UC Davis
UC Davis Electronic Theses and Dissertations

Title
Fingerprinting SoC FPGA via Measuring Communication Link

Permalink
https://escholarship.org/uc/item/47g9737q

Author
Zhou, Jiacheng

Publication Date
2023

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Fingerprinting SoC FPGA via Measuring Communication Link

By

JIACHENG ZHOU

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCES

in

Electrical and Computer Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Houman Homayoun, Chair

Avesta Sasan

Setareh Rafatirad

Committee in Charge

2023

Table of Contents

ABSTRACT IV

ACKNOWLEDGMENTS V

1 INTRODUCTION 1

1.1 FPGA Hardware Security 1

1.2 Why Side-Channel Attack? 2

1.3 Contributions 3

1.4 Content Overview 3

2 BACKGROUND 5

2.1 Edge SoC-FPGAs 5

2.2 DE1-SoC and Communication Link 6

2.2.1 DE1-SoC 6

2.2.2 Communication Link 7

2.3 Side-Channel Attack 7

2.3.1 Long Wire Side-Channel Attack 8

2.3.2 Power Side-Channel Attack 9

2.3.3 PCIe Side-Channel Attack 9

2.4 Related Works 10

3 METHOD AND IMPLEMENTATION 12

3.1 Testing Environment 12

3.2 Benchmark Design 13

3.2.1 Host Program 13

3.2.2 Benchmark Accelerator 14

3.3 Machine Learning Modules 15

4 EVALUATION 16

4.1 Experiment Setting 16

4.1.1 Hardware Environment 16

4.1.2 Dataset Collection 16

4.1.3 Victim Circuits 16

4.1.4 Machine Learning Classifier Setting 17

4.2 Results 18

5 DISCUSSION 25

5.1 Compare with Previous Work 25
5.2 Parameter Choice 28
5.3 Mitigations 29
5.4 Limitations and Future Work 30
6 CONCLUSION 32
REFERENCES 34

ABSTRACT

Fingerprinting SoC FPGA via Measuring Communication Link

Over the past few decades, Field Programmable Gate Arrays (FPGAs) have found widespread application in various fields, including signal processing, hardware acceleration, machine learning, etc. To address the requirements of heterogeneous computing, a new category of devices called System-on-Chip FPGAs (SoC-FPGAs) has emerged. These devices combine the capabilities of microcontrollers and FPGAs to leverage the advantages offered by both general-purpose CPUs and FPGAs. However, the introduction of SoC-FPGAs also introduces potential security vulnerabilities since both CPUs and FPGAs are impressionable to side-channel attacks. On SoC-FPGAs, the CPUs and FPGAs are connected through the communication link, which can be utilized as an attacking interface for such side-channel attacks. In this paper, we proposed such side-channel attacks targeting SoC-FPGAs to fingerprint the hardware accelerators running on them by measuring the communication link. We implemented a benchmark program that stresses the communication link and records the I/O bandwidth of the running accelerators. After analyzing the collected data traces using machine learning classifiers, we successfully distinguished the unique I/O pattern of different victim accelerators. Our highest classification accuracy result is 93.3% with a random forest classifier, and it proved that attackers could use such an attack to perform accelerator fingerprinting on SoC-FPGAs. As far as we know, this is the first attack by which fingerprinting accelerator on SoC-FPGAs via measuring communication link.

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to Professor Houman Homayoun, my committee chair, for his invaluable guidance and unwavering support throughout my study. His assistance has been instrumental in my successful completion of this study.

I would also like to thank my thesis committee members, Professor Avesta Sasan and Professor Setareh Rafatirad, as they provided advice for me to strengthen my thesis.

Thirdly, I would like to thank Chongzhou Fang, Ning Miao, and Jiawei Liu as my collaborators. They provide valuable feedback and guidance throughout my course of study. They also did a fantastic job of conducting this research.

Lastly, I would like to thank my family and friends, who always give support when I am going through tough times.

Chapter 1

Introduction

1.1 FPGA Hardware Security

Field Programmable Gate Arrays (FPGAs) are a type of integrated circuit that allows programming by the customer to have the desired function or circuit. With its flexibility, FPGAs can be configured to perform various computations, including signal processing, cryptography, and machine learning. Therefore, FPGAs become a popular choice in a wide range of applications, from aerospace to consumer electronics. However, with great power comes great security threat. Hardware attacks on FPGAs can take various forms, such as tampering with the configuration memory, probing the input and output pins, or injecting faults to alter the circuits' behavior. Such attacks can lead to critical security breaches, including data theft, system malfunction, and unauthorized access.

Ensuring FPGA hardware security requires a multi-layered approach that involves the design, implementation, and verification of secure FPGA circuits. At the design stage, security requirements must be identified and integrated into the system design, including secure booting, cryptographic key management, and access control. The implementation stage involves the selection of secure FPGA components and the integration of security mechanisms, such as antitamper features, into the FPGA design. Verification ensures that the FPGA circuit meets the security requirements and is resilient to hardware attacks. To enhance FPGA hardware security, various techniques have been developed, such as obfuscation, encryption, and authentication. Obfuscation involves modifying the FPGA design to make it harder to reverse-engineer and understand. Encryption consists in encrypting the FPGA configuration bitstream to prevent unauthorized access or modification. Authentication involves verifying the authenticity of the FPGA configuration bitstream and ensuring that only authorized users can program the FPGA.

Besides the manufacturing security of FPGAs, recent research works showed that FPGAs are vulnerable to various types of security attacks. For example, bitstream fault injection [1], hardware trojan [2], rowhammer attacks [3], power side-channel attacks [4], etc. In this research, our goal is to prove that SoC-FPGAs are vulnerable to side-channel attacks by measuring the communication link.

1.2 Why Side-Channel Attack?

Side-channel attacks are a type of hardware attack that can reveal information leakage by measuring the system's physical characteristics, including power consumption, electromagnetic emissions, and timing variation. Side-channel attacks on FPGAs are particularly dangerous because they can provide the attacker with access to sensitive information, such as cryptographic keys, stored in the FPGA's configuration memory or the user's design. There are several types of side-channel attacks that target FPGAs, such as power analysis side-channel attacks, electromagnetic analysis side-channel attacks, glitch attacks, timing analysis side-channel attacks, etc. Power analysis side-channel attacks extract sensitive information by measuring the power consumption of the FPGA during its operation. Electromagnetic analysis side-channel attacks extract sensitive information by monitoring the electromagnetic radiation emitted by the FPGA during its operation. Glitch attacks introduce controlled faults into the FPGA to cause it to malfunction. Finally, timing analysis side-channel attacks rely on the variations in the FPGA's timing caused by the input data. Several mitigations have been developed to prevent the FPGAs from side-channel attacks, such as masking, noise injection, and circuit-level countermeasures. However, with the increasing use of FPGAs in critical systems, such as aerospace, defense, banking, and cloud computing, exploring new types of side-channel attacks and developing mitigations against them is essential.

1.3 Contributions

Thus, our approach is to fingerprint edge SoC-FPGAs via the communication link. To achieve this, our method is first to design a benchmark to stress the communication link and conduct read/write to host memory blocks in the SoC-FPGA (DE1-SoC board in this work). Then we collect traces of the bandwidth of our benchmark circuit when different kernels are running on the same FPGA from AXI. The details of how we build our dataset will be covered in the corresponding section. After collecting the traces, we feed them to the machine learning model to classify the victim circuit. The results of different machine learning classifiers will be discussed at the end. To the best of my knowledge, this is a new type of side-channel attack targeting SoC-FPGAs.

1.4 Content Overview

Chapter 2 covers the background information about what SoC FPGAs are, what side-channel attacks are, and introduces some other techniques that have been studied in FPGA security. Chapter 3 covers the experiment environment, the design and implementation of our benchmark program, and the machine learning models we used for trace classification. Chapter 4 contains the evaluations of the experiment results. Chapter 5 will discuss the experiment result compared to our previous work [22], the parameters we have chosen to use for the benchmark programs, possible mitigations to our proposed attack, limitations, and future work. Finally, Chapter 6 concludes the thesis.

Chapter 2
Background

2.1 Edge SoC-FPGAs

An edge system on-chip (SoC) is a type of integrated circuit that consists of a central processing unit, memory, input/output interface, and other processing components on a single chip. It is designed to process data or make decisions at the edge of a network. Field programmable gate arrays (FPGAs) are integrated circuits that can be programmed after being manufactured. With its reprogramming function, it is often used as the hardware accelerator for custom applications, such as machine learning. Thus, an edge SoC-FPGA is a type of integrated circuit that combines the functionalities of edge SoC and FPGA. One of the critical advantages of edge SoC-FPGAs is their flexibility. With the programmable logic, not only a wide range of applications are able to be deployed onto the chip, but it also makes high levels of customization possible. On the other hand, an edge SoC-FPGA contains processing units such as CPUs, memory, and I/O interfaces. This allows it to perform processing tasks on the chip rather than sending the data to a centralized data center for processing, which reduces latency, improves security, and reduces the amount of data that needs to be transmitted over the network. The different types of I/O interfaces, including Ethernet, USB, PCIe, and serial interfaces, allow the device to communicate with other components in the system. The I/O interfaces also make integrating with existing systems and devices easy, providing a seamless integration process. Another advantage of edge SoC-FPGAs is they integrate multiple functions onto a single chip. This integration reduces the number of components required in a system, which can reduce the power consumption, and system cost and improves the system reliability.

Also, with the CPU on board, developers can run operating systems such as Linux on the chip and write software using familiar programming languages such as C and Python. The on-chip memory can be used for program and data storage and can be accessed at high speed, which gives it the ability to process the data in real time. They are commonly used in a variety of applications, such as industrial automation, automotive, and robotics.

In conclusion, edge SoC-FPGAs are integrated circuits that combine CPU, memory, I/O interfaces, and programmable logic onto a single chip. They are designed to be flexible, low-power, and cost-effective. In addition, they can provide hardware acceleration for a wide range of applications, including machine learning, image processing, and edge computing. Thus, makes them the ideal choice for the systems that require real-time processing for high-speed data.

2.2 DE1-SoC and Communication Link

2.2.1 DE1-SoC

In this research, we choose the DE1-SoC Development board as our targeting platform. DE1-SoC is an Altera SoC-FPGA developed by Terasic Technologies Inc, which combines a dual-core ARM Cortex-A9 processor with FPGA fabric. It provides many features for users to conduct a wide range of projects, including Cyclone V SoC FPGA with 85k programmable logic elements (LEs), 1GB DDR3 SDRAM and 2GB HPS DDR3, ethernet, USB, HDMI, and VGA ports for connectivity, onboard sensors, including temperature, light, and accelerometer sensors, etc. As a result, it is a popular choice for academic and research purposes and for device prototyping.

2.2.2 Communication Link

On DE1-SoC, the HPS portion is connected to the FPGA portion through the HPS-FPGA interfaces, which are the Advanced eXtensible Interface (AXI) bridges. There are mainly three bridges used for communication, HPS-to-FPGA bridge, FPGA-to-HPS bridge, and Lightweight HPS-to-FPGA bridge. Both the HPS-to-FPGA bridge and FPGA-to-HPS bridge are high-performance AXI interfaces that have a configurable data width of 32, 64, and 128 bits. They allow the HPS or FPGA to master transactions to slaves to the FPGA or HPS fabric and vice versa. Furthermore, the FPGA-to-HPS bridge enables the FPGA to fabric to have complete visibility into the HPS address space and provides access to the coherent memory interface. The Lightweight HPS-to-FPGA bridge is primarily used for control and status register accesses to peripherals in the FPGA fabric and only supports a fixed data width of 32 bits for the HPS to master transactions to slaves in the FPGA fabric. The connections between the FPGA fabric and the L3 interconnect part of the HPS are shown in Figure 2.1. The letters M and S indicate master and slave. The level 3 (L3) main switch masters the HPS-to-FPGA bridge, and the L3 slave peripheral switch masters the lightweight HPS-to-FPGA bridge. The L3 main switch is mastered by the FPGA-to-HPS bridge, which allows the master in the FPGA to have access to most of the slaves in the HPS.

2.3 Side-Channel Attack

As mentioned in section 1.2, side-channel attacks are a type of attack that measures the physical characteristics to reveal sensitive information about the system. Once the attacker breaches the FPGAs, it can lead to unwanted data leakage and harm the customer’s profit. In

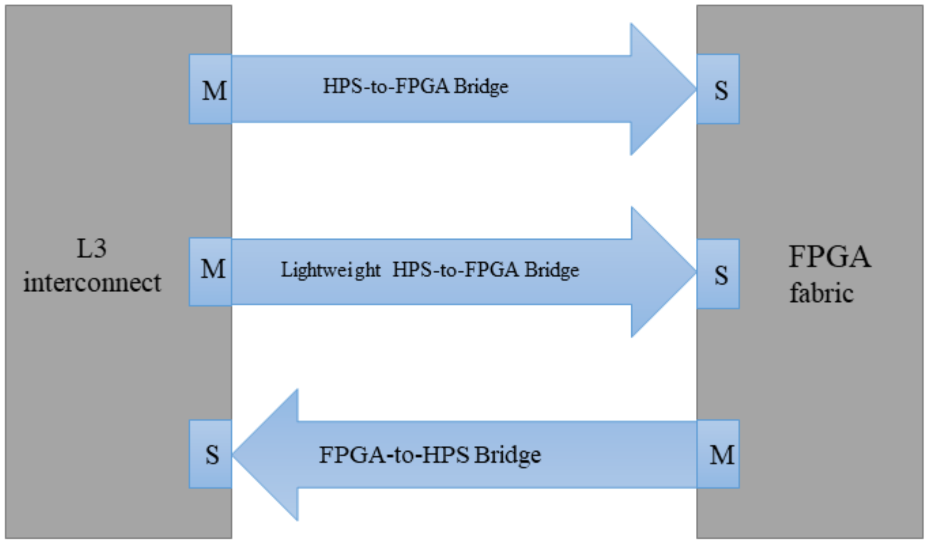


Figure 2.1 AXI Bridge Connectivity (adapted from Altera, 2014)

recent research works, FPGAs have been proven that are vulnerable to side-channel attacks. In literature, the most extensively researched types of side-channel attacks are listed below.

2.3.1 Long Wire Side-Channel Attack

In the previous research work [5], the authors found that on FPGAs, there is a routing resource that is used to connect configurable logic blocks (CLBs), called long wire, which can be used as a source for information leakage. Long wire has the following physical characteristics when transferring data: the delay of the nearby long wires is shorter when transferring a logical 0 than a logical 1. Based on this phenomenon, the authors monitor the data transfers in the target long wire by connecting the ring oscillator (RO) to it. The frequency of the ring oscillator will increase when the target long wire transfers a logical 1, and it will decrease when the target long wire transferring a logical 0. By using this long wire side-channel attack, they successfully recovered 99% of the bits that were being transmitted in the target long wire. Similarly, [6] recovered the secret key of an AES implementation using the long wire side-channel attack.

2.3.2 Power Side-Channel Attack

Power side-channel attacks usually are achieved by monitoring the power consumption of the FPGAs. Since the power consumption of the FPGAs is closely tied to the data being processed or transmitted, attackers can use the power consumption of the target FPGA to recover secret information such as the cryptographic key. Usually, power side-channel attacks require physical access to the target FPGAs, but [4] proposed a remote power side-channel attack by designing an RO-based on-chip power monitor on the FPGA to monitor the RSA crypto module which is working on the same FPGA in the cloud environment. Furthermore, [7] proposed a new design of the RO-based on-chip power monitor, which can measure the internal voltage on a nanosecond scale. Their work is able to retrieve the secret of an AES encryption circuit using the power side-channel attack. Also, the power side-channel can be used for accelerator fingerprinting, as shown in [8].

2.3.3 PCIe Side-Channel Attack

In [9], the authors used PCIe congestion side-channel to recover the secret information of the victim machine in 3 scenarios, keystroke typing, webpage browsing, and training machine-learning model. Their attack is achieved by sending a delay to the PCIe link to cause it to be in the congestion stage and observe the I/O pattern of the device. However, [9] focuses on revealing secret information about GPUs.

In [10], the author proposed a side-channel attack using PCIe contention on the AWS cloud server. The authors note that where PCIe slots are situated in the PCIe layout can cause varying levels of delay and data transfer rates. As a result of this, they can recognize alterations in data transfer rates when different FPGAs on the same server try to access memory simultaneously, leading to PCIe conflict, and they can deduce the specific location of various FPGAs in the AWS server through this method. Although [10] also targets FPGAs, their work aims to reveal the infrastructure information. Unlike our work, we focus on the information about the applications running on the FPGAs.

2.4 Related Works

Recently, several types of side-channel attacks have been studied. Long wire side-channel attack is one major type, where attackers use long wire to extract secret information on the FPGAs. [5] found the physical characteristics of the long wire, which can cause different delays to the nearby long wire when transferring a logical 1 and logical 0. The authors use ROs to monitor the frequency change to determine whether the bits transmitted in the target long wire are a logical 1 or 0. Similarly, [6] performs the long wire side-channel attack to recover the secret key of an AES implementation. In [4], the authors proposed another type of side-channel attack targeting FPGAs, which is the power side-channel attack. The authors designed an RO-based on-chip power monitor to observe the power consumption of the FPGA in a fixed time interval since different processing stages consume different power. This attack successfully recovers the secret key of an RSA crypto module and also shows that it is possible to perform an FPGA-to-CPU attack on the same SoC.

Our work focused on fingerprinting FPGA accelerators using communication links. [9] proved that such side-channel attacks using the PCIe communication link can reveal secret information about the victim device. They stress the PCIe link by sending data through Remote Direct Memory Access (RDMA) NICs and observing the I/O pattern of the device. Their attack captured the keystroke timing, web rendering, and the machine learning modules that are running on the victim device. [10] conduct a similar attack that stresses the PCIe link to generate PCIe contention. By using this attack, they reveal the infrastructure information on the AWS cloud.

Although both attacks use PCIe communication links to extract secret information, [9] targets GPUs, and [10] focuses on the infrastructure information.

In our previous research [22], we implemented such side-channel attacks targeting Intel’s cloud service, Devcloud [23]. We stressed the PCIe communication link using benchmark accelerators and measured the bandwidth of communication. The data collected is classified using machine learning models. As a result, our proposed attack successfully classified the six different accelerators deployed on cloud FPGAs with a random forest classifier’s accuracy at 88%. We also looked into how different parameters used for our benchmark programs will affect the classification accuracy results and found the optimal set of parameters. The work in this paper is transplanted from our previous work [22], which targets the SoC-FPGAs using the same attack method. The optimal set of parameters for the benchmark programs is used in this work.

Chapter 3

Method and Implementation

3.1 Testing Environment

A DE1-SoC Development board was chosen to be our targeting platform. According to [11], our DE1-SoC has a Dual-core ARM Cortex-A9 MPCore processor as CPU, Altera Cyclone V FPGA, 64MB SDRAM on the FPGA fabric, 1GB DDR3 SDRAM and Micro SD card socket for storage. The detailed device specification is shown in Figure 3.1.

In this experiment, our goal is to measure the bandwidth of the communication link when running different benchmark kernels and then feed the collected data to machine learning classifiers to fingerprint the victim circuit. The experiment procedure can be divided into the following stages:

1) Design and implement benchmark kernels to stress the AXI communication link between the HPS and FPGA fabric.
2) Collect corresponding I/O measurement data of the victim circuit and divide the data into training sets and testing sets.
3) Feed the training set to the machine learning classifiers and then use the testing set to obtain fingerprinting results.

HPC	FPGA
Dual-core ARM Cortex-A9 MPCore processor	Altera Cyclone V
1GB DDR3 SDRAM	85K programming logic elements
Micro SD card socker	64MB SDRAM
UART to USB, ISB Mini-B connector	10 red user LEDs
1GB Ethernet PHY with RJ45 connector	Six 7-segment display

Figure 3.1 Device Specification of DE1-SoC

3.2 Benchmark Design

In this section, we will discuss the specific implementation details of the benchmark that was used to stress the AXI communication link and monitor the I/O bandwidth. The benchmark's analysis helps to understand the I/O patterns of victim circuits, which can then be utilized to determine the type of victim employed for our fingerprinting attack on the SoC FPGA. Our benchmark program is developed using OpenCL [12]. The benchmark comprises two parts, the master host program executed by the CPU and the kernel executed by the FPGA. The implementation details of the program will be described below.

3.2.1 Host Program

Our host program will first allocate NUM_BUFFER memory blocks of size BUFFER_SIZE in the 1GB SDRAM of the HPC. The memory blocks will be accessed by the benchmark kernel using read and write operations. While the benchmark accelerator is running, our host program will record the time of the execution using the profiling APIs provided by OpenCL. To eliminate the impact of any noise, the operations of a benchmark kernel will be repeated and then averaged for NUM_REPEAT times for each memory block. Lastly, the host program will return a trace which is formed by combining all the bandwidth measurements. Algorithm 1 shows the pseudocode of our host program.

Algorithm 1: Host Program

```
Input:  number of buffers NUM_BUFFER, trace length, BUFFER_SIZE,
        number of executions, NUM_REPEAT
Output: trace
Initialization: assign 1 to the variable i and j, allocate buffer [BUFFER_SIZE] [NUM_BUFFER]
while ( i < NUM_BUFFER ) do
    exe_time = 0
        While iterations ≤ NUM_REPEAT do
            Run benchmark accelerator and operate on buffer[i];
            exe_time += time of execution;
        end
    exe_time /= NUM_REPEAT;
    trace [i] = 1 / exe_time;
end
```

Algorithm 2: Benchmark Kernel

```
Input:  Pointer to pre-allocated host memory *dst, access number NUM_ACCESS
Initialization: id = work-item's ID number
while ( i < NUM_ACCESS ) do
    dst [id] = ( int ) ( i*dst[id] );
end
```

3.2.2 Benchmark Accelerator

The main functionality of our benchmark accelerator is to stress the AXI communication link by massive reading and writing to the designated memory destination. To ensure that our benchmark accelerator is accessing the designated memory destination, we implement an address pointer `dst` pointing to the pre-allocated host memory. This configuration also ensures that our benchmark accelerator is accessing the host memory through the AXI communication link. We use NUM_ACCESS to control how many times the memory location is accessed by the benchmark accelerator. Algorithm 2 shows the operation of our benchmark accelerator.

3.3 Machine Learning Modules

In our experiment, the traces we collect are fixed-length, 1-D vectors, which are determined by our benchmark program. Machine learning modules were employed to evaluate the relationship between bandwidth leakage and its corresponding benchmark accelerator. The following machine learning modules were used in our experiment: 1D-Convolution [13], Multi-layer Perceptron (MLP) [14], Support Vector Machine (SVM) [15], and Random Forest [16].

Chapter 4

Evaluation

4.1 Experiment Setting

4.1.1 Hardware Environment

The system OS we used on DE1-SoC is obtained from Terasic Inc [17], which is an Ubuntu mirror with OpenCL support. The OpenCL kernel files are compiled on the PC since it requires significant computational resources. Then we copy the bitstream files to the board. To ensure optimal OpenCL compatibility, the host files are compiled directly on the board.

4.1.2 Dataset Collection

During the data collection phase, each benchmark accelerator is repeatedly executed to ensure we have enough traces for the classification part. By initiating each execution with new arguments, we can generate an arbitrary number of trace vectors based on the specified loop number. This allows us to conduct the benchmark and victim kernel operations and obtain diverse trace vectors.

4.1.3 Victim Circuits

In our experiment, we download and modify the FPGA-accelerated workloads from the Xilinx Vitis Accelerator Example repository [18] and [19]. There are six workloads used in our experiment: `adder`, `apply_watermark`, `fir`, `matmul`, `nw`, and `pathfinder`. The detailed description of each workload is shown in Table I.

TABLE I: Descriptions of our Victim accelerators.

Name	Code	Function
`adder`	A	Adder. Reading inputs from an input buffer, performing computations, and writing the results to an output buffer.
`apply_watermark`	AW	Image processing. Reading images from an input buffer, adding watermark andhiheh hack to an output buffer.
`fir`	F	Signal processing. Reading input and coefficient data from an input buffer.
`matmul`	M	Matrix multiplication. Reading two matrices, A and B, from the input buffer, calculate AB, writing the result to an output buffer.
`nw`	NW	Generating two sequences randomly with the same length, which can be divided by 16.
`pathfinder`	P	Finding a path on a 2-D grid from the bottom row to the top row with the smallest accumulated weights.

TABLE II: Configuration details of our machine learning classifiers.

Name	Configuration
1D-Convolute	Build using Pytorch, cross entropy loss, and stochastic gradient descent (SGD) learning rate = 0.001, trained for 1500 epochs.
MLP	Build using Pytorch, cross entropy loss, and Adam optimizer, learning rate = 0.001, trained for 1500 epochs
Random-forest	RandomForestClassifier ( ). Obtained from Scikit-learning library, depth = 32768
SVM	SVC ( ). Obtained from Scikit-learning library

4.1.4 Machine Learning Classifier Setting

The four machine learning classifiers used in our experiment are built using Python machine learning libraries such as Pytorch [20] and Scikit-learn [21]. Table II shows the configuration details of our classifiers. We split the collected dataset into two groups, 70% of the dataset will be used to train the classifier, and the remaining 30% will be used in the testing stage.

4.2 Results

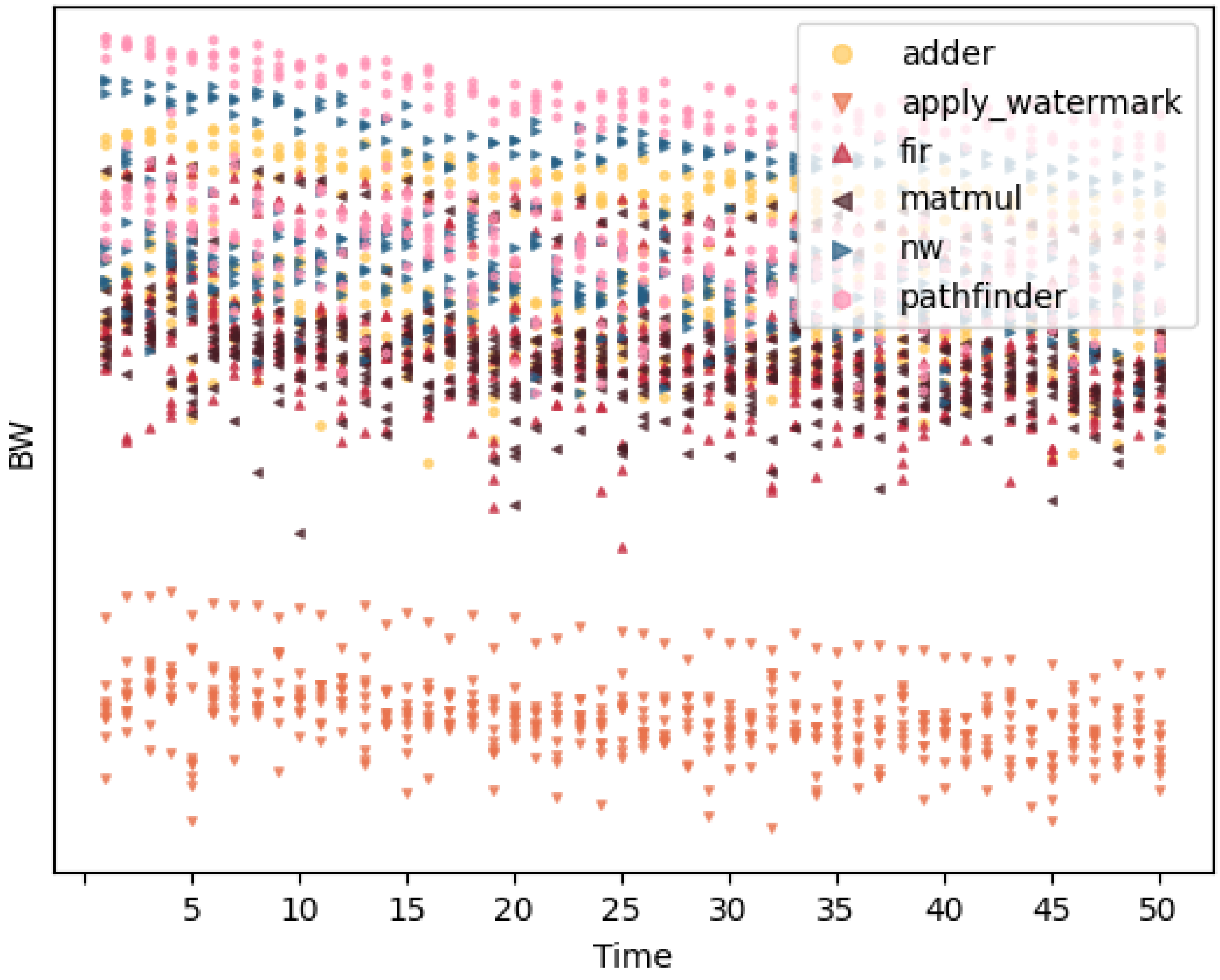


Figure 4.1 Data collected under the default setting.

The collected data traces for our six accelerator workloads are shown in Figure 4.1. The parameters we used to collect data traces in this experiment are listed below:

- NUM_ACCESS = 1000,
- NUM_REAPET = 10,
- BUFFER_SIZE = 4 Bytes,
- NUM_BUFFER = 100.

From Figure 4.1, we can see that the traces of each accelerator workload lie in different bandwidth regions. This indicates that the traces of our accelerator workloads can be separated and also prove that our benchmark kernels can capture the unique I/O patterns leaked from the communication link (AXI communication link in this experiment) during the execution of the accelerator workloads.

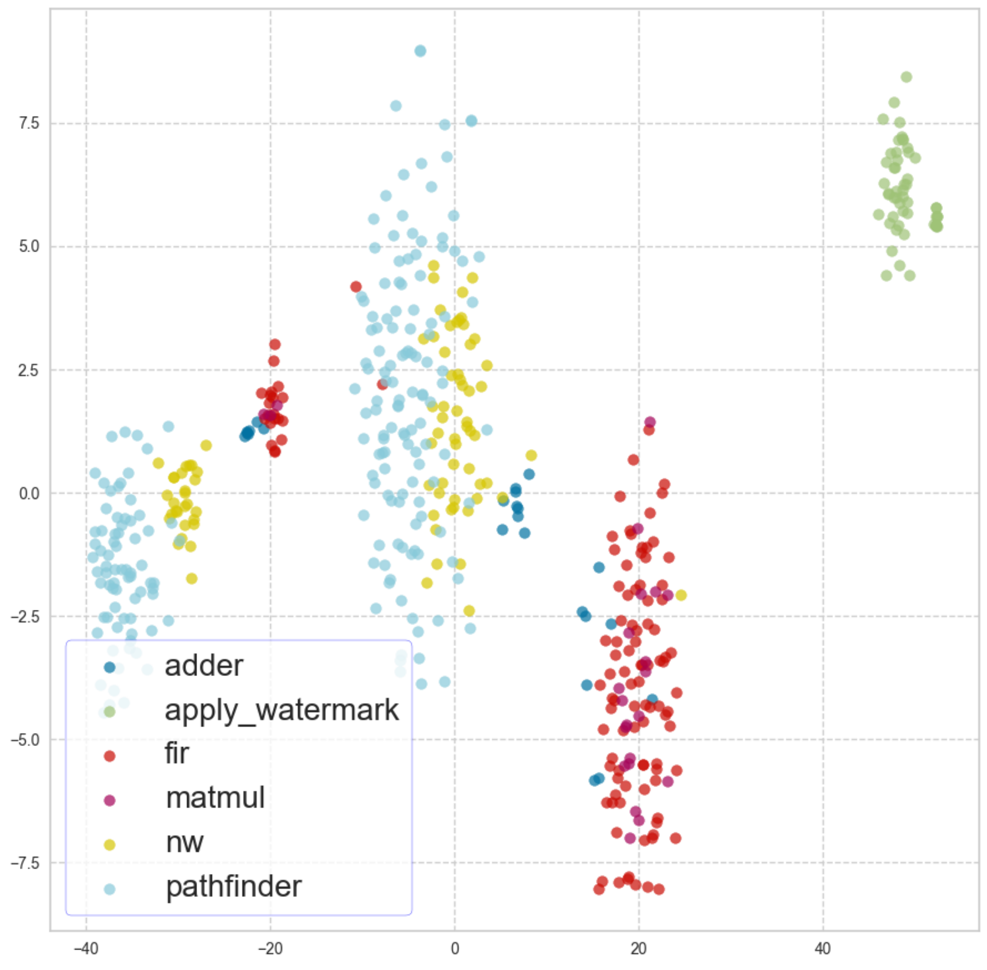


Figure 4.2 Corresponding t-SNE visualization

TABLE III: Accuracy results.

Model	Accuracy
1D-Convolution	66.7%
MLP	68.7%
SVM	74.7%
Random forest	93.3%

The collected data traces are then normalized, with the maximum bandwidth value in the dataset being 1 and the minimum bandwidth value being 0. The pre-processed dataset is then visualized using t-Distributed stochastic neighbor embedding (t-SNE) [20]. T-SNE stands as a highly popular technique for visualizing high-dimensional data. It assigns a high probability to similar objects and a low probability to dissimilar object. Then the objects are grouped with the ones that have similar probability and placed on a 2D map. The result of our data visualization is displayed in Figure 4.2. From Figure 4.2, we can see that the data points of each accelerator workload are placed in different regions on the map, which can also prove that our traces are distinguishable.

Table III shows the classification accuracy result of the four machine learning classifiers used in this experiment. The random forest has the highest accuracy result at 93.3%. SVM has an acceptable accuracy of 74.7%. To our surprise, the 1D-Convolution model, despite its complexity, exhibits the poorest classification accuracy performance.

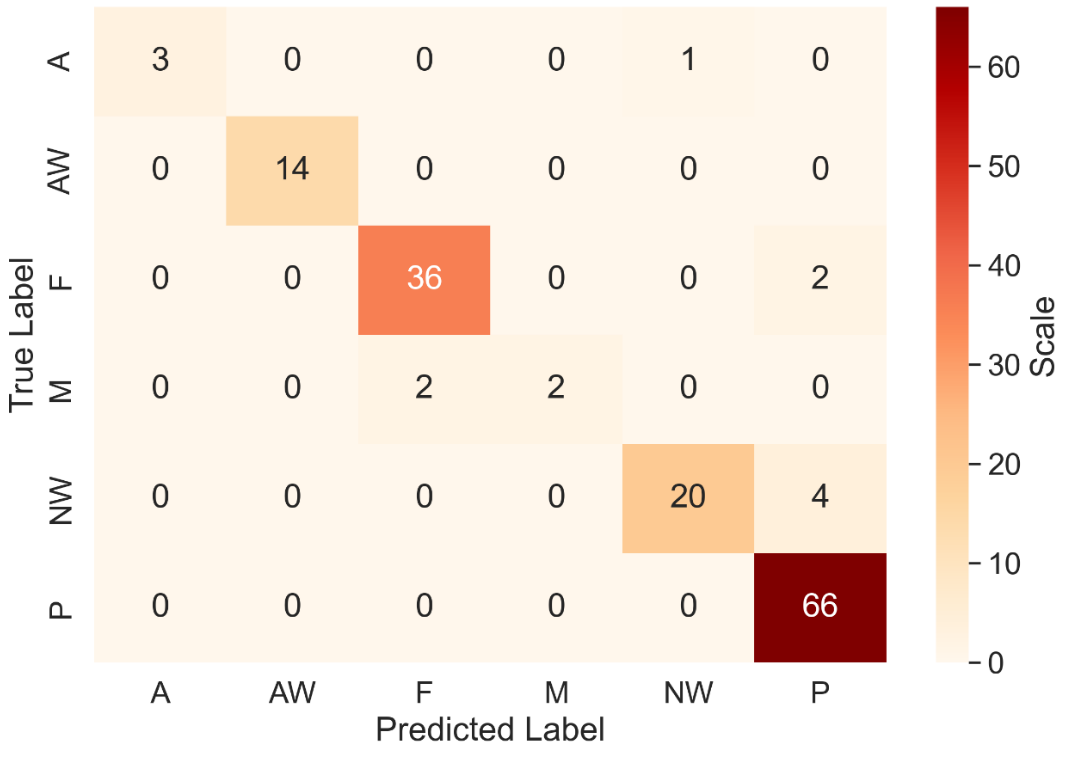
Since random forest and SVM are the classifiers that have higher accuracy results, we provide the metrics details of random forest and SVM in Table IV. These metrics serve as evaluation measures to assess the quality of classification results generated by a classifier, which include:

- Accuracy: represents how many items are classified correctly in a dataset. It is widely considered the most intuitive way to assess the classifier's performance.
- Precision score: represents how many items are classified correctly out of all correctly classified items.
- Recall score: represents how many items are classified correctly out of all the correctly labeled items.
- F-1 score [21]: represents how well the classifier is performing in terms of precision score and recall score. In this measurement, the precision score and the recall score have the same weight.

TABLE IV: Metrics of our random forest and SVM classifiers

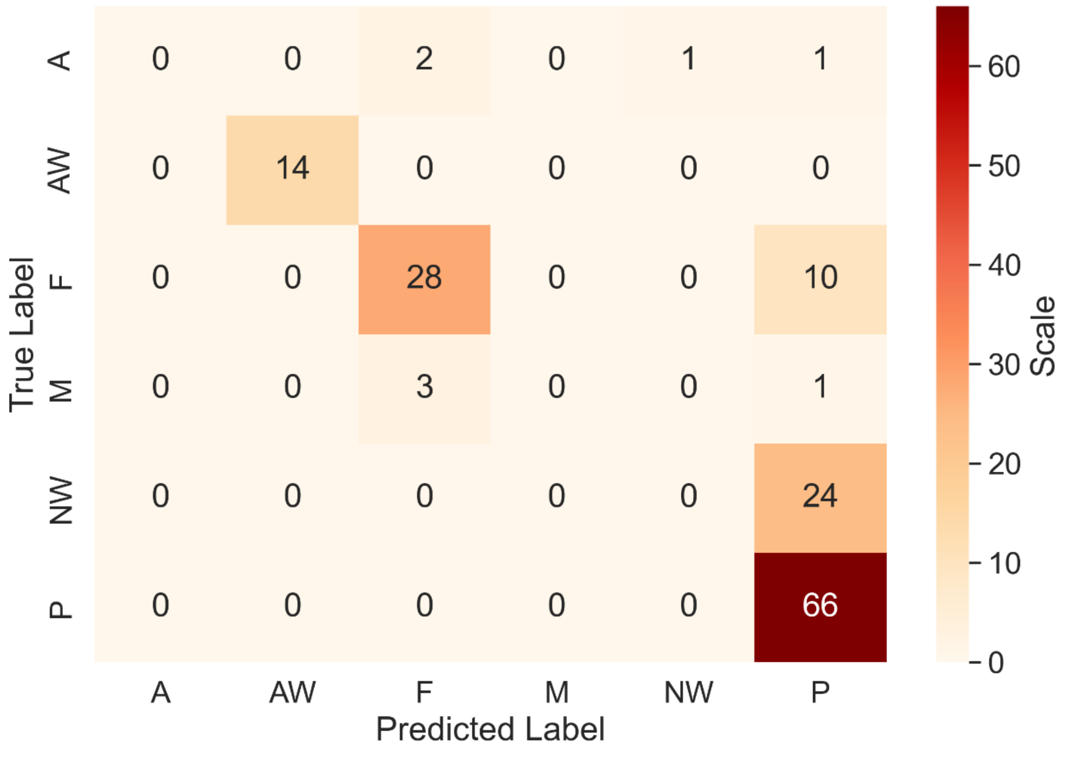
Model	Accuracy	Class	Precision	Recall	F-1
SVM	74.7%	adder	0.00	0.00	0.00
		apply_watermark	1.00	1.00	1.00
		fir	0.72	0.82	0.77
		matmul	0.00	0.00	0.00
		nw	0.00	0.00	0.00
		pathfinder	0.71	1.00	0.83
Random forest	93.3%	adder	1.00	0.86	0.92
		apply_watermark	1.00	1.00	1.00
		fir	0.88	1.00	0.94
		matmul	1.00	0.33	0.50
		nw	1.00	0.75	0.86
		pathfinder	0.92	1.00	0.96

We also include the confusion matrixes of our random forest and SVM classifier in Figure 4.3 and Figure 4.4. The confusion matrixes illustrate the accuracy of the random forest and SVM classifiers used in this experiment in classifying each of the accelerator workloads. The values within each cell of the confusion matrix indicate the number of traces corresponding to the predicted label and the true label. Upon observing Figure 4.3 and Figure 4.4, we can infer that the random forest has a better performance, and the SVM did a poor job at predicting 3 of the accelerator workloads, which are `adder`, `matmul`, and `nw`.



True \ Predicted	A	AW	F	M	NW	P
A	3	0	0	0	1	0
AW	0	14	0	0	0	0
F	0	0	36	0	0	2
M	0	0	2	2	0	0
NW	0	0	0	0	20	4
P	0	0	0	0	0	66

Figure 4.3 Confusion matrix of our random forest classifier.



True \ Predicted	A	AW	F	M	NW	P
A	0	0	2	0	1	1
AW	0	14	0	0	0	0
F	0	0	28	0	0	10
M	0	0	3	0	0	1
NW	0	0	0	0	0	24
P	0	0	0	0	0	66

Figure 4.4 Confusion matrix of our SVM classifier.

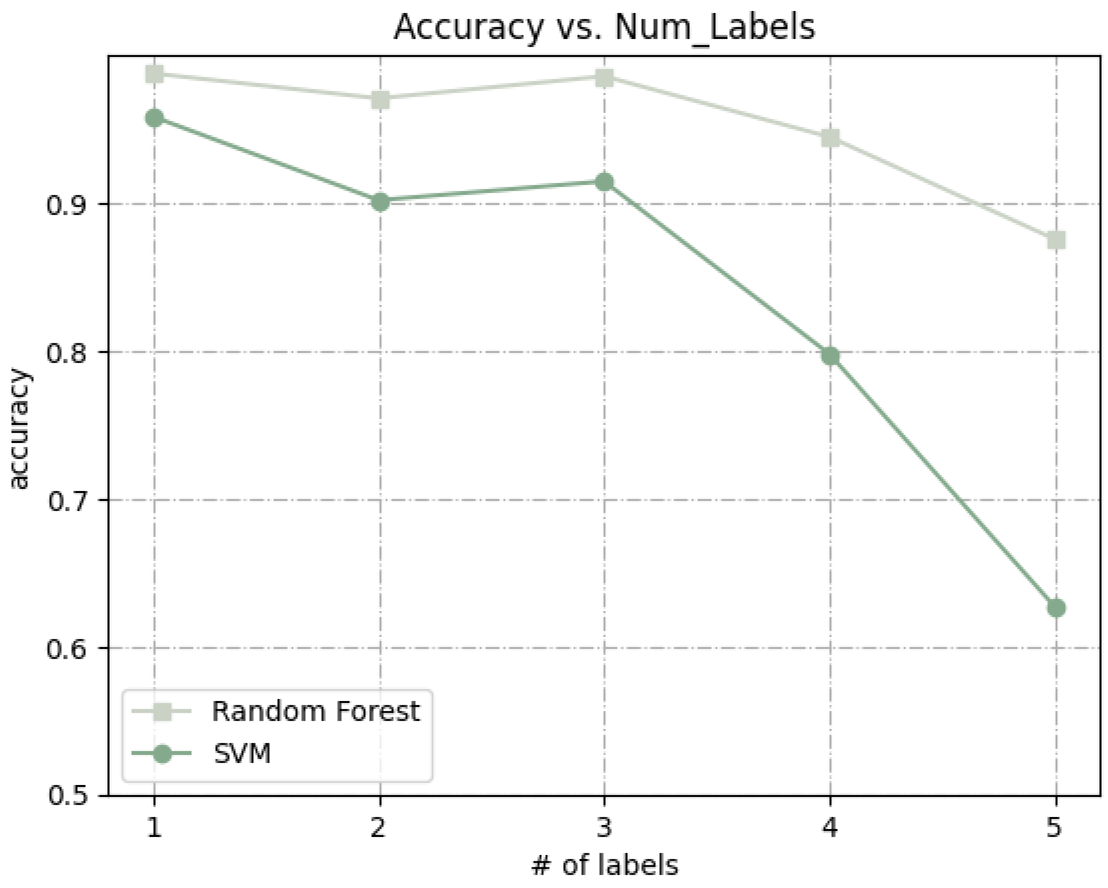


Figure 4.5 Accuracy results with the different number of labels used.

In this experiment, we also verify the capability of our random forest and SVM classifiers to distinguish between target accelerator traces and unwanted data. To do so, we select target accelerators from Table I and label the rest of the data as "unwanted," for example, if we select `adder` and `nw` as our target accelerators, the traces collected from `apply_watermark`, `fir`, `matmul`, and `pathfinder` will be labeled as "unwanted." Since there could have multiple combinations of the target accelerators when selecting one to five target accelerators, we experiment with all possible combinations and then use the average accuracy as our final result. On the other hand, the number of labeled traces is modified to be equal to remove the situation, such as the unwanted labeled data traces outnumber the target accelerator traces, causing the accuracy results to become inaccurate. Our result is shown in Figure 4.5.

By examining Figure 4.5, it becomes evident that our classifiers demonstrate they can distinguish between the target accelerator traces and unwanted data traces. When the number of target accelerators is equal to one to three, both classifiers maintain high accuracy results which are above 90%. Starting from four target accelerators, both classifiers' accuracy is dropping. However, our random forest still maintains an accuracy of around 90%, and the SVM's accuracy dropped tremendously.

Chapter 5

Discussion

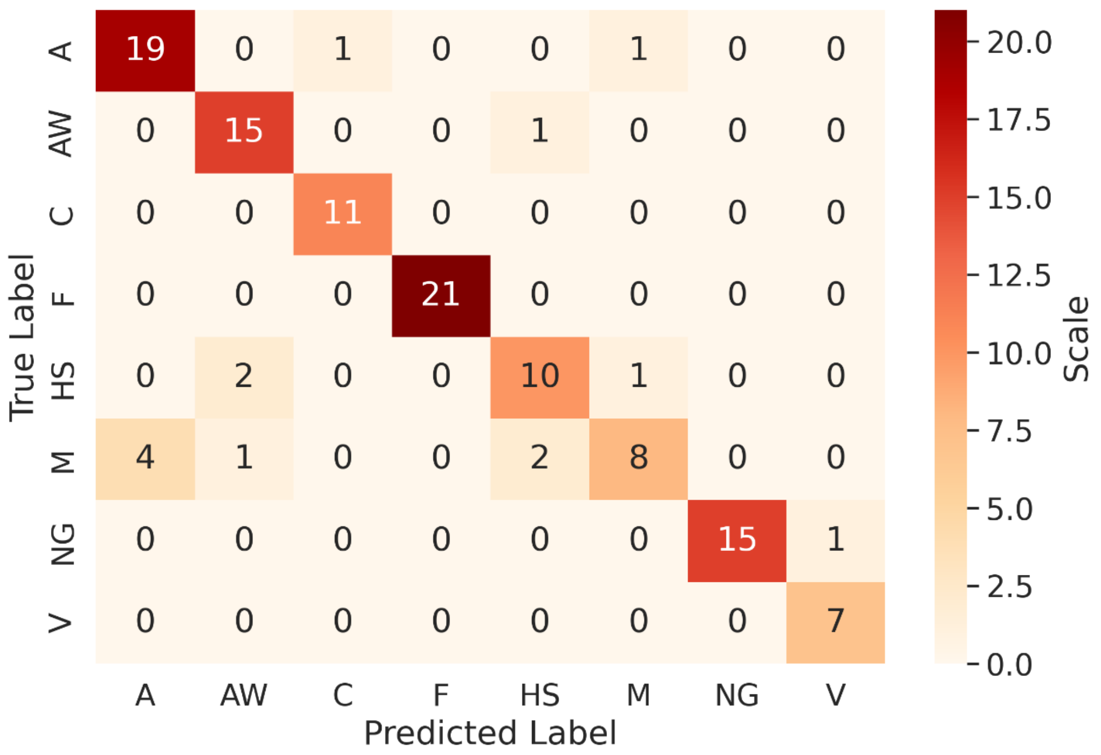
In this section, we will compare the results of this work with our previous work [22], talk about the parameters we have chosen for the benchmark programs, possible mitigations for our proposed attack, limitation, and future work.

5.1 Compare with Previous Work

In our previous work [22], we proposed a novel attack that fingerprints the accelerators running on the cloud FPGA via measuring the PCIe communication link. The experiment was conducted on Intel's cloud service, Devcloud [23]. The same attack method was used, which stressed the communication link (PCIe in our previous work) using benchmark accelerators and measuring the I/O bandwidth. Six benchmark accelerators were deployed onto the victim circuit for data trace collection in both works. Then the collected data traces were classified using machine learning models, including 1D-convolute, MLP, SVM, and random forest. The comparison of the classification accuracy result is shown in Table V. Both works successfully distinguished the communication pattern of the different benchmark accelerators deployed on the victim circuit. However, our current work has better accuracy result compared to this work. In our previous work, the random forest had an accuracy of 88%, and SVM had an accuracy of 69%.

TABLE V: Accuracy results in comparison with [22]

	Model	Accuracy
Current work	1D-Convolution	66.7%
	MLP	68.7%
	SVM	74.7%
	Random forest	93.3%
[22]	1D-Convolution	26.0%
	MLP	55.0%
	SVM	69.0%
	Random forest	88.0%



True \ Predicted	A	AW	C	F	HS	M	NG	V
A	19	0	1	0	0	1	0	0
AW	0	15	0	0	1	0	0	0
C	0	0	11	0	0	0	0	0
F	0	0	0	21	0	0	0	0
HS	0	2	0	0	10	1	0	0
M	4	1	0	0	2	8	0	0
NG	0	0	0	0	0	0	15	1
V	0	0	0	0	0	0	0	7

Figure 5.1 Confusion matrix of random forest classifier (adapted from [22]).

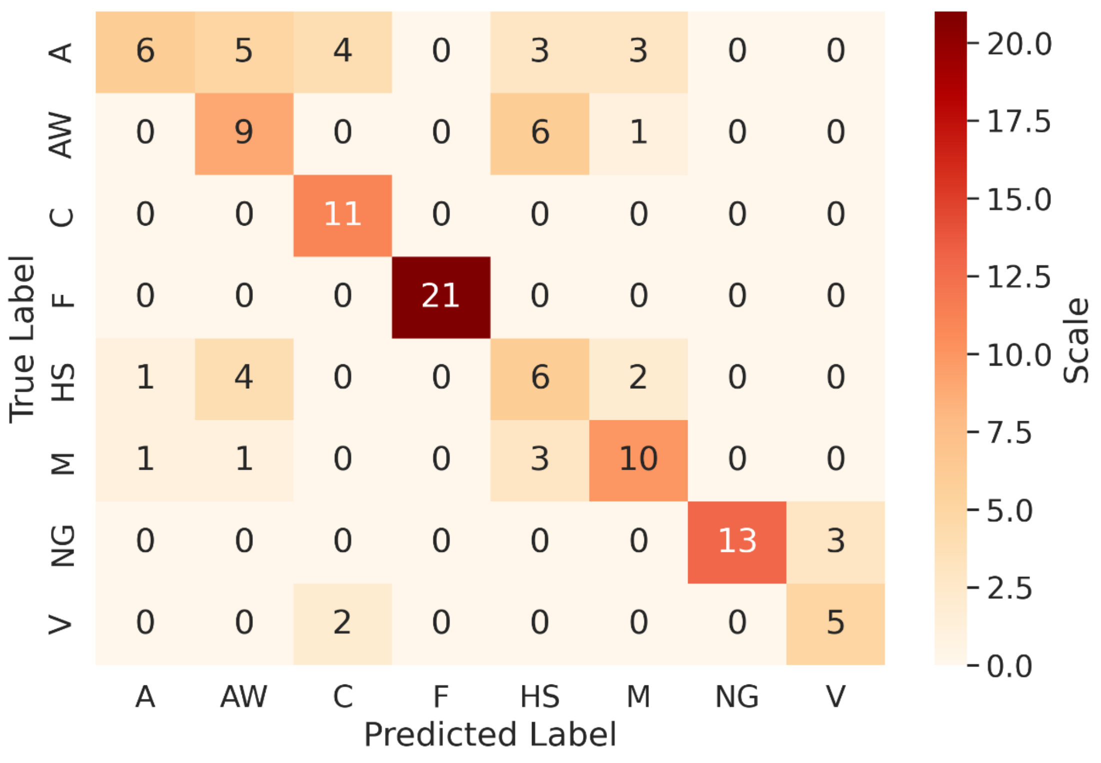


Figure 5.2 Confusion matrix of SVM classifier (adapted from [22]).

In both of our works, random forest and SVM classifiers had the better accuracy result. To find out why our current work had better accuracy results, we include the confusion matrix of random forest and SVM classifier models in Figure 5.1 and Figure 5.2.

After comparing the confusion matrix of both works, random forest maintains a high accuracy when classifying all traces. Still, we can see from Table III, the recall score and F-1 score on `matmul` of our current work are low. And in the confusion matrix, we can see that half of the traces of `matmul` are placed in the wrong category. This could be caused by the total number of traces collected for each victim accelerator being different in both works. In our previous work, we collected 200 traces for each victim accelerator, but in this work, the data traces we collected for each benchmark accelerator are fewer. However, our accuracy result with random forest shows that even when fewer data traces were provided, our random forest classifier can still distinguish the different communication patterns of the different benchmark accelerators used in the experiment. On the other hand, our SVM classifier could not predict the traces of `adder`, `matmul`, and `nw` to the correct categories. As of `adder` and `matmul`, it could be caused by insufficient amounts of data traces. On the other hand, the misprediction of `nw` could be caused by the SVM classifier not being good at making predictions when the data traces are similar to each other. In Figure 4.2, the t-SNE visualization of our traces, we can see that although the data traces of `nw` and `pathfinder` are close to each other, we can still claim that they lie in different areas of the 2D map. However, Figure 4.4 shows that all of the `nw` data traces are identified as it belongs to `pathfinder`. As we can see from Figure 5.1 and Figure 5.2, most of the data traces are placed to the correct slot in the confusion matrix in our previous work, but the overall accuracy of our previous work is lower than our current work. This could be caused by fewer victim accelerators being used in this work. In our previous work, we used eight victim accelerators, but in this work, we only used six victim accelerators.

We can conclude that our random forest classifier can still distinguish the I/O patterns of each victim accelerator even with fewer data traces provided. However, with more data traces provided, it clearly can do a better job at the classification task.

5.2 Parameter Choice

Since this work is transplanted from our previous work [22], the parameters we used in this experiment for our benchmark programs are adapted from the previous work, which is described in Section 4.2. In our previous work, we conducted experiments using different parameters for the benchmark programs. The details of the parameters we tested for the benchmark programs are listed below:

- NUM_ACCESS = 250, 500, 1000, 2000, and 4000.
- NUM_BUFFER = 50, 100, 200, 400, and 800.
- BUFFER_SIZE = 1 byte, 2 bytes, 4 bytes, 8 bytes, and 16 bytes.
- NUM_REPEAT = 1, 5, 10, 20, and 50.

We concluded that as the NUM_ACCESS and NUM_REPEAT increase, the accuracy will increase, but after a certain point (NUM_ACCESS = 1000 and NUM_REPEAT = 10), the accuracy will decrease. This is caused by as the NUM_ACCESS and NUM_REPEAT increase, the benchmark program's execution time will also increase and capture more I/O patterns. However, after NUM_ACCESS = 1000 and NUM_REPEAT = 10, the long execution time caused the benchmark program to lose the ability to capture the I/O patterns. Therefore, the accuracy of classification starts to drop. On the other hand, NUM_BUFFER and BUFFER_SIZE do not affect the accuracy of classification as much. Thus, we used the parameters that give the best accuracy results, which are NUM_ACCESS = 1000, NUM_BUFFER = 100, BUFFER_SIZE = 4 bytes, and NUM_REPEAT = 10.

5.3 Mitigations

Our experiment reveals the security vulnerability of the SoC-FPGAs, which is the communication link (i.e., AXI) between the HPS and FPGA fabric. Our attack method utilizes benchmark accelerators to stress the communication link and capture the information leakage from the unique I/O pattern of different accelerators as fingerprints. The data traces we collected are further used by machine learning models to perform classification tasks. The classification accuracy shows that our attack can accurately distinguish the unique I/O pattern of different accelerators.

To mitigate our proposed attack, which uses the information leakage from the communication link on the SoC-FPGAs, manufacturers could add an additional layer to the communication link for obfuscation. In this additional layer, random latency will be introduced to the data transmission process instead of transmitting raw data. Therefore, the I/O pattern of accelerators will be destroyed.

On the other hand, the board users could implement their host program, which transmits data to the different storage clusters to mitigate our attack. For example, board users could use both the onboard SDRAM and the Micro SD card as storage and set up a time interval for their host program, which changes the data transmitting destination between the onboard SDRAM and the Micro SD card. However, it can also disturb the communication pattern.

5.4 Limitations and Future Work

In this paper, we proposed a new side-channel attack targeting SoC-FPGAs. The experiment was conducted under the closed-world assumption, which means all the data traces we collected from the benchmark accelerators are predetermined and labeled. We only aimed to reveal the possible security vulnerabilities that existed in communication links. Our experiment proved that implementing benchmark programs that stress the communication link and measure the communication bandwidth with a random forest machine learning classifier is enough to conduct such fingerprint attacks under the closed-world assumption. For the open-world scenario, which requires including data collected from unknown and unwanted benchmark accelerators, the training dataset is more closely related to the real world. In the real world, attackers typically target a few significant and sensitive victim accelerators with other indifferent accelerators. Since our experiment only considered the closed-world scenario, it is possible that our attack method does not apply to the real-world setting.

For our future work, we will enhance the proposed attack and extend it to cover an open-world experiment setting. We could introduce more victim accelerators to the experiment to strengthen such attacks since we only target six victim accelerators in this experiment. It is valuable to see how the machine learning classifiers will perform when more victim accelerators are included in the training dataset. To extend the coverage to an open-world setting, we could introduce a small portion of the benchmark accelerators into the data collection phase, which leaves them unlabeled. Then we could determine if we should recognize the unlabeled data as a new instance of victim accelerators.

Chapter 6

Conclusion

In conclusion, we have proposed a new method of side-channel attack targeting SoC-FPGAs, which reveal the information leakage using the communication link. By using our attack method, attackers can tell what hardware accelerators are running on the SoC-FPGAs. Our experiment implemented six benchmark programs that stress the AXI communication link between the HPS and FPGA fabric on the DE1-SoC Development board and measure the I/O bandwidth. Machine learning models further use the collected data traces to perform classification tasks. The highest accuracy we achieved for the machine learning classification is 93.3% with random forest. Our experiment results show that the communication pattern of the running accelerators can be leaked from communication links such as AXI. It is beneficial for such side-channel attacks targeting SoC-FPGAs because we revealed a new security vulnerability of SoC-FPGAs. As far as we know, this is the first work that uses information leakage from the communication link to fingerprint accelerators on the SoC-FPGAs. In comparison with our previous work [22], which used a similar attack method targeting cloud-FPGAs, we can conclude that we successfully transplanted such an attack method from cloud-FPGAs to SoC-FPGAs. However, applying such attacks under the open-world assumption is still a challenging research problem. Therefore, we will continue implementing and enhancing our attack method to suit the open-world setting.

REFERENCES

[1] P. Swierczynski, G. T. Becker, A. Moradi, and C. Paar, "Bitstream fault injections (bifi)-automated fault attacks against sram-based fpgas," *IEEE Transactions on Computers*, vol. 67, no. 3, pp. 348–360, 2017.

[2] C. Krieg, C. Wolf, and A. Jantsch, "Malicious lut: a stealthy fpga trojan injected and triggered by the design flow," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD).* IEEE, 2016, pp. 1–8.

[3] Z. Weissman, T. Tiemann, D. Moghimi, E. Custodio, T. Eisenbarth, and B. Sunar, "Jackhammer: Efficient rowhammer on heterogeneous fpga-cpu platforms," *arXiv preprint arXiv:1912.11523*, 2019.

[4] M. Zhao and G. E. Suh, "Fpga-based remote power side-channel attacks," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 229–244.

[5] I. Giechaskiel, K. B. Rasmussen, and K. Eguro, "Leaky wires: Infomation leakage and covert communication between fpga long wires," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, ser. ASIACCS '18*. Association for Computing Machinery, 2018, p. 15–27.

[6] C. Ramesh, S. B. Patil, S. N. Dhanuskodi, G. Provelengios, S. Pillement, D. Holcomb, and R. Tessier, "Fpga side channel attacks without physical access," in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2018, pp. 45–52.

[7] J. Gravellier, J.-M. Dutertre, Y. Teglia, and P. Loubet-Moundi, "High-speed ring oscillator based sensors for remote side-channel attacks on fpgas," in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, 2019, pp. 1–8.

[8] M. Gobulukoglu, C. Drewes, W. Hunter, R. Kastner, and D. Richmond, "Classifying computations on multi-tenant fpgas," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 1261–1266.

[9] M. Tan, J. Wan, Z. Zhou and Z. Li, "Invisible Probe: Timing Attacks with PCIe Congestion Side-channel," *2021 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2021, pp. 322-338, doi:10.1109/SP40001.2021.00059

[10] S. Tian, I. Giechaskiel, W. Xiong, and J. Szefer, "Cloud fpga cartography using pcie contention," in *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2021, pp. 224–232.

[11] Terasic, "DE1-SoC User Manual" 2015, [Online; accessed 10 May 2023].

[12] A. Munshi, "The opencl specification," in *2009 IEEE Hot Chips 21 Symposium (HCS).* IEEE, 2009, pp. 1–314.

[13] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," *Mechanical systems and signal processing*, vol. 151, p. 107398, 2021.

[14] S. I. Gallant et al., "Perceptron-based learning algorithms," *IEEE Transactions on neural networks*, vol. 1, no. 2, pp. 179–191, 1990.

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[16] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[17] Terasic Inc. De1-SoC Board Support Package [Online].

[18] Xilinx, “Vitis Accel Examples’ Repository,” 2020, [Online; accessed 10 May 2023].

[19] H. R. Zohouri, N Maruyama, “Rodinia Benchmark Suite for OpenCL-based FPGAs,” 2018, [Online; accessed 10 May 2023].

[20] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.” *Journal of machine learning research*, vol. 9, no. 11, 2008.

[21] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[22] C. Fang, N. Miao, H. Wang, J. Zhou, T. Sheaves, J. M. Emmert, A. Sasan, H. Homayoun, “Gotcha! I Know What You are Doing on the FPGA Cloud: Fingerprinting Co-Located Cloud FPGA Accelerators via Measuring Communication Links,” *arXiv preprint arXiv: 2305.07209*, 2023.

[23] “Intel Devcloud,” 2021, [Online; accessed 10 March 2022].

[24] Altera, “Introduction to Cyclone V Hard Processor System (HPS),” 2014, [Online; accessed 10 May 2023].