Detection and Classification of Malicious Bitstreams for FPGAs in Cloud Computing

Jayeeta Chaudhuri and Krishnendu Chakrabarty
Department of Electrical and Computer Engineering, Duke University
Durham, NC 27708, USA
{jayeeta.chaudhuri,krishnendu.chakrabarty}@duke.edu

ABSTRACT

As FPGAs are increasingly shared and remotely accessed by multiple users and third parties, they introduce significant security concerns. Modules running on an FPGA may include circuits that induce voltage-based fault attacks and denial-of-service (DoS). An attacker might configure some regions of the FPGA with bitstreams that implement malicious circuits. Attackers can also perform side-channel analysis and fault attacks to extract secret information (e.g., secret key of an AES encryption). In this paper, we present a convolutional neural network (CNN)-based defense to detect bitstreams of RO-based malicious circuits by analyzing the static features extracted from FPGA bitstreams. We further explore the criticality of RO-based circuits in order to detect malicious Trojans that are configured on the FPGA. Evaluation on Xilinx FPGAs demonstrates the effectiveness of the security solutions.

ACM Reference Format:
Jayeeta Chaudhuri and Krishnendu Chakrabarty. 2023. Detection and Classification of Malicious Bitstreams for FPGAs in Cloud Computing. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3566097.3568346

1 INTRODUCTION

Field-programmable gate-arrays (FPGAs) are now integrated in various cloud computing infrastructures and reconfigurable system-on-chips (SoCs). The availability of FPGA in cloud data centers has enabled users to improve application performance by enabling them to implement customizable hardware accelerators directly on the FPGA fabric. In addition to increasing computational efficiency at reduced cost, partial reconfiguration allows new types of FPGA designs that would be otherwise impossible to implement. Consequently, Amazon and Microsoft have incorporated them for specialized compute-intensive services [1] [2].

Multi-tenant FPGAs are split into logically isolated regions that can be occupied by multiple users at a time. A majority of integrated circuits are supplied by a common power distribution network (PDN) for the entire FPGA board. As a result, an electrical connection exists between the victim and attacker modules [17]. An attacker can use voltage sensors such as ring oscillators (ROs) and time-to-digital converters to measure voltage fluctuations caused at the victim end. Moreover, a grid of ROs can be activated simultaneously to generate high-frequency oscillations; this may overheat the FPGA and launch a denial-of-service (DoS) attack [4].

In order to prevent the attacker from directly configuring the FPGA with an invalid or malicious bitstream, various countermeasures have been adopted that detect and block such bitstreams before loading them to the FPGA fabric. In [5], the FPGA bitstream structures are scanned to detect signatures that satisfy the requirements of FPGA-based fault attacks. However, [5] requires reverse engineering (RE) of the bitstreams to their corresponding netlists; this technique is computationally intensive. Also, the RE tools are specific to each FPGA family and each FPGA vendor. Therefore, applying RE techniques for malicious bitstream detection is not always practical.

In this paper, we propose a convolutional neural network (CNN)-based methodology to learn malicious RO-like signatures from the data-series representation of bitstreams and detect a malicious bitstream before it is used for FPGA configuration. The key contributions of this paper are as follows:

- Generation of different RO variants and loop-free ROs, which have been identified as a rising threat to cloud FPGAs;
- Extraction of malicious features from FPGA bitstreams;
- A CNN-based classification framework that learns features extracted from RO patterns;
- Evaluation of the proposed solution for multiple FPGA families;

ASPDAC '23, January 16–19, 2023, Tokyo, Japan
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9783-4/23/01.
https://doi.org/10.1145/3566097.3568346

- Criticality classification of FPGA bitstreams using Fourier transform-encoded images and identification of RO-based Trojans that are capable of launching power and voltage-based attacks.

The remainder of the paper is organized as follows. Section II presents the threat model, discusses the possible security concerns for multi-tenant FPGAs, and describes related prior work on FPGA bitstream checking. Section III describes the overall CNN-based feature extraction framework for malicious bitstream classification. In Section IV, we propose a CNN-based method to classify ROs as benign or critical, based on their Fourier transform-encoded representations. Section V presents the experimental results. Section VI concludes the paper.

2 BACKGROUND, RELATED WORK, AND THREAT MODEL

2.1 Threat Model

An FPGA PDN is represented by an RLC circuit. The voltage drop V_{drop} in this network is given by the equation: $V_{drop} = IR + L \times di/dt$ where R is the resistance of the PDN, L is the inductance of the PDN, and di/dt is the rate of change of the electric current inside the PDN, which depends on the workload that the PDN is subjected to [4]. ROs generate oscillations with a frequency that depends on the gate delays of the inverters. High-frequency oscillation of ROs have been shown to maliciously affect the power consumption of the FPGA and ultimately lead to DoS [8].

In multi-tenant FPGAs, both the attacker and victim can configure the FPGA with their own modules. Although these modules are logically isolated from each other and the attacker has no physical access to the FPGA device, they may still be capable of launching voltage drop-based attacks and side-channel attacks; these attacks can significantly affect the victim FPGA module [4]. Fig. 1 illustrates the threat model with the possible attacks on cloud-based FPGAs.

Our proposed CNN pipeline for malicious bitstream detection is based off-chip. The end-user inputs a bitstream to the pre-trained CNN model for authentication before loading it to the FPGA for configuration. If the CNN classifies the bitstream as malicious, it is blocked from FPGA configuration. Fig. 2 illustrates the proposed CNN pipeline.

2.2 Types of Attacks on Multi-tenant FPGAs

FPGA-based systems are subject to power analysis side-channel attacks, fault attacks, as well as voltage drop-based attacks. The work in [8] demonstrates a DoS attack as well as a timing fault-based attack by deploying a grid of ROs on the FPGA fabric. In [12], customized power sensors using delay lines have been configured on a multi-tenant FPGA, which

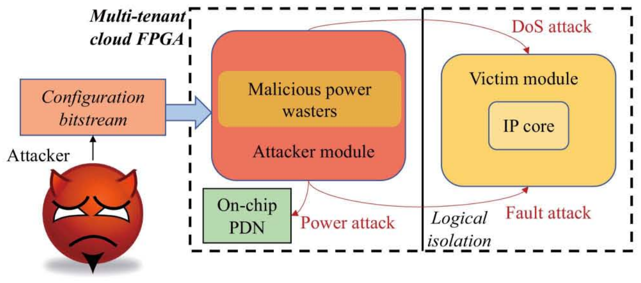


Figure 1: Illustration of the threat model.

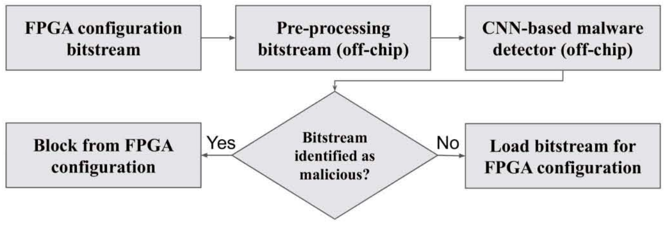

FPGA configuration bitstream
Pre-processing bitstream (off-chip)
CNN-based malware detector (off-chip)
Bitstream identified as malicious?
Yes
No
Block from FPGA configuration
Load bitstream for FPGA configuration

Figure 2: CNN pipeline for malicious bitstream detection.

perform side-channel analysis attacks on an AES-128 core present in the same FPGA. In [4], it has been demonstrated how a DoS attack on an FPGA requires only a small number of ROs, occupying about 12% of the available LUTs. In [6], benign circuits such as ripple-carry adders are utilized as voltage sensors. Such circuits evade any bitstream-checking mechanism and can still be used to extract the AES key by performing correlation power analysis (CPA) attack.

Non-combinational oscillators have been proposed in [15]; these circuits escape design rule check (DRC) by FPGAs in the cloud e.g., in the case of Amazon Web Services (AWS). As AWS rejects a design containing combinational loops, an attacker can resort to generating loop-free oscillators, which can be a major threat to cloud FPGAs.

2.3 Prior Work on Malicious Bitstream Detection

Several methods have been proposed to check bitstreams before using them to configure an FPGA. The *icebox_vlog* tool is used to reverse-engineer a bitstream to the technology-mapped netlist [5]. The *Yosys* tool is extended to analyze combinational cycles and other similar patterns to detect malicious structures. However, the development of these reverse-engineering tools is complex and time-consuming; moreover, the tools vary from one FPGA to another.

An approach to analyze FPGA bitstreams using neural networks is presented in [10]. The dataset used in this work consists of partial bitstreams with different IPs, including adders, multipliers and subtractors. It focuses on partial bitstreams because they can be trained faster, as compared to full bitstreams. However, [10] does not consider the more complicated cases of detecting malicious ROs and RO variants, especially when they are embedded within larger designs. Machine learning-based approaches for malicious circuit detection (especially hardware Trojans) have been proposed in [16] and [7]. These methods detect malicious circuits either from gate-level netlists or by using frequency domain signals and layout images.

Table 1 provides a qualitative comparison of this work with prior work on malicious bitstream detection before the bitstream is configured on the FPGA. Note that [5], [9], and [11] require computation-intensive reverse-engineering (RE) techniques to generate the technology-mapped netlist from the FPGA bitstream.

3 CNN-BASED FEATURE EXTRACTION

3.1 Data Collection

We have generated a large dataset consisting of benign and malicious bitstreams that are used to configure an FPGA. We implement the circuits corresponding to the benign and malicious bitstreams in Verilog. The bitstreams generated in this work are full bitstreams represented as .bin format (i.e., binary data files without the ASCII header at the beginning of the file). We use full bitstreams because they are of fixed size and can configure the entire FPGA at a single shot. A full bitstream is used in time-sharing applications where multiple users can access the FPGA at different times. The following bitstreams are used in our experiments:

Benign Bitstreams: We generated bitstreams that implement arithmetic cores, keyboard controllers, AES cores, MIPS cores, and VGA OpenCores. We also generated bitstreams that implement ISCAS '85, ITC '99, and EPFL benchmarks. We ensured that these bitstreams are representative of data used in real-life benign applications.

Malicious Bitstreams: We generated bitstreams that implement circuits that are capable of causing significantly high voltage fluctuations and power-based attacks on the FPGA, thereby leading to DoS. We focus on simple ROs as well as non-combinational ROs. Non-combinational or loop-free ROs escape DRC and also supports successful bitstream generation on cloud FPGAs. We implemented the self-clocked and latched non-combinational ROs and requested bitstream generation using the *write_bitstream* command. FPGA configuration bitstreams are generated for both the latch-based RO and the self-clocked RO, supporting the claim that they are capable of evading DRC.

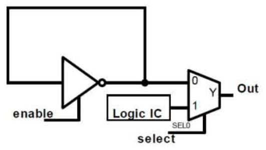


Figure 3: Conditionally active RO.

Conditionally active RO: An attacker might not use a bare RO for FPGA configuration. Instead, they might conceal the malicious activity using conditionally active ROs. Hence, we generate MUX-based conditionally active ROs and include them in our experimental dataset. An implementation of a conditionally active RO is shown in Fig. 3.

3.2 Mapping Bitstreams to Data-Series Representation

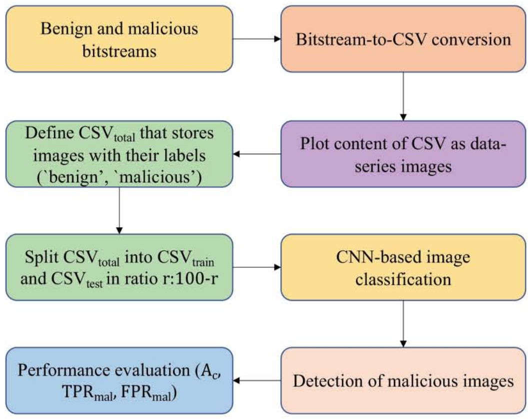
After generating the dataset of benign and malicious bitstreams, we proceed to convert the bitstreams to their corresponding data series. The obtained data is then represented as image files for each bitstream. Representing bitstreams as images enables us to:

- Identify specific patterns in the images that represent malicious behaviour;
- Utilize a CNN-based framework that learns malicious patterns in images;
- Apply image augmentation to increase the size of the training dataset and enhance the model performance.

Note that the procedure of bitstream generation takes ~ 15 minutes and plotting the bitstream as 2D data series takes

Table 1: Comparison of our work with previous malicious bitstream detection methods.

Characteristics	[5]	[9]	[11]	**Proposed method**
Types of attacks prevented	Fault and power-based	Power-hammering	DoS, timing faults	Voltage and power-based, DoS
Bitstream type	Full	Partial	Partial	Full
Dataset	Bitstreams converted to technology-mapped netlist	Bitstreams converted to netlist graphs	Manipulating existing bitstreams	Bitstreams plotted as data-series, FFT-encoded images
RE used?	Yes	Yes	Yes	No
Self-clocked ROs analyzed?	No	Yes	No	Yes
Conditional ROs analyzed ?	No	No	No	Yes
Features extracted	Combinational cycles	Combinational cycles, invalid routing	Interconnects	Combinational and non-combinational cycles
Experimental framework	*icebox_vlog*, *yosys* tools	BitMan	Reconfiguration-based defense	CNN-based feature extraction
Extended to new FPGA families?	No	Yes	Yes	Yes



Benign and malicious bitstreams
Bitstream-to-CSV conversion
Plot content of CSV as data-series images
Define CSV$_{total}$ that stores images with their labels (`benign', `malicious')
Split CSV$_{total}$ into CSV$_{train}$ and CSV$_{test}$ in ratio r:100-r
CNN-based image classification
Detection of malicious images
Performance evaluation (A$_c$, TPR$_{mal}$, FPR$_{mal}$)

Figure 4: Steps involved in the training and evaluation of the CNN-based malicious bitstream detector.

upto 10 minutes. Therefore, generating a large dataset of benign and malicious bitstreams and then converting them to image files is time-consuming. Image augmentation, a well-known type of data augmentation technique, is used in such scenarios. In this work, we specifically apply image augmentation for the following reasons:

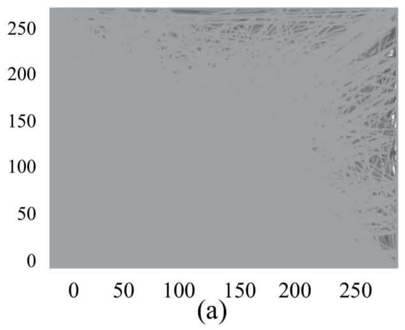
- Image augmentation improves the performance of the CNN model by extracting meaningful features (in this case, RO-like patterns) from image representations of bitstreams and using them to classify the image files as being either benign or malicious.
- It artificially expands the training dataset with new and realistic examples from existing training data.

We use the data-series representation of bitstreams in our training dataset to train our CNN-based model. To obtain the bitstream as data series, we first convert the bitstreams into comma-separated values (CSV) files. Next, we plot the content of each CSV file as two-dimensional series data and store them as image files (.png format).

3.3 Malicious Bitstream Detection using CNN-based Framework

After obtaining the training dataset, we apply our CNN-based image classification framework for learning and evaluation. We present the overall steps involved in the CNN-based detection model in Fig. 4. The CNN used in our work has four convolutional layers, four max pooling layers, and four linear layers. We select this architecture for the following reasons:

(1) The image received at the input layer may have noise included in it. To account for this, we add more convolutional layers in our model and attempt to extract meaningful features as the network gets deeper.



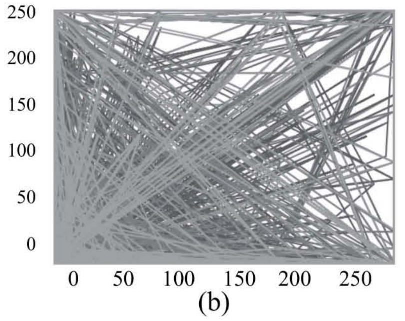


Figure 5: Images corresponding to: (a) benign bitstream; (b) malicious bitstream.

(2) In the specific problem of identifying malicious patterns in FPGA bitstreams, we utilize gray-scale images for our CNN model. However, extracting meaningful features from a gray-scale image is much more complex than extracting features from a colored image. In such a scenario, it is desirable to have more linear layers to improve the performance of the model in such a scenario. We select the Rectified Linear Unit (ReLU) as our activation function because it trains the model faster and efficiently, without causing a significant drop in classification accuracy [3].

Further, we add a dropout layer after every maxpooling layer to prevent overfitting. We perform hyperparameter tuning to select the dropout value p. We have considered values of p in the range $0.1 < p < 0.8$. However, in our experiments, we use $p = 0.25$ because it gives the best classification accuracy on the test dataset.

The image files corresponding to a benign bitstream and a malicious bitstream are shown in Fig. 5(a) and Fig. 5(b), respectively. The following qualitative observations are obtained from the patterns in the benign and malicious images. For malicious bitstreams, the intensity of patterns across the image is not uniform; it follows a non-uniform distribution. On the other hand, we observe a higher intensity of a particular pattern in the image corresponding to a benign bitstream; the same region appears with a lesser intensity in the case of a malicious bitstream. Such observations guide us to use

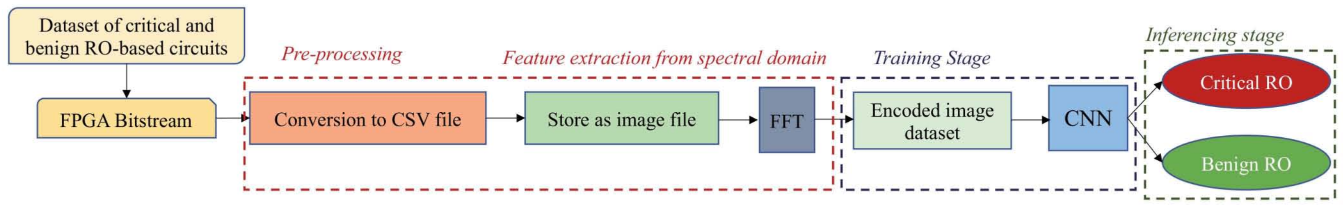


Figure 6: Criticality analysis of ROs present in FPGA configuration circuits.

these specific attributes as features to train our proposed CNN-based malicious bitstream detector.

4 CRITICALITY ANALYSIS OF RO-BASED CIRCUITS

We next look at different implementations of ROs in circuits employed in real-world applications. Note that there exist RO-based digital circuits that are used for genuine, real-life applications, e.g., true random number generators, phase-locked loops and so on. Therefore, if we utilize a mechanism that blocks all circuits with ROs, it can incorrectly block benign circuits that require ROs for correct operation. Hence, we focus on performing a criticality classification of RO-based circuits. This step is essential for the following reasons:

- ML-based criticality analysis will help detect RO-based Trojans before they are configured on the FPGA, with a high classification accuracy;
- Prevent misclassification of ROs used for legitimate purposes;

We proceed to identify abstract features in the spectral domain that distinguish RO-based Trojans from benign RO-based circuits. Fig. 6 illustrates the proposed strategy. In order to perform criticality analysis, we convert all the images in the experimental dataset to the Fourier domain using Fast Fourier Transform (FFT) and pass them through our curated CNN model for training and future evaluation. Fig. 7 shows the FFT-encoded images corresponding to a benign RO and a critical RO-based Trojan. The images represent the frequencies and the corresponding amplitudes present in the original data-series representation of the FPGA bitstreams. From Fig. 7, we can clearly classify between benign and critical ROs. We train our CNN model on FFT-encoded images of the benign and critical RO-based circuits. The CNN architecture is carefully chosen to increase the classification accuracy.

5 EXPERIMENTAL RESULTS

5.1 Experimental Setup

We implement the malicious power-wasting circuits using Verilog. We obtain the benign circuits from several benchmarks and OpenCore repository. Next, we generate bitstreams corresponding to the benign and malicious circuits using Xilinx Vivado 2018.2. We focus on the Virtex Ultrascale FPGA in all of our experiments.

We implement the overall CNN-based classification framework using *Pytorch*. The CNN is trained using the *Adam* optimizer, with a learning rate of 0.00075. Dropout layers with $p = 0.25$ and batch normalization layers have been added after every convolutional layer for better training accuracy.

5.2 Evaluation Metrics

We use the following performance metrics to evaluate our CNN-based classification model.

- TPR_{mal} is the percentage of malicious bitstreams that are correctly classified as malicious.
- FPR_{mal} is the percentage of benign bitstreams that are incorrectly classified as malicious.
- Classification accuracy (A_c) is the ratio of the number of correct predictions to the total number of predictions. It is computed as: $A_c = \frac{C_p}{T_p}$, where C_p is the

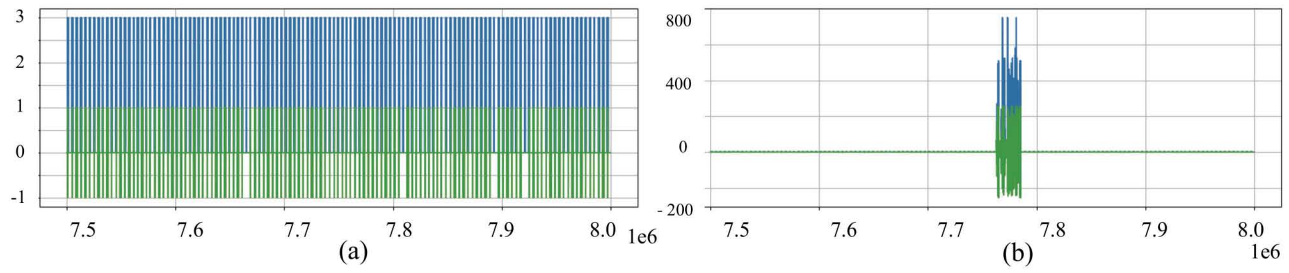


Figure 7: FFT of images corresponding to (a) Benign RO (b) Critical RO.

Table 2: Exploring $flip$ techniques.

Technique	Training acc. (%)	Test acc. (%)
$flip_{lr}$	93.9%	95.7%
$flip_{ud}$	99.2%	96.4%
$flip$ (along axis $(1, 2)$)	90.8%	87.4%

Table 3: Comparison of our model with the CNN architecture used for FPGA bitstream classification in [10].

Characteristics	CNN architecture in [10]	**Proposed method**
No. of conv. layers	One	Four
No. of FC layers	Two	Four
Training accuracy	91.6%	99.2%
Training loss	0.36	0.031
Test accuracy	85.7%	96.4%

number of correct predictions and T_p is the total number of predictions.

5.3 Evaluation of Proposed CNN-based Classification Framework

Our experimental dataset comprises of 95 image files generated from benign bitstreams and 80 image files generated from malicious bitstreams. Since the dataset is of relatively small size, we use image augmentation to increase the size of our training dataset. We choose the *rotation* and *flip* tools from the Scikit-learn library [13]; these image augmentation techniques show the highest classification accuracy compared to other commonly used image augmentation tools, such as the *equalize* and the *translate* operations [18]. The *rotation* and *flip* image augmentation techniques enable our model to extract meaningful features from the image representation of bitstreams. We performed several experiments to determine the best $flip$ operation among - $flip_{lr}$, $flip_{ud}$, and $flip$. These operations are described below:

(1) $flip_{lr}$: Flip the image horizontally, either left or right;
(2) $flip_{ud}$: Flip the image vertically, either up or down;
(3) $flip$: Flip the image along any axis or multiple axes.

The evaluation results are presented in Table 2. We observe that $flip_{ud}$ gives significantly better results when used as the image augmentation technique.

After image augmentation, our dataset contains 314 image files. We use k-fold cross validation to evaluate multiple versions of train-test split. In our experiments, we select $k = 5$ in accordance with common practice [14]. We obtain an average training accuracy of 99.2% and an average test accuracy of 96.4% after 300 epochs. Also, we obtain TPR_{mal} = 97.08% and FPR_{mal} = 4.29%.

Next, we compare our approach with prior work on classification of FPGA bitstreams. In [10], a CNN model is used to detect a particular hardware module (e.g. adder, subtractor, or multiplier). To highlight that our proposed CNN model is customized for the specific application of detecting malicious bitstreams, we apply the CNN architecture described in [10] to our dataset. Evaluation results in Table 3 show that our CNN architecture is carefully designed to be suitable for the security problem of malicious bitstream detection and authentication.

5.4 Timing Overhead

We next investigate the timing overhead of each stage of the CNN-based image classification framework. The conversion of a user-input bitstream to data series requires negligible time – less than 4 minutes of CPU time on a 2.4 GHZ Intel Xeon Gold 5115 CPU with 768 GB of RAM. We also observe that CNN inferencing requires only 0.03 seconds on a NVIDIA GeForce GTX 1080 GPU.

5.5 Evaluation on Multiple FPGA Families

We present evaluation results of our proposed CNN-based malicious bitstream detection framework for other FPGA versions. This procedure requires only retraining our model, while the CNN-based framework remains unchanged. We generated benign and malicious bitstreams for the Xilinx Kintex Ultrascale FPGA, and then prepared the training dataset for our model. After image augmentation, we obtain 84 images in the training dataset and 28 images in the test dataset. Using the CNN-based feature classification framework, we obtain a training accuracy of 98.4%. A high classification accuracy of 95.7% highlights the flexibility of utilizing our CNN model over multiple FPGA versions, without the need for reverse-engineering methods.

5.6 Analysis of RO Criticality

The test dataset includes the FFT-encoded images of 48 different bitstreams corresponding to benign and critical ROs. These are as follows:

(1) 15 bitstreams corresponding to power-wasting RO and conditional RO circuits (**Critical**);
(2) 6 bitstreams implementing variants of the latched RO circuit (**Critical**);
(3) 7 bitstreams corresponding to the self-clocked RO circuit (**Critical**);
(4) 20 bitstreams that implement N wrapper-based TRNGs, $250 < N < 300$ (Benign).

We perform FFT to obtain the spectral images of the test dataset. Next, we use our pre-trained neural network to classify the type of RO (i.e., benign or critical) based on the unique spectral signatures of each FFT-encoded images. The overall classification results are presented in Table 4. This experiment therefore highlights the importance of analyzing the criticality of any RO-based circuit that is configured on an FPGA and further using the features from the spectral domain to detect and block malicious Trojans.

Table 4: Classification accuracy of pre-trained CNN model on FFT-encoded benign and critical ROs.

Decision	Power-wasting RO	Cond. RO	Latched RO	Self-clocked RO	TRNG
Critical	9	6	4	6	1
Benign	0	0	2	1	19
A_c	100	100	66.67	85.7	95
Average A_c	**89.47**				

6 CONCLUSION

We have presented an efficient CNN-based malicious bitstream detection framework. By embedding image augmentation in our framework, we have demonstrated a high classification accuracy of the model. The proposed CNN model utilizes specific patterns from the data-series representation of malicious bitstreams and efficiently distinguishes between benign and malicious bitstreams with an accuracy of 96.4%. We have also presented a CNN-based solution using FFT techniques for criticality analysis of ROs that are configured on multi-tenant FPGAs.

7 ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under grant no. CNS-2011561.

REFERENCES

[1] Amazon. 2021. Amazon EC2 F1 Instance. https://go.aws/3ENtUj9.

[2] Ken Eguro and Ramarathnam Venkatesan. 2012. FPGAs for Trusted Cloud Computing. In *International Conference on Field-Programmable Logic and Applications* (international conference on field-programmable logic and applications ed.). IEEE. https://bit.ly/30BEUS0.

[3] Xavier Glorot et al. 2013. A semantic matching energy function for learning with multi-relational data. *Machine Learning* (2013).

[4] D Gnad et al. 2017. Voltage drop-based fault attacks on FPGAs using valid bitstreams. *Proc. FPL* (2017), 1–7.

[5] Dennis Gnad et al. 2018. Checking for Electrical Level Security Threats in Bitstreams for Multi-tenant FPGAs. In *FPT*. https://doi.org/10.1109/FPT.2018.00055.

[6] Dennis R. E. Gnad, Vincent Meyers, Nguyen Minh Dang, Falk Schellenberg, Amir Moradi, and Mehdi B. Tahoori. 2021. Stealthy Logic Misuse for Power Analysis Attacks in Multi-Tenant FPGAs. In *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1012–1015. https://doi.org/10.23919/DATE51398.2021.9473938

[7] Kento Hasegawa et al. 2016. Hardware Trojans classification for gate-level netlists based on machine learning. In *IOLTS*. https://doi.org/10.1109/IOLTS.2016.7604700

[8] J. Krautter et al. 2018. FPGAhammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018).

[9] Tuan Minh La, Kaspar Matas, Nikola Grunchevski, Khoa Dang Pham, and Dirk Koch. 2020. FPGADefender: Malicious Self-Oscillator Scanning for Xilinx UltraScale + FPGAs. *ACM Trans. Reconfigurable Technol. Syst.* 13, 3, Article 15 (sep 2020), 31 pages. https://doi.org/10.1145/3402937

[10] S. Mahmood et al. 2019. IP Core Identification in FPGA Configuration Files using Machine Learning Techniques. In *Proc. IEEE ICCE-Berlin*.

[11] Hassan Nassar, Hanna AlZughbi, Dennis R. E. Gnad, Lars Bauer, Mehdi B. Tahoori, and Jörg Henkel. 2021. LoopBreaker: Disabling Interconnects to Mitigate Voltage-Based Attacks in Multi-Tenant FPGAs. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. 1–9. https://doi.org/10.1109/ICCAD51958.2021.9643485

[12] F Schellenberg et al. 2021. An inside job: Remote power analysis attacks on FPGAs. *IEEE Design & Test* 38, 3 (2021), 58–66.

[13] Scikit-learn. [n. d.]. Machine learning in Python. https://bit.ly/3OzdLBZ. [Online; accessed 25-July-2022].

[14] SKlearn. [n. d.]. Grid Search with Cross Validation. https://bit.ly/3hEHNnQ. [Online; accessed 13-August-2020].

[15] T. Sugawara et al. 2019. Oscillator without a combinatorial loop and its threat to FPGA in data centre. *Electronics Letters* (2019). https://doi.org/10.1049/el.2019.0163

[16] Nidish Vashistha et al. 2018. Trojan scanner: Detecting hardware trojans with rapid sem imaging combined with image processing and machine learning. In *ISTFA*. https://doi.org/10.31399/asm.cp.istfa2018p0256

[17] Kenneth M. Zick et al. 2013. Sensing nanosecond-scale voltage attacks and natural transients in FPGAs. In *FPGA*.

[18] Barret Zoph et al. 2019. Learning Data Augmentation Strategies for Object Detection. *CoRR* (2019). http://arxiv.org/abs/1906.11172.