Voltage-based Covert Channels using FPGAs

Dennis R. E. Gnad[*], Cong Dang Khoa Nguyen[†], Syed Hashim Gillani[‡] and Mehdi B. Tahoori[*]

Karlsruhe Institute of Technology (KIT), Germany.
[*]{dennis.gnad,mehdi.tahoori}@kit.edu, [†]khoa-nguyen@gmx.net, [‡]hashimsyed01@gmail.com

Abstract. FPGAs are increasingly used in cloud applications and being integrated into Systems-on-Chip (SoCs). For these systems, various side-channel attacks on cryptographic implementations have been reported, motivating to apply proper countermeasures. Beyond cryptographic implementations, maliciously introduced covert channel receivers and transmitters can allow to exfiltrate other secret information from the FPGA. In this paper, we present a fast covert channel on FPGAs, which exploits the on-chip power distribution network. This can be achieved without any logical connection between the transmitter and receiver blocks. Compared to a recently published covert channel with an estimated 4.8 Mbit/s transmission speed, we show 8 Mbit/s transmission and reduced errors from around 3% to less than 0.003%. Furthermore, we demonstrate proper transmissions of word-size messages and test the channel in the presence of noise generated from other residing tenants' modules in the FPGA. When we place and operate other co-tenant modules that require 85% of the total FPGA area, the error rate increases to 0.02%, depending on the platform and setup. This error rate is still reasonably low for a covert channel. Overall, the transmitter and receiver work with less than 3–5% FPGA LUT resources together. We also show the feasibility of other types of covert channel transmitters, in the form of synchronous circuits within the FPGA.

Keywords: fpga, multi-tenant, accelerator, SoC, covert-channel, side-channel, power distribution network, PDN, on-chip, remote, software, hardware, trojan

1 Introduction

Field Programmable Gate Arrays (FPGAs) are increasingly integrated with processor cores in SoCs, and even offered via remote access in the cloud by various providers such as Amazon, Alibaba, and Microsoft [AWS18, Ali18, PCC+14]. In such systems, users can remotely access the programmable logic. By multi-tenant access, a single FPGA is split among multiple users [EV12, BSB+14, FVS15, KLP+18]. Since typical FPGA and digital design flows involve the use of *Intellectual Property* (IP) cores from third party vendors, an increasing risk is that an adversarial vendor integrates malicious Trojan logic to exfiltrate information secretly [LKG+09, TK10].

For such threats, secure FPGA design flows try to limit the possible information exchange between IP cores [HBW+07, Cor12]. However, recent publications have shown that through the already existing electrical connections in the Power Distribution Network (PDN), a malicious IP core can circumvent these countermeasures and still perform side-channel attacks. These attacks allow extracting secret keys of cryptographic cores in the same FPGA without any logical connections [SGMT18a, KGT18, ZS18, GDT+19], and can even be exploited when the victim is outside of the FPGA [SGMT18b]. Such attacks rely on side-channel information leakage during operation of cryptographic cores, and might require hundreds or thousands of measurements to extract a small secret key.

Until now, the achievable data throughput through the PDN side channel is not systematically analyzed, when a deliberate communication channel is established. Such *Covert Channels* can be used to secretly exfiltrate information in more complex attack scenarios. For instance, if Hardware Trojans [KHD+08, LKG+09, TK10] or malicious Software [Gir87] can be placed into a system, they can use a covert channel to exfiltrate secret information to a less privileged security level [KHD+08], or outside of the system [Gir87]. Furthermore, they can also be used for complex attacks within a chip, like Spectre Attacks [KGG+18].

Covert channels are established by modulating or hiding information in various media, which can be an existing communication channel such as in a computer network [Gir87]. When no logical connection exists, side channel information can be used. These can be physical variations, such as temperature, power consumption, acoustic or electromagnetic emanations [PAK99], or micro-architectural effects inside a chip, such as the difference in timing of cache memory access [Per05, KGG+18]. By causing or observing these variations, covert transmitters and receivers can be implemented [Gir87].

Specifically for on-chip communication in FPGAs, not many covert channels have been reported yet. Most of them still require a logical connection or shared resources [PRP+19, SBE11] to achieve a data rate of 3.4 Mbit/s at maximum, but are prevented by established isolation design flows for security [Cor12]. Others achieve 1 bit/s when circumventing the isolation design flow using thermal fluctuations [INK11], which has also specifically been discussed for Cloud FPGAs [TS19]. The work in [GRS19] has shown basic covert channel transmission using on-chip voltage fluctuations on different FPGA platforms. A proper transmission channel for more than single-bit messages is not shown, but estimated to achieve 4.8 Mbit/s with an estimated error rate of 2.4%. What is also missing, is an evaluation under realistic conditions, when other circuits cause voltage noise due to their activity.

In this paper, we improve voltage covert channels in multi-tenant FPGAs, by being resistant to noisy conditions and establishing the transmission of messages at different word sizes. In the FPGAs used in this paper, we can achieve a much lower error rate of 0.003% (30×10^{-6}). This paper is an extension of our preliminary work in [Ngu18]. We extend the results with another method for receiving covert messages, by evaluating the communication under noisy conditions, and additional evaluations on other FPGA boards. The presented voltage covert channel is based on a shared PDN, which supplies the FPGA fabric on the electrical level, typical for most integrated circuits. By that, two isolated IP cores at opposite ends of the FPGA chip can communicate. To achieve this goal, a transmitter and receiver module are designed, which can modulate and demodulate data words into voltage fluctuations. Using our method, the voltage fluctuations can be used for a high-speed covert channel with up to 8 Mbit/s transmission speed at a lower than 30×10^{-6} bit error ratio (BER). The proposed covert channel communication can still be upheld when additional circuits, from other co-tenants, are active in the system, which has not been evaluated before. The presented design can self-calibrate to the process variation of any FPGA (chip-to-chip and within-chip variations), and thus enable attacks in the cloud or SoCs, without the need of a separate bitstream per device. Furthermore, we also show that the transmitter does not necessarily require special building blocks, such as ring oscillators (ROs), which were needed in [GRS19]. Our evaluations show that it is also possible to use synchronous designs, which is not easily detected.

The contributions of this paper can be summarized to the following:

- Voltage-based covert channel communication between modules inside an FPGA, with proper multi-bit transmissions

- Improvements for higher transmission speed and lower error rate over previous work

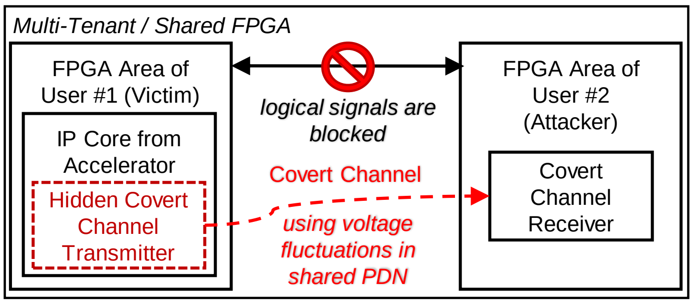

Multi-Tenant / Shared FPGA
FPGA Area of User #1 (Victim)
IP Core from Accelerator
Hidden Covert Channel Transmitter
logical signals are blocked
Covert Channel
using voltage fluctuations in shared PDN
FPGA Area of User #2 (Attacker)
Covert Channel Receiver

Figure 1: Voltage-based Covert Channels as a Threat in Multi-Tenant FPGAs

- Covert channel established and tested under a more realistic scenario with noisy conditions, when other co-tenants are active and present in the FPGA, up to almost maximum utilization
- On-chip covert channel communication method that is also portable to other chips, because the sensor used in the receiver self-calibrates to various FPGAs to counteract the impact of variations (inter and intra-die)
- Basic proof that ROs are not necessarily required for this covert channel, thus being a more stealthy attack

The remaining paper is structured as follows: In Section 2, we will explain background knowledge and explain existing covert channels with focus on FPGAs. The following Section 3 will then explain the design of the covert channel we used for this paper. Section 4 will explain our experimental setup and the final results are discussed in Section 5. Section 6 contains a discussion on channel capabilities and countermeasures. The paper is concluded in Section 7.

2 Related Work and Attacker Model

2.1 Attacker Model

Our basic attacker model is depicted in Figure 1 and is comparable to those in related work on multi-tenant FPGA covert channel attacks [GRS19, TS19, RGS20]. We assume an attacker which has partial or complete access to an IP core or accelerator used in various FPGA tasks. An unsuspecting victim user downloads and uses the IP core from an accelerator store, which often exists for FPGAs in the cloud. The IP core fulfills its normal task, but at the same time it contains logic that uses a covert channel to transmit the information the victim processes, and imprints it on the PDN, which is shared for the full FPGA fabric. We also consider that the FPGA is shared among multiple users as a *multi-tenant* FPGA or in an FPGA-SoC, and thus the attacker can reside in another area of the FPGA or chip as a receiver of the covert channel. The added difficulty in this multi-tenant setting is that other circuits can cause noise in the system. In this scenario, any secret asset from the user that goes through the malicious IP core can be exfiltrated to the attacker.

2.2 Covert Channels

Covert channels are a way to exfiltrate information from a system, and can thus be used to impact security [Gir87, PAK99]. This is done in various ways, such as altering the delay of network packets [Gir87], hiding data by steganography [JJ98] in multimedia streams, through various side channels in modern computer architecture [WL06, GYLH16], or physical phenomena [PAK99, INK11]. In principle, any side effect occurring during operation of a computing system can be used as a covert channel [PAK99]. Power or voltage-based covert channels can exfiltrate information to the outside of a chip depending on CPU utilization or power management [ZBT10, GZBE18, KWKK18, LKG$^+$09]. Security breaches can occur if such a channel is used where secure isolation is demanded.

Side-channel attacks use very similar mechanisms as covert channel communication, and thus sometimes the term is used interchangeably. However, while side channels are usually described to extract data from *accidental* information leakage by measurements, physical covert channels *deliberately* cause side effects and use them as a communication mechanism from the transmitter to the receiver side.

2.3 Voltage-based Security Hazards in Multi-Tenant FPGAs

Every modern integrated circuit (IC) has a Power Distribution Network (PDN), which is often shared among all transistors in the IC, and consists of resistive, capacitive and inductive components [MF04, ASM07]. The power consumption P in an IC is proportional to the toggling rate f of its transistors, and the supply voltage V as $P \propto f \times V^2$. When considering a voltage regulator, it tries to keep V as stable as possible, since transistor delays depend on it $\tau_d \propto 1/V$. However, due to parasitic resistive and inductive components, the voltage is still influenced, leading to fluctuations in the supply voltage available to an individual transistor, and thus its delay τ_d. This influence depends on the current consumption in time $i(t)$ and resistive and inductive components, such that a voltage drop can be described as $V_{drop} = i(t)R + Ldi(t)/dt$.

The behavior of the on-chip PDN has been analyzed for FPGAs before [ZH12, GOKT18, ZAB$^+$18]. Subsequently, it was exploited to cause faults through undervolting from within FPGAs [GOT17, KGT18, SSN$^+$19, MS19] without requiring dedicated fault injection equipment. Furthermore, power analysis side-channel attacks can catch chip-internal voltage fluctuations to extract secret keys from logic in another area of the FPGA [SGMT18a, ZS18, GDT$^+$19]. In both classes of attacks, FPGA primitives are diverted from their intended use, in which voltage can be measured or affected. By that, isolation features on the logical level can be circumvented.

For fault attacks, ring oscillators (ROs) are implemented using FPGA logic [GOT17, SSN$^+$19] with an additional enable-signal to control their activity. By suddenly activating all of them simultaneously using the enable-signal, they reduce the voltage level, using a high change in current in a small time ($di(t)/dt$) [GOT17]. A sufficiently high voltage drop can then cause computation errors [KGT18, MS19], or crash the entire FPGA [GOT17].

Power analysis side-channel attacks on the other hand passively measure voltage fluctuations by either counting how fast ring oscillators run [ZS18], or use Time-to-Digital Converters (TDCs) to achieve a higher sampling rate [SGMT18a, GDT$^+$19] to attack faster symmetric cryptography algorithms.

Similar to ROs, TDC-based sensors rely on the difference in circuit speed, depending on voltage [ZSZF13, GOKT18]. However, instead of an oscillation, they use a single long path as a cascade of buffers, as we also use in this paper and show in the *Receiver*-part at the bottom of Figure 3. Between the buffers, flip-flops ('FD') are added, and the first buffer is sourced from the clock signal, which typically requires an initial delay also made out of buffers. Like this, the registers together show how far the clock propagates within one clock cycle, and thus also the respective relative voltage level. Based on their power consumption, cryptographic circuits cause voltage fluctuations. When these fluctuations are measured with a TDC-based sensor, *Correlation Power Analysis* can analyze them statistically to extract secret keys [SGMT18a, GDT$^+$19].

In FPGAs, various covert channels have been described already, that are mainly based on the PDN. One of them uses capacitive coupling within nearby wires in the same interconnect [PRP+19] to gather the information of the next wire. However, this coupling is easily prevented by adding unused slices between the IP cores as isolation, which is suggested by secure FPGA design flows [HBW+07, Cor12]. Furthermore, if modules already have logical connections with restricted functionality, additional data can still be modulated on top of them in form of small timing variations, or difference in communication speed [SBE11]. Using such methods, about 8 kbit/s can be reached inside an FPGA, while the same authors suggested that IO pins can be re-used to exfiltrate information out of the FPGA, reaching about 3.4 Mbit/s. Furthermore, a covert channel using voltage has also been described for FPGAs, which can transfer data to the outside of the chip with up to 8 Mbit/s at 20% error rate, or as the authors note, at a realistic 500 kbit/s [ZBT10].

Nevertheless, these covert channels all require physical contact or proximity, or existing logical connections, which may not work in multi-tenant FPGAs and particularity in a cloud or SoC environment. To work logically independent, temperature covert channels have been reported that do not require a logical connection and could thus be a security threat to multi-tenant FPGAs. Using temperature changes, 0's or 1's can be represented by high or low temperatures, reaching a speed of about 1 bit/s [INK11, TS19].

More recently, voltage-based covert channels have shown that they also allow for covert channel communication in FPGAs, reaching up to 5 Mbit/s [Ngu18, GRS19], when no other circuit except debugging or system shell logic is active in the system. At lower speed, this has even been shown feasible to escalate between CPU and FPGA accelerator in server-grade systems [RGS20].

2.4 Adjustable Delay Line Voltage Sensor

To show an attack scenario feasible in multi-tenant FPGAs in a cloud setting with remote access, it is not practical to require a fine-tuned design to a particular FPGA device. However the TDC-based voltage sensor we implement here is sensitive to manufacturing process variation [ZSZF13, GOKT18], and requires some adjustment.

The main challenge with the baseline design, proposed by Zick et al. [ZSZF13], is to have the right length *initial delay*, which can not be sufficiently determined using the report from the timing analysis tool alone. In order to make a single bitstream work on all devices, the delay of that design needs to be adjusted for a specific FPGA as well as the modules used within that FPGA, and thus needs testing on the actual device. Based on our experiments with various samples of the same type of FPGA board, the idle value of the sensor can be extremely different. For instance, while one FPGA can be within the range of a 6-bit sensor (0-63), another FPGA might always show 0 or 63 constantly. In [ZSZF13] it is suggested to use a phase-shifted clock for either the entry to the delay line, or the flip-flops used for sampling. In our experiments, that approach increases the idle sensor noise significantly, for which Krautter et al. [KGT20] found a better solution, which we outline here:

In that solution, it can be selected how early or late the clock enters the initial delay line, and by that adjust its length dynamically, calibrated to a specific FPGA board. In Figure 2 we show such a complete tapped delay line that can be adjusted in its path length. It is matched to CARRY4 and LUT Xilinx FPGA primitives [Xil16]. Similar elements exist for other vendors (e.g. Intel Cascade Chain). In the top of Figure 2 we show the exit bins of the TDC, configured as registers connected to the output signals of a CARRY4 element, which still follows the design from [ZSZF13]. The parts below that contain our proposed adjustable-length delay line. The lower part of Figure 2 shows one *fine* calibration slice, and the center part shows two LUTs of a *coarse* calibration slice. The output of the fine calibration stages is fed into coarse calibration stages, and finally can be read from the flip-flops (FD) in the exit slices.

Fine Calibration: To allow different entry points of the clock signal into the path, the fine calibration stages are based on fast CARRY4 elements. The clock signal *clk*

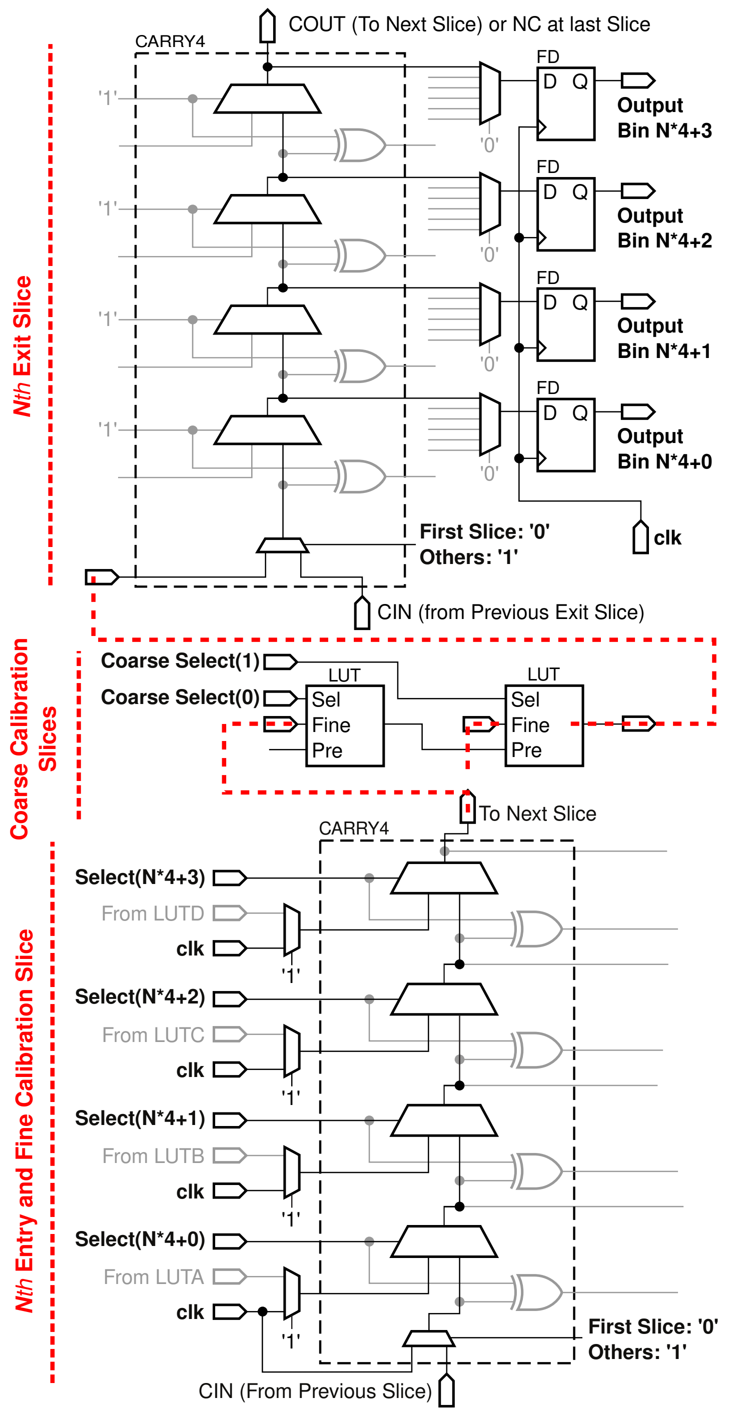


Figure 2: Principle of a delay line based on LUT and CARRY4 elements. At the bottom slice, selectable entry points of the clock *clk* form fine calibration steps. The center slices use the output from fine calibration as the input to LUT-based coarse calibration. In the top exit slices, multiple bins of the delay line are routed to flip-flops (FD) as the output.

Table 1: Amount of coarse and fine calibration slices used in this paper.

FPGA	Fine Calibration Slices (each 1x CARRY4)	Coarse Calibration Slices (each 4x LUT)
Pynq-Z1	4 (=16 stages)	8 (=32 stages)
Kintex-7 KC705	4 (=16 stages)	12 (=48 stages)

can enter the carry-chain at different depths, and therefore provide small delay shifts in the time-scale of single MUX elements within the CARRY4, while still benefiting from a balanced clock tree. The fine calibration is more sensitive to small timing differences and hence it is put before the coarse stages, to be able to use the clock tree with almost the same arrival time of the clock signal. This way, it allows us to keep the steps of the fine calibration more linear. The fine stages output will then be connected to the inputs of the coarse stages through the interconnect wires meant for data signals, which is sufficient for the higher delays of the coarse calibration.

Coarse Calibration: The output from the fine calibration slices is fed to slices with LUTs configured in a similar cascade way as the CARRY4 elements, also allowing a later or earlier entry point of the previous signal. However, instead of the clock, the single output from fine calibration is used as the inputs to all the coarse calibration LUTs. For instance, in the shortest calibrated path, the clock propagates through the last MUX of the fine calibration CARRY4 elements and a single LUT in the coarse calibration stage, until it reaches the CARRY4 elements of the exit slices and the first flip-flop at the output.

The amount and type of required calibration stages can be adjusted to be roughly below the worst-case delay provided by timing analysis, while the remaining calibration can be handled on-the-fly by selecting the right inputs. If more delay is required, latches can also be added in the slices used for coarse calibration. With this design, we do not need the phase-shifted clock tree as in [ZSZF13], and by that have less noise in the sensor output. Furthermore, the sensor can be checked using the same clock (*clk*), synchronous to the remaining design.

With this adjustable length delay line, a simple state machine first goes through coarse and then fine-calibration stages just after reset of the system. In each step, the state machine changes a calibration setting and then observes the sensor value, to finally reach a reasonable idle sensor value in the middle of the possible sensor range. By that, the sensor can be used to observe relative voltage fluctuations in either positive or negative direction.

In Table 1 we show the amount of coarse and fine calibration slices we used for the experiments later in this paper.

3 Voltage-based Covert Channels in FPGAs

The idea of creating a voltage-based covert channel is to use the full-chip PDN in a way similar to a shared bus. Thus, by using similar circuits as used for fault or side-channel attacks, we can *read* or *write* to the PDN, just like to a shared bus network. The previous section explained the various building blocks for side-channel and fault attacks. We adapt the ROs for causing a voltage drop, and TDC sensors to measure the voltage drop. Like that, the ROs can become a transmitter for voltage fluctuations, while TDC or other ROs can be used as receivers. An overview of this principle is shown in Figure 3.

3.1 Signal Modulation and Line Coding

The most basic and straightforward way to modulate a digital signal is to module ‘0’ as low voltage an ‘1’ as high voltage, as in standard CMOS logic. To adhere to the unique properties of the on-chip PDN, we will still alter this scheme from being a *unipolar* code to a *bipolar* code.

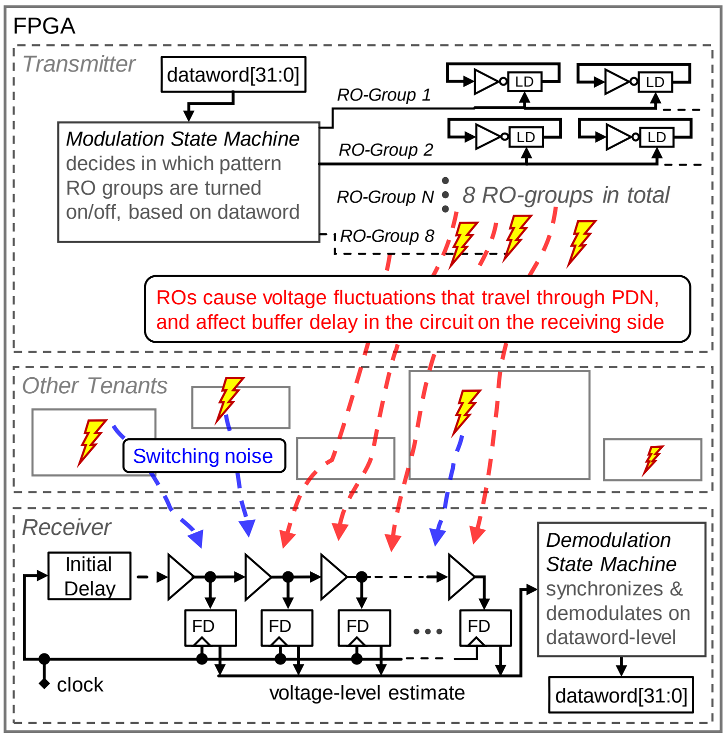

FPGA
Transmitter
dataword[31:0]
Modulation State Machine
decides in which pattern RO groups are turned on/off, based on dataword
RO-Group 1
RO-Group 2
RO-Group N
RO-Group 8
8 RO-groups in total
LD
ROs cause voltage fluctuations that travel through PDN, and affect buffer delay in the circuit on the receiving side
Other Tenants
Switching noise
Receiver
Initial Delay
FD
clock
voltage-level estimate
Demodulation State Machine
synchronizes & demodulates on dataword-level
dataword[31:0]

Figure 3: Principle of transmitter and receiver circuits in the FPGA. The top shows the *Transmitter*, which modulates data onto enable-patterns (in time) by activating RO-Groups on/off to shape voltage fluctuations. The *Receiver* at the bottom is based on a TDC sensor, which is affected by the voltage fluctuations, demodulated by another state machine. The middle resembles other circuits in the FPGA through which the voltage fluctuations travel.

From related work and our own analysis, we know that voltage fluctuations from suddenly activating or deactivating ROs will lead to voltage spikes in either the negative or positive direction, which gradually recover over a certain timespan. An example of that is shown in the waveform diagram in Figure 4. The output of our TDC sensor `tdc_s[5:0]` shows negative and positive voltage spikes, which depend on ROs being active or inactive, as `active_part_reg[7:0]` indicates. We leverage on these properties to move from a unipolar code (0, 1) to a bipolar code (0, —, 1), using the following states:

- Positive Voltage Spike: Encodes '1'
- Negative Voltage Spike: Encodes '0'
- Neutral Voltage Level: 'No Data', between the spikes

With this scheme, we also facilitate synchronization, because 'No Data' can be a synchronization between the data bits.

3.2 Transmitter Design

To implement a transmitter for the described signal modulation, we implement ROs as shown in Figure 3. The 'LD' is a transparent latch with an enable-signal and the inverter

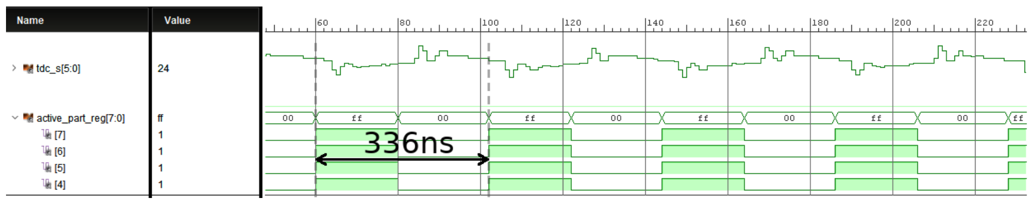


Figure 4: Measurement inside the FPGA on the Pynq-Z1, recorded with Xilinx ILA. It shows the ROs being activated and deactivated in a period of 336 ns, and the respective response of the TDC sensor (tdc_s[5:0]). The timescale is the number of clock cycles of the 125 MHz system clock.

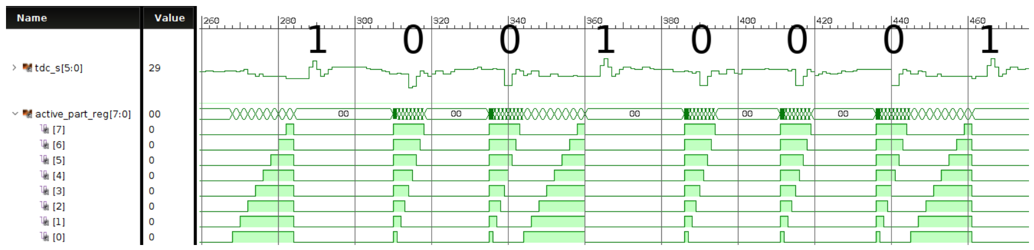


Figure 5: Measurement inside the FPGA on the Pynq-Z1, recorded with Xilinx ILA. It shows how '0' or '1' are encoded as voltage fluctuations. When (de-)activating ROs gradually, tdc_s[5:0] is just slightly affected. When (de-)activating the ROs all at once, tdc_s[5:0] spikes in either direction. The timescale is in clock cycles of 125 MHz.

is realized with a FPGA LUT. This implementation follows one suggestion by Sugawara et al. [SSN+19] for ROs that are not rejected by cloud FPGA design flows. Each RO group thus has a signal to enable or disable their oscillation. All enable signals are then controlled by a state machine that will iterate bit-per-bit of the data word that has to be transmitted, and apply the respective activation pattern. For each bit of the data word, the state machine will enable the ROs in a way to cause a negative or positive voltage spike, or no spike for synchronization in-between.

Using all the ROs at once is not sufficient to transmit data in this way, as it would result in an alternating transmission of '0' and '1', from sudden activation and deactivation of all ROs. The solution is thus to gradually enable and disable ROs, if no voltage spike should be provoked. For that, we separate the ROs into groups. To represent '1'-bits we activate the groups step by step, and then suddenly deactivate them to cause a voltage spike. Vice versa, we suddenly activate all of them, and then deactivate them step by step to represent '0'. We recorded the behavior inside the FPGA with the Xilinx Integrated Logic Analyzer (ILA), and show it in Figure 5 where 8 groups are activated as indicated by the 8 bits of the *active_part_reg[7:0]* signal. The simultaneous decrease in load causes a positive voltage spike, to be interpreted as '1' on the receiving side. Since that method causes a maximum, it should in reverse cause a minimum. Hence activating the ROs all at once and then deactivating them gradually will result in a negative voltage spike as is also shown in Figure 5.

To decide how fine the granularity needs to be for gradually enabling or disabling the ROs, we empirically checked different groupings. For that, we wanted to see how many groups are needed that the voltage fluctuations from enabling or disabling a single group is small enough, i.e. only lead to a change of 2-3 sensor values. We determined that 8 groups result in sufficiently smooth voltage fluctuations, as seen in the waveform on the receiver side, which is tdc_s[5:0] in Figure 5.

3.3 Receiver Logic and Demodulation

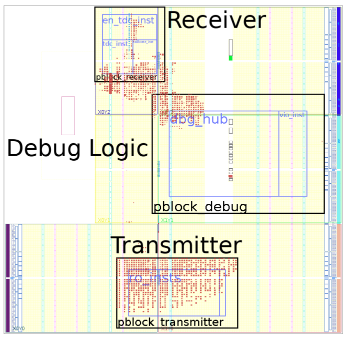
The complete receiver consists of the previously described adjustable delay line that will give a unary one-hot encoded value of N bins that can be encoded in $log_2(N)$-bits and then processed by the additional demodulation logic.

In this work, we use a sensor with 64 bins, resulting in a 6-bit value. The sensor is self-calibrated during power-up for experimental purposes, but this calibration does not need to be precise and can also be performed while the system is at full operation. The only important part is that the sensor still has sufficient margin for positive and negative swings without clipping at its maximum or minimum value. To calibrate, a state machine first adjusts the length of the coarse and then the fine delay line, to finally target the output to be somewhere in the middle of the sensor range, i.e. around 32 for all our experiments performed here with a 6-bit sensor (0–63 sensor range). We propose two demodulation strategies that are evaluated later:

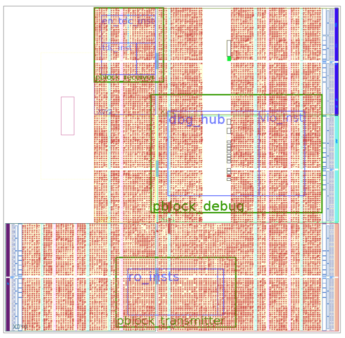
Threshold-based Strategy: In this strategy, the demodulation logic takes the average sensor value tdc_{avg} at idle as the baseline, and from that computes upper and lower bounds that it will use to detect the voltage spikes in either negative or positive direction. To acquire tdc_{avg} we just need to perform a significant long averaging, to filter out all potential AC noise in the signal, since the absolute DC voltage is stable in the system. To detect single bits in the signal, we performed a few tests to experimentally determine reasonable lower and upper thresholds for negative and positive voltage spikes. We found values of -8 and $+4$ at 125 MHz, and -6 and $+3$ at 200 MHz to be reasonable bounds, in the cases where this threshold-based strategy was successful. These thresholds were determined by visually observing the typical strength of the undershoots or overshoots, as observed in the sensor output. The bounds depend on the operating frequency of the TDC, since higher frequencies typically mean less sensitivity, which is in line with related work [SGMT18a]. An adversary can find out such values by experimenting on an own copy of the FPGA of the same model/make. Like that he can test reasonable upper and lower bounds. Even an automated approach is feasible, if the adversary also implements the receiver and performs test-transmissions with various bounds.

Gradient-Estimation-based Strategy: The value of tdc_{avg} needed for the threshold-based strategy can drift over time, since it is also temperature-dependent, as well as dependent on the average power consumption (IR-drop). To prevent the need to acquire tdc_{avg}, we propose a strategy based on estimating the gradient of voltage fluctuations. By that, the signal will only depend on the offset added from the transmitter, and the general offset added by signals below a certain level is ignored. For that, the 6-bit sensor value needs to be recorded for two clock cycles as $tdc_{t=1}$ and $tdc_{t=2}$, and a simple difference between the two values then leads to an estimated gradient $tdc_{grad} = tdc_{t=2} - tdc_{t=1}$. Even in this strategy, a certain threshold needs to be defined for the gradient itself, however it is independent of a baseline idle value, and just needs to calculate the difference between consecutive sensor values. For the gradient-threshold we experimentally found a tdc_{grad} value of $> +7$ or < -7 to be suitable across all the performed experiments. There might be different optimal value depending on the system with more experimentation, but for our proof-of-concept, this was sufficient. However, we later show that the gradient-based approach is not necessarily better, since we only estimate the gradient within one clock cycle. Some voltage drops might require two or more, leading to a reduced gradient and thus wrongly detected bits, for which a more advanced calculation is helpful.

Improved Two-Cycle Gradient-Estimation-based Strategy: In some of the later experiments with synchronous senders, the basic gradient-estimation was still leading to an unacceptably high error rate. Thus, we implemented gradient-estimation that considers the gradient of two consecutive clock cycles, i.e. in total three clock cycles $tdc_{t=1,2,3}$ and differences $tdc_{grad1} = tdc_{t=2} - tdc_{t=1}$ and $tdc_{grad2} = tdc_{t=3} - tdc_{t=2}$. Here, both clock cycles need to have a gradient in the same direction (rising or falling), or it



(a) Floorplan showing the design separation into three modules.



(b) Floorplan with 60× s13207 instances from ISCAS'89 [BBK89] added for voltage noise.

Figure 6: Floorplans of the designs implemented in the Zynq ZC-7020 on the PYNQ-Z1 board.

gets ignored, to reject a certain amount of noise. Otherwise, it is calculated as $tdc_{grad} = tdc_{grad1} + tdc_{grad2}$. For this strategy, the gradient-threshold was reduced to $> +6$ or < -6, which is increasing the sensitivity, while noise is still rejected due to the two-cycle approach.

3.4 Sending and Receiving Word-Size Messages

By detecting single bits, still no communication between modules can be performed, and we thus need to consider a real transmission mode. Since the measured voltage spikes or their gradients can be higher or lower than the threshold for more than a single clock cycle, the decoder in the receiver might wrongly identify more than one bit. Thus, after the decoder identifies a bit, a certain timeframe is added in which it waits before data from the TDC is interpreted again. Within that timeframe, if higher gradients appear, they can still override the initially identified one. An additional problem with the threshold-based approach is that after many transmissions, the decoder could fail to identify some bits, because the maximum or minimum were not within the determined thresholds. In that case, we specify a maximum time that has to be waited, and if the time is up insert a default '0' or '1'.

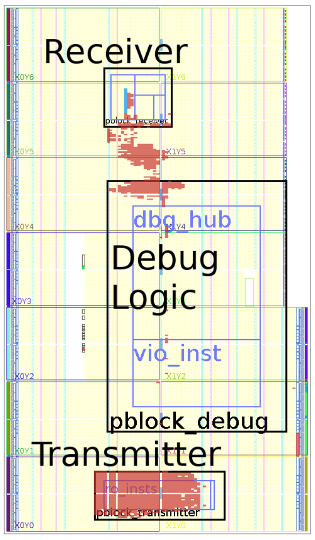
We found good results when our transmitter was implemented such that the time from spike to spike is 200 ns or 125 ns, which in each case is 25 clock cycles, for an operating frequency of 125 MHz or 200 MHz respectively. Based on that, we chose a timeframe with minimum and maximum values of 15 and 31 clock cycles of the respective clock. If no value is received in that time, the default value is assumed. As a dynamic rate control, we adjust the timeframe until the next bit, depending on when the last bit was found.

4 Experimental Setup and Implementation

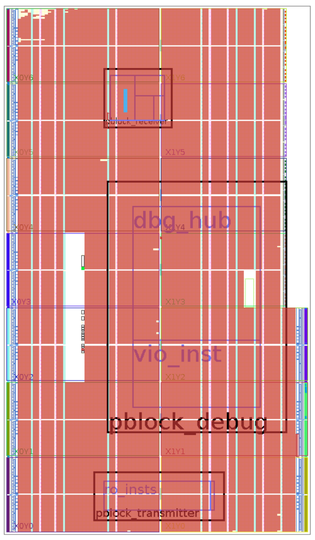
Figure 6 (a) shows the complete floorplan of the implemented design on the Xilinx Pynq-Z1 board, which hosts a Xilinx Zynq XC7Z020CLG400-1 FPGA SoC that integrates a Dual-Core ARM Cortex-A9 CPU with FPGA logic. Figure 6 (b) shows the extended floorplan to check the error tolerance of the covert channel when other circuits are active. In our experiments, the two CPUs of the Zynq SoC were entirely unused. Furthermore, some experiments are performed on a Xilinx KC705 development board, hosting a Kintex-7 XC7K325TFFG900-2 FPGA. The KC705 as well as the Pynq-Z1 board are both powered

Table 2: Logic utilization and estimated power per module on the Pynq-Z1 platform, reported in Xilinx Vivado. Ranges are due to various setups (gradient vs. threshold decoding; 125 vs. 200 MHz). *Small differences between the sum and **Total Max** are possible, due to mapping heuristics.*

Module	**Utilization**			**Estimated Power**
	Slices	**LUTs**	**Registers**	
Receiver (125 MHz)	0.71 %	0.70 %	0.30 %	0.007 W
Receiver (200 MHz)	1.02 %	0.73 %	0.35 %	0.018 W
Transmitter, 2504 ROs	5.45 %	5.0 %	2.5 %	0.013 W
Transmitter, 5000 FFs	10.27 %	9.73 %	4.82 %	0.289 W
Transmitter, 24× s13207	43.59 %	37.98 %	20.47 %	0.597 W
Debug Logic	4.58 %	2.2 %	2.2 %	0.015 W
Clock Generator	—	—	—	0.0 – 0.108 W
Device Static	—	—	—	0.122 – 0.130 W
Circuits used to imitate voltage noise of co-residing tenant:				
s1494+PRNG	1.45 %	1.20 %	0.05 %	0.009 W
s13207+PRNG	2.08 %	1.43 %	0.85 %	0.012 W
60× s13207+PRNG	91.05 %	84.2 – 87.6 %	50.9 %	0.596 – 1.085 W
Total Max (2504 ROs)	99.8 %	92.1 %	55.9 %	1.24 W



(a) Floorplan showing the design separation into three modules.



(b) Floorplan with 250× s13207 instances from ISCAS'89 [BBK89] added as a source of voltage noise.

Figure 7: Floorplans of the designs implemented in the Kintex XC7K325 on the KC705.

Table 3: Logic utilization and estimated power per module on the KC705 platform, reported in Xilinx Vivado. Ranges are due to various setups (gradient vs. threshold decoding; 125 vs. 200 MHz). *Small differences between the sum and **Total Max** are possible, due to mapping heuristics.*

Module	**Utilization**			**Estimated Power**
	Slices	**LUTs**	**Registers**	
Receiver (200 MHz)	0.33 %	0.18 %	0.09 %	0.016 W
Transmitter, 5000 ROs	2.71 %	2.53 %	1.92 %	0.022 W
Transmitter, 10000 ROs	5.23 %	4.99 %	2.48 %	0.025 W
Debug Logic	0.64 %	0.21 %	0.54 %	0.016 W
Clock Generator	—	—	—	0.114 W
Device Static	—	—	—	0.159–0.190 W
Circuit used to imitate voltage noise of co-residing tenant:				
250× s13207+PRNG	96.26 %	89.2 %	55.3 %	4.53 W
Total Max (5000 ROs)	99.64 %	92.5 %	57.2 %	4.89 W

by their dedicated power supplies. We briefly tested USB power for the Pynq-Z1, but did not see any significant difference to the dedicated power supply.

The floorplan of our tested design on the Kintex-7 is shown in Figure 7 (a), again with additional circuits as noise source in Figure 7 (b). For both platforms, we used Xilinx Vivado 2017.1 to develop our design. We marked the respective areas for the transmitter, receiver, and the debugging logic.

Xilinx Vivado estimates for area and power consumption for the PYNQ-Z1 platform are listed in Table 2, and for the KC705 platform in Table 3. Transmitter and receiver contain the previously discussed setup as explained in Section 3 and shown in Figure 3. Please note that the FPGA mapping tools follow respective timing constraints such that the receiver needs more resources to operate error-free at 200 MHz than at 125 MHz. The debug logic consists of standard Xilinx Vivado Debug Cores. We use the Xilinx Integrated Logic Analyzer (ILA) and Virtual Input-Output (VIO) Debug Cores. ILA is used to record and visualize the respective waveforms of our sensor, mainly for debugging. VIO is used to supply and read transmitted and received datawords to perform an evaluation at higher speed. We control both cores using a set of TCL scripts to perform automatic tests for characterizing the Bit Error Ratio (BER) in our results.

To allow automated testing, we start by changing the output datawords of VIO randomly, and set up a *trigger* in ILA that waits for a finished transmission, shown through a ‘ready’ signal. ILA is thus triggered when all bits (8, 16 or 32) of a word have been transmitted. As soon as it is finished, the data demodulated on the receiver side is stored in a VIO register that can be read from our workstation PC. The TCL script compares the sent and received word. If they are not equal, the ILA trace from the FPGA internal BRAM is transferred to the PC for later visualization and debugging. Using this mechanism, tests can be performed in which multiple lengths of datawords are transmitted. Transmissions of 8-bit, 16-bit or 32-bit datawords can be tested, and the option to replace undetected spikes with either a ‘0’-bit or with a ‘1’-bit.

We performed preliminary experiments on the FPGA platforms to decide on the following parameters and use them for all further evaluations and results.

- Distance between transmitter and receiver (floorplan)
- Amount of ROs for sufficient voltage drop, and tradeoff between area usage

- Reasonable minimum time between transmitted bits, based on observed PDN behavior

More extensive evaluation could further tune these parameters to increase the throughput and robustness, making our results a *baseline* for voltage-based FPGA covert channels.

Later, we also evaluate our setup in the presence of running s1494 and s13207 instances of the ISCAS'89 [BBK89] benchmark suite mapped into the FPGA, which according to their publication s1494 is a synthesized controller and s13207 is the design of a real chip.

4.1 Floorplan, Distance

We chose a rather high distance between transmitter and receiver on the chip, which we kept across all experiments by restricting the modules into Xilinx *pblock* regions. This can prove that traditional design practices for isolating IP cores [HBW+07, Cor12] can be bypassed easily by using the presented covert channel. The resulting receiver and transmitter module positions are marked in the floorplan in Figure 6 (a), in which the *Receiver* is placed on the other side of the FPGA, opposite of the *Transmitter*. The same positions are kept for the floorplan in which noise sources were added, shown in Figure 6 (b).

4.2 Amount of ROs as Power Consuming Elements

For the amount of ROs as power-consuming switching elements we use a relatively small number. Since about 13% of the LUTs used for ROs can crash an FPGA [GOT17], the covert communication should be able to work with less LUT utilization. We also verified that no timing errors occured in other circuits by performing experiments on an adder circuitry, similar to the one shown in [KGT18]. We experimentally decided on using 2504 ROs divided into eight groups (8×313) for the PYNQ-Z1 platform, corresponding to 4.7% of its total available LUTs configured as ROs. For the KC705 platform we respectively chose 5000 or 10,000 ROs depending on the experiment, which is equal to about 2.45% and 4.9% of its total available LUTs. In both platforms this is much less in percentage of the total logic cells, when considering other primitives available in the FPGA. To be able to activate just a part of the ROs, we separate them into 8 groups. We expect that when more ROs are used, higher voltage spikes can be generated, which are easier to discern on the receiver side, increasing robustness and potentially speed. On the other hand, we will also test synchronous oscillating elements, which would escape security checks more easily.

4.3 Minimum Time between Bits

We initially decided on a minimum amount of waiting time from bit to bit using experiments on the Pynq-Z1 at 125 MHz, resulting in a 5 Mbit/s transmission rate. Nevertheless, the same amount of clock cycles instead of real time was used for the 200 MHz operating frequency for some experiments on the KC705, which was still successful and leading to a 8 Mbit/s transmission rate. We think that with more ROs, the time for this recovery could also increase, and thus again reduce the possible communication speed.

In order to acquire all these necessary parameters, the intuitive approach is to first check the response on the PDN when all of the ROs are toggled on or off at the same time, and observe the results on the TDC. We show the results of this basic evaluation in a screenshot from the Xilinx Integrated Logic Analyzer (ILA) Debug Core, in Figure 4. In these results, we already use 2504 ROs (8×313) on the Pynq-Z1 and have settled on the floorplan as shown in Figure 6. The activation signal *active_part_reg[7:0]* indicates when all the ROs get activated or deactivated. In this case all 8 groups of ROs are toggled simultaneously. Shortly after activating all the ROs, a negative voltage spike appears, while a positive spike appears after all ROs get deactivated. We can see that the typical time after the effect from the ROs has worn off is about 80 *ns*, i.e. 10 clock cycles. We

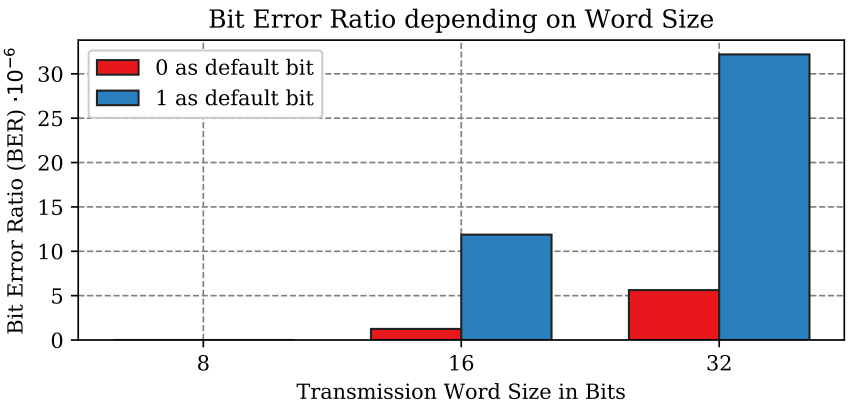


Figure 8: Pynq-Z1 instance #1, 125 MHz and threshold-based decoding. Bit error ratio of 100,000 word transmissions based on 8, 16 or 32-bit word sizes and comparison between '0' or '1' as the default fallback for undetected voltage spikes. It shows that '0' is misdetected more often.

consider that we need about twice this time per bit in order to allow gradually enabling or disabling the ROs from bit to bit. That would lead to a data rate of up to 6 Mbit/s, but if some margin is added, 5 Mbit/s. The results in the following Section 5 show that 8 Mbit/s is also feasible.

5 Results

The line coding we used and present in Figure 5 already proves the basic function of our covert channel. By the duration of the 8-bit that are transmitted in the example of the experimental setup, a fast transmission rate of about 5 Mbit/s is experimentally confirmed. In the detailed results presented in this section, we evaluate this covert channel with respect to potential noise sources, transmission modes, and other boards or platforms. Furthermore, we show an increase in transmission speed up to 8 Mbit/s when using the gradient-based demodulation.

5.1 Basic Results on Pynq-Z1 with Threshold-based Demodulation

As a baseline, we provide the results when 100,000 words are transmitted in a system where only debug logic, receiver, and transmitter are implemented, i.e. as shown in Figure 6 (a). We vary the wordsize between 8-bit, 16-bit and 32-bit, after which a re-synchronization happens, i.e. first timeframe is estimated from spike to spike. In the existing setup, this re-synchronization is still handled through the workstation PC and debugging logic, taking a significant amount of time, which we did not measure. However in a full attack, it can be implemented by waiting the timeframe of 1 or 2 bits. On the receiver side, we evaluate two cases, for either having the default fallback for undetected voltage spikes set to '0' or set to '1'. A summary of these results is shown in Figure 8. The results show that typically no errors happen for 8-bit transmissions, while longer messages with 16- or 32-bit have a small BER at least below 40×10^{-6}. Please note that with only 100,000 words transmitted, we expect there can be a high statistical variance if we would repeat these experiments more often, but these results still show an interesting trend.

To find the reason of different BER depending on the fallback 'default' bit, we look into the errors per each bit of 32-bit dataword transmissions. Figure 9 compares the results per

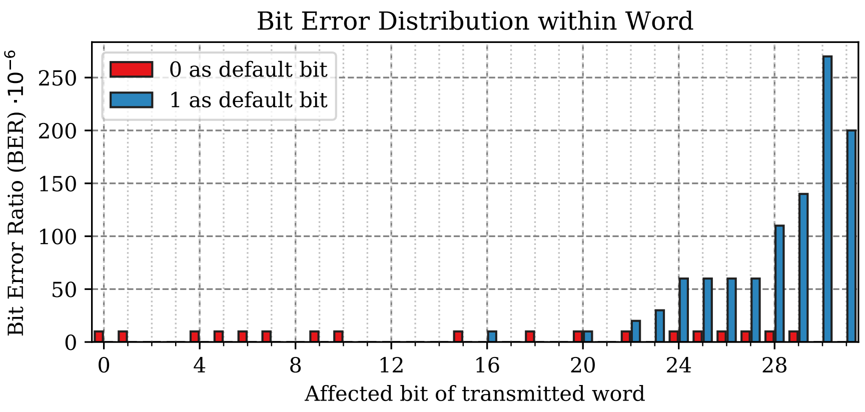


Figure 9: Pynq-Z1 instance #1, 125 MHz and threshold-based decoding. Distribution of bit error ratio across bits of 100,000 32-bit transmissions, comparing '0' and '1' as the default fallback for undetected voltage spikes. For '1' as default it shows that errors typically happen towards the later bits.

bit in the word, from transmissions with '0' or '1' as the default fallback. For '0' as the default, only one incorrect transmitted word occurs out of 100,000. This dataword though, has 18 erroneous bits. As the plot shows, the errors happen nearly everywhere and not in a specific pattern. That means that the basic synchronization did fail to recognize that specific word, and can be considered as an outlier case. On the other hand, when '1' is the default, 101 transmitted words were erroneous with a total of 103 erroneous bits. Hence, overall most transmission errors happen with '0'-bits, i.e. undetected negative voltage spikes. Please note that these error rates can not exactly show the error rate of 1-bits or of 0-bits, but just a tendency if most misdetections are '0' or '1', by making the opposite bit value the default. Figure 9 also shows that most of the errors happen towards the last half of the transmitted word, suggesting that on longer words, the transmitter and receiver can get out of synchronization. Nevertheless, a very low overall error rate can be achieved for a covert communicaton channel, which can be considered sufficient in case error correction codes (ECC) are applied.

5.2 Threshold-based, Error Rate with Noise Sources on Pynq-Z1

The robustness of the covert channel is also tested against more realistic conditions, in which other circuits operate on the FPGA and cause voltage noise to mimic the effect of switching noise coming from other tenants on the FPGA. The transmission was tested in the presence of running s1494 and s13207 instances of the ISCAS'89 [BBK89] benchmark suite mapped into the FPGA. We specifically found s13207 to be well-suited when trying to reach the maximum FPGA utilization, since it has a relatively high register utilization over its logic utilization. The designs were clocked with the same 125 MHz clock and reset properly. The inputs of these circuits were fed with random data, generated from Linear Feedback Shift Registers (LFSRs) inside the system. The circuit outputs are connected to registers with a 'keep' property, to not allow their removal. In the most extreme case, 60 instances of s13207 are instantiated and integrated together with the covert channel circuits, resulting in an FPGA project that uses 92% of existing LUTs and 56% of the flip-flops, leading to 99.8% slice utilization. The floorplan of that design is shown in Figure 6 (b), and the used resources are listed in Table 2.

For these experiments we used another instance of the Pynq-Z1 board, noted as #2 in the respective figure captions. Since this board can be physically different, it proves

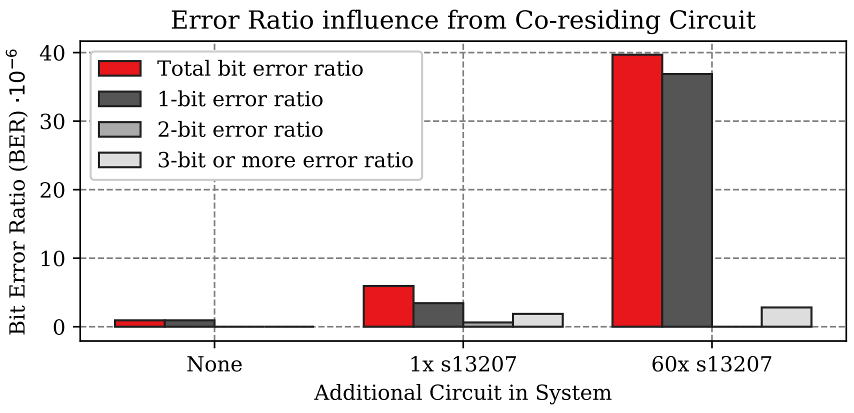


Figure 10: Pynq-Z1 instance #2, 125 MHz and threshold-based decoding. Bit error ratio of 100,000 32-bit word transmissions. Comparison between the first experimental setup and designs with additional modules as a source of voltage noise.

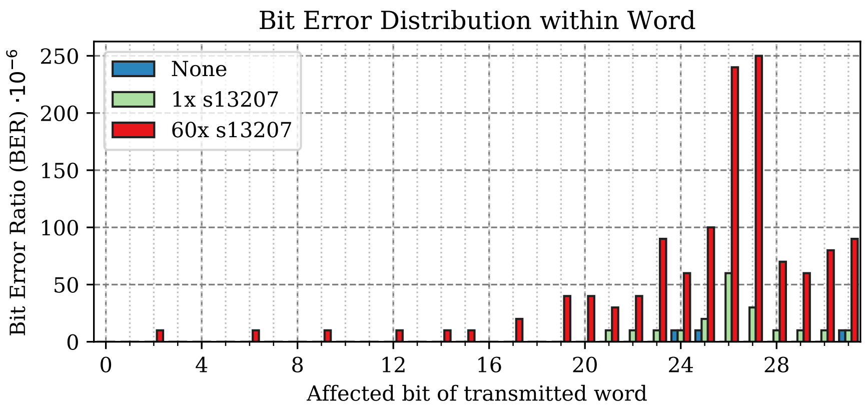


Figure 11: Pynq-Z1 instance #2, 125 MHz and threshold-based decoding. Distribution of bit error ratio across bits of 100,000 32-bit transmissions, comparing the influence from different circuits as noise source. Typically errors happen towards the later bits.

that our approach can adapt to it without changing of parameters. Because there can still be differences between the two boards, we repeated all experiments necessary for the comparisons we show in this section. In these experiments we exclusively used 32-bit transmissions with '0' as the default bit.

For a single s1494 or s13207 there are only moderate resource requirements in the FPGA (cf. Table 2), leading to an increase in error rate by a factor of 6. With the increased noise source of 60× s13207 that leads to a 99.8% slice utilization, as reported in Table 2, the BER was increased by about 40×. As previously observed, the bit errors also occur towards the end of a transmitted word, and mostly 1-bit errors occured, which can still be managed using the widely used Single Error Correction Double Error Detection (SECDED) coding. We show an overview of the errors depending on the circuit noise source in Figure 10, and depending on where in the transmission they occur in Figure 11.

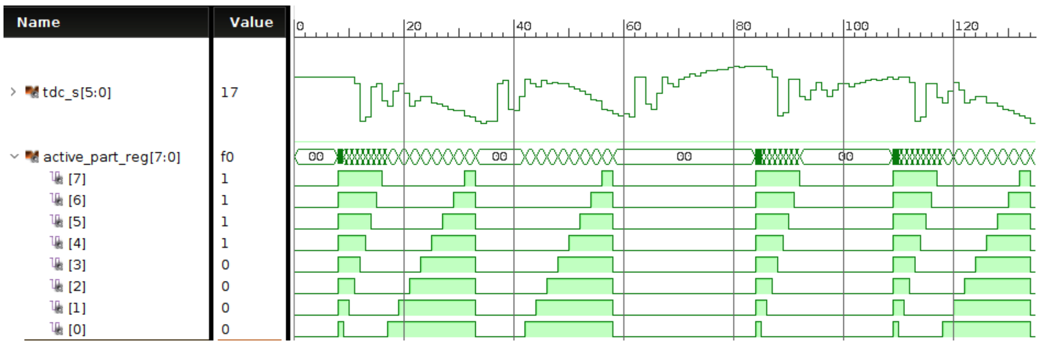


Figure 12: Measurement inside the FPGA on the KC705 board, recorded with Xilinx ILA. It shows that the tdc_s[5:0] signal does almost never reach above the idle value (before time), thus making it infeasible to encode a binary '1' in a simple high-value above a threshold. Previously this was possible on the Pynq-Z1 board as shown in Figure 5.

5.3 Gradient-based Demodulation on Pynq-Z1

Next to threshold-based demodulation, we also test gradient-based demodulation on the Pynq-Z1. We test this demodulation scheme at 125 MHz and 200 MHz, with and without 60x s13207 added as a noise source. In all of these setups, the threshold-based demodulation leads to a lower error rate, as we show in comparisons in Figure 15. We will see later that for the KC705, this approach is superior than threshold-based.

5.4 Threshold-based Demodulation on KC705

Here, we test the KC705 with the same threshold-based demodulation as in the Pynq-Z1. We keep the same percentage of resources used for ROs as in the Pynq-Z1 and also use a 125 MHz operating frequency. In the Kintex-7 of the KC705, a clock generator has to be used to convert the given differential 200 MHz clock to 125 MHz. Since the FPGA on the KC705 has about 4× the resources of the Pynq-Z1, we increase the ROs from 2504 (8×313) to 10,000, resulting in similar resource use for the transmitter of ≈5% LUTs and ≈2.5% Registers (cf. Table 2 with Table 3). Like this, the threshold-based approach performs very poorly on the KC705 board, leading to an error rate of ≈50% when the default bit is set to '0'. When we switch to a default bit of '1' we still get an almost as high error rate of 48%, because almost every bit is detected as '0'.

The reason for this poor performance can be seen when looking at a recorded voltage waveform in Figure 12. After the first activation of ROs, the voltage drops really fast, and does never recover back to the idle value during the transmission. Thus, only negative voltage spikes can be decoded, with a lack of proper synchronization from the idle signal. One alternative would be to switch to a bipolar coding using spikes and idle times, while relying on synchronized timing. However, we did not evaluate this, since a better solution is the gradient-based demodulation.

5.5 Gradient-based Demodulation on KC705, including Noise

Since threshold-based demodulation has some drawbacks on the KC705, we apply the gradient-based demodulation. Because of extremely low BER in preliminary experiments, we can even reduce the amount of ROs to the half of just 5000, requiring only 2.5% of LUTs and 1.9% of Registers. All data for the gradient-based demodulation is recorded for this amount of ROs. We reach zero errors for 100,000 32-bit word transmissions if no additional noise circuits are added to the FPGA.

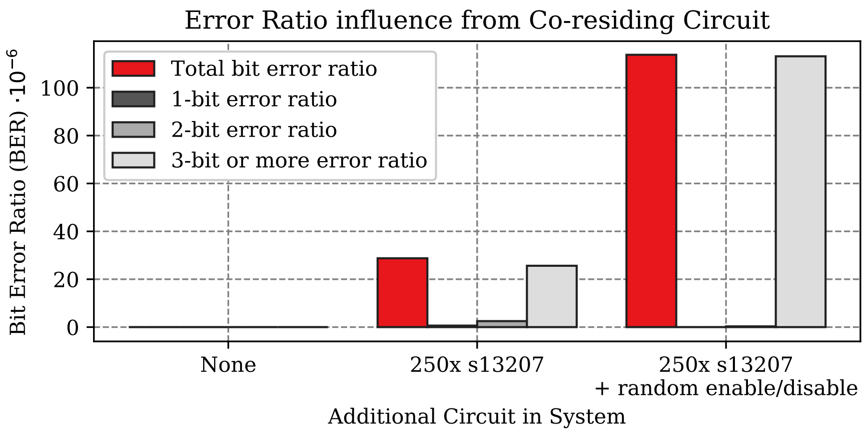


Figure 13: KC705 at 200 MHz (8 Mbit/s) with gradient-based decoding, 2.5% ROs. Bit error ratio of 100,000 32-bit word transmissions. Comparison between the normal experimental setup and designs with additional modules as a source of voltage noise.

To further test the robustness of the gradient-based demodulation, we similarly add circuits as noise sources as we did for the Pynq-Z1 platform before (cf. Section 5.2). To reach an almost maximum utilization in the Kintex-7 XC7K325T, we insert 250× instances of s13207 and respective LFSR-based PRNGs. Additionally to running the 250 instances constantly to generate noise, we added another mode that increases its impact. On top of running all the circuits, we additionally enable and disable them randomly to simulate realistic power changes in the system with higher $di(t)/dt$ load. For that, we chose intervals of 277 clock cycles, such that 2-3 of such load changes can happen within one transmission. In each of these intervals, we look up a random number generator to choose if the enable signal of a circuit is active or inactive.

The floorplan of the KC705 designs with and without noise sources are shown in Figure 7, and the used resources are listed in Table 3. We show the overall error rates in Figure 13, and also the errors depending on the bit inside a word in Figure 14. These results show that gradient-based demodulation leads to very low overall errors on the KC705 platform, even under noisy conditions. However, in case of errors happening, we typically see many bits of a transmitted word being affected, as the category 'more than 3-bit errors' shows. Thus, more effort might be needed for error correction.

5.6 Comparison between Pynq-Z1 and KC705

We performed some additional experiments and show a comparison between the two platforms, where we use board #2 for the Pynq-Z1. We show these results in Figure 15. This overview shows that the Pynq-Z1 board typically performs better with threshold-based demodulation, and at 125 MHz (5 Mbit/s), while 200 MHz (8 Mbit/s) is not necessarily advisable for this platform, except when there is not much noise in the system and with gradient-based demodulation. Interestingly, the gradient-based approach operates better at the higher frequency. The KC705 board does not work well with threshold-based demodulation, since the error rate reaches almost 50%. With gradient-based decoding, it performs very well with a low error rate, even when there is noise in the system.

In summary, both FPGA boards can effectively use this covert channel, but need to use the right demodulation scheme on the receiver side, but without any changes in the transmitter. Thus, the module that leaks the information does not have to be changed

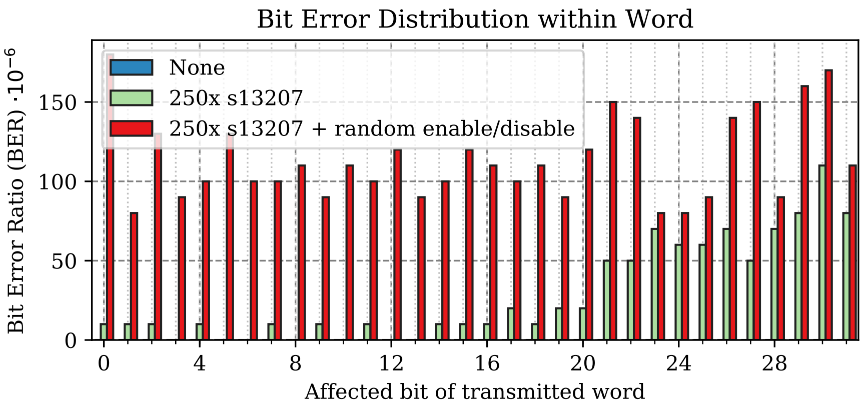


Figure 14: KC705 at 200 MHz (8 Mbit/s) with gradient-based decoding, 2.5% ROs. Distribution of bit error ratio across bits of 100,000 32-bit transmissions, comparing the influence from an almost fully utilized FPGA versus no additional noise source. There are zero errors without noise source, and with noise, never more than 155×10^{-6}, depending on the bit position.

between these two demodulation variants, but only the demodulation logic on the receiving side, i.e. the information leaked into the PDN just needs to be interpreted correctly.

5.7 Transmitting with Synchronous Designs on Pynq-Z1

The shown covert channel so far can be prevented by using bitstream checking for combinational loops, as discussed in related work [SSN+19, KGT19, LMG+20]. Thus, we performed a basic experiment with another design. For that, the LUT+LD based ROs are replaced with synchronous LUT+FD elements. The flip-flop (FD) is clocked at 800 MHz from a clock generated inside the FPGA, by using a Xilinx Clock Generator sourced from the global 125 MHz clock. This was tested on the Pynq-Z1 platform. However, since it is shown that [KGT19, LMG+20] can still detect high-fanout nets, we also evaluate the use of multiple ISCAS'89 s13207 circuits clocked at 600 MHz to modulate the voltage fluctuations. This is similar to what has been previously done to inject faults with AES circuits, in [PHT]. We visualize how these synchronous circuits look in principle in Figure 16.

When using the same resources as used for 2504 (8 × 313) ROs before, these new synchronous elements did produce voltage fluctuations visible in the receiver, but not large enough to reach the previous thresholds. When increasing the amount to 5000 LUT+FD elements and changing the lower threshold from -8 to -7, we can get reasonable results. Similarly, we use 24x s13207 as a sender that can also generate sufficient voltage fluctuations. In both setups we also evaluate gradient-based demodulation, and test the influence of a single s13207 circuit as a source for disturbing voltage noise. These results are reported in Figure 17.

Interestingly, for synchronous senders, the gradient-based demodulation performs better than threshold-based, while this was the other way around for RO-based senders on the Pynq platform (cf. Figure 15). However, in total, the synchronous circuits have a very high error rate, even with just a limited amount of noise from a single s13207 circuit. Nevertheless, we increased the FPGA utilization to 95% by adding as much as possible additional s13207 circuits as sources of noise, which reaches to an almost 50% error rate in gradient and threshold-based demodulation. Thus, here we apply the previously presented improved gradient-based demodulation (cf. Section 3.3).

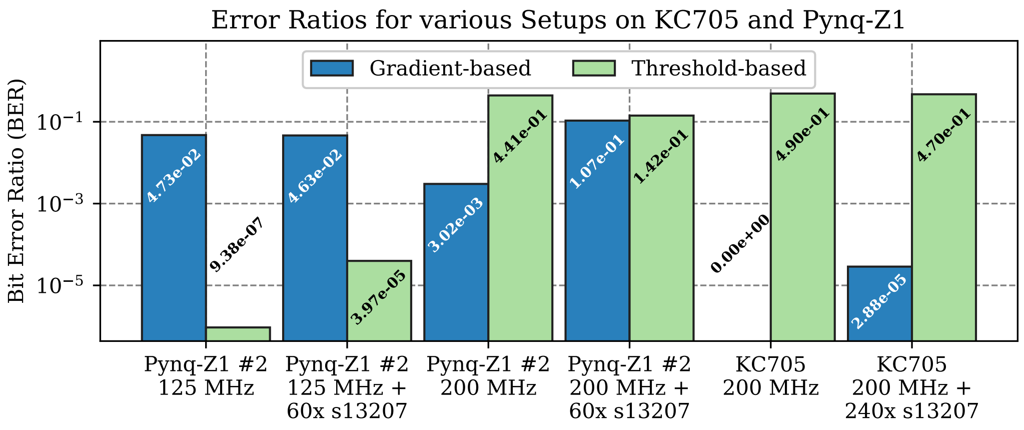


Figure 15: Comparison between various experiments on the KC705 and Pynq-Z1 #2. KC705 with 2.5% LUT utilization for ROs and Pynq-Z1 with 5% LUT utilization for ROs. These results show that gradient-based demodulation works better on the KC705 platform, while threshold-based demodulation works better on the Pynq-Z1.

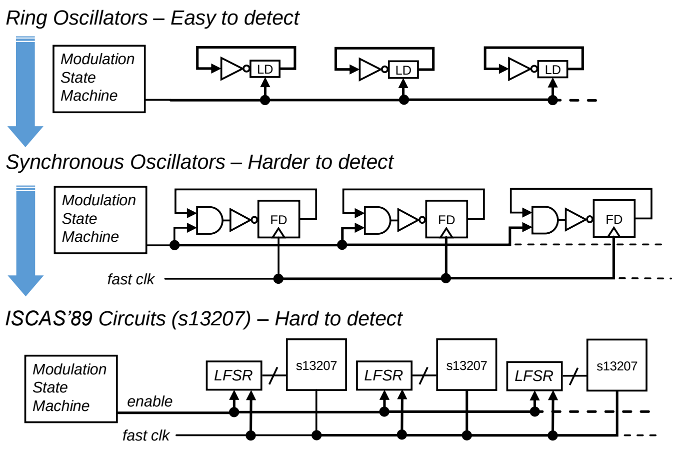

Ring Oscillators – Easy to detect
Modulation State Machine
LD
Synchronous Oscillators – Harder to detect
Modulation State Machine
FD
fast clk
ISCAS'89 Circuits (s13207) – Hard to detect
Modulation State Machine
enable
fast clk
LFSR
s13207

Figure 16: Principle of different types of synchronous senders used to evaluate a more stealthy strategy of modulating voltage fluctuations onto the PDN.

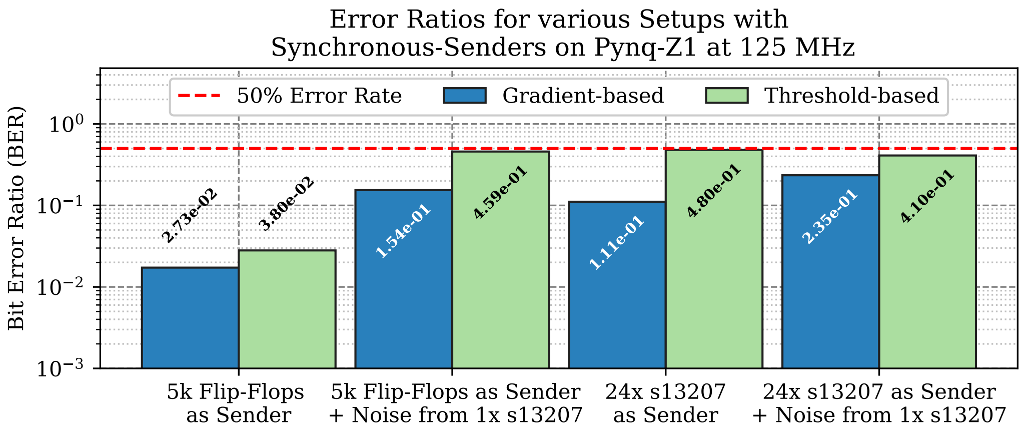


Figure 17: Testing the performance of transmitting with synchronous elements or circuits. Comparison between various experiments on a Pynq-Z1.

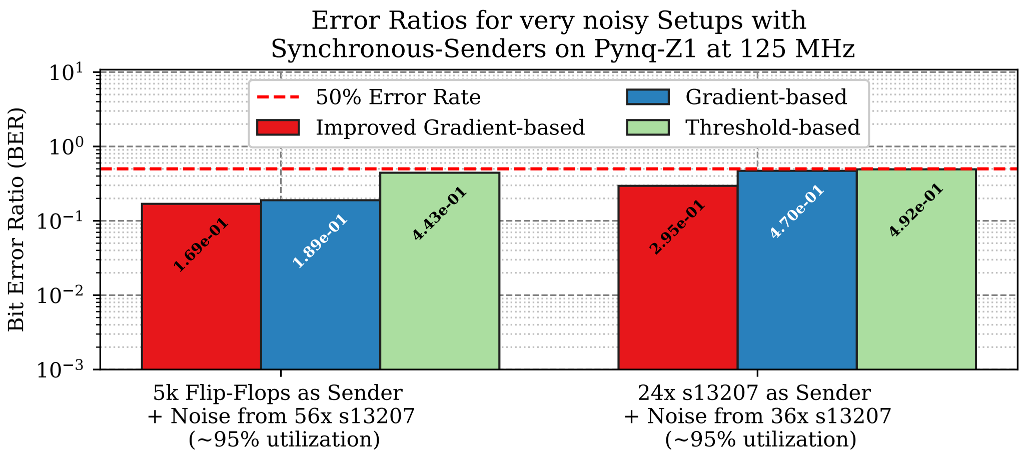


Figure 18: Testing the performance of transmitting with a lot of noise in the system and synchronous elements or circuits. Comparison between various experiments on a Pynq-Z1.

The results are presented in Figure 18. With the improved gradient-based strategy, the error rate can be reduced to about 17% when using Flip-Flops as Senders or about 30% when using 24x s13207 circuits as senders. Thus in this second scenario, the data rate would at least need to be reduced to about 3.5 MBit/s, plus additional error correction overhead.

In total, we can conclude that checking for asynchronous parts of a design [SSN+19, KGT19, LMG+20] is not enough to thwart the shown covert channel attacks, but it increases the required effort, and will need a sufficient amount of error correction for a successful communication.

6 Discussion and Countermeasures

6.1 Capabilities of the Channel

The shown covert channel can provide a very fast communication up to 8 Mbit/s, only by using the shared PDN of a FPGA chip, reaching from one to the other end of the die. That is achieved with a relatively low area usage of less than 3 or 5%, depending on the FPGA, and power consumption of 20 mW. Furthermore, when choosing the right demodulation scheme for the respective FPGA, the communication channel is relatively error-free, even when the full FPGA is utilized by other tenants or modules. This can be explained because normal circuits usually do not have massive changes in toggling (i.e. *switching activity*), and thus barely influence the on-chip voltage level, allowing the covert channel transmission to almost work as before. Even in the case of various circuits being turned on and off during runtime, the error rate stays below 150 $\times 10^{-6}$ BER when the appropriate mode is used for the respective device, i.e. gradient-based for KC705 and threshold-based for Pynq-Z1. For the setups when synchronous flip-flops or circuits are used, the BER is still below 30% even in the noisiest setup, which in the end could still be used for transmissions with a data rate below 3.5 MBit/s.

To improve the transmission further, we think more ROs can generate more clear voltage spikes, slightly increasing the needed area and power consumption. On the other hand, increased voltage spikes also require more time to wear off in the PDN and thus lead to slightly lower data rates. By increasing the ROs, the risk for actual faults will also increase in the system, which has to be taken into consideration. For the amount of ROs we have used in the experiments here, no errors were observed. This confirms that the established channel is for sure “covert”, as it causes no (timing) errors in any running modules in the FPGA.

The absolute limit of this covert channel can still be explored in future work, since we have already seen difference in the KC705 and Pynq-Z1 platforms. Other platforms could again behave differently. However, it is clear that ROs will generate a certain type of voltage spikes due to typical PDN behavior. We have also seen some tendencies of device aging having an impact on these results, which will be explored in future work.

What has also not been extensively evaluated in this paper, is the actual maximum achievable speed. The time between desired voltage spikes of either 125 or 200 ns was found with just a few experiments. With fine-grained optimization, higher data rates will become achievable.

In its current form, this covert channel can become a dangerous building block if maliciously integrated into 3rd party libraries or accelerators, or used as a Hardware Trojan. In that way, a voltage covert channel based on voltage fluctuations can be a risk beyond FPGAs, in other integrated circuits. For instance, it is also feasible that existing synchronous circuit elements are toggled simultaneously to create enough fluctuations for this covert channel.

6.2 Countermeasures

Countermeasures to prevent this electrical covert channel could be either established at the side of the victim, or on system level:

- The system supervisor can check IP cores or entire bitstreams, before they are loaded to the FPGA, to detect the design elements that can potentially be used as transmitter (ROs) and receiver (TDCs). This approach was suggested against side-channel attacks in [KGT19]. However, some stealthier malicious constructs, such as advanced ROs, are much harder to detect with such checking schemes [SSN$^+$19]. We have also explored the use of more complex synchronous ROs for the transmitter, which is not easily prevented.

- The victim side could employ power equalization countermeasures as used in *hiding*-schemes against side-channel attacks [KGS+19]. However, to protect against side-channel analysis, the circuit that should be protected is usually known, while a hidden covert channel transmitter would need to be equalized, which is usually more challenging.

Further investigation and research are required to evaluate these or other possibilities for countermeasures against this covert channel, which is in our future research direction.

7 Conclusion

FPGAs are increasingly adopted as accelerators in Systems-on-Chip (SoCs) or Cloud-based systems, such that multiple user contexts exist in a single FPGA chip. Existing side-channel attacks have shown that new security issues emerge from that. In this work, we show a related security issue in which logically, or even physically, separated untrusted third party IP cores, or user accelerators can perform covert communication inside an entire system. Especially at the high data rate and reliability that we have shown under noisy conditions inside a multi-tenant FPGA, such unwanted communication channels can facilitate various attacks, such as exfiltrating secret information across privilege levels. Effective countermeasures for such attacks yet need to be explored, but could be similar to those that are being developed for FPGA-internal voltage-based side-channel attacks. Overall, we believe that system-wide power distribution needs to be designed and optimized from a security perspective against possible covert or side-channel attacks.

References

[Ali18] Instance type families – Alibaba Cloud Documentation Center, 2018. `https://www.alibabacloud.com/help/doc-detail/25378.html`.

[ASM07] K. Arabi, R. Saleh, and X. Meng. Power Supply Noise in SoCs: Metrics, Management, and Measurement. *IEEE Des. Test. Comput.*, 24(3):236–244, 2007.

[AWS18] Amazon EC2 F1 Instances, 2018. `https://aws.amazon.com/ec2/instance-types/f1/`.

[BBK89] Franc Brglez, David Bryan, and Krzysztof Kozminski. Combinational profiles of sequential benchmark circuits. In *IEEE International Symposium on Circuits and Systems,*, pages 1929–1934. IEEE, 1989. https://ddd.fit.cvut.cz/prj/Benchmarks/index.php.

[BSB+14] S. Byma, J. G. Steffan, H. Bannazadeh, A. L. Garcia, and P. Chow. FPGAs in the Cloud: Booting Virtualized Hardware Accelerators with Openstack. In *Field-Programmable Custom Computing Machines (FCCM)*, pages 109–116. IEEE, 2014.

[Cor12] D. Corbett. The Xilinx Isolation Design Flow for Fault-Tolerant Systems, 2012.

[EV12] K. Eguro and R. Venkatesan. FPGAs for trusted cloud computing. In *Field Programmable Logic and Applications (FPL)*, pages 63–70. IEEE, 2012.

[FVS15] S. A. Fahmy, K. Vipin, and S. Shreejith. Virtualized FPGA Accelerators for Efficient Cloud Computing. In *CloudCom*, pages 430–435. IEEE, 2015.

[GDT+19] J. Gravellier, J.-M. Dutertre, Y. Teglia, P. Loubet-Moundi, and O. Francis. Remote Side-Channel Attacks on Heterogeneous SoC. In *Smart Card Research and Advanced Applications, 18th International Conference, CARDIS 2019*, Pragues, Czech Republic, November 2019.

[Gir87] C. Gray Girling. Covert Channels in LAN's. *IEEE Transactions on software engineering*, 13(2):292, 1987.

[GOKT18] D. R. E. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori. An Experimental Evaluation and Analysis of Transient Voltage Fluctuations in FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(10), 2018.

[GOT17] D. R. E. Gnad, F. Oboril, and M. B. Tahoori. Voltage drop-based fault attacks on FPGAs using valid bitstreams. In *Conference on Field Programmable Logic and Applications (FPL)*, pages 1–7. IEEE, 2017.

[GRS19] I. Giechaskiel, K. Rasmussen, and J. Szefer. Reading Between the Dies: Cross-SLR Covert Channels on Multi-Tenant Cloud FPGAs. In *International Conference on Computer Design (ICCD)*, 2019.

[GYLH16] Q. Ge, Y. Yarom, F. Li, and G. Heiser. Your Processor Leaks Information-and There's Nothing You Can Do About It. *arXiv preprint arXiv:1612.04474*, 2016.

[GZBE18] M. Guri, B. Zadov, D. Bykhovsky, and Y. Elovici. PowerHammer: Exfiltrating Data from Air-Gapped Computers through Power Lines. *CoRR*, abs/1804.04014, 2018.

[HBW+07] T. Huffmire, B. Brotherton, G. Wang, T. Sherwood, R. Kastner, T. Levin, T. Nguyen, and C. Irvine. Moats and drawbridges: An isolation primitive for reconfigurable hardware based systems. In *Symposium on Security and Privacy (S&P)*, pages 281–295. IEEE, 2007.

[INK11] T. Iakymchuk, M. Nikodem, and K. Kępa. Temperature-based covert channel in FPGA systems. In *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pages 1–7. IEEE, 2011.

[JJ98] N. F. Johnson and S. Jajodia. Exploring steganography: Seeing the unseen. *IEEE Computer*, 31(2):26–34, 1998.

[KGG+18] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom. Spectre Attacks: Exploiting Speculative Execution. In *Spectre Attacks: Exploiting Speculative Execution.* IEEE, 2018.

[KGS+19] J. Krautter, D. R. E. Gnad, F. Schellenberg, A. Moradi, and M. B. Tahoori. Active Fences against Voltage-based Side Channels in Multi-Tenant FPGAs. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2019.

[KGT18] J. Krautter, D. R. E. Gnad, and M. B. Tahoori. FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs, suitable for DFA on AES. *IACR Trans. on Cryptographic Hardware and Embedded Systems (TCHES)*, (3), 2018.

[KGT19] J. Krautter, D. R. E. Gnad, and M. B. Tahoori. Mitigating Electrical-level Attacks Towards Secure Multi-Tenant FPGAs in the Cloud. *ACM Trans. Reconfigurable Technol. Syst.*, 2019.

[KGT20] Jonas Krautter, Dennis Gnad, and Mehdi Tahoori. CPAmap: On the Complexity of Secure FPGA Virtualization, Multi-Tenancy, and Physical Design. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, (3):121–146, June 2020.

[KHD$^+$08] F. Kiamilev, R. Hoover, R. Delvecchio, N. Waite, S. Janansky, R. McGee, C. Lange, and M. Stamat. Demonstration of hardware trojans. *DEFCON*, 16, 2008.

[KLP$^+$18] A. Khawaja, J. Landgraf, R. Prakash, M. Wei, E. Schkufza, and C. J. Rossbach. Sharing, Protection, and Compatibility for Reconfigurable Fabric with AmorphOS. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 107–127, 2018.

[KWKK18] S. K. Khatamifard, L. Wang, S. Köse, and U. R. Karpuzcu. A New Class of Covert Channels Exploiting Power Management Vulnerabilities. *IEEE Computer Architecture Letters*, 17(2):201–204, July 2018.

[LKG$^+$09] L. Lin, M. Kasper, T. Güneysu, C. Paar, and W. Burleson. Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In *Cryptographic Hardware and Embedded Systems (CHES)*. Springer Berlin Heidelberg, 2009.

[LMG$^+$20] Tuan Minh La, Kaspar Matas, Nikola Grunchevski, Khoa Dang Pham, and Dirk Koch. FPGADefender: Malicious Self-Oscillator Scanning for Xilinx UltraScale+ FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 13(3):1–31, 2020.

[MF04] A. V. Mezhiba and E. G. Friedman. Scaling trends of on-chip power distribution noise. *IEEE Trans. VLSI Syst.*, 12(4):386–394, 2004.

[MS19] D. Mahmoud and M. Stojilovic. Timing Violation Induced Faults in Multi-Tenant FPGAs. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1745–1750, 2019.

[Ngu18] C. D. K. Nguyen. Voltage-based Covert Channel Communication between logically separated IP Cores in FPGAs. Bachelor's thesis, Karlsruhe Institute of Technology (KIT), August 2018. https://cdnc.itec.kit.edu/downloads/Other/online_thesis_final_Khoa_Nguyen.pdf.

[PAK99] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Information hiding-a survey. *Proceedings of the IEEE*, 87(7):1062–1078, 1999.

[PCC$^+$14] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger. A Reconfigurable Fabric for Accelerating Large-scale Datacenter Services. In *Symposium on Computer Architecuture (ISCA)*, pages 13–24, Piscataway, NJ, USA, 2014. IEEE Press.

[Per05] C. Percival. Cache missing for fun and profit, 2005.

[PHT] George Provelengios, Daniel Holcomb, and Russell Tessier. Power wasting circuits for cloud FPGA attacks. In *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, pages 231–235. IEEE.

[PRP+19] G. Provelengios, C. Ramesh, S. B. Patil, K. Eguro, R. Tessier, and D. Holcomb. Characterization of Long Wire Data Leakage in Deep Submicron FPGAs. In *Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 292–297. ACM, 2019.

[RGS20] K Rasmussen, I Giechaskiel, and Jakub Szefer. CAPSULe: Cross-FPGA covert-channel attacks through power supply unit leakage. In *IEEE Symposium on Security and Privacy*, volume 1. IEEE, 2020.

[SBE11] J. Sun, R. Bittner, and K. Eguro. FPGA side-channel receivers. In *Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 267–276. ACM, 2011.

[SGMT18a] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori. An Inside Job: Remote Power Analysis Attacks on FPGAs. In *Design, Automation & Test in Europe (DATE)*. IEEE, 2018.

[SGMT18b] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori. Remote inter-chip power analysis side-channel attacks at board-level. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, pages 1–7. IEEE/ACM, November 2018.

[SSN+19] T. Sugawara, K. Sakiyama, S. Nashimoto, D. Suzuki, and T. Nagatsuka. Oscillator without a combinatorial loop and its threat to FPGA in data centre. *Electronics Letters*, 2019.

[TK10] M. Tehranipoor and F. Koushanfar. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Des. Test. Comput.*, 27(1):10–25, 2010.

[TS19] S. Tian and J. Szefer. Temporal Thermal Covert Channels in Cloud FPGAs. In *Symposium on Field-Programmable Gate Arrays (FPGA)*, 2019.

[WL06] Z. Wang and R. B. Lee. Covert and Side Channels Due to Processor Architecture. In *Annual Computer Security Applications Conference (ACSAC)*, pages 473–482, 2006.

[Xil16] Xilinx. 7 Series FPGAs Configurable Logic Block - User Guide (v1.8), September 2016.

[ZAB+18] S. Zhao, I. Ahmed, V. Betz, A. Lotfi, and O. Trescases. Frequency-Domain Power Delivery Network Self-Characterization in FPGAs for Improved System Reliability. *IEEE Transactions on Industrial Electronics*, 65(11):8915–8924, 2018.

[ZBT10] D. Ziener, F. Baueregger, and J. Teich. Using the Power Side Channel of FPGAs for Communication. In *International Symposium on Field-Programmable Custom Computing Machines*, pages 237–244, May 2010.

[ZH12] K. M. Zick and J. P. Hayes. Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2012.

[ZS18] M. Zhao and G. E. Suh. FPGA-Based Remote Power Side-Channel Attacks. In *Symposium on Security and Privacy (S&P)*. IEEE, 2018.

[ZSZF13] K. M. Zick, M. Srivastav, W. Zhang, and M. French. Sensing Nanosecond-scale Voltage Attacks and Natural Transients in FPGAs. In *Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 101–104. ACM, 2013.