Introducing High-Level Synthesis in Computer Engineering Curricula

Prof. Amr Hassan, University of Pittsburgh

Dr. Amr Hassanreceived his B.Sc. degree in Electronics and Electrical Communications Engineering and the M.Sc degree in Engineering Physics from Cairo University, Egypt, in 2011 and 2015, respectively. He earned his PhD in Computer Engineering from the Electrical and Computer Engineering Department at the University of Pittsburgh, USA. Currently, he is an Assistant Professor with the same department. His Research interests include, but not limited to: Machine Learning, especially Deep Learning, for Image Processing and Video Prediction, Neuromorphic Computing Systems and its applications. and Innovation in Engineering Education.

Dr. Ahmed Dallal, University of Pittsburgh

Dr. Dallal is an assistant professor at the department of electrical and computer engineering, Unversity of Pittsburgh, since August 2017. Dr. Dallal's primary focus is on education development and innovation. His research interests include biomedical signal processing, biomedical image analysis, and computer vision, as well as machine learning, networked control systems, and human-machine learning.

Mr. Mohamed A. S. Zaghloul,

Mohamed A. S. Zaghloul was born in Cairo, Egypt, in 1987. He received his B.E. degree in Electronics and Electrical Communications Engineering in 2009, and his M.Sc. degree in Engineering Physics in 2012, both from the Faculty of Engineering at Cairo Univ

Dr. Samuel J Dickerson, University of Pittsburgh

Dr. Samuel Dickerson is an assistant professor at the University of Pittsburgh Swanson School of Engineering. His general research interests lie in the area of electronics, circuits and embedded systems and in particular, technologies in those areas that have biomedical applications. He has expertise in the design and simulation of mixed-signal integrated circuits and systems that incorporate the use of both digital and analog electronics, as well as optics, microfluidics and devices that interface to the biological world. Dr. Dickerson is also interested in enhancing undergraduate engineering education, and investigates new and innovative methods for improving the learning experience for electrical and computer engineering students.

Introducing High-Level Synthesis in Computer Engineering Curricula

Abstract

In this paper, the introduction of High-Level Synthesis (HLS) concepts and hands-on activities into an undergraduate computer engineering program is presented. Students are provided instruction on the basics of digital hardware design using VHDL (VHSIC Hardware Description Language). That instruction is followed by modules on HLS. Students are tasked with putting what they learned to practice via a project in which they must implement 2D image convolution on a Field-Programmable Gate Array (FPGA), using HLS tools. The impact of the instruction on students is assessed via surveys and the results show that students find the material to be challenging, yet interesting. The results also show that there is ample room to provide additional instruction to students on topics that are core to modern hardware design, such as Direct-Memory Access (DMA).

1. Introduction

Courses on digital design are essential components of all computer engineering curricula [1]. Most programs have an introductory digital design course, where students learn the basics of logic gates, hardware description languages, and FPGA design. This introduction is typically followed up by an advanced course where students delve deep into digital design flows, Register Transfer Level (RTL) design, and FPGA synthesis. While these advanced topics are still essential for computer engineering students to learn, the professional practice has taken leaps forward in their use of FPGAs by adopting newer technologies. Specifically, the incorporation of FPGAs that support HLS (Fig. 1), the automatic creation of register-transfer level descriptions from abstract, high-level programming languages (such as C++), is now the most prolific use-case for FPGAs in cutting-edge, real-world applications [2-4]. Furthermore, modern FPGAs have more than just programmable logic, they also contain integrated CPUs that can be leveraged for additional functionality. In a recent article published in 2019 [5], the authors surveyed more than 40 papers published since 2010 about the Quality of Results (QoR) and productivity of HLS, compared to RTL design flows. The results showed that using HLS tools, the design time can be cut down to third and the designer productivity can be increased to 4 times, while the QoR of HLS being slightly below that of the RTL design flow [5].

Computer engineering programs have been slow to adopt these technological changes and as a result, examples of the use of HLS in the undergraduate classroom are few and far between. Recently published studies have been exploring the advantages of including HLS in undergraduate computer engineering programs [5-7], yet many questions need to be answered. Some of the research questions that we are going to answer in this work are: 1) How and when should HLS be introduced in computer engineering curricula 2) What preparation do students need to understand HLS? 3) In what context/application area should HLS be introduced and 4) What is the impact on students and their future prospects?

In this work, we address these research questions via the revamping of an Advanced Digital Design course, offered by the Electrical and Computer Engineering Department at the school hosting this study. The instructors incorporated the use of state-of-the-art FPGA kits and deployed hands-on, HLS modules. Additionally, students were taught how to leverage the integrated CPUs to automate testing throughout the course.

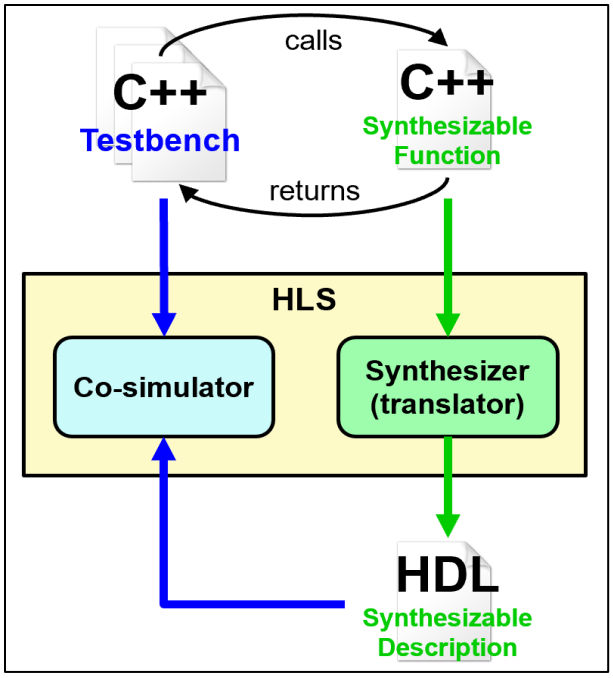

Figure 1: High-level Synthesis Design Flow.

The rest of the paper is organized as follows: Section II briefly mentions previous work in this area. Section III introduces the course under investigation and the details of the implement HLS lab module. Section IV provides the assessment methos used and a discussion on the obtained results, while section V concludes the paper.

2. Background, Previous Work, and Motivation

Examples of incorporating HLS in engineering curricula, and in computer engineering curricula specifically have been scarce to find. However, recent published studies show that universities around the world have been noticing the urgent need to start incorporating HLS in different engineering curricula. Navarro et al [8] introduced the use of HLS tools in digital electronics course to implement a BUCK closed-loop controller for power converters. Comparing the performance of the HLS generated HDL code, with previous implementations in fixed and floating point VHDL, students were able to recognize the benefits of using HLS tools to quickly explore and optimize the design space [8].

Nelson et al [6] designed an experiment where they had a sophomore student implement a RISC-V processor in a simple looped design in both HDL and using an HLS design flow in C. the discussion revealed the student was very pleased with the experiment, but still needed more time to fully understand and unlock the potential of using HLS tools.

Additionally, Skliarova et al explored the use of HLS in reconfigurable digital systems course [9]. The students implemented a system to sort N M-bit data items using an iterative even-odd transition network [9]. Students were asked to implement this system in both RTL and HLS design flow and were asked to compare the trade-offs between both methods in terms of design footprints and required design time.

Last but not least, Huang et al designed a four-hours mini-course to teach computer engineering students about leveraging HLS to implement neural networks hardware accelerator on FPGAs [7]. Students were

only required to know programing language concepts, without prior exposure to digital or hardware design [7].

While all the previous examples are valuable and show a move in the right direction, they lack the research aspect of how well Computer Engineering (CoE) students will perceive this new knowledge. This simple fact was the main motivation behind this work. Not only do we introduce HLS modules in the current Advanced Digital Design course, but we also assess students' needs and comprehensibility of the materials and adapt the modules accordingly. This will be discussed later in the coming sections.

3. Course Format

Before the newly revamped course was rolled out, the older version of this course focused only on advanced RTL design and used obsolete FPGA boards with limited in/out peripherals. Although students were designing complex RTL blocks, they were only limited by the FPGA in/out peripherals for testing purposes. These limitations forced the students to design very primitive testbenches that did not thoroughly test their design and left it full of bugs. This was one of the main reasons the authors decided to completely revamp it. The revamped course in which this study is conducted has two distinct segments. First, students receive basic education on digital design flows, with a focus on design for synthesis. Afterwards, students learn about HLS.

3.1. Digital Hardware Design

The course presented in this work is a semester-long undergraduate course. At the University where this study took place, the course is required for computer engineering majors and elective for electrical engineering majors. The course is typically taken by students in their junior or senior year, after they have completed an introductory course in digital logic. In the introductory course, students are taught the basics of VHDL, FPGAs, and typical digital hardware design flows.

Although in the preceding introductory course students are taught VHDL, the emphasis is on logic design. Thus, the details of logic synthesis and writing efficient or high-performance HDL is not discussed extensively. The follow-on course begins by reintroducing students to VHDL and FPGA design flows via a series of projects. In the first project, students are taught concepts that revolve around combinational logic and design an arithmetic logic unit. The focus is placed on the design of hardware synthesizable VHDL.

After the design of the arithmetic logic unit, students are tasked with the design of a hardware multiplier. The students are asked to construct a multiplier using registers, combinational logic, and a finite state machine (Fig. 2). The primary goal is to introduce students to concepts behind (RTL) design where a datapath is controlled via a finite state machine. This is the level of hardware design that is typically emphasized in undergraduate programs and is still a vital part of their education in digital hardware design. This project also serves as a great introduction to the concept of converting an algorithm to synthesizable hardware, which is vital background for HLS.

After students implement the multiplication algorithm in hardware, they are tasked with a large-scale project, the design of RISC MIPS CPU that supports several instructions. This project is included so that students can develop into proficient digital hardware designers having to exercise their knowledge

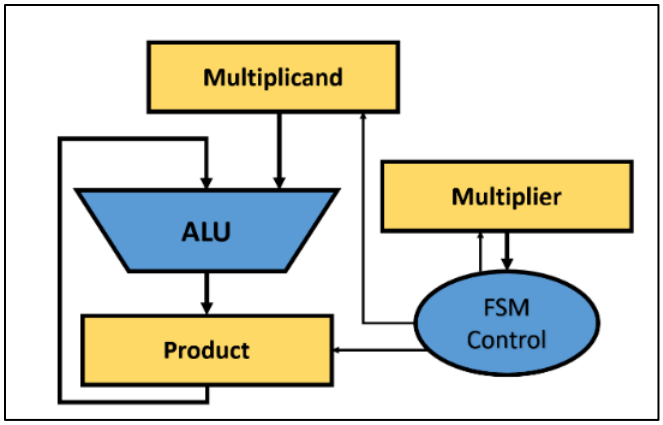

Figure 2: Register-Transfer Level (RTL) design of the multiplication algorithm.

of RTL design and design for synthesis at a large scale. After the completion of this project, students move on to learning about HLS.

3.2. High Level Synthesis

The training and modules delivered to the students up until the introduction of HLS is typical for most undergraduate computer engineering programs. Although typical, it is a prerequisite to teaching HLS. Without a proper background in RTL design, students may incorrectly view HLS design flows as if it were traditional software development.

The first key to designing the educational modules and project is to provide a problem that is situated in a context for which HLS is desirable. For this reason, a project centered on image processing was selected. Specifically, the students implemented an edge detection filter, which requires 2D convolution of a kernel over an image. This project topic has several appealing features. First, the solution can be easily implemented in software and hardware. Thus, students are afforded the opportunity to directly compare the two realizations. Second, image processing is a domain for which industry stakeholders have a keen interest, especially the convolution process of a kernel over an image, which is heart of Convolutional Neural Networks CNNs [10]. Finally, the results are visual, and students are highly motivated to "see" their algorithm in action.

Instructions begin by providing students with a primer on convolution of a kernel over an image. In the explanation, the math and theory are secondary to the software implementation. What is of most interest is that the naïve software approach leads to two nested loops (Fig. 3). Students are shown how these loops lead to inefficient memory accesses, with several image pixels having to be repeatedly fetched in memory, even when not being acted upon.

This leads directly to a discussion that teaches how to rewrite software in such a way that lends itself to hardware realizations. The students are instructed on a "Stream Approach" by which the image is interpreted as a 1D vector, eliminating the inefficient memory access issues (Fig. 4). This stream approach requires the addition of temporary storage buffers so that pixels that are not yet being acted upon can be stored for future use without the need for additional memory accesses.

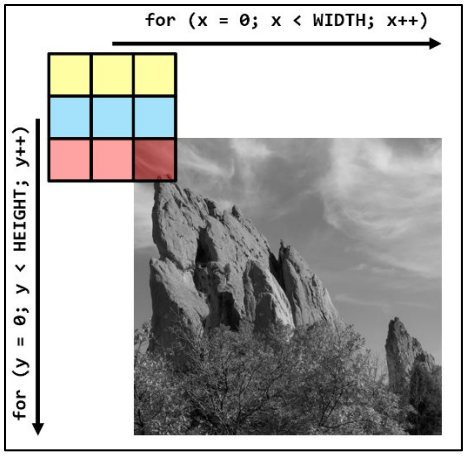

Figure 3: Depiction of basic approach to 2D Image

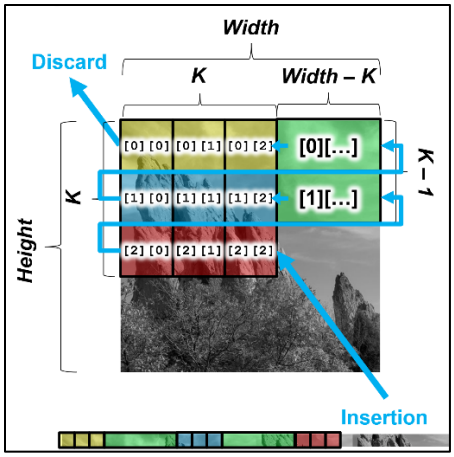

Figure 4: Memory efficient streaming approach to image

One of the primary outcomes from seeing how to process the image via the "Stream Approach" is the students see the reduction of the software implementation go from two nested loops to a single loop with ROW*HEIGHT iterations (Fig. 4). This reformation of the code primes it for HLS and takes advantage of its features for high-performance computation, namely loop unrolling, pipelining, and direct memory access. An explanation and subsequent analysis of the dataflow graph reveals to the students where loop unrolling and pipelining can be applied to increase performance. For many students this serves as their first introduction to the concept of "unrolling" a loop or breaking it up into parallel computations and pipelining outside of the context of CPU instruction fetches.

Figure 5 shows an example image that has been correctly processed by a student's FPGA implementation. The final piece of instruction provided to the students is on the concept of Direct Memory Access (DMA) and how to stream data to accelerate computation. Figure 6 shows the acceleration framework used in this course and the basis for the student project. Students are tasked with the design of a hardware image processing "accelerator" (ACCEL). This unit is the only scope for which students must write their own code, all the other modules are provided. In the accelerator, students are tasked with implementing the convolution. They are first asked to implement it as they would with a pure software approach, and then later asked to redesign their implementation such that it streams input from DMA and takes advantage of unrolling and pipelining.

The DMA unit utilizes the industry standard AXIS bus that fetches images direct from DDR memory. The FPGA chip used for this course also contains a host ARM microprocessor that is used to host the user application and testbench.

4. Assessment Methods and Results

For assessment, students were surveyed over the recent four offerings of this course to capture their perspectives and opinions on the impact of the new components on their learning. The very first two offerings of the revamped course were during Fall 2020 and Spring 2021, during the COVID-19 pandemic, and were delivered remotely. In Spring and Fall of 2022, a more comprehensive assessment of the impact and effectiveness of the HLS instruction was conducted.

(a) (b)

Figure 5: Example image processed by a student's FPGA implementation. (a) shows the test image [11] and (b) shows the convoluted image after applying the edge detection filer.

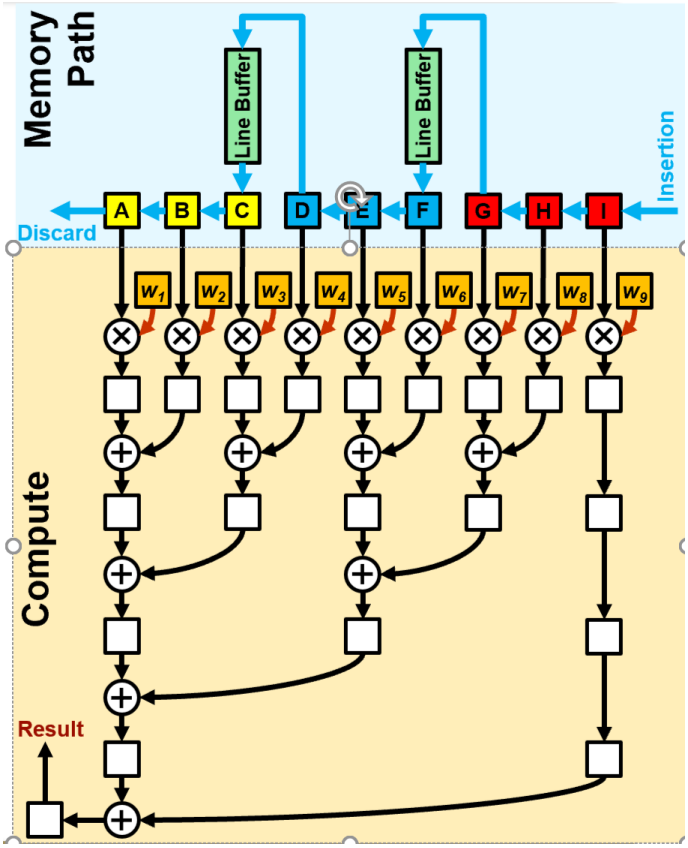

Figure 6: Data Flow Graph illustrating the unrolled and pipelined computations.

4.1. Assessment of High-Level Synthesis Instruction (Spring 2022)

To assess the effectiveness of the instruction specific to HLS and its impact on students, a survey was conducted in the Spring semester of 2022 (n = 14). It is worth mentioning here that the HLS lab component constituted about 15% of the time allocated for this class. The results displayed in Table 1 show that students overwhelmingly feel that the subject matter is interesting and engaging (Q1) even though they also perceived the lab as challenging (Q2). Students felt that they had a strong understanding of HLS in general and how to unroll loops (Q3 and Q4). However, the results showed that the students struggled the most with understanding how DMA fit into the overall acceleration framework.

Table 1. Assessment of HLS Instruction (Spring 2022)

Question (n=14)	**Strongly Disagree**	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
I thought Lab #5 was interesting	0%	0%	7%	79%	14%
I thought Lab #5 was challenging	0%	7%	29%	50%	14%
I have a good understanding of the capabilities and goals of HLS	0%	0%	21%	71%	7%
I have a good understanding of how to apply the concept of loop-unrolling in the context of HLS	0%	0%	14%	57%	29%
I have a good understanding of how to apply the concept of pipelining in the context of HLS	0%	0%	36%	50%	14%
I have a good understanding of how Direct-Memory Access (DMA) was used in this lab	7%	29%	29%	36%	0%
I am interested in learning more about the topics covered in Lab #5	0%	7%	29%	50%	14%

4.2. Assessment of High-Level Synthesis Instruction (Fall 2022)

In order to address the students' concern about DMA, in particular, and to give them a better understanding of HLS, in general, we introduced a new intermediate lab before the 2D image convolution lab discussed in section 3.2. This lab acts as a soft transition into the 2D image convolution lab, where the students get introduced into HLS via designing a simple 2D matrices multiplier. Matrix multiplication is simple enough for the students to implement in C/C++, with very simple three nested loops, so they can focus more on HLS concepts. In addition, students get to implement different optimization techniques, like loop unrolling and pipelining, and see the difference in the utilized resources on the FPGA board.

With the new lab introduced, the HLS labs compromised about 23% of the time allocated for this class and the different HLS concepts are divided between the two labs as follows:

1) Lab #5 2D Matrix Multiplication: HLS introduction, loop unrolling, pipelining, and reading utilization reports.

2) Lab #6 2D Image Convolution: array partitioning, DMAs, and AXI Stream Interface.

The same assessment was conducted during Fall 2022 (n = 34). Table 2 depicts the individual results of the survey questions and Fig. 7 shows a comparison between the results of the two semesters. The horizontal axis shows the results categorized into: Disagree (strongly disagree and disagree together), Neutral, and Agree (agree and strongly agree together). While the students still think that Lab #6 is still challenging (Q2), they now have better understanding for loop unrolling (Q4), pipelining (Q5), and DMA (Q6), as indicated by the increase in the percentage of the students under the "Agree" category.

Table 2. Assessment of HLS Instruction (Fall 2022)

Question (n=34)	**Strongly Disagree**	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
I thought Lab #6 was interesting	0%	3%	3%	50%	44%
I thought Lab #6 was challenging	0%	0%	9%	68%	23%
I have a good understanding of the capabilities and goals of HLS	0%	9%	17%	53%	21%
I have a good understanding of how to apply the concept of loop-unrolling in the context of HLS	0%	0%	9%	56%	35%
I have a good understanding of how to apply the concept of pipelining in the context of HLS	0%	9%	23%	44%	24%
I have a good understanding of how Direct-Memory Access (DMA) was used in this lab	0%	26%	24%	38%	12%
I am interested in learning more about the topics covered in Lab #6	3%	6%	21%	44%	26%

4.3. Qualitative Assessment of Students Employability

While it is hard to track down every student who took this class and see how this course affected their potential employability in the field of hardware digital design, the instructors received some very promising feedback from three students regarding this matter. Two of the two students were having a co-op at a big tech company doing RTL design with the Integrated Design Environment (IDE) that we use in class. When they saw who skilled they are with the IDE and HLS, they offered them a full-time position just after a month into their co-op. One of the students said, and we quote, "The job is exactly like your course, but easier!" The third student reached out saying that she was able to secure an internship during summer 2023 at a company doing image processing using FPGAs. She said, and we quote, "The internship is basically a derivative from Lab 6!"

5. Summary and Conclusions

In summary, an introduction of HLS into an undergraduate course was presented. This subject matter, which is normally reserved for graduate level courses, was introduced by first, instructing students on the basics of digital design and RTL methodologies. After that foundation was laid, students are taught HLS in the context of accelerating image processing applications.

The results show that the instructions were effective, as most students found the material to be engaging. The assessment reveals that students became confused when DMA was introduced. Thus, the instructors introduced a new intermediate lab to ease the transition into HLS and to decouple the basics of HLS from advanced concepts like DMA and AXI. This modification had led to students better understanding the advanced topics. Yet, they still find HLS labs challenging.

As per the results obtained and students feedback mentioned in section 4.3, the instructors believe the answer to research question 1 is that HLS should be introduced during junior/senior year for computer engineering students. This is when the students have had enough exposure to basic HDL in previous courses, and ready to receive leverage HLS for more complex designs. As per question 3, the instructors believe that using HLS to build image processing accelerator is an excellent application for the students at this level, but it needs to be even more optimized. In future iterations, the instructors will allocate even more time for HLS labs divide the labs into different milestones to target the different challenges the students face.

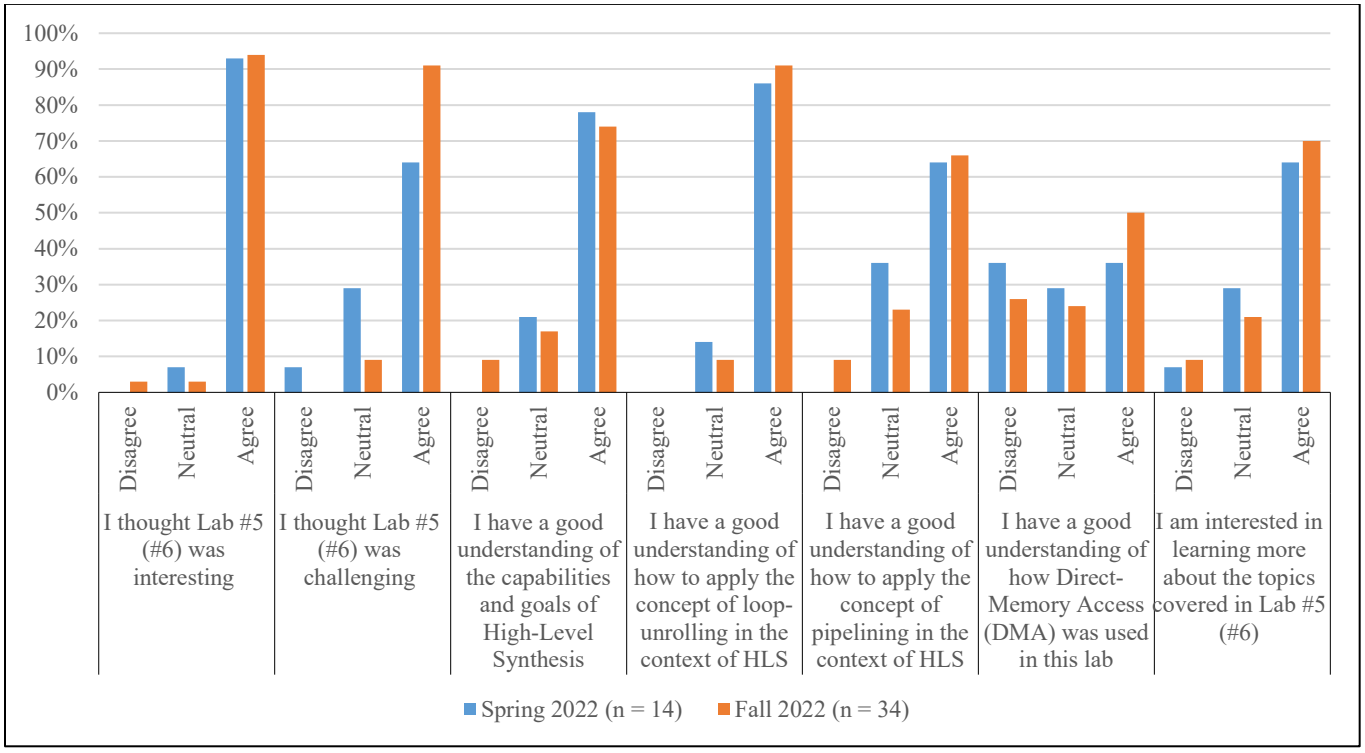

Figure 7: Comparison between survey results in Spring and Fall of 2022.

References

[1] "The Joint Task Force on Computing Curricula, IEEE Computer Society, Association for Computing Machinery." https://www.acm.org/education/curricula-recommendations (accessed 4/25/2022.

[2] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-level synthesis for FPGAs: From prototyping to deployment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 30, no. 4, pp. 473-491, 2011.

[3] K. Vissers, S. Neuendorffer, and J. Noguera, "Building real-time HDTV applications in FPGAs using processors, AXI interfaces and high level synthesis tools," in *2011 Design, Automation & Test in Europe*, 2011: IEEE, pp. 1-3.

[4] R. Nane *et al.*, "A survey and evaluation of FPGA high-level synthesis tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 35, no. 10, pp. 1591-1604, 2015.

[5] S. Lahti, P. Sjövall, J. Vanne, and T. D. Hämäläinen, "Are we there yet? A study on the state of high-level synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 38, no. 5, pp. 898-911, 2018.

[6] I. Nelson, R. Ferreira, J. A. Nacif, and P. Jamieson, "Is it time to include High-Level Synthesis design in Digital System Education for Undergraduate Computer Engineers?," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021: IEEE, pp. 1-5.

[7] N.-S. Huang, J.-M. Braun, J. C. Larsen, and P. Manoonpong, "Teaching hardware implementation of neural networks using high-level synthesis in less than four hours for engineering education of intelligent embedded computing," in *2019 20th International Carpathian Control Conference (ICCC)*, 2019: IEEE, pp. 1-7.

[8] D. Navarro, Ó. Lucía, L. A. Barragán, I. Urriza, and J. I. Artigas, "Teaching digital electronics courses using high-level synthesis tools," in *2013 7th IEEE International Conference on e-Learning in Industrial Electronics (ICELIE)*, 2013: IEEE, pp. 43-47.

[9] I. Skliarova, V. Sklyarov, A. Sudnitson, and M. Kruus, "Integration of high-level synthesis to the courses on reconfigurable digital systems," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015: IEEE, pp. 166-171.

[10] L. Alzubaidi *et al.*, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *Journal of big Data,* vol. 8, pp. 1-74, 2021.

[11] S. Bae, S. Paris, and F. Durand, "Two-scale tone management for photographic look," *ACM Transactions on Graphics (TOG),* vol. 25, no. 3, pp. 637-645, 2006.