Timing Violation Induced Faults in Multi-Tenant FPGAs

Dina Mahmoud and Mirjana Stojilović
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
{dina.mahmoud, mirjana.stojilovic}@epfl.ch

***Abstract*—FPGAs have made their way into the cloud, allowing users to gain remote access to the state-of-the-art reconfigurable fabric and implement their custom accelerators. Since FPGAs are large enough to accommodate multiple independent designs, the *multi-tenant* user scenario may soon be prevalent in cloud computing environments. However, shared use of an FPGA raises security concerns. Recently discovered hardware Trojans for use in multi-tenant FPGA settings target denial-of-service attacks, power side-channel attacks, and crosstalk side-channel attacks. In this work, we present an attack method for causing timing-constraints violation in the multi-tenant FPGA setting. This type of attack is very dangerous as the consequences of timing faults are temporary errors, which are often impossible to notice. We demonstrate the attack on a set of self-timed true random number generators (STRNGs), frequently used in cryptographic applications. When the attack is launched, the STRNG outputs become biased and fail randomness tests. However, after the attack, STRNGs recover and continue generating random bits.**

***Index Terms*—FPGA, cloud, multi-tenancy, security, random number generator, timing fault, voltage drop**

I. Introduction

FPGAs are used in a variety of computing domains: embedded platforms for automotive, military, or aerospace industry and, since recently, cloud computing. As FPGAs are growing in size, nothing prevents them from accommodating multiple applications from independent cloud users. Yet, this multi-tenant use of FPGAs cannot happen until all possible security risks are discovered and mitigated. In this paper, we show how an already known security vulnerability can be used to create a much more insidious type of attack: a timing violation induced fault attack (or, in short, a *timing-fault attack*). Timing faults are dangerous as they can cause potentially undetectable errors in computation or communication. To demonstrate the effects of timing faults, we attack a set of self-timed true random number generators (STRNGs).

Other researchers have demonstrated that it is possible, from inside the FPGA, to create power supply voltage drops in a way that can cause FPGA to reset [1]. We build our work on this approach, but we control the voltage drop amplitude and duration so that the reset is avoided. By creating a voltage drop, we cause an increase in the delay of the FPGA logic, i.e., longer combinational path delays. As a consequence, a clock edge may violate the setup time of a flip-flop, resulting in an incorrect output data sample or metastability.

To present our work, we structure the paper as follows. After discussing the related research (Section II), we give the details of our models of the adversary and the target (Section III). Then, we describe the experimental setup (Section IV), present the experimental results (Section V) and discuss them (Section VI). We end the paper with the ideas on the future work (Section VII) and with the conclusions (Section VIII).

II. Related Work

Research contributions on the security vulnerabilities of FPGAs in multi-tenant setting are all very recent. Gnad et al. [1] were the first to configure the FPGA logic to create malicious voltage drops and use them to demonstrate a denial-of-service attack. Schellenberg et al. [2] and Zhao et al. [3], followed by demonstrating a multi-tenant power side-channel attack. Ramesh et al. [4] and Giechaskiel et al. [5] demonstrated a side-channel attack leveraging the crosstalk effects between the neighboring long wires of the FPGA. Our work differs from these in that we use the voltage-drop attack with the aim to cause timing constraints violations. Additionally, we aim at producing transient effects, with the device operation returning to normal after the attack.

III. Adversary and Target Model

A. Adversary

Zussa et al. demonstrated that both power supply glitches and clock glitches induce timing constraints violations [6]. In a multi-tenant FPGA setting, creating glitches in the clock signal of another tenant is not feasible because one has no control over that clock. However, creating power glitches is possible, as demonstrated by Gnad et al. [1].

1) Causing Voltage Glitches: In this work, similarly to the work by Gnad et al. [1], we employ ring oscillators (ROs) to internally create voltage variations. ROs consist of an inverter whose output is connected back to its input. To add a control signal, we insert an AND gate at the output of the RO, and connect both the inverter output and the enable signal to its inputs. The ROs are implemented using LUTs. To have better control over the voltage drop, we choose to instantiate two groups of ROs and control them independently. Fig. 1 shows the two groups of ROs and the control circuit, connected to the clock and a trigger.

2) Controlling Voltage Glitch Duration: If the ROs are activated using a high-frequency signal, the power distribution circuitry compensates for the drop and the drop itself does not last sufficiently long to cause reset [1]. Conversely, if the ROs

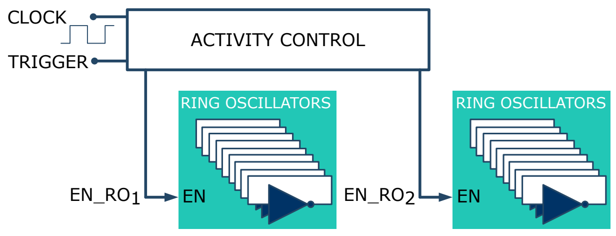

Fig. 1: Adversary design.

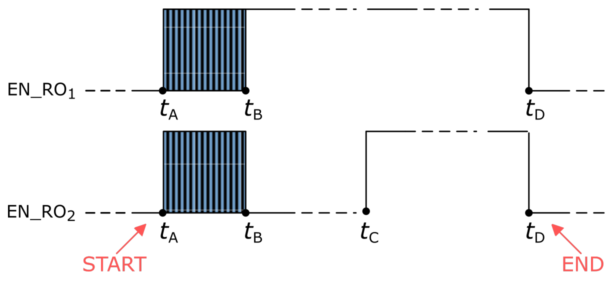

Fig. 2: Waveforms of the RO enable signals. Initially (t_A, t_B), all ROs are controlled using a fast-changing signal. Then, one RO group remains active until the end of activity at t_D, while the other is first disabled, to become active at t_C.

are activated using a low-frequency signal, the voltage drop is higher but the power supply tends to recover from the shock and, if the drop does not last too long, reset does not happen.

Our goal is to keep the board from resetting by carefully controlling the duration of the voltage drop. This is why we propose a particular activation pattern shown in Fig. 2. At first, during the time interval (t_A, t_B), we enable all ROs using a fast-changing signal; this causes a notable but not dangerous voltage drop, sufficient to start disturbing the power supply circuit. Then, we keep the first group of ROs active, causing a stronger and longer-lasting voltage drop. Before the power supply manages to recover, we initiate a second strike by activating the second group of ROs and keeping it active until the end of the attack. This activation scheme is easily tuned for a given board and the FPGA device.

3) Sensing Delay Changes: To evaluate and monitor the effect of the voltage drop on the delay of FPGA logic, we implement an integrated delay-line based sensor [7], [8]. The sensor is composed of a chain of buffers driven by a clock signal. Buffer outputs are connected to a register, clocked at the same frequency but with a 90° phase shift. Hence, the falling clock edge propagates through the delay line during a quarter of the period, when the buffer outputs are registered. Effectively, the sensor records the delay-line propagation depth. A change in voltage affects the sensor output, because it affects the buffer delay too:

$$d \propto \frac{1}{V_{dd} - V_{drop}}. \qquad (1)$$

Here V_{dd} is the steady-state voltage, V_{drop} is the voltage drop, while d is the delay of a buffer [9]. To reduce the width of the sensor output, we add a priority encoder [8], whose output is

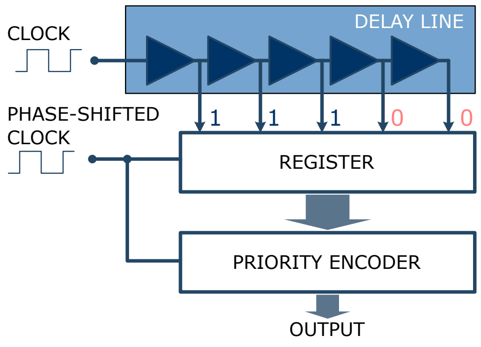

Fig. 3: Voltage-drop sensor design.

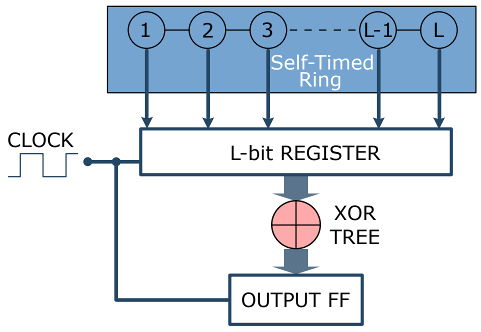

Fig. 4: Self-timed true random number generator design.

the position of the highest '1' in the delay line (Fig. 3). In the rest of the paper, we refer to these sensors as *voltage* sensors.

B. Target

The attack target was chosen with the following criteria in mind: it should be a circuit useful in many applications and whose output can not be easily verified for occasional errors. Thus, an ordinary arithmetic unit would not make a suitable attack target because its result can be easily validated. Finally, we wanted a target whose output would be a critical input to the rest of the victim's design. Following the above criteria, we chose a self-timed true *random number generator* (RNG). True RNGs are used in many cryptographic applications to generate random numbers for security protocols, for ephemeral keys, or for countermeasure implementations [10].

We implemented a self-timed true RNG by Cherkaoui et al. [11], composed of a self-timed ring, intermediate registers, an XOR tree, and an output flip-flop (Fig. 4). The ring consists of Muller gates (or stages), implemented using instances of LUT5. Each of the L stages is fed an input from the previous stage as well as a feedback signal from the following stage. Then, the chain of the stages is closed to form a ring. The jitter, occurring at each stage, results in a jitter in the output signals of the stages. The intermediate registers capture samples of these jittery signals, which, once XOR-ed, produce the random bit output [11]. We implemented a 512-stages ring [10], the XOR tree composed of instances of LUT2, and connected the last XOR stage to the output 1-bit register.

C. Causing Timing Violations

The amplitude of the voltage drop caused by the current drawn by the FPGA depends on several factors: power-delivery network of the FPGA and the board, the number of the circuits toggling, and the waveform of the enable signals. To create a timing fault, the attacker needs to be able to cause an increase in the victim's critical path to the point that it becomes longer than the clock period, and yet prevent the FPGA from resetting. Consequently, the attacker needs to be able to sense and monitor the voltage drop and the delay increase.

We use the voltage sensor output for the delay estimate as follows. For the average delay of a single buffer in the carry chain d, the clock period T_{CLK}, and the critical path delay of the victim T_{CRIT}, the voltage drop required to cause a timing fault corresponds to the following change in sensor readings:

$$\Delta S = S_{\text{OUT, INACTIVE}} - S_{\text{OUT, MIN}} \geq \frac{T_{\text{CLK}} - T_{\text{CRIT}}}{d}, \quad (2)$$

where $S_{\text{OUT, INACTIVE}}$ is the sensor output when ROs are disabled and $S_{\text{OUT, MIN}}$ is the sensor output when the attack is achieving the strongest effect. Since d is very small, the attack is easier to achieve if T_{CLK} and T_{CRIT} are close.

IV. Experimental Setup

For experiments, we used a Virtex-7 FPGA VC707 evaluation board and Vivado Design Suite 2018.2. The clock frequency was set to 150 MHz (we experimentally found that this frequency allows us to use a reasonable length of the sensor for measuring the voltage drop required for a successful attack). The *integrated logic analyzer* (ILA) was used to capture sensor readings and RNG output bitstreams. The ILA and the registers of the voltage sensor were connected to a clock signal delayed 90° relative to the main clock signal. The placement of ROs and RNGs was controlled using placement block (Pblock) constraints of Vivado. Triggering the attack was done manually, for simplicity, via an on-board DIP switch.

A. Design Floorplanning

We conducted three groups of experiments, each with a somewhat different design floorplan (Fig. 5).

1) Two-Sided Attack: In the initial experiment, we placed the ring oscillators on both sides of the RNGs, which were all vertically aligned (Fig. 5a). Although this is an unlikely floorplan for two partitions of an FPGA (as the victim is entirely surrounded by the attacker), we chose this unfavorable setup to test whether it is at all possible to bias the RNG outputs. To observe the voltage drop created by the ROs, we placed one sensor close to the central RNG.

2) Attack from the Left: In this set of experiments, we placed the ROs along the left edge of the device. The RNGs remained vertically aligned. Two voltage sensors were used, one in the upper and the other in the bottom half of the device. The floorplan of the first attack in this series is shown in Fig. 5b. Then, we translated along the x-axis the sensors and the RNGs to increase the distance between them and the ROs. In total, three equidistant positions were tested: RNGs in the left half, in the middle, and in the right half of the FPGA.

TABLE I: Worst-case critical path delay of the RNGs, for system clock frequency of 150 MHz and setup in Fig. 5b.

Critical path delay (ns)				
RNG0	**RNG1**	**RNG2**	**RNG3**	**RNG4**
5.886	5.806	5.927	5.923	5.784

3) Attack from the Top: In the third set of experiments, ROs were placed along the top edge of the FPGA. The RNGs were aligned horizontally. Initially, RNGs were close to the ROs, as shown in Fig. 5c. Two voltage sensors were used, one on the side of the leftmost RNG and one on the side of the rightmost RNG. Then, we translated the sensors and the RNGs along the y-axis, to increase the distance between them and the ROs. In total, five configurations were tested, one for every y-region (Y0, Y1, Y2, Y3, and Y4).

B. Tuning the Voltage Drop Duration

Fig. 2 illustrates the waveform of the RO enable signals. In the time period (t_A, t_B), we chose to have two clock periods of inactivity after every ten clock periods of activity, 15 times, for a total of 180 clock cycles. After that, the first block of ROs would become active for 10^4 clock cycles ($\approx$66.67μs). The second block of ROs would become active at time instant t_C, $2.5 \cdot 10^3$ clock cycles ($\approx$16.67μs) after t_B. At time instant t_D, 10^4 clock cycles after t_B, all ROs would be disabled. One can choose different values for t_B, t_C, and t_D, as long as the attack does not last too long, as otherwise the FPGA would reset. We configured the attack to last $\approx$68μs (less than 150μs, or else the FPGA resets itself [1]).

C. Tuning the Amplitude of the Voltage Drop

For VC707 evaluation board, the system clock frequency of 150 MHz, the RNG critical path delay through the XOR tree as in Table I, and the average delay of a CARRY4 element of 72.3 ps (equivalent to $4d$), the voltage drop required to cause a timing fault should create a change in the sensor reading ΔS of at least 40–50 bits. Knowing that, the only missing information is the number of ROs required to achieve the desired ΔS. For that purpose, we ran a set of experiments in which we instantiated only the ROs and the voltage sensors in the floorplan shown in Fig. 5a. We would observe the sensor output, measure the ΔS and check whether it satisfies Eq. 2. As long as it did not, we increased the number of ROs and reran the experiment. Empirically, we found that a good number of ROs is $\approx$140,000 (24.5% of the available LUTs).

D. Data Collection and Validation

For every RNG, we collected twenty output bitstreams of 2^{14} (16,384) bits each: ten times *without* ROs being active and ten times *with* ROs active (100 bits were collected prior to the attack start in this case), and tested both for randomness. For that purpose, we applied the NIST statistical test suite [12]:

- The frequency test is used for measuring the frequency of '1's and '0's in the output; this test fails if the output is too biased towards one value.

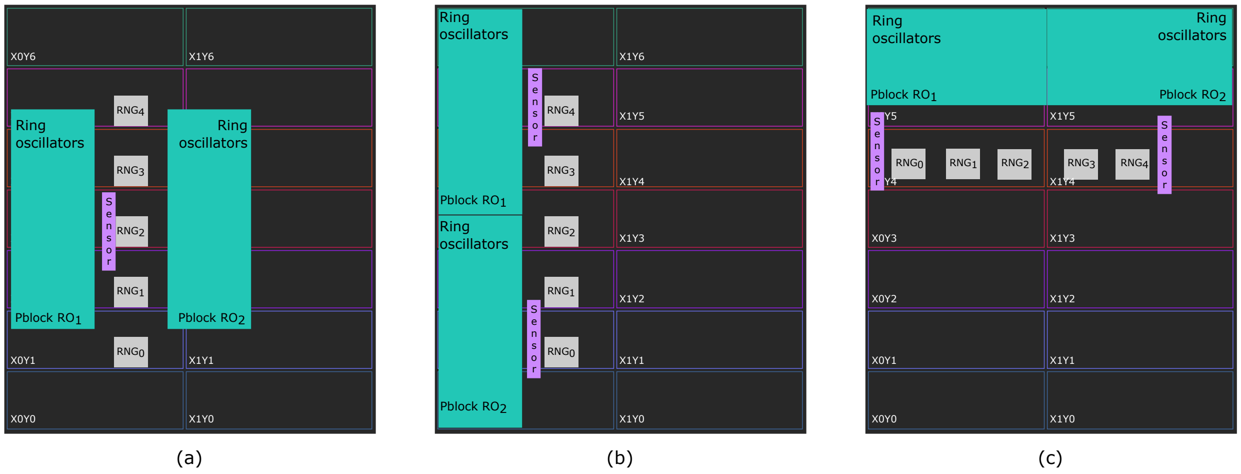

Fig. 5: Design floorplans: (a) the two-sided attack, (b) attack from the left, and (c) the attack from the top.

- The block frequency evaluates the same criteria for a block of bits of specified length; we chose the length of 165, in accordance with the test suite's constraints [12].
- Other tests included the runs test, to check the number of uninterrupted sequences of identical bits, and the template matching, to check the number of occurrences of specified templates in the bitstream [12].

Additionally, we created figures of bitstreams (bitmaps), where '1's are black and '0's are white, to see if there is any observable change in the output bit pattern.

V. Experimental Results

In this section, we present and discuss the results of the experiments.

1) Two-Sided Attack: The floorplan of the two-sided attack is shown in Fig. 5a. We varied the number of ROs and observed both the RNG outputs and the voltage sensor outputs. All RNGs had 512 stages. When the number of ROs reached 140,000 it was possible to detect visually—by looking at the output bitstreams—that the outputs of all five RNGs were biased. In most of the trials, the bias was towards '1', but not always. The outputs of the RNGs in the upper half of the FPGA would become biased even before the activation of the second (lower) group of ROs. The outputs of the RNGs in the lower half of the FPGA would become biased only after the activation of the lower group of the ROs. The output of the sensor and the bitstream of the RNG3 are shown in Fig. 6.

For the same experimental setup, we decreased the number of levels in every RNG to 256 and gradually decreased the number of ROs, while observing the bitstreams. For 120,000 ROs ($\approx 20\%$ of all LUTs), RNG0, RNG1, and RNG4 failed statistical tests for randomness. RNG2 bitstreams had some visible bias, and failed only two statistical tests (block frequency and runs). RNG3 passed the statistical tests and showed no visible bias.

2) Attacks from the Left and from the Top: Table II summarizes the effects that the position of the ROs, the position of the RNGs, and the activity of ROs (Section IV-B) had on the

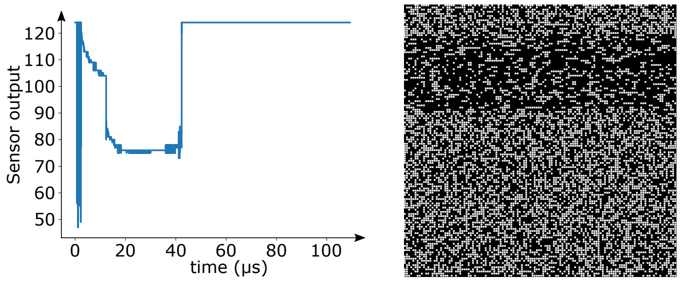

Fig. 6: The sensor output during the attack (left) and the visibly biased output of the RNG3 (right), for the two-sided attack with 140,000 ROs. ROs were activated as in Fig. 2 and $(t_A, t_B) = 1.6\mu s$, $(t_B, t_C) = 10\mu s$, while $(t_B, t_D) = 40\mu s$.

output bitstreams. The results indicate that varying the position of the ROs and the position of the RNGs affects the observable bias and randomness of the output bitstream. Depending on their placement, the RNGs were either affected or not: when affected, they would fail statistical test (sometimes with and sometimes without the observable bias).

3) Varying the Duration of the Attack: To test whether the duration of the attack influences the duration of the observable bias, we repeated the tests with half of the activity period (the time between t_B and t_D reduced by half). Fig. 7 shows the bitstream of the output of RNG2 when attack-from-the-top scenario is applied and the RNGs are aligned horizontally at y-coordinate Y1. The bias starts to appear after the activation of the second block of ROs. Additionally, the bias remains observable until the end of the ROs activity. Hence, the bias duration is affected by the duration of the RO activity.

4) Increasing the Clock Frequency: For the setup in Fig. 5b, we repeated the experiment, this time with the clock frequency increased to 180 MHz. Moreover, we let Vivado decide the implementation and placement—within the pblock—of the LUTs in the XOR trees of the RNGs, to obtain the best critical path delays. This resulted in the average critical path becoming shorter: 4.6 ns. And, due to the unconstrained implementation

TABLE II: Effect of the ROs on the five RNGs for attack-from-the-left and attack-from-the-top scenario ($\star$ = failed statistical tests and observable bias in the generated bitmap, $\times$ = failed statistical tests, – = no effect, n/a = failed statistical tests in the absence of the attack).

	Output bitstream properties				
Location	**RNG0**	**RNG1**	**RNG2**	**RNG3**	**RNG4**
X-left	–	$\star$	$\times$	$\star$	–
X-middle	–	$\star$	$\star$	$\times$	n/a
X-right	$\star$	n/a	n/a	$\times$	$\star$
Y4	$\star$	–	–	–	$\star$
Y3	$\star$	$\times$	$\times$	$\star$	–
Y2	–	$\star$	$\star$	$\star$	n/a
Y1	$\star$	$\star$	$\star$	$\star$	–
Y0	$\star$	$\star$	$\star$	$\times$	n/a

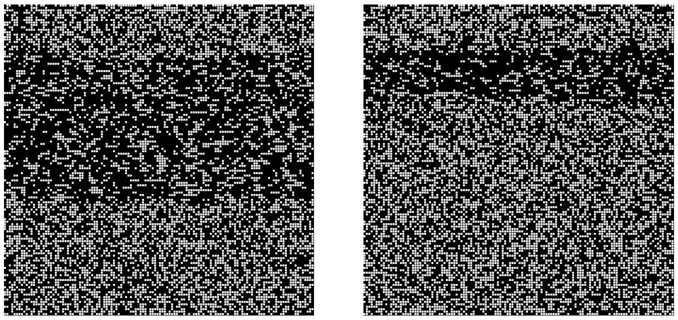
Fig. 7: The bitmaps of the outputs of the RNG2 when the attack-from-the-top scenario is applied and the RNGs are aligned horizontally at y-coordinate Y1. On the left, the duration of the activity period of the ROs (t_A, t_D) is twice longer than the activity period of the ROs on the right.

and placement of the XORs, the critical path of different RNGs varied more: ± 0.4 ns. These changes were only slightly reflected in the RNG outputs: RNG3 exhibited observable bias, while RNG1 and RNG2 exhibited observable bias a tad later.

VI. Discussion

In our experimental setup, several factors were controllable:

1) Parameters of the attack: number of ROs, their implementation, placement, activation pattern, and period.
2) Parameters of the target: placement of the RNGs, number of stages, and implementation of the XOR tree.

In a realistic multi-tenant FPGA scenario, the attacker has limited freedom and can control only the parameters of the attack. All the target parameters will be out of the attacker's control and knowledge. Yet, our experiments demonstrate that these attacks can be effective even when the target is on the opposite side of the FPGA.

A. Non-uniform Effects

With all the controllable factors we took advantage of in the experiments, there was a difference in the observable bias among RNGs. The variation arose between different RNGs and different relative placements of ROs and RNGs. This could be due to various reasons. First, the placement of the RNGs with respect to the ROs has a notable effect. Gnad et al. [8] have reached to the similar conclusions: they analyzed the spatial impact of a transient voltage drop and suggested that the location of the activity can have different effects on the delay increase on the chip. In the experimental setup in this paper, we measured the delay increase near the RNGs and found it to always be sufficient for the critical path delay through the XOR chain to exceed the clock period.

Another reason for the variability could be the placement of the target with respect to other components on the FPGA that also draw current, e.g., PLL, ILA, the reset system processor. With the activation of the ROs, the effect of both the RO blocks and other working logic could accumulate and affect the random bit outputs.

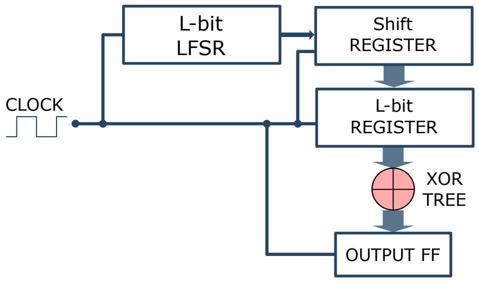

Fig. 8: Alternative target design in which the self-timed ring is replaced by a pseudo-random number generator. The pseudo-random bitstream is generated by a linear-feedback shift register and used to fill in an equally long shift register.

B. Excluding the Effects of the Self-Timed Ring

The observed bias in the output bitstream may not be due exclusively to the timing-fault attack at the XOR-tree, as the asynchronous nature of Muller gates makes the self-timed ring susceptible to delay changes caused by the voltage drop [10]. To exclude the influence of the timing-sensitive ring, we attempt the attack on a modified target illustrated in Figure 8. Instead of the 512-stages ring we instantiate a 512-bit linear-feedback shift register [13] and store the bits it generates in an equally long shift register. Once full, the shift register contains a sequence of pseudo-random bits. The rest of the design is identical to that of the self-timed RNG: an intermediate register, followed by an XOR tree, and an output flip-flop. Then, we repeat the attack scenario in Fig. 5b by replacing all RNGs with pseudo-RNGs. Figure 9 shows the results for pseudo-RNG2 before (left) and after the attack (right). The effects of the timing-fault attack are again clearly visible.

VII. Future Work

There are various future research directions, of which we will mention those that we find most promising.

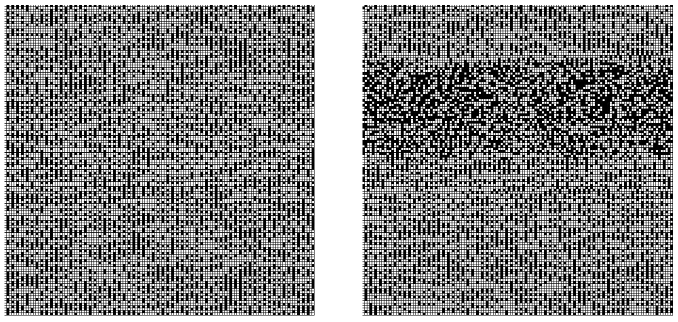
Fig. 9: The bitmaps of the outputs of the pseudo-RNG2 when the attack-from-the-left scenario in Fig. 5b is applied and all self-timed RNGs are replaced with pseudo-RNGs. Left, before the attack. Right, during the attack.

A. Automating the Attack

In this work, we performed a series of experiments in which we varied the relative position of the attacker and the victim, and used the sensor output as guidance for choosing the activation waveform parameters and for setting the required number of ROs. In reality, all this can be automated, and this will be our next step. In the automated attack, instead of using two block of many ROs, one can split them into many smaller groups and provide an activation signal for every group. Then, based on the target delay increase ΔS, the sensor readings could be used to decide on the fly how many of the ROs to activate, when to do it and for how long (to prevent reset).

B. Know Thy Neighbor

More work can be carried out to understand better how the location of the target with respect to the attacker influences the delay increase. This investigation can be followed by an analysis of the combined effects of the RO number, the floorplan, and the activity pattern on the delay increase.

C. Detecting the Attack

In this work, we focused on performing a timing-fault attack from inside FPGA; we did not investigate how one could detect that the attack is taking place or protect from it. Protection from the attack is not obvious [1]. However, one might be able to design a circuit for detecting a potentially dangerous situation. For example, for a given critical path and design clock frequency, one could use the same voltage sensor to estimate whether a timing fault might have happened. If the probability that the timing fault had happened is high, one could choose to discard the data from the previous clock cycles, if at all possible, or restart the computation.

VIII. Conclusion

In this work, we described how malicious FPGA users can create subtle timing faults in the circuits sharing the same FPGA die. As an adversary, we used a large number of ring oscillators (ROs), carefully controlled to avoid resetting the FPGA. As a target, we used self-timed true random number generators. We experimentally demonstrated that, while the attack is taking place, ROs create a voltage drop that induces increased logic delay. The latter, if high enough, causes the critical path of the RNGs to become longer than the clock period, often resulting in a non-random and visibly biased output. Future work will focus on automating the attack, detecting it, and protecting from it.

Acknowledgment

We thank Dennis R.E. Gnad (Karlsruhe Institute of Technology) for sharing his design of the delay-based voltage-fluctuation sensor for Xilinx FPGAs and Markus Ding (École Polytechnique Fédérale de Lausanne) for performing the experiments involving pseudo-RNGs.

References

[1] D. R. Gnad, F. Oboril, and M. B. Tahoori, “Voltage drop-based fault attacks on FPGAs using valid bitstreams,” in *Proceedings of the 27th International Conference on Field-Programmable Logic and Applications*, Ghent, Belgium, Sep. 2017, pp. 1–7.

[2] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, “An inside job:remote power analysis attacks on FPGAs,” in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Dresden, Mar. 2018, pp. 1111–1116.

[3] M. Zhao and G. E. Suh, “FPGA-based remote power side-channel attacks,” in *Proceedings of IEEE Symposium on Security and Privacy*, San Francisco, CA, US, May 2018, pp. 805–820.

[4] C. Ramesh, S. B. Patil, S. N. Dhanuskodi, G. Provelengios, S. Pillement, D. Holcomb, and R. Tessier, “FPGA side channel attacks without physical access,” in *Proc. of the 26th IEEE Symp. on Field-Programmable Custom Computing Machines*, Boulder, CO, USA, May 2018, pp. 1–8.

[5] I. Giechaskiel, K. B. Rasmussen, and K. Eguro, “Leaky wires: Information leakage and covert communication between FPGA long wires,” in *Proceedings of 13th ACM ASIA Conference on Information, Computer and Communications Security (ASIACCS)*, Songdo, Incheon, Republic of Korea, Jun. 2018, pp. 15–27.

[6] L. Zussa, J.-M. Dutertre, J. Clédière, and A. Tria, “Power supply glitch induced faults on FPGA: An in-depth analysis of the injection mechanism,” in *On-Line Testing Symposium (IOLTS)*, Chania, Greece, Sep. 2013, pp. 110–115.

[7] K. M. Zick, M. Srivastav, W. Zhang, and M. French, “Sensing nanosecond-scale voltage attacks and natural transients in FPGAs,” in *Proceedings of the 21st ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, Calif., Feb. 2013, pp. 101–04.

[8] D. R. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, “Analysis of transient voltage fluctuations in FPGAs,” in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Xi’an, China, Dec. 2016, pp. 1–8.

[9] K. Arabi, R. Saleh, and X. Meng, “Power supply noise in SoCs: Metrics, management, and measurement,” *IEEE Design and Test of Computers*, vol. 24, no. 3, pp. 236–44, Aug. 2007.

[10] H. Martín, T. Korak, E. S. Millán, and M. Hutter, “Fault attacks on STRNGs: Impact of glitches, temperature, and underpowering on randomness,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 266–277, Feb 2015.

[11] A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert, “A very high speed true random number generator with entropy assessment,” in *Cryptographic Hardware and Embedded Systems - CHES 2013*, G. Bertoni and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, Aug. 2013, pp. 179–196.

[12] L. Bassham, A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, N. Heckert, and J. Dray, “A statistical test suite for random and pseudorandom number generators for cryptographic applications,” Apr. 2010.

[13] H. Wallker, “Table of linear feedback shift registers,” 2017. [Online]. Available: http://courses.cse.tamu.edu/walker/csce680/lfsr_table.pdf