DeepStrike: Remotely-Guided Fault Injection Attacks on DNN Accelerator in Cloud-FPGA

Yukui Luo◇, Cheng Gongye◇, Yunsi Fei, and Xiaolin Xu

(◇ indicates equal contribution)

Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA

Abstract—As Field-programmable gate arrays (FPGAs) are widely adopted in clouds to accelerate Deep Neural Networks (DNN), such virtualization environments have posed many new security issues. This work investigates the integrity of DNN FPGA accelerators in clouds. It proposes DeepStrike, a remotely-guided attack based on power glitching fault injections targeting DNN execution. We characterize the vulnerabilities of different DNN layers against fault injections on FPGAs and leverage time-to-digital converter (TDC) sensors to precisely control the timing of fault injections. Experimental results show that our proposed attack can successfully disrupt the FPGA DSP kernel and misclassify the target victim DNN application.

Index Terms—Neural network hardware, Field programmable gate arrays, Physical layer security

I. INTRODUCTION

The recent advancement of deep learning has made it a powerful tool in solving various challenging problems with superb performance. Many real-world applications have high throughput requirements and a stringent power consumption budget for the deep neural network (DNN) engines. Hardware-based DNN accelerators have been proposed and deployed on different computing platforms, including graphic processing units (GPUs), application-specific integrated circuits (ASICs), and field-programmable gate arrays (FPGAs). FPGA has shown unique advantages among different types of platforms, offering higher design and implementation flexibility than ASICs and higher power efficiency than GPUs [1]. Leading cloud service providers such as Amazon [2] and Microsoft [3] have integrated powerful FPGAs in their cloud servers, enabling machine learning as a service (MLaaS). The commercialization of MLaaS has facilitated deep learning in various compute-intensive applications, including medical diagnosis assistance [4] and risk and fraud management [5].

To increase the resource utilization and reduce the cost of cloud services, many recent works have enabled cloud-FPGA to be shared by multiple users, i.e., independent tenants utilize an FPGA chip in their allocated time or concurrently [6], [7]. However, such a co-tenancy usage model of cloud-FPGA also poses new security issues and creates new attack surfaces. In [8], Krautter *et al.* showed successful fault injection attacks on AES running on an FPGA, in which the adversary utilizes a periodically enabled power-hungry circuit to disrupt the FPGA power distribution network (PDN). As a result, the victim AES

circuit, sharing the same PDN, generates transient computation errors which lead to faulty ciphertext outputs. Differential fault analysis (DFA) then utilizes the faulty outputs to retrieve the secret key.

The wide deployment of DNNs on cloud-FPGA has rendered DNN engines a new vulnerable victim to potential security attacks. There have been some prior fault injection attacks on DNNs, targeting either a microcontroller using laser beam [9] or DRAMs with software row-hamming [10]. Several recent attacks on DNN FPGA implementations use hardware fault injection such as memory collisions [11], clock glitch [12], and weight loading perturbation [13].

This paper presents DeepStrike, a novel fault attack on DNN accelerators in cloud-FPGA with power-glitching fault injections. Unlike the existing fault injection attacks requiring full knowledge of the DNN model implementation, Deep-Strike deduces the execution details of the victim DNN model through side-channel analysis. Towards this goal, we propose to leverage an on-chip delay-sensor built with time-to-digital converters (TDC) [14], [15]. The delay sensor can identify the execution of different DNN layers with high temporal resolution. Informed by such execution details of the victim DNN model from TDC, the adversary can remotely guide and launch the fault injections with fine timing control. We propose a novel power striker to induce glitches on the power distribution network of cloud-FPGA, disrupting the victim DNN execution with fault injection. Unlike other commonly used power-hungry circuits, the proposed circuit scheme can pass the design rule checking (DRC), making it a viable design choice. We characterize the fault sensitivities of different types of DNN layers. With such knowledge, the fault injections are guided to target the most vulnerable DNN layers, making the end-to-end attack more efficient and stealthy. We demonstrate the effectiveness of the proposed attack with LeNet-5 architecture implementation on Xilinx PYNQ-Z1 FPGA with MNIST dataset.

The rest of the paper is organized as follows. Section II presents the background and related work. Section III illustrates the proposed DeepStrike with important components and attack procedures. Section IV describes our end-to-end attack experiments and analyzes the results. Section V concludes this paper and discusses the future work.

II. BACKGROUND AND RELATED WORK

A. Threat model

This work adopts a common threat model of cloud-FPGA used by many other related works [16]–[19]. It can be summarized as follows: 1) Enabled by FPGA virtualization, multiple users co-reside on an FPGA chip and there is no physical interaction between the circuits of different users, and these circuits can execute simultaneously [20]. 2) All users of the same cloud-FPGA chip share certain hardware resources like the power distribution network (PDN). 3) The two circuit applications running on the cloud-FPGA are from a benign user and an adversary, respectively, i.e., a DNN accelerator is the victim and a malicious circuit aims to breach the integrity of the victim execution. Additionally, we consider two more, strict but realistic, conditions in our threat model: 4) The adversary does not have implementation details of the target DNN model, nor access to the DNN input and output (i.e., a black-box attack). 5) The design rule checking of modern cloud-FPGA does not allow the implementation of combinational loop circuit, such as a ring-oscillator (RO).

B. Power Distribution Network of FPGAs

Sharing the FPGA hardware resources between different users make it possible for the adversary to interfere with other co-located benign users. Among the shared resources, the PDN of a cloud-FPGA becomes a new attack surface with all the users sharing it. Recently several attack methods targeting PDNs have been presented. Confidentiality of the victim application can be breached by passive side-channel leakage. Various on-chip sensors, such as TDC and RO, have been designed to infer the behavior of the victim FPGA users. For example, the transient power trace of a victim RSA encryption engine is sensed by RO-based power sensors for off-line key retrieval [21]. In another prior work [22], TDC is used to capture the transient voltage fluctuations of the victim application for side-channel attacks. The TDC-based delay-sensor is also constructively used as a sensor for defending the FPGA against power side-channel attacks [23]. Moreover, the integrity of the victim application on cloud-FPGA can also be compromised by active fault injections by malicious users, as detailed next.

C. Related Work

A few recent works have also explored the security of DNN implementations on FPGAs. In [11], Alam *et al.* proposed to attack the DNN model through memory collision. Specifically, they inject faults to the DNN model by writing complementary data to both ports of a memory cell. Liu *et al.* [13] utilized clock glitches to introduce timing violations to the DNN accelerators on FPGA so as to cause misclassification. In [24], Zhao *et al.* simulated the performance of DNN models under fault injection attacks. Specially, they randomly choose and flip certain parameters of the DNN model and test corresponding model accuracy.

Although these existing attacks demonstrate effectiveness in reducing the inference accuracy of DNN models, several important drawbacks have limited their practical applicability: 1) Most work [11] [13] [24] adopt a white-box attack, in which the adversary has full knowledge of the victim DNN model as well as implementation details (e.g., the memory location of DNN parameters), which is impractical. 2) Some attack scheme [24] is only validated with simulation, which may not be applicable to real FPGA DNN implementations.

III. DEEPSTRIKE ATTACK

A. Attack Overview

1) The Victim: We target DNN FPGA accelerators as the victim, leveraging parallel high-performance processing engines (PEs). These engines are typically implemented by digital signal processing (DSP) slices, the dedicated hardware units on modern FPGAs for acceleration. For example, in DNN accelerators, DSPs are mainly utilized to speed up multiplications and summations. Additionally, the DSP slice is also one of the most used hardware components by state-of-the-art Xilinx Deep Learning Processor Unit (DPU) [25].

2) The Attacker: The proposed attack mainly consists of two salient parts, namely *Attack scheduler* and *Power striker*. The attack scheduler is important and responsible for 1) Monitoring and profiling the victim DNN model execution through side-channel leakage (e.g., transient voltage fluctuation) and 2) Activating the power striker at critical timing points. Directed by the well-informed attack scheduler, the power striker will inject faults to the execution of specific DNN layers, in a targeted fashion. Specially, we utilize the TDC-based delay-sensor and a novel power-wasting circuit to construct the attack scheduler and power striker, respectively. More detailed design schemes of these two parts are presented below.

B. Attack Scheduler

The schematic of the attack scheduler is illustrated in Fig. 1(a), which mainly consists of a TDC-based delay-sensor, a clock management tile, and an encoder. Taking FPGA implementation as an example, the TDC circuit is composed of two elements: DL_{LUT}, a look-up-table (LUT) based delay-line, and DL_{CARRY}, a carry-chain built with MUX and D flip-flop. The length of DL_{LUT} determines the resolution of the TDC-based delay-sensor, and the DL_{CARRY} can scale the output range of the TDC. During the operation of TDC-based delay-sensor, two clock signals of the same frequency will be generated by the clock management tile. One clock drives the DL_{LUT}, and the other clock is for sampling the registers connected to the carry-chain outputs. There exists a phase difference θ between these two clocks, which is used for calibrating the readout. The direct output of TDC is a binary vector generated by the carry chain, which consists of different numbers of consecutive "1s" and "0s" determined by the voltage/delay. The encoder can convert these direct outputs of registers into a binary code, i.e., from 128-bit to 8-bit unsigned int value (to count the number of "1"s in the 128 bits), as the sensor readout.

Since the propagation delay of the two clock signals through the delay-lines is closely impacted by the transient voltage
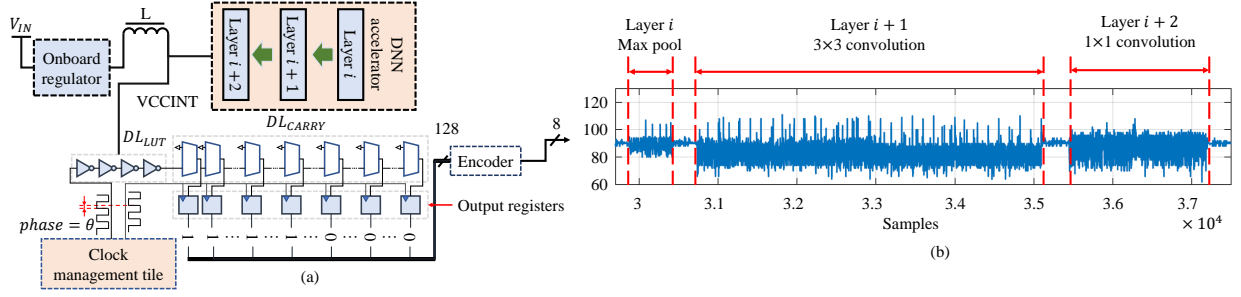
Fig. 1: (a) The proposed TDC-based delay-sensor and victim DNN accelerator sharing the power distribution network on an FPGA. (b) Voltage fluctuation associated with three DNN layers' execution collected by TDC-based delay-sensor.

level, the TDC sensor readout becomes an indicator of the real-time voltage. In other words, when the FPGA is executing applications, the voltage will fluctuate and the readings of the sensor can depict the voltage profile. As illustrated in Fig. 1, while the TDC-based delay-sensor shares PDN with another circuit application (e.g., DNN model), its readout can be used to profile the voltage fluctuation caused by the execution of that application. In practical usage, the driving clock frequency (F_{dr}) and the length of DL_{LUT} (L_{LUT})and DL_{CARRY} (L_{CARRY}) should be carefully designed to avoid counting errors.

A primary challenge for a remotely-guided fault attack on a multi-user FPGA is that the attacker does not have knowledge of the model execution. To mitigate this issue, we propose to use the TDC-based delay-sensor to profile and infer the target DNN model execution. In our preliminary study, we sequentially execute three layers of a DNN model: a max-pooling layer, a convolutional layer with a 3×3 kernel, and a convolution layer with a 1×1 kernel. Meanwhile, the TDC readout is collected in parallel. The specific configuration of the TDC-based delay-sensor for this victim is $F_{dr} = 200MHz$, $L_{LUT} = 4$, $L_{CARRY} = 128$, and we calibrate θ to get approximate 90 consecutive "1" outputs when the FPGA works under a nominal voltage. Fig. 1(b) gives a tracing example for the tested DNN execution, which shows that the sensor readouts clearly present different patterns for executions of different DNN layers. We also notice clear "stalls" between different layer executions (the readout stays around 90), and the fluctuation during convolutional layers' execution is much larger than that of the max-pooling layer. Therefore, we conclude that the side-channel leakage of the victim DNN model execution can be used to build a library of sensor readout patterns for different types of DNN layers at different sizes for future attack use.

C. Power Striker

Another important component of the proposed DeepStrike attack is the power striker. It is a malicious controllable power-wasting circuit used for aggressively overloading the shared PDN, incurring well-timed voltage glitches. The design requirement for the power striker is even when the malicious circuit is activated for a short period (e.g., a few clock cycles),
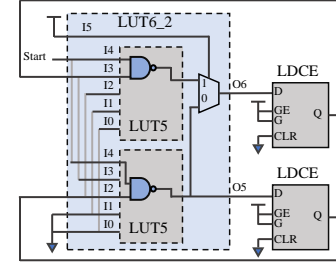

Fig. 2: A controllable power striker design scheme.

it draws a significant amount of power, creating an immediate voltage drop on the shared PDN. As a result, the voltage drop increases the signal propagation time in FPGA components that share the same PDN, inducing timing violations and computation or data loading faults [8]. Previous works mainly utilized LUT-based combinational loops (e.g., RO) to construct such malicious circuits [6], [26]. Although those circuit schemes are effective, they trigger design rule checking (DRC) warnings and are commonly banned by security and privacy-sensitive cloud-FPGAs [27].

We develop a circuit scheme that can pass the DRC checking by inserting data latches in the combination loop, for our *power striker*. Fig. 2 depicts the basic circuit cell, which utilizes a two-output LUT ($LUT6\_2$) with two latch registers (LDCE). When enabled (Start=1), the $LUT6\_2$ is configured as two parallel inverters, with their outputs, $O6$ and $O5$, connected to two LDCEs, respectively, to form two oscillating loops. Compared with the combinational loop, this method increases the loop's length and utilizes one LUT for two self-oscillating loops. As a result, the proposed circuit scheme can provide higher attack efficiency with less hardware overhead. Moreover, it can pass the DRC checking. An adversary of cloud-FPGA can apply a large number of such power striker cells, and use the Start signal to control the duration of their activation.

D. Attack Scheduler and Power Striker Integration

As described in Sec. II-A, our threat model assumes that a practical attacker may not have any knowledge about the victim DNN model's parameters. Thus, a fine-tuning attack on the specific weight or pixel computing is impossible. Instead,

DeepStrike targets at activating the power striker multiple times, starting at a guided moment by the attack schedule, i.e., during the execution of a particular DNN layer. As illustrated in Sec. III-B, the TDC-based delay-sensor can be used to track and characterize the execution of target DNN. Once enough characteristics (e.g., time duration, TDC readout, etc.) for each distinct DNN layer are gathered, we can build a profile to assist with scheduling the activation of the power striker. Practically, the profiling procedure can be accomplished during the normal target DNN model, i.e., classifying different input images.

Fig. 4 shows the integrative schematic of the attack scheduler and power striker, including some other auxiliary circuits/components like the *DNN start detector* and *signal RAM*. The design schemes and functionalities of these components are as follows.

1) DNN start detector: From Fig. 1(b), we can observe that there always exist small voltage fluctuations (i.e., the "stall" zones) on the FPGA PDN even when the DNN models are not being executed. These small voltage fluctuations, although can be detected by the TDC-based delay-sensor, cannot be used to guide the proposed attacks. Thus, to filter out the impact of these small voltage fluctuations, we need to purify the voltage fluctuation sensed by the TDC sensor. To realize this, we build the *DNN start detector* with a finite-state machine (FSM), with its inputs connected to outputs of the TDC-based delay-sensor. We partition the 128-bit TDC output into five zones, and select 1-bit from each zone as the input of the *DNN start detector*. Leveraging such voltage fluctuation purification, we apply the *DNN start detector* to detect the DNN model (The same DNN model we used in Fig. 1)execution, and the results are shown in Fig. 3. Compared to the results by the TDC-based delay-sensor shown in Fig. 1(b)), the purified voltage fluctuation can provide more accurate and controllable guidance to start the DeepStrike attack. For example, when the *DNN start detector* gets an input Hamming weight (HW) equals to 3, indicating the first layer - MaxPool just starts, we set up a "start point" for our attack scheduler.
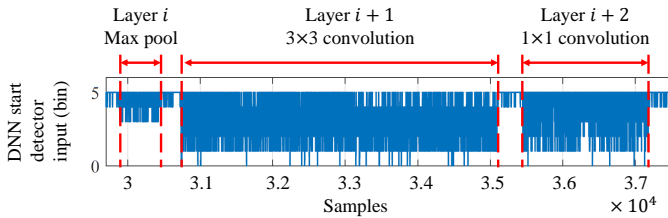

Fig. 3: Input of the DNN start detector.

2) Signal RAM: To make the proposed attack configurable, we develop another component *signal RAM* with the on-chip BRAM, which is used to store the *attacking scheme file*. The attacking scheme file mainly includes three parameters: *attack delay*, *attack period*, and the *number of attacks*. Specifically, these parameters are denoted as binary vectors and each bit represents the action of DeepStrike during a separate clock cycle. We use "1" to enable and "0" to disable the power striker, respectively. Therefore, the parameter *number of*
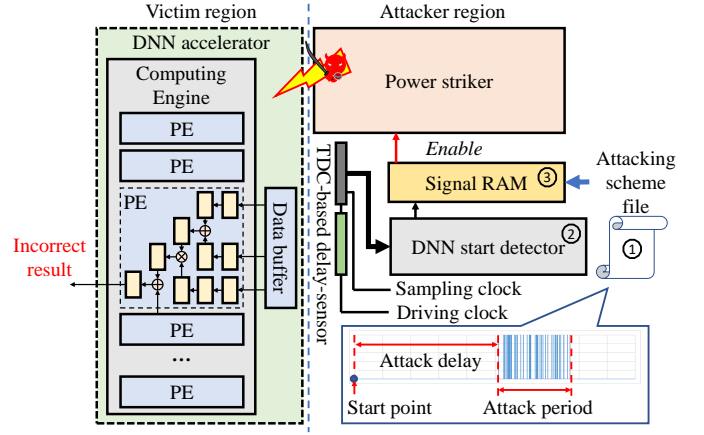

Fig. 4: Integrative schematic of DeepStrike.

attacks can be configured by using different 1/0 composition. Additionally, to control the time duration (i.e., clock cycles) elapsed before enabling a power strike, we define *attack delay*, which is represented by a series of "0s". With the *signal RAM* (i.e., on-chip BRAM) being read at a specific clock frequency f_{sRAM}, the duration of *attack delay* is jointly determined by the number of "0s" in it and f_{sRAM}. For example, a *attack delay* consisting N "0s" will pause DeepStrike for N clock cycle, with time duration of $\frac{N}{f_{sRAM}}$. Similarly, the duration of *attack period* is configured in this way with consecutive "1"s.

In summary, the proposed DeepStrike attack can be accomplished in three steps: 1) Profiling the voltage fluctuation associated with the victim DNN accelerator execution to make a corresponding attack plan, i.e., determining these different parameters like *number of attacks*, *attack delay*, etc., store these parameters in the *signal RAM*; 2) Using *DNN start detector* to sense the execution of victim DNN accelerator; 3) Launching DeepStrike following the pre-scheduled attack strategy in *signal RAM*. We would like to highlight that with the proposed attack scheme, the attacker have high flexibility to load different attack strategies at run-time, i.e., dynamically target at different DNN layers.

IV. EXPERIMENTAL SETUP AND VALIDATION RESULTS

In this section, we present an end-to-end attack experiment on a PYNQ FPGA evaluation kit, which is an open-source project that integrates the Linux system with the Xilinx FPGA. Here we apply the Xilinx PYNQ-Z1 FPGA board to build a prototype of the cloud-FPGA. Without loss of generality, in our experimental validation, we choose an open-source DNN accelerator engine [28] and train a LeNet-5 neural network [29], [30] with the MNIST dataset [31]. In our threat model, the hypervisor in the virtualized cloud-FPGA will compile and combine applications of all the tenants (including the attacker's malicious circuits and the victim's DNN inference), generate an unified bitstream and deploy it on one FPGA device [7]. Note that although the tenants co-locate on the FPGA, they do not share hardware including the I/O bus, BRAM, and clock sources. In our experiment, the adversary connects to
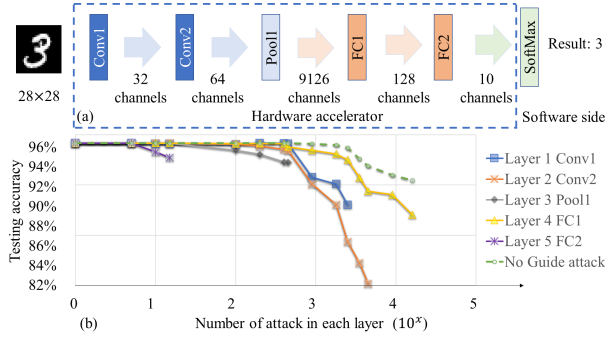
Fig. 5: Case study: apply DeepStrike on MNIST application.

this prototyped cloud-FPGA from the UART serial port, with which the adversary can gather on-chip side-channel leakage from the TDC-based delay-sensor and dynamically configure the the *attacking scheme file*.

The pre-trained LeNet-5 model on the MNIST dataset is deployed on the prototype cloud-FPGA. The data type of the model is fix-point 8-bit value, with 3-bits for the integer and the rest for the mantissa representation. The MNIST dataset includes $60,000$ training samples and $10,000$ testing samples. Our un-tampered model achieves a testing accuracy of 96.17% on the FPGA. The architecture of LeNet-5 is shown in Fig. 5(a), which consists of two convolutional layers for feature extraction (Conv1 and Conv2), one pooling layer for downsampling (Pool1), and two fully connected layers (FC1 and FC2) for classification. The output of the FC2 is a vector of 10 prediction scores, which go through a SoftMax layer to pick the class with the largest score as the prediction. Note as we use the unsigned fixed-point quantization method, the activation function we use in this case study is the hyperbolic tangent (tanh).

We target each layer separately and apply a series of fault injections while the corresponding acceleration kernel is executing, guided by the *attack scheduler*. The power striker circuit consumes 15.03% logic slices, and each power glitching strike lasts for $10ns$. We observe the inference accuracy to evaluate the end-to-end effect of fault injections on different layers. Fig. 5 (b) shows that the testing accuracy drops as the number of power strikes increases. Note that due to the different execution length of different layers, the maximum number of strikes on different layer also varies. As observed in the results, CONV2 is the most fault-sensitive layer, and the maximum accuracy drop reaches 14% when 4500 strikes are applied. Additionally, we provide the results of non-TDC guiding attacks as our baseline, which is the top curve, where the fault injections happen randomly along with the model execution. We conclude that our proposed TDC guiding DeepStrike fault attack is much more efficient than the blind attack while applying the same attack intensity.

Moreover, the experimental results show that the vulnerability to the power glitching fault injections of each layer depends on the layer's type, the layer's size, and its execution time. As CONV2 is larger than CONV1 and takes longer to execute,

more fault injection strikes can be applied onto CONV2 and result in the largest testing accuracy reduction. FC1 takes the longest time to execute. However, it is a fully connected layer and only adds $k \times k$ prior multiplication results to generate one pixel in a feature map. Convolution layers contain more complex multiplications. We find that these most vulnerable layers (e.g., CONV2 and FC1) are implemented with DSP slices. One reason that DSP slice-based DNN layers are more vulnerable lies in the design rules. To increase the performance of the DNN accelerators, the designers usually adopt double-data-rate while using DSP, enabling doubled running speed of the DSP slices compared to other components. This design choice, although makes the DSP slices faster, also renders it more vulnerable to fault injection attacks due to the tighter timing constraints.

A. Fault Characterization of DSP Slices under Power Strikes

We designed experiments to investigate the faults in DSP slices caused by power glitching strikes. The layout of the attack is shown in Fig. 6a. We put the victim circuit far from the attacker circuit to minimize the influence of temperature changes, which sharing the PDN. The DSP slices are configured to add two inputs and multiply with the third input, which is the configuration for convolution computation [1]. Since the DSP slices do not have a result-ready signal, we designed a circuit that fetches the result of the DSPs after five clock cycles. This circuit works correctly and the timing analysis of the FPGA mapping tool does not complain about violations of timing constraints.

We fed the DSP slices 10,000 randomly generated inputs and launched the power striker circuit for one clock cycle at the same time we enabled the DSP slices. According to our experiment, we only need to enable the attack for one cycle to induce fault in a single DSP computation operation. Enabling the power striker circuit longer will work as well but it may increase the temperature of the FPGA chip or even crash it.

We observed two types of faults from the experimental results, namely 1) Duplication fault, where the DSP output is the correct result of the previous input. In this case, the DSP computation simply takes more cycles to complete and cannot produce the correct result in time; and 2) Random faults, where the faulty output does not have obvious patterns. In Fig. 6b, we demonstrated both types of faults, in which the x-axis is the number of power striker cells, and the y-axis denotes the fault rate, number of faults divided by the total number of experiments. The experimental result shows that we can control the fault injection intensity by adjusting the number of power striker cells. For example, the total fault rate[2] is nearly 100% with 24,000 power strike cells.

In conclusion, the power glitching fault injection results in random faults or duplication faults in the DSP slices. With

---

[1]The fully connected layers are usually implemented on DSP slices with these configurations as they could be treated as a special case of convolution

[2]The total fault rate is the sum of duplication fault rate and random fault rate. When launching the proposed attack in practice, much fewer power striker cells is needed because other victim components also consume power, further reducing the voltage of the PDN and strengthening fault injection.

(a) The Xilinx FPGA (XC7Z020) layout for DSP fault injection test.

(b) Duplication fault rate and random fault rate of double data rate DSP slices with different numbers of power striker cells.
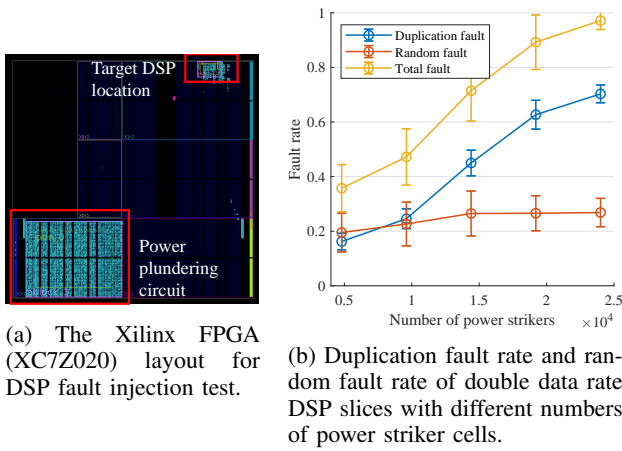
Fig. 6: DSP fault injection test: configurations and results.

duplication faults, the correct product appears in the next clock cycle, and can be absorbed by more serial summations, mitigating the adverse impact of stale results in `FC` layers. Also convolutional layers involve much more multiplications, possibly experiencing more random faults and making them more vulnerable. These experimental results well explain that `FC1` achieves much less accuracy reduction than `CONV2` under the same number of fault injection strikes.

V. CONCLUSION AND FUTURE WORK

We demonstrate DeepStrike, a remotely-guided power glitching fault injection attack targeting DNN accelerators in cloud-FPGA. Different from other attacks that require implementation details of the victim DNN model, Deep-Strikeleverages the voltage fluctuations associated with the on-chip DNN model execution as a side-channel information, to launch well-scheduled fault injection attacks. We prototyped an experimental cloud-FPGA on a PYNQ FPGA development board, and conducted end-to-end attacks to validate the effectiveness of the proposed attack scheme on a LeNet-5 neural network trained with the MNIST dataset. The experimental result demonstrates that the proposed attack scheme can significantly lower the inference accuracy. We also investigate the possible reasons why different DNN model layers show different resilience against power glitching fault injections.

In future work, we plan to extend the proposed attack scheme to more complicated execution environments, e.g., more than three tenants on the FPGA, which may be representative multi-user scenarios for cloud-FPGA. We will also consider more DNN architectures, and experiment with commercial cloud-FPGAs.

REFERENCES

[1] E. Nurvitadhi, G. Venkatesh *et al.*, "Can fpgas beat gpus in accelerating next-generation deep neural networks?" in *FPGA*.
[2] "Amazon ec2 f1," https://aws.amazon.com/ec2/instance-types/f1/.
[3] "Inside the microsoft fpga-based configurable cloud," 2017, https://azure.microsoft.com/en-us/resources/videos/build-2017-inside-the-microsoft-fpga-based-configurable-cloud/.
[4] "Ai, machine learning as a service set to overhaul healthcare," https://healthitanalytics.com/news/ai-machine-learning-as-a-service-set-to-overhaul-healthcare.
[5] "Global machine learning as a service (mlaas) market is projected to reach a value of over usd 12.7 billion by 2027," https://forencisresearch.medium.com/global-machine-learning-as-a-service-mlaas-market-is-projected-to-reach-a-value-of-over-usd-12-7-7aca50e695b.
[6] G. Provelengios, D. Holcomb, and R. Tessier, "Power wasting circuits for cloud fpga attacks," in *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*.
[7] Y. Zha and J. Li, "Virtualizing fpgas in the cloud," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*.
[8] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "Fpgahammer: Remote voltage fault attacks on shared fpgas, suitable for dfa on aes," *IACR TCHES*, Aug. 2018.
[9] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu, "Practical fault attack on deep neural networks." New York, NY, USA: Association for Computing Machinery, 2018.
[10] F. Yao, A. S. Rakin, and D. Fan, "Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," in *29th USENIX Security Symposium (USENIX Security 20)*.
[11] M. M. Alam, S. Tajik, F. Ganji, M. Tehranipoor, and D. Forte, "Ramjam: Remote temperature and voltage fault attack on fpgas using memory collisions," in *FDTC'19*.
[12] A. S. Rakin, Y. Luo, X. Xu, and D. Fan, "Deep-dup: An adversarial weight duplication attack framework to crush deep neural network in multi-tenant fpga," *arXiv preprint arXiv:2011.03006*, 2020.
[13] W. Liu, C. H. Chang, F. Zhang, and X. Lou, "Imperceptible misclassification attack on deep learning accelerator by glitch injection," in *DAC'2020*.
[14] D. R. Gnad, F. Oboril, and M. B. Tahoori, "Voltage drop-based fault attacks on fpgas using valid bitstreams," in *FPL'17*.
[15] K. M. Zick, M. Srivastav, W. Zhang, and M. French, "Sensing nanosecond-scale voltage attacks and natural transients in fpgas," in *FPGA'13*.
[16] C. Ramesh, S. B. Patil *et al.*, "Fpga side channel attacks without physical access," in *FCCM'18*.
[17] I. Giechaskiel *et al.*, "Leaky wires: Information leakage and covert communication between fpga long wires," in *AsiaCCS*.
[18] S. Yazdanshenas and V. Betz, "The costs of confidentiality in virtualized fpgas," *IEEE TVLSI*, 2019.
[19] A. Khawaja, Landgraf *et al.*, "Sharing, protection, and compatibility for reconfigurable fabric with amorphos," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*.
[20] Y. Luo and X. Xu, "Hill: A hardware isolation framework against information leakage on multi-tenant fpga long-wires," in *International Conference on Field-Programmable Technology (ICFPT)*. IEEE, 2019, pp. 331–334.
[21] M. Zhao and G. E. Suh, "Fpga-based remote power side-channel attacks," in *2018 IEEE Symposium on Security and Privacy (SP)*.
[22] F. Schellenberg *et al.*, "An inside job: Remote power analysis attacks on fpgas," in *2018 DATE*.
[23] D. R. Gnad, S. Rapp *et al.*, "Checking for electrical level security threats in bitstreams for multi-tenant fpgas," in *FPT*.
[24] P. Zhao, S. Wang, C. Gongye *et al.*, "Fault sneaking attack: A stealthy framework for misleading deep neural networks," 2019.
[25] Xilinx. (2020, jul) Zynq dpu v3.2 product guide. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/dpu/v3_2/pg338-dpu.pdf
[26] T. M. La, K. Matas *et al.*, "Fpgadefender: Malicious self-oscillator scanning for xilinx ultrascale+ fpgas," *TRETS*.
[27] T. Sugawara, K. Sakiyama *et al.*, "Oscillator without a combinatorial loop and its threat to fpga in data center," *Electronics Letters*, 2019.
[28] "Yolov2 accelerator in xilinx's zynq-7000 soc," https://github.com/dhm2013724/yolov2_xilinx_fpga.
[29] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks."
[30] E. Wang, J. J. Davis, and ohters, "A PYNQ-based Framework for Rapid CNN Prototyping," in *FCCM*, 2018.
[31] "The mnist database of handwritten digits," http://yann.lecun.com/exdb/mnist/.