell library; it is thus an alternative to handshake controllers built upon Muller's C-elements [21], [35] and latches [32]. In this manner, a hardware engineer is able to use conventional EDA software to perform static timing analysis, easing the design of asynchronous circuits with the aid of established tools [14], [40].

A design of the click element is shown in Fig. 3. If `req_m` $\neq$ `ack_m`, new data coming from the sender `m` are available at the input channel. On the other hand, the receiver `n` has collected the output data when `req_n = ack_n`. When new input data, derived from the preceding block, are available, and the current data have been received by the following neighbor, a local pulse, denoted `fire`, is generated to trigger the corresponding flip-flop. Note that, as expected for the two-phase protocol, both the rising and the falling edges of the request signal can produce a fire pulse, as shown in Fig. 4.

We opt to use the click element as a handshake control circuit because it only requires typical cells present in any technology library. Furthermore, the generated pulses can be instantiated as clock objects, allowing us to perform static timing analysis. We do not use `dont_touch` constraints on our click element description.

III. Related Work: Low-Latency Masking

The first work that borrows asynchronous primitives to implement low-latency masking was presented by Moradi and Schneider in [20]. In their work, they implement fully unrolled first-order Threshold Implementations (TI) [23] of PRINCE and Midori built upon Wave Dynamic Differential Logic (WDDL) [36], which relies on dual-rail encoding to produce bitwise operators.

Sasdrich et al. [29] employed the LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL) [13] masking scheme to implement a low-latency AES. By using the pre-charge / evaluation logic with monotonic functions they were able to obtain a glitch-free circuit [25]. Nevertheless, besides being limited to first-order security, their AES design presents a high silicon area cost, as the dual-rail blocks are duplicated in order to perform the evaluation and pre-charge phases in parallel.

More recently, Nagpal et al. [22] presented a low-latency DOM implementation also built upon WDDL gates, but employing Muller C-elements [21] as synchronization modules, whose results have shown to be higher-order secure. A similar approach is proposed in [31], employing data-driven handshake logic built upon Muller C-elements to replace clocked register barriers with self-timed latches in a DOM architecture.

In contrast to the dual-rail approach, but also based on the DOM scheme, Gross et al. proposed the first generic low-latency masking (GLM) [8]. In their work, they skip the compression of shares after the non-linear layer, eliminating the register barrier in (1). However, the number of shares shows a quadratic growth after each masked multiplication. In consequence, the area and randomness costs increase substantially, and special care has to be taken during implementation to avoid collisions when composing multiple masked multipliers.

With an algorithmic approach, Arribas et al. proposed the Low-Latency Threshold Implementations (LLTI) [1]. However, their AES S-box design brings high area overhead and is computed in two combinatorial steps, divided by a register layer, limiting its composability in some low-latency scenarios.

In this paper, we present a secure area-efficient and low-latency masking design approach with arbitrary protection order, as a consequence of using the DOM as a case study. To trigger the register layers, we employ an asynchronous control circuit whose gate-level implementation requires logic cells present in any technology library, making it more convenient to traditional synchronous design flows and easing the application of our masking methodology using already established EDA tools. Also, we focus on bundled-data circuits to avoid the limitations inherent to the dual-rail encoding [12], [18]. Furthermore, we want to eliminate the need of pre-charging a glitch-resistant masking, which may improve throughput

IV. Click-Based Design Methodology

Fig. 5 illustrates a simple bundled-data pipeline with click elements. When input data are ready, a transition of the request signal `req_m` indicates their availability. For instance, the `req_m` triggers the `fire_1` pulse allowing the register R_1 to capture the `data_m`.

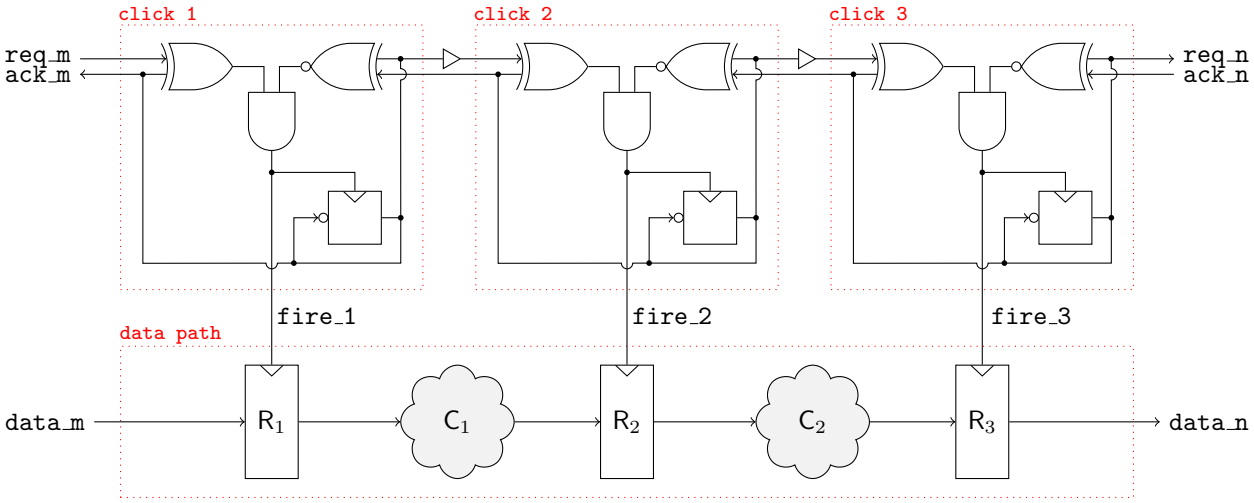

Fig. 5. A simple bundled-data pipeline with click elements.

After capturing the data and processing them through the combinatorial circuit C_1, the acknowledgement output of the second handshake controller transitions, signaling the completion of this step. Now, the next transition of the request signal can produce a new fire pulse. In this manner, both the rising and the falling edges of the request and acknowledgement signals indicate, respectively, data availability and that the data have been processed. The register barriers neighboring the combinatorial blocks are triggered by the correspondent `fire` signal. The `ack_n` and `req_n` of the `click i` are the `ack_m` and `req_m` of the `click j`, with `j = i + 1`.

Note that the timing requirements of a combinatorial circuit must match the request channel delay between the neighboring pair of click elements. For example, the block C_1 must have a stable output before the arrival of a `fire_2` pulse at the R_2 register. To match the delay between click elements, a chain of buffers (illustrated as $\triangleright$) is added to the request channels. We apply a technique based on the works presented in [40] and in [14] to determine the delay lines. The core idea behind their approach is defining the fire pulses as clocks during synthesis in order to report the timing requirements of a path. Thus, we can design the necessary delay chain of a request channel.

A. Delay Matching

The process to match the delays of the request channels is described as follows. First, we declare the variables used to store the delay values for each request channel. Initially, these variables are initialized to zero.

Then, we use the command `create_clock` to define the first fire signal as a physical clock in the design. Next, we derive new clock objects for the following fire pulses from its preceding fire signal as a master clock. For that, we use the `create_generated_clock` command to generate the new clock objects from an existing physical clock in the design. This command also allows us to define a phase relationship among the local pulses.

For instance, in our work we use the configuration below to derive a generated clock `fire` shifted by `t` time units from the source clock `clk`.

```
create_generated_clock -name fire \
-source clk -edges {1 2 3} -edge_shift {t t t}
```

Once the fire pulses are defined as clocks, we can report the timing constraints of a combinatorial path between two of them. For example, let us take the timing path between `fire_1` and `fire_2` in Fig. 5, whose start point is the output of R_1 and the end point is the input of R_2. The setup and the hold time of R_1 and R_2 have to be satisfied to ensure the correct functioning of the circuit. Equation (3) shows a rough way to estimate the necessary delay $t_{\texttt{delay,req}}$ of a request

$$t_{\texttt{delay,req}} > t_{\texttt{clk}\rightarrow\texttt{q}} + t_{\texttt{logic}} + t_{\texttt{setup}} \tag{3}$$

With $t_{\texttt{clk}\rightarrow\texttt{q}}$ the propagation delay of the flip-flop, $t_{\texttt{logic}}$ the propagation delay of the combinatorial circuit and $t_{\texttt{setup}}$ the setup time of the flip-flop. Similarly, the hold time $t_{\texttt{hold}}$ is expressed in (4).

$$t_{\texttt{hold}} < t_{\texttt{delay,ack}} + t_{\texttt{clk}\rightarrow\texttt{q,cd}} + t_{\texttt{logic,cd}} \tag{4}$$

With $t_{\texttt{delay,ack}}$ the delay in the acknowledgement channel, $t_{\texttt{clk}\rightarrow\texttt{q,cd}}$ the contamination delay of the flip-flops and $t_{\texttt{logic,cd}}$ the contamination delay of the combinatorial logic. In general, $t_{\texttt{clk}\rightarrow\texttt{q,cd}} > t_{\texttt{hold}}$ and, due to the gate-level design of the click element, $t_{\texttt{delay,ack}} > t_{\texttt{hold}}$. Therefore, no buffer is needed in the acknowledgement channel.

To ensure appropriate setup time and hold time, we can for instance report the slack of a timing path using the `report_timing` command from `fire_1` to `fire_2`. The slack, expressed in (5), is the difference between the data required time and the data arrival time, see Fig. 6. If the static timing analysis reports a negative slack, the delay in the correspondent request channel is not enough and has to be increased.

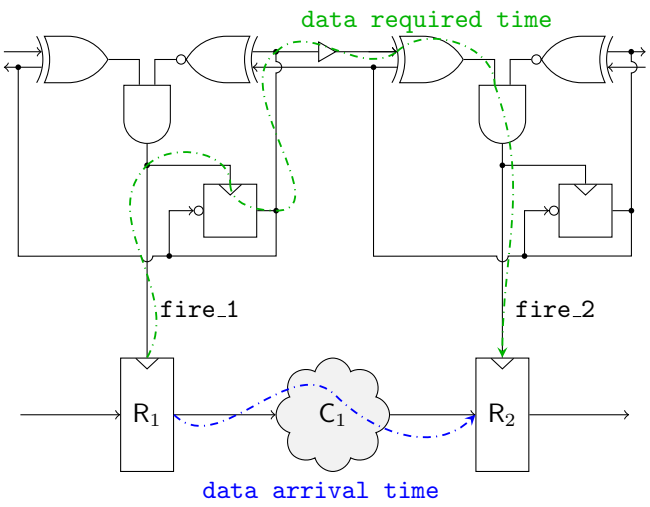

Fig. 6. The data arrival time and the data required time.

$$t_{\texttt{slack}} = t_{\texttt{required}} - t_{\texttt{arrival}} \tag{5}$$

Based on the reported value, we use the `set_min_delay` command to redefine the delay of the request channel. This command adds a chain of buffers on the defined path in order to satisfy the desired delay. To avoid the suppression of the delay lines during the ASIC synthesis, we add — manually — a single buffer with `set_dont_touch` constraints to each request channel. Then, we are able to define the buffer chain length using the `set_min_delay` command, to set a delay of `t` time units from the `req_n` output to the `req_m` input between two click elements.

Initially, we report the slack values without adding the appropriate delay elements. Hence, we could perform a first synthesis following the definition of the clock pulses. After the first compilation, we report the timing requirements in order to obtain the slack values for each timing path. Based on these values, we redefine the delay lines with a 10% margin [40] and the created clocks to match the data required time. Then we can re-synthesize the design with the correct delays. In general, two iterations are necessary to meet the timing requirements. The flowchart in Fig. 7 illustrates the synthesis flow described in this section.

V. Masked Implementation: DOM Case Study

Since this work focuses on the design methodology of low-latency masking, we rely on a secure scheme with arbitrary protection order. Thus, we can evaluate the effectiveness of our approach against side-channel analysis and estimate the implementation overheads. Indeed, this methodology can be applied to any masking scheme containing register barriers in order to obtain a single-cycle circuit.

For benchmark reasons, our low-latency AES S-box implementation is based on the Canright's design [2], which is composed of eight combinatorial stages. Each stage is preceded by a barrier of registers, as shown in Fig. 8.

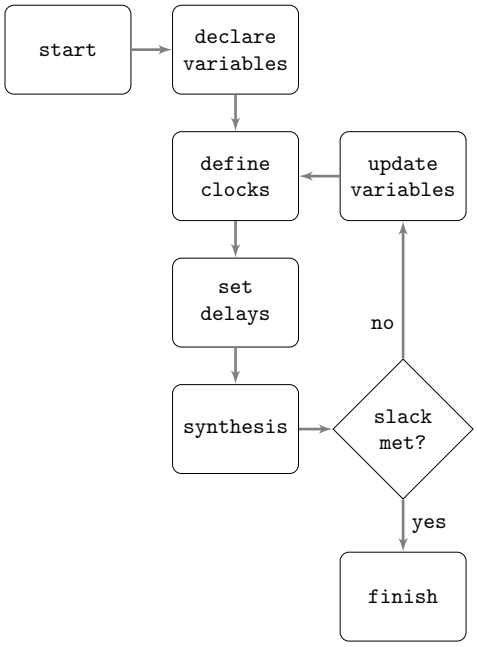

Fig. 7. Synthesis flow with delay matching process.

To obtain a single-cycle design, we trigger each register barrier with a local clock pulse derived from a click element. There are eight click elements $\texttt{C}_\texttt{i}$ in total, one for each register barrier. To match the timing of each combinatorial block, we add a chain of buffers between the control circuits in such a way that the request signal propagation delay is higher than its corresponding data arrival time.

Our AES architecture is locally asynchronous and globally synchronous. To shift the circuit out of steady state into normal operation, we introduce a clocked element, named source, that produces the first request signal, triggering the handshake logic to process the current data. Therefore, the asynchronous pipeline is triggered by a synchronous clock, whose period must satisfy the throughput of the click-based pipeline. The last click element, identified as sink, does not output a request signal. Fig. 9 shows the design of the source and sink blocks used in our design.

Since the input of the S-box remains constant during the computation of the masked byte, we remove the registers after the inner-domain operations of the DOM multiplier, as shown in Fig. 10. Moreover, note that we do not use registers to store the LSB and MSB in $GF(2^2)$ and $GF(2^4)$, as shown in Fig. 8. This results in a smaller design — compared with the original DOM implementation in [9] — by reducing the number of flip-flops in our click-based masked version of the Canright's AES S-box.

A. Synthesis Results

We use Synopsys Design Compiler to synthesize our design using a STM40 nm standard cell library. The area results are normalized in terms of gate equivalent metric (GE) with a two-input NAND gate from the selected library as reference. We do not use `compile_ultra` scripts.

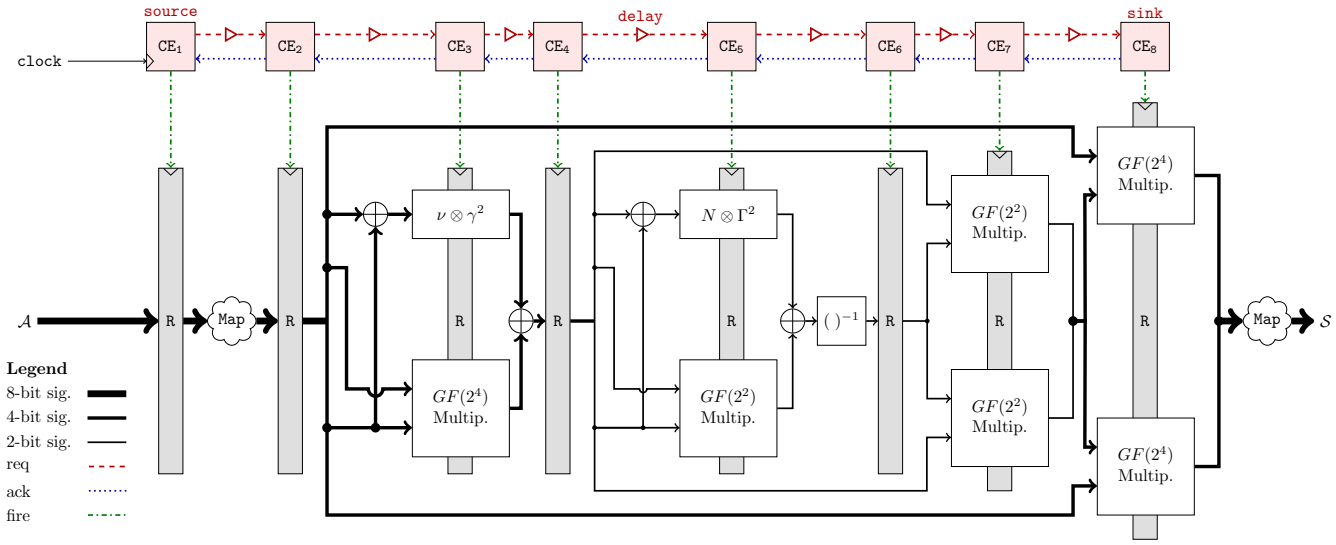

Fig. 8. A low-latency implementation of the Canright's AES S-box design based on click elements.

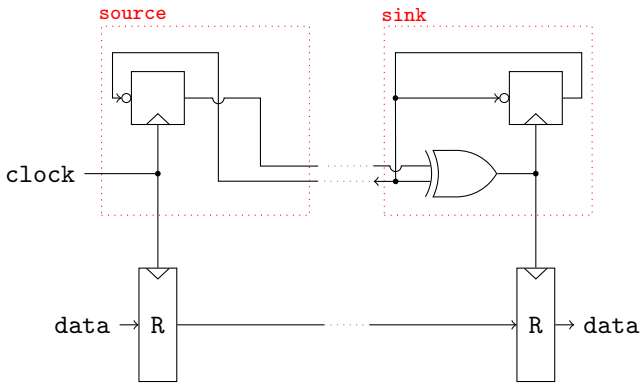

Fig. 9. The source and sink handshake control components.

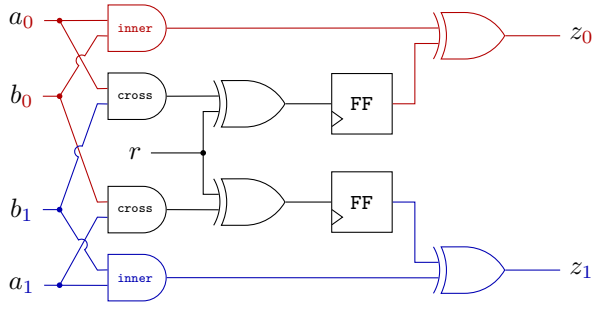

Fig. 10. The first-order DOM multiplier for designs without a pipeline.

Table I reports the performance figures of different masked AES S-box designs. We show several first-order designs, but we focus on the low-latency solutions to compare the higher-order approaches whose results are available. We present the implementation outcomes of our design up to the third protection order.

Our click-based implementation is very competitive in terms of gate counting. Indeed, by employing the two-phase bundled-data protocol, instead of the dual-rail encoding, we present the smallest design — with a significant margin — among the low-latency solutions. The removal of the flip-flops formerly used to store inner-domain products combined with the withdrawn of the LSB and MSB registers contributes with the reduction of the overall silicon area of our masked AES S-box based on Galois-field arithmetic. Therefore, applying our technique to the AES S-box shown in [37], which relies on a similar approach to achieve higher efficiency compared to the Canright's proposal, could result in an even more compact and efficient design.

References [22] and [31] eliminate glitches as a result of a monotonic pre-charge / evaluate logic. However, the use of Muller cells may require engineering tricks in order operate correctly under conventional design flows, as it is not a default element in most technology libraries. Moreover, they rely on the DOM multiplier, which is designed to be secure even in the presence of glitches, diminishing the benefits of using complex glitch-free techniques. Furthermore, despite allowing to completely eliminate glitches, dual-rail circuits tend to be significantly larger and may be susceptible to exploitable side-channel leakages due to unbalanced routing [18] and early propagation effect [12].

A few low-latency solutions eliminate the need of fresh random masks. Arribas et al. [1] presents two low-latency TI without online randomness. In addition to the four input shares — which requires three random variables to mask the secret — to achieve first-order masking, the resulting implementation costs preclude its use in area-efficient scenarios. Furthermore, higher-order TI needs fresh random masks in order to be higher-order secure [27]. Similarly, the d^{th}-order "zero-latency" implementation, by Gross et al. [8], brings unpractical area costs and requires the duplication and random re-sharing of the inputs to avoid exploitable side-channel leakages due to collisions of shares.

TABLE I
PERFORMANCE FIGURES OF DIFFERENT MASKED S-BOX IMPLEMENTATIONS.

Design	Input Shares	Area [kGE]	Refresh [bits/cycle]	Latency [cycles]
First-order masked implementations				
Ueno et al. [37]	2	1.39	64	5
Wegener and Moradi [38]	4	4.20	0	16
Sugawara [34]	3	3.50	0	3
Gross et al. [9]	2	2.60	18	8
Gross et al. [8]	2	6.74	416	2
Gross et al. [8]	2	60.73	2048	1
Gross et al. [8]	2	17.83	0	0
Arribas et al. [1]	4	25.78	0	1
Arribas et al. [1]	4	58.41	0	1
Simões et al. [31]	2	6.10	36	1
Sasdrich et al. [29]	2	3.48	36	1
Nagpal et al. [22]	2	3.98	34	1
Nagpal et al. [22]	2	7.59	18	1
this work	2	1.64	36	1
Second-order masked implementations				
Gross et al. [8]	3	57.11	4446	2
Simões et al. [31]	3	11.40	108	1
Nagpal et al. [22]	3	9.34	102	1
Nagpal et al. [22]	3	14.78	51	1
this work	3	3.43	108	1
Third-order masked implementations				
this work	4	5.84	216	1

TABLE II
TIMING PERFORMANCE OF DIFFERENT LOW-LATENCY FIRST-ORDER MASKED AES128 ENCRYPTION.

Design	Technology	S-box Latency [cycles]	Frequency [MHz]
Gross et al. [8]	UMC 90nm	0	288
Gross et al. [8]	UMC 90nm	1	356
Gross et al. [8]	UMC 90nm	2	584
Sasdrich et al. [29]	UMC 90nm	1	400
Arribas et al. [1][a]	NanGate 45nm	1	277
Arribas et al. [1][b]	NanGate 45nm	1	40
Nagpal et al. [22]	UMC 65nm	1	192
Simões et al. [31]	STM 40nm	1	5
this work	STM 40nm	1	55

[a]Low-Latency Threshold Implementation
[b]Threshold Implementation

Our implementation requires $18(d^2+d)$ fresh random bits to achieve d^{th}-order masking, as the Canright's design has 36 AND gates [10]. The original DOM design in [9] uses half of the same amount as 18 AND gates are computed within a single combinatorial circuit between two register layers. Note that computing one AES round within a single cycle requires twenty times this amount of randomness — if the AES key schedule is masked.

We refer to Table II for the timing performance figures of several first-order low-latency masking implementation of the AES. Despite achieving secure single-cycle masking computation, the maximum frequency of our solution is lower compared to the low-latency masking techniques present in the state-of-the-art. Indeed, there are eight synchronization stages within our AES S-box, whose delays have to be matched correctly and with a security margin. Thus, applying the click-based masking to a more compact architecture with lower synchronization stages may improve the throughput.

As mentioned, despite adopting asynchronous techniques, a synchronous clock signal triggers the S-box computation. Thus, our AES S-box is processed within a single-cycle and allows the computation of the shift rows, mix columns and add round key functions during the same period. Therefore, we could classify our implementation as a "zero-latency" one, as defined in [8], since no globally-clocked registers are required once the AES S-box computation is started. This is not the case for the LLTI [1] and for the single-cycle masking with arbitrary protection order shown in [8].

Therefore, our S-box design allows the secure computation of an AES round within a single cycle. However, the clock period must match the circuit's critical path. Since we perform one round per cycle with locally clocked layers, the maximum operation frequency of our application is limited. For the first-order masking, our synthesized AES S-box design achieves a frequency of 55 MHz, while its synchronous counterpart can operate at 250 MHz. Considering the synthesis of both designs under the same technology, the maximum frequency is reduced by a factor of five, approximately.

Certainly, a traditional synchronous design may achieve higher clock frequency due to the smaller critical path. Nevertheless, our approach considers the local propagation time of a combinatorial block instead of using a global clock whose frequency is defined by the worst-case critical path. Moreover, eight clock cycles are needed to compute the synchronous S-box, while our design can output the correct byte in one clock cycle, which makes our solution more suitable in low-latency scenarios with an appropriate clock frequency.

From the click-based AES S-box, we implemented three d^{th}-order masked AES128 encryption variants: 8-bit, 32-bit and 128-bit serialized data path. We refer to Table III for the performance figures of our different AES128 encryption implementations.

The appropriate length of the bit serialized data path depends on the application. Thus, the current comparison aims at presenting the performance figures — in terms of area, fresh randomness and encryption latency — of a click-based AES architecture for different cases. We also present the Area $\times$ Latency product as a metric of the trade-off between both characteristics.

The 8-bit and 32-bit versions have an architecture similar to [19]. The 8-bit variant has only one S-box, which is also used during the key schedule, as well as the 32-bit case, which contains four S-boxes. The 128-bit variant performs a round function within one clock cycle and solves the sixteen SubBytes in parallel alongside with the key schedule, totaling twenty S-boxes. Therefore, the round keys used in the encryption are masked in all variants. Unmasking the key would improve the area and latency results, but we decide to maintain the key masked for benchmark reasons.

TABLE III
PERFORMANCE FIGURES OF OUR MASKED AES ENCRYPTION IMPLEMENTATIONS BASED ON CLICK ELEMENTS.

Data Path [bits]	**Area** [kGE]	**Refresh** [bits/cycle]	**Latency** [cycles]	**Area *vs* Latency** [GE×cycles×10^{-6}]
First-order masked implementations				
8	6.42	36	216	1.39
32	12.73	144	54	0.69
128	43.45	720	11	0.48
Second-order masked implementations				
8	10.64	108	216	2.30
32	22.77	432	54	1.23
128	83.82	2160	11	0.92
Third-order masked implementations				
8	15.35	216	216	3.32
32	35.25	864	54	1.90
128	136.92	4320	11	1.51

B. Toggle Rate

With the objective of gauging the power consumption of our approach compared to the clocked implementation, that is, the original design proposed in [9], we observe the toggle rate of the synchronous and click-based DOM of the Canright's AES S-box. We use a hundred thousand traces to compute the average signal toggle rate of the simulated devices during the computation of a SubBytes; thereupon, we integrate the resulting toggle activity to compute the mean, which gives us a rough estimation of the power consumption. The results are shown in Fig. 11.

Our first-order solution has a toggle rate ≈77% smaller than the synchronous implementation operating in pipeline mode. When the synchronous design operates without a pipeline, similar to our application, this gap is reduced to ≈ 57%. Indeed, besides operating without a pipeline, this synchronous design updates the random refresh bits every positive clock edge, which explains the difference in the toggle rate despite its similarity with the click-based design. Therefore, our asynchronous solution shows a lower toggle rate, which indicates a potential reduction in power consumption.

C. FPGA Implementation

We managed to set minimum path delays on the request channel using an ASIC synthesis flow with Synopsys Design Compiler. By adding a buffer manually and applying `dont_touch` constraint on it, we prevent the tool from modifying the buffer chains during optimization. However, we were not able to do the same for the FPGA implementation using Xilinx Vivado. To solve this issue, we describe the chain of buffers whose length is a user-defined parameter. We report the slack in the same manner to find the parameter values that meet the timing requirements. Fig. 12 shows a hardware description of a delay line used to implement a chain of buffers in an FPGA.

VI. SIDE-CHANNEL ANALYSIS

We simulate the signal switching activity of our low-latency AES S-box to evaluate its robustness against side-channel attacks.

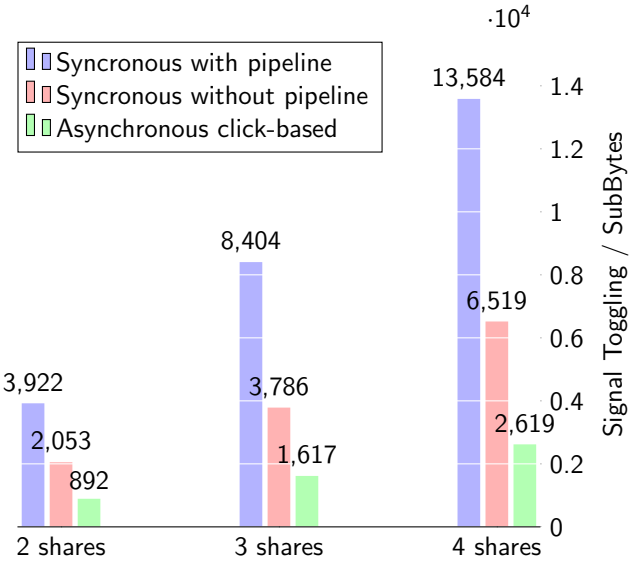

Fig. 11. Comparison of the toggle rate of the click-based and synchronous DOM implementations of the Canright's AES S-box.

```
module delay_line #(parameter LENGTH = 1)
   (input A, output Z);

   genvar i;
   (* dont_touch = "true" *) wire [LENGTH:0] channel;

   assign channel[0] = A;
   assign Z = channel[LENGTH];

   generate
      for (i = 0; i < LENGTH; i = i + 1) begin : BUFFER
         LUT1 #(.INIT(2'h2))
         LUT1_inst (.O(channel[i+1]), .I0(channel[i]));
      end
   endgenerate
endmodule
```

Fig. 12. Hardware description of a chain of buffers using LUTs.

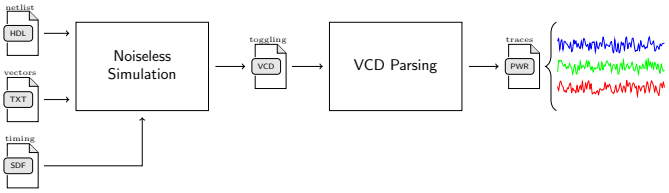

Fig. 13. Modeling power traces from VCD files generated after back-annotated simulation.

Our back-annotated analysis considers the standard cell timing characteristics in order to perform a realistic modeling, taking into account the occurrence of glitches. We use Mentor Graphics QuestaSim to perform logic simulations; the tool outputs VCD files which are parsed to obtain the toggle behavior of all internal wires of the masked circuit. This method allows us to model the system's side-channel behavior in a noiseless manner with a sampling frequency of 10 GHz. Fig. 13 shows the block diagram illustrating the acquisition of simulated traces used in our side-channel leakage assessment.

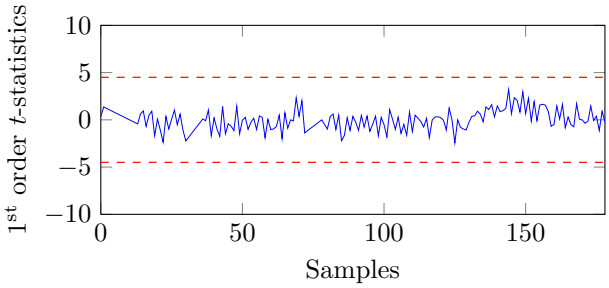

Fig. 14. First-order t-test results for the first-order masked AES S-box using ten million traces.

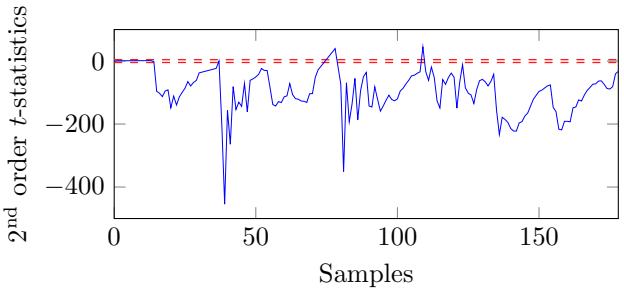

Fig. 15. Second-order t-test results for the first-order masked AES S-box using ten million traces.

We apply the *fixed vs random t-test* methodology[1] proposed by Goodwill et al. [6]. It uses the t-statistics, as expressed in (6), to determine whether the difference in the means of two distributions provides sufficient evidence to reject the null hypothesis. Thus, a distribution $\mathcal{Q}_0$ represents the set of traces corresponding to a random input and a second population $\mathcal{Q}_1$ groups the traces obtained after computing the pre-defined input.

$$t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{v_0}{n_0} + \frac{v_1}{n_1}}} \tag{6}$$

Where μ_i represents the expected value of the population $\mathcal{Q}_i$, while v_i denotes its variance and n_i the cardinality of the associated set. In short, a side-channel leakage can be potentially exploited during an attack when the resulting t-statistics exceeds a threshold of ± 4.5 [30].

A. Univariate analysis

In the univariate analysis, all samples are processed independently as the shares were manipulated in parallel and leaked at the same instant, which is typically the case for hardware designs. Fig. 14 shows the first-order t-test on the first-order masked AES S-box using ten million traces. No exploitable side-channel leakages were identified in the first-order analysis. Nevertheless, second-order leakages were spotted for this implementation, as we can see in Fig. 15. Both results were expected outcomes.

[1]We use SCALib for side-channel analysis: *github.com/simple-crypto/scalib*

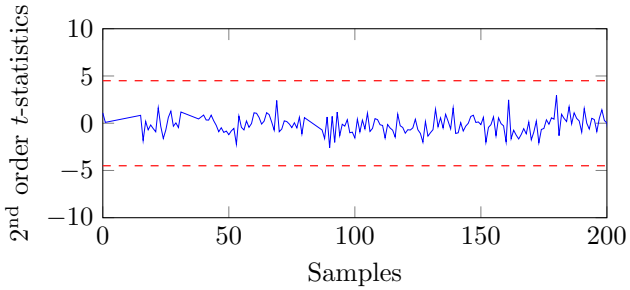

Fig. 16. Second-order t-test results for the second-order masked AES S-box using ten million traces.

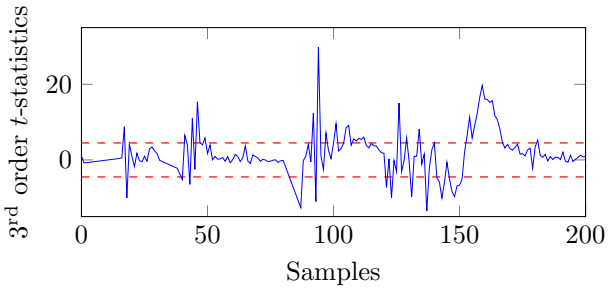

Fig. 17. Third-order t-test results for the second-order masked AES S-box using ten million traces.

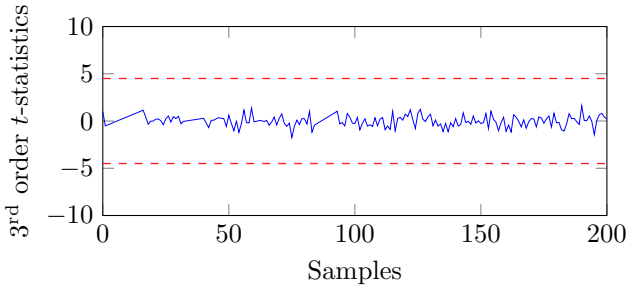

Fig. 18. Third-order t-test results for the third-order masked AES S-box using ten million traces.

To protect the system against second-order analysis, we increased the number of shares up to three to obtain a second-order masked implementation. Fig. 16 shows the second-order t-test on our second-order masked AES S-box using ten million traces. No exploitable side-channel leakages were detected, indicating the effectiveness of our implementation against higher-order attacks.

However, by increasing the order of the side-channel analysis to $d = 3$, exploitable leakages were detected, as shown in Fig. 17, which demonstrates the vulnerability of a second order masked design against third-order side-channel attacks.

We also perform the same third-order univariate analysis on our AES S-box masked with four shares, Fig. 18. As expected, no side-channel leakages were detected in this case, indicating that our single-cycle S-box is robust against higher-order attacks.

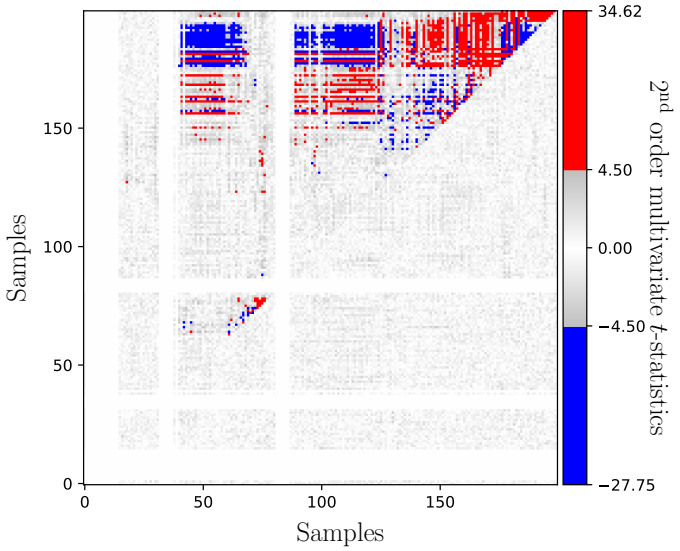

Fig. 19. Second-order bivariate analysis results for the second-order masked low-latency implementation of the AES S-box. The lower triangle shows the t-test results when the random mask refreshing is enabled. The upper triangle shows the t-test results when the random mask refreshing is turned off.

From these results, we can confirm that our approach does not weaken the DOM and can be potentially applied in different cryptographic systems in order to achieve single-cycle implementation without penalizing its security properties. We highlight that, the higher the masking order, the more complex it is to perform a successful side-channel attack in practice.

B. Bivariate analysis

The univariate setting considers the case in which the shares are processed in parallel, resulting in the sum of the share's exploitable side-channel leakage at the same sample point. However, if different shares leak at different spots, a univariate analysis is not capable of proving reliable side-channel leakage assessment. In this case, the different sample points of the set of traces have to be combined prior to the t-test. [30].

To ensure the robustness of our click-based approach, we also perform bivariate analysis on our second-order design. Fig. 19 shows the bivariate analysis for the second-order AES S-box using ten million traces. The upper triangle shows the results when the random mask refresh is disabled, while the lower triangle illustrates the side-channel analysis when fresh randomness is employed. The result obtained for the unprotected setting uses only 0.1% of the number of traces used in the protected scenario, that is, ten thousand against ten million traces.

The blue and red dots shown in Fig. 19 represent sample points in which the multivariate t-statistics exceeds the ± 4.5 threshold. As expected, without refreshing the random masks, exploitable leakages were detected in our design, confirming the need of online randomness to obtain a secure masking implementation. It also reaffirms that our click-based approach does not bring any weakness to the secure design, even in bivariate analysis.

VII. Conclusion

Low-latency masking is an important topic in secure hardware implementation. Indeed, given the growth of IoT applications, many embedded systems require cryptographic solutions offering lower latency and better area efficiency. In this context, this work presents the smallest masking implementation of a single-cycle AES S-box with arbitrary protection order to date.

To achieve low-latency, we rely on a click-based asynchronous design methodology, allowing hardware engineers to use a conventional synthesis flow to implement single-cycle cryptographic blocks that are secure against d^{th}-order side-channel analysis. Also, despite the potential low-power and high performance of asynchronous circuits, this design approach is not widely used in secure hardware applications yet. Therefore, this work also helps to bridge the gap between asynchronous circuit design and hardware masking schemes.

Compared to similar asynchronous low-latency solutions based on the dual-rail protocol, we achieve a smaller silicon area of the AES S-box by employing the bundled-data protocol. Furthermore, the logic depth is reduced, which may improve the overall performance of our design compared to these dual-rail implementations under the same technology and circuit logic. Also, bundled-data circuits resemble traditional synchronous design, which eases the application of this methodology by hardware security engineers familiar with already established synthesis flows.

Lower latency comes at the cost of larger critical path, which limits the maximal operation frequency of the system. However, instead of waiting several clock cycles to compute a secure function, our approach reduces the latency to a single clock while satisfying the necessary formal security properties. Additionally, one can potentially improve the throughput of the AES S-box by applying our click-based approach to a more compact design, with lower synchronization stages, which could also result in an even smaller implementation, depending on the chosen architecture.

We present the case study of the domain-oriented masking scheme to illustrate the methodology employed in this work. By relying on a known secure implementation, we aim at measuring the overheads of converting a synchronous design into an asynchronous circuit based on the two-phase bundled-data protocol. Moreover, we were also interested in assessing the leakages to evaluate the robustness of our click-based solution against side-channel attacks. Indeed, our implementation approach does not reduce the reliability of the originally synchronous design against exploits, indicating an efficient way to implement higher-order secure low-latency masking based on synchronous d^{th}-order protected implementations.

Compared to its synchronous counterpart, the overall toggle rate of our DOM implementation is reduced, which indicates a potential advantage in low power scenarios.

Finally, despite presenting an AES S-box use case, the proposed method can be applied to different cryptographic modules to improve latency.

References

[1] Victor Arribas, Zhenda Zhang, and Svetla Nikova. LLTI: low-latency threshold implementations. *IEEE Trans. Inf. Forensics Secur.*, 16:5108–5123, 2021.

[2] David Canright. A very compact s-box for AES. In Rao and Sunar [26], pages 441–455.

[3] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Wiener [39], pages 398–412.

[4] 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 Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):89–120, 2018.

[5] Alberto Ghiribaldi, Davide Bertozzi, and Steven M. Nowick. A transition-signaling bundled data NoC switch architecture for cost-effective GALS multicore systems. In Enrico Macii, editor, *Design, Automation and Test in Europe, DATE 13, Grenoble, France, March 18-22, 2013*, pages 332–337. EDA Consortium San Jose, CA, USA / ACM DL, 2013.

[6] Gilbert Goodwill, Benjamin Jun, Joshua Jaffe, and Pankaj Rohatgi. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing (NIAT) workshop*, volume 7, pages 115–136, 2011.

[7] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[8] Hannes Groß, Rinat Iusupov, and Roderick Bloem. Generic low-latency masking in hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):1–21, 2018.

[9] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In Begül Bilgin, Svetla Nikova, and Vincent Rijmen, editors, *Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016*, page 3. ACM, 2016.

[10] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[11] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Wiener [39], pages 388–397.

[12] Konrad J. Kulikowski, Mark G. Karpovsky, and Alexander Taubin. Power attacks on secure hardware based on early propagation of data. In *12th IEEE International On-Line Testing Symposium (IOLTS 2006), 10-12 July 2006, Como, Italy*, pages 131–138. IEEE Computer Society, 2006.

[13] Andrew J. Leiserson, Mark E. Marson, and Megan A. Wachs. Gate-level masking under a path-based leakage metric. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 580–597. Springer, 2014.

[14] Zhiyu Li, Yuhao Huang, Longfeng Tian, Ruimin Zhu, Shanlin Xiao, and Zhiyi Yu. A low-power asynchronous RISC-V processor with propagated timing constraints method. *IEEE Trans. Circuits Syst. II Express Briefs*, 68(9):3153–3157, 2021.

[15] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

[16] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In Rao and Sunar [26], pages 157–171.

[17] Gabriele Miorandi, Marco Balboni, Steven M. Nowick, and Davide Bertozzi. Accurate assessment of bundled-data asynchronous NoCs enabled by a predictable and efficient hierarchical synthesis flow. In *23rd IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC 2017, San Diego, CA, USA, May 21-24, 2017*, pages 10–17. IEEE Computer Society, 2017.

[18] Amir Moradi, Mario Kirschbaum, Thomas Eisenbarth, and Christof Paar. Masked dual-rail precharge logic encounters state-of-the-art power analysis methods. *IEEE Trans. Very Large Scale Integr. Syst.*, 20(9):1578–1589, 2012.

[19] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of AES. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer, 2011.

[20] Amir Moradi and Tobias Schneider. Side-channel analysis protection and low-latency in action — case study of PRINCE and Midori. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *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*, volume 10031 of *Lecture Notes in Computer Science*, pages 517–547, 2016.

[21] David E. Muller and W. S. Bartky. A theory of asynchronous circuits. In *Proceedings of an International Symposium on the Theory of Switching, April 1957, Part I*, volume XXIX of *the annals of the computation laboratory of Harvard University*, pages 204–243, Cambridge, MA, USA, 1959. Cambridge University Press.

[22] Rishub Nagpal, Barbara Gigerl, Robert Primas, and Stefan Mangard. Riding the waves towards generic single-cycle masking in hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4):693–717, 2022.

[23] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.

[24] Ad M. G. Peeters, Frank te Beest, Mark de Wit, and Willem C. Mallon. Click elements: An implementation style for data-driven compilation. In *16th IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC 2010, Grenoble, France, 3-6 May 2010*, pages 3–14. IEEE Computer Society, 2010.

[25] Thomas Popp and Stefan Mangard. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In Rao and Sunar [26], pages 172–186.

[26] Josyula R. Rao and Berk Sunar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*. Springer, 2005.

[27] Oscar Reparaz. A note on the security of higher-order threshold implementations. *IACR Cryptol. ePrint Arch.*, page 1, 2015.

[28] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

[29] 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.

[30] Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015.

[31] Mateus Simões, Lilian Bossuet, Nicolas Bruneau, Vincent Grosso, Patrick Haddad, and Thomas Sarno. Self-timed masking: Implementing first-order masked s-boxes without registers. *IACR Cryptol. ePrint Arch.*, page 641, 2022.

[32] Montek Singh and Steven M. Nowick. MOUSETRAP: ultra-high-speed transition-signaling asynchronous pipelines. In *19th International Conference on Computer Design (ICCD 2001), VLSI in Computers and Processors, 23-26 September 2001, Austin, TX, USA, Proceedings*, pages 9–17. IEEE Computer Society, 2001.

[33] Jens Sparsø and Steve Furber. *Principles of Asynchronous Circuit Design: A Systems Perspective*. Springer, 01 2001.

[34] Takeshi Sugawara. 3-share threshold implementation of AES S-box without fresh randomness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):123–145, 2019.

[35] Ivan E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, 1989.

[36] 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*, pages 246–251. IEEE Computer Society, 2004.

[37] Rei Ueno, Naofumi Homma, and Takafumi Aoki. A systematic design of tamper-resistant Galois-field arithmetic circuits based on threshold implementation with (d + 1) input shares. In *47th IEEE International Symposium on Multiple-Valued Logic, ISMVL 2017, Novi Sad, Serbia, May 22-24, 2017*, pages 136–141. IEEE Computer Society, 2017.

[38] Felix Wegener and Amir Moradi. A first-order SCA resistant AES without fresh randomness. In Junfeng Fan and Benedikt Gierlichs, editors, *Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings*, volume 10815 of *Lecture Notes in Computer Science*, pages 245–262. Springer, 2018.

[39] Michael J. Wiener, editor. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.

[40] Hui Wu, Zhe Su, Jilin Zhang, Shaojun Wei, Zhihua Wang, and Hong Chen. A design flow for click-based asynchronous circuits design with conventional EDA tools. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 40(11):2421–2425, 2021.