Work-in-Progress: HyFlex Hands-On Hardware Security Education During COVID-19

Robert A. Karam
Dept. of Comp. Sci. and Eng.
University of South Florida
Tampa, Florida, USA

Srinivas Katkoori
Dept. of Comp. Sci. and Eng.
University of South Florida
Tampa, Florida, USA

Mehran Mozaffari Kermani
Dept. of Comp. Sci. and Eng.
University of South Florida
Tampa, Florida, USA

***Abstract*—Practical, hands-on hardware experience is an essential component of computer engineering education. Due to the COVID-19 pandemic, courses with laboratory components such as Computer Logic Design or FPGA Design were subject to interruption from sudden changes in course modality. While simulators can cover some aspects of laboratory work, they cannot fully replace the hands-on experience students receive working with and debugging hardware. For hardware security in particular, experimenting with attacks and countermeasures on real hardware is vital. In this paper, we describe our approach to designing a practical, hands-on hardware security course that is suitable for HyFlex delivery. We have developed a total of nine experiments utilizing two inexpensive, portable, and self-contained development boards which generally obviate the need for bench equipment. We discuss the trade-offs inherent in the course and experiment design, as well as issues relating to deployment and support for the required design software.**

***Index Terms*—hardware security education, hyflex, computer engineering, hardware laboratory**

I. Introduction

The COVID-19 pandemic has caused significant disruption to education at all levels and in every discipline [1], requiring institutes of higher education to rapidly identify and implement alternatives course modalities [2]. In computer engineering education, courses with hands-on or practical laboratory components were no exception, with laboratory sections often facing additional challenges beyond those encountered in lecture. In a typical semester, in courses such as Computer Logic Design, FPGA Design, Computer System Design, or Embedded Systems Design, students may need to physically build and test circuits on breadboards, implement and interact with designs on Field Programmable Gate Arrays (FPGAs), or work with embedded systems and components (e.g. sensors, actuators, microcontrollers). While students may be required to purchase individual lab kits for the course, they generally would not have access to their own laboratory equipment such as oscilloscopes, power supplies, or function generators at home. Hence, alternative instructional methods are required.

Gamage et al note that online delivery of experiments can meet some, but not all high level aims of experimental work in engineering curricula [3]. SPICE-based simulators with interactive front-ends can emulate the assembly of logic circuits, including the selection of discrete components and breadboard wiring. However, real-world measurement and implementation issues – noise, grounding, interference, and parasitic capacitance, and especially for circuits operating in the 10s of MHz range – are only really observed under non-ideal, real-world conditions. Meanwhile, FPGA or Computer System Design courses can at least focus on the use of the electronic design automation (EDA) software and simulators, but without access to individual development boards, students are not able to actually map synthesized designs to the FPGA and interact through pushbuttons, LEDs, or other peripheral components. Moreover, there are important differences in the design of hardware for simulation and for physical implementation, including register initialization and pin mapping.

In the Fall of 2019, we began developing an upper level (senior / graduate) hands-on hardware security "laboratory course" - a lecture course with interspersed hands-on activities using real hardware. Students were expected to work in teams of two and follow along in real-time with the instructor's demonstration using the hardware at their lab station, following the experiential learning cycle. Contrary to courses with discrete lecture and lab sections, this approach would allow for real-time integration of lecture-based theory with guided practical application and reflection / discussion among student groups. Several assignments consisting of additional hands-on experiments were also designed to complement the in-class activities. All activities and assignments were designed assuming students would be in class, at lab stations with suitable bench equipment and shared development boards for teams of two students, and have access to the equipment later on to complete assignments. Ultimately it became apparent that this course design would not be appropriate during the COVID-19 pandemic, and so we made a number of changes which would allow us to offer the course in a synchronous HyFlex modality, or completely remotely if needed.

In this work-in-progress paper, we describe the basic requirements and trade-offs for modifying this course to be suitable for HyFlex or remote delivery. In particular, we describe the restructuring of experiments, selection of hardware, and considerations for software tools necessary for enabling greater flexibility in course delivery. While the focus of this course was on hardware security, the same concepts can be applied to other hands-on labs.

This material is based upon work supported by the National Science Foundation under Grant No. DGE-1954259. Contact: rkaram@usf.edu

978-1-6654-8336-0/22/$31.00 ©2022 IEEE

TABLE I
OVERVIEW OF THE HARDWARE SECURITY LABORATORY EXPERIMENTS AND RELEVANT LANGUAGES / TOOLS.

Lab	Name	Tools and Components
1	Physical Unclonable Functions (PUFs) Implementation and Evaluation	FPGA, Verilog, Python
2	Deep Learning and Modeling Attacks on Arbiter PUFs	Python
3	Implementation and Evaluation of Pseudo- and True Randomness in Hardware	FPGA, Verilog, Python
4	Introduction to Side Channel Analysis and Leakage Assessment	ChipWhisperer, Python, C
5	Correlation Power Analysis Attack and Countermeasure for AES	ChipWhisperer, Python, C
6	Timing Attacks and Countermeasures for Password Protected Embedded Systems	ChipWhisperer, Python, C
7	Hardware Trojan Design and Countermeasures in a Soft CPU	Verilog, RISC-V Assembly
8	Security for the Internet of Things: Attack and Countermeasure for RSA	ChipWhisperer, Python, C
9	Post-Quantum Cryptographic Hardware and Embedded Systems	Verilog

II. BACKGROUND

Hardware security is a rapidly growing field which views the security of a computer system as starting with the hardware on which all other applications and networks are built. Hence, hardware is the "root of trust". Numerous hardware-based attacks and countermeasures have been described in the literature in the past two decades, including physical and side channel analysis (SCA) attacks, intellectual property piracy, hardware Trojan insertion, and others. More recently, courses which cover fundamentals of hardware security and trust have been introduced into the undergraduate curriculum at a number of universities. These courses generally cover supply chain security, testing and verification, hardware security primitives including Physical Unclonable Functions and True Random Number Generators (TRNGs), and invasive/semi-invasive/side-channel attacks and countermeasures from a theoretical perspective. We have previously offered hardware security course and have received very positive feedback from students about course material. However, much of the material is theoretical, with little practical or hands-on work. Some assignments require the use of datasets containing pre-measured values from real hardware, and students are asked to analyze the data, build machine learning models, or similar coding tasks. However, using pre-recorded datasets without access to the original hardware from which the data was gathered means that students cannot actively experiment with different implementations and countermeasures, reacquire data, and test the impact. Students cannot hypothesize about these effects, nor can they reflect on the concrete experience after experimenting. These are integral components of experiential learning [4], and are vital to a hands-on lab. While simulation may work to some degree in more general courses, in hardware security, working with real hardware is imperative. Simplified mathematical models cannot fully encapsulate real-world variations that are leveraged for hardware security primitives like physical one-way functions or true random number generators. For SCA, pre-recorded data only give students one dimension for analysis; they can evaluate and compare results from different implementations, but they cannot experiment themselves to see how changes to the design impact the information leakage, nor how different countermeasures manifest themselves in the side channel recordings. This motivated us to re-design our lab to be more amenable to a HyFlex or remote delivery.

III. DESIGN FOR A HYFLEX LAB

A total of nine hands-on experiments were designed for this course, which are listed in Table I. These experiments give students an opportunity to explore a wide range of topics related to hardware security, including implementations of PUFs and TRNGs in hardware, deep learning / modeling attacks on hardware, SCA and power / timing attacks, security in hardware / software interfaces, as well as advanced topics in post-quantum cryptographic hardware. Each lab involves the use of various tools, either software only, hardware only, or a mix of both. This is suitable for computer science and engineering students as they gain exposure to many different facets of hardware and software, including embedded software (C/C++), high level scripting (Python), hardware design (Verilog), and implementation on FPGAs (Xilinx Vivado). Ultimately we decided on a set of two development boards - one for FPGA called the Digilent Cmod S7, and one for SCA called the ChipWhisperer (CW) Nano. More details on these boards are provided in Section III-B. In particular, we needed to ensure that the boards were self-sufficient, and did not need any external hardware for measurement and test so that all experiments could be carried out at home, just as easily as on campus. This helps ensure alignment between students who are attending in-person and those who are attending from home, which is an important aspect of a hyflex course [5].

A. Description of Experiments

Each experiment consists of a guided in-class portion that not only introduces students to the theory, but also serves as a tutorial / walk-through for the related experiment. Whenever possible, we aimed to have students work together, including requiring teammates to exchange data acquired from their own boards as part of the data analysis and evaluation. The following is a brief summary of two of the experiments which particularly highlight the importance of working with real hardware, as well as situations where the lab topic and design requires direct collaboration between teammates.

1) Lab #1: Implementation and Evaluation of Physical Unclonable Functions: Physical Unclonable Functions (PUFs) are circuits that leverage nanoscale process variations during circuit fabrication that lead to minute differences in circuit structure. Because these variations cannot be controlled to such a fine degree, signatures extracted from these variations are

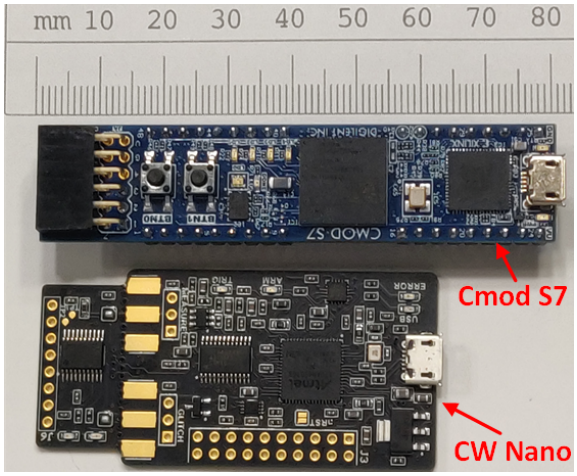

Fig. 1. The Digilent Cmod S7 and NewAE ChipWhisperer Nano boards used in the hardware security lab. The small size and free-to-use or open-source tools made these attractive for the hyflex course.

physically unclonable, and may be used as a chip identifier or for authentication [6]. In this experiment, students are provided with multiple FPGA configuration files (bitstreams) which they can program into their respective boards. The bitstreams given to each student are identical, but when programmed into their own FPGAs, the nanoscale variations result in different outputs for the same input. Students are also provided a file of sample inputs, called *challenges*, and must write a script in Python that sends the *challenges* to the FPGA, and reads back a *response* string. Due to the nature of PUFs, it is expected that each student will read back a different response for each challenge. Hence, it is vital that each student have their own physical hardware for data acquisition.

For the analysis, students will need to compute two metrics which correspond to the uniqueness and reliability of the PUF circuit - uniqueness is measured by how different the responses are for the same challenge on *different* boards, while the reliability is measured by how similar the response is to the same challenge on the *same* board if it is sent repeatedly. To compute the uniqueness, students must swap response sets with their partner, who independently acquire their own response set from their own board to evaluate reliability. Teams can make use of pair programming [7], either in-person, or using a platform with screen sharing such as Microsoft Teams. In future labs, students will again work in teams to try and "attack" their implementations using machine learning to model and the PUFs mapped on their respective FPGAs [8].

2) Lab #6: Timing Attacks and Countermeasures for Password Protected Embedded Systems: In this experiment, students will develop a system to break a simple password checking application in an embedded system, using the CW Nano board. Two firmware files implementing a password checking routine will be provided – one that has no countermeasures against side channel analysis, and one that does. For the unprotected file, students will not only know the password, but also will be able to view and modify the source code. In this routine, one character is checked at a time, and the function immediately returns upon finding a mismatch between the entered password and the correct password. This serves as a "practice" file on which students will mount various timing attacks. Initially they will use a clock function in Python to determine if there is a measurable difference between, for example, having 2 of 6 characters correct, versus having 5 of 6 characters correct. Students will have to implement a simple countermeasure, introducing a random delay before returning, to prevent this attack. Next, students will measure the power consumption during the password checking routing using the CW Nano as an oscilloscope. Patterns in the power consumption which indicate the processor is performing a comparison between characters will be clearly visible: when only n characters are correct, students will see n unique peaks in the power trace. This is true even for the first countermeasure they implement. They will need to then implement a second countermeasure against this attack by modifying the code to continue performing the comparison, even if a mismatch is detected. Once complete, students can then move on to the the attack phase and work together using pair programming to find the password for the second, protected file. Because the power consumption will depend on the instructions executing within the processor, it is vital that students have access to physical hardware for performing this experiment.

B. Hardware and Software Selection

Since it was not feasible to have all students in the lab at the same time, teaming up to work in close proximity on the same devices, we decided to redesign and re-implement certain aspects of the course to enable HyFlex course delivery. In particular, each student would need their own board(s) for the course and the experiments. This required careful selection of devices which met the following criteria:

- Board(s) should be relatively inexpensive, not only for initial acquisition of the hardware, but also in case a replacement is needed
- Board(s) should be small / portable with minimal peripheral components, as students would need to either take them to class regularly (if face-to-face) or take them home
- Software should be free and/or open source to avoid licensing issues, and it should be easy to install and run on student's computers

We identified two suitable boards for this course with software that met these criteria: the Digilent Cmod S7 and the ChipWhisperer (CW) Nano. Both boards are small - the Cmod S7 is about 79 x 18 x 18 mm, and the CW Nano is about 60 x 30 x 3 mm. Both boards are powered by a micro-USB cable, which doubles as a programming interface for the board. Hence, they are both portable, have minimal external components (no power supplies / separate power cables), and both can use the same micro-USB cable. Since no lab requires both boards to be programmed simultaneously, only one cable is needed. The software for both is freely available. For the Cmod S7, it can be programmed by the free version of Xilinx Vivado, which is available for Windows and Linux. Mac users can freely use a Linux-based virtual machine to run the software as well. The CW software is deployed as a

virtual machine image which greatly simplifies deployment on student's systems. VirtualBox is freely available for Windows, Mac, and Linux systems, so there are no compatibility issues. Since the CW software requires an installation of Python with various libraries, a C compiler, bash scripting support, etc., deployment as a VM also addresses potential configuration issues and cuts down on the required support time (potentially remote) diagnosing problems unique to one student's setup.

The Cmod S7 has limited I/O, including two pushbuttons, four LEDs, one RGB LED, and a small expansion header. It is, however, breadboardable, and so additional user I/O, e.g. extra switches, buttons, displays, etc. can be integrated. The board contains a Xilinx Spartan 7 FPGA with sufficient logic resources for all of the labs. The micro-USB connection allows for communicating with designs via USB Serial, which is an integral component of multiple labs and allows students to interrogate and read outputs from the mapped hardware through a simple Python script and the pySerial library, rather than rely on memory initialization files and recording outputs mapped to top-level pins using a logic analyzer or oscilloscope. This board supports labs 1, 3, and 7. Note that labs 1 and 3 cannot be simulated, as they exploit real-world variations in the hardware, whereas lab 7 can be run in simulation.

Meanwhile, the CW Nano has two onboard processors, one which can be controlled via a Python API and Jupyter notebooks, and a second "victim" microcontroller which can be programmed in C. Dozens of practical experiments related to side channel analysis can be arranged using this board. The main Python-controlled acquisition board can be used as a kind of USB oscilloscope which records the power consumed by the victim in real time, as it executes various functions. The victim can be controlled using a straightforward serial protocol, which allows students to write, flash, and execute various functions on the victim, while recording, observing, analyzing, and plotting the results in real-time from within the Jupyter notebook environment. This board supports labs 4, 5, 6, and 8. Note that, for all labs, it is possible to analyze previously-acquired data without requiring students to access the hardware. However, learning to use the equipment and troubleshoot / problem solve on their own is a key outcome. Moreover, simply providing students with pre-recorded data does not enable them to make changes (e.g. implement a countermeasure against the attack) to the firmware and observe how that impacts the resulting power consumption trace (and success of the original attack). Hence, students must have physical access to their own hardware for data collection.

IV. Future Work

This course was offered for the first time in Fall 2021. The course was open to undergraduate computer science and computer engineering majors, and was cross-listed as a graduate course. A total of three surveys are planned - a pre-survey, mid-term survey, and post-survey. The surveys will include a question asking students to self-report their own familiarity with various tools and topics in Computer Science and Engineering, including 1) mixed-signal / digital oscilloscopes, 2) logic analyzers, 3) EM / current probes, 4) multimeters, 5) function generators, 6) digital power supplies, 7) Matlab, 8) C/C++, 9) Java/C#, 10) Python, 11) Machine Learning / Artificial Intelligence, 12) GPGPU, 13) Cryptography (Theory), 14) Cryptography (Implementation), 15) Microcontroller / Embedded Development, 16) FPGA, 17) Verilog / VHDL, 18) EDA, 19) circuit simulation, 20) hardware prototyping, 21) PCB design, 22) 3D modeling, and 23) Git. Some of these topics are extensively covered in class, while others to a lesser degree or not at all. We expect those topics which are covered in more detail, especially those that were covered in multiple experiments and in-class activities, will demonstrate the greatest increase in self-reported improvement over the course of the semester. In general, we hope to improve our understanding of how the students' perceptions of their own knowledge and expertise in a given topic area or with a particular tool varies throughout the semester and compares with assessment grades. This study is expected to run for a total of four semesters, through the spring of 2023.

V. Conclusion

In this paper, we have described our experience in redesigning a hands-on hardware lab for HyFlex course delivery. Two development boards were selected which cover the full range of experiments as well as in-class activities. This approach ensures students can benefit from experiential learning associated with working with physical hardware, which is especially important for a course focused on hardware security where simulation is not a viable alternative. Labs require students to work together, either in the programming itself, or both programming and sharing of independently acquired results as part of overall data analysis. In future work we will report on the effectiveness of this hyflex lab using student survey responses, assessment scores, and other metrics.

References

[1] L. Mishra, T. Gupta, and A. Shree, "Online teaching-learning in higher education during lockdown period of covid-19 pandemic," *International Journal of Educational Research Open*, vol. 1, p. 100012, 2020.

[2] V. J. García-Morales, A. Garrido-Moreno, and R. Martín-Rojas, "The transformation of higher education after the covid disruption: Emerging challenges in an online learning scenario," *Frontiers in Psychology*, vol. 12, p. 196, 2021.

[3] K. A. Gamage, D. I. Wijesuriya, S. Y. Ekanayake, A. E. Rennie, C. G. Lambert, and N. Gunawardhana, "Online delivery of teaching and laboratory practices: continuity of university programmes during covid-19 pandemic," *Education Sciences*, vol. 10, no. 10, p. 291, 2020.

[4] D. A. Kolb, *Experiential learning: Experience as the source of learning and development*. FT press, 2014.

[5] S. Binnewies and Z. Wang, "Challenges of student equity and engagement in a hyflex course," in *Blended learning designs in STEM higher education*. Springer, 2019, pp. 209–230.

[6] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *2007 44th ACM/IEEE Design Automation Conference*. IEEE, 2007, pp. 9–14.

[7] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the case for pair programming," *IEEE software*, vol. 17, no. 4, pp. 19–25, 2000.

[8] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 237–249.