Remote Power Side-Channel Attacks on BNN Accelerators in FPGAs

Shayan Moini*, Shanquan Tian†, Daniel Holcomb*, Jakub Szefer†, and Russell Tessier*
*Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, USA
†Department of Electrical Engineering, Yale University, New Haven, CT, USA

Abstract—**Multi-tenant FPGAs have recently been proposed, where multiple independent users simultaneously share a remote FPGA. Despite its benefits for cost and utilization, multi-tenancy opens up the possibility of malicious users extracting sensitive information from co-located victim users. To demonstrate the dangers, this paper presents a remote, power-based side-channel attack on a binarized neural network (BNN) accelerator. This work shows how to remotely obtain voltage estimates as the BNN circuit executes, and how the information can be used to recover the inputs to the BNN. The attack is demonstrated with a BNN used to recognize handwriting images from the MNIST dataset. With the use of precise time-to-digital converters (TDCs) for remote voltage estimation, the MNIST inputs can be successfully recovered with a maximum normalized cross-correlation of 75% between the input image and the recovered image.**

I. INTRODUCTION

Most major cloud providers now offer some form of remote, pay-per-use access to FPGAs [1], [2]. Furthermore, recent proposals for multi-tenancy have the promise of increasing FPGA utilization, especially in data center settings, by fitting multiple users' designs onto a single FPGA at the same time. The sharing of an FPGA by many users, unfortunately, opens up multi-tenant FPGA platforms to many new, potential attacks in which a malicious user can be co-located next to a victim user on the same FPGA.

Once co-located, a malicious user can try to learn information about the victim through a side channel. When the multi-tenant FPGAs are deployed in a remote data center, the malicious user is limited to only using attacks that do not require physical access. For example, voltage and power-based attacks [3] have been used to remotely extract encryption keys [4], [5] using circuits implemented on an FPGA by a malicious user. The danger of such attacks becomes especially worrisome as there is more and more interest in the FPGA acceleration of machine learning for image recognition, or other tasks, where sensitive information is processed. Existing work [6] shows that FPGA accelerators can speed up machine learning (ML) inference operations. Further, many cloud providers use FPGAs for the acceleration of ML workloads [7].

However, we show that ML accelerators are also vulnerable to remote attacks in a multi-tenant FPGA setting. This paper demonstrates a remote power-based side-channel attack on a binarized neural network (BNN) in an FPGA. In our attack, voltage fluctuations, caused by the changes in the power consumption of the convolution unit in the BNN are used to reconstruct the images on which inference is performed. Being able to recover the images that are processed by the ML algorithm could reveal sensitive imagery [8].

Unlike previous side channel attacks on machine learning accelerators on FPGAs [8], [9], [10], [11], our attack can be performed *remotely with no physical hardware access by the attacker*. We illustrate the details of our attack on the convolution unit of a BNN-based circuit that is used to recognize the handwriting images from the MNIST dataset [12]. Our attack and corresponding image recovery is successfully demonstrated on multiple generations of Xilinx FPGAs including a ChipWhisperer CW305 board [13] (Artix 7), a ZCU104 board [14] (Zynq UltraScale+), and a VCU118 board [15] (Virtex UltraScale+). Based on the evaluation we show that clearly recognizable input images can be retrieved and a maximum normalized cross-correlation of 75% is observed between the original and recovered images on the FPGA boards.

II. BACKGROUND AND RELATED WORK

A. Binarized Neural Networks

Deep neural networks (DNNs) [16] are a class of artificial neural networks that use multiple layers. In a DNN, each layer is responsible for extracting relevant features, and the output of each layer is passed as the input to the next layer. DNNs combine feature extraction with the capability of classical neural networks to map input data to a set of predictions.

Convolutional neural networks (CNNs) [16] are a subset of DNNs that are mostly used for classifying multi-dimensional data (e.g., images or video). The main distinctive property of CNNs is the convolution layer, which implements feature extraction by performing a convolution operation between the high-dimensional input data (called input feature maps) and kernels (small matrices of parameters that are computed during the training phase) to generate the output of the layer (called output feature maps).

Binarized neural networks [16] are CNNs that use aggressive quantization so that each element of the convolution kernel can be represented as either -1 or $+1$. In BNNs, all convolution input feature maps and kernels are comprised of binary values except for the first input layer which generally receives its input feature maps as matrices of integer values. We assume the input to the BNN is a grayscale image with each pixel being represented by an integer value. This image is the input feature map to the first convolution layer which convolves the input with $n \times n$ binary kernels. Each element of the convolution output (an output feature map) is calculated by multiplying a kernel with a $n \times n$ window of the input feature map and summing the resulting values. Sweeping an $n \times n$ kernel across the input feature map generates an output feature map. Additional BNN layers include pooling, batch normalization, fully-connected, and non-linear function layers. Our attack targets the first convolution layer in a BNN, which processes the input images directly.

For this work, the BNN is pre-trained with the MNIST dataset and the derived parameters, including convolution kernel values, are used in the BNN accelerator on an FPGA. We used the Keras framework [17] to train the BNN. The trained network is used during the inference stage to classify the input images of digits into one of ten categories (0 to 9).

Several researchers [8], [9], [10], [11] have explored side channel attacks on DNN accelerators on FPGAs. All of these approaches used physical access to the FPGA to collect needed information for the attacks. Meanwhile, we present a remote, power-based side channel that does not require physical access to FPGA supply voltage pins, uses on-chip voltage sensors to detect voltage fluctuations, and is demonstrated to work with three different FPGA boards.

B. *Voltage Sensing Using Time-to-Digital Converters*

In FPGAs, small drops in supply voltage occur in the vicinity of power consumption due to both IR and $L\frac{di}{dt}$ drops in the power distribution network and the chip packaging. Given that the propagation delay of combinational logic varies as a function of supply voltage, circuit delay in a sensor circuit can be used as a proxy for measuring the changes in the supply voltage. This approach is commonly used in voltage sensors based on time-to-digital converter (TDC) circuits [5]. In TDCs, each measurement reflects the delay of a circuit within a single clock cycle by observing how far through a tapped delay line a signal can travel during the cycle. This makes TDC sensors suitable for sensing short transient voltage fluctuations on the order of a single clock cycle. As we show in Section III-B, following others who previously exploited TDC designs [5], the high-speed carry logic in modern FPGAs makes a suitable delay line with taps that are on the order of 5-25 ps apart, depending on the FPGA technology and architecture.

III. Details of the Attack

A. *Threat Model*

This work focuses on a multi-tenant FPGA scenario where the victim user is running a machine learning inference module that is co-located on the same FPGA with the malicious user's modules. We assume that the attacker knows the structure of the BNN architecture that is used by the victim, but does not know the input, which is what they are trying to extract using our new attack. The victim input image is sent to the BNN accelerator in the FPGA in a secure manner (e.g., the input may be encrypted). The same input image is sent by the victim to the FPGA multiple times, a common case in video processing (e.g., of surveillance video). The output is likewise assumed to be securely sent back to the user. Thus the attacker's goal is to extract the input only by analyzing the behavior (voltage changes) of the BNN. The attacker can estimate the voltage drop across the FPGA power distribution network (PDN) during the execution of the convolution layer, as the BNN accelerator does the image classification. The acquired voltage estimates by using the TDC sensor serve as a side channel for extracting the victim's input image. The recovered image approximates the input image by distinguishing between foreground and background pixels of the image.

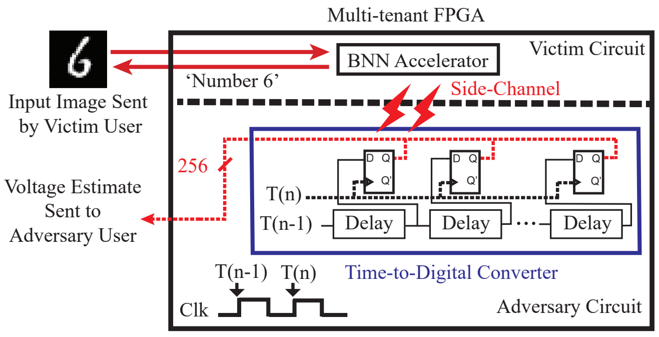


Fig. 1: Overview of attack implementation. The TDC outputs voltage estimates for each clock cycle of the first convolution layer. These estimates are used to reconstruct the input image.

B. *Attack Implementation*

The attack setup is shown in Figure 1. To extract the input image from the BNN accelerator, the adversary focuses on the first convolution layer which directly processes the input image. In the first convolution layer, an image is convolved with multiple distinct kernels to generate multiple output feature maps. In our attack, we use a voltage estimate trace from the execution of the first kernel of the first convolution layer for an input image. Since we assume that the same image is evaluated by the FPGA accelerator multiple times, multiple (N) similar traces are collected using the same input image and averaged. A high-pass filter is then used to remove noise. We leverage the observation that the background and foreground pixels can then be distinguished by analyzing the different magnitudes of the voltage measurements. This information can be represented by a histogram of instance counts of magnitude values in the filtered trace. Points in the histogram are used to label pixels as belonging to the image foreground or background based on the magnitude of their voltage measurement. The result of the analysis is a reconstructed image that approximates the image that was input to the BNN.

For the first convolution layer of a BNN trained on the MNIST dataset with a 28×28 grayscale image as the input and 64 kernels of size 3×3, the convolution operation can be represented as:

$$O^j_{x,y} = \sum_{a=0}^{2}\sum_{b=0}^{2} \omega^j_{a,b} \times I_{x+a,y+b}, \quad j \in [1,64] \qquad (1)$$

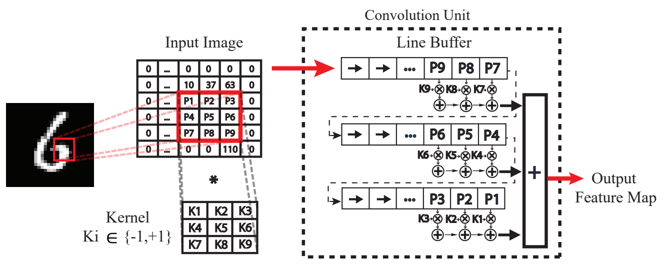


Fig. 2: Detailed view of the convolution unit. Output is generated from the 3×3 image window, shown in the red box, and the kernel.

The $O^j_{x,y}$ parameter represents the location (x,y) in the jth output feature map which is calculated by convolving a window (same size as the kernel) of the input feature map (I) and the corresponding kernel (ω^j).

In our setup, the convolution unit uses a line buffer architecture to hold and provide data values to the convolution [8]. As shown at the right in Figure 2, the line buffer is arranged in three rows, each of which processes one line of the convolution operation. The line buffer is a shift register that receives one pixel from the input feature map (the image) per clock cycle and shifts its values to the right. The length of each row in the line buffer matches the length of the input feature map of the convolution operation (28 for the first layer in our implementation). The rightmost word of each row of the line buffer enters the next row from the left, and the rightmost word of the last row is discarded. The rightmost three words of each of the three rows of the line buffer constitute the image window whose values are multiplied with values from the 3×3 kernel. Since binary kernels are used in a BNN, each image pixel in the current image window is added to or subtracted from (based on a kernel value of -1 or +1) the other pixels in one clock cycle using a combinational adder tree. One output feature map value is generated every clock cycle.

An adversary can abuse the shared FPGA PDN to sense local supply voltage changes, which can reveal information about the per-cycle power consumption in the convolution unit. The power consumption is due in part to the switching activity in the BNN accelerator, including the convolution unit, which causes supply voltage to be correlated to the data processed. The small PDN fluctuations are reflected in the sampled values of the TDC, and the TDC samples can then be used to recover a facsimile of the input image.

To observe the voltage fluctuations, the TDC measures the delay of signal propagation among the TDC stages. The 256-stage TDC contains a chain of fast fixed-purpose FPGA elements typically used to perform timing-critical carry operations in arithmetic circuits (*Carry4* or *Carry8* depending on FPGA family) [18]. The TDC is activated by sending the rising edge of a clock through the adjustable delay and the carry chain to the flip-flops attached to the 256 stages of the carry logic. The Hamming weight of the sample indicates how far through the carry chain the rising edge has propagated by the time the next rising clock edge arrives. When the supply voltage drops, the propagation delay of the circuit increases, and the rising edge will have propagated through fewer carry stages before the next rising clock edge, and hence the sample captured in the flip flops will have a lower Hamming weight.

TABLE I: Details of the evaluation boards used for the experiments. The system clock generates the clock for the BNN accelerator and TDC module.

Board Name	Device	FPGA Family	Clk (MHz)
ChipWhisperer	XC7A100T	Artix 7	50
ZCU104	XCZU7EV	Zynq UltraScale+	120
VCU118	XCVU9P	Virtex UltraScale+	100

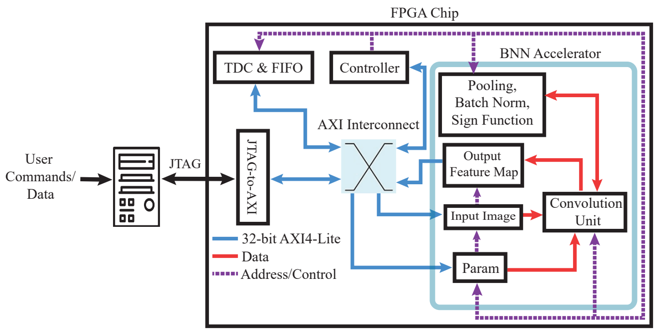


Fig. 3: Overview of the architecture implemented on the ChipWhisper, ZCU104, and VCU118.

C. Experimental Approach

Three Xilinx FPGA-based boards, listed in Table I, were used for experimentation. The ChipWhisperer CW305 board [13] provides a platform for examining power side-channel attack scenarios. The board supports off-chip voltage measurement using a capture board via a low-noise and high-bandwidth connection to the main FPGA 1V DC supply pin. The ChipWhisperer-Lite capture board [19] contains a 10-bit analog-to-digital converter (ADC) with 105 MS/s sampling rate. As described in Section IV, both the capture board and an on-FPGA TDC were used with the ChipWhisperer to obtain voltage traces. Xilinx ZCU104 [14] and VCU118 [15] evaluation boards were also used for evaluation, with on-FPGA TDC-based sensors used to collect voltage traces. Off-chip FPGA supply voltage measurements were not collected for these two boards.

The implementation of the attack architecture for the three boards is similar. The BNN accelerator and supporting test circuitry, as well as the TDC and FIFO used for performing the attack, are shown in Figure 3. The data movement between different components of the design takes place through an AXI4-Lite on-chip communication protocol. Off-chip communication (data movement and control commands) uses a Xilinx JTAG-to-AXI converter module to provide user access to the on-chip AXI bus through a JTAG interface.

The on-chip controller in Figure 3 sets memory addresses and controls the operation of the convolution unit. This controller has registers that set the parameters of the three on-chip block memories used to store on-chip data. *Input Image* stores

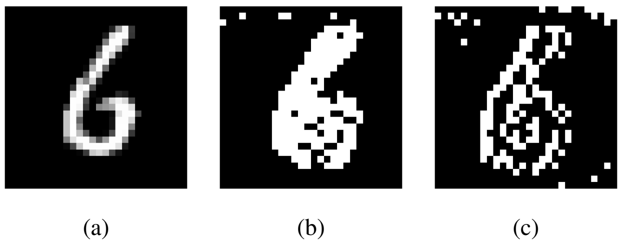


Fig. 4: (a) Input image to the convolution unit from the MNIST data set, (b) recovered image with supply voltage traces from the ChipWhisperer board, (c) recovered image with TDC traces.

the input feature map, *Output Feature Map* stores the result of the convolution, and $Param$ stores the binary values of the convolution kernels for the current layer with +1 represented by the bit value 1 and -1 by the bit value 0. For each layer of the BNN, the input feature map and corresponding kernel values are loaded into *Input Image* and $Param$ memories by the user, then the convolution operation is performed, and finally the results are collected from *Output Feature Map*.

IV. Attack Analysis on ChipWhisperer

In this section, we describe characterization experiments using the ChipWhisperer CW305 board. These experiments use both on- and off-FPGA voltage measurements to examine voltage fluctuations during the convolution operation as input images are processed. The ChipWhisper is an *ideal* board in that its bypass capacitors have been removed and dedicated voltage measurement resources are provided. With information gathered from the ChipWhisperer, the attack was then deployed on other, more realistic boards.

A. Off-Chip Characterization of Convolution Operation

In an initial set of experiments, the ADC on the ChipWhisperer-Lite capture board was used to sample the FPGA core supply voltage level at the rising edge of each convolution unit clock cycle. The supply voltage level drops of all 28×28 (784) convolution operations for the first kernel applied to the input image, illustrated in Figure 4a, were measured. The experiment was run 10 times and the mean values of the voltage drops observed at the FPGA supply input at each clock cycle are shown in Figure 5. The 125 orange circles in Figure 5 show clock cycles during which the 125 pixels from the image foreground are used in convolution for the first time (the clock cycle when the foreground pixel is in location $P9$, multiplied by $K9$ in Figure 2). Figure 5 shows that the clock cycles corresponding to foreground pixels have a higher voltage drop compared to other clock cycles. These differences can be used to differentiate between foreground and background pixels.

The voltage drops induced by the foreground pixels can be explained by examining Equation 1. Each pixel of the output feature map ($O_{x,y}$) is calculated using an image window and a kernel. The image (a grayscale picture of a digit with each pixel an integer between 0 and 255) has low-valued pixels

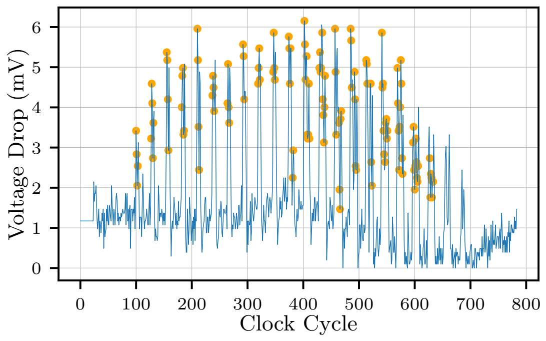


Fig. 5: Voltage trace from the ChipWhisperer FPGA while running the convolution unit shown in Figure 2. The y axis illustrates the absolute value of the measured voltage drop due to convolution unit activity. The 125 orange circles correspond to the clock cycles that process foreground pixels of the input image (Figure 4a).

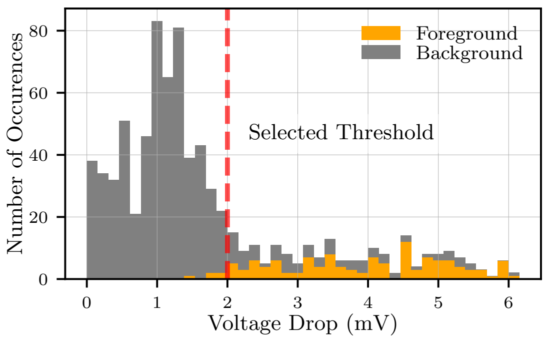
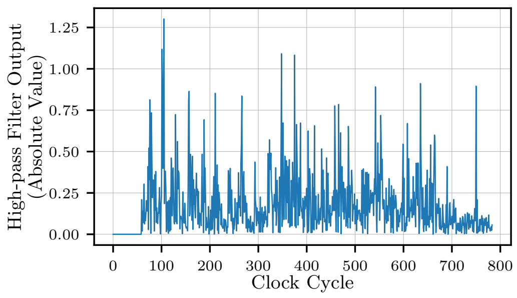


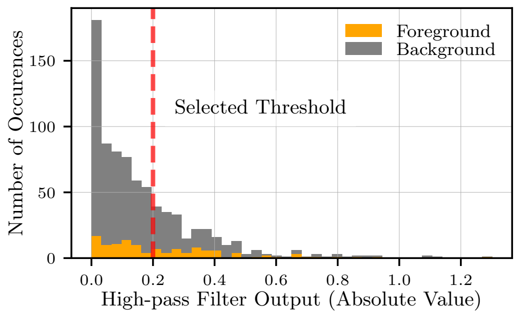
Fig. 6: Histogram of the voltage drop due to convolution unit operation for convolution operations with the same input image and kernel. Each occurrence in the histogram represents the average of ten trials of processing the same pixel and kernel. The bars corresponding to foreground pixels are colored in orange and those corresponding to background pixels are colored dark grey. The selected threshold (boundary) between foreground and background pixels is marked in the histogram.

(close to 0) for background and high-valued pixels (close to 255) for the foreground. For the calculation of the output feature map, the kernel values are constant. However, the values of the processed input image pixels in specific locations in the line buffer change between background and foreground pixels during the convolution operation. The dynamic power consumption (and resulting voltage drop) of processing foreground pixels is larger than for background pixels. Specifically, foreground pixels result in the generation of larger magnitude results for the multiply and accumulate operations when the convolution operation processes these pixels. As a result of generating these values, significant switching activity takes place in the adder tree of the convolution unit and resultant voltage drops can be observed.

To illustrate the range of voltage changes due to the convolution of the input image, a histogram of the absolute value of voltage drop measurements in Figure 5 is shown in Figure 6. The histogram contains 40 bins evenly distributed in value

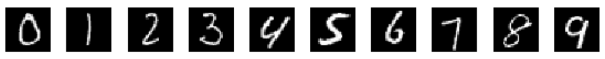


(a) Recovered trace after applying a high-pass filter to the TDC Hamming weight values (absolute value).

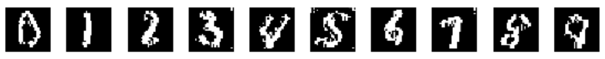


(b) Histogram of the filtered TDC trace with the selected threshold shown.

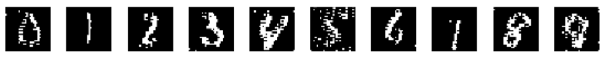
Fig. 7: TDC data recovered from ChipWhisperer: (a) Trace after applying high-pass filter and removing low-frequency envelope (absolute value), (b) Histogram of the filtered TDC trace with the selected threshold.



(a) Input images



(b) Recovered images from ZCU104



(c) Recovered images from VCU118

Fig. 8: Input images and recovered images from ZCU104 and VCU118 boards using only TDC measurements.

between 0 to 6 mV. The boundary between foreground and background pixels can be distinguished with a threshold.

Generally, the processing of background pixels leads to small voltage drops that are clustered on the left of the histogram and the processing of foreground pixels leads to a range of larger voltage drops on the right of the histogram. The threshold can be identified by locating a downward gradient in occurrence counts over multiple voltage bins. In the ChipWhisperer, this transition took place over five bins located just before 2 mV.

In Figure 6, the dashed red line shows the chosen threshold value. All voltage drops created by input pixels that fall to the left of the line are classified as background pixels, while the ones to the right are classified as foreground pixels. The recovered image using the threshold is shown in Figure 4b.

B. *TDC-Based Characterization of Convolution Operation*

The characterization of convolution unit voltage drops described in the previous subsection was performed using voltage traces obtained by the ChipWhisperer-Lite capture board. In this section, we describe characterization experiments that use voltage measurements obtained by a TDC sensor implemented in the ChipWhisperer FPGA. The TDC architecture was described in Section III-B. The 256-bit TDC carry chain for the Artix-7 FPGA on the ChipWhisperer board consists of $Carry4$ carry primitives. The sensitivity for each TDC stage is close to 25 ps.

For each clock cycle, the flip-flop values from the TDC were saved in a 256-bit wide FIFO, forming one voltage estimate. This experiment was performed 100 times using the same input image and kernel. The voltage estimates at each clock cycle for the 100 traces were then averaged to minimize noise, forming a collection of 784 Hamming weights, one for each pixel. A high-pass Butterworth digital filter was applied to the values to retain voltage fluctuations due to convolution unit activity while removing a low-frequency supply voltage envelope. For each point in the plot, the filter determines an average Hamming weight value over the previous ten clock cycles (a running average window). This value is then subtracted from the Hamming weight value at the current clock cycle, leading to the plot shown in Figure 7a. Subsequently, the image was recovered with the histogram threshold shown in Figure 7b, identified with the gradient method described in Section IV-A. The recovered image is shown in Figure 4c.

C. *TDC-Based Attack Summary*

To summarize, the following steps are performed to recover a reconstructed image using the on-FPGA TDC:

1) Voltage estimates are collected for each input pixel during operation of the convolution unit for the first kernel of the first convolution layer.
2) Voltage estimates for each pixel are averaged across all runs with the image to generate a single trace. The averaged estimates are represented using Hamming weights.
3) A Butterworth high-pass filter is used to remove low-freqency power supply ripple from the averaged Hamming weights.
4) A histogram of the resulting values is created and a threshold is used to differentiate foreground and background pixels, forming a recovered image.

V. Image Extraction using the Attack

After initial experimentation with the ChipWhisperer CW305, our attack was applied to the two boards described in Section III-C to see how well the attack can perform on

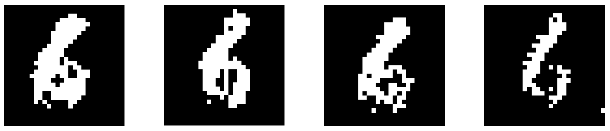

(a) ZCU104, cross-die (b) ZCU104, adjacent (c) VCU118, cross-die (d) VCU118, adjacent

Fig. 9: Recovered images with adjacent and cross-die placement for 3,000 runs.

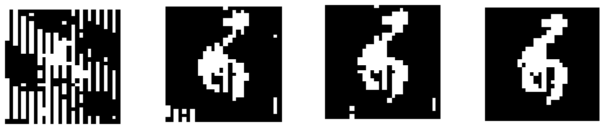

(a) 100, 0.19 (b) 500, 0.61 (c) 1,000, 0.65 (d) 3,000, 0.75

Fig. 10: Recovered images for the ZCU104 board for (number of runs, normalized cross-correlation with the original image).

commercial off-the-shelf boards that were not designed to study side channel attacks. The hardware for these platforms was not modified for our experimentation. The experimental setup for these platforms including the BNN accelerator is shown in Figure 3.

Recovered images for the ZCU104 and VCU118 boards using TDC measurements are shown in Figure 8. For these experiments, the TDC was placed adjacent to the BNN accelerator in the FPGA fabric (in the next row of logic blocks). For the ZCU104 and VCU118, the same input image and kernel were used 3,000 times.

To study the importance of TDC location on the FPGA die relative to the location of the BNN accelerator, the BNN was moved to a location on the opposite side of the die for the ZCU104's UltraScale+ FPGA. The experiments were rerun for the digital image shown in Figure 4a. To compare the quality of the recovered images with cross-die placement of the TDC versus the results from adjacent placement for the selected digit, the normalized cross-correlation (CCR_N), derived from cross-correlation (CCR), between the recovered images and the input image for both adjacent and cross-die TDC placements were calculated using Equations 2 and 3. Here, $\bar{A}$ and $\bar{B}$ represent the mean pixel values of the images. The CCR_N value provides a quantitative metric for comparing the similarity of the input image and a recovered image.

$$CCR = \sum_{(i,j)\in N^{28\times 28}} \left[(A[i,j]-\bar{A})\times(B[i,j]-\bar{B})\right] \quad (2)$$

$$CCR\_N = \frac{CCR}{\sqrt{\sum\left(A[i,j]-\bar{A}\right)^2 \times \sum\left(B[i,j]-\bar{B}\right)^2}} \quad (3)$$

The normalized cross-correlations for adjacent and cross-die placement for the ZCU104 are 0.745 and 0.594, respectively. The corresponding values for the VCU118 are 0.678 and 0.646. Figure 9 shows the recovered images for different placement strategies for the two boards.

To obtain recognizable reconstructed images, the same input image is processed by the same kernel numerous times. To evaluate the effect of number of runs on image quality, we again used the image shown in Figure 4a. For both FPGA boards, the normalized cross-correlation (Equation 3) of the recovered image and the original image versus the number of times the input image was processed by the first kernel was calculated. Figure 10 shows recovered images for an increasing number of runs.

VI. Conclusion

This paper presents a remote power side-channel attack on binarized neural networks targeting multi-tenant FPGAs. We show that it is possible to accurately extract image inputs to a BNN by collecting and analyzing on-chip voltage estimates with TDCs. Our approach has been applied to three FPGA boards. Our experiments successfully recovered recognizable images for all ten digits from the MNIST dataset.

Acknowledgment

This research was funded in part by NSF grants CNS-1901901 and CNS-1902532.

References

[1] Amazon Web Services, "Amazon EC2 F1 instances," https://aws.amazon.com/ec2/instance-types/f1/, Accessed: 2020-8-23.

[2] Baidu Cloud, "FPGA cloud compute," https://cloud.baidu.com/product/fpga.html, Accessed: 2020-03-29.

[3] O. Glamocanin *et al.*, "Are cloud FPGAs really vulnerable to power analysis attacks?" in *DATE*, 2020.

[4] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *IEEE S&P*, 2018.

[5] F. Schellenberg *et al.*, "An inside job: Remote power analysis attacks on FPGAs," in *DATE*, 2018.

[6] S. Zeng *et al.*, "Enabling efficient and flexible FPGA virtualization for deep learning in the cloud," in *FCCM*, 2020.

[7] Microsoft, "What are Field-Programmable Gate Arrays (FPGA) and how to deploy," https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-fpga-web-service, Accessed: 2020-05-14.

[8] L. Wei *et al.*, "I know what you see: Power side-channel attack on convolutional neural network accelerators," in *CSAC*, 2018.

[9] A. Dubey *et al.*, "Maskednet: A pathway for secure inference against power side-channel attacks," in *HOST*, 2020.

[10] K. Yoshida *et al.*, "Model-extraction attack against FPGA-DNN accelerator utilizing correlation electromagnetic analysis," in *FCCM*, 2019.

[11] W. Hua *et al.*, "Reverse engineering convolutional neural networks through side-channel information leaks," in *DAC*, 2018.

[12] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database." http://yann.lecun.com/exdb/mnist/, Accessed: 2020-05-19.

[13] NewAE Technology, Inc., "CW305 ChipWhisperer Artix FPGA target board," https://rtfm.newae.com/Targets/CW305ArtixFPGA/, Accessed: 2020-11-22.

[14] Xilinx, Inc., "ZCU104 evaluation board," https://www.xilinx.com/products/boards-and-kits/zcu104.html, 2020, Accessed: 2020-05-19.

[15] ——, "VCU118 evaluation board," https://www.xilinx.com/products/boards-and-kits/vcu118.html, 2020, Accessed: 2020-05-19.

[16] V. Sze *et al.*, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295 – 2329, Dec. 2017.

[17] K. Ding, "Binarized dense and Conv2D layers for Keras," https://github.com/DingKe/BinaryNet, Accessed: 2020-05-19.

[18] S. Moini *et al.*, "Understanding and comparing the capabilities of on-chip voltage sensors against remote power attacks on FPGAs," in *MWSCAS*, 2020.

[19] NewAE Technology, Inc., "CW1173 ChipWhisperer-Lite capture board," https://wiki.newae.com/CW1173_ChipWhisperer-Lite, Accessed: 2020-05-22.