Blurring Boundaries: A New Way to Secure Approximate Computing Systems

Pruthvy Yellu, Landon Buell, Dongpeng Xu, and Qiaoyan Yu*
University of New Hampshire
Durham, NH, USA
*qiaoyan.yu@unh.edu

ABSTRACT

Approximate computing (AC) techniques have been widely used to improve the performance of computing systems by trading off accuracy. However, recent literature projects that the utilization of approximation could bring in new security threats to computing systems. This work presents two practical attacks on the AC systems for multilayer perceptron (MLP) and Sobel algorithm based image edge detection. The case studies in this work indicate that the approximation mechanism in AC systems can be exploited to conduct stealthy attacks, which suddenly cause significant degradation in accuracy and lead to unpredictable primary outputs. To address the emerging threats on AC systems, this work proposes to blur the boundary between approximate and precise computing submodules in AC systems. This new defense method obscures that boundary with three obfuscation schemes such that adversary could not easily identify the right target to precisely perform hardware tampering attacks. Simulation results show that the proposed method can effectively reduce the attack success rate.

CCS CONCEPTS

• **Security and privacy → Security in hardware**; • **Computer systems organization → Neural networks**; • **Hardware → Digital signal processing**.

KEYWORDS

Approximate computing; artificial neural network; machine learning; edge detection; hardware Trojan; obfuscation.

ACM Reference Format:
Pruthvy Yellu, Landon Buell, Dongpeng Xu, and Qiaoyan Yu*. 2020. Blurring Boundaries: A New Way to Secure Approximate Computing Systems. In *Proceedings of the Great Lakes Symposium on VLSI 2020 (GLSVLSI '20), September 7–9, 2020, Virtual Event, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3386263.3407593

1 INTRODUCTION

Approximate Computing (AC) has emerged as a promising technology to further improve the performance and energy efficiency of computing systems at the cost of reduced accuracy [4]. Particularly, AC techniques have great potential to benefit error-tolerant applications such as image recognition, signal processing in multimedia and machine learning [11, 13, 20]. Existing literature has demonstrated that various approximate computing techniques are available for system design, software, storage architecture and arithmetic circuits [17]. At system level, approximation can be achieved by using different computation accelerators [7] or modifying Instruction Set Architecture (ISA) [3] to activate special approximation operators. Software techniques such as loop perforation [1] and memoization [9] reduce execution time to save power consumption. Voltage over-scaling [16] and memory refresh rate reduction [5] techniques trade accuracy for better performance and energy efficiency. At hardware level, precision scaling [12] and logic minimization [21] are effective for power reduction. More approximation techniques are summarized in the survey papers [8, 15].

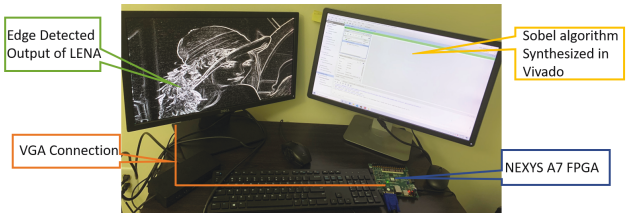
In spite of the advantages on performance improvement and power saving, AC techniques also bring in new security vulnerabilities to AC systems. The work [6] implies that the utilization of approximate adders will make it easier for reverse engineers to identify the critical path. The work [10] briefly analyzes whether approximate techniques could facilitate reverse engineering, integrated circuit piracy, side-channel analysis, and hardware Trojan attacks. Security vulnerabilities on approximate storage (DRAM, SRAM and Phase Change Memory) are discussed in the visionary work [17]. Attack models for approximate arithmetic circuits are proposed in our recent work [18]. Although existing literature envisions the potential attacks in AC systems, more concrete attack implementations in practical applications are needed to inspire researchers to investigate countermeasures against those projected attacks in AC systems.

Since the attacks mentioned above may cause severe consequences, it is imperative to explore effective defense mechanisms to harden AC systems. The work [18] proposes a method that shuffles the inputs for the approximate computing modules to thwart input interconnect attacks. In addition, that work also uses an alternate approximate function to identify hardware tampering attacks on the approximation hardware. To facilitate the detection of hardware attacks on AC systems, the work [19] proposes and compares different accuracy metrics for the evaluation of approximated outputs. Following those footprints, this work analyzes the root cause of potentials attacks on AC systems and proposes the general principles of the defense mechanisms customized for AC systems. More specifically, our main contributions are as follows.

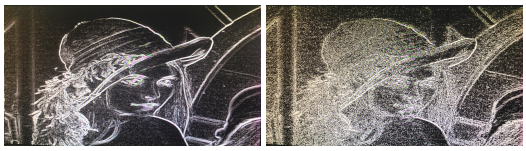
- Two case studies are performed to demonstrate that malicious manipulations on AC operations could lead AC systems to have significant performance degradation.

GLSVLSI '20, September 7–9, 2020, Virtual Event, China

ACM ISBN 978-1-4503-7944-1/20/09…$15.00
https://doi.org/10.1145/3386263.3407593



(a)



(b) (c)

Figure 1: Attack on approximated edge detection. (a) Experimental setup, and image edges extracted by the approximate implementation of Sobel edge detection algorithm (b) without and (c) with hardware Trojan attacks.

- We envision that a new type of stealthy attack could be implemented in AC systems.
- To obscure approximate computing systems, this work proposes to blur the approximate-precise boundary (APB) by obfuscating when to approximate (WNA), what to approximate (WTA), and how to approximate (HWA).

The rest of this work is organized as follows. Section 2 demonstrates two practical attacks on AC systems. Section 3 introduces a new potential attack, error compensated attack, that could harm AC systems. Section 4 proposes a novel obfuscation method for AC systems. Experimental results are provided in Section 5. This work is concluded in Section 6.

2 PRACTICAL ATTACKS IN APPROXIMATE COMPUTING SYSTEMS

Recent literature [10, 17, 18] predicts that approximation at different levels could incur stealthy attacks, which could hide the attack effect in the relaxed noise margin tolerated by the approximate system. In this section, we use two case studies to demonstrate practical attacks in AC systems and their impacts.

2.1 Case Study 1: Attacks on Image Edge Detection

Sobel edge detection [14] is a prominent algorithm for image edge extraction. As numerous addition operations are utilized in the Sobel algorithm, we replace a portion of precise additions with approximate additions to improve the speed of processing. In the approximate addition, the last three least important bits (LSB) are computed approximately using the revised Boolean expression introduced in [2]. Our experimental setup is shown in Fig. 1(a). We implemented the Sobel algorithm in VerilogHDL and mapped its hardware description to a Xilinx NEXYS A7 FPGA. Then, the detected edges of the image *Lena* with a resolution of 640*480 was displayed on a monitor.

As approximate computing relaxes the precision, the verification on AC systems generally relies on the metric of average accuracy.

Table 1: FPGA utilization and power consumption of Sobel implementation

FPGA Utilization	Precise Sobel	Approx. Sobel w/o HT	Approx. Sobel w/ HT
Bounded IOB	34.29%	34.29%	34.29%
LUT as Logic	0.16%	0.15%	0.21%
Slice	0.21%	0.19%	0.16%
Power (W)	11.933	9.089	14.987

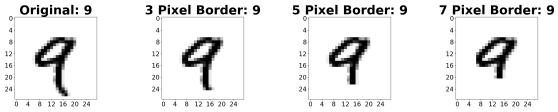
Thus, approximate modules may be prone to hardware Trojan (HT) attacks [18]. In this case study, we implemented an always-on HT that swaps the MSB and LSB input bits of the module for Sobel edge detection. The comparison between Figs. 1 (b) and (c) shows that the hardware Trojan inserted in the approximate processing module could lead to notable impact on the extracted edges. Note that, we introduced the always-on HT to exaggerate the attack consequence. In reality, most HTs will be rare-event triggered and the HT effect will also be localized at a particular region of the image.

We further assessed the hardware implementation cost of the Sobel edge detection modules: precise version, approximate version without HT, and approximation version with HT. As shown in Table 1, the input/output blocks (IOB) for three versions are exactly the same and the approximate addition reduces the FPGA utilization (including LUT and slice). Even though the hardware Trojan consumes some FPGA resources, the number of slices used by the approximate Sobel with HT is still less than those occupied by the precise Sobel. This is problematic. If we only compare the FPGA utilization rate between precise and approximate versions, we may overlook the existence of stealthy HTs in the approximate module. The power utilization of the approximate Sobel algorithm is 24% less than the power consumed by the precise version. The hardware Trojan inserted in the Sobel core causes the power consumption to increase by 65% over the approximate Sobel without HT. However, the HT induced power increment could be negligible if the entire image processing system contains other complicated function modules.

2.2 Case Study 2: Multilayer Perceptron

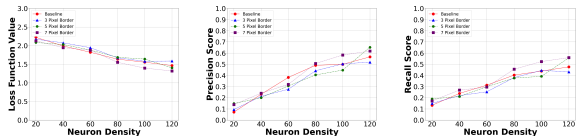
A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN), which has been widely used in the classification for speech recognition, image recognition, and machine translation software. To accelerate classification, approximate computing is often leveraged in the MLP for complex problems. In this case study, we implemented a practical attack on the AC operations adopted in MLP and demonstrated the impact of that attack on the critical metrics typically assessed in classification. A subsection of the MNIST data set containing 28*28 pixel images of handwritten digits was used to perform our assessment. Each pixel was given by an integer 0 through 255 to indicate the gray-scale value. From the full data set, the first 10,000 images were chosen to train each model, and the next 6,000 were utilized to test the model.

As shown in Fig. 2, the handwritten digit is roughly centered in the MNIST image. It is reasonable to assume that approximating a border of pixels around the outside of the image would only produce small changes in the performance of the classifier model. We tested a baseline set of samples against the samples that have varying depth pixel borders approximated. This effectively created



(a) (b) (c) (d)

Figure 2: MNIST handwritten digit samples showing how approximating a border of pixels around each images affects the figure.



(a) (b) (c)

Figure 3: Impact of border approximation on classification metrics (a) loss function value, (b) precision score and (c) recall score.

a padding of 3, 5, or 7 pixels on each image. For some samples, even when approximating a large border of pixels, the characteristics of the digit can be preserved as shown in Fig. 2(d). We adjusted the hidden layers to contain 20, 40, 60, 80, 100, and 200 neurons per layer (referred to as *neuron density*). Ten training epochs were used in all cases, with batch sizes of 100 samples in each step for a Stochastic Gradient Descent Optimizer. As shown in Fig. 3, the border approximation presents no major deviation in performance from the baseline model. This means that while a subset of pixels are approximated, the dominant features preserved in the figure still allows the optimization process to converge on a similar set of parameters as the baseline samples. Despite of the small change in performance from the baseline, the overall performance of the model remains consistent for varied number of neuron density and approximated border pixels.

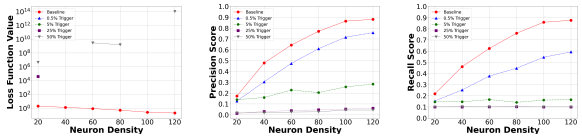
Approximation in MLP could be exploited to develop an attack surface. In general, an MLP neural network model is composed of multiple layers of *neurons*, each containing an *activation* function. Given a layer l of activations $\vec{x}^{(l)}$, the activations in the next sequential layer, $\vec{x}^{(l+1)}$, are computed with weighting matrix $\hat{W}^{(l)}$, bias vector $\vec{b}^{(l)}$, and activation function f such that:

$$\vec{x}^{(l+1)} = f\left(\hat{W}^{(l)} \cdot \vec{x}^{(l)} + \vec{b}^{(l)}\right) \quad (1)$$

The recurrence of Eq. (1) is used to pass information forward through the network. In a network with L layers, raw information is given with $\vec{x}^{(0)}$ and the network's prediction is provided by the final layer, $\vec{x}^{(L-1)}$. In this case study, we selectively manipulated the arithmetic operation in neurons. More specifically, we introduced a *bit-muting* attack function to the matrix-vector product in Eq. (1). When a Trojan trigger condition *HTtrigger* is met, Eq. (1) was replaced with its attack variant expressed in Eq. (2), where the attack function $A[\cdot]$ forces the MSB of the exponent field for the matrix-vector product to zero.

$$\vec{x}^{(l+1)} = f\left(A\left[\hat{W}^{(l)} \cdot \vec{x}^{(l)}, HTtrigger\right] + \vec{b}^{(l)}\right) \quad (2)$$

The same MLP modeling and MINST images were utilized in the following experiments. For each network model, there were always 784 input neurons and 10 output neurons (one for each class). To assess the impact of our attacks on the MPL classification, we adopted the following metrics: (1) accumulated loss function for the final iteration, (2) average precision score, and (3) average recall score. The average of each metric is plotted in Fig. 4. The baseline was the MLP without activation attacks, and four different attack trigger rates were executed for the purpose of comparison. Note that the lack of a data point means that the value of that metric diverged to infinity or some other undefined values.



(a) (b) (c)

Figure 4: Impact of bit-muting attacks in approximate neurons on MLP classification outcome. (a) Loss function, (b) precision score, and (c) recall score for one hidden layer.

As shown in Fig. 4, the bit-muting attack on the approximate neurons leads to dramatic reductions in the average precision and recall scores across all tested classifier models when compared to the un-attacked baseline. Since this type of attack targets the feed-forward mechanism of the network, information passed through the network will not produce consistent or reliable outputs, thus preventing the classifier from making appropriate decisions. The exclusion of MSB in the double precision floating-point entries in the network destroys the numerical integrity and thus essentially renders this subset of classifier models ineffective entirely, which is indicated by the inability to minimize the loss function. Our case study indicates that the classifier could fail significantly if the approximation function in MLP neurons is compromised (even with a low trigger probability, e.g. 0.5%).

3 PROJECTED NEW ATTACK IN AC SYSTEMS: ERROR COMPENSATED ATTACK (ECA)

We project that there will emerge a new attack in AC systems: error compensation attack (ECA). Without losing generality, we assume that 70% precise operations in an AC system guarantee that the overall accuracy meets the system requirement and the rest 30% approximated operations reduce the total power consumption or improve the average performance. As shown in Fig. 5(a), the error compensated attack first creates an attack zone (including 25% operations) in the original precise computing region to fulfill the malicious intention, and then replaces the same amount of approximated operations with precise ones to compensate the reduced accuracy. Thus, when one performs functional testing, the average accuracy of the tampered AC system remains as same as its original way. However, when the tampered AC system is deployed in practical applications, the system operation will suffer from ECA. For example, a neural network is adopted to recognize the handwritten number shown in Fig. 5(b). In the original design, the pixels in the red region (carrying critical features) are analyzed by precise computation and other pixels (carrying non-critical feature) are processed by approximate functions. The proposed ECA could alter the extraction of critical features, although the average accuracy

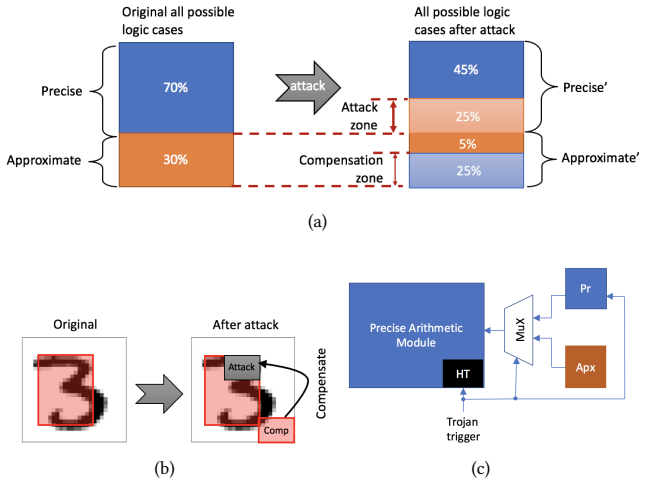


Figure 5: Proposed Error Compensated Attack (ECA). (a) Concept, (b) illustration of an example, and (c) hardware Trojan design for ECA implementation.

obtained by the neural network still matches the system specification. Figure 5(c) shows one possible implementation for ECA, in which the hardware Trojan (HT) changes the accuracy of the target area by switching the precision to approximate submodule (or vice versa).

ECA is stealthy in nature because of error compensation and the low triggering probability of hardware Trojans. Functional testing is typically used to assess the impact of Trojans on the primary output of the victim modules. In our analysis, we used an 8-bit approximate adder, in which seven are 1-bit precise full adders and one is an 1-bit approximate full adder, to estimate the error rate of that hybrid adder with a hardware Trojan inserted. Random test vectors were applied in the testbench simulated in a Cadence NC-Verilog simulator. As shown in Table 2, the triggered hardware Trojans indeed incur more error cases on the primary outputs. However, for a given 50 random test cases, ECA reduces the error rate from 32% to 12%, which is even lower than that of the Trojan free case. This means, if a user relies on the average error rate to detect hardware Trojans in approximate computing systems, he or she may overlook the attack since the error rate of ECA does not exceed the declaimed average error rate 24%. This conclusion also applies to 150 random test cases. This experiment proves that ECA is stealthy from a user's point of view. Next, we applied ECA to the Sobel edge detection. As shown in Fig. 6, ECA successfully compensates the reduced number of edges due to HTs. The effect of error compensation varies with input images. The image Baboon shows the best fit for ECA. We envision that ECA will challenge the security of AC systems, as attackers can hide the Trojan effect in the average error rate.

4 PROPOSED METHOD: BLURRING BOUNDARIES TO OBSCURE AC SYSTEMS

The practical attacks demonstrated in Section 2 and the advanced attacks projected in existing literature [17, 18] and our Section 3 all indicate that explicit exposure of the boundary between the AC and non-AC modules (hereafter, we refer it to Approx-Precise Boundary

Table 2: Error rate comparison of Trojan free, Trojan inserted and ECA hybrid adders.

Classification	Random Test Cases					
	50			150		
	HTFree	HT	ECA	HTFree	HT	ECA
Num. triggered HT	0	10	10	0	38	38
Num. error cases	12	16	6	41	57	22
Error rate	24%	32%	12%	27.3%	38%	14.66%

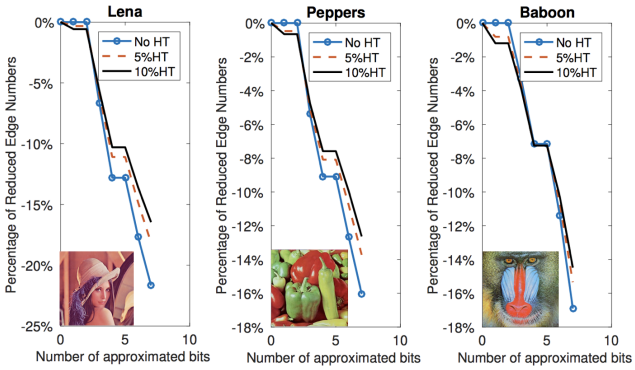
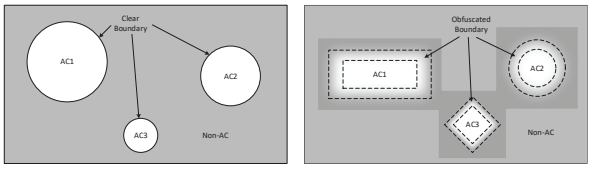


Figure 6: Impact of error compensation HT on the number of edge detection based on the Sobel algorithm.

(APB)) leaves adversary new attack surfaces to intrude the computing system. Thus, we propose to obfuscate AC systems by blurring APB such that the attacks from AC systems' rogue users can be thwarted. As depicted in Fig. 7(a), the non-obfuscated AC modules AC1, AC2, and AC3 distinct themselves from the non-AC modules with an explicit boundary. In contrast, after our obfuscation method is applied, the AC modules are not only mingled into the non-AC modules but they are also presented in different shapes from the AC system users' point of view. In Fig. 7(b), the areas highlighted in dashed lines stand for the obfuscated APB, which impede attackers from precisely locating the AC components before they implement their targeted attacks. Note that the shape changing means, when adversary try to identify APB on our obfuscated code, they will mistakenly include some non-boundary part and also miss some real boundaries. In general, our boundary blurring method obfuscates an AC system with one of the following three schemes: (1) *obfuscating when to approximate (WNA)*, (2) *obfuscating what to approximate (WTA)*, and (3) *obfuscating how to approximate (HWA)*. Figures 8, 9 and 10 depict the key idea of each scheme, respectively. The object of our obfuscation includes preliminary inputs, internal signals, processing logic, and the combination of three above. The WNA scheme aims for primary inputs, the WTA scheme protects internal signals from being successfully tampered, and the HWA scheme strengthens the entire AC system by hardening the processing logic and the connection between submodules.

4.1 WNA Scheme: Obfuscating When to Approximate

To save power consumption, a submodule M_{k+1} for some precise operations in a computing systems is replaced with its approximate version M_{k+1}'. As there is data exchanging between M_{k+1}' and other precise submodules, adversary could breach the system through



(a) (b)

Figure 7: The comparison of an approximate computing system (a) before and (b) after our proposed blurring approximate-precise boundaries method. Note that AC and non-AC components are in white and gray, respectively.

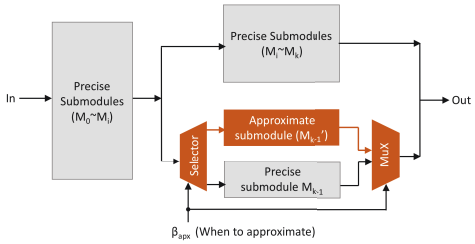


Figure 8: Blurring APB with WNA scheme.

that approximate submodule. The proposed WNA scheme shown in Fig. 8 brings the precise implementation of M_{k+1} back to the computing system and then uses a control signal β_{apx} to enable the toggling between M_{k+1} and M_{k+1}'. As long as β_{apx} is secretly controlled by the legitimate system designer, the WNA scheme can reduce the attack success rate. Depending on the percentage of approximate submodules in the entire computing system, the new attack success rate will be close to the product of system approximation rate and β_{apx}.

4.2 WTA Scheme: Obfuscating What to Approximate

Due to the accessibility of primary inputs, it is easier for adversary to bypass the obfuscation utilized on the primary inputs than those applied to internal signals. The proposed WTA scheme selectively obfuscates internal signals. Assume the entire computing system takes N phases to produce the final output. Our WTA scheme first shuffles the internal signals after phase j and then employs a masking vector $V_{masking}$ to mute a portion of the internal signals before de-shuffling. The abstract view of WTA is shown in Fig. 9. Due to the muting operation, several internal signals remain constant and thus the corresponding precise submodules triggered by those muted signals do not consume switching power. Since the masking operation controlled by $V_{masking}$ is applied between shuffling and de-shuffling, the masking vector can be considered as an obfuscated input as long as the shuffling algorithm is not public to AC system users. In general, a longer masking vector $V_{masking}$ achieves more reduction on the attack success rate. To ensure the effectiveness of the WTA scheme, the shuffling algorithm should introduce as much unpredictable non-linearity as possible. Logic locking techniques can be used to enhance the attack resilience of the proposed shuffling and masking operations.

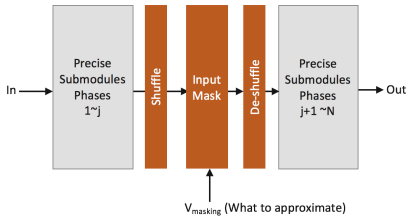


Figure 9: Blurring APB with WTA scheme.

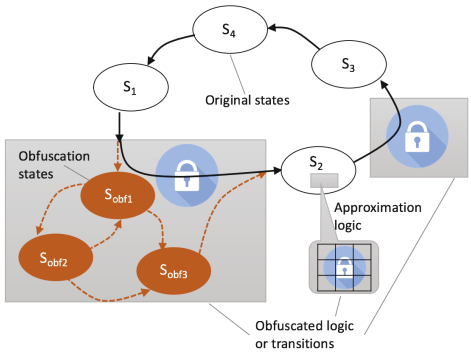


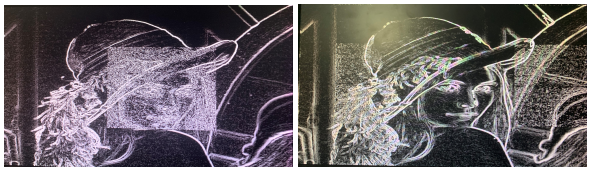
Figure 10: Blurring APB with HWA scheme.

4.3 HWA Scheme: Obfuscating How to Approximate

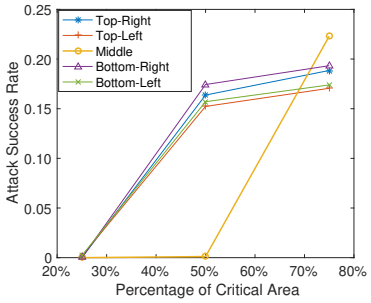
In the HWA scheme shown in Fig. 10, we obfuscate the state transition (e.g., from S1 to S2 and from S2 to S3) with a key vector, preventing unauthorized users from accessing the state that handles approximate operations. Incorrect key vectors will lead the system to enter obfuscation states, which cause system malfunctions. In addition, the truth table for approximation logic (e.g., for S2) is re-designed by integrating a key vector. This type of obfuscation extends the logic network by adding a unique 'nonce' to the original approximation and thus introducing some designer-controlled 'dummy logic'. Compared to WNA and WTA schemes, HWA is the most powerful obfuscation in our boundary blurring based obfuscation method. We assume that the approximate system utilizes N_{Sorig} original states, among which $\eta \cdot N_{Sorig}$ are assigned to be the states for approximation. Assume η is the percentage of approximation states. If the size of the applied key vector is K and the number of obfuscation states is N_{Sobf}, the attack success rate after HWA blurring boundary can be estimated with Eq. (3).

$$\Gamma_{howobf} = \prod_{j=1}^{\eta \cdot N_{Sorig}} \left(\prod_{i=0}^{N_{Sobf}} \left(\frac{1}{2^{K-i}} \cdot \gamma_{logic_i} \right) \right) \tag{3}$$

In which, the term γ_{logic_i} is the coefficient of logic masking associated with the approximate state. This parameter will vary with the specific logic function obfuscated by the key vector. More logic masking effect achieved by the encryption key will produce a lower attack success rate. The increasing key size and the number of obfuscation states collaboratively contribute to reducing the success rate of attacks.



(a) (b)



(c)

Figure 11: The output of approximate edge detection hardware module implemented in FPGA (a) before and (b) after obfuscation to resist the regionally targeted attack on the approximation submodule. (c) Reduced attack success rate achieved by our method.

5 EXPERIMENTAL RESULTS

The proposed WTA scheme was applied to the approximate Sobel edge detection application. We assume that hardware tampering in the Sobel application happened after the image pixels were downloaded to the dedicated memory on the FPGA board shown in Fig. 1(a). If the attacker knows the starting memory address of the image to be processed, it would be likely to predict the memory zone for critical image pixels. The proposed shuffling in WTA obfuscated the data sequence stored in the memory by altering the linear relation between the memory index and the order of image pixels. We first thoroughly analyzed the information processing flow in the Sobel edge detection and then applied a suitable non-linear shuffling algorithm (so that the shuffling does not affect the correctness of the subsequent edge detection operations). A static masking vector was applied to determine the portion for approximate computing.

We compared the outputs of edge detection before and after our WTA scheme. As shown in Fig. 11(a), the attack makes the Lena face vague if no shuffling is applied. In contrast, as the WTA scheme obfuscates the memory index by shuffling and masking, our method successfully thwarts the attack and pushes the vague pixels to the non-important image zone shown in Fig. 11(b). To comprehensively evaluate the attack success rate after WTA was applied to the Sobel edge detection, we varied the percentage of the critical pixel zone in the entire image and the attack locations. As shown in Fig. 11(c), our method reduces the attack success rate to 0% if the critical image zone does not exceed 25% of the entire image. When the critical image feature is in the middle of the image, our method can completely thwart the attack even if 50% of the image carries important features. As the percentage of critical pixels increases, the performance of WTA starts to degrade and other obfuscation techniques should be used to further improve the attack resilience.

6 CONCLUSION

Approximate Computing (AC) techniques trade accuracy for performance improvement and energy efficiency, being increasingly attractive in various computation-intensive applications. Unfortunately, the unique approximate behavior and computational uncertainty in AC systems expose plenty of new attack opportunities. Two case studies provided in this work demonstrate the impact of hardware Trojan attacks on the performance of AC systems for artificial neural network and image edge detection. We further project an error compensated attack, which is unique and unexplored attacks on AC systems. In this work, we highlight that the main cause of attacks on AC systems is the explicit boundary between approximate and precise modules. Furthermore, we propose to blur the explicit boundary with three obfuscation schemes, which obscure when, what and how to approximate in the emerging AC systems. Our FPGA emulation confirms the effectiveness of the proposed obfuscation method. In the Sobel edge detection application, our WTA scheme can successfully thwart the attack and reduce the attack success rate to 0% if the critical image zone does not exceed 25% of the entire image.

REFERENCES

[1] W. Baek and C. Trishul. 2010. Green: A Framework for Supporting Energy-Conscious Programming using Controlled Approximation. In *Proc. PLDI'10.* 198–209.
[2] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. 2013. Low-Power Digital Signal Processing Using Approximate Adders. *TCAD'13* 32, 1 (Jan 2013), 124–137.
[3] L. Ceze D. Burger H. Esmaeilzadeh, A. Sampson. 2012. Architecture Support for Disciplined Approximate Programming. In *Proc. ASPLOS'12.* 301–312.
[4] J. Han. 2016. Introduction to approximate computing. In *Proc. VTS'16.* 1–1.
[5] M. Jung, D. M. Mathew, C. Weis, and N. Wehn. 2016. Invited: Approximate computing with partially unreliable dynamic random access memory — Approximate DRAM. In *Proc. DAC'16.* 1–4.
[6] S. Keshavarz and D. Holcomb. 2017. Privacy leakages in approximate adders. In *Proc. ISCAS'17.* 1–4.
[7] D. S. Khudia, B. Zamirai, M. Samadi, and S. Mahlke. 2016. Quality Control for Approximate Accelerators by Error Prediction. *Proc. D&T'16* 1 (Feb 2016), 43–50.
[8] S. Mittal. 2016. A Survey of Techniques for Approximate Computing. *Proc. CSUR'16* 48 (2016), 62:1–62:33.
[9] A. Rahimi, L. Benini, and R. K. Gupta. 2013. Spatial Memoization: Concurrent Instruction Reuse to Correct Timing Errors in SIMD Architectures. *Proc. TCAS II'13* 12 (Dec 2013), 847–851.
[10] F. Regazzoni, C. Alippi, and I. Polian. 2018. Security: The Dark Side of Approximate Computing?. In *Proc. ICCAD'18.* 1–6.
[11] A. G. M. Strollo and D. Esposito. 2018. Approximate computing in the nanoscale era. In *Proc. ICICDT'18.* 21–24.
[12] Y. Tian, Q. Zhang, T. Wang, F. Yuan, and Q. Xu. 2015. ApproxMA: Approximate Memory Access for Dynamic Precision Scaling. In *Proc. GLSVLSI'15.* 337–342.
[13] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan. 2014. AxNN: Energy-efficient neuromorphic systems using approximate computing. In *Proc. ISLPED'14.* 27–32.
[14] O. R. Vincent and O. Folorunso. 2009. A Descriptive Algorithm for Sobel Image Edge Detection. In *Proc. InSITE'09.*
[15] Q. Xu, T. Mytkowicz, and N. S. Kim. 2016. Approximate Computing: A Survey. *D&T'16* 33, 1 (Feb 2016), 8–22.
[16] L. Yang and B. Murmann. 2017. SRAM voltage scaling for energy-efficient convolutional neural networks. In *Proc. ISQED'17.* 7–12.
[17] P. Yellu, N. Boskov, M. Kinsy, and Q. Yu. 2019. Security Threats on Approximate Computing Systems. In *Proc. GLSVLSI'19.* 387–392.
[18] P. Yellu, M. R. Monjur, T. Kammerer, D. Xu, and Q. Yu. 2020. Security Threats and Countermeasures for Approximate Arithmetic Computing. In *Proc. ASP-DAC'20.* 259–264.
[19] P. Yellu and Q. Yu. 2020. Can We Securely Use Approximate Computing?. In *ISCAS'20, Forthcoming.* Seville, Spain.
[20] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu. 2015. ApproxANN: An approximate computing framework for artificial neural network. In *Proc. DATE'15.* 701–706.
[21] N. Zhu, W. L. Goh, and K. S. Yeo. 2009. An enhanced low-power high-speed Adder For Error-Tolerant application. In *Proc ISIC'09.* 69–72.