IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2024, No. 1, pp. 157–179. DOI:10.46586/tches.v2024.i1.157-179

Who Watches the Watchers: Attacking Glitch Detection Circuits

Amund Askeland[1,3], Svetla Nikova[1,2] and Ventzislav Nikov[4]

[1] University of Bergen, Bergen, Norway, firstname.lastname@uib.no
[2] COSIC, KU Leuven, Leuven, Belgium, firstname.lastname@esat.kuleuven.be
[3] Nasjonal Sikkerhetsmyndighet, Oslo, Norway
[4] NXP Semiconductors, Leuven, Belgium, venci.nikov@gmail.com

Abstract. Over the last decades, fault injection attacks have been demonstrated to be an effective method for breaking the security of electronic devices. Some types of fault injection attacks, like clock and voltage glitching, require very few resources by the attacker and are practical and simple to execute. A cost-effective countermeasure against these attacks is the use of a detector circuit which detects timing violations - the underlying effect that glitch attacks rely on. In this paper, we take a closer look at three examples of such detectors that have been presented in the literature. We demonstrate four high-speed clock glitching attacks, which successfully inject faults in systems, where detectors have been implemented to protect. The attacks remain unnoticed by the glitch detectors. We verify our attacks with practical experiments on FPGA.

Keywords: fault analysis · glitch detectors · clock glitching attacks

1 Introduction

In recent decades, electronic devices have become ubiquitous in everyday life. Some of these devices are made for applications where security is of obvious importance such as access control systems, smart cards, car keys, etc. The importance of security is, however, not limited to these types of systems. It is very common for devices to use cryptographic algorithms for securing communications, securing a boot process, and protecting firmware and other intellectual property from being accessible.

The security of these devices can be compromised by a range of different methods, one of which is Fault Injection Attacks (FIA). FIA is a range of attacks that are based on the attacker causing a fault to happen within the electronic device and exploiting the result of this fault. Examples of exploitable faults include incorrect instruction execution in microprocessors, corruption of data, or the release of incorrect output of cryptographic algorithms. As shown by the seminal paper "On the Importance of Checking Cryptographic Protocols for Faults" [BDL97], the release of incorrect ciphertexts or signatures can result in attackers recovering secret keys from asymmetric ciphers. Shortly after, a related type of attack against symmetric ciphers called Differential Fault Analysis (DFA) was published [BS97]. Faults can be introduced into a target device by a range of different fault injection methods which is an active research area in itself. Some of the early fault injection techniques included voltage and clock glitching and were used by the pay-tv hacking community in the 90s [AK96, AK97]. More advanced fault injection techniques have also been developed, such as optical fault injection [SA03], laser fault injection, and electromagnetic (EM) fault injection.

As a natural response to FIA, countermeasures have been developed. A simple countermeasure is duplication, where the same computation is performed twice in order to

Received: 2023-07-15 Accepted: 2023-09-15 Published: 2023-12-04

check for faults. Duplication can be effective but has the drawback of being vulnerable to duplicated faults and having a large cost. Various error detection/correction techniques have also been proposed to thwart FIA [BBK+03, KKG03, KKT04]. Malkin et al. [MSY06], demonstrated that many of the proposed schemes had costs comparable to that of simple duplication. More recently, countermeasures that protect against both side-channel attacks and FIA such as ParTI [SMG16] and M&M [DAN+18] have been presented.

An alternative approach to algorithmic countermeasures, usually with far lower overhead, is to use sensors for detecting physical attacks. The idea behind using detectors is that appropriate action, such as performing a reset or not releasing potentially faulty data, can be taken by the system if an attack is detected. Examples of such countermeasures include light sensors protecting against optical attacks, sensors for detecting rapid voltage changes [YPA08], and clock period monitors based on monitoring a critical clock by means of a faster clock. An important group of detection countermeasures is timing-violation detection circuits which have been described many times in the literature [SBGD11, ELH+12, ISYT13, GRG+14]. More recently Intel announced that such a detector is deployed in Intel Core processors starting from the 12th generation [NT22]. As several different fault injection methods rely on timing violations, these detectors can be suitable for detecting several types of attacks such as overclocking, underpowering, clock and voltage glitching [ZDCT13], and to some extent electromagnetic fault injection [ZDT+14]. These detection circuits have similarities with timing-issue detection mechanisms such as Razor [EKD+03] and canary logic [SK07]. Razor and canary logic are however not intended to protect against FIA, and in [KSYH11] the authors show how canary logic can fail to detect timing-violations.

Throughout this paper, we will discuss methods for injecting faults into a target circuit while avoiding detection from detectors. We will refer to these as attacks against the detection circuits. Since there are many possible targets of fault injection attacks, and the types of exploitable faults depend heavily on the type of target, we will consider these attacks successful if we are able to cause any faulty behaviour in the target circuit without detection.

In Section 2 we present the relevant theoretical background, including models for propagation delay in CMOS gates. In Section 3 we discuss the design of three closely related FIA countermeasures that detect timing violations. We point out that there are certain cases where these detectors can fail to detect injected faults, leading to attacks that can bypass these detectors. In Section 4, we perform practical experiments where we confirm that these attacks work against FPGA implementations of the detectors.

Our Contribution This paper presents our analysis of three glitch detection circuits that are designed to detect timing violations and thwart fault injection attacks. These detectors represent different variations of a common design concept, which involves comparing the output values of parallel delay lines. Our main contribution is the presentation of four different clock glitching attacks against these detection circuits. These specially crafted attacks can inject faults into a target device undetected by the detection circuit, thereby bypassing the countermeasure. We demonstrate two single-glitch attacks that use very short injected clock periods to bypass the detectors, as well as two double glitch attacks that offer the attacker more freedom in choosing the length of the glitched clock periods and enable different types of faults to be injected into the target device. By highlighting these attacks and the potential limitations of detection circuits, our work provides a valuable contribution to ongoing efforts to enhance the security of digital systems.

2 Preliminaries

2.1 Propagation Delay in CMOS

The speed of a digital gate, such as the inverter in Figure 1, is determined by the propagation delay. The propagation delay is largely due to stray capacitances such as capacitance between different conductors and gate capacitance. These capacitances can be modelled as a load capacitance connected to the output of the inverter. The time it takes for the output to change from low to high, t_{PLH}, is the time it takes to charge the load capacitance through the PMOS transistor. Similarly, the time it takes to switch from high to low, t_{PHL}, is the time it takes for the load capacitance to discharge through the NMOS transistor. These times can be approximated as in equations (1) and (2). In the equations, k_N and k_P are parameters of the transistors, while V_{TP} and V_{TN} are the threshold voltages for the PMOS and NMOS transistors, respectively [SB15].

$$t_{PLH} = \frac{C_L V_{DD}}{k_P (V_{DD} - V_{TP})^2} \tag{1}$$

$$t_{PHL} = \frac{C_L V_{DD}}{k_N (V_{DD} - V_{TN})^2} \tag{2}$$

From equations (1) and (2), we can see that the propagation delay increases with a higher load capacitance, and decreases with a higher supply voltage. Naturally, when we have larger gates or combine multiple gates together to form combinatorial logic, the propagation delay will increase.

Synchronous circuits can be formed by using flip-flops to hold the inputs and outputs of combinatorial circuits. A D-type flip-flop is a circuit that operates in such a way that the input (D) will be sampled at a rising clock edge and become the new value of the output (Q). A D-type flip-flop will have its own timing properties associated with it, and these can be modelled as the setup and hold time as well as the clock-to-Q delay, denoted T_{setup}, T_{hold}, and T_{clk2Q}. The setup and hold time is the time an input has to be stable before and after a clock edge in order to avoid meta-stability, respectively. Clock-to-Q delay is the time between a rising clock edge and a new stable value at the output of the flip-flop. Figure 2 shows a generic synchronous circuit, where m flip-flops hold the inputs to a combinatorial circuit that produces n outputs which are also held by flip-flops. In order for a synchronous circuit to behave correctly, the clock period T_{clk} has to satisfy inequality (3), where T_{comb} is the propagation delay of the combinatorial circuit.

$$T_{clk} > T_{clk2Q} + T_{comb} + T_{setup} \tag{3}$$

Larger digital circuits typically consist of many synchronous circuits, where the combinatorial circuit with the longest propagation delay will decide the maximum clock speed. This slowest path is referred to as the *critical path* of a design. This is a simplified model of the behaviour of synchronous circuits, and we will simplify it further by omitting T_{clk2Q} for the remainder of the paper.

2.2 Clock Manipulation

As discussed in the previous section, the clock period of a synchronous circuit, T_{clk}, has to satisfy inequality (3) in order to avoid incorrect computations. Faults can be injected into a system by increasing the clock frequency, thereby intentionally violating inequality (3). If the clock frequency is increased for a long period of time, this is referred to as *overclocking*. If only a single or a few clock cycles are manipulated, this is referred to as *clock glitching*. Clock glitching can be performed by an attacker who has access to manipulate the clock of the target device. One method for inserting a glitch into a clock signal is to generate a

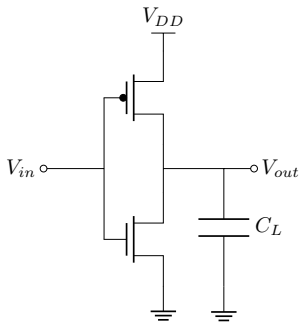


Figure 1: CMOS inverter.

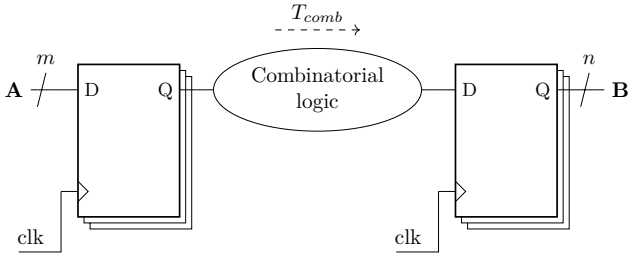


Figure 2: Timing in synchronous circuits.

pulse that is combined with the original clock signal. Figure 3 depicts this method with the relevant parameters. A clock signal with clock period T_{clk} can be manipulated by adding in a pulse with width W and offset O from a selected rising edge of the clock. This results in an extra inserted rising clock edge at time $T_G = O + W$ after the rising edge of the regular clock signal. The inserted clock edge results in an abnormally short clock cycle of length T_G that can cause faults through timing violations, followed by another short cycle of length $T_{clk} - T_G$. The outputs of combinatorial circuits typically change multiple times before they reach stable values. Because of this, the value of T_G is likely to have a significant effect on the type of fault that is injected. For example, in microprocessors, different types of faulty instructions can be executed depending on T_G [BGV11, KHEB14].

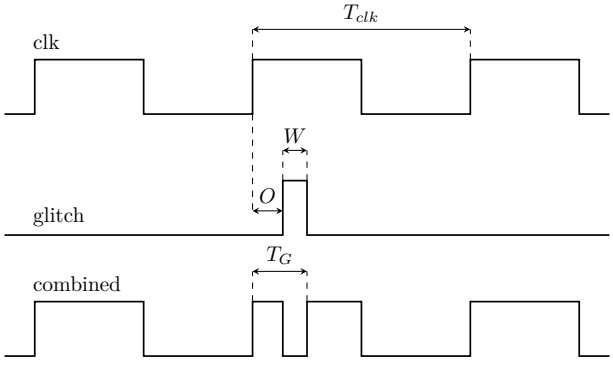


Figure 3: Glitch inserted in the high part of a clock period.

In Figure 3, the glitch pulse was added during the high part of a clock period. Alternatively, the pulse can also be added during the low part of the clock signal as depicted in Figure 4. In this case, we choose to define $T_G = (T_{clk} - O)$, in order for T_G to represent

the distance from the rising edge of the inserted glitch to the nearest rising edge of the regular clock signal.

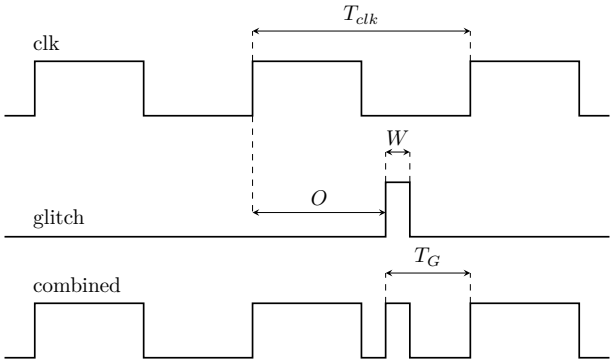


Figure 4: Glitch inserted in the low part of a clock period.

As noted before, an inserted glitch with the method from Figures 3 and 4, will result in two abnormally short clock periods. For an attacker, this might cause complications if the intention is to only cause one timing violation. Two possible solutions to this are to either use a base clock frequency that is much larger than the delay of the critical path or to remove a clock pulse from the original clock signal.

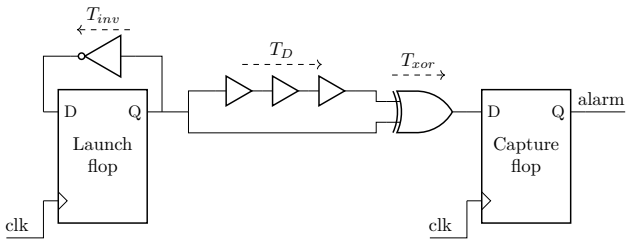
3 Detection Circuits

Since a wide range of fault injection methods work by causing timing violations, it makes sense to build countermeasures that work by detecting these violations. Circuits that raise an alarm flag if they detect timing violations can be made with few resources and can be included as sub-circuits in larger designs, thus offering protection against fault injection attacks at a very low cost compared to algorithmic countermeasures. The security and resilience of such detection circuits is under-represented in the literature, and in this section, we take a closer look at a set of detectors and ways in which they can fail.

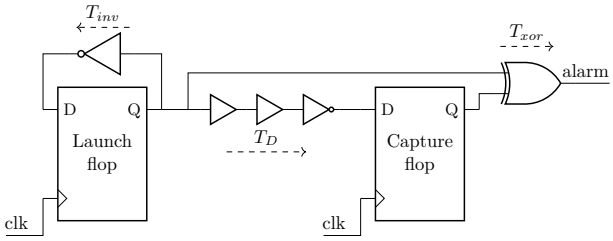
3.1 Parallel Delay Lines

A common design for timing violation detectors is the parallel delay lines circuit. This circuit consists of two parallel paths with different propagation delays. One such design is shown in Figure 5a. In the circuit, a D-type flip-flop, referred to as the launch flop, inverts its output on every rising clock edge. This output is connected to two paths, one with very low propagation delay and one with intentionally high propagation delay. The high propagation delay path is represented by a chain of buffers in the figure. The ends of these two paths are connected to an XOR gate that compares the value at the end of the two paths. The output of the XOR gate is sampled by another flip-flop, referred to as the capture flop. If two rising clock edges are too close, the output value of the launch flop will not have had enough time to propagate through the long delay path and the XOR gate will compare two different values. This will cause the alarm to be raised which is how this circuit can detect fault injections caused by timing violations. The propagation delay of the long delay path, T_D, can be adjusted by adding more or fewer gates in the path. In this way, the detector can be tuned to match the critical path delay of other parts of the device so that the alarm is raised if any timing violations occur. This type of circuit is sometimes referred to as a replica circuit, because the long delay path aims to replicate, or to be slightly longer, than the critical path of a circuit that should be protected.

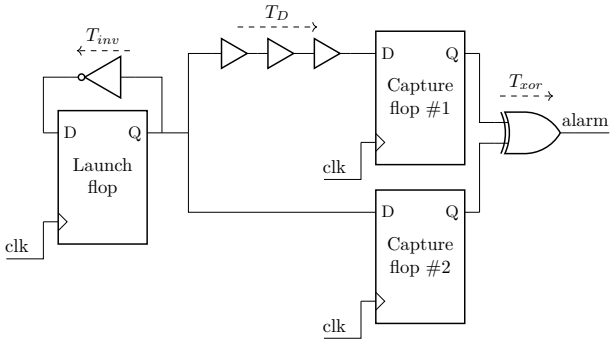
This type of design has been described several times in the literature with some slight variations. In [GRG+14], the authors present several circuits designed to detect timing violations and evaluate them for detection of voltage glitching attacks. One of their designs is equivalent to the one in Figure 5c. Another is equivalent to the design in Figure 5b, except for the position of the inverter. In [SBGD11], the authors perform clock glitching attacks on an FPGA and suggest the design in Figure 5b as a possible countermeasure. In [ISYT13] and in a white paper by Intel [NT22], a design equivalent to the circuit in Figure 5a is presented. In the Intel design, the long delay path is configurable. This change allows the threshold at which the alarm activates to be adjustable to account for manufacturing variations, and it is explained that setting the delay length is part of a calibration step in the manufacturing process. This design is referred to as the "Tunable Replica Circuit".



(a) PDL-1



(b) PDL-2



(c) PDL-3

Figure 5: Three variants of the parallel delay line circuit.

3.2 False Negative Attacks

The result of an attempted fault injection attack against a target device with a glitch detector can be divided into four categories. If a fault is injected and an attack is detected, the event is categorised as a positive. If no fault is injected and an attack is undetected, the event is categorised as a negative. If no fault is injected and an attack is detected, the

event is categorised as a false positive. Finally, if a fault is successfully injected and an attack is undetected, the event is categorised as a false negative.

For any glitch detector, it is very important that successful fault injections are detected, and false negatives should be avoided to the extent possible. If an attacker can reliably cause false negative events, the detection mechanism can effectively be bypassed.

3.2.1 Single Glitch Attacks

Attack 1 Attack 1 is based on causing the launch flop output to not invert on a rising clock edge, and is applicable to PDL-1 and PDL-2. All the detector circuits in Figure 5 use a launch flop with a feedback inverter such that the launch flop output inverts on every rising clock edge. We note that if two rising clock edges are so close that the output of the launch flop does not have time to propagate through the feedback inverter, the launch flop will not change its output on the second rising clock edge. For PDL-2, this will directly result in no alarm being raised even though we have a clear timing violation. For PDL-1, there will also not be any raised alarms, since the change in the launch flop output does not have time to propagate through the XOR gate. More formally, if we have a glitched cycle such that $T_G < T_{inv}$, the launch flop output will not invert, and if $T_G < T_{inv} + T_{setup}$, it might invert. We refer to this attack as attack 1. Figure 6 shows a clock signal with an inserted glitch such that $T_G < T_{inv}$, together with the resulting values of the launch flop input (LD) and output (LQ). We can see that the latest launch flop output has not yet propagated through the inverter when the extra rising edge is inserted, and therefore the launch flop output is not updated at this edge.

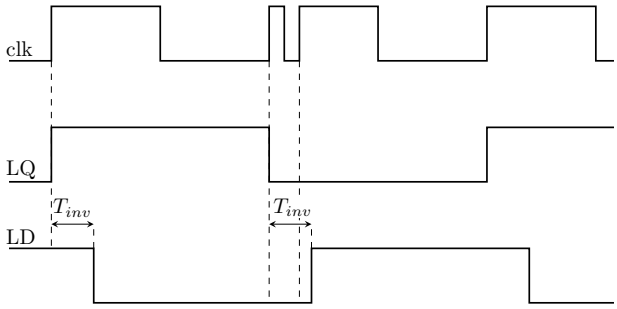


Figure 6: Timing diagram for attack 1. From top to bottom: Clock, launch flop output, launch flop input.

For PDL-3, this attack will not work unless the delay between the launch flop and capture flop #2 is sufficiently large. As there is a direct connection between these two flip-flops, we don't consider the attack applicable to PDL-3.

Attack 2 For PDL-1 in Figure 5a there is no direct connection between the launch flop and the capture flop. This means that a change in the launch flop output will have to propagate through the XOR gate before it is noticed at the capture flop input. Attack 2 is based on having two rising clock edges so close together, that a change in the launch flop output at the first edge has not yet been noticed at the capture flop at the time of the second edge. This attack is applicable to PDL-1.

For PDL-1, the input to the capture flop is the output of the XOR gate. After a rising clock edge, the first change in the input value of the capture flop will be after T_{xor}. We note that if $T_{inv} + T_{setup} < T_G < T_{xor}$, the launch flop will be updated, while the capture flop will still sample a low value, i.e. 0, such that no alarm will be raised even though this is a clear timing violation. In order for this to be possible, we need that $T_{inv} < T_{xor}$. In CMOS this is common, as an inverter is typically the fastest gate with only two transistors

while an XOR gate has a higher delay. Figure 7 shows a timing diagram for this attack where we see a regular clock cycle to the left before we have a glitched clock cycle. We can see that the capture flop input is low at all rising clock edges, and therefore the alarm will not be activated.

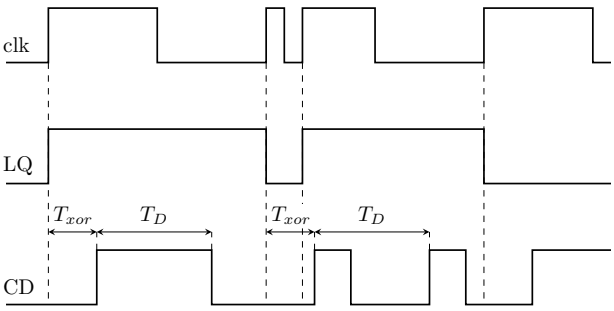


Figure 7: Timing diagram for attack 2. From top to bottom: Clock, launch flop output, capture flop input.

3.2.2 Double Glitch Attacks

We have seen how attack 1 and attack 2 can be used to cause timing violations that go undetected by PDL-1 and PDL-2. Both these two attacks are based on inserting a clock edge such that we get a very low value for T_G, meaning there is very little room for choosing appropriate values for T_G. Since the types of faults that are injected can depend heavily on T_G, it is desirable for an attacker to have more freedom in choosing this value. We will now discuss how this can be achieved with double glitch attacks.

We extend the concept of combining glitches with a clock signal from Figure 3 so that we have two glitches instead of one. If we add a requirement that $O_2 > O_1 + W_1$, then we will have three rising edges close together. The first is the normal clock edge, the second comes at time T_{G1} after the normal clock edge, and the third at time T_{G2} after the normal clock edge as shown in Figure 8. The clock signal now has two short clock periods of length T_{G1} and $T_{G2} - T_{G1}$.

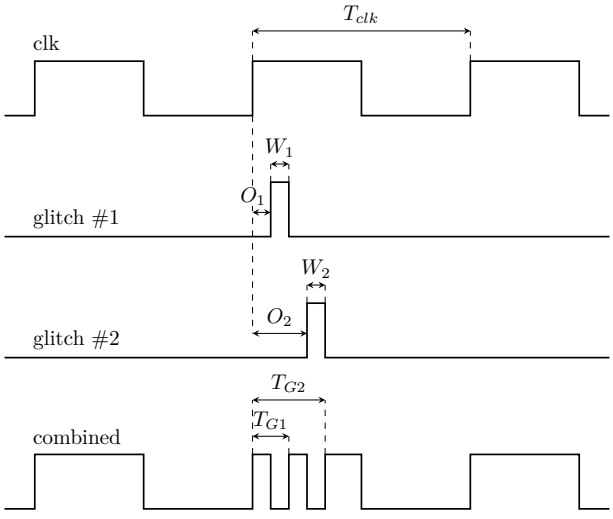


Figure 8: Clock signal with two inserted glitches.

Attack 3 The alarm output of PDL-2 and PDL-3 is driven by an XOR gate and is not held by a flip-flop as opposed to PDL-1. The papers describing the design of PDL-2 and PDL-3 do not specify how to sample the output of the XOR gate. We will assume that the output of the XOR gate is synchronously sampled, for example, by a D-type flip-flop. We believe this to be a natural assumption because the XOR output should be held for an entire clock cycle. If asynchronous logic, such as a latch, is used to sample the XOR output, issues could arise due to clock skew or different delays between the two flip-flops and the XOR gate. The result of this assumption is that the alarm from PDL-2 and PDL-3 will be activated only if the XOR output is high during a rising clock edge. The basic idea of attack 3 is that the attacker first applies a glitch that causes the desired fault, but also causes the two inputs to the XOR gate to take different values. A second glitch is then applied before the changes have propagated through the XOR gate, and at the same time causing the inputs to the XOR gate to hold equal values again. This attack is applicable to PDL-2 and PDL-3. If we have T_{G1} such that $T_{G1} < T_D$, the two flip-flops holding the inputs of the XOR gate will sample different values. This means that the output of the XOR gate will be raised at time T_{xor} after the glitched clock edge. A second glitch is then applied before the XOR output is raised, i.e. $T_{G2} - T_{G1} < T_{xor}$. This second glitch also has to put the two flip-flops back in a state where they hold the same value. This will be the case for PDL-3 if $T_{G2} < T_D$, and for PDL-2 we have the additional requirement that the launch flop has to invert, i.e. $T_{inv} + T_{setup} < T_{G2} - T_{G1}$. Figure 9 shows a timing diagram for this attack when applied to PDL-3, we can see that even though the alarm signal is high for a brief moment, it is never high during a rising clock edge.

To some extent, attack 3 can also be achieved with just a single inserted glitch. This can happen if the glitch is inserted in the low region of the clock signal as shown in Figure 4, and the clock frequency is just right so that the attack can be achieved with the two resulting short clock periods of length T_G and $T_{clk} - T_G$. The downside to this is that it will only work if the detection threshold is tuned very close to the critical path delay, and the attacker has very little room to select the values of the short clock periods. We also note that adding two glitches to a clock signal is not much more difficult than adding a single glitch.

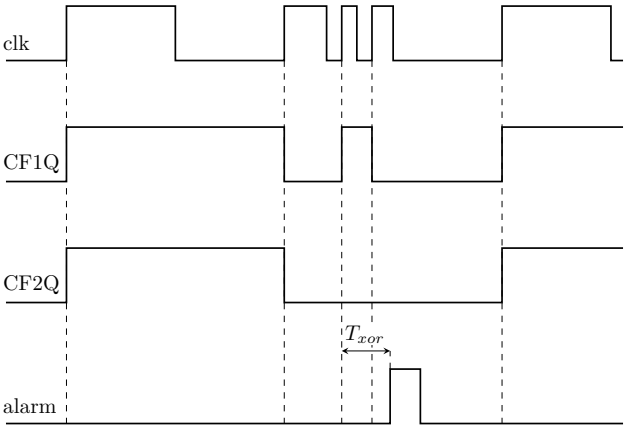
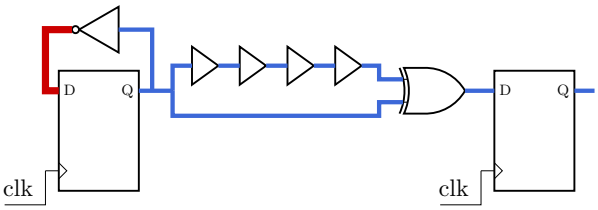


Figure 9: Timing diagram for attack 3. From top to bottom: Clock, capture flop #1 output, capture flop #2 output, alarm output.

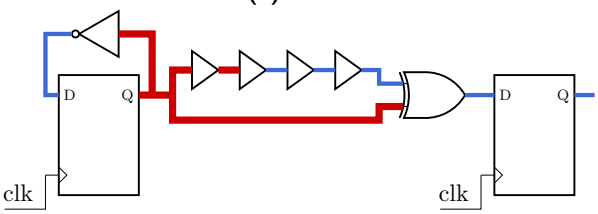
Attack 4 Attack 4 is a double glitch attack applicable to PDL-1 that can be seen as an extension of attack 2. As for attack 2, the first glitch is applied at a time such that $T_{inv} + T_{setup} < T_{G1} < T_{xor}$. In the timing diagram for attack 2 in Figure 7 we can see that after the inserted glitch, the XOR output, or equivalently the capture flop input, is

high between T_{xor} and $T_{G1} + T_{xor}$ as well as between $T_D + T_{xor}$ and $T_{G1} + T_D + T_{xor}$. This means that a second glitch can be applied between $T_{G1} + T_{xor}$ and $T_D + T_{xor}$ without being detected, which is how attack 2 is extended to attack 4. The advantage of attack 4 over attack 2, is that the first glitch enables the attacker to place a second glitch in a large area in time without detection. This can be useful when attempting to trigger specific behaviour in the target circuit.

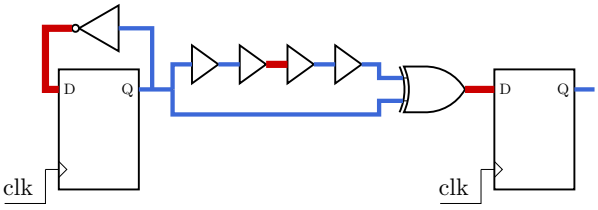
In Figure 10, we have drawn the internal signals of PDL-1 using blue for low and thick red for high. Figure 10a shows the internal signals when the circuit has reached a steady state after the previous rising clock edge. We have arbitrarily chosen a starting point of a low launch flop output. Figure 10b shows the internal signals shortly after a rising clock edge, when the change in the launch flop output has had time to propagate through the feedback inverter, but not yet through the XOR gate. At this point in time, a glitch is inserted corresponding to attack 2. Figure 10c shows the signals shortly after the glitch, where the XOR output will be high for a brief moment. Note that there is a high pulse of width T_{G1} propagating through the long delay path. Figure 10d shows the internal signals as they would be T_{xor} after the inserted glitch, and before the high pulse has propagated through the long delay path and the XOR gate. This period represents the time when a second glitch can be added without causing the alarm to be raised.



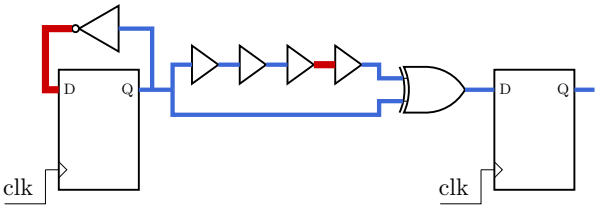
(a) $t = 0$



(b) $t = T_{G1}$ and $T_{inv} < T_{G1} < T_{xor}$



(c) $T_{xor} < t < T_{G1} + T_{xor}$



(d) $t = T_{G2}$ and $T_{G1} + T_{xor} < T_{G2} < T_D + T_{xor}$

Figure 10: Internal signals of PDL-1 during attack 4 at times, t, relative to a clock edge.

Summary The different clock glitching attacks and the timing conditions for the inserted glitches are summarised in Table 1.

Table 1: Summary of the different attacks.

Attack	Timing conditions	Applicable to
Attack 1	$T_G < T_{inv}$	PDL-1 and PDL-2
Attack 2	$T_{inv} + T_{setup} < T_G < T_{xor}$	PDL-1
Attack 3	$T_{G1} < T_D$ and $T_{G2} - T_{G1} < T_{xor}$ Additionally, for PDL-2: $T_{inv} + T_{setup} < T_{G2} - T_{G1}$	PDL-2 and PDL-3
Attack 4	$T_{inv} + T_{setup} < T_{G1} < T_{xor}$ and $T_{G1} + T_{xor} < T_{G2} < T_{xor} + T_D$	PDL-1

4 Experiments

In Section 3.2 several situations that could lead to false negatives, i.e. the detector failing to detect actual timing violations, were described. In this section, we perform practical experiments in which we attempt to execute the attacks.

4.1 Setup

4.1.1 Target Device

We use the CW305-A100 Artix-7 FPGA board from NewAE Technology [New18] as a platform for our target implementation. We load the FPGA with two implementations of the Advanced Encryption Standard (AES), and the three detector designs from Figure 5 simultaneously. This enables us to perform the experiments across all glitch detectors concurrently, subjecting them to the same operational conditions. The interface to the target device is such that the key and plaintext can be chosen before each encryption, and after an encryption we can read the resulting ciphertexts as well as whether any timing violations where detected. For any attempted attack, we use the correctness of the ciphertext and the detection status to categorise the results into (false) negatives/positives as mentioned earlier. This will be done separately for every combination of AES and detector implementation. We choose to use AES as an example target, and we use the correctness of the ciphertext to determine if a fault was successfully injected or not. Our main goal with the AES target is to observe whether or not faults have been injected, but we also investigate the occurrence of faulty ciphertexts that can be used in DFA.

AES Implementations We use two different implementations of AES-128 on the target device, a round-based design with a 16-byte datapath and a serialised design with a single byte datapath. Both implementations use the s-box design from [BP12], which is a fairly complex combinatorial circuit that is part of the critical path for both implementations. The round-based design uses 11 clock cycles to complete an encryption, while the serialised design uses 200 clock cycles. The two implementations are run in parallel, but the round-based design is delayed by 150 clock cycles such that they are both actively encrypting at the same time. Unless specified, experimental results are on attacks against the round based design. We use two different types of AES implementations in order to observe whether the conditions for which they fail are dependent on the architecture.

Detection Circuit Implementations We use the Artix-7 FPGA as our target device, which organizes the main FPGA primitives, flip-flops and lookup tables (LUTs), into units called slices. Each slice consists of four 6-input LUTs and eight flip-flops. We implement all three variants of the PDL timing violation detectors from Figure 5 on the target FPGA.

Each of the various logic gates in the detector circuits is mapped to a LUT on the FPGA, and the flip-flops are naturally mapped to the FPGA flip-flops. It's worth noting that the flip-flop primitives in this device lack a complementary output that could substitute the feedback inverter. For increased repeatability and control, we manually assign each component to primitives in a single slice per detection circuit, except for the long delay path, which is allocated to the surrounding slices. This long delay path is constructed by utilizing LUTs configured as buffers, interconnected in a chain. To make the delay configurable, we employ additional LUTs configured as multiplexers, enabling the selection of any point within the buffer chain as an output. This approach allows us to modify the value of T_D, without altering the overall implementation. In order to conduct initial testing of the detector designs, we measure the detection threshold for all the delay settings. For each adjustment setting of T_D, we apply a glitch and observe the values of T_G at which the detector triggers an alarm. The results of this test are presented in Figure 11. The figure demonstrates that the detectors can be adjusted to raise alarms for short clock periods ranging from 2 ns to 10 ns in a roughly linear manner.

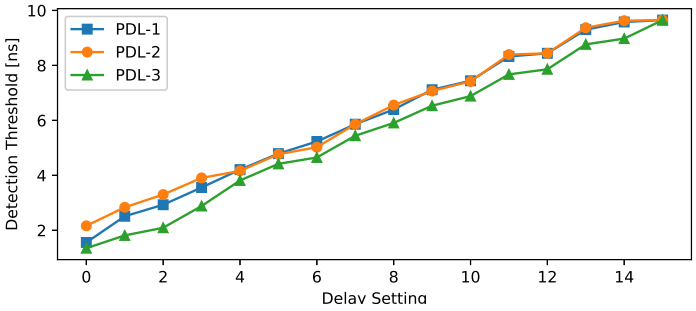


Figure 11: Detector delay settings vs. alarm threshold.

The design tool used for the implementation, Xilinx Vivado, can be used to report estimates on the various delays. Table 2 shows the reported delay for the paths through the inverter and XOR gates of the three detector designs. Note that since both the inverters and XOR gates are implemented in the FPGA using LUTs one might expect quite similar delays for the respective paths, while in the table we see that the XOR-delays are slightly higher. The reason for this lies in the delay of the nets, which is slightly more complicated for the XOR gates. The feedback inverters only have one input and one output, and the inverter LUTs and the launch flops are placed very close together. This might explain the slight difference in the timings.

Table 2: Estimates on inverter and XOR delay for the detectors as reported by Vivado.

	T_{inv} (ns)	T_{xor} (ns)
PDL-1	1.471	1.919
PDL-2	1.463	1.911
PDL-3	1.455	1.832

Because attack 1 and attack 2 are differentiated by whether or not the launch flop was inverted by the inserted glitch, we have also added a shadow flip-flop to the launch flop of the detectors. The output of the launch flop is connected to the input of the shadow flip-flop, and we can therefore compare the values held by the two flip-flops in order to see if the launch flop has held two equal values in a row.

4.1.2 Clock Glitch Generator

The realisation of the attacks from Section 3.2 requires the generation of a clock signal with very short pulses, and the ability to generate several independent glitches for the double glitch attacks. We implement a clock glitch generator based on Xilinx 7-series FPGAs using an approach similar to that described in [ESH+11] and [ADN+10]. The glitch generator uses several Mixed-Mode Clock Manager (MMCM) modules, which is a hardware module available in some Xilinx FPGAs. We use the MMCMs to create phase-shifted versions of a clock signal where the phase shift can be controlled dynamically. Our glitch generator follows the concept from Figure 3. We use two phase-shifted clock signals that we combine in order to create a signal with adjustable phase and pulse width. This signal can be enabled for a single clock cycle in order to form a pulse with adjustable offset and width. The glitch pulse is then combined together with the original clock signal using a logic function. We end up with a clock glitch generator which outputs a 12 MHz clock signal and which has a trigger input. Two glitches can be controlled independently in width, phase and clock cycle delay. The clock cycle delay decides how many clock cycles after the trigger the glitch is inserted into the 12 MHz clock signal. The width and phase of the glitches can be controlled in 3584 steps between 0 and $1/12 \cdot 10^{-6}$ s, corresponding to a step size of roughly 23 ps. For ease of use, the glitch generator has a simple serial interface which can be controlled for example by a python script.

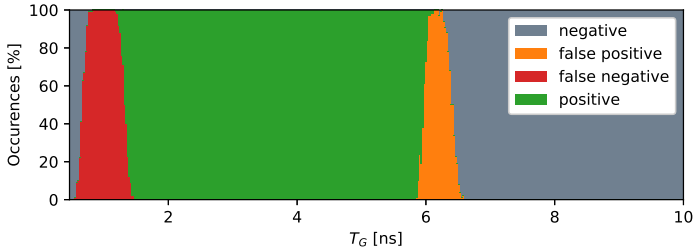
High Speed Challenges In order to be able to execute the attacks described in Section 3.2, glitchy clock signals that have several rising edges in a very short period of time have to be generated. While the glitch generator can control the width and position of a glitch in very small steps, there are practical limitations. Various analog effects such as stray capacitances effectively create a low-pass filter that limits how fast we are able to change the logic level of the clock signal. This in turn results in a lower limit for how close together two rising clock edges can be. These analog effects are heavily dependent on the physical path, or channel, from the glitch generator to the target. We choose to use two versions of the clock glitch generator. One that is implemented on the same FPGA chip as the target circuit, and one that is implemented on a separate FPGA board. The internal generator will be used to explore the different attacks, without the added challenges of the analog effects of the channel between the boards. The external generator obviously represents a more realistic scenario and will be used to show that the attacks do not rely on using an internal generator. We expect that the performance of the external clock glitch generator can be improved with a mixture of a better channel between the boards and a generator board with better high-speed output capabilities.

4.2 Attacks Using an Internal Glitch Generator

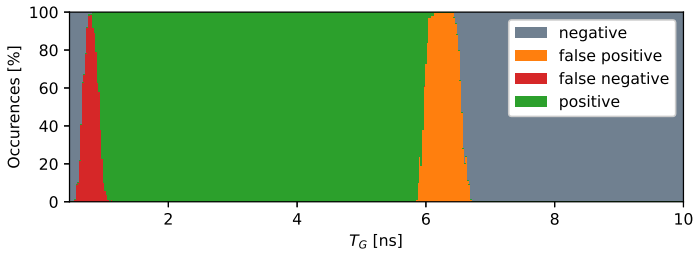
We conducted a series of experiments in which we introduced additional glitches to a clock signal during the 9th round of the round-based AES implementation. The internal glitch generator was used in these experiments. We used a range of values for the position of the glitches, performed 100 encryptions for each glitch position and recorded the correctness of the resulting ciphertext as well as the state of the alarms. The reason for performing multiple repetitions of each experiment was to demonstrate the repeatability of the results, given that we are dealing with very short time periods and subtle effects that can potentially influence the outcomes. We will use stacked bar plots, where the results of the 100 repetitions are divided into categories represented by different coloured bars that are stacked on top of each other.

4.2.1 Single Glitch Attacks (Attack 1 & 2)

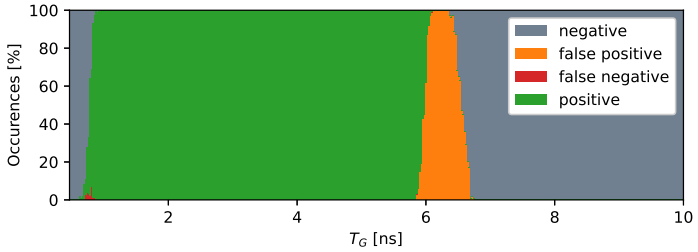
The results of a single glitch applied to the detection circuits are shown in Figure 12. The figures show that when T_G is greater than approximately 6.5 ns, the glitch has no impact. Between approximately 5.8 ns and 6.5 ns, there is a region of false positives, which occurs because the glitch detectors raise an alarm before the AES implementation fails. For lower values of T_G, faults are successfully injected and the ciphertext become faulty. As T_G becomes very small, false negatives begin to appear, consistent with attack 1 and attack 2 described earlier. Notably, PDL-1 is impacted by both attack 1 and attack 2, while PDL-2 is only affected by attack 1. This is reflected in the larger range of T_G values causing false negatives for PDL-1. We can also see a few false negatives for PDL-3. As discussed in Section 3.2 this can be explained by a slight delay between the launch flop and capture flop #2. At the far left of the figures there is a narrow region where the ciphertext is not faulty and the alarm is not raised. This region is likely there because the glitch is effectively filtered away due to analog effects when T_G becomes too small.



(a) PDL-1



(b) PDL-2



(c) PDL-3

Figure 12: Single glitch attacks.

4.2.2 Differentiating Attack 1 & 2

Given that PDL-1 is affected by both attack 1 and attack 2, it is important to be able to distinguish between these two events. The key difference between the two cases is that for attack 1, the launch flop is not inverted by the glitch, whereas for attack 2 it is. To differentiate the two situations, we will use a shadow flip-flop in PDL-1 as mentioned

earlier. Figure 13 displays the false negatives from Figure 12a, where they have been split into attack 1 and attack 2 events. We can see that we can reliably cause both attack 1 and attack 2 events. We also see that successful attacks only happen for values of T_G below roughly 1.4 ns, which is lower than the expected 1.919 ns from Table 2. Reasons for this discrepancy could be that the timing estimates from Vivado are conservative, or that our values of T_G have a systematic error.

Note that we see a portion of attack 2 events to the far left in the figure. These are likely there because the shadow flip-flop of the launch flop does not always sample the correct value when T_G gets too small, and these events are thus not really attack 2 events. Although there is a direct connection between the two flip-flops, there will be a slight delay in wiring and the setup time of the shadow flip-flop could be violated.

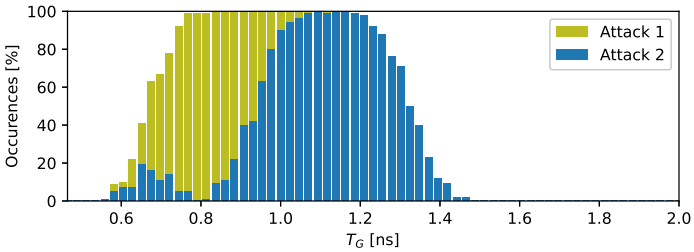


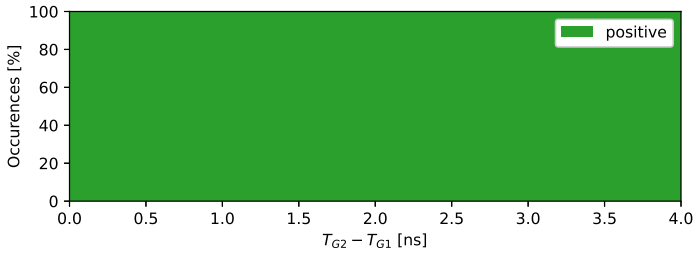
Figure 13: Differentiating between attack 1 and attack 2 for PDL-1.

4.2.3 Attack 3

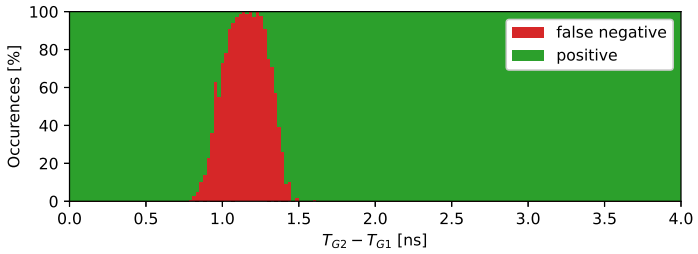
We apply a double glitch attack where we fix T_{G1} at 4 ns, which will cause the ciphertexts to become faulty, and normally also the alarms to be raised as we saw in Figure 12. We then apply another glitch and perform a set of experiments with different distances between the two glitches. Figure 14 shows the results of this experiment. Here we can see that when the glitches are sufficiently close together, PDL-2 and PDL-3 stops reporting the glitch, meaning that attack 3 has been successfully applied. At the far left in the figures we have a section where the alarm is activated, this is because the two glitches are too close together and analog filtering effects effectively turn the two glitches into a single glitch. We can also see that PDL-1 detects the glitches regardless of the distance between them, which is expected since this attack is not applicable to PDL-1. Note that the region of false negatives happens slightly later for PDL-2, which is consistent with the requirement for PDL-2 that the launch flop has to invert its value for attack 3 to work against this detector. Also note that the region of false negatives ends slightly earlier for PDL-3 than PDL-2, which can be explained by Table 2 where we can see that there is a difference in T_{xor} between the two detectors.

4.2.4 Attack 4

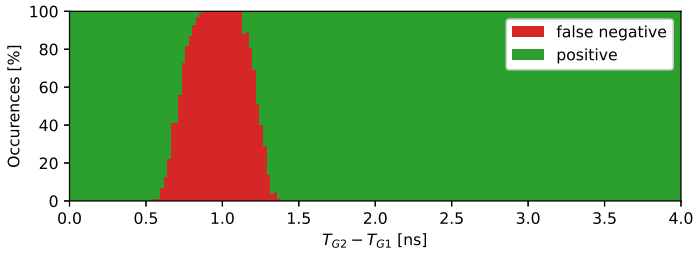
We apply a double glitch attack where we fix T_{G1} at 1.1 ns. This value is chosen based on Figure 13, and is meant to maximise the probability of causing a successful attack 2, which is necessary for attack 4 to work. We then perform a set of experiments where we sweep the position of the second glitch, and record the results. The results are presented in Figure 15, and seem consistent with the expected behaviour explained in Section 3.2.2. To the left in the figure, we have a period where the detector alarm is active, which corresponds to the time between T_{G1} and $T_{G1} + T_{xor}$. Following this, we have an area with false negatives corresponding to the time between $T_{G1} + T_{xor}$ and $T_D + T_{xor}$. Around 7 ns we can see a region where the alarm is active, which should correspond to the T_{G1}-wide pulse that propagates through the long delay path as shown in Figure 10. To the right in the figure



(a) PDL-1



(b) PDL-2



(c) PDL-3

Figure 14: Attack 3 applied to all detectors.

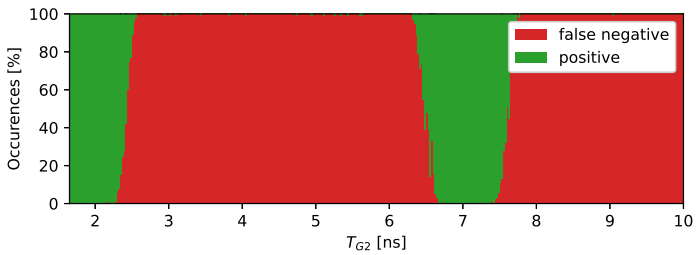


Figure 15: Attack 4 applied to PDL-1.

there is another region of false negatives, which are caused by the first glitch that is at a fixed position. In this region, the second glitch is far enough away from any other rising edge to not cause any timing violations or alarms by itself.

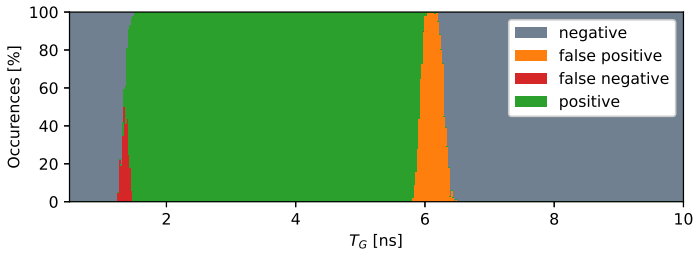
4.3 Attacks Using an External Glitch Generator

We repeated the experiments from the previous section, this time with an external FPGA board supplying the glitchy clock signal to the target board. The clock signal is shared between the boards through a coaxial cable. As noted earlier, this makes it harder to achieve very short glitched clock periods.

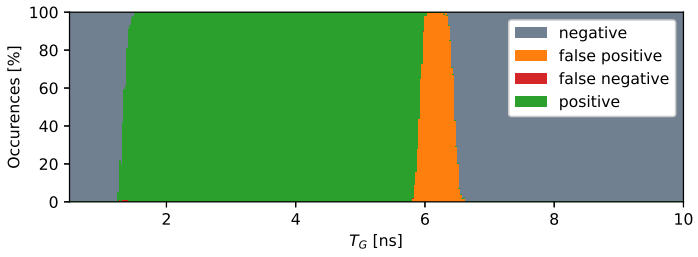
4.3.1 Attack 1 & 2

Figure 16 shows the results when we performed a single glitch attack in the same way as in Figure 12, except that we used the external generator. As we can see, the rate of false negatives have dropped down to a maximum of roughly 50% for PDL-1 and close to zero for PDL-2. This happens because the lower limit of T_G that we can achieve with our external glitch generator is higher than for the internal generator. In order to increase the success rate, we can lower the core voltage of the target device. From equations (1) and (2) we can see that lowering the supply voltage will result in higher propagation delays, thus making it easier to apply our attacks.

Figure 17 shows the repetition of the experiment, but where the target device has a core voltage of 0.93 V as opposed to the nominal 1.0 V. Here we can see that the rate of false negatives have increased to almost 100% for PDL-1. Note also that both the area of positives and false positives have stretched out in time due to the increased propagation delays. Since the rate of false negatives for PDL-2 is still close to zero, the false negatives in PDL-1 are likely attack 2 events.



(a) PDL-1

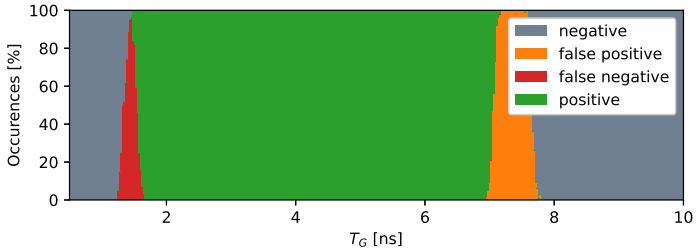


(b) PDL-2

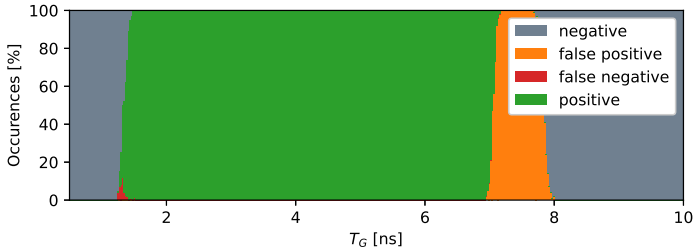
Figure 16: Single glitch with external generator, nominal target voltage.

4.3.2 Attack 3

We fix the position of the first glitch such that $T_{G1} = 4\,\text{ns}$, like before, and sweep the position of the second glitch. Again, we have used a target core voltage of 0.93 V in order



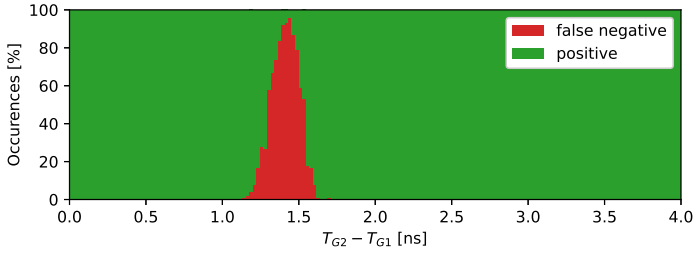
(a) PDL-1



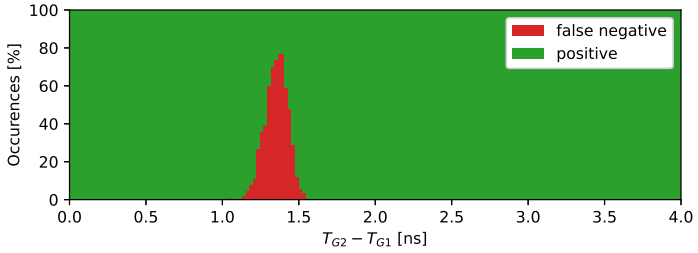
(b) PDL-2

Figure 17: Single glitch with external generator, low target voltage.

to increase the propagation delays. Figure 18 shows the results of this experiment. We note that compared to the same experiment performed with an internal glitch generator, shown in Figure 14, the peak of the false negative regions are shifted slightly to the right due to the increased propagation delay. We also note that the left edge of the false negative regions seems to be in the same position for PDL-2 and PDL-3 in Figure 18, although that was not the case for the internal glitch generator. This is consistent with the minimal distance between two rising edges that we can practically achieve being larger than $T_{inv} + T_{setup}$, as we also saw in the attempt of applying attack 1 to PDL-2 in Figure 17b.



(a) PDL-2



(b) PDL-3

Figure 18: Attack 3 applied to PDL-2 and PDL-3 using an external glitch generator and low target voltage.

4.3.3 Attack 4

Figure 19 shows attack 4 applied to PDL-1 using the external glitch generator and a target core voltage of 0.93 V. We can see that we do not reach 100% rate of false negatives, which is due to the first applied glitch not always causing a successful attack 2, as we also saw in Figure 17a. Note the small area of false positives just below 8 ns. This happens when the first glitch and the clock edge become one single edge. Normally this results in detection, but in this specific region the distance between the second glitch and the clock edge is too large to cause faulty ciphertexts, yet small enough to cause alarms.

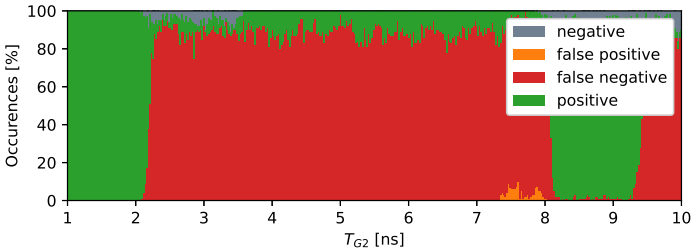


Figure 19: Attack 4 applied to PDL-1 using an external generator and low target voltage.

4.4 Faults Suitable for DFA

The various attacks all rely on using very short glitched clock periods that might affect large parts of the targeted device. This suggests that the faults produced by the attacks are coarse and best suited for things like disrupting a microprocessor's instruction execution, and not for causing the single-byte faults commonly used in DFA. In order to assess the suitability of the attacks for DFA, we perform some attempted attacks against the serialised implementation of AES. The focus is on this implementation due to its single-byte datapath, which should make it easier to affect small parts of the implementation with an attack. We count the number of single-byte faults in round 8, since two faulty ciphertexts of this type are usually enough for a full key recovery using DFA [PQ03].

We perform a set of experiments where we try out the various combinations of clock cycle and glitch phase(s) in order to find a suitable set of parameters for the attacks. Using the parameters that maximise the number of single-byte faults in round 8 of AES for each attack, we attempt to fault 10000 encryptions. The results of these experiments are shown in Table 3 along with the parameters used. These experiments used the internal glitch generator, and the encryptions used a fixed key and random plaintexts for each of the 10000 experiments. In these experiments, we found no set of parameters for attack 2 or attack 4 that caused single-byte faults in round 8. Note that for attack 3 none of the 245 exploitable faults were undetected by PDL-2, the value of $T_{G2} - T_{G1}$ is likely too low for the additional constraint for PDL-2 from Table 1 to be satisfied.

Table 3: Number of faulty ciphertexts suitable for DFA out of 10000 glitched encryptions.

Attack	Clock cycle	T_G /T_{G1}	$T_{G2} - T_{G1}$	Faults	Bypassed detectors
Attack 1	170	0.67 ns	-	12	PDL-1 and PDL-2
Attack 3	161	3.77 ns	0.80 ns	245	PDL-3

5 Discussion

In Section 4, we found that the attacks described in Section 3.2 can be performed in practice for a target implemented on an FPGA. All the attacks require a glitched clock signal that

has two rising edges so close together that the timing requirements of a single-gate path are violated. The generation of a glitchy clock signal that switches fast enough, and the bandwidth of the channel for this signal, can make the attacks challenging to apply. We saw how under-powering the target device can partly compensate for these challenges, and we also believe that the performance of the external glitch generator can be improved. Faults caused by timing violations can also be injected through voltage glitching, which is sometimes mentioned as equivalent to clock glitching. The specific attacks presented here require fine control and are not likely to have a direct equivalent in voltage glitching.

5.1 Applicability of the Attacks

These attacks are applicable to target devices with similar glitch detection circuits to the ones shown in this paper. As the attacks are clock glitching attacks, access to manipulate the clock signal is required. The experiments showed how the attacks can work against FPGA implementations, but in principle they should also be applicable to non-reconfigurable hardware such as smart cards, microcontrollers, and ASICs. Because of the challenges of generating clock signals with high enough switching speed, the attacks will, in general, be harder to apply to faster (i.e. lower propagation delay) devices. We have reported to Intel the attacks applicable to PDL-1 (similar to Intel's Tunable Replica Circuit) following the "Report Potential Security Vulnerabilities" procedure. Intel processors should have propagation delays that are orders of magnitude lower than FPGAs, and the attacks are not likely to be feasible against these.

5.2 Countermeasures

Since the described attacks, like most clock glitching attacks, require direct access to manipulate the clock of the target device, a countermeasure can be to avoid using external clock signals directly. Many devices typically use an external clock or oscillator as a reference and then generate internal clocks from the reference by using a Phased-Locked Loop (PLL). Fast changes in the reference clock would not typically propagate through the PLL, and a PLL is therefore an effective countermeasure against most clock glitching attacks including these ones. While using PLLs is common, they are not always available in low resource devices and sometimes external clocks are used for synchronisation reasons.

While we have presented attacks against all three tested detectors, no single attack is effective against all of them. In Figure 12c we can see that there are almost no cases of false negatives caused by a single-glitch attack against PDL-3, and in Figure 14a we can see that attack 3 is ineffective against PDL-1. Thus, we note that the attacks presented here will be ineffective against a combination of PDL-1 and PDL-3. However, the goal of the paper is not to design detectors. Therefore we pose as an open question the development of design methodology for provably secure detectors.

6 Conclusion

We have analysed detection circuits meant to be countermeasures against fault injection attacks and presented clock glitching attacks that can go undetected by these circuits. In our experiments, we showed how faults could be injected into a target device while avoiding detection from the glitch detection circuits. The detection circuits can add security to a system since they protect against more basic attacks, and the attacks presented in this paper can be challenging to apply. Still, we question the effectiveness of these detectors against clock glitching attacks.

Acknowledgements.

This work was supported by CyberSecurity Research Flanders with reference number VR20192203.

References

[ADN+10] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson, and Assia Tria. When clocks fail: On critical paths and clock faults. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application*, pages 182–193, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[AK96] Ross Anderson and Markus Kuhn. Tamper resistance – a cautionary note new. In *2nd USENIX Workshop on Electronic Commerce (EC 96)*, Oakland, CA, November 1996. USENIX Association.

[AK97] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *5th Security Protocols Workshop*, volume 1361 of *LNCS*, pages 125–136. Springer, Heidelberg, April 1997.

[BBK+03] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri. Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. *IEEE Transactions on Computers*, 52(4):492–505, 2003.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 37–51. Springer, Heidelberg, May 1997.

[BGV11] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. An in-depth and black-box characterization of the effects of clock glitches on 8-bit mcus. In *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 105–114, 2011.

[BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the AES s-box. In Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou, editors, *Information Security and Privacy Research - 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012, Heraklion, Crete, Greece, June 4-6, 2012. Proceedings*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 287–298. Springer, 2012.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 513–525. Springer, Heidelberg, August 1997.

[DAN+18] Lauren De Meyer, Victor Arribas, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. M&M: Masks and macs against physical attacks. *IACR TCHES*, 2019(1):25–50, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7333.

[EKD+03] D. Ernst, Nam Sung Kim, S. Das, S. Pant, R. Rao, Toan Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: a low-power pipeline based on circuit-level timing speculation. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, pages 7–18, 2003.

[ELH+12] Sho Endo, Yang Li, Naofumi Homma, Kazuo Sakiyama, Kazuo Ohta, and Takafumi Aoki. An efficient countermeasure against fault sensitivity analysis using configurable delay blocks. In *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 95–102, 2012.

[ESH+11] Sho Endo, Takeshi Sugawara, Naofumi Homma, Takafumi Aoki, and Akashi Satoh. An on-chip glitchy-clock generator for testing fault injection attacks. *Journal of Cryptographic Engineering*, 1(4):265–270, December 2011.

[GRG+14] Kamil Gomina, Jean-Baptiste Rigaud, Philippe Gendrier, Philippe Candelier, and Assia Tria. Power supply glitch attacks: Design and evaluation of detection circuits. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 136–141, 2014.

[ISYT13] Hiroaki Igarashi, Youhua Shi, Masao Yanagisawa, and Nozomu Togawa. Concurrent faulty clock detection for crypto circuits against clock glitch based dfa. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1432–1435, 2013.

[KHEB14] Thomas Korak, Michael Hutter, Baris Ege, and Lejla Batina. Clock glitch attacks in the presence of heating. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 104–114, 2014.

[KKG03] Ramesh Karri, Grigori Kuznetsov, and Michael Gössel. Parity-based concurrent error detection of substitution-permutation network block ciphers. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 113–124. Springer, Heidelberg, September 2003.

[KKT04] Mark Karpovsky, Konrad J. Kulikowski, and Alexander Taubin. Differential fault analysis attack resistant architectures for the advanced encryption standard. In Jean-Jacques Quisquater, Pierre Paradinas, Yves Deswarte, and Anas Abou El Kalam, editors, *Smart Card Research and Advanced Applications VI*, pages 177–192, Boston, MA, 2004. Springer US.

[KSYH11] Yuji Kunitake, Toshinori Sato, Hiroto Yasuura, and Takanori Hayashida. Possibilities to miss predicting timing errors in canary flip-flops. In *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1–4, 2011.

[MSY06] Tal G. Malkin, François-Xavier Standaert, and Moti Yung. A comparative cost/security analysis of fault attack countermeasures. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography*, pages 159–172, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[New18] NewAE Technology. *CW305 Artix FPGA Target*, 2018. Product Datasheet.

[NT22] Daniel Nemiroff and Carlos Tokunaga. Fault-Injection Countermeasures Deployed at Scale. Technical report, Intel Corporation, 2022.

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against spn structures, with application to the aes and khazad. In *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003.

[SA03] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 2–12. Springer, Heidelberg, August 2003.

[SB15] B.G. Streetman and S. Banerjee. *Solid State Electronic Devices*. Always Learning. Pearson Education Limited, 2015.

[SBGD11] N. Selmane, Shivam Bhasin, Sylvain Guilley, and Jean-Luc Danger. Security evaluation of application-specific integrated circuits and field programmable gate arrays against setup time violation attacks. *Information Security, IET*, 5(4):181–190, December 2011.

[SK07] Toshinori Sato and Yuji Kunitake. A simple flip-flop circuit for typical-case designs for dfm. In *8th International Symposium on Quality Electronic Design (ISQED'07)*, pages 539–544, 2007.

[SMG16] Tobias Schneider, Amir Moradi, and Tim Güneysu. ParTI – towards combined hardware countermeasures against side-channel and fault-injection attacks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 302–332. Springer, Heidelberg, August 2016.

[YPA08] Asier Goikoetxea Yanci, Stephen Pickles, and Tughrul Arslan. Detecting voltage glitch attacks on secure devices. In *2008 Bio-inspired, Learning and Intelligent Systems for Security*, pages 75–80, 2008.

[ZDCT13] Loïc Zussa, Jean-Max Dutertre, Jessy Clédière, and Assia Tria. Power supply glitch induced faults on fpga: An in-depth analysis of the injection mechanism. In *2013 IEEE 19th International On-Line Testing Symposium (IOLTS)*, pages 110–115, 2013.

[ZDT+14] Loic Zussa, Amine Dehbaoui, Karim Tobich, Jean-Max Dutertre, Philippe Maurine, Ludovic Guillaume-Sage, Jessy Clediere, and Assia Tria. Efficiency of a glitch detector against electromagnetic fault injection. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, 2014.