IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2024, No. 4, pp. 451–482. DOI:10.46586/tches.v2024.i4.451-482

# Robust but Relaxed Probing Model

Nicolai Müller[1] and Amir Moradi[2]

[1] Ruhr University Bochum, Horst Görtz Institute for IT Security, Bochum, Germany
firstname.lastname@rub.de
[2] Technische Universität Darmstadt, Darmstadt, Germany
firstname.lastname@tu-darmstadt.de

**Abstract.** Masking has become a widely applied and heavily researched method to protect cryptographic implementations against Side-Channel Analysis (SCA) attacks. The success of masking is primarily attributed to its strong theoretical foundation enabling it to formally prove security by modeling physical properties through so-called probing models. Specifically, the robust $d$-probing model enables us to prove the security for arbitrarily masked hardware circuits, manually or with the assistance of automated tools, even when considering the imperfect nature of physical hardware, including the occurrence of physical defaults such as glitches. However, the generic strategy employed by the robust $d$-probing model comes with a downside: It tends to over-conservatively model the information leakage caused by glitches meaning that the robust $d$-probing model considers glitches that can never occur in practice. This implies that in theory, an adversary could gain more information than she would obtain in practice. From a designer's perspective, this entails that (1) securely designed hardware circuits may need to be withdrawn due to potential insecurity under the robust $d$-probing model and (2) designs that satisfy the security requirements of the robust $d$-probing model may incur unnecessary overhead, such as increased circuit size or latency.
In this work, we refine the formal treatment of glitches within the robust $d$-probing model to address glitches more accurately within a formal adversary model. Unlike the robust $d$-probing model, our approach considers glitches based on the operations performed and the data processed, ensuring that only manifesting glitches are accounted for. As a result, we introduce the Robust but Relaxed (RR) $d$-probing model, a formal adversary model maintaining the same level of security as the robust $d$-probing model but without the overly conservative treatment of glitches. Leveraging our new model, we prove the security of LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL) gadgets, a class of physically secure gadgets reported as insecure based on the robust $d$-probing model. We provide manual proofs and automated security evaluations employing an updated version of PROLEAD capable of verifying the security of masked circuits under our new model.

**Keywords:** Side-Channel Analysis · Masking · Hardware · Robust Probing Model

## 1 Introduction

More than two decades after Kocher's seminal discovery [Koc96], the task of securing cryptographic hardware implementations against the pervasive threat of SCA attacks remains a formidable challenge. In this context, the masking countermeasure [CJRR99], based on encoding secret variables through multiple shares [Sha79], emerges as the most widely adopted and thoroughly comprehended technique, especially concerning its formal security assurances [ISW03, PR13, DDF14, DFS15]. In concrete terms, if the leakages of the shares achieve a sufficiently high noise level and are independent of each other,


Received: 2024-04-15 Accepted: 2024-06-15 Published: 2024-09-05

the number of required measurements (i.e. the complexity) of SCA attacks increases exponentially with the number of shares [CJRR99]. Masking aims to randomize every sensitive variable $X$ by splitting it into $d+1$ shares, denoted as $X^0, \ldots, X^d$, in a manner such that their sum[1] equals $X$. For instance, in the prevalent Boolean masking, where $X \in \mathbb{F}_2$, it holds that $X = X^0 \oplus \ldots \oplus X^d$, with all shares $X^i \in \mathbb{F}_2$ $(0 \leq i \leq d)$ following a uniform distribution. To assess the practical fulfillment of the aforementioned requirements concerning independence and noise, formal security models abstract the physical security of masking. The basic $d$-probing model introduced by Ishai et al. [ISW03] enables to validate if an adversary needs to observe at least $d$ intermediates, i.e. stable signals carried by physical wires, in a masked hardware circuit to make any inference about a secret variable. This is modeled by placing up to $d$ so-called probes, each recording one stable signal. In essence, evaluation under the $d$-probing model ensures the preservation of the independence property, while achieving a sufficient noise level is imperative for security even in the most realistic noisy leakage model [PR13]. However, despite the adequacy of the noise level, multiple case-studies have demonstrated that a scheme deemed $d$-probing secure may still fall short of providing the expected level of security against SCA attacks in practical scenarios [MPO05, MPO05]. This unexpected leakage arises from the asynchronous arrival of input signals at a combinational gate. In such scenarios, the gate may produce a transient and unintended change in its output signal before stabilizing, commonly known as a glitch [Rab96]. Therefore, if the inputs of a combinational gate toggle, it does not merely result in a single toggle from one stable output value to another, as presumed by the $d$-probing model. Instead, it may trigger a sequence of toggles caused by glitches before the output stabilizes. This pattern, which can be observed at the gate's output, subsequently propagates as an input signal to the following combinational logic while leaking not only the stable output of the gate but also additional information about the input signals, an aspect not accounted for by the $d$-probing model. Therefore, the robust $d$-probing model, as introduced in [FGP+18], extends the foundational $d$-probing model to encompass the additional information leaked due to glitches. In this model, glitch-extended probes are introduced to capture not only stable signals but also potential signals revealed by glitches. However, due to the computational hardness of checking for specific glitches, the model assumes a generic worst-case scenario. Thus, a glitch-extended probe on a wire can record all stable signals from the combinational logic contributing to the probed wire. This assumption enables the assessment of security under the robust $d$-probing model without requiring knowledge of the specific placement and routing of the final design. Consequently, security proofs under this model hold for arbitrary circuits, independent of their particular placement and routing. The simplicity and broad applicability of the robust $d$-probing model have led to its widespread adoption for evaluating masked hardware. It is integrated into various tools such as [BGI+18, KSM20, MM22, BMRT22], facilitating automatic assessment of security for masked implementations. However, the model adopts a worst-case scenario approach, accounting for all potential glitches, despite only a subset of these glitches manifesting in practical scenarios. This tendency towards over-conservatism implies that a masked circuit, potentially optimized for efficiency, may remain secure in real-world applications even if deemed insecure under the robust $d$-probing model. This disparity between robust probing security and real-world security becomes particularly evident when considering glitch-free circuits. For instance, in [MLM23], it was demonstrated that such circuits leak more information than assumed by the $d$-probing model. However, considering glitches in such circuits is irrelevant as no glitches can occur. Specifically, the study revealed that LMDPL is not secure under the robust probing model, while also confirming its security and noting the absence of an accurate model to evaluate such circuit types.

[1] By "sum" we refer to the result of the addition of all shares in the underlying finite field.

### 1.1 Our Contributions

In this work, we propose a refinement of the robust $d$-probing model, aiming to reduce its over-conservatism while maintaining an equivalent level of security assurance. Our contributions are as follows:

- We introduce the RR $d$-probing model, a formal adversary model that provides a more precise treatment of glitches compared to the robust $d$-probing model, while ensuring an equivalent level of security[2]. Therefore, we propose a probe-extension procedure that specifically accounts for glitches based on the performed operations and processed data. Consequently, our model exclusively address glitches that manifest within a hardware circuit in a formal manner.
- We close the lack of accurate and formal proofs when assessing the security of glitch-free circuits as pointed out in [MLM23]. In particular, within our model, we can formally prove the security of LMDPL [LMW14] as well as the insecurity of Self-Synchronized Masking (SESYM) [NGPM22].
- To automate the security evaluation within our refined model, we extended the functionality of the existing PROLEAD tool [MM22] with the evaluation of a provided gate-level netlist under our model. Our extension is available as an open-source contribution via GitHub[3].

## 2 Background

### 2.1 Notations

We denote all random variables, like $X \in \mathbb{F}_2$, by capital letters and sets, like $\mathbf{X}$, by bold capital letters. We use subscripts to denote specific elements within a set, e.g., we denote the elements of a set $\mathbf{X}$ with $|\mathbf{X}| = n$ as $X_0, \dots X_{n-1}$. When denoting functions, e.g. $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$, we use sans-serif fonts. Further, we use superscripts to denote individual shares of a shared variable. For example, a sensitive variable $X$ is shared as $X^0 \oplus \dots \oplus X^{d-1}$. Furthermore, the set of shares associated with a set of $n$ secret variables $\mathbf{X}$ is denoted as $Sh(\mathbf{X}) = [X_0^0, \dots X_0^{d-1}, \dots, X_{n-1}^0, \dots X_{n-1}^{d-1}]$.

### 2.2 Circuit Model

We model any stateful and deterministic circuit as a Directed Acyclic Graph (DAG) $\mathsf{G} = (\mathbf{V}, \mathbf{E})$ whose vertices in set $\mathbf{V}$ are atomic components (gates), embodying the underlying logic, and edges in set $\mathbf{E}$ are interconnections (wires), transmitting elements from $\mathbb{F}_2$ [ISW03]. In this work, we specify different types of gates as follows:

- Combinational gates of fan-in at most 2 and fan-out 1 compute a logic function on its inputs without any dependence on the prior states of inputs. Without loss of generality, we restrict the logic functions processed by combinational gates to $\{\mathsf{NOT}, \mathsf{AND}, \mathsf{NAND}, \mathsf{OR}, \mathsf{NOR}, \mathsf{XOR}, \mathsf{XNOR}\}$.
- Sequential gates of fan-in 1 and fan-out 1 compute a logic function on its current inputs but also on previous inputs. All sequential gates are clock-synchronized.
- Input gates of fan-in 0 and fan-in 1 write elements in $\mathbb{F}_2$ on a wire.

[2] With "equivalent level of security" we mean that both models capture all leakages caused by glitches that can physically manifest.

[3] https://github.com/ChairImpSec/PROLEAD

- Output gates of fan-in 1 and fan-in 0 read elements in $\mathbb{F}_2$ from a wire.

However, this circuit model does not apply for iterative circuits, i.e. circuits with the ability to evaluate the same gate within multiple clock cycles, and was, therefore, extended by the structural circuit model given in [CS21]. This model defines a structural circuit as a directed graph, without any combinational loop, whose vertices are structural gates and edges are structural wires, carrying different elements across various time instances (clock cycles). Hence, a structural wire $W$ is represented as a pair $(E, t)$, comprising a wire $E$ and a specific time instance $t$. According to Definition 4 in [CS21], the execution of a structural circuit for a set of clock cycles models the state of the circuit for the given set of clock cycles as well as the latency introduced by sequential gates.

## 2.3 Circuit Compiler

A circuit compiler encompasses three different algorithms $(\mathsf{CC}, \mathsf{ENC}, \mathsf{DEC})$, defined as follows [AIS18]:

- The deterministic circuit compilation algorithm $\mathsf{CC}$ expects a (structural) circuit $\mathsf{C}$ as input and returns a randomized (masked) (structural) circuit $\tilde{\mathsf{C}}$.
- The probabilistic encode algorithm $\mathsf{ENC}$ expects the set of unshared inputs $\mathbf{X}$ of $\mathsf{C}$ and returns the $d$-shared input $\tilde{\mathbf{X}} = Sh(\mathbf{X})$ of $\tilde{\mathsf{C}}$.
- The deterministic decode algorithm $\mathsf{DEC}$ expects the shared output $\tilde{\mathbf{Y}} = Sh(\mathbf{Y})$ of $\tilde{\mathsf{C}}$ and returns the set of unshared outputs $\mathbf{Y}$ of $\mathsf{C}$.

We note that the resulting $\tilde{\mathsf{C}}$ may include additional randomness gates with a fan-in 0 and fan-out 1, which produce a uniformly distributed random value in $\mathbb{F}_2$. When viewed as a structural circuit, all randomness gates in $\tilde{\mathsf{C}}$ generate a fresh uniformly distributed random value per clock cycle.

## 2.4 Adversary Model

In this work, we focus on passive adversaries possessing the capability to observe a masked circuit $\tilde{\mathsf{C}}$. This implies that the adversary lacks direct access to secret data $\mathbf{X}$ or $\mathbf{Y}$, even if the underlying circuit compiler is public. Initially, we assume that $\tilde{\mathsf{C}}$ is modeled as a DAG as introduced in Subsection 2.2 while we introduce all relevant definitions for structural circuits when necessary.

### 2.4.1 $d$-probing Model

The basic $d$-probing model [ISW03] defines the $d$-probing adversary with the ability to observe up to $d$ intermediate elements processed by $\tilde{\mathsf{C}}$. In the following, we denote the set of observations made by an adversary as $\mathbf{Q}$. The observation is abstracted by permitting an adversary to place up to $d$ standard probes, as formalized in Definition 1, on arbitrary wires of $\tilde{\mathsf{C}}$.

**Definition 1** (Standard Probe)**.** Let $\tilde{\mathsf{C}}$ be a masked circuit with $\mathsf{G} = (\mathbf{V}, \mathbf{E})$. A standard probe $P_E \in \mathbb{F}_2$ on wire $E \in \mathbf{E}$ symbolically represents the element in $\mathbb{F}_2$ carried by wire $E$.

Intuitively, $\tilde{\mathsf{C}}$ achieves $d$-probing security if no $d$-probing adversary can extract secret (unshared) data by observing the joint distribution of at most $d$ intermediate elements. Formally, we define $d$-probing security based on the concept of statistical independence. Let $\mathbf{A}$ and $\mathbf{B}$ be two sets of discrete random variables. Then, $\mathbf{A}$ is statistically independent of $\mathbf{B}$ iff it holds that $Pr[\mathbf{A}|\mathbf{B}] = Pr[\mathbf{A}]$.

**Definition 2** ($d$-Probing Security)**.** A masked circuit $\tilde{\mathsf{C}}$ with secret input $\mathbf{X}$ is $d$-probing secure iff for any set of standard probes $\mathbf{P}$ with $|\mathbf{P}| \leq d$, the joint distribution over all observations $\mathbf{Q}$ made by the standard probes in $\mathbf{P}$ is statistically independent of $\mathbf{X}$.

### 2.4.2 Robust $d$-Probing Model

The robust $d$-probing model [FGP+18] extends the $d$-probing model by incorporating the impact of a limited set of physical defaults, namely combinational recombinations (glitches) [MPG05], memory recombinations (transitions) [CGP+12], and routing recombinations (couplings) [CBG+17], on the formal security of $\tilde{\mathsf{C}}$. The robust $d$-probing model introduces a novel class of probes, referred to as extended probes, for each of the three discussed physical defaults. These extended probes not only observe the elements carried by the wires they are placed on but also observe elements carried by other wires. Hence, every extended probe symbolically represents a set of standard probes.

**Glitch-Extended Probe.** A Glitch-extended probe models the fact that when measuring the signal driven by a physical wire we do not just see the stable signal immediately. Instead, we observe a sequence of transient toggles before the signal stabilizes. Since glitches are data-dependent, observing such a sequence can yield information about multiple signals connected to the observed wire through combinational logic. In the robust-probing model, it is assumed conservatively that glitches might enable an adversary to access all signals that contribute to the computation of the observed wire via combinational logic. Consequently, a glitch-extended probe on wire $E$, denoted as $P_E^{\text{ext}}$, is converted into a set of standard probes placed on all primary inputs and register outputs that contribute to $E$. As shown in [KM22], we formalize the glitch-extension procedure with function name `glitch_extend` in Algorithm 1.

**Algorithm 1** Glitch Extension (`glitch_extend`)

**Input:** $P_E^{\text{ext}}$ ▷ Glitch-extended probe on wire $E$
**Output:** $\mathbf{P}$ ▷ The set of standard probes
1: **if** $P_E^{\text{ext}}$ is placed on an output of a combinational gate **then**
2: $\quad\mathbf{P} \leftarrow \bigcup_{i=0}^{n}$ `glitch_extend`$(P_i^{\text{ext}})$ ▷ $P_i^{\text{ext}}$ with $0 \leq i < n$ denotes glitch-extended probes on all inputs of the combinational gate
3: **else**
4: $\quad$**if** $P_E^{\text{ext}}$ is placed on an output of a sequential gate or a primary input **then**
5: $\quad\quad\mathbf{P} \leftarrow \{P_E\}$
6: $\quad$**end if**
7: **end if**

**Transition-Extended Probe.** A transition-extended probe models the dependence of the consumed energy on both the preceding and subsequent elements when overwriting a wire. Consequently, a transition-extended probe placed on a structural wire $W = (E, t)$ observes a pair of elements carried by wire $E$ during clock cycles $t - 1$ (the preceding element) and $t$ (the subsequent element)[CS21]. Similarly, the transition-extension procedure transforms a transition-extended probe on wire $W = (E, t)$ into two standard probes. These standard probes separately represent the elements carried by $e$ during clock cycles $t$ and $t - 1$. For simplicity, we omit the specific value of $t$ if it is not relevant and refer to the result of a transition-extension procedure as $(P_{E'}, P_E)$. Here, $P_{E'}$ denotes a standard probe on wire $E$ observing the preceding element, while $P_E$ observes the subsequent element. While glitches and transitions are formally treated as distinct phenomena, it's noteworthy that glitch-extended probing, in isolation from transitions, oversimplifies the situation, given

that glitches are a direct consequence of transitions. Consequently, we advocate for a holistic approach that incorporates both glitches and transitions when employing formal adversary models. To consider glitches and transition simultaneously, both extended probes can be merged resulting in the extension procedure `robust_extend` outlined in Algorithm 2. Now, a so-called robust probe on wire $E$, denoted as $P_E^{\text{robust}}$, is converted into a set of standard probes recording two consecutive states of all primary inputs and register outputs that contribute to $E$. First, the glitch-extension procedure is performed as shown in Algorithm 1 until the probe cannot be further extended based on glitches. Afterwards, in the result of the glitch extension becomes transition-extended (cf. Line 5) by storing standard probes on the respective wire's preceding and subsequent element.

**Algorithm 2** Robust (Glitch and Transition) Extension (`robust_extend`)

**Input:** $P_E^{\text{robust}}$ ▷ Robust probe on wire $E$
**Output:** $\mathbf{P}$ ▷ The set of standard probes
1: **if** $P_E^{\text{robust}}$ is placed on an output of a combinational gate **then**
2: $\mathbf{P} \leftarrow \bigcup_{i=0}^{n}$ `robust_extend`$(P_i^{\text{robust}})$ ▷ $P_i^{\text{robust}}$ with $0 \leq i < n$ denotes robust probes on all inputs of the combinational gate
3: **else**
4: **if** $P_E^{\text{robust}}$ is placed on an output of a sequential gate or a primary input **then**
5: $\mathbf{P} \leftarrow \{P_{E'}\} \cup \{P_E\}$ ▷ Do the transition extension
6: **end if**
7: **end if**

We remark that we do not consider couplings in this work as the circuit models discussed in Subsection 2.2 do not incorporate any information regarding placement and routing. Consequently, in the variant of the robust $d$-probing model that considers glitches and transitions but excludes couplings, which we refer to as the robust probing model for the rest of this work, an adversary, denoted as a robust $d$-probing adversary, can place a maximum of $d$ robust probes on arbitrary wires within $\tilde{\mathsf{C}}$.

**Definition 3** (Robust $d$-Probing Security)**.** A masked circuit $\tilde{\mathsf{C}}$ with secret input $\mathbf{X}$ is robust $d$-probing secure iff for any set of robust probes $\mathbf{P}$, with $|\mathbf{P}| \leq d$, the joint distribution over all observations $\mathbf{Q}$ made by the robust probes is statistically independent of $\mathbf{X}$.

### 2.4.3 Composability

However, finding a compiler that transforms a circuit $\mathsf{C}$ into an efficient and robust $d$-probing secure masked circuit $\tilde{\mathsf{C}}$ as well as verifying the $d$-probing security of $\tilde{\mathsf{C}}$ poses a challenge, especially if $\mathsf{C}$ is complex and $d$ is high. Therefore, composable gadgets were introduced in [CGLS21] to establish a standard compiler. This compiler replaces every gate of $\mathsf{C}$ with its corresponding gadget. Each gadget itself is a robust $d$-probing secure circuit, implementing the functionality of a basic logic gate. Furthermore, these gadgets achieve composability under a certain composability notion implying robust $d$-probing security, even when multiple gadgets are arbitrarily composed to form a larger circuit. Before introducing the composability notion employed in this work, we first introduce the concept of probe simulatability [CS20]. This concept enables us to formally reason about the dependencies between (robust) probes and input shares to $\tilde{\mathsf{C}}$.

**Definition 4** (Perfect Probe Simulation)**.** Let $\mathbf{P} \in \mathbb{F}_2^t$ be a set of robust probes on a masked circuit $\tilde{\mathsf{C}}$ and $\mathbf{S} \subset \tilde{\mathbf{X}}$ be a set of input shares. Further, let $\mathsf{SIM}(\mathbf{S}) : \mathbb{F}_2^{|\mathbf{S}|} \rightarrow \mathbb{F}_2^t$ be a probabilistic polynomial time simulator. $\mathbf{P}$ is perfectly simulatable by a set $\mathbf{S}$ iff there exist a simulator $\mathsf{SIM}$, such that for any values of the inputs to $\tilde{\mathsf{C}}$, the joint probability distribution over $\mathbf{P}$ and $\mathsf{SIM}(\mathbf{S})$ are equal.

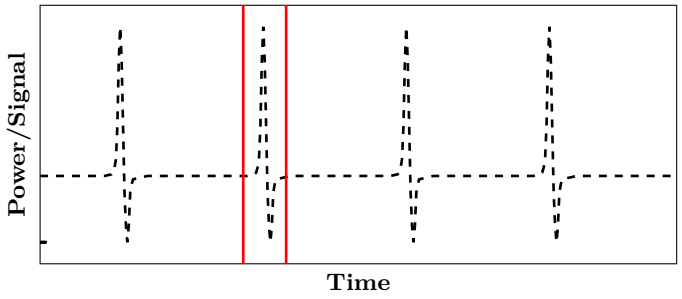
Now, let $P \in \mathbb{F}_2$ be a probe on a masked circuit (gadget) $\tilde{\mathsf{C}}$. We say that $P$ propagates into an input of $\tilde{\mathsf{C}}$ if the input is required to perfectly simulate $P$. Consequently, the concept of probe propagation defines the information leaked by a probe and how this leakage is propagated throughout the circuit [CS20]. To ensure the composability of a gadget, the Probe-Isolating Non-Interference (PINI) composability notion [CS20] constrains the probe propagation of all probes on outputs to its own share domain. Similarly, it limits the probe propagation of probes on internal wires of the gadget to a single but arbitrary share domain. The concept of share domains is defined in [GMK16], while Definition 5 formally introduces the PINI notion.

**Definition 5** (Robust $d$-Probe-Isolating Non-Interference (PINI))**.** Let $\tilde{\mathsf{C}}$ be a masked circuit with secret input $\mathbf{X}$. Further, let $\mathbf{P}_\mathrm{I}$ be a set of robust probes on internal wires of $\tilde{\mathsf{C}}$ with $|\mathbf{P}_\mathrm{I}| = t_1$ and $\mathbf{I}_\mathrm{O}$ be the set of indices assigned to the robust probes on output wires in the set $\mathbf{P}_\mathrm{O}$ with $|\mathbf{I}_\mathrm{O}| = t_2$. $\tilde{\mathsf{C}}$ is robust $d$-PINI iff for every $\mathbf{P} = \mathbf{P}_\mathrm{I} \cup \mathbf{P}_\mathrm{O}$ with $t_1 + t_2 \leq d$, there exists a set of indices $\mathbf{I}_\mathrm{I}$ with $|\mathbf{I}_\mathrm{I}| \leq t_1$ such that $\mathbf{P}$ is perfectly simulatable from $\mathbf{S} = \hat{\mathbf{X}}^{\mathbf{I}_\mathrm{I} \cup \mathbf{I}_\mathrm{O}}$.

## 3 Dynamic Power Consumption of CMOS Circuits

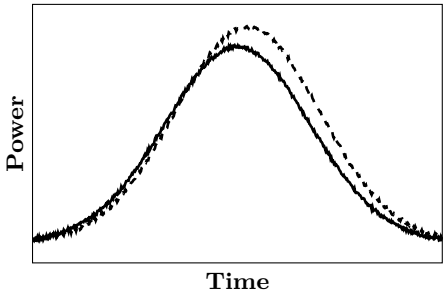
Following the formal introduction to how glitches are modeled within the robust $d$-probing model, it becomes crucial to elaborate on why this model accurately reflects the physical reality regarding glitches. To illustrate this, we examine the dynamic power consumption of a Complementary Metal-Oxide-Semiconductor (CMOS) circuit, as schematically depicted in Figure 1a. In CMOS circuits, synchronization is achieved through a clock signal, with all sequential gates evaluating with every (rising or/and falling) edge of the clock signal. Consequently, the dashed signal in Figure 1a exhibits distinct power peaks corresponding to each rising edge of the clock, with each peak dependent on the power consumption of the combinational logic. Notably, every change at a CMOS gate contributes to the amount of dynamic power consumption. Hence, glitches have a significant impact on this amount. To further elaborate this, we consider two exemplary patterns of glitches as visualized in Figure 1b which can be the result of two distinct transitions at the input of a combinational circuit, i.e., at the registers which provide such inputs.

Clearly, distinguishing between these patterns is straightforward when the entire patterns are observable. Additionally, it can be assumed that each transition at the input of the combinational logic yields a unique glitch "fingerprint", manifested as a pattern of glitches in the outputs of the combinational logic. Consequently, observing a pattern of glitches may reveal all inputs of the combinational logic. One might argue that exploiting such a pattern necessitates an SCA attack that combines a sequence of leakage samples, namely a multivariate attack [GBPV10], which can be challenging to mount in practice. However, it is important to note that, in the context of power consumption, the measured power is filtered by a low-pass filter [MOP07]. The limited bandwidth of this filter implies that obtaining distinct power signals for logic cells that switch within a few nanoseconds, such as those related to glitches, is not feasible. We sketched the power consumption peaks associated with the shown glitch patterns in Figure 1c. Notably, these peaks encompass the sum of all consumed power within a single clock cycle, i.e. all glitches occurring within one clock cycle, their shape, and their duration contribute to the peak. Consequently, a single leakage sample in a measured power trace is influenced by the switching activities of cells within a certain time interval shorter than a full clock cycle [MOP07]. Therefore, it is possible to find at least a single leakage sample to differentiate these power consumption peaks. Hence, the whole information leaked by a pattern of glitches becomes observable in a single leakage sample effectively making univariate first-order attacks applicable [MPO05]. The robust $d$-probing model operates under the premise that each input state of the combinational logic results in a distinctive




**(a)** The clock signal of a CMOS circuit (solid line) together with its corresponding dynamic power consumption trace (dashed line).


**(b)** Two exemplary glitch patterns occurring within one clock cycle.




**(c)** The power consumption peaks associated with these glitch patterns.

**Figure 1:** The dynamic power consumption of a CMOS circuit.

power consumption peak and that these peaks can be distinguished by a single leakage sample. In particular, it acknowledges that observing a single leakage sample may provide information on all inputs of the combinational logic. It achieves this by defining robust probes that grant access to additional standard probes across all these inputs.

# 4 Technique

Let $\tilde{\mathsf{C}}$ be a masked circuit with secret input $\mathbf{X}$ and $P_E^{\text{robust}}$ be a robust probe on an arbitrary wire $E$ of $\tilde{\mathsf{C}}$. According to Algorithm 2, if $E$ is the output of a combinational gate $P_E^{\text{robust}}$ gets substituted by robust probes on all input wires of the combinational gate. However, if $E$ is a primary input or the output of a sequential gate, $P_E^{\text{robust}}$ is not further extended but substituted by two standard probes $P_{E'}$ and $P_E$ on two consecutive clock cycles.

Line 1 of Algorithm 2 highlights a significant disparity between the treatment of glitches in the robust probing model and their manifestation in physical reality. The robust $d$-probing model relies on the conservative assumption that a probe on the output of a combinational gate extends to all inputs of such a gate independent of the operation performed by the combinational gate and independent of the processed data. However, the physical manifestation of glitches is significantly influenced by both the operation executed by the gate and the data processed by it. We illustrate this in Example 1. In particular, we show one scenario where the operation performed by the gate together with the processed data leads to a stop of the glitch-propagation.

**Example 1.** Consider an AND-gate with two inputs $X$ and $Y$ and output $Z$. If $X$ (resp. $Y$) is constantly set to zero, a glitch on $Y$ (resp. $X$) cannot lead to a glitch on $Z$. This means that the glitch-propagation stops at this gate without the need for a sequential gate.

Moreover, if $X$ (resp. $Y$) is constantly set to zero, an adversary who observes $Z$ being constantly zero cannot learn $Y$ (resp. $X$) while a robust $d$-probing adversary gets access to two robust probes $P_X^{\text{robust}}$ and $P_Y^{\text{robust}}$ by placing one robust probe $P_Z^{\text{robust}}$.

To investigate the data- and operation-dependent propagation of glitches in more detail, we examine all possible transitions at the inputs of AND- and OR-gates with inputs $X$ and $Y$ and output $Z$. We assume that both gates are in a stable state, i.e. no glitches on $X$ and $Y$, before switching to another state. In Table 1, we show all possible transitions, denoted as a set $\mathbf{S}$, visualized by its corresponding signal waveforms. While our discussion focuses solely on specific AND- and OR-gates, we provide similar tables for other combinational gates with fan-in two in the Appendix.

**Table 1:** Signal waveforms of AND- and OR-gates.

| **S** | $X$ | $Y$ | $Z$ |
|---|---|---|---|
| $S_0$ | | | |
| $S_1$ | | | |
| $S_2$ | | | |
| $S_3$ | | | |
| $S_4$ | | | |
| $S_5$ | | | |
| $S_6$ | | | |
| $S_7$ | | | |
| $S_8$ | | | |
| $S_9$ | | | |
| $S_{10}$ | | | |
| $S_{11}$ | | | |
| $S_{12}$ | | | |
| $S_{13}$ | | | |
| $S_{14}$ | | | |
| $S_{15}$ | | | |

**(a)** Signal waveforms of an AND-gate.

| **S** | $X$ | $Y$ | $Z$ |
|---|---|---|---|
| $S_0$ | | | |
| $S_1$ | | | |
| $S_2$ | | | |
| $S_3$ | | | |
| $S_4$ | | | |
| $S_5$ | | | |
| $S_6$ | | | |
| $S_7$ | | | |
| $S_8$ | | | |
| $S_9$ | | | |
| $S_{10}$ | | | |
| $S_{11}$ | | | |
| $S_{12}$ | | | |
| $S_{13}$ | | | |
| $S_{14}$ | | | |
| $S_{15}$ | | | |

**(b)** Signal waveforms of an OR-gate.

## 4.1 Data-Dependent Probe-Extension of an AND-gate

Initially, we investigate Table 1a and discuss the information gained by observing the waveform associated with $Z$

*Case* 1. For $S_0$-$S_4$, $S_8$, and $S_{12}$ $X$ or $Y$ remains at a constant value of 0. These cases are similar to Example 1 and lead to a signal waveform for $Z$ that is constantly 0. In particular, if $X$ (resp. $Y$) is constantly 0 then $Z$ is constantly 0 independent of any toggle on $Y$ (resp. $X$). Hence, these observations hold true independent of any placement and routing considerations.

*Case* 2. For $S_{15}$ $X$ and $Y$ remain at a constant value of 1. Hence, due to the underlying logic of the AND-gate, $S_{15}$ is the only case where $Z$ is constantly 1. Again, this holds true independent of any placement and routing considerations.

*Case* 3. For $S_5$, $S_7$, $S_{10} - S_{11}$, and $S_{13} - S_{14}$ either $X$ and $Y$ both toggle from the same stable state to another state state (see $S_5$ and $S_{10}$) or only $X$ (resp. $Y$) toggles while $Y$ (resp. $X$) remains at a constant value of 1. Consequently, the waveform of $Z$ shows the same toggle as the waveform of at least one of the inputs. As already indicated in [MLM23] the exact time of toggle on $X$ (resp. $Y$) and $Z$ depends on the delays of the

wires and the AND-gate itself. This means that the observation of a single toggle on $Z$ is not influenced by placement and routing while the exact time of toggle depends on the particular routing of the wires.

*Case* 4. For $S_6$ and $S_9$ $X$ and $Y$ both toggle inversely, potentially resulting in a glitch on $Z$. Specifically, in $S_6$, a glitch arises only if $X$ toggles prior to $Y$, whereas in $S_9$, a glitch arises only if $Y$ toggles before $X$. However, if these conditions are not satisfied, no toggles occur on $Z$. Consequently, the presence and timing of glitches are profoundly affected by placement and routing considerations.

We reiterate that, concerning a single AND-gate, the robust $d$-probing model assumes that a robust probe $P_Z^{\text{robust}}$ is replaced by four standard probes $P_{X'}$, $P_X$, $P_{Y'}$, and $P_Y$, indicating that an adversary can learn two consecutive and stable states of all inputs just by observing $Z$. This conservative assumption is justified by the scenarios pointed out in *Case* 4. If an adversary observes a glitch at $Z$, it becomes evident that we are either in scenario $S_6$ or $S_9$. Furthermore, the adversary can distinctly differentiate between $S_6$ and $S_9$ based on the specific characteristics of the glitch, such as its timing and duration. We note that Table 1 simplifies the visualization by disregarding any potential propagation delays. In reality, however, glitches observed in scenarios like $S_6$ and $S_9$ may exhibit variations. Likewise, an adversary may be able to distinguish between the scenarios highlighted in *Case* 3 by analyzing the precise timing of the toggles observed at $Z$, whereas the scenario outlined in *Case* 1 remains inherently unique. This alignment of our observations underscores the validity of the robust $d$-probing model.

However, the assumption made by the robust $d$-probing model is too conservative regarding the scenarios associated with *Case* 1. In these scenarios, an adversary cannot observe any toggle on $Z$. Specifically, no timing-related information is transmitted from $X$ or $Y$ to $Z$, rendering it impossible to differentiate between the scenarios outlined in *Case* 1. Therefore, an adversary observing $Z$ can only learn two consecutive states of $Z$, without being able to extract any additional information, about $X$ and $Y$.

Based on the aforementioned observations, we propose modifying the probe-extension methodology based on the processed data and the functionality of the AND-gate. Specifically, we suggest stopping the probe-extension for all scenarios outlined in *Case* 1, while retaining the same probe-extension approach as the robust $d$-probing model for all other cases. Therefore, we define a so-called probe-extension function $\mathsf{F}_Z^{\mathsf{AND}} : \mathbb{F}_2^4 \to \mathbb{F}_2$ for the output $Z$ of an AND-gate over its consecutive input states $X', X, Y', Y \in \mathbb{F}_2$ as given in Equation 1.

$$\mathsf{F}_Z^{\mathsf{AND}}(X', X, Y', Y) = (X' \vee X) \wedge (Y' \vee Y) \tag{1}$$

Further, we denote a variable indicating whether a robust probe on $Z$ should be extended or not as $F_Z \in \mathbb{F}_2$ with $F_Z = \mathsf{F}_Z^{\mathsf{AND}}(X', X, Y', Y)$. Equation 1 does only return 0 if one of the inputs stays at constant 0, i.e. $X' = 0$ and $X = 0$ or $Y' = 0$ and $Y = 0$, otherwise Equation 1 returns 1. Hence, if a robust probe $P_Z^{\text{robust}}$ is placed on $Z$ we propose to place robust probes $P_X^{\text{robust}}$ and $P_Y^{\text{robust}}$ on both inputs $X$ and $Y$ if $F_Z = 1$. Otherwise, if $F_Z = 0$, we propose to stop the probe-extension and place two standard probes $P_{Z'}$ and $P_Z$ recording two consecutive states of $Z$.

## 4.2 Data-Dependent Glitch-Propagation of an AND-Gate

So far, we disregarded the possibility of glitches on $X$ and $Y$ in Table 1a meaning that we assume that the inputs of AND-gates are always free of glitches. While this assumption holds true for stable inputs, such as primary inputs or register outputs which are directly forwarded to an AND-gate, we acknowledge that glitches can occur within a combinational circuit and propagate to other combinational gates. For example, an AND-gate can receive the glitchy output of another AND-gate as input. Thus, we must adjust $\mathsf{F}_Z^{\mathsf{AND}}$ to account for glitches on $X$ and $Y$. Therefore, we define a so-called glitch-propagation

function $\mathsf{G}_Z^{\mathsf{AND}} : \mathbb{F}_2^6 \to \mathbb{F}_2$ for the output $Z$ of an AND-gate over its consecutive input states $X', X, Y', Y \in \mathbb{F}_2$ and the information whether $X$ or $Y$ are already glitchy. We denote the variable indicating whether a wire $X$ is glitchy or not as $G_X \in \mathbb{F}_2$ and assume $G_X = 0$ if $X$ is a primary input or the output of a sequential gate. Otherwise, it holds that $G_X = \mathsf{G}_X^{\mathsf{AND}}$, i.e. the presence of glitches on $X$ must be computed based on the inputs of the gate that calculates $X$. Below, we give $\mathsf{G}_Z^{\mathsf{AND}}$ as well as a revised version of $\mathsf{F}_Z^{\mathsf{AND}} : \mathbb{F}_2^6 \to \mathbb{F}_2$ accounting for glitches on $X$ and $Y$.

$$\begin{aligned}\mathsf{F}_Z^{\mathsf{AND}}(X', X, G_X, Y', Y, G_Y) &= (X' \vee X \vee G_X) \wedge (Y' \vee Y \vee G_Y)\\ \mathsf{G}_Z^{\mathsf{AND}}(X', X, G_X, Y', Y, G_Y) &= (X' \wedge \bar{X} \wedge \bar{Y}' \wedge Y) \vee (\bar{X}' \wedge X \wedge Y' \wedge \bar{Y})\\ &\quad \vee (G_X \wedge (Y' \vee Y \vee G_Y)) \vee (G_Y \wedge (X' \vee X \vee G_X))\end{aligned}$$

In contrast to the initial version of $\mathsf{F}_Z^{\mathsf{AND}}$, the revised version integrates the additional constraint that a signal remains constantly zero only if it is free of glitches. Adhering to a conservative approach, we acknowledge that every glitch occurring on $X$ or $Y$ could potentially provide information to an adversary probing $Z$. Consequently, the revised version of $\mathsf{F}_Z^{\mathsf{AND}}$ indicates a probe extension for every glitch detected on $X$ or $Y$. We adopted the same conservative approach in defining $\mathsf{G}_Z^{\mathsf{AND}}$. The first two monomials of $\mathsf{G}_Z^{\mathsf{AND}}$ address scenarios where a glitch on $Z$ can occur even if $X$ and $Y$ are free of glitches (see $S_6$ and $S_9$ in Table 1a). However, if $X$ (or $Y$) is glitchy, we assume that the glitch propagates to $Z$ unless $Y$ (or $X$) is constantly zero and free of glitches.

## 4.3 Data-Dependent Probe-Extension of an OR-gate

We continue by examining Table 1b to derive $\mathsf{F}_Z^{\mathsf{OR}}$ and $\mathsf{G}_Z^{\mathsf{OR}}$, which represent a probe-extension function and a glitch-extension function respectively. These functions aim to model the probe extension through an OR-gate in a less conservative manner compared to the robust $d$-probing model.

*Case* 1. For $S_7$, and $S_{11}$-$S_{15}$, $X$ or $Y$ remains at a constant value of 1. These scenarios lead to a signal waveform for $Z$ that is constantly 1. In particular, if $X$ (resp. $Y$) is constantly 1 then $Z$ is constantly 1 independent of any toggle on $Y$ (resp. $X$). Hence, these observations hold true independent of any placement and routing considerations.

*Case* 2. For $S_0$ $X$ and $Y$ remain at a constant value of 0. Hence, due to the underlying logic of the OR-gate, $S_0$ is the only case where $Z$ is constantly 0. Again, this holds true independent of any placement and routing considerations.

*Case* 3. For $S_1 - S_2$, $S_4 - S_5$, $S_8$, and $S_{10}$ either $X$ and $Y$ both toggle from the same stable state to another state state (see $S_5$ and $S_{10}$) or only $X$ (resp. $Y$) toggles while $Y$ (resp. $X$) remains at a constant value of 0. Consequently, the waveform of $Z$ shows the same toggle as the waveform of at least one of the inputs. The observation of a single toggle on $Z$ is not influenced by placement and routing while the exact time of toggle depends on the particular routing of the wires.

*Case* 4. As for the AND-gate, for $S_6$ and $S_9$ $X$ and $Y$ both toggle inversely, potentially resulting in a glitch on $Z$.

The observations regarding the OR-gate closely resemble the findings discussed earlier concerning the AND-gate. Specifically, for the scenarios outlined in *Case* 1, the assumptions made by the robust $d$-probing model are too conservative while they are valid for *Cases* 2-4. Therefore, the probe-extension function $\mathsf{F}_Z^{\mathsf{OR}} : \mathbb{F}_2^4 \to \mathbb{F}_2$ for the output $Z$ of an OR-gate over its consecutive input states $X', X, Y', Y \in \mathbb{F}_2$ must be defined in a way that the probe-extension stops all all scenarios outlined in *Case* 1 and otherwise continues. We show a possible equation for $\mathsf{F}_Z^{\mathsf{OR}}$ in Equation 2 under the assumption that there are no glitches on $X$ or $Y$.

$$\mathsf{F}_Z^{\mathsf{OR}}(X', X, Y', Y) = (\bar{X}' \vee \bar{X}) \wedge (\bar{Y}' \vee \bar{Y}) \tag{2}$$

Contrary to Equation 1, Equation 2 does only return 0 if at least one of the inputs stays at constant 1, i.e. $X' = 1$ and $X = 1$ or $Y' = 1$ and $Y = 1$. For all other input vectors, Equation 2 returns 1 indicating that a robust probe $P_Z^{\text{robust}}$ must be extended to two robust probes $P_X^{\text{robust}}$ and $P_Y^{\text{robust}}$.

### 4.4 Data-Dependent Glitch-Propagation of an OR-Gate

To account for glitches on $X$ and $Y$, the inputs of an OR-gate, we introduce another glitch-propagation function $\mathsf{G}_Z^{\mathsf{OR}} : \mathbb{F}2^6 \to \mathbb{F}_2$ for the output $Z$ of an OR-gate across its consecutive inputs $X', X, Y', Y$, along with two variables $G_X$ and $G_Y$ indicating the presence of a glitch on $X$ or $Y$ respectively. Additionally, we revise $\mathsf{F}_Z^{\mathsf{OR}} : \mathbb{F}2^6 \to \mathbb{F}_2$ to accommodate glitches on $X$ and $Y$.

$$\begin{aligned}
\mathsf{F}_Z^{\mathsf{OR}}(X', X, G_X, Y', Y, G_Y) &= (\bar{X}' \vee \bar{X} \vee G_X) \wedge (\bar{Y}' \vee \bar{Y} \vee G_Y) \\
\mathsf{G}_Z^{\mathsf{OR}}(X', X, G_X, Y', Y, G_Y) &= (X' \wedge \bar{X} \wedge \bar{Y}' \wedge Y) \vee (\bar{X}' \wedge X \wedge Y' \wedge \bar{Y}) \\
&\quad \vee (G_X \wedge (\bar{Y}' \vee \bar{Y} \vee G_Y)) \vee (G_Y \wedge (\bar{X}' \vee \bar{X} \vee G_X))
\end{aligned}$$

Now, $\mathsf{F}_Z^{\mathsf{OR}}$ stops the probe-extension only if at least one of the inputs remains constant at 1 and is free from glitches simultaneously. Put differently, any glitch occurring on $X$ or $Y$ triggers probe-extension. $\mathsf{G}_Z^{\mathsf{OR}}$ is build in the same way as $\mathsf{G}_Z^{\mathsf{AND}}$. The first two monomials of $\mathsf{G}_Z^{\mathsf{OR}}$ address scenarios where a glitch on $Z$ can occur even if $X$ and $Y$ are free of glitches (see $S_6$ and $S_9$ in Table 1b). However, if $X$ (or $Y$) is glitchy, we assume that the glitch propagates to $Z$ unless $Y$ (or $X$) is constantly one and free of glitches.

### 4.5 Other Gates

While we have introduced the concept of data-dependent probe and glitch propagation using the AND- and OR-gates as examples, we remark that the same principle can be applied to derive equivalent functions for other gates with a fan-in of two, such as NAND, NOR, XOR, and XNOR, in a straightforward manner. The corresponding results for these gates are provided in the appendix. However, the NOT-gate, which takes one input $X$ and produces one output $Z$, is treated differently. This is because every consecutive input is directly observable through the output $Z$. Consequently, for a NOT-gate, two key observations hold: (1) there exists no scenario where probe propagation should stop at $Z$, and (2) every glitch occurring on $X$ propagates to $Z$. Formally, this implies that the probe- and glitch-propagation functions return 1 for every possible input. To maintain simplicity, we opt not to define such functions explicitly and instead assume that a robust probe $P_Z^{\text{robust}}$ is replaced directly by a robust probe $P_X^{\text{robust}}$, aligning with the principles of the robust $d$-probing model.

## 5 The Robust but Relaxed (RR) $d$-Probing Model

Building upon our observations, we introduce the RR $d$-probing model, designed to relax the overly conservative glitch treatment of the robust $d$-probing model. Therefore, we define a new class of probes, referred to as relaxed probes, which are, when placed on the output signal of a combinational gate, conditionally propagated depending on the operation performed by the gate and its processed data. Before extending a relaxed probe $P_E^{\text{relaxed}}$ placed on an arbitrary wire $E$, it is imperative to ensure that all probe-extension

and glitch-propagation functions have been evaluated. When evaluating a masked circuit $\tilde{\mathsf{C}}$, i.e. initializing input gates and sequentially processing gates from input gates to output gates, we initially set $G_Z = 0$ if $Z$ is the output of an input gate. Subsequently, after evaluating the output of a combinational gate $Z$, we compute $G_Z$ and then $F_Z$ based on the functionality of the gate. Once this step is completed, and the information regarding probe and glitch propagation, i.e. all $F_E$ and $G_E$ for every $E$ which is an output of a combinational gate, is available, we proceed with the actual probe-extension procedure. We formalize the relaxed probe-extension procedure, denoted `relaxed_extend`, in Algorithm 3.

**Algorithm 3** Relaxed Extension (`relaxed_extend`)

**Input:** $P_E^{\text{relaxed}}$ ▷ Relaxed probe on wire $E$
**Output:** $\mathbf{P}$ ▷ The set of standard probes

**if** $P_E^{\text{relaxed}}$ is placed on the output of combinational gate with fan in 2 **then**
  **if** $F_E = 1$ **then**
    $\mathbf{P} \leftarrow \{\texttt{relaxed\_extend}(P_X^{\text{relaxed}})\} \cup \{\texttt{relaxed\_extend}(P_Y^{\text{relaxed}})\}$ ▷ $F_X$ and $F_Y$ denote the relaxed probes on both inputs of the combinational gate
  **else**
    $\mathbf{P} \leftarrow \{P_{E'}\} \cup \{P_E\}$ ▷ Do the transition extension
  **end if**
**else if** $P_E^{\text{relaxed}}$ is placed on the output of an NOT-gate **then**
  $\mathbf{P} \leftarrow \texttt{relaxed\_extend}(P_X^{\text{relaxed}})$ ▷ $P_X^{\text{relaxed}}$ denotes the relaxed probe on the input of the NOT-gate
**else**
  **if** $P_E^{\text{relaxed}}$ is placed on an output of a sequential gate or a primary input **then**
    $\mathbf{P} \leftarrow \{P_{E'}\} \cup \{P_E\}$ ▷ Do the transition extension
  **end if**
**end if**

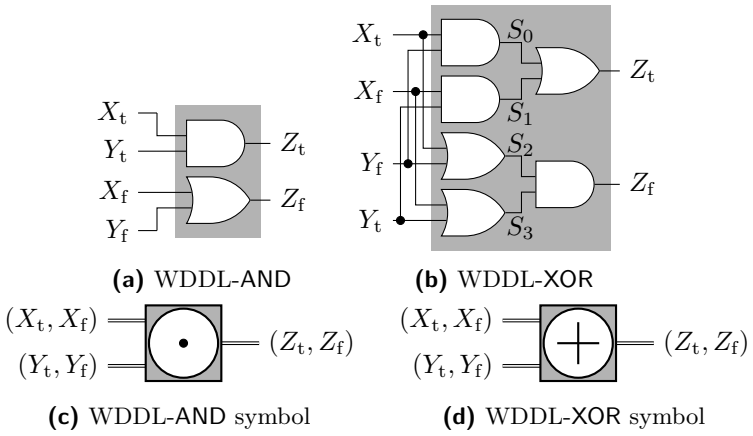
Algorithm 3 conditionally extends $P_E^{\text{relaxed}}$ based on $F_E$, i.e the decision whether a probe should be extended based on the operations and data. If such extension is deemed necessary, $P_E^{\text{relaxed}}$ gets substituted by relaxed probes on all inputs of the gate computing $E$. Otherwise, $P_E^{\text{relaxed}}$ gets directly replaced by two standard probes recording two consecutive states of $E$ without further extension.

## 5.1 An Illustrative Example

To provide an illustrative example of applying Algorithm 3 to a straightforward yet concrete design, and to support the argumentation for more complex designs, we briefly introduce the basic Wave Dynamic Differential Logic (WDDL)-AND and WDDL-XOR gadgets in Figure 2. We use the style and symbols introduced in [MLM23]. WDDL utilizes Dual-Rail Pre-charge (DRP) logic, where every variable $X$ is represented as a tuple $(X_\mathrm{t}, X_\mathrm{f})$, with $X_\mathrm{t} = X$ and $X_\mathrm{f} = \bar{X}$, in a way that it is guaranteed that no glitch can occur within a WDDL-protected circuit.

**Example 2** (WDDL-AND)**.** Consider a WDDL-AND (cf. Figure 2a) with two primary inputs each given in dual-rail representation as $(X_\mathrm{t}, X_\mathrm{f})$ and $(Y_\mathrm{t}, Y_\mathrm{f})$, and one primary output, also in dual-rail representation, $(Y_\mathrm{t}, Y_\mathrm{f})$. According to the DRP logic, we assume that the inputs are pre-charged, i.e. $(X'_\mathrm{t}, X'_\mathrm{f}, Y'_\mathrm{t}, Y'_\mathrm{f}, Z'_\mathrm{t}, Z'_\mathrm{f}) = (0,0,0,0,0,0)$ before being set to their final state and that $(G_{X_\mathrm{t}}, G_{X_\mathrm{f}}, G_{Y_\mathrm{t}}, G_{Y_\mathrm{f}}) = (0,0,0,0)$. Initially, let $P_{Z_\mathrm{t}}^{\text{relaxed}}$ be a relaxed probe placed on $Z_\mathrm{t}$. Before extending $P_{Z_\mathrm{t}}^{\text{relaxed}}$ by Algorithm 3 we compute $G_{Z_\mathrm{t}}$ and $F_{Z_\mathrm{t}}$ as follows:

$$G_{Z_\mathrm{t}} = \mathsf{G}_{Z_\mathrm{t}}^{\mathsf{AND}}(X'_\mathrm{t}, X_\mathrm{t}, G_{X_\mathrm{t}}, Y'_\mathrm{t}, Y_\mathrm{t}, G_{Y_\mathrm{t}}) \qquad F_{Z_\mathrm{t}} = \mathsf{F}_{Z_\mathrm{t}}^{\mathsf{AND}}(X'_\mathrm{t}, X_\mathrm{t}, G_{X_\mathrm{t}}, Y'_\mathrm{t}, Y_\mathrm{t}, G_{Y_\mathrm{t}})$$




**(a)** WDDL-AND **(b)** WDDL-XOR

**(c)** WDDL-AND symbol **(d)** WDDL-XOR symbol

**Figure 2:** WDDL gadgets.

When executing Algorithm 3, the decision to extend $P_{Z_t}^{\text{relaxed}}$ depends on $F_{Z_t}$. We show the concrete extensions for all valid states of $(X_t, Y_t, Z_t)$ in the following:

$$\begin{aligned}
(X_t, Y_t, Z_t) &= (0,0,0) \qquad F_{Z_t} = 0 \qquad P_{Z_t}^{\text{relaxed}} \to \{P_{Z_t'}, P_{Z_t}\} \\
(X_t, Y_t, Z_t) &= (0,1,0) \qquad F_{Z_t} = 0 \qquad P_{Z_t}^{\text{relaxed}} \to \{P_{Z_t'}, P_{Z_t}\} \\
(X_t, Y_t, Z_t) &= (1,0,0) \qquad F_{Z_t} = 0 \qquad P_{Z_t}^{\text{relaxed}} \to \{P_{Z_t'}, P_{Z_t}\} \\
(X_t, Y_t, Z_t) &= (1,1,1) \qquad F_{Z_t} = 1 \qquad P_{Z_t}^{\text{relaxed}} \to \{P_{X_t'}, P_{X_t}, P_{Y_t'}, P_{Y_t}\}
\end{aligned}$$

As $G_{Z_t}$ is zero for all valid input sequences, we can conclude that the DRP logic indeed leads to a glitch-free WDDL-AND. However, this does not hold if the gate is not properly pre-charged. For example, it holds that:

$$\begin{gathered}
(X_t', X_t, Y_t', Y_t, Z_t', Z_t) = (1,0,0,1,0,0) \qquad (G_{Z_t}, F_{Z_t}) = (1,1) \\
P_{Z_t}^{\text{relaxed}} \to \{P_{X_t'}, P_{X_t}, P_{Y_t'}, P_{Y_t}\}
\end{gathered}$$

Now, let $P_{Z_f}^{\text{relaxed}}$ be a relaxed probe placed on $Z_f$. We compute $G_{Z_f}$ and $F_{Z_f}$ as follows:

$$G_{Z_f} = \mathsf{G}_{Z_f}^{\mathsf{OR}}(X_f', X_f, G_{X_f}, Y_f', Y_f, G_{Y_f}) \qquad F_{Z_f} = \mathsf{F}_{Z_f}^{\mathsf{OR}}(X_f', X_f, G_{X_f}, Y_f', Y_f, G_{Y_f})$$

The decision to extend $P_{Z_f}^{\text{relaxed}}$ depends on $F_{Z_f}$. We show the concrete extensions for all valid input states in the following:

$$\begin{aligned}
(X_f, Y_f, Z_f) &= (0,0,0) \qquad F_{Z_f} = 1 \qquad P_{Z_f}^{\text{relaxed}} \to \{P_{X_f'}, P_{X_f}, P_{Y_f'}, P_{Y_f}\} \\
(X_f, Y_f, Z_f) &= (0,1,1) \qquad F_{Z_f} = 1 \qquad P_{Z_f}^{\text{relaxed}} \to \{P_{X_f'}, P_{X_f}, P_{Y_f'}, P_{Y_f}\} \\
(X_f, Y_f, Z_f) &= (1,0,1) \qquad F_{Z_f} = 1 \qquad P_{Z_f}^{\text{relaxed}} \to \{P_{X_f'}, P_{X_f}, P_{Y_f'}, P_{Y_f}\} \\
(X_f, Y_f, Z_f) &= (1,1,1) \qquad F_{Z_f} = 1 \qquad P_{Z_f}^{\text{relaxed}} \to \{P_{X_f'}, P_{X_f}, P_{Y_f'}, P_{Y_f}\}
\end{aligned}$$

**Example 3** (WDDL-XOR)**.** Consider a WDDL-XOR (cf. Figure 2b) with two primary inputs each given in dual-rail representation as $(X_t, X_f)$ and $(Y_t, Y_f)$, and one primary output, also in dual-rail representation, $(Y_t, Y_f)$. According to the DRP logic, we assume that the inputs are pre-charged, i.e. $(X_{t'}, X_{f'}, Y_t', Y_f', S_0, S_1, S_2, S_3, Z_t', Z_f') = (0,0,0,0,0,0,0,0,0,0)$ before being set to their final state and that $(G_{X_t}, G_{X_f}, G_{S_0}, G_{S_1}, G_{S_2}, G_{S_3}, G_{Y_t}, G_{Y_f}) = (0,0,0,0,0,0,0,0)$. Let $P_{Z_t}^{\text{relaxed}}$ (resp. $P_{Z_f}^{\text{relaxed}}$) be a relaxed probe placed on $Z_t$ (resp.

$Z_\mathrm{f}$). Here, we start by computing $(G_{S_0}, G_{S_1}, G_{S_2}, G_{S_3})$ followed by $(F_{S_0}, F_{S_1}, F_{S_2}, F_{S_3})$ as:

$$\begin{aligned}
G_{S_0} &= \mathsf{G}^{\mathsf{AND}}_{S_0}(X'_\mathrm{t}, X_\mathrm{t}, G_{X_\mathrm{t}}, Y'_\mathrm{f}, Y_\mathrm{f}, G_{Y_\mathrm{f}}) & F_{S_0} &= \mathsf{F}^{\mathsf{AND}}_{S_0}(X'_\mathrm{t}, X_\mathrm{t}, G_{X_\mathrm{t}}, Y'_\mathrm{f}, Y_\mathrm{f}, G_{Y_\mathrm{f}}) \\
G_{S_1} &= \mathsf{G}^{\mathsf{AND}}_{S_1}(X'_\mathrm{f}, X_\mathrm{f}, G_{X_\mathrm{f}}, Y'_\mathrm{t}, Y_\mathrm{t}, G_{Y_\mathrm{t}}) & F_{S_1} &= \mathsf{F}^{\mathsf{AND}}_{S_1}(X'_\mathrm{f}, X_\mathrm{f}, G_{X_\mathrm{f}}, Y'_\mathrm{t}, Y_\mathrm{t}, G_{Y_\mathrm{t}}) \\
G_{S_2} &= \mathsf{G}^{\mathsf{OR}}_{S_2}(X'_\mathrm{t}, X_\mathrm{t}, G_{X_\mathrm{t}}, Y'_\mathrm{f}, Y_\mathrm{f}, G_{Y_\mathrm{f}}) & F_{S_2} &= \mathsf{F}^{\mathsf{OR}}_{S_2}(X'_\mathrm{t}, X_\mathrm{t}, G_{X_\mathrm{t}}, Y'_\mathrm{f}, Y_\mathrm{f}, G_{Y_\mathrm{f}}) \\
G_{S_3} &= \mathsf{G}^{\mathsf{OR}}_{S_3}(X'_\mathrm{f}, X_\mathrm{f}, G_{X_\mathrm{f}}, Y'_\mathrm{t}, Y_\mathrm{t}, G_{Y_\mathrm{t}}) & F_{S_3} &= \mathsf{F}^{\mathsf{OR}}_{S_3}(X'_\mathrm{f}, X_\mathrm{f}, G_{X_\mathrm{f}}, Y'_\mathrm{t}, Y_\mathrm{t}, G_{Y_\mathrm{t}})
\end{aligned}$$

Next, $(G_{S_0}, G_{S_1}, G_{S_2}, G_{S_3})$ is used to compute $(G_{Z_\mathrm{t}}, G_{Z_\mathrm{f}})$ and $(F_{Z_\mathrm{t}}, F_{Z_\mathrm{f}})$ as:

$$\begin{aligned}
G_{Z_\mathrm{t}} &= \mathsf{G}^{\mathsf{OR}}_{Z_\mathrm{t}}(S'_0, S_0, G_{S_0}, S'_1, S_1, G_{S_1}) & F_{Z_\mathrm{t}} &= \mathsf{F}^{\mathsf{OR}}_{Z_\mathrm{t}}(S'_0, S_0, G_{S_0}, S'_1, S_1, G_{S_1}) \\
G_{Z_\mathrm{f}} &= \mathsf{G}^{\mathsf{AND}}_{Z_\mathrm{f}}(S'_2, S_2, G_{S_2}, S'_3, S_3, G_{S_3}) & F_{Z_\mathrm{f}} &= \mathsf{F}^{\mathsf{AND}}_{Z_\mathrm{f}}(S'_2, S_2, G_{S_2}, S'_3, S_3, G_{S_3})
\end{aligned}$$

When executing Algorithm 3, the decision to extend $P^{\mathrm{relaxed}}_{Z_\mathrm{t}}$ now depends on $(F_{S_0}, F_{S_1}, F_{Z_\mathrm{t}})$ while the extension of $P^{\mathrm{relaxed}}_{Z_\mathrm{f}}$ depends on $(F_{S_2}, F_{S_3}, F_{Z_\mathrm{f}})$. We show the concrete extensions for all valid input states in Table 2 and Table 3.

**Table 2:** Results from Algorithm 3 with $P^{\mathrm{relaxed}}_{Z_\mathrm{t}}$ depending on the input state.

| $(X_\mathrm{t}, X_\mathrm{f}, Y_\mathrm{t}, Y_\mathrm{f}, S_0, S_1, Z_\mathrm{t})$ | $(F_{S_0}, F_{S_1}, F_{Z_\mathrm{t}})$ | $P^{\mathrm{relaxed}}_{Z_\mathrm{t}}$ |
|---|---|---|
| $(0,1,0,1,0,0,0)$ | $(0,0,0)$ | $\{P_{Z'_\mathrm{t}}, P_{Z_\mathrm{t}}\}$ |
| $(0,1,1,0,0,1,1)$ | $(0,1,1)$ | $\{P_{S'_0}, P_{S_0}, P_{X'_\mathrm{f}}, P_{X_\mathrm{f}}, P_{Y'_\mathrm{t}}, P_{Y_\mathrm{t}}\}$ |
| $(1,0,0,1,1,0,1)$ | $(1,0,1)$ | $\{P_{S'_1}, P_{S_1}, P_{X'_\mathrm{t}}, P_{X_\mathrm{t}}, P_{Y'_\mathrm{f}}, P_{Y_\mathrm{f}}\}$ |
| $(1,0,1,0,0,0,0)$ | $(0,0,0)$ | $\{P_{Z'_\mathrm{t}}, P_{Z_\mathrm{t}}\}$ |

**Table 3:** Results from Algorithm 3 with $P^{\mathrm{relaxed}}_{Z_\mathrm{f}}$ depending on the input state.

| $(X_\mathrm{t}, X_\mathrm{f}, Y_\mathrm{t}, Y_\mathrm{f}, S_2, S_3, Z_\mathrm{f})$ | $(F_{S_2}, F_{S_3}, F_{Z_\mathrm{f}})$ | $P^{\mathrm{relaxed}}_{Z_\mathrm{f}}$ |
|---|---|---|
| $(0,1,0,1,1,1,1)$ | $(1,1,1)$ | $\{P_{X'_\mathrm{t}}, P_{X_\mathrm{t}}, P_{X'_\mathrm{f}}, P_{X_\mathrm{f}}, P_{Y'_\mathrm{t}}, P_{Y_\mathrm{t}}, P_{Y'_\mathrm{f}}, P_{Y_\mathrm{f}}\}$ |
| $(0,1,1,0,0,1,0)$ | $(0,1,0)$ | $\{P_{Z'_\mathrm{f}}, P_{Z_\mathrm{f}}\}$ |
| $(1,0,0,1,1,0,0)$ | $(1,0,0)$ | $\{P_{Z'_\mathrm{f}}, P_{Z_\mathrm{f}}\}$ |
| $(1,0,1,0,1,1,1)$ | $(1,1,1)$ | $\{P_{X'_\mathrm{t}}, P_{X_\mathrm{t}}, P_{X'_\mathrm{f}}, P_{X_\mathrm{f}}, P_{Y'_\mathrm{t}}, P_{Y_\mathrm{t}}, P_{Y'_\mathrm{f}}, P_{Y_\mathrm{f}}\}$ |

## 5.2 Security Notions

Building upon the probe-extension described in Algorithm 3 we introduce definitions for probing security (see Definition 6) and PINI (see Definition 7) within the RR $d$-probing model. Consequently, we denote an adversary capable of placing a maximum of $d$ relaxed probes on arbitrary wires of $\tilde{\mathsf{C}}$ as RR $d$-probing adversary.

**Definition 6** (RR $d$-Probing Security)**.** A masked circuit $\tilde{\mathsf{C}}$ with secret input $\mathbf{X}$ is RR $d$-probing secure iff for any set of relaxed probes $\mathbf{P}$, with $|\mathbf{P}| \leq d$, the joint distribution over all observations $\mathbf{Q}$ made by the relaxed probes is statistically independent of $\mathbf{X}$

**Definition 7** (RR $d$-Probe-Isolating Non-Interference (PINI))**.** Let $\tilde{\mathsf{C}}$ be a masked circuit with secret input $\mathbf{X}$. Further, let $\mathbf{P}_\mathrm{I}$ be a set of relaxed probes on internal wires of $\tilde{\mathsf{C}}$ with $|\mathbf{P}_\mathrm{I}| = t_1$ and $\mathbf{I}_\mathrm{O}$ be the set of indices assigned to the relaxed probes on output wires in the set $\mathbf{P}_\mathrm{O}$ with $|\mathbf{I}_\mathrm{O}| = t_2$. $\tilde{\mathsf{C}}$ is relaxed $d$-PINI iff for every $\mathbf{P} = \mathbf{P}_\mathrm{I} \cup \mathbf{P}_\mathrm{O}$ with $t_1 + t_2 \leq d$, there exists a set of indices $\mathbf{I}_\mathrm{I}$ with $|\mathbf{I}_\mathrm{I}| \leq t_1$ such that $\mathbf{P}$ is perfectly simulatable from $\mathbf{S} = \tilde{\mathbf{X}}^{\mathbf{I}_\mathrm{I} \cup \mathbf{I}_\mathrm{O}}$.

### 5.3 Relation to Other Probing Models

Having introduced a new model, it is essential to establish its relationship with previously discussed models, specifically the d-probing model and the robust d-probing model. To achieve this, we define two new probe-extension functions: $\mathsf{F}^0$ and $\mathsf{F}^1$. It holds that $\mathsf{F}^0$ constantly returns 0 independent of the underlying gate and data while $\mathsf{F}^1$ constantly returns 1 in the same manner. Based on $\mathsf{F}^0$ and $\mathsf{F}^1$ we formalize the relation between the RR $d$-probing model and the $d$-probing model (resp. robust $d$-probing model) in Lemma 1 (resp. Lemma 2).

**Lemma 1.** *The RR d-probing model with probe-extension function* $\mathsf{F}^0$ *is equivalent to the d-probing model.*

We remark that $\mathsf{F}^0$ implies no probe-extension, i.e. a relaxed probe $P_E^{\text{relaxed}}$ on $E$ is directly replaced by a standard probe[4] $P_E$ on $E$ if the probe-extension is performed based on $\mathsf{F}^0$ what is equivalent to placing $d$ standard probes on arbitrary wires as suggested by the $d$-probing model.

**Lemma 2.** *The RR d-probing model with probe-extension function* $\mathsf{F}^1$ *is equivalent to the robust d-probing model.*

Analogously, $\mathsf{F}^1$ implies a full probe-extension, i.e. a relaxed probe $P_E^{\text{relaxed}}$ on $E$ is extended backwards until a primary input or output of a sequential gate is reached. This corresponds to the probe-extension described in Algorithm 2 which is related to the robust $d$-probing model. As the probe-extension functions applied in Algorithm 3 conditionally return 0 or 1, we positioned our model between the $d$-probing model and the robust $d$-probing model. Hence, it holds that (1) every RR $d$-probing secure circuit is also $d$-probing secure (but not vice versa) and (2) every robust $d$-probing secure circuit is also RR $d$-probing secure (but not vice versa).

## 6 Tool-Assisted Evaluation

To automate the evaluation of arbitrary masked circuits, given as a gate-level netlist, we developed an extension of the PROLEAD tool which allows to verify $d$-probing security based on our new model. PROLEAD synergistically merges the advantages of formal verification, offering verification under well-defined adversary models, with leakage simulation, renowned for its efficiency. This integrated framework currently assesses $d$-probing security within the robust probing model by simulating intermediate signals within the target hardware circuit. Thanks to this approach, PROLEAD can analyze circuits that fall beyond the scope of existing formal verification tools, such as a fully masked AES core, with sufficient accuracy. Below, we focus on the adjustments we made to PROLEAD to encompass leakage evaluation under the RR $d$-probing model.

### 6.1 Extraction of Relevant Wires

When evaluating a masked circuit under the robust $d$-probing model, it suffices to focus exclusively on robust probes on primary outputs and inputs of sequential gates, denoted by a set of, so-called, relevant wires $\mathbf{H}$:

$$\mathbf{H} = \{E \in \mathbf{E} | E \textit{ is the input of a sequential gate or output gate}\}$$

Robust probes placed on other wires consistently provide less information than a robust probe in $\mathbf{H}$. This property is for example checked in Algorithm 1 of [MM22] to extract $\mathbf{H}$.

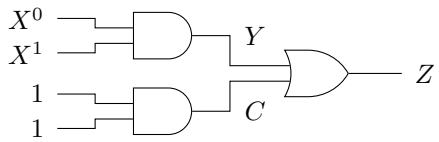



**Figure 3:** Exemplary circuit with constant inputs

To discuss the selection of relevant wires for the evaluation under the RR $d$-probing model we analyze the exemplary (masked) circuit depicted in Figure 3.

We assume that a secret input $X \in \mathbb{F}_2$ is represented as two shares $X^0, X^1$ satisfying $X = X^0 \oplus X^1$. Both shares serve as primary inputs while all other primary inputs are constantly kept at 1. Now, let $P_Z^{\text{relaxed}}$ be a relaxed probe on the primary output $Z$. As one input of the OR-gate is constantly set to 1, $P_Z^{\text{relaxed}}$ gets not extended further but is instead replaced by two standard probes $P_{Z'} = 1$ and $P_Z = 1$. The joint distribution made by $P_{Z'}$ and $P_Z$ is statistically independent of $X$. Consequently, if we only consider $P_Z^{\text{relaxed}}$ this circuit will be reported as RR 1-probing secure. However, let $P_Y^{\text{relaxed}}$ be a relaxed probe on $Y$ which remains unaffected by the primary inputs constantly set to 1. Hence, $P_Y^{\text{relaxed}}$ gets extended, in some cases, to the primary inputs $X^0$ and $X^1$ ultimately resulting in a set of standard probes $\mathbf{P} = \{P_{X^{0'}}, P_{X^0}, P_{X^{1'}}, P_{X^1}\}$ effectively violating the RR 1-probing security. Therefore, relaxed probes must be placed on all wires of the circuit increasing the number of sets of relaxed probes to $\binom{|\mathbf{E}|}{d}$, compared to the number of sets of robust probes $\binom{|\mathbf{R}|}{d}$. This overhead becomes particularly significant for higher security orders $d$. Technically, we adjusted Algorithm 1 of [MM22] to include every wire in the set of relevant wires.

## 6.2 Probe-Extension

Unlike the probe extension of the original PROLEAD version, which involves executing Algorithm 5 of [MM22], the probe extension for the RR $d$-probing model relies on the processed data, namely the simulations generated by PROLEAD. Thus, a direct replacement of Algorithm 5 with our probe-extension procedure, as formalized in Algorithm 3, is not possible. The rationale behind this is that, for the robust $d$-probing model, PROLEAD precomputes the probe extensions for all $E \in \mathbf{H}$ because the probe extension is independent of the processed data. However, this independence does not hold for the relaxed-robust probing model. As a result, we must compute the probe extension for every execution simulated by PROLEAD, which significantly increases the runtime dedicated to the evaluation process. To accomplish this efficiently, we extended PROLEAD's simulation routine to compute a triple $(E, G_E, F_E) \in \mathbb{F}_2^3$ for every wire $E$ serving as an output of a combinational gate, employing a bit-sliced approach. Technically, this incurs an overhead equivalent to evaluating the circuit three times, effectively tripling both the simulation runtime and the memory required to store the circuit's state. Further, when dealing with the RR $d$-probing model we precompute a simplified version of Algorithm 5 of [MM22] just to get the maximum of standard probes $n_{max}$ after the probe extension while integrating the actual data-dependent probe extension into the distribution-update routine.

## 6.3 Update Distributions

The original version of PROLEAD generates a distribution table for every set of standard probes $\mathbf{P}$ with $|\mathbf{P}| = n_{max}$ derived from extending up to $d$ robust probes. Subsequently,

[4]For the sake of simplicity, we ignore the transition-extension in our argumentation.

PROLEAD evaluates the distribution table by using a statistical hypothesis test. Specifically, for each simulation PROLEAD generates the observation set $\mathbf{Q}$ with $|\mathbf{Q}| = n_{max}$ by replacing every standard probe in $\mathbf{P}$ with its corresponding simulated state. Afterwards, the distribution table, counting how often a particular observation occurs, gets updated by $\mathbf{Q}$. As already explained, we generate $\mathbf{P}$, derived from extending relaxed probes, for every simulation individually, with $n \leq n_{max}$ varying accordingly which is shown in Line 4 of Algorithm 5. However, PROLEAD's contingency tables are structured such that they can only accommodate updates with observation sets of equal size. To achieve this, we pad every $\mathbf{Q}$ with trailing zeros such that $|\mathbf{Q}| = n$ becomes $n_{max}$. Unfortunately, this leads to another problem which we demonstrate by a practical example shown in Example 4.

**Example 4.** Consider the exemplary circuit shown in Figure 4 and a relaxed probe $P_Z^{\text{relaxed}}$.

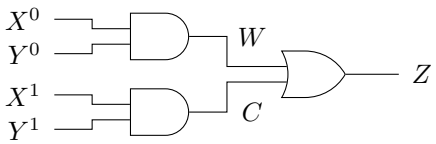



**Figure 4:** Exemplary circuit

Applying, the probe-extension on $P_Z^{\text{relaxed}}$ may result in a set of standard probes $\mathbf{P}$ such that $\mathbf{P} = \{P_{X^{0\prime}}, P_{X^0}, P_{X^{1\prime}}, P_{X^1}, P_{Y^{0\prime}}, P_{Y^0}, P_{Y^{1\prime}}, P_{Y^1}\}$ which also leads to $n_{max} = 8$. However, as we sort $\mathbf{P}$ in order to remove duplicates, it can be the case that two equal observation sets are derived although the underlying standard probes differ. To illustrate this, we consider two sorted sets, here and in the following denoted by square brackets, of standard probes, both ultimately resulting in $\mathbf{Q} = [0, 0, 0, 1, 0, 1, 0, 0]$.

$$\mathbf{P}_0 = [P_{X^{0\prime}}, P_{X^0}, P_{Y^{0\prime}}, P_{Y^0}, P_{C\prime}, P_C], F_Z = 1, F_W = 1, F_C = 0$$
$$\mathbf{P}_1 = [P_{X^{1\prime}}, P_{X^1}, P_{Y^{1\prime}}, P_{Y^1}, P_{W\prime}, P_W], F_Z = 1, F_W = 0, F_C = 1$$

Hence, the contingency table will falsely consider both observations as equal even if the observation relate to different variables.

To distinguish between equal observations in the contingency table, we track the probe-extension variables similar to the probes themselves. The ensuing procedure is delineated in Algorithm 4. Basically, Algorithm 4 returns a set of probe-extension variables $\mathbf{F}$ arranged in the order they are assessed during the probe-extension. Afterwards, the observation set is appended to $\mathbf{F}$. Applied to Example 4 this implies that $\mathbf{Q} = [0, 0, 0, 1, 0, 1, 0, 0]$ will be extended to two distinct observation sets, namely $\mathbf{Q}_0 = [1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0]$ and $\mathbf{Q}_1 = [1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0]$. Consequently, both sets become distinguishable.

The final procedure for generating the observation set is outlined in Algorithm 5. Essentially, Algorithm 5 iterates through all relaxed probes placed by an adversary and appends all standard probes derived through the probe extension to a global set of standard probes $\mathbf{P}$ in Line 4. Similarly, the sequence of all accessed probe-extension variables in generated in Line 5. Subsequently, we remove all duplicates in $\mathbf{P}$ in Line 7 and concatenate both sets to form the final observation set.

# 7 Case Studies

In the following, we demonstrate the versatility of the RR $d$-probing model, both in formal proofs and its seamless integration into PROLEAD for tool-assisted evaluation. However,

**Algorithm 4** Get Probe-Extension Variables (`get_probe_prop`)

**Input:** $F_E$ ▷ probe-extension variable of wire $E$
**Output:** $\mathbf{F}$ ▷ The set probe-extension variables

**if** $E$ is the output of a combinational gate with fan in 2 **then**
  $\mathbf{F} \leftarrow [\mathbf{F}, F_E]$
  **if** $F_E = 1$ **then**
    $\mathbf{F} \leftarrow [\mathbf{F}, \texttt{get\_probe\_prop}(F_X), \texttt{get\_probe\_prop}(F_Y)]$ ▷ $F_X$ and $F_Y$ denote the probe-extension variables of both inputs of the combinational gate
  **end if**
**else if** $E$ the output of an NOT-gate **then**
  $\mathbf{F} \leftarrow [\mathbf{F}, \texttt{get\_probe\_prop}(F_X)]$ ▷ $F_X$ denotes the probe-extension variable of the the NOT-gate input
**end if**

**Algorithm 5** Observation Generation

**Input:** $\mathbf{P}^{\text{relaxed}}$ ▷ The set of relaxed probes
**Output:** $\mathbf{Q}$ ▷ The observed variables

$\mathbf{P} \leftarrow \emptyset$
$\mathbf{F} \leftarrow \emptyset$
**for** $\forall P_E^{\text{relaxed}} \in \mathbf{P}^{\text{relaxed}}$ **do**
  $\mathbf{P} \leftarrow [\mathbf{P}, \texttt{relaxed\_extend}(P_E^{\text{relaxed}})]$
  $\mathbf{F} \leftarrow [\mathbf{F}, \texttt{get\_probe\_prop}(F_E)]$
**end for**
$\mathbf{P} \leftarrow \texttt{sort\_and\_remove\_duplicates}(\mathbf{P})$
$\mathbf{Q} \leftarrow [\mathbf{F}, \mathbf{P}]$

due to the shortage of sound masking schemes claiming security but without being robust $d$-probing secure, we focus on the schemes analyzed in [MLM23], specifically LMDPL [LMW14] and SESYM [NGPM22], both introduced at CHES. Both schemes utilize DRP logic, where every variable $X$ is represented as a tuple $(X_\mathrm{t}, X_\mathrm{f})$, with $X_\mathrm{t} = X$ and $X_\mathrm{f} = \bar{X}$, in a way that it is guaranteed that no glitch can occur within a LMDPL or SESYM-masked circuit. However, proving the security of either LMDPL or SESYM under the robust $d$-probing model would disregard this glitch-free characteristic, rendering both schemes insecure. Nevertheless, the introduced RR $d$-probing model acknowledges that LMDPL and SESYM-masked circuits cannot produce any glitch, making it ideally suited for analyzing such masked circuits. Therefore, our focus lies in providing the missing details outlined in [MLM23], specifically regarding the formal security (or insecurity) and composability proofs of LMDPL and SESYM within the RR $d$-probing model.

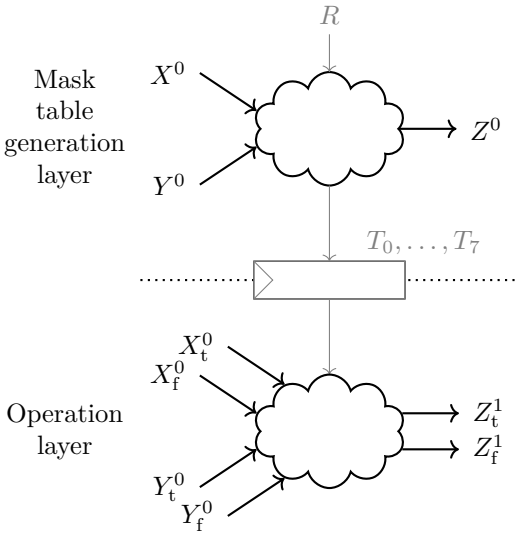
## 7.1 Setup

The following tool-assisted evaluations were conducted on an Ubuntu 22.04 server equipped with an Intel(R) Xeon(R) Gold 5320 CPU operating at 2.2 GHz and 2 TB of physical RAM. Additionally, we allowed PROLEAD to parallelize its execution using 52 hyper-threading cores. Given the substantial amount of available RAM, we adjusted PROLEAD to evaluate all sets of probes within a single run, allowing their resulting contingency tables to be kept in RAM simultaneously. This strategy improves evaluation time while increasing RAM consumption. We only limit PROLEAD's memory consumption by restricting the number of evaluated sets per batch if 2 TB proves insufficient. PROLEAD continuously monitors its process's virtual memory size to manage this. Specifically, PROLEAD can be configured to split the evaluation of all probing sets into multiple batches, each containing

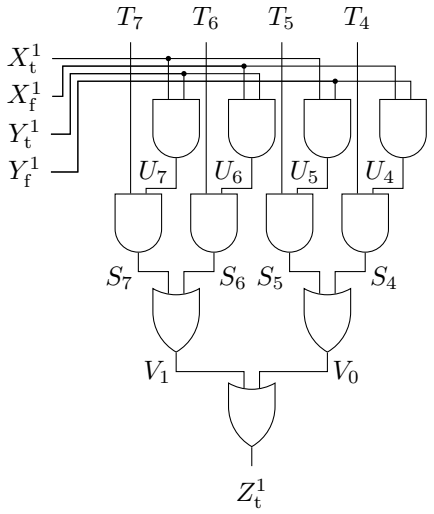
fewer probing sets, which are then evaluated sequentially. For example, if evaluating a particular design requires the assessment of one million probing sets, which is too large to fit in RAM, PROLEAD can evaluate it as 1000 batches of 1000 probing sets each. After the evaluation of the 1000 probing sets within a batch is completed, the results are reported, and the probing sets are subsequently deleted to free up space for the next batch meaning that PROLEAD only needs to keep the evaluation results of 1000 probing sets in memory at a time. This strategy enables PROLEAD to operate on machines with limited RAM, provided that a longer runtime is acceptable. In this case, PROLEAD performs multiple runs but with fewer sets of probes. We conducted all executions of PROLEAD in a fixed-vs-random setting based on its default false-negative probability $\beta = 10^{-5}$ and effect size $\varphi = 0.1$ as proposed in [MM22].

## 7.2 Non-Linear LMDPL Gadget

LMDPL, as introduced in [LMW14] and further explored in [SBHM20] and [MLM23], enables the construction of first-order secure composable gadgets designed to implement arbitrary Boolean functions, particularly tailored for low-latency requirements. Specifically, circuits composed solely of LMDPL gadgets achieve a constant latency, i.e. a constant amount of clock cycles, regardless of the underlying Boolean function. We sketch the construction of a generic LMDPL gadget in accordance with our circuit model in Figure 5.




**(a)** The two layers of a LMDPL gadget.




**(b)** One output rail of the operation layer based on gates with fan-in 2.

**Figure 5:** The generic structure on a LMDPL gadget based on gates with fan-in 2.

We denote the shared primary inputs of an LMDPL gadget as $(X^0, X^1)$ and $(Y^0, Y^1)$ satisfying $X = X^0 \oplus X^1$ and $Y = Y^0 \oplus Y^1$. Analogously, $(Z^0, Z^1)$ denotes the shares primary output with $Z = Z^0 \oplus Z^1$. We denote the dual-rail representation of all second shares as $(X_t^1, X_f^1, Y_t^1, Y_f^1, Z_t^1, Z_f^1)$ satisfying the following equations:

$$X^1 = X_t^1 \qquad Y^1 = Y_t^1 \qquad Z^1 = Z_t^1 \qquad \bar{X}^1 = X_f^1 \qquad \bar{Y}^1 = Y_f^1 \qquad \bar{Z}^1 = Z_f^1$$

The operations conducted by LMDPL gadgets are divided into two layers. The first layer, known as the mask table generation layer, processes the initial input shares $(X^0, Y^0)$ along with a fresh mask $R$ to compute the first output share as $Z^0 = R$. Additionally,

it computes a set of intermediate values $T_0, \ldots, T_7$, that all non-linear gadgets have to synchronize, according to the underlying Boolean function $\mathsf{T} : \mathbb{F}_2^2 \to \mathbb{F}_2$ as follows:

$$T_{4+2i+j} = \mathsf{T}(X^0 \oplus j, Y^0 \oplus i) \oplus R \qquad T_{2i+j} = T_{4+2i+j} \oplus 1$$

In the subsequent operation layer, the second shares $(X^1, Y^1)$ serve as select signals for two 4-to-1 multiplexers, efficiently directing two of the synchronized intermediates to serve as the second output shares $(Z_\mathrm{t}^1, Z_\mathrm{f}^1)$.

### 7.2.1 Probing Security

Prior to validating the security of LMDPL within the RR $d$-probing model, we underscore the disparity between the robust $d$-probing model and the physical security considerations pertinent to glitch-free circuits. Specifically, we ignore the glitch-free nature of LMDPL and prove its insecurity under the robust 1-probing model. Given the alternation of pre-charge and evaluation phases in the operation layer, we assume that all subsequent values processed by the operation layer are zero. Consequently, this renders transitions devoid of any informational gain. For the following proofs, we thus omit explicit delineation of additional probes aimed at capturing informational gains arising from transitions in the operation layer.

**Lemma 3.** *Any generic non-linear LMDPL is not robust* 1*-probing secure.*

*Proof.* Consider a robust 1-probing adversary placing one robust probe $P_{z_\mathrm{t}^1}^{\mathrm{robust}}$ on $z_\mathrm{t}^1$. According to Algorithm 1, it holds that:

$$P_{Z_\mathrm{t}^1}^{\mathrm{robust}} \to \{P_{X_\mathrm{t}^1}, P_{X_\mathrm{f}^1}, P_{Y_\mathrm{t}^1}, P_{Z_\mathrm{f}^1}, P_{T_4}, P_{T_5}, P_{T_6}, P_{T_7}\}$$

As all $T_4, \ldots, T_7$ are blinded by the same fresh mask $R$, adversary can remove the blinding from, e.g., $T_4$ and $T_5$, thereby acquiring information on both shares of $X$ and $Y$. Consequently, the resulting observation set is not statistically independent of, say, $X$. □

Lemma 3 directly underscores the rationale behind our new model. We proved that LMDPL is not robust 1-probing secure, while, conversely, analyses presented in [SBHM20] and [MLM23] affirm the security of LMDPL. The inconsistency arises because the robust $d$-probing model presupposes a full probe extension in the operation layer due to glitches. However, the operation layer is free of glitches, resulting in a constrained probe-extension scenario. To address this, we prove the security of LMDPL under the RR $d$-probing model.

**Lemma 4.** *Any generic non-linear LMDPL is RR* 1*-probing secure.*

*Proof.* We categorize the potential RR 1-probing adversaries according to the layer in which the relaxed probe is placed.

- Consider a RR 1-probing adversary placing one relaxed probe $P_E^{\mathrm{relaxed}}$ on an arbitrary wire $E$ in the mask table generation layer. For every $E$ in the mask table generation layer, it holds that $P_E^{\mathrm{relaxed}} \to \{P_{X^0}, P_{Y^0}, P_R\}$ is the most informative probe-extension according to Algorithm 3 (all probe-extension variables are 1). However, as only variables within the first share domain are observed, the resulting observation set is statistically independent of $X$ and $Y$.

- Consider a RR 1-probing adversary placing one relaxed probe in the operation layer. We focus on one output rail of the operation layer as shown in Figure 5b while the second rail behaves analogously. First, consider relaxed probes from the set $\{P_{U_4}^{\mathrm{relaxed}}, \ldots, P_{U_7}^{\mathrm{relaxed}}, P_{S_4}^{\mathrm{relaxed}}, \ldots, P_{S_7}^{\mathrm{relaxed}}\}$. While the observation sets made by $P_{U_4}^{\mathrm{relaxed}}, \ldots, P_{U_7}^{\mathrm{relaxed}}$ exclusively encompass variables within the second share domain

and are, therefore, statistically independent of $X$ and $Y$, it holds that the observation sets made by $P_{S_4}^{\text{relaxed}}, \ldots, P_{S_7}^{\text{relaxed}}$ encompass only variables within the second share domain along with one on of the synchronized values from $T_4, \ldots, T_7$. It holds that all $T_4, \ldots, T_7$ are blinded by $R$ and therefore uniformly distributed. Hence, the observation sets made by $P_{S_4}^{\text{relaxed}}, \ldots, P_{S_7}^{\text{relaxed}}$ are statistically independent of $X$ and $Y$. For analyzing the probe-extension of $P_{V_0}^{\text{relaxed}}$, $P_{V_1}^{\text{relaxed}}$, and $P_{Z_{\text{t}}^1}^{\text{relaxed}}$ it is important that, during the pre-charge phase, all inputs of the operation layer are set to 0 while, during the evaluation phase exactly one variable from $U_4, \ldots, U_7$ switches to one. Hence, it holds that $(F_{U_7}, F_{U_6}, F_{U_5}, F_{U_4})$ can only reach one of the states from the set $\{(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)\}$. As $U_4, \ldots, U_7$ serves as inputs when computing $S_4, \ldots, S_7$, it further holds that $(F_{S_7}, F_{S_6}, F_{S_5}, F_{S_4})$ can only reach one of the states from the set $\{(0,0,0,0),(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)\}$. As $(F_{S_7}, F_{S_6}, F_{S_5}, F_{S_4})$ restricts the probe-extension, it holds that $P_{V_0}^{\text{relaxed}}$ (resp. $P_{V_1}^{\text{relaxed}}$) extends to either one of the following sets:

$$
\begin{aligned}
P_{V_0}^{\text{relaxed}} &\to \{P_{S_4}^{\text{relaxed}}, P_{S_5}^{\text{relaxed}}\} \to \{P_{S_5}, P_{X_{\text{f}}^1}, P_{Y_{\text{f}}^1}, P_{T_4}\} \\
P_{V_0}^{\text{relaxed}} &\to \{P_{S_4}^{\text{relaxed}}, P_{S_5}^{\text{relaxed}}\} \to \{P_{S_4}, P_{X_{\text{t}}^1}, P_{Y_{\text{f}}^1}, P_{T_5}\} \\
P_{V_0}^{\text{relaxed}} &\to \{P_{S_4}^{\text{relaxed}}, P_{S_5}^{\text{relaxed}}\} \to \{P_{S_4}, P_{S_5}\} \\
P_{V_1}^{\text{relaxed}} &\to \{P_{S_6}^{\text{relaxed}}, P_{S_7}^{\text{relaxed}}\} \to \{P_{S_7}, P_{X_{\text{t}}^1}, P_{Y_{\text{f}}^1}, P_{T_6}\} \\
P_{V_1}^{\text{relaxed}} &\to \{P_{S_6}^{\text{relaxed}}, P_{S_7}^{\text{relaxed}}\} \to \{P_{S_6}, P_{X_{\text{t}}^1}, P_{Y_{\text{t}}^1}, P_{T_7}\} \\
P_{V_1}^{\text{relaxed}} &\to \{P_{S_6}^{\text{relaxed}}, P_{S_7}^{\text{relaxed}}\} \to \{P_{S_6}, P_{S_7}\}
\end{aligned}
$$

Further, all $P_{S_4}, \ldots P_{S_7}$ within the sets constantly record 0 and are, therefore, not informative. Hence, depending on the actual probe-extension, $P_{V_0}^{\text{relaxed}}$ (resp. $P_{V_1}^{\text{relaxed}}$ is as informative as a probe from the set $\{P_{S_4}^{\text{relaxed}}, \ldots, P_{S_7}^{\text{relaxed}}\}$ and its resulting observation set is statistically independent of $X$ and $Y$. The same arguments hold for $P_{Z_{\text{t}}^1}^{\text{relaxed}}$ with the following possible probe-extensions:

$$
\begin{aligned}
P_{Z_{\text{t}}^1}^{\text{relaxed}} &\to \{P_{S_4}^{\text{relaxed}}, P_{S_5}^{\text{relaxed}}, P_{S_6}^{\text{relaxed}}, P_{S_7}^{\text{relaxed}}\} \to \{P_{X_{\text{f}}^1}, P_{Y_{\text{f}}^1}, P_{T_4}, P_{S_5}, P_{S_6}, P_{S_7}\} \\
P_{Z_{\text{t}}^1}^{\text{relaxed}} &\to \{P_{S_4}^{\text{relaxed}}, P_{S_5}^{\text{relaxed}}, P_{S_6}^{\text{relaxed}}, P_{S_7}^{\text{relaxed}}\} \to \{P_{S_4}, P_{X_{\text{t}}^1}, P_{Y_{\text{f}}^1}, P_{T_5}, P_{S_6}, P_{S_7}\} \\
P_{Z_{\text{t}}^1}^{\text{relaxed}} &\to \{P_{S_4}^{\text{relaxed}}, P_{S_5}^{\text{relaxed}}, P_{S_6}^{\text{relaxed}}, P_{S_7}^{\text{relaxed}}\} \to \{P_{S_4}, P_{S_5}, P_{X_{\text{f}}^1}, P_{Y_{\text{t}}^1}, P_{T_6}, P_{S_7}\} \\
P_{Z_{\text{t}}^1}^{\text{relaxed}} &\to \{P_{S_4}^{\text{relaxed}}, P_{S_5}^{\text{relaxed}}, P_{S_6}^{\text{relaxed}}, P_{S_7}^{\text{relaxed}}\} \to \{P_{S_4}, P_{S_5}, P_{S_6}, P_{X_{\text{t}}^1}, P_{Y_{\text{t}}^1}, P_{T_7}\} \\
P_{Z_{\text{t}}^1}^{\text{relaxed}} &\to \{P_{S_4}^{\text{relaxed}}, P_{S_5}^{\text{relaxed}}, P_{S_6}^{\text{relaxed}}, P_{S_7}^{\text{relaxed}}\} \to \{P_{S_4}, P_{S_5}, P_{S_6}, P_{S_7}\}
\end{aligned}
\tag{3}
$$

Again, all $P_{S_4}, \ldots P_{S_7}$ are not informative. Hence, depending on the actual probe-extension, $P_{Z_{\text{t}}^1}^{\text{relaxed}}$ is as informative as a probe from the set $\{P_{S_4}^{\text{relaxed}}, \ldots, P_{S_7}^{\text{relaxed}}\}$ and its resulting observation set is statistically independent of $X$ and $Y$.

□

### 7.2.2 Composability

Based on the probe-extensions discussed above, we show that LMDPL is also composable.

**Lemma 5.** *Any generic non-linear LMDPL is RR* 1*-PINI.*

*Proof.* According to Definition 7, for $d = 1$, it holds that either $t_1 = 1$, $t_2 = 0$ or $t_1 = 0$, $t_2 = 1$. We prove the perfect probe simulation for every relaxed probe in both layers separately.

- Let $E$ be an intermediate wire in the mask table generation layer. It holds that every $P_E^{\text{relaxed}}$ with at most $P_E^{\text{relaxed}} \to \{P_{X^0}, P_{Y^0}, P_R\}$ is perfectly simulatable with a set of share index $\{0\}$ and by tossing a fair coin. Further, $P_{Z^0}^{\text{relaxed}}$ with $\mathbf{I}_\text{O}$ is perfectly simulatable just by tossing a fair coin.
- For the operation layer, all probes in $\{P_{U_4}^{\text{relaxed}}, \ldots, P_{U_7}^{\text{relaxed}}, P_{S_4}^{\text{relaxed}}, \ldots, P_{S_7}^{\text{relaxed}}\}$ are perfectly simulatable with $\mathbf{I}_\text{I} = \{1\}$ and by tossing a fair coin. Further, we already showed that $P_{V_0}^{\text{relaxed}}$, $P_{V_1}^{\text{relaxed}}$, and $P_{Z_\text{t}^1}^{\text{relaxed}}$ are as informative as a probe from $\{P_{S_4}^{\text{relaxed}}, \ldots, P_{S_7}^{\text{relaxed}}\}$. Hence, $P_{V_0}^{\text{relaxed}}$, $P_{V_1}^{\text{relaxed}}$, and $P_{Z_\text{t}^1}^{\text{relaxed}}$ are perfectly simulatable with a set of share index $\{1\}$ and by tossing a fair coin.

□

### 7.2.3 Tool-Assisted Evaluation

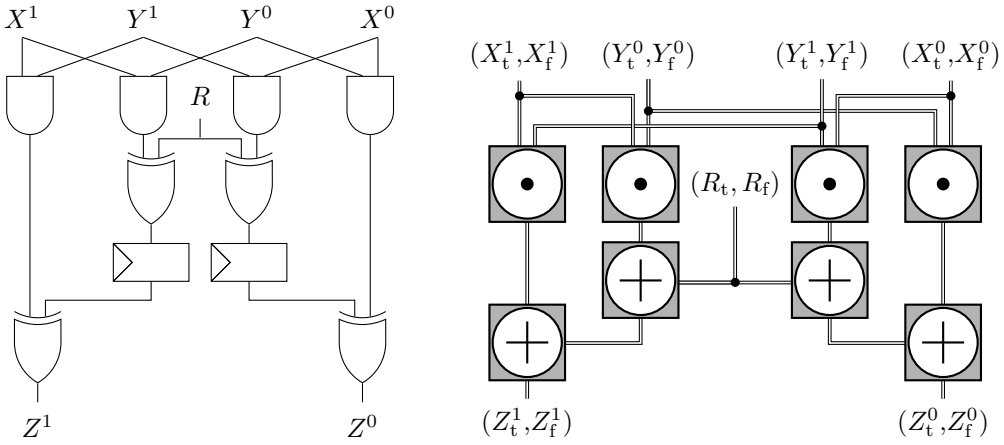
To confirm the provided proofs using PROLEAD, we conducted multiple case studies utilizing automatically generated masked circuits produced by AGEMA [KMMS22] publicly available on GitHub[5]. AGEMA takes an unprotected gate-level netlist and converts it into a gadget-based masked design. In our approach, AGEMA replaces all circuit gates with their corresponding LMDPL equivalents, ensuring full protection by LMDPL. The designs provided by AGEMA are then synthesized by using Synopsys Design Compiler (DC) and the NanGate 45 nm standard cell library and evaluated by PROLEAD. Initially, we analyzed the robust 1-probing security of a single LMDPL-AND. The results of PROLEAD confirm our claim made in Lemma 3 as it immediately detects significant first-order leakage after around 500 simulations. Furthermore, PROLEAD reports $P_{Z_\text{t}^1}^{\text{robust}}$ as a leaking probe. Contrary, if we switch to an evaluation under the RR 1-probing model, PROLEAD reports the security of the design. Since PROLEAD lacks the capability to verify the composability of a gadget, we proceed to analyze the probing security of a composed circuit, namely an Advanced Encryption Standard (AES) sbox, and an iterative design, namely a full byte-serial LMDPL-AES design. For the sbox, we utilized the unprotected netlist of the Boyar-Peralta sbox [BP12], sourced from the public AGEMA repository, and employed AGEMA to protect it with LMDPL gadgets. As expected, PROLEAD identifies significant first-order leakage after approximately 200000 simulations. We remark that the number of required simulation to detect the leakage is increased compared to the analysis of a single LMDPL-AND. The main reason for this is that the set of standard probes derived via the probe-extension procedure is increased. However, PROLEAD conveniently reports the necessary trace count, providing clarity on the simulation requirements. Further, PROLEAD reports the LMDPL-protected version of the Boyar-Peralta sbox as secure under the RR 1-probing model. Finally, we explore an iterative byte-serial LMDPL-AES design where its output state feeds back to its inputs. As before, we took the unprotected byte-serial AES-128 netlist from the public AGEMA repository and used AGEMA to protect it with LMDPL gadgets. The complete architecture comprises the LMDPL-based round-function logic which receives the outputs from a LMDPL-based multiplexer stage. These multiplexers forward either the shared primary inputs or the outputs of the state registers, determined by a select signal. The resulting output of the round function is then stored in the state register, and the primary output is also obtained from this register. We remark that the additional control logic must not be protected by LMDPL and that one encryption procedure takes 454 clock cycles. Once more, PROLEAD identifies leakage

[5] https://github.com/Chair-for-Security-Engineering/AGEMA

in the byte-serial LMDPL-AES design under the robust 1-probing model and confirms it security under the RR 1-probing model.

## 7.3 Non-linear SESYM Gadget

SESYM presents a deterministic algorithm capable of transforming any robust $d$-probing secure circuit into a masked circuit assumed to be secure, achieving this with a latency of only one clock cycle [NGPM22]. This algorithm first converts all signals from single-rail into their dual-rail representation and replaces all combinational gates by their corresponding WDDL gadgets. Due to the usage of WDDL gadgets in conjunction with alternating pre-charge and evaluation phases, the resulting circuit becomes glitch-free. Therefore, all sequential gates required to achieve robust $d$-probing security are removed from the final SESYM-masked circuit while Muller C-elements are employed to maintain stable output values. We visualize this transformation based on a robust 1-probing secure Domain-Oriented Masking (DOM)-AND gadget in Figure 6.




**(a)** DOM-AND-gadget **(b)** SESYM-AND-gadget

**Figure 6:** Conversion of a DOM-AND-gadget into a SESYM-AND-gadget.

### 7.3.1 Probing Security

Similar to LMDPL, we start by proving the insecurity of a $d$-th order masked SESYM-AND gadget. To provide the following proofs, it is sufficient to consider the last share domain as depicted in Figure 7.

**Lemma 6.** *Any d-th order masked SESYM-AND gadget is not robust* 1*-probing secure.*

*Proof.* Consider a robust 1-probing adversary placing one robust probe $P_{z_t^d}^{\text{robust}}$ on $z_t^d$. According to Algorithm 1, it holds that:

$$P_{Z_t^d}^{\text{robust}} \rightarrow \{P_{X_t^d}, P_{X_f^d}, P_{Y_t^0}, P_{Y_f^0}, \ldots, P_{Y_t^d}, P_{Y_f^d}\}$$

As the set encompasses standard probes on all shares of $Y$, i.e. $P_{Y_t^0}, P_{Y_f^0}, \ldots, P_{Y_t^d}, P_{Y_f^d}$, it holds that the resulting observation set made by $P_{z_t^d}^{\text{robust}}$ on $z_t^d$ is not statistically independent of $Y$. □

**Lemma 7.** *Any d-th order masked SESYM-AND gadget is not RR* 1*-probing secure.*

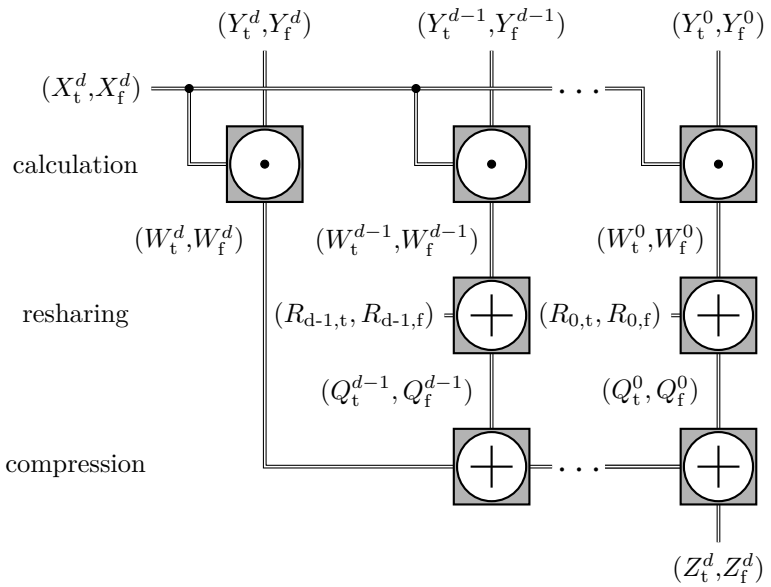



**Figure 7:** Last share domain of a $d$-th order SESYM-AND.

*Proof.* Consider a robust 1-probing adversary placing one relaxed probe $P^{\text{relaxed}}_{z_t^d}$ on $z_t^d$. Further, consider a particular state of inputs during the evaluation phase with $(X_t^d, X_f^d) = (1,0)$, $(R_{i,t}, R_{i,f}) = (1,0)$, and $(Y_t^i, Y_f^i) = (1,0)$ for all $0 \leq i \leq d$. According to Figure 2a it holds that all $W_t^i$ toggles from 0 to 1 while all $W_f^i$ stay constantly at 0. Hence, it holds that $F_{W_t^i} = 1$ (cf. $S_5$ of Table 1a) and $F_{W_f^i} = 1$ (cf. $S_0$ of Table 1b). Now, consider the WDDL-XOR gadgets computing $(Q_t^i, Q_f^i)$. As all $(W_t^i, R_{i,t})$ toggle from 0 to 1 while all $(W_f^i, R_{i,f})$ stay constantly at 0, it holds that $F_{Q_f^i} = 1$. Specifically, both OR gates on the negative output rail exhibit $S_1$ or $S_4$ of Table 1b while the following AND gate, computing $Q_f^i$, exhibits $S_5$ of Table 1a. However, $Q_t^i$ stays constantly at zero and it holds that $F_{Q_t^i} = 0$. Finally, we consider the leftmost WDDL-XOR gadget in the compression layer receiving $(W_t^d, W_f^d)$ and $(Q_t^{d-1}, Q_f^{d-1})$ as inputs. As already shown, $W_t^d$ and $Q_f^{d-1}$ toggle from 0 to 1 while both other signals stay constantly at 0. In this case, it holds that the output provided by the positive rail of the WDDL-XOR gadget toggles from 0 to 1 and that its corresponding probe-extension variable is equal to 1. In particular, one of the AND gates on the positive output rail exhibits $S_5$ of Table 1a while the following OR gate, computing the positive output rail, exhibits $S_5$ of Table 1a. However, as the output of this gadget again leads to a toggle on the positive output rail, the same extension procedure applies to all following WDDL-XOR gadgets in the compression layer, ultimately leading to the same set of standard probes as before which is not statistically independent of $Y$:

$$P^{\text{relaxed}}_{Z_t^d} \rightarrow \{P_{X_t^d}, P_{X_f^d}, P_{Y_t^0}, P_{Y_f^0}, \ldots, P_{Y_t^d}, P_{Y_f^d}\}$$

□

### 7.3.2 Tool-Assisted Evaluation

We also confirmed the insecurity of SESYM using PROLEAD. Therefore, we implemented, synthesized, and evaluated a first- and second-order masked SESYM-AND based on the same setup as previously described. Both versions were found susceptible to leakage, swiftly detected by PROLEAD through evaluation of the robust 1-probing security and the RR 1-probing security.

## 7.4 Benchmark

From Section 6, it becomes clear that the main drawback of evaluating under the RR $d$-probing model, compared to the robust $d$-probing model, is the additional evaluation overhead due to the more complex handling of glitches. To quantify the overhead in terms of runtime and memory requirements, we conducted several benchmarks using our presented designs. For comparison, we, additionally, benchmarked several masked designs based on DOM: One first-order and one second-order secure DOM-AND implemented by us, one first- and one second-order secure DOM-based AES S-box, and a first-order secure DOM-based design byte-serial AES presented in [GMK16]. Since these designs were previously tested [MM22] and are publicly available on GitHub[6], we used the same behavior-level designs and corresponding configuration files. The results, including absolute runtimes, memory consumption, and the relative overhead between the robust $d$-probing model and the RR $d$-probing model, are summarized in Table 4.

**Table 4:** Performance metrics of evaluations under the robust $d$-probing model and the RR $d$-probing model using enough simulations to detect effects of size $\varphi \geq 0.1$.

| | Design | Security | Robust | | Relaxed | | Overhead | |
|---|---|---|---|---|---|---|---|---|
| | | [expected] | [time] | [ram] | [time] | [ram] | [time] | [ram] |
| AND | DOM [GMK16] | 1 | 0.037 sec | 3.80 GB | 0.009 sec | 3.80 GB | ×0.25 | ×1.00 |
| | DOM [GMK16] | 2 | 0.072 sec | 3.80 GB | 0.079 sec | 3.80 GB | ×1.10 | ×1.00 |
| | SESYM [NGPM22] | 1 | 0.033 sec | 3.80 GB | 0.026 sec | 3.80 GB | ×0.79 | ×1.00 |
| | SESYM [NGPM22] | 2 | 0.046 sec | 3.72 GB | 0.091 sec | 3.81 GB | ×1.98 | ×1.03 |
| | LMDPL [KMMS22] | 1 | 0.038 sec | 3.80 GB | 0.027 sec | 3.80 GB | ×0.72 | ×1.00 |
| sbox | TI [MPL+11] | 1 | 1.169 min | 39.07 GB | 11.65 min | 93.91 GB | ×7.20 | ×2.41 |
| | DOM [GMK16] | 1 | 17.01 sec | 9.40 GB | 1.691 min | 10.52 GB | ×5.97 | ×1.12 |
| | DOM [GMK16] | 2 | 1.998 hr | 1.25 TB | 1.133 day | 1.51 TB | ×13.61 | ×1.21 |
| | LMDPL [KMMS22] | 1 | 16.12 sec | 4.62 GB | 13.76 min | 52.46 GB | ×51.21 | ×11.36 |
| AES | TI [MPL+11] | 1 | 19.23 min | 387.07 GB | 1.13 day | 1.95 TB | ×84.42 | ×5.04 |
| | DOM [GMK16] | 1 | 7.24 min | 154.31 GB | 46.53 min | 471.29 GB | ×6.43 | ×3.06 |
| | LMDPL [KMMS22] | 1 | 12.65 hr | 1.57 TB | 10.31 day | 1.96 TB | ×16.62 | ×1.41 |

While the concrete evaluation overhead of the RR $d$-probing model compared to the robust $d$-probing model strongly depends on the design itself, particularly on the total number of wires and the extent to which a robust or relaxed probe on a single wire can be extended, it is not feasible to provide a general evaluation performance overhead. However, based on the benchmarks in Table 4, we can make the following observations:

- Surprisingly, we observe a runtime improvement for the RR $d$-probing model, with only negligible memory overhead, when analyzing the first-order security of gadgets. This improvement occurs because the variance of different observations made by a set of probes decreases when switching from robust to relaxed probes, resulting in a distribution with fewer values. At the same time, the number of relevant wires grows only marginally, and each relaxed probe can only be extended through a small number of gates, effectively minimizing the overhead of the RR $d$-probing model.

- When analyzing more complex circuits, such as masked sboxes or full cipher cores, the robust $d$-probing model proves to be several orders of magnitude faster than the RR $d$-probing model. This disparity arises because the RR $d$-probing model needs to consider significantly more relevant wires, and relaxed probes must be conditionally extended through long cascaded chains of gates.

- Unfortunately, the relative runtime and memory overhead is significantly higher for LMDPL- and SESYM-based designs compared to DOM- and TI-based designs. This is due to the fact that probes placed on LMDPL- and SESYM-based designs can be

[6] https://github.com/ChairImpSec/PROLEAD

partially extended back to primary inputs without being interrupted by registers. Consequently, several intermediate relaxed probes must also be evaluated, whereas this is not required in the robust $d$-probing model.

- The relative runtime and memory overhead also increase with the security order $d$, primarily due to the increased number of relevant wires. As the number of relevant wires increases, the number of sets of probes grows exponentially with $d$.

Due to the significant runtime and memory overhead of the RR $d$-probing model, we do not recommend using it as a direct replacement for the robust $d$-probing model. Therefore, instead of evaluating every design under the RR $d$-probing model, we recommend evaluating the design under the robust $d$-probing model first. Only if the initial evaluation fails should one consider using our model. Furthermore, although even full masked cipher cores can be evaluated using our model, requiring a powerful server and potentially hours or days of computation time, we recommend following the established approach of designing and evaluating provably secure and composable gadgets under our new model. This approach avoids the complexities and potential pitfalls of hand-crafted masked designs that do not ensure composability.

# 8 Conclusions

In this work, we introduced the RR $d$-probing model, a formal adversary model capable of modeling the same level of security as the robust $d$-probing model, but with a less conservative treatment of glitches. Specifically, the probe-extension procedure, outlined in this work, addresses glitches based on the performed operations and processed data, effectively disregarding glitches that cannot occur within the analyzed masked circuit. Furthermore, we developed an updated version of PROLEAD to automatically evaluate the security of masked circuits within our new model. Demonstrating the capabilities of our approach, we conducted formal security assessments of LMDPL and SESYM, confirming the findings previously reported in [SBHM20] and [MLM23]. Despite the increased evaluation overhead in terms of runtime and memory requirements due to the utilization of a more complex probe-extension procedure, it is worth noting that the current runtime and memory demands of PROLEAD remain within acceptable limits. This ensures that PROLEAD can effectively evaluate both individual gadgets and fully masked cipher cores. We firmly believe that our model represents a crucial step towards the development of new masked circuits (gadgets) that are secure under our proposed model but may not be under the robust $d$-probing model. We must once again reference LMDPL because its existence serves as proof that such gadgets exist, i.e. RR $d$-probing secure and composable gadgets that are not robust $d$-probing secure. Additionally, the favorable properties of LMDPL concerning its constant amount of latency seem to be incompatible with security under the robust $d$-probing model. Therefore, exploring the development of new (low-latency) gadgets alongside LMDPL, such as a higher-order RR $d$-probing secure version of LMDPL, would be an intriguing avenue for future research. Furthermore, since PROLEAD is designed solely for security checking, there is a critical need to extend or develop a formal verification tool capable of verifying the composability of new gadget constructions under our new model.

# Acknowledgments

The work described in this paper has been supported in part by the German Research Foundation (DFG) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972, and by the Federal Ministry of Education and Research of Germany through the Project KOSEF (16KIS1597).

## References

[AIS18] Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private circuits: A modular approach. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018*, volume 10993 of *LNCS*, pages 427–455. Springer, 2018.

[BGI+18] Roderick Bloem, Hannes Groß, Rinat Iusupov, Bettina Könighofer, Stefan Mangard, and Johannes Winter. Formal verification of masked hardware implementations in the presence of glitches. In *EUROCRYPT 2018*, volume 10821 of *Lecture Notes in Computer Science*, pages 321–353. Springer, 2018.

[BMRT22] Sonia Belaïd, Darius Mercadier, Matthieu Rivain, and Abdul Rahman Taleb. Ironmask: Versatile verification of masking security. In *IEEE SP 2022*, pages 142–160. IEEE, 2022.

[BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the AES s-box. In *IFIP TCSEC 2012*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 287–298. Springer, 2012.

[CBG+17] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does Coupling Affect the Security of Masked Implementations? In *COSADE 2017*, volume 10348 of *LNCS*, pages 1–18. Springer, 2017.

[CGLS21] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware Private Circuits: From Trivial Composition to Full Verification. *IEEE Trans. Comp.*, 70(10):1677–1690, 2021.

[CGP+12] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In *COSADE 2012*, volume 7275 of *LNCS*, pages 69–81. Springer, 2012.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *CRYPTO 1999*, volume 1666 of *LNCS*, pages 398–412. Springer, 1999.

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

[CS21] Gaëtan Cassiers and François-Xavier Standaert. Provably secure hardware masking in the transition- and glitch-robust probing model: Better safe than sorry. *IACR TCHES*, 2021(2):136–158, 2021.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 423–440. Springer, 2014.

[DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 401–429. Springer, 2015.

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

[GBPV10] Benedikt Gierlichs, Lejla Batina, Bart Preneel, and Ingrid Verbauwhede. Revisiting higher-order dpa attacks:. In *Topics in Cryptology - CT-RSA 2010*, pages 221–234, 2010.

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In *TIS @ CCS 2016*, page 3. ACM, 2016.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, 2003.

[KM22] David Knichel and Amir Moradi. Low-latency hardware private circuits. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *CCS 2022*, pages 1799–1812. ACM, 2022.

[KMMS22] David Knichel, Amir Moradi, Nicolai Müller, and Pascal Sasdrich. Automated Generation of Masked Hardware. *IACR TCHES*, 2022(1):589–629, 2022.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO 1996*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.

[KSM20] David Knichel, Pascal Sasdrich, and Amir Moradi. SILVER - statistical independence and leakage verification. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020*, volume 12491 of *LNCS*, pages 787–816. Springer, 2020.

[LMW14] Andrew J. Leiserson, Mark E. Marson, and Megan A. Wachs. Gate-level masking under a path-based leakage metric. In *CHES 2014*, volume 8731 of *LNCS*, pages 580–597. Springer, 2014.

[MLM23] Nicolai Müller, Daniel Lammers, and Amir Moradi. A deep analysis of two glitch-free hardware masking schemes sesym and lmdpl. *IACR Cryptol. ePrint Arch.*, 2023.

[MM22] Nicolai Müller and Amir Moradi. PROLEAD A Probing-Based Hardware Leakage Detection Tool. *IACR TCHES*, 2022(4):311–348, 2022.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-Channel Leakage of Masked CMOS Gates. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 351–365. Springer, 2005.

[MPL+11] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of AES. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer, 2011.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully Attacking Masked AES Hardware Implementations. In *CHES 2005*, volume 3659 of *LNCS*, pages 157–171. Springer, 2005.

[NGPM22] Rishub Nagpal, Barbara Gigerl, Robert Primas, and Stefan Mangard. Riding the waves towards generic single-cycle masking in hardware. *IACR TCHES*, 2022(4):693–717, 2022. Git repository: https://extgit.iaik.tugraz.at/sesys/self-synchronized-masking.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881, pages 142–159. Springer, 2013.

[Rab96] Jan M. Rabaey. *Digital integrated circuits: a design perspective*. Prentice-Hall, Inc., USA, 1996.

[SBHM20] Pascal Sasdrich, Begül Bilgin, Michael Hutter, and Mark E. Marson. Low-latency hardware masking with application to AES. *IACR TCHES*, 2020(2):300–326, 2020.

[Sha79] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

# A Probe-Extension of Other Gates

**Table 5:** Signal waveforms of NAND- and NOR-gates.

| S | $X$ | $Y$ | $Z$ |
|---|---|---|---|
| $S_0$ |  |  |  |
| $S_1$ |  |  |  |
| $S_2$ |  |  |  |
| $S_3$ |  |  |  |
| $S_4$ |  |  |  |
| $S_5$ |  |  |  |
| $S_6$ |  |  |  |
| $S_7$ |  |  |  |
| $S_8$ |  |  |  |
| $S_9$ |  |  |  |
| $S_{10}$ |  |  |  |
| $S_{11}$ |  |  |  |
| $S_{12}$ |  |  |  |
| $S_{13}$ |  |  |  |
| $S_{14}$ |  |  |  |
| $S_{15}$ |  |  |  |

**(a)** Signal waveforms of an NAND-gate.

| S | $X$ | $Y$ | $Z$ |
|---|---|---|---|
| $S_0$ |  |  |  |
| $S_1$ |  |  |  |
| $S_2$ |  |  |  |
| $S_3$ |  |  |  |
| $S_4$ |  |  |  |
| $S_5$ |  |  |  |
| $S_6$ |  |  |  |
| $S_7$ |  |  |  |
| $S_8$ |  |  |  |
| $S_9$ |  |  |  |
| $S_{10}$ |  |  |  |
| $S_{11}$ |  |  |  |
| $S_{12}$ |  |  |  |
| $S_{13}$ |  |  |  |
| $S_{14}$ |  |  |  |
| $S_{15}$ |  |  |  |

**(b)** Signal waveforms of an NOR-gate.

When comparing the scenarios outlined in Table 1 with those in Table 5, it becomes evident that they are symmetrically inverted. For instance, while $S_6$ and $S_9$ of Table 1a exhibit a positive glitch, the corresponding scenarios in Table 5a show a negative glitch. Consequently, it follows that $\mathsf{F}_Z^{\mathsf{AND}} = \mathsf{F}_Z^{\mathsf{NAND}}$ and $\mathsf{F}_Z^{\mathsf{OR}} = \mathsf{F}_Z^{\mathsf{NOR}}$ for all $Z$.

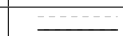
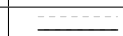
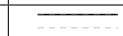
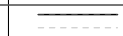
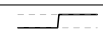
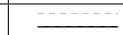
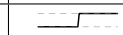
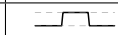
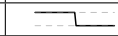
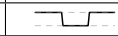
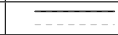
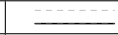
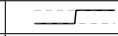
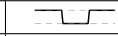
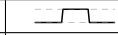
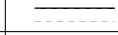
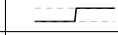
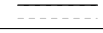
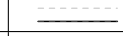
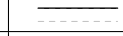
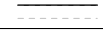
$$\begin{aligned}\mathsf{F}_Z^{\mathsf{NAND}}(X', X, G_X, Y', Y, G_Y) &= (X' \vee X \vee G_X) \wedge (Y' \vee Y \vee G_Y)\\ \mathsf{G}_Z^{\mathsf{NAND}}(X', X, G_X, Y', Y, G_Y) &= (X' \wedge \bar{X} \wedge \bar{Y}' \wedge Y) \vee (\bar{X}' \wedge X \wedge Y' \wedge \bar{Y})\\ &\quad \vee (G_X \wedge (Y' \vee Y \vee G_Y)) \vee (G_Y \wedge (X' \vee X \vee G_X))\end{aligned}$$

$$\begin{aligned}\mathsf{F}_Z^{\mathsf{NOR}}(X', X, G_X, Y', Y, G_Y) &= (\bar{X}' \vee \bar{X} \vee G_X) \wedge (\bar{Y}' \vee \bar{Y} \vee G_Y)\\ \mathsf{G}_Z^{\mathsf{NOR}}(X', X, G_X, Y', Y, G_Y) &= (X' \wedge \bar{X} \wedge \bar{Y}' \wedge Y) \vee (\bar{X}' \wedge X \wedge Y' \wedge \bar{Y})\\ &\quad \vee (G_X \wedge (\bar{Y}' \vee \bar{Y} \vee G_Y)) \vee (G_Y \wedge (\bar{X}' \vee \bar{X} \vee G_X))\end{aligned}$$

Again, Table 6a and Table 6b are symmetrically inverted yielding identical probe-extension and glitch-propagation functions for both XOR and XNOR gates. However, only $S_3$ and $S_{12}$, as well as $S_0$ and $S_{15}$, lack effective differentiation, effectively stopping the probe extension. Regarding glitch propagation, we assume that any input glitch invariably propagates to the output, akin to the robust $d$-probing model. Notably, we consider a doubled number of scenarios $\{S_5, S_6, S_9, S_{10}\}$ leading to glitches compared to the other gates.

$$\begin{aligned}\mathsf{F}_Z^{\mathsf{XOR}}(X', X, G_X, Y', Y, G_Y) &= ((X' \oplus X) \vee G_X) \vee ((Y' \oplus Y)) \vee G_Y)\\ \mathsf{G}_Z^{\mathsf{XOR}}(X', X, G_X, Y', Y, G_Y) &= (X' \oplus X) \wedge (Y' \oplus Y) \vee G_X \vee G_Y\end{aligned}$$

**Table 6:** Signal waveforms of XOR- and XNOR-gates.

| **S** | $X$ | $Y$ | $Z$ |
|---|---|---|---|
| $S_0$ |  |  |  |
| $S_1$ |  |  |  |
| $S_2$ |  |  |  |
| $S_3$ |  |  |  |
| $S_4$ |  |  |  |
| $S_5$ |  |  |  |
| $S_6$ |  |  |  |
| $S_7$ |  |  |  |
| $S_8$ |  |  |  |
| $S_9$ |  |  |  |
| $S_{10}$ |  |  |  |
| $S_{11}$ |  |  |  |
| $S_{12}$ |  |  |  |
| $S_{13}$ |  |  |  |
| $S_{14}$ |  |  |  |
| $S_{15}$ |  |  |  |

**(a)** Signal waveforms of an XOR-gate.

| **S** | $X$ | $Y$ | $Z$ |
|---|---|---|---|
| $S_0$ |  |  |  |
| $S_1$ |  |  |  |
| $S_2$ |  |  |  |
| $S_3$ |  |  |  |
| $S_4$ |  |  |  |
| $S_5$ |  |  |  |
| $S_6$ |  |  |  |
| $S_7$ |  |  |  |
| $S_8$ |  |  |  |
| $S_9$ |  |  |  |
| $S_{10}$ |  |  |  |
| $S_{11}$ |  |  |  |
| $S_{12}$ |  |  |  |
| $S_{13}$ |  |  |  |
| $S_{14}$ |  |  |  |
| $S_{15}$ |  |  |  |

**(b)** Signal waveforms of an XNOR-gate.

$$\mathsf{F}_Z^{\mathsf{XNOR}}(X', X, G_X, Y', Y, G_Y) = ((X' \oplus X) \vee G_X) \vee ((Y' \oplus Y)) \vee G_Y)$$
$$\mathsf{G}_Z^{\mathsf{XNOR}}(X', X, G_X, Y', Y, G_Y) = (X' \oplus X) \wedge (Y' \oplus Y) \vee G_X \vee G_Y$$