Sneaky Glitch: A Clock Glitch Generator to attack the AMD-Xilinx PLL

Lukas Leuenberger

lukas.leuenberger@ost.ch

IMES - Institute for Microelectronics, Embedded Systems and Sensors, OST - Eastern Switzerland University of Applied Sciences

Roman Willi

IMES - Institute for Microelectronics, Embedded Systems and Sensors, OST - Eastern Switzerland University of Applied Sciences

Dorian Amiet

IMES - Institute for Microelectronics, Embedded Systems and Sensors, OST - Eastern Switzerland University of Applied Sciences

Paul Zbinden

IMES - Institute for Microelectronics, Embedded Systems and Sensors, OST - Eastern Switzerland University of Applied Sciences

Research Article

Keywords: field programmable gate array (FPGA), clock glitch, active side-channel attack, mixed-mode clock manager (MMCM), AMD-Xilinx

Posted Date: September 26th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-4472610/v1

Additional Declarations: No competing interests reported.

Sneaky Glitch: A Clock Glitch Generator to attack the AMD-Xilinx PLL

Lukas Leuenberger [1*], Roman Willi[1], Dorian Amiet [1], Paul Zbinden [1]

[1]IMES - Institute for Microelectronics, Embedded Systems and Sensors, OST - Eastern Switzerland University of Applied Sciences, Oberseestrasse 10, 8640 Rapperswil, Switzerland.

*Corresponding author(s). E-mail(s): lukas.leuenberger@ost.ch;
Contributing authors: roman.willi@ost.ch; dorian.amiet@ost.ch; paul.zbinden@ost.ch;

Abstract

Cryptographic algorithms are indispensable from today's world and often require high-throughput implementations accelerated by specialized hardware like field programmable gate arrays (FPGAs). However, these implementations are vulnerable to side-channel attacks such as clock glitch attacks. This paper presents a clock glitch generator designed to insert glitches into a clock signal using only FPGA clocking resources to ensure signal purity. By incorporating an FPGA-internal calibration method, the start and width of the glitches can be set with a precision of 0.53°, equal to 14.8 ps, for a 100 MHz clock. The generator's architecture allows glitches to be inserted in the positive and the negative phase of the clock. This increases the versatility of the possible attacks. The clock glitch generator was used to attack both the mixed-mode clock manager (MMCM) and phase-locked loop (PLL) primitives within the 7-Series and Ultrascale+ family. Despite the PLL's ability to filter out clock glitches, an attack scenario was successfully identified that stealthily increased the output frequency of a 7-Series MMCM and PLL by up to 68 %.

Keywords: field programmable gate array (FPGA), clock glitch, active side-channel attack, mixed-mode clock manager (MMCM), AMD-Xilinx

1 Introduction

Cryptography plays a vital role in both the digital and physical world. In the banking sector's day-to-day operations, it provides a foundation of trust by guaranteeing secure financial transactions, protecting data, and facilitating seamless online operations. Cryptographic algorithms are utilized to encrypt communications, secure payments, and authenticate documents. In addition, cryptography plays a central role in providing secure communications and is of great importance in emerging technologies such as blockchains. New

algorithms strengthen cybersecurity, and technologies such as internet of things (IoT) and mobile networks are secured with cryptographic algorithms. Essentially, cryptography is indispensable in our digital world.

A widely neglected problem when implementing cryptographic algorithms in hard- and software is the risk of private data being intercepted by side-channel attacks. At its simplest, this can be done with a power-analysis side-channel attack [1], where an attacker measures the power consumption of a device running the cryptographic algorithm, or an electromagnetic attack [2], where the radiated electromagnetic field is measured. The measured data is then evaluated to draw conclusions about the secure key used. More sophisticated attacks are active side-channel attacks, where faults are introduced in the hardware to reveal internal secure data. One of the first attacks against a cryptographic algorithm using that technique was described in [3]. Common attacks are voltage glitch [4], electromagnetic fault injection [5] as well as clock glitch attacks [6].

In many applications, cryptographic algorithms require high throughput; therefore, they are accelerated using special hardware such as field programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs). When implementing such algorithms in FPGAs, it is common practice to use an internal FPGA clock. This clock is derived from an externally generated clock through a phase-locked loop (PLL). The use of a PLL ensures that attacks on the external clock signal resulting in disruptions or glitches have little impact on the internal clock [7].

This paper presents a new structure of a clock glitch generator. The generator can be used to perform active side-channel attacks against cryptographic hardware. The structure of the generator is designed to exclusively use the internal clock resources of the FPGA, resulting in a pure clock signal. The structure also allows glitches to be inserted into the positive and negative halves of the clock. The generator can be calibrated to generate very specific clock glitches and the calibration is performed entirely internally on the FPGA. Additionally, the proposed glitch generator is also used to investigate how a mixed-mode clock manager (MMCM) and PLL from a 7-series and an Ultrascale+ FPGA from Xilinx react to clock glitches and whether possible attack scenarios exist.

1.1 Contributions

There following contributions are presented in this paper:

- A novel clock glitch generator structure inside an FPGA that uses only clocking resources that allows to create a pure clock.

- A structure that allow to insert glitches in the positive and negative half of the clock without the need to decide at synthesis time.
- A calibration procedure for the glitch generator which can be done entirely inside the FPGA.
- An in-depth investigation on the behavior of MMCMs and PLLs from AMD-Xilinx FPGAs on clock glitches with possible attack scenarios.

1.2 Structure

The paper is structured as follows: Section 2 gives an overview over published glitch generators. Afterwards, the structure of the proposed clock glitch generator together with the calibration procedure is described in Section 3. This is followed by Section 4, which presents an investigation of how various MMCMs and PLLs react when clock glitches are applied to them. Finally, Section 5 concludes the paper.

2 Glitch Generators

Different clock glitch generators have already been published in the last two decades. The first, to our knowledge, clock glitch generator was presented in 2009 by Fukunaga et al. [8]. This generator used an external pulse generator which generated two clocks with different phases relative to each other. A clock glitch is then generated by switching from one clock source to the other by using an FPGA-internal multiplexer (MUX). The timing of the glitch is defined by the phase between the two clocks and has to be set manually by changing the parameters of the pulse generator.

This concept was further refined in [9] and [10]. Both publications used two digital clock managers (DCMs) to generate two phase-shifted clock signals directly inside the FPGA. The use of DCMs allowed to change the period and width of the glitch at runtime. The same approach was later also used in [11–13]. All these structures usually only allow to insert glitches either at the positive or negative half of the clock and this must be determined at synthesis time.

A disadvantage not covered in these papers is the need for calibration. As each clock is routed differently inside the FPGA, the corresponding path delay has a direct influence on the base phase-shift of each clock. To achieve a glitch as desired, this base phase-shift needs to be determined prior to the generation of the glitched clock signal.

Another method is to use two clocks with different frequencies to create a glitched clock signal. A glitch is created by switching from a low-speed clock to the high-speed clock for one clock cycle of the high-speed clock. The width of the glitch is equal to one period of the high-speed clock an thus might require a very fast clock. Further, the

phase of the two clocks needs to be matched. This approach has first been presented by Balasch et al. [11]. Later on, it was also used in [14–16].

A similar approach was also used by Obermaier et al. [17]. However, instead of using a known high-frequency clock, the outputs of two ring oscillators running at different speeds are combined using an XOR gate. This method generates a random and nondeterministic glitch signal.

Ludovic et al. proposed a different approach in 2021 [18]. Instead of creating a glitch, a clock is cut-off during the rising edge. The clock generator is based on two clocks that are phase-shifted with respect to each other. The delay of the cut-off is determined by the phase-shift of the second clock. The idea of this generator is to achieve a behavior similar to an electromagnetic fault injection.

In summary, the glitch generators can be categorized into two categories:

A) Phase-shifted clocks: Multiple phase-shifted clocks are generated and the glitch is defined by the phase difference of the clocks. By using FPGA internal clock managers (like the MMCM from AMD-Xilinx), it is possible to dynamically set the phase of these clocks. A disadvantage is the need for a calibration.

B) Clock-switching: A low-frequency and a high-frequency clock are generated. The start of the glitch is defined by the phase difference between the two clocks, while the width is given by the period of the high-frequency clock. Since FPGA clock managers have a limit on the maximum achievable clock frequency, the minimum width is usually larger compared to glitch generators based on phase-shifted clocks. In addition, the phase has to be calibrated in order to achieve a known start position of the glitch.

3 Architecture

The structure of the glitch generator is explained in Section 3.1 while Section 3.2 introduces the calibration procedure required for the glitch generator. This is followed by Section 3.3 where measurements are provided. The chapter is then concluded in Section 3.4.

3.1 Structure

Figure 1 shows the structure of the proposed glitch generator consisting of three MMCMs and five global clock buffers (BUFGCTRLs). The first MMCM generates the base clock *clkC* and its inverted twin *clkCn*. The two other MMCMs generate the clocks *clkA* and *clkB*. A BUFGCTRL performs a logical OR operation between these signals which results in the signal *clkAB*. The input *enG* is used to generate the select signal for the glitch insertion. A final BUFGCTRL is then used to insert the glitch into the signal *clkC*. This is done by switching to the inverse clock

clkCn for the duration of the glitch. This structure allows to insert glitches in the positive as well as the negative part of the clock at runtime. In addition, two more BUFGCTRLs are deployed which are required for the calibration (see Section 3.2). In summary, the following logical function is represented by the glitch generator:

$$clkG = (clkA \vee clkB \vee \overline{enG}) \oplus \overline{clkC} \qquad (1)$$

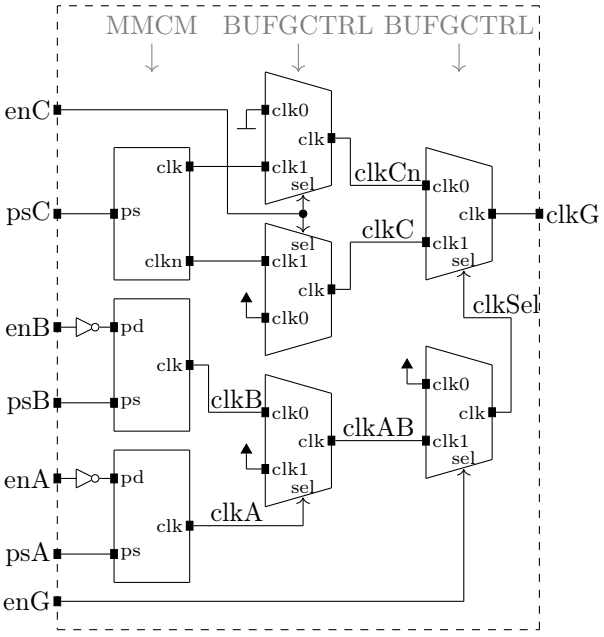


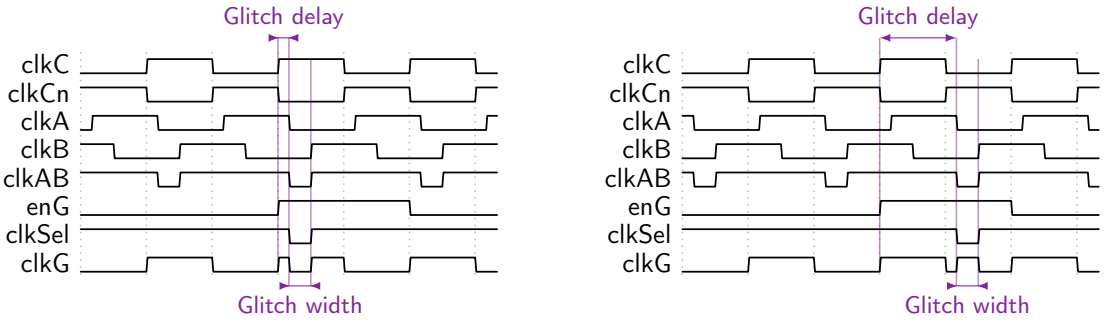
Fig. 1: Architecture of the glitch generator. The glitch generator consists of three MMCMs and five BUFGCTRLs. One MMCM is used to create the base clock *clkC*. The other two MMCMs are used together with two BUFGCTRLs to create the glitch selection signal *clkSel*. A third BUFGCTRL is used to insert the glitch into the clock signal. The remaining two BUFGCTRLs are required for the calibration procedure.

The start of the glitch is determined by the delay between the positive edge of clock *clkC* and the negative edge of clock *clkA*, while the width of the glitch is given by the delay between the negative edge of clock *clkA* and the positive edge of clock *clkB*. Figure 2 visualizes two examples of such a glitch. The glitch gets inserted into the clock *clkC*, when the signal *enG* is active. This allows to precisely control in which and for how many clock cycles a glitch is inserted.

To set the start and the width of the glitch, the fine-phase-shift interface of the MMCMs is used. This interface allows a lightweight way to shift the output clocks in steps of $\frac{1}{56}$ of the VCO period [19]. In Figure 1 the three ports *psA*, *psB* and *psC* stand for this fine-phase-shift interface. It is worth mentioning, that the phase will be reset when the MMCMs are powered down (*pd* port of the MMCMs).

The power-down port of the MMCM generating *clkC* could not be used, as during power-down all outputs are set to zero. For the calibration, it is however required that one of the two clocks generated by this MMCM is a constant one (see Figure 3a, ①). Therefore, a BUFGCTRL is added in this path. A second BUFGCTRL is added to try to keep the phase of the two clocks (*clkC* and *clkCn*) as similar as possible.

The structure of this glitch generator uses only FPGA-internal clock resources to generate the



(a) Glitch during positive phase of clock.

(b) Glitch during negative phase of clock.

Fig. 2: Two examples of a glitch inserted in a clock. The start of the glitch is defined by the negative edge of *clkA* while the duration of the glitch is given by the positive edge of *clkB*. The glitch is inserted when the signal *enG* is active.

clock *clkG*. These optimized resources ensure that clock jitter and distortions are minimized.

3.2 Calibration

The structure of the glitch generator allows to determine the initial phase required for the two MMCMs generating the clocks *clkA* and *clkB*. This can be done by using an FPGA-internal time-to-digital converter (TDC) to measure the clock signal *clkG*. In our implementation, the TDC is based on a delay-chain TDC and an edge detector as used in [20]. The edge detector calculates the position of the first positive edge in the delay chain.

The calibration procedure is provided in Algorithm 1. First, the position *pC* of the positive edge of the clock *clkC* is determined. For this the glitch generator is switched to output only *clkC* as shown in Figure 3b. Multiple measurements are averaged to filter out the jitter from the MMCM. In a second step, the generator is switched to forward *clkA*. The configuration is provided in Figure 3a. The clock is then phase-shifted multiple times until a shift of 360° is reached. For each shift, multiple measurements are taken and averaged. Finally, the base phase-shift *pA* is determined by searching for the shift where the position of the edge of *clkA* is the closest to *clkC*. Note that all possible shifts need to be measured, as the initial phase between *clkC* and *clkA* is not known and can vary with each implementation run. The same procedure is then repeated for *clkB* (configuration shown in Figure 3a) to get the base phase-shift *pB*.

The number of phase-shift steps depends on the divide value of the output clock in the MMCM and can be calculated for the 7-Series FPGAs as follows [19]:

$$\text{numberOfSteps} = 56 \cdot \text{CLKOUT\_DIVIDE} \quad (2)$$

Algorithm 1: Calibration of glitch generator

```
   // Get position of clkC
1  Set configuration of glitch generator to clkC
2  posC ← 0
3  for i ← 0 to numberOfMean − 1 do
4  |  posC ← posC + posC_Meas
5  end
6  pC ← posC / numberOfMean
   // Calibrate clkA
7  Set configuration of glitch generator to clkA
8  for p ← 0 to numberOfSteps − 1 do
9  |  posA(p) ← 0
10 |  for i ← 0 to numberOfMean − 1 do
11 |  |  posA(p) ← posA(p) + posA_Meas
12 |  end
13 |  posA(p) ← posA(p) / numberOfMean
14 end
   // index of minimum value
15 pA ← min_idx(abs(posA(:) − pC))
   // Calibrate clkB
16 Set configuration of glitch generator to clkB
17 for p ← 0 to numberOfSteps − 1 do
18 |  posB(p) ← 0
19 |  for i ← 0 to numberOfMean − 1 do
20 |  |  posB(p) ← posB(p) + posB_Meas
21 |  end
22 |  posB(p) ← posB(p) / numberOfMean
23 end
   // index of minimum value
24 pB ← min_idx(abs(posB(:) − pC))
```

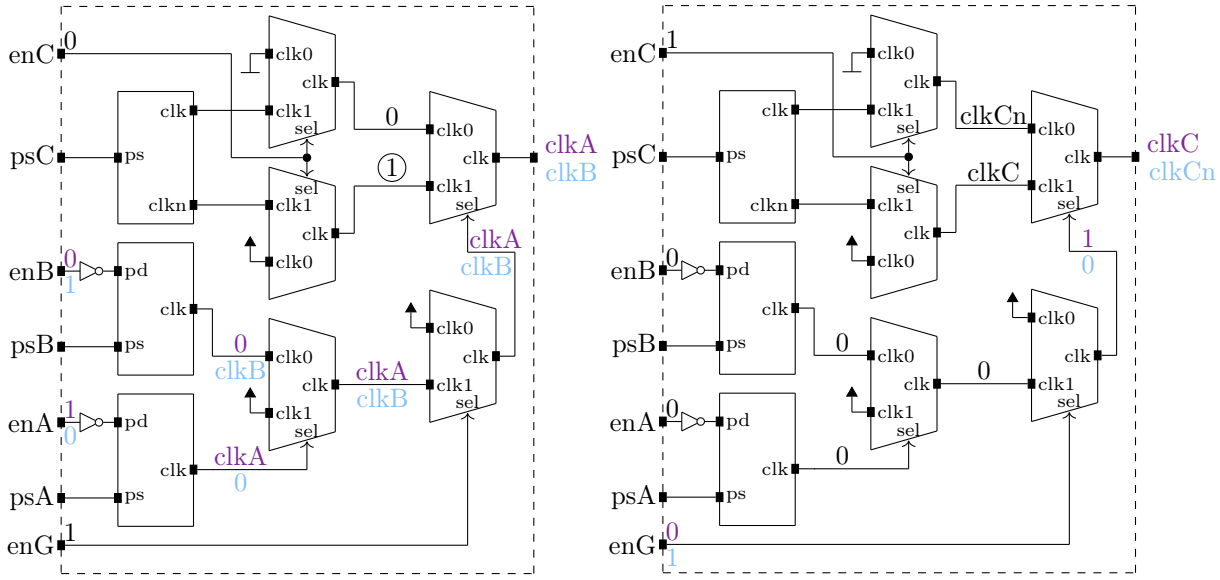
3.3 Measurements

The glitch generator was integrated and tested on the Zedboard, which utilizes the FPGA XC7Z020-CLG484. Vivado 2023.2 from AMD-Xilinx was used for synthesis and implementation. To measure the generated clock, a custom-built printed circuit board (PCB) deploying an FPGA mezzanine card (FMC) connector and multiple high-speed SubMiniature version A (SMA) connectors was used. One of these connectors was connected to a LeCroy HDO9404-MS oscilloscope, which captured all measurements. The oscilloscope was configured to have a sampling rate of 20 GSample/s and a resolution of 9 bit. Figure 4 shows a block diagram of the measurement setup. All measurements have been done at room temperature.

The on-board 100 MHz clock of the Zedboard was used as main input clock. This clock has been filtered with an MMCM to optimize for jitter while maintaining a frequency of 100 MHz. The MMCMs of the glitch generator were set to have a divide value of 12, which leads to the maximum allowed voltage controlled oscilator (VCO) frequency f_{VCO} of 1.2 GHz [21]. This allows to have a minimum possible phase-shift of 0.53° which is equal to 14.8 ps.

The generator has been calibrated with the procedure described in Section 3.2. Determining the base phase-shift was done in MATLAB, but it could be easily integrated into the FPGA as well.

Two different measurements have been conducted. The first measurement verified the correct functionality of the generator while the second was made to investigate the smallest possible phase-shift. For the first measurement, various glitches with different start positions and lengths were generated. Figure 5 shows some of these generated glitches. Two glitches in the positive half of the clock and one glitch in the negative half of the clock are shown as an example.



(a) Measurement of clkA and clkB.

(b) Measurement of clkC and clkCn.

Fig. 3: Configurations to measure the four created clocks.

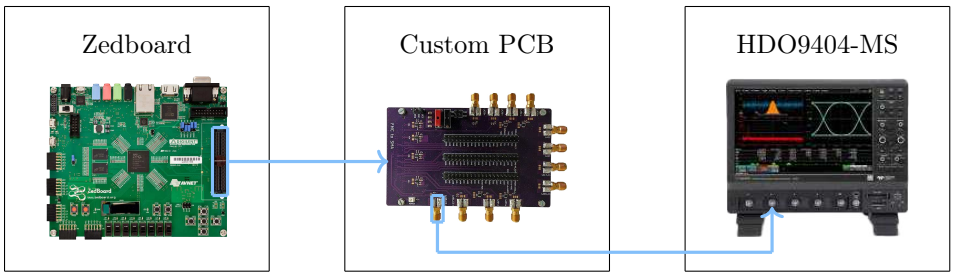


Fig. 4: Block diagram of the measurement setup. The glitch generator running on the Zedboard is connected via an FMC connector to a custom-built PCB and from there via an SMA connector to the oscilloscope.

For the second measurement, several glitches with the same starting position but different lengths were generated. The length was varied in steps of 14.8 ps. For each glitch, 100 measurements were averaged to reduce the influence of clock jitter. Figure 6 shows ten consecutive glitches. The averaged measured step between the width of these glitches is 18 ps.

Our measurements revealed limitations in the ability of the FPGA's output drivers and inputs/outputs (I/Os) to produce short glitches. Throughout our measurements, we determined that a glitch becomes visible at a width of 19° (531 ps), necessitating at least 33° (921 ps) for the glitch to attain 20 % of the supply voltage in the

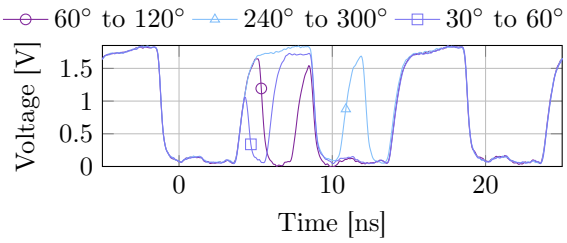


Fig. 5: Three examples of generated glitches. Two glitches are inserted in the positive half of the clock while one glitch is inserted in the negative half of the clock.

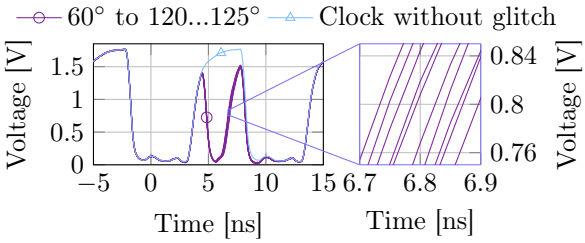


Fig. 6: Ten glitches with the same starting position of 60° and a width of 60° that is incremented with the smallest possible step of 0.53° (14.8 ps). The average measured step is 18 ps. As reference, the distance between the two most left curves in the right plot is 14.9 ps.

positive half, and 71° (1983 ps) in the negative half to reach 80 % of the supply voltage.

3.4 Advantages

To summarize, the proposed structure offers multiple advantages compared to already published clock glitch generators:

- The glitched clock signal is routed only within dedicated clock routing routes, as only clocking resources are used in this path. This helps to preserve the pure clock signal as generated by the MMCM. Almost all other signals are also routed within the dedicated clock routes. Exceptions are the signals *clkA* and *clkSel*. However, routing through the regular routing resources is kept to a minimum. The signals switch from the clocking resources into the regular routing channels at the center interconnect block. This interconnect is placed right next to the BUFGCTRLs.
- An FPGA-internal calibration of the glitch generator is possible. This is an often neglected problem in other publications where the phase-shift introduced by the routing is ignored. The proposed calibration allows to insert glitches with a well defined start and width. The start and the width can be set in increments of 14.8 ps which is limited by the used FPGA technology.

4 Attack on MMCM and PLL

This section investigates the behavior of MMCMs and PLLs in the presence of clock glitches for two different FPGA families from AMD-Xilinx. Two commercial boards, the Zedboard with a Zynq-7000 and the ZCU104 with a Zynq-Ultrascale+, were used for this purpose.

Section 4.1 explains the background of this attack. This is followed by Section 4.2 where the measurement setup is described. Finally, Section 4.3 presents the results of the attack and Section 4.4 discuses these results.

4.1 Purpose

FPGAs use internal PLLs to create the required clocks. The internal clocks are derived from an externally provided clock. However, due to the structure of such a PLL, small disturbances in the input signal are either filtered away [7] or the PLL leaves its locked state while it tries to adjust itself to the changed input clock. In the AMD-Xilinx FPGAs, the locked state is achieved when the phase and frequency are aligned with the input clock [19]. This can be monitored with a signal called «locked» that is present in the primitives.

This makes it difficult, if not impossible, to attack a well-designed hardware design with clock glitches. The purpose of the attack described here is therefore to find clock glitches that increase the output frequency of the PLL while the PLL does not loose its locked state. If such attack scenarios exist, it is possible to create setup and hold time violations within the FPGA (and thus induce errors) without the attack being detected.

4.2 Setup

The device-under-attack (DUA) has a hardware design that is kept to the bare minimum. Only an MMCM or a PLL is instantiated using the clocking wizard intellectual property (IP) block from AMD-Xilinx. The input and output clocks are routed directly to two I/Os from the FPGA. Note that Vivado automatically inserts some clock buffers in these signal paths. Additionally, also the locked signal from the MMCM or PLL is brought to the outside world.

The three signals are connected to the FMC connector on the board. Again the same custom-built FMC board as in Section 3.3 is used to connect the DUA to the glitch generator. The locked signal and the output clock are captured by a LeCroy HDO9404-MS oscilloscope. The oscilloscope was configured to have a sampling rate of 20 GSample/s and a resolution of 9 bit. Figure 7 shows a block diagram of the measurement setup. All measurements have been done at room temperature for a single board.

4.3 Results

Each device has been attacked with all possible glitches that can be generated with the proposed glitch generator. The start of the glitch has been swept from 0° to 360° while the width of the glitch has been swept from 0° to 180°. For each glitch, the maximum frequency of the output clock from the DUA within ten clock cycles after each glitch is determined. Ten clock cycles was chosen because our experiments have shown that the frequency returns to the normal operating frequency after ten cycles at the latest. Figure 8 shows a graphical representation of that measurement, including a reduced map where all measurements where the DUA lost its locked state, have been removed. Note that some glitches have the ability to reduce

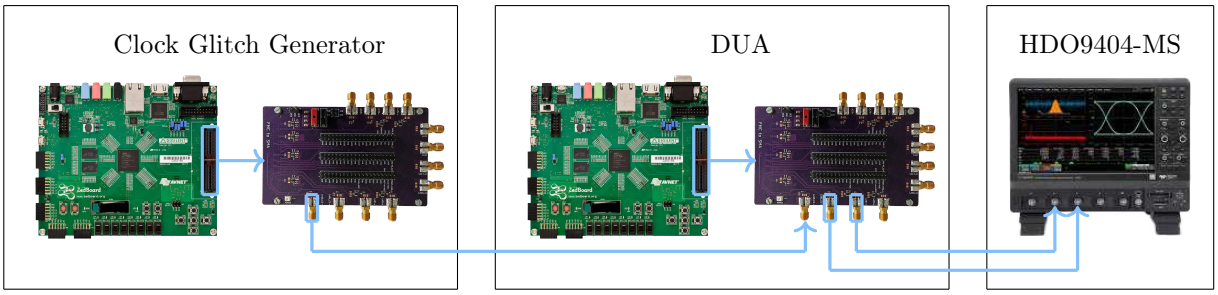


Fig. 7: Block diagram of the measurement setup. The glitch generator running on the Zedboard is connected via an FMC connector to a custom-built PCB and from there via an SMA connector to the DUA. The locked signal and the output clock from the DUA are routed to the oscilloscope via two additional SMA connectors.

the frequency of the DUA. However, since these glitches do not cause timing violations, they were not investigated further.

Figure 8 shows that clock glitches can significantly increase the output frequency of the MMCM in two regions: the lower left and upper right. However, the MMCM from the Zynq-7000 loses its locked state in the lower left region, making it unsuitable for an undetectable attack. In contrast, the upper right region provides a valid attack scenario for both FPGA families, as the locked state is not lost.

The measurements have shown that the MMCM/PLL in both the Zynq-7000 and the Zynq-Ultrascale+ exhibit a similar behavior to the introduced glitches. The frequency can be increased in similar regions, although the frequency change in the Zynq-7000 FPGA is much larger. Table 1 reports the measured maximum frequency and the glitch that lead to that frequency.

Table 1: Overview of the impact of clock glitches on the frequency of the MMCM and the PLL from two different FPGA families. The table lists the maximum achieved frequency and the glitch that lead to that frequency.

FPGA Zynq	Primitive	Frequency [MHz]	Glitch	
			Start [°]	Width [°]
7000	MMCM	168	300	100
7000	PLL	165	240	160
UltraScale+	MMCM	120	250	160
UltraScale+	PLL	131	300	130

The frequency in the MMCM from the 7-Series could be increased by up to 68 %. As the glitches actually change the frequency of the internal VCO, the relative increase can also be applied to other output frequencies. For example, for an output frequency of 200 MHz, the frequency would increase to $1.68 \cdot 200\,\text{MHz} = 336\,\text{MHz}$.

The glitches that have the worst impacts on the DUA have an interesting property. The glitch is inserted in the negative phase of the clock and reaches into the high phase of the next clock period. This results in the insertion of two glitches. The first glitch occurs in the negative-phase where

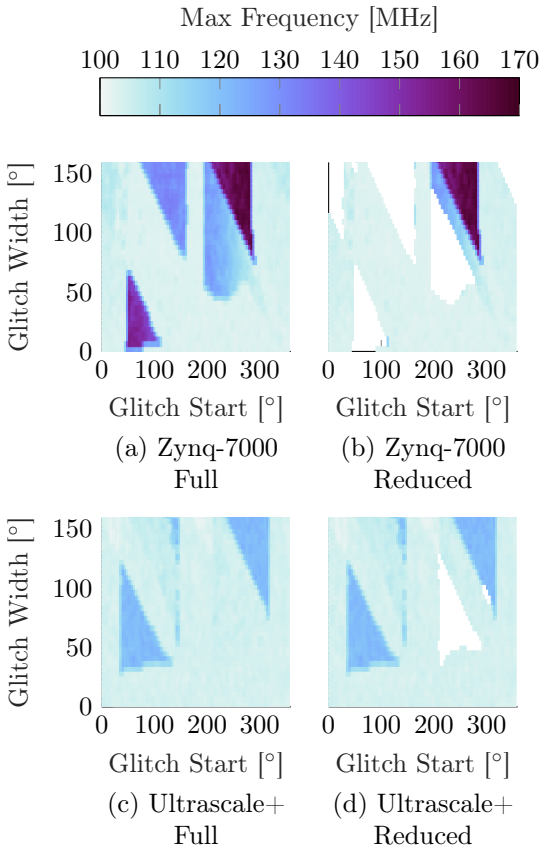


Fig. 8: The impact of clock glitches on the output frequency of 100 MHz of the MMCM. The maximum frequency is determined within ten clock cycles after the glitch occurs. The full map shows all measurements while the reduces map contains only the measurements where the MMCM did not loose its lock. The measurements of the PLL provided similar results and are not shown here.

a positive glitch is inserted, while the second glitch occurs at the beginning of the high-phase where a negative glitch is inserted. Figure 9 shows the glitch that had the worst impact on the MMCM on the Zynq-7000 FPGA.

The maximum output frequency is reached within four clock cycles after a glitch is introduced into the MMCM. Within three further clock cycles

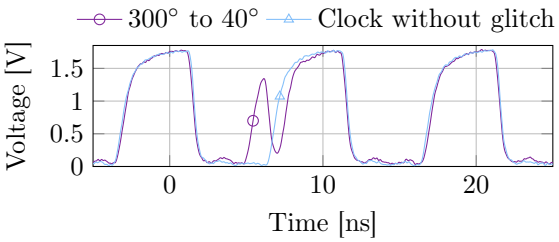


Fig. 9: Glitch which has the worst impact on the MMCM from the Zynq-7000 FPGA. The glitch is inserted in the low phase of the clock and reaches into the high phase of the next period, resulting in a double glitch (positive and negative glitch). A clock without glitch is provided as reference.

the frequency is back to the normal operating frequency. Figure 10 visualizes this behavior. As the frequency is only increased for a very short time, this allows to precisely control when the DUA is attacked, and might offer some possibilities to attack a very specific part of a cryptographic algorithm.

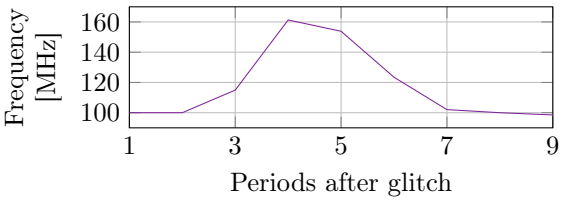


Fig. 10: The maximum frequency is achieved four clock periods after a glitch has been inserted. Within an additional three clock periods, the MMCM has returned back to the normal operation frequency.

4.4 Countermeasure

The attacks on the investigated FPGA primitives revealed a group of glitches that can increase the frequency of the generated output clock by up to 68 % (Zynq-7000), respectively up to 30 % (Zynq-Ultrascale+). Since the group of glitches is relatively large, the glitch itself does not need to be timed exactly. Moreover, as the MMCM/PLL maintains its lock, detecting such an attack becomes challenging.

Xilinx provides a clock monitor interface with its clocking wizard IP block [22]. This block allows to detect glitches in the input clock and frequency errors in the generated clock. However, glitches can only be detected if they are at least one clock cycle long. Glitches generated by the proposed glitch generator are therefore not detected. In addition, very short frequency changes are not detected. A frequency overshoot or undershoot is detected by comparing two counters, one running on a reference clock and one running on the monitored clock. Since the frequency is increased only for about five clock cycles, the values of these two counters do not differ enough.

A better method is to use a TDC to continuously monitor the input clock. The TDC could for example count the number of measured edges or compare the measured signal to a previously stored reference signal. As it takes four clock cycles for the MMCM to reach its maximum frequency after a glitch has occurred, this might be enough time to detect anomalies in the input clock and shut down a cryptographic algorithm before any secret can be leaked.

5 Conclusion

In this work, we presented a novel structure of a clock glitch generator, that uses only clocking resources to create the glitching clock. The glitch generator is accompanied by a calibration procedure that can be done fully inside of an FPGA. The calibration together with the use of clocking resources allows to create a pure clock signal with well defined glitches. The generator was used to find attack scenarios on two FPGA families from AMD-Xilinx. We found that especially the MMCM and PLL on a 7-series FPGA from AMD-Xilinx are vulnerable to certain clock glitches that increase the output frequency up to 68 %. As the primitives do not loose their lock, it is hard to detect such an attack. A possible countermeasure could be the use of a TDC to monitor the input clock.

Declarations

Conflict of interest. The authors declare no competing interests.

Author contribution. L. L. wrote the main text and prepared the figures. All authors reviewed and agreed to the manuscript.

References

[1] Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99, pp. 388–397. Springer, Berlin, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25

[2] De Mulder, E., Buysschaert, P., Ors, S.B., Delmotte, P., Preneel, B., Vandenbosch, G., Verbauwhede, I.: Electromagnetic analysis attack on an fpga implementation of an elliptic curve cryptosystem. In: EUROCON 2005 - The International Conference on "Computer as a Tool", vol. 2, pp. 1879–1882 (2005). https://doi.org/10.1109/EURCON.2005.1630348

[3] Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) Advances in Cryptology — EUROCRYPT '97, pp. 37–51. Springer, Berlin, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_4

[4] Zussa, L., Dutertre, J.-M., Clédière, J., Tria, A.: 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), pp. 110–115 (2013). https://doi.org/10.1109/IOLTS.2013.6604060

[5] Schmidt, J.-M., Hutter, M.: Optical and em fault-attacks on crt-based rsa: Concrete results. In: Austrochip 2007, 15th Austrian Workhop on Microelectronics, 11 October 2007, Graz, Austria, Proceedings, pp. 61–67. Verlag der Technischen Universität Graz, Graz (2007)

[6] Ning, B., Liu, Q.: Modeling and efficiency analysis of clock glitch fault injection attack. In: 2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), pp. 13–18 (2018). https://doi.org/10.1109/AsianHOST.2018.8607175

[7] Selmke, B., Hauschild, F., Obermaier, J.: Peak clock: Fault injection into pll-based systems via clock manipulation. In: Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop. ASHES'19, pp. 85–94. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3338508.3359577

[8] Fukunaga, T., Takahashi, J.: Practical fault attack on a cryptographic lsi with iso/iec 18033-3 block ciphers. In: 2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 84–92 (2009). https://doi.org/10.1109/FDTC.2009.34

[9] Agoyan, M., Dutertre, J.-M., Naccache, D., Robisson, B., Tria, A.: When clocks fail: On critical paths and clock faults. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) Smart Card Research and Advanced Application, pp. 182–193. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12510-2_13

[10] Endo, S., Sugawara, T., Homma, N., Aoki, T., Satoh, A.: An on-chip glitchy-clock generator for testing fault injection attacks. Journal of Cryptographic Engineering **1**(4), 265–270 (2011) https://doi.org/10.1007/s13389-011-0022-y

[11] Balasch, J., Gierlichs, B., Verbauwhede, I.: 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, pp. 105–114 (2011). https://doi.org/10.1109/FDTC.2011.9

[12] Korak, T., Hoefler, M.: On the effects of clock and power supply tampering on two microcontroller platforms. In: 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 8–17 (2014). https://doi.org/10.1109/FDTC.2014.11

[13] Matsubayashi, M., Satoh, A., Ishii, J.: Clock glitch generator on sakura-g for fault injection attack against a cryptographic circuit. In: 2016 IEEE 5th Global Conference on Consumer Electronics, pp. 1–4 (2016). https://doi.org/10.1109/GCCE.2016.7800490

[14] Korczyc, J., Krasniewski, A.: Evaluation of susceptibility of fpga-based circuits to fault injection attacks based on clock glitching. In: 2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), pp. 171–174 (2012). https://doi.org/10.1109/DDECS.2012.6219047

[15] Liu, H., Liu, Z., Qiao, Y., Lu, Z., *et al.*: Clock glitch fault injection attacks on an fpga aes implementation. Journal of Electrotechnology, Electrical Engineering and Management **1**(1), 23–27 (2017) https://doi.org/10.23977/jeeem.2017.11005

[16] Kazemi, Z., Papadimitriou, A., Souvatzoglou, I., Aerabi, E., Ahmed, M.M., Hely, D., Beroulle, V.: On a low cost fault injection framework for security assessment of cyber-physical systems: Clock glitch attacks. In: 2019 IEEE 4th International Verification and Security Workshop (IVSW), pp. 7–12 (2019). https://doi.org/10.1109/IVSW.2019.8854391

[17] Obermaier, J., Specht, R., Sigl, G.: Fuzzy-glitch: A practical ring oscillator based clock glitch attack. In: 2017 International Conference on Applied Electronics (AE), pp. 1–6 (2017). https://doi.org/10.23919/AE.2017.8053601

[18] Claudepierre, L., Péneau, P.-Y., Hardy, D., Rohou, E.: Traitor: A low-cost evaluation platform for multifault injection. In: Proceedings of the 2021 International Symposium on Advanced Security on Software and Systems. ASSS '21, pp. 51–56. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3457340.3458303

[19] Xilinx: Ug472 7 series fpgas clocking resources. (2018). https://docs.amd.com/v/u/en-US/ug472_7Series_Clocking

[20] Leuenberger, L., Amiet, D., Wei, T., Zbinden, P.: An fpga-based 7-enob 600 msample/s adc without any external components. In: The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. FPGA '21, pp. 240–250. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3431920.3439287

[21] Xilinx: Ds187 zynq-7000 soc: Dc and ac switching characteristics. (2021). https://docs.amd.com/v/u/en-US/ds187-XC7Z010-XC7Z020-Data-Sheet

[22] Xilinx: Pg065 clocking wizard logicore ip product guide. (2022). https://docs.amd.com/r/en-US/pg065-clk-wiz