Hands-On Teaching of Hardware Security for Machine Learning

Ashley Calhoun, Erick Ortega, Ferhat Yaman, Anuj Dubey, Aydin Aysu
Department of Electrical and Computer Engineering
North Carolina State University, Raleigh, NC, U.S.A
{ancalhou,eortega,fyaman,aanujdu,aaysu}@ncsu.edu

ABSTRACT

Hardware security for machine learning (ML) and artificial intelligence (AI) circuits is becoming a major topic within the cybersecurity framework. Although much research is ongoing on this front, the community omits the educational components. In this paper, we present a training module comprised of a set of hands-on experiments that allow teaching hardware security concepts to newcomers. Specifically, we propose 5 experiments and related training material that teach side-channel attacks and defenses on the hardware implementations of neural networks. We report the organization and the findings after testing these experiments with sophomore undergraduate students at North Carolina State University. The students first study the basics of neural networks and then build a neural network inference circuit on a breadboard. They then conduct a differential power analysis attack on the hardware to steal trained weights and a circuit-balancing (hiding) style defense to mitigate the attack. The students develop all related hardware and software codes to perform attacks and build defenses. The results show that such complex notions of digital circuits design, neural networks, and side-channel analysis can be instructed at the sophomore level with a well-thought set of experiments. Future extensions could include establishing an online infrastructure for remote teaching and efficient scaling to a broader audience.

CCS CONCEPTS

• **Security and privacy** → **Security in hardware**; **Side-channel analysis and countermeasures**; • **Applied computing** → **Education**.

KEYWORDS

education, hardware security, neural networks, side-channels

ACM Reference Format:
Ashley Calhoun, Erick Ortega, Ferhat Yaman, Anuj Dubey, Aydin Aysu. 2022. Teaching the Next Generation of Cryptographic Hardware Design to the Next Generation of Engineers. In *Proceedings of the Great Lakes Symposium on VLSI 2022 (GLSVLSI '22), June 6–8, 2022, Irvine, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3526241.3530828

GLSVLSI '22, June 6–8, 2022, Irvine, CA, USA

ACM ISBN 978-1-4503-9322-5/22/06…$15.00
https://doi.org/10.1145/3526241.3530828

1 INTRODUCTION

Hardware security is an emerging field in the broader cybersecurity framework. This field focuses on how to establish trust in hardware. Indeed, trusted computing in hardware is fundamental to information security practices. If the hardware, which typically forms the root of trust, is compromised, security mechanisms at higher abstraction levels such as software will also likely fail. Outsourced semiconductor fabrication, counterfeit digital chips in the supply chain, hardware backdoors, leaks in hardware through side-channels, and other attacks on hardware have triggered a significant research activity in hardware security over the past decade [2, 17].

The research in hardware security has been traditionally studying cryptographic systems given the importance cryptography plays in cybersecurity. However, in recent years, with the rise of machine learning (ML) and artificial intelligence (AI) applications, there is a growing interest in the hardware security aspects of AI/ML systems. [20]. For example, a recent line of research has started exploring how the electromagnetic (EM) and power consumption-based side-channel attacks/defenses extend from cryptographic workloads to AI/ML applications [1, 7, 8]. Others have likewise extended other threat vectors to AI/ML such as hardware fault attacks [10], hardware Trojans [4], hardware IP theft [14], and cold boot attacks [19].

Despite the visible change in research focus toward hardware security of AI/ML systems, the educational aspects have been unfortunately omitted so far. Teaching these topics is critically important to raise the next-generation science, technology, engineering and math (STEM) workforce. It was estimated that cybersecurity job openings are about 3.5 million [3] and the number is growing due to the talent gap. Some of those jobs will inevitably be related to the security aspects of AI/ML applications given their widespread and growing use-cases in a range of domains.

In this paper, we propose a training module—a set of experiments/assignments along with related training material—that allows teaching the complex hardware security and machine learning topics as early as the sophomore undergraduate level. We designed four hands-on experiments that act as a step-by-step guide to practice fundamentals of logic design, implementing neural network circuits, conducting side-channel attacks, and building defenses for mitigation. These experiments have been conducted by undergraduate sophomores over a span of 6 months. Upon completion, the students have shown a working understanding of hardware design, security risks, and related protections of neural network implementations. Our study, therefore, shows that such complex topics can be taught at the sophomore level and even potentially before.

The rest of the paper is organized as follows. Section 2 describes the preliminaries needed for this work. Section 3 gives an overview of the experiments, organization, and the design rationale behind them. Section 4 provides the details of the experiments and the results. Section 5 discusses the challenges and potential approaches for resolution, and Section 6 concludes the paper.

2 PRELIMINARIES

This section introduces the basics of neural networks and side-channel analysis.

2.1 Basics of ML/AI: Neural Networks

A neural network (NN) is a trainable, deterministic mathematical function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that can be used for classification tasks. For a classification task, $\mathcal{Y}$ is a categorical set, e.g., a collection of dog breeds, and $x \in \mathcal{X}$ can be a dog photo. *Training* and *inference* are the two modes of operation of neural networks. Training allows the network to adjust its functional form based on *examples* it has seen. An example is a pair (x, y), where x is called the *input* and y is called the *label*. NNs gradually adapt to a functional form by a learning algorithm that makes the calculated label $\hat{y} = f(x)$ as close to the true label y as possible for all $x \in \mathcal{X}$. The inference is the process of calculating the label for an input x.

A trained NN model contains parameters such as weights and biases, which define the function $f : \mathcal{X} \rightarrow \mathcal{Y}$. NNs execute a sequence of linear operations (*e.g.,* multiply-accumulate) using weights and biases, and they also perform non-linear operations such as a Rectified Linear Function based on the chosen network topology. This work focuses on Binarized Neural Networks (BNNs) [11] given their conceptual simplicity and edge-device friendly nature. In a BNN, all weights and linear operation results are binarized (to ±1), reducing the memory overheads and simplifying floating-point multiplications operations to XNORs.

2.2 Side-Channel Security

Rather than applying a direct, mathematical attack, side-channel attacks extract secret information computed inside the device through side information such as power consumption [12]. These attacks evaluate the statistical correlation of the predicted secret-dependent operations to the side information leaking from the device. The differential power analysis (DPA) is a common side-channel attack. In DPA, the typical assumption is for the adversary to have physical access and to observe/send multiple inputs/outputs and record corresponding side information for estimating the correct secret with sufficient statistical evidence. Previous works have shown that cryptographic devices are vulnerable to such attacks unless they employ a countermeasure [13].

2.3 Side-Channel Countermeasures: WDDL

Numerous works over the past two decades have explored building effective countermeasures against physical side-channel attacks [15]. Every countermeasure aims to minimize the correlation of power/EM leakage of the target with the data processed in the target. These countermeasures can be classified but not restricted to be based on either masking or hiding. Masking transforms the original function to manipulate random values called secret shares instead of directly manipulating the secret [9, 16]. Hiding aims to equalize the power consumption in the target during the entire computation [18, 21].

For this work, we use a hiding defense because of two reasons. First, it is conceptually simpler for undergraduate level compared to masking, which requires mathematical background. Second, unlike masking, it does not require PRNGs, making breadboard implementations easier. We specifically choose the Wave Dynamic Differential Logic (WDDL) style of hiding defense [18]. In a WDDL circuit, there exists a complimentary net for each net of the original circuit making the logic differential—every 0–>1 transition is accompanied by a 1–>0 transition, and vice versa. Differential logic alone is not fully secure because some transitions continue to remain distinguishable, for instance, 0–>0 and 0–>1. Thus, it is combined with dynamic logic which precharges every net to a constant in before computation in each cycle.

3 TRAINING FLOW AND OVERVIEW

This section provides a high-level overview about the organization of our training module.

3.1 Learning Objectives

The primary objective of our experiments is to teach side-channel attacks on neural networks and corresponding defenses at the circuit-level design. Doing so requires a deep understanding of hardware design, side-channel analysis, and neural network algorithms. We thus set the learning objectives as follows.

- Analyze the details of neural networks and study their hardware implementation.
- Apply circuit simulation and verification techniques using electronic design automation tools and Verilog hardware description language.
- Practice and demonstrate an actual digital circuit implementation on a breadboard.
- Understand the foundations of side-channel secret extraction and execute it on a real target.
- Explore power-balancing side-channel countermeasures at the circuit-level and quantify their effectiveness.

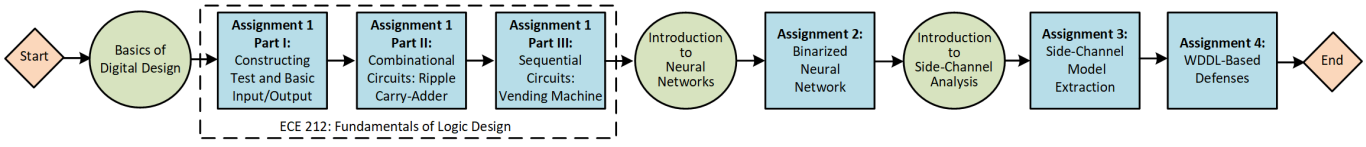


Figure 1: Flow of our training module. The first part consists of understanding the basics of digital design and implementation—this can be also done as part of an undergraduate course. The students are then guided through the implementation of a neural network inference hardware, conducting side-channel attacks, and building defenses with Assignments 2, 3, and 4, respectively.

Table 1: Potential Schedule for Our Training Module

Day	Hours	Topics
1 (Mon.)	09–12AM	Lecture: Hardware Design Basics
1 (Mon.)	12:30–5PM	Hands-on: Implementing Ripple Carry-Adder
2 (Tue.)	09–12AM	Lecture: HDL and Verilog
2 (Tue.)	12:30–5PM	Hands-on: Implementing Vending Machine
3 (Wed.)	09–12AM	Lecture: Neural Networks Fundamentals
3 (wed.)	12:30–5PM	Hands-on: Implementing BNN
4 (Thu.)	09–12AM	Lecture: Side-Channel Attacks (DPA)
4 (Thu.)	12:30–5PM	Hands-on: Implementing (DPA)
5 (Fri.)	09–12AM	Lecture: Side-Channel Defenses (WDDL)
5 (Fri.)	12:30–5PM	Hands-on: Implementing WDDL

3.2 The Training Structure

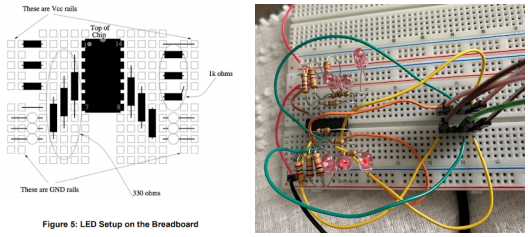
Figure 1 outlines the overall flow of our training module. We have organized four assignments/experiments and related training material that follows sequential steps. First, the students learn about the basics of digital circuit design. Then, they implement the first exercise which has three parts: (1) blink an LED on the breadboard, (2) design and implement a simple combinational circuit design, (3) and a sequential circuit design. After completing this experiment, the students will be able to implement any given circuit on the breadboard and simulate it to verify functionality both on a Verilog module as well as on the real hardware. The second experiment is about implementing a BNN in hardware. This allows students to grasp the details of neural network implementation. The third experiment is on attacking the neural network built in the second experiment with a DPA-style attack. The fourth experiment is on building a WDDL-style defense and quantifying the reduction in side-channel leakage. Section 4 provides the details of these experiments.

3.3 Student Background and Schedule

Our training module targets students with a basic understanding of logic design but does not require a background in other related concepts such as cryptographic engineering, or neural network design, We have hired five REU students to carry out the experiments. These students were chosen from the participants of an undergraduate sophomore class on logic design.

The students spend about 3-5 hours each week on average (self-reported). There were minimal interactions with the instructor. Over the summer, the students prepared a monthly report upon completing their assignments and met with the instructor once a month to present the details of their report in 30 minutes. All five students were able to finish the first two parts of the first experiment, and 3 of them were able to complete the last part. Over the fall semester, 2 of these students were re-hired to complete the last three experiments with the same time commitment.

We estimate that with intensive hands-on supervision, the entire experience can be compressed into a single week and be taught at a summer camp to early-phase undergraduates. Table 1 demonstrates a possible schedule of events. The first two days can be spent teaching fundamentals and completing the first experiment. The last three days could be sufficient to cover the rest by completing one experiment each day.



(a) Fig 1 (b) Fig 2

Figure 2: Schematic and circuit construction of LED setup

4 ASSIGNMENT DEVELOPMENT & RESULTS

This section describes the four assignments created to enable the hands-on teaching of hardware security for machine learning.

4.1 Assignment I: Introduction to Circuit Building Basics

The objective of the first assignment is to teach the basics of digital circuit design and implementation. Indeed, the students need to study many basic aspects before they can understand specifics of hardware security for machine learning and artificial intelligence.

We choose the participants enrolled in the electrical and computer engineering program. This program already covers the basic logic design concepts as part of a sophomore undergraduate required course. The course material including a guided walk-through of Modelsim software is sufficient to cover the basics.

4.1.1 Assignment 1 Part 1. This part includes building simple circuit functions using breadboards, wires, battery packs, voltage regulators, integrated chips for basic Boolean operators, resistors, and LEDs. Consequently, the students learn how to properly connect a power source, apply input/output to the breadboard, and construct simple combinational circuits with a logic verification test. The first step of Part 1 is to establish power rails on the breadboard using a power supply. The students used a 4 D-Cell battery pack as the power supply and regulated it using the LM7805 voltage regulator. Next, the students replace the battery pack with the power outputs of Analog Discovery 2 (AD2). AD2 also orchestrates the events in the experiments such as sending/receiving data from the breadboard and starting capture on the oscilloscope. It communicates to the breadboard via Analog Discovery Toolkit.

Next, the students learn to place LEDs and resistors on the breadboard. They test the circuit via AD2 Static I/O waveforms and LEDs [6]. This step requires studying the datasheets for IC chips that implement simple Boolean gates (AND, OR, Inverter, etc.), placing them accordingly into the breadboard, and individually testing them. The LEDs are helpful in debugging the circuit in later assignments by probing specific points. The AD2 device comes with 30 color-coded connections. The students learn how to use the AD2 waveforms software after making the appropriate circuit connections. This involves using the Static I/O, Patterns, and Logic Analyzer windows of the software. The students create a binary bus in the Patterns window to verify the design. Figure 2 shows the resulting schematic and the circuit on breadboard.

4.1.2 Assignment 1 Part 2. The objective of this part is to use the previously tested basic Boolean logic gates and build a (relatively) complex combinational circuit design. The students will learn how

to design, verify, and implement circuits using logical gates, which is critical in gaining confidence for the later assignments.

Specifically, the students implement a two-bit ripple-carry adder in this part. The two-bit ripple-carry adder consists of two cells. One cell uses traditional logic gates and the other uses 4:1 multiplexers. The students first design the circuit on paper and then verify the design in Verilog using ModelSim software. The students are instructed to draw formal schematics for the circuits rather than rough hand-drawn figures. Next, the students test the circuit on breadboard using the binary bus feature of AD2 software.

4.1.3 Assignment 1 Part 3. The objective of Part 3 is to design, test, and implement a sequential circuit on the breadboard. Specifically, the goal is to realize a vending machine controller as a Mealy state machine. The vending machine has to meet specific requirements: it stocks two items (A & B, where A costs 5 cents and B costs 10 cents), only accepts nickels up to 15 cents, and gives back the change along with the requested item if the amount exceeds selection.

The first step is to assign the states for the finite state machine (FSM). After designing the FSM, the assignment asks to develop the symbolic state table, the state-encoded transition table, and construct the K-Map to design the schematic. The students are asked to minimize the number of states and exploit the "don't care" conditions. The students use the resulting schematic to know the IC chips they need to implement the design on a breadboard. The resulting design uses 2 D flip-flops and several AND, OR, and INV gates.

Finally, the students test the state machine for its correctness through the AD2 waveforms software. They use the patterns window on the AD2 to supply the clock and two inputs. The students add these signals along with the outputs to the Logic window and verify the correct functionality of the circuit.

4.2 Assignment II: Implement a Neural Network Inference Circuit

The second assignment teaches the students how to design the hardware for a simple neural network classifier. Specifically, they build a BNN hardware inference on breadboard [5]. Learning how to design such hardware enables them to execute effective attacks and implement related mitigation techniques in the next assignments.

The assignment includes an initial lecture on understanding the basics of BNN, and further learnings through educational videos and doubt-clearing sessions. The students follow the general guideline of drawing the schematic first, followed by Verilog-based modeling, and finally the functional verification using Modelsim simulations. Each step is closely monitored by the instructors to ensure that the students are on the right track.

The BNN consists of an input layer with four nodes, a single hidden layer with one node, and an output layer with one node. Figure 3 shows the schematic of the BNN circuit and Figure 4 shows the corresponding breadboard circuit. XNOR gates multiply 1-bit inputs with the binarized weights in the input layer. The students implement the weighted-summation using XNOR-POPCOUNT, which is typical for BNNs to save area, power, and memory. Next, the non-linear sign function extracts the sign of the summation and stores it in a D flip-flop. The hidden and the output layer also compute

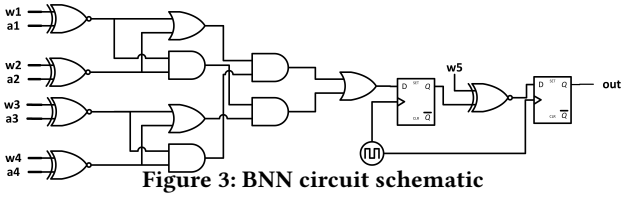


Figure 3: BNN circuit schematic

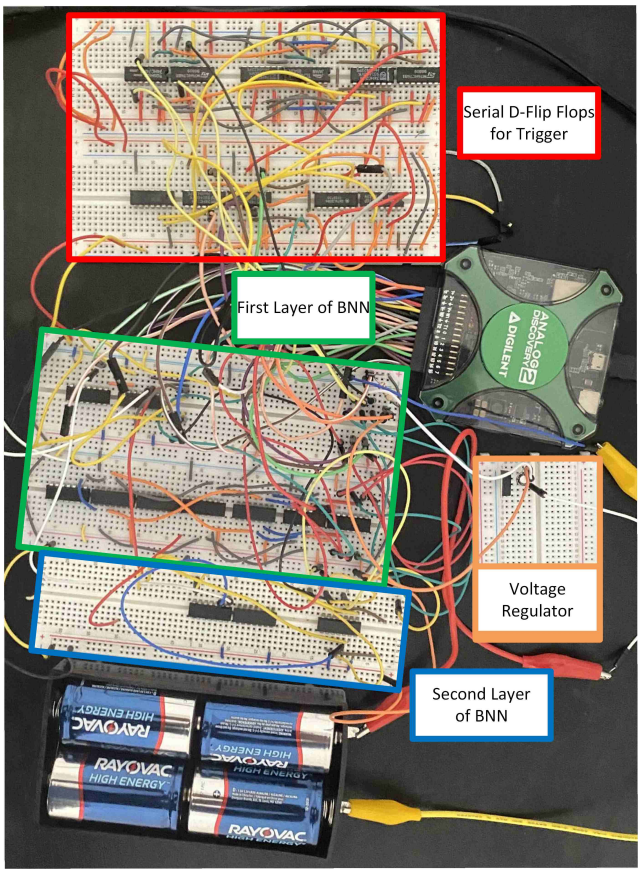


Figure 4: BNN breadboard and attack setup

the summations in a similar way using the weights corresponding to those layers to finally generate the inference result. The students use LEDs to functionally verify the final output from the breadboard.

4.3 Assignment III: DPA of BNN Hardware

The objective of this assignment is to conduct a DPA attack on the BNN hardware designed in Assignment II to steal the valuable trained neural network weights. Indeed, such attacks are of growing concern given the high costs of ML training [7, 8]. The successful completion of this task allows the students to practice state-of-the-art attack techniques.

The assignment asks following the standard practices in DPA: apply random inputs to the target circuit (BNN in this case), record power measurements during computations with an oscilloscope, and finally extract the secret by correlating a power model with the recorded power measurements. The students first create a side-channel evaluation setup. They use AD2 to send/receive inputs/outputs to/from the BNN circuit and to also measure power.

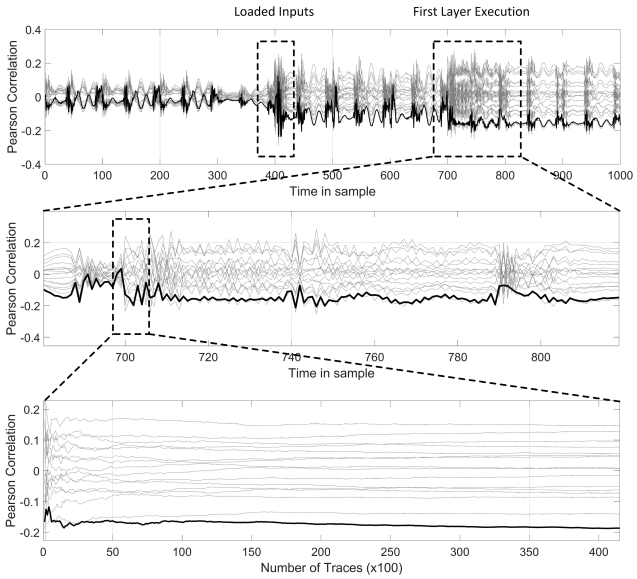


Figure 5: DPA of BNN. Full trace (top), Pearson correlation scores of weight guesses (middle). Last chart shows number of required traces for the correct key guess.

They automate the communication and subsequent power capture by writing a script in JavaScript for AD2. Figure 4 shows the picture of this entire setup along with the BNN inference circuit under attack.

The students developed a trace acquisition script that triggers oscilloscope and simultaneously sends the randomized inputs to the breadboard. The script sends a zero input between two measurements to reset flip flops between measurements. The BNN circuit registers the asynchronous inputs sent from AD2 using flip-flops for proper alignment in the captured power traces. To ensure that the oscilloscope is triggered before loading of inputs, the circuit adds more flip-flops in the input path compared to the trigger path. The complete inference finishes in 2 clock cycles. Design performs 4 XNOR operations between input activations and weights and a subsequent POPCOUNT in the first cycle. Then, it executes one more XNOR operation between the first layer activation and the weight of the second layer in the second cycle. D flip flops store the output of the first and second cycles.

Next, the students conduct the DPA. They use the hamming weight (HW) power model for the attack because their setup resets the design between two measurements. The assignment prescribes them to use the Pearson's correlation coefficient as the side-channel evaluation metric. First, they compute the correlation between the input power model (HW of inputs) and the power traces. A successful input correlation test validates the correctness of the measurement setup because it confirms the loading of inputs in the design. Second, the students target the flip flops of the first and second layer for the DPA attack. They hypothesize on the respective weights and compute the correlation between the power model of the registers for each weight guess and the power trace. The first layer contains four weights, which results in 16 possible hypotheses. The second layer only contains one weight and thus results in two possible hypotheses. The following equations present the Boolean

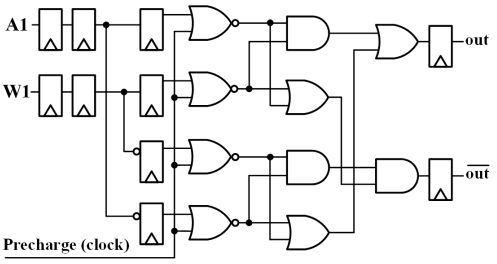


Figure 6: Circuit schematic using the WDDL technique

equation of the target hidden layer registers.

$$o_1 = \overline{a_1 \oplus w_1}, o_2 = \overline{a_2 \oplus w_2}, o_3 = \overline{a_3 \oplus w_3}, o_4 = \overline{a_4 \oplus w_4} \quad (1)$$

$$HLOut_0 = (o_1 \cdot o_2) \cdot (o_3|o_4) + (o_2 \cdot o_4) \cdot (o_1|o_2) \quad (2)$$

The students recover the hidden layer weights using DPA. They also plot the evolution of the correlation coefficient with the number of measurements. This plot helps them to find the number of measurements needed to find the hidden weights with a statistically reasonable confidence, as show in Figure 5.

4.4 Assignment IV: Hiding Defenses for Neural Network Side-Channels

The objective of the fourth assignment is to implement a countermeasure on the BNN against DPA. The successful completion of this assignment leads to a deep understanding of state-of-the-art defenses. The defense follows WDDL technique (see Section 2.3). WDDL reduces the side-channel leakage by precharging each net in every clock cycle and by adding complementary logic paths for every original logic path. This technique equalizes the power consumption throughout the computations performed in the circuit and thus, hinders the attacker's ability to analyze the power consumption to extract the secret, which is the BNN weights.

The assignment focused on hardening the XNOR operation since, in a BNN, the multiplications are equivalent to an XNOR operation. The WDDL XNOR gate complementary part was formulated via the DeMorgan's Law resulting in the function $a'|b' \cdot a|b$. For each original input, the WDDL circuit also requires its complement. Therefore, the assignment instructed the students to design a preparation circuit before feeding the inputs to the WDDL circuit.

The students also implemented the precharge logic at the input. The assignment allowed exploring circuit designs for WDDL that resulted in implementing precharge and evaluation phases in the same cycle using clock signals as precharge. This approach is easier to understand and implement compared to others like precharge and evaluation in consecutive clock cycles with Master-Slave DDL flip flops [18]. The students used a NOR gate to precharge the inputs by connecting one of the NOR inputs to the clock. The NOR-based precharge ensures that every time the clock is high all the inputs are precharged to zero and the circuit evaluates the outputs when the clock is low. Figure 6 depicts the resulting schematic.

The assignment asks the students to implement the WDDL circuit with careful consideration. They implemented both paths on identical breadboards and placed logic gates at the exact same location on each breadboard to achieve close electrical properties in the paths. They also used jumper wires of similar lengths to again ensure a minimum difference between the paths.

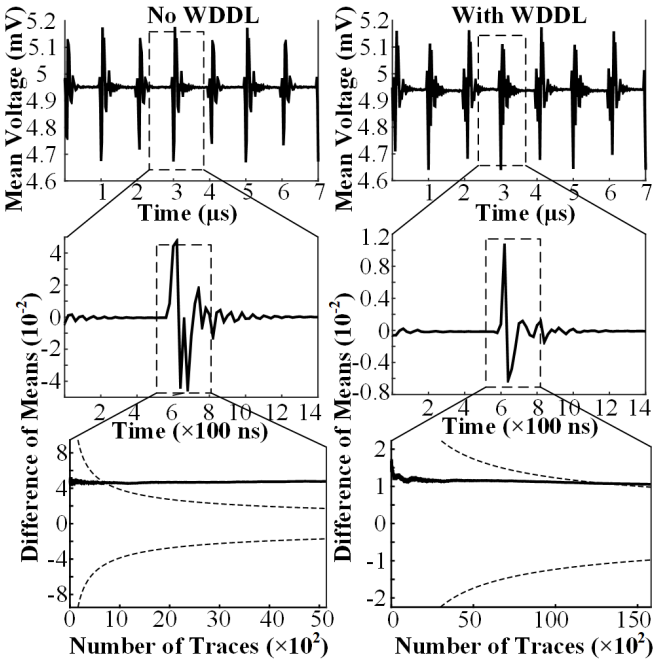


Figure 7: Mean power trace (top), DoM trace (middle), and the evolution of DOM with number of measurements (bottom) for unprotected (left) and WDDL-protected circuits (right). The number of traces needed to cross the 99.99% confidence intervals increase from 1k to 15k–an increase of 15×.

After verifying the functionality of the Verilog code and then the breadboard implementation, the assignment proceeds to instruct the students to verify the results with a security test. To that end, the assignment dictates the common difference-of-means (DoM) metric. The goal is to demonstrate that the number of traces needed to get a statistically significant DoM is more for the secure circuit, compared to the baseline insecure circuit. Figure 7 (left) and (right) shows the attack on the unprotected and protected designs, respectively. Moreover, it shows the mean traces (top), the DoM plot during the targeted operation (middle), and the evolution of DoM with respect to the number of measurements (bottom). The results show a reduction of 5× in the DoM of the WDDL protected circuit and a further improvement of 15× when comparing the number of traces needed for statistical significance.

5 EXPERIMENT OUTCOME AND ANALYSIS

5.1 Student Participation and Interest

We have tested the proposed training with five undergraduate students. These students are chosen from the participants of the "ECE 212: Fundamentals of Logic Design" course in Spring 2021. All students were engaged during the experiments.

5.2 Final Results

We gave three months for students to complete the experiments who spent an average of 3–5 hours per week including the documentation and meeting times. The students spent most of their time learning "how to debug" and then debugging small issues such as fixing loose wires–the whole process thus can be significantly accelerated with a close, hands-on supervision. 2 out of 5 students could not complete the last part of Assignment 1. We picked 2 out of 3 students who completed and employed them for another three months with the same workload to complete the rest of the assignments. Both students have succeeded in finishing the assignments and became the co-authors of this paper. Therefore, the results show that it is feasible to teach complex notions in hardware security to early-phase undergraduate students, but closer supervision is needed.

5.3 Experiment Challenges

To help adoption of a similar setup, this section discusses the major challenges, potential resolutions, and our future plans.

5.3.1 Participant Demographics. The proposed training module has been executed by 5 undergraduate students. These includes student from a diverse background (including underrepresented minorities in STEM) and those who do not have perfect GPA—the GPAs range between 3.2 and 4.0. Having such a range of students allowed us to test the effectiveness of our assignments to a broader audience. Unsurprisingly, students with a lower GPA were more challenged especially when it comes to designing and debugging new circuits that they have not seen before. The issues can be addressed with a closer engagement with those students.

5.3.2 First-Time and One-Time Practical Challenges. Attempting to debug breadboard through zoom made it hard to provide helpful feedback. Students needed to verify their designs through ModelSim and test alone. Not all students could run ModelSim; this made it hard for students to finish the last design. Due to chip shortages, not all students had voltage regulators. Some students used the AD2 power supplies or batteries. When collecting traces; not all students had enough RAM to process the data on MatLab.

An issue arose from the incompatibility of the AD2 script and the Graphical User Interface (GUI). AD2 GUI at high clock speeds could not capture measurements. The lag of the GUI made it so the script would also control the trace acquisition. Students were challenged when writing the script due to their lack of JavaScript knowledge, but they learned it within a week. A potential remedy for this challenge is to provide the full script except a few lines and to guide students to complete those parts.

Another issue was the lack of knowledge regarding probe positioning: the probes can be differential or single-ended—it was confusing how to place the probe pins to accurately capture the power consumption of the breadboard. During Assignment 3, one student decided to represent the targeted hidden layer output of 1-bit by using two flip-flops ('01' to represent value '1' and '10' to represent value '0'). This differential encoding inadvertently acted as a defense and suppressed the DPA signal. The research team provided supplemental videos and even guided walkthroughs of DPA students that could not figure out the issue.

5.3.3 COVID-19 Challenges. COVID-19 has had significant challenges in purchasing the needed parts and hands-on collaboration/tutoring. The parts ordered arrived after three months due to semiconductor supply chain issues. This can be a significant challenge if these experiments are to be integrated into a sophomore-level course that typically has 100+ students. Building and debugging breadboard circuits require significant physical labor. The

basics and troubleshooting can be done much simpler if the instructor or teaching assistant can be co-located in the same physical medium for an extended amount of time. This will still create an issue in scaling the experiments to a large class during COVID-19 or future pandemics. A remote hardware security testing ecosystem could be helpful.

5.3.4 Planned Future Changes. In light of this experience, we aim to apply the following changes.

- Record a single, multi-parted but full training video that teaches all basics of neural networks, side-channel attacks on neural networks, and application of defenses. This will help understand the fundamentals and theory before proceeding to the hands-on component.
- Provide hands-on tutoring in case COVID-19 or future pandemics allow.
- Include a rubric or layout for the reports in order to ensure professional and organized content.
- Include further experiments for covering other concepts in hardware security of ML.

6 CONCLUSIONS AND FUTURE DIRECTIONS

Hardware security of ML/AI applications is an emerging field of research with critical importance for future cyberinfrastructure. This paper described a set of hands-on experiments that aims teaching hardware security and ML/AI hardware implementations to a broad audience such as early undergraduate students. We revealed 4 experiments that allow teaching hardware implementation basics, neural networks fundamentals, side-channel analysis, and circuit-level hardware defenses, respectively. The results have shown that the students can successfully complete these experiments with minimal supervision beyond the experiment documentation. Therefore, the proposed training module can help address the problem of scaling the education of the next-generation STEM workforce. Our framework can further extend in the future by adding experiments related to the other emerging hardware security topics such as fault attacks, hardware Trojans, and logic locking.

7 ACKNOWLEDGEMENTS

This paper was funded in part by NSF CNS Grants no. 1943245. We thank anonymous reviewers for their valuable feedback.

REFERENCES

[1] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2019. {CSI}{NN}: Reverse engineering of neural network architectures through electromagnetic side channel. In *28th USENIX Security Symposium (USENIX Security 19)*. 515–532.

[2] Swarup Bhunia and Mark Tehranipoor. 2018. Hardware Security: A Hands-on Learning Approach.

[3] Jennifer Callen and Jason E James. 2020. CYBERSECURITY ENGINEERING: THE GROWING NEED. *Issues in Information Systems* 21, 4 (2020).

[4] Joseph Clements and Yingjie Lao. 2019. Hardware Trojan Design on Neural Networks. In *2019 IEEE International Symposium on Circuits and Systems*. 1–5.

[5] Matthieu Courbariaux and Yoshua Bengio. 2016. BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *CoRR* abs/1602.02830 (2016). arXiv:1602.02830 http://arxiv.org/abs/1602.02830

[6] Mircea Dabacan. 2018. Analog Discovery 2 Reference Manual. *Analog Discovery 2 Reference Manual-Digilent Reference* (2018).

[7] Anuj Dubey, Afzal Ahmad, Muhammad Adeel Pasha, Rosario Cammarota, and Aydin Aysu. 2021. ModuloNET: Neural Networks Meet Modular Arithmetic for Efficient Hardware Masking. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2022, 1 (Nov. 2021), 506–556.

[8] Anuj Dubey, Rosario Cammarota, and Aydin Aysu. 2020. MaskedNet: The First Hardware Inference Engine Aiming Power Side-Channel Protection. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2020, San Jose, CA, USA, December 7-11, 2020*. IEEE, 197–208.

[9] Louis Goubin and Jacques Patarin. 1999. DES and Differential Power Analysis The "Duplication" Method. In *Cryptographic Hardware and Embedded Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 158–172.

[10] Xiaolu Hou, Jakub Breier, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. 2020. Security Evaluation of Deep Neural Network Resistance Against Laser Fault Injection. In *2020 IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*. 1–6.

[11] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. *Advances in neural information processing systems* 29 (2016).

[12] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *Advances in Cryptology — CRYPTO' 99*, Michael Wiener (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 388–397.

[13] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. 2011. Introduction to differential power analysis. *Journal of Cryptographic Engineering* 1, 1 (01 Apr 2011), 5–27.

[14] Yuntao Liu, Yang Xie, Abhishek Charkraborty, and Ankur Srivastava. 2019. Secure and effective logic locking for machine learning applications. *Cryptology ePrint Archive* (2019).

[15] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2008. *Power analysis attacks: Revealing the secrets of smart cards*. Vol. 31. Springer Science & Business Media.

[16] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. 2006. Threshold Implementations Against Side-Channel Attacks and Glitches. In *Information and Communications Security*, Peng Ning, Sihan Qing, and Ninghui Li (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 529–545.

[17] Nicolas Sklavos, Ricardo Chaves, Giorgio Di Natale, and Francesco Regazzoni. [n. d.]. Hardware security and trust. ([n. d.]).

[18] K. Tiri and I. Verbauwhede. 2004. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, Vol. 1. 246–251 Vol.1.

[19] Yoo-Seung Won, Soham Chatterjee, Dirmanto Jap, Arindam Basu, and Shivam Bhasin. 2021. DeepFreeze: Cold Boot Attacks and High Fidelity Model Recovery on Commercial EdgeML Device. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–9.

[20] Qian Xu, Md Tanvir Arafin, and Gang Qu. 2021. Security of Neural Networks from Hardware Perspective: A Survey and Beyond. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 449–454.

[21] Pengyuan Yu and Patrick Schaumont. 2007. Secure FPGA Circuits Using Controlled Placement and Routing. In *Proceedings of the 5th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*. ACM, New York, NY, USA, 45–50.