Introducing FPGA-based Machine Learning on the Edge to Undergraduate Students

Rajesh C Panicker
Department of Electrical
and Computer Engineering
National University of Singapore
Singapore
Email: rajesh@nus.edu.sg

Akash Kumar
Centre for Advancing
Electronics Dresden
Technische Universität Dresden
Germany
Email: akash.kumar@tu-dresden.de

Deepu John
School of Electrical
and Electronic Engineering
University College Dublin
Ireland
Email: deepu.john@ucd.ie

***Abstract*—This innovative practice category work in progress paper describes a project in a final-year undergraduate course on implementing a neural accelerator on an FPGA for edge computing. In our university, an undergraduate course on Embedded Hardware System Design introduces students to advanced hardware design techniques with the goal of integrating the created hardware into a complete system. Students learn concepts such as high-level synthesis (HLS), logic synthesis and physical design, with an emphasis on FPGA-based designs. They also understand bus systems such as Advanced eXtensible Interface (AXI) to interconnect the various components. The concepts are put into practice through a project. A series of labs provide scaffolding to students through the course of implementing the project. These labs take students systematically through an introduction to hardware-software co-design, hardware design, creation and interfacing of custom co-processors and HLS. Important hardware design and optimization concepts, as well as managing the data interaction between hardware and software were reinforced through the project. The project also provided many students with the opportunity to be introduced to neural networks and machine learning (ML). Quantitative and qualitative results from a survey indicate that students gained a lot of knowledge and experience through the course of the project. The current form of the project streams in data from a local computer to which the FPGA is connected. Future work includes true and direct cloud connectivity and improved use cases for making it a true Internet of Things (IoT) project.**

***Index Terms*—Hardware acceleration, machine learning, FPGA.**

I. Introduction

Project based learning is a powerful way to impart conceptual and practical knowledge simultaneously; it gives students a broader context to theoretical knowledge learnt in lectures and motivates students to take greater responsibility for their learning by becoming more active learners. This results in improved knowledge retention, and being able to integrate knowledge gained from various courses [1]. A well-designed project-based learning experience can provide students opportunities to perform at all 6 levels of cognitive domain classified in Bloom's taxonomy of learning [2], ultimately making engineering graduates better prepared for the workplace [3].

As ML is rapidly gaining popularity in a variety of applications, an increasing number of undergraduate curricula are incorporating aspects of ML [4]. In our university, there are no compulsory courses in ML in the electrical or computer engineering curricula for students in their senior years. Even when ML is not introduced as a stand-alone course, aspects of machine learning are incorporated in other courses [5]. However, many such courses miss out one aspect - performing computationally intensive ML on traditional hardware such as CPUs might not be efficient [6]. The availability of libraries which can make use of GPUs address this to a good extent. However, a number of companies and universities are working on creating dedicated chips for ML, which deviates from the traditional instruction set architecture-based computation. This is even more important considering the industry trend of doing a significant amount of computation at the edge of the network, thereby reducing storage and transmission requirements. Custom hardware allows it to be more power efficient, which is crucial in battery operated sensor nodes [7].

However, teaching students to create a custom application specific IC is not practical due to time and cost constraints. Hence, FPGAs have been widely used in both industry and academia for prototyping. Worldwide, many courses use FPGA for teaching digital design and embedded systems [8]–[10]. FPGA-based systems can also adapt to different algorithms and computation requirements without investing in new hardware. For this reason, FPGAs are gaining immense popularity in edge computing. Traditionally, this requires programming in HDL, which imposes a high barrier to entry. However, the advent of HLS tools which can generate hardware from high-level code such as C or OpenCL has reduced this barrier to entry, and educators are incorporating it into their courses [11]. It is imperative to give both high-level and low-level synthesis experience to students.

Giving a ML problem for custom hardware acceleration fulfils both goals - it allows students who are new to ML to gain some ML experience, while teaching them how to create hardware accelerators for embedded systems capable of performing efficient edge computation. This forms the motivation for our paper.

The rest of the paper is organized as follows: Section II provides details of the course, which gives the context of this paper, as well as the details of the preparatory labs designed to guide students towards the project. Details of the ML project are given in Section III. The student performance and feedback results are given in Section IV. The paper concludes with Section V, which also details future directions.

II. Course Details and Preparatory Labs

A. Course Details

The course EE4218 at the National University of Singapore is meant to give students knowledge of hardware design and its eventual integration into a complete embedded system. Topics covered include hardware design for embedded systems, hardware specification and modelling, hardware vs. software, hardware implementation choices and design flow. The target audience for this is year 3 or 4 undergraduates in electrical engineering and computer engineering. Expected prior knowledge includes a course on digital design and a course on computer organization or microcontroller programming. The learning outcomes are given in the first column of Table. II.

40% of the grade for the course is based on an open-book final exam. 10% grade is based on a series of 3 quizzes, conducted online. The rest of the 50% comes from labs and the project. There are 13 weeks of lectures for the course, with each lecture being 3 hours long. Students worked in pairs for the labs and the project and a hardware board was loaned out to each pair for the duration of the semester.

B. Preparatory Labs

The hardware platform used in this course is Zedboard - a popular board based on Zynq System-on-Chip (SoC), it offers a combination of dual ARM Cortex A9 and other essential peripherals forming the processing system (PS), and programmable logic (PL, the FPGA part) in a single package. This makes it an ideal platform for experimenting with hardware-software co-design and hardware acceleration.

Four labs were designed to give students hands-on experience in developing hardware platforms, hardware/software co-design and high-level synthesis. Each lab had one part which served as a step-by-step tutorial, and another part which required students to use knowledge and experience gained to solve a fairly simple problem. These labs formed the building blocks and the eventual platform which was used to implement and use the hardware accelerator. Labs 1 and 2 carried 5% weight; and Labs 3 and 4 carried 10% and 2.5% weight towards the final grade respectively. The project, described in the next section, carried 27.5% weight.

Lab 1: This introduced students to hardware/software co-design. Students were guided on creating a hardware platform using a Xilinx IP integrator of Vivado, followed by a Board Support Package (BSP, a collection of driver functions and other utilities), and eventually the software application which runs on the ARM Cortex A9 processor and interacts with the peripherals. Lab 1 involved a homework task to multiply a matrix sent from the console by its transpose, and then printing the result back on the console.

Lab 2: This acted mainly as a refresher to the basic digital design course that students had done in their first year. The task was to multiply a small matrix by its transpose on the press of a button. The results are displayed on LEDs, one element at a time. The whole process is sequenced by a state machine.

Lab 3: This introduced students to connecting peripherals to the system using AXI and AXI Stream (AXIS) interfaces. The exercise involves connecting a Xilinx IP block for driving a VGA display, as well as a custom co-processor. AXIS is chosen as the interface for the custom co-processor for two reasons - it is simpler for students to understand and implement as the data streams in and out, and it is supported by the Vivado HLS tool for hardware/software co-simulation. However, it requires an AXI Direct Memory Access (DMA) controller to connect it to the AXI bus. A typical block diagram for Lab 3, with VGA not included is given in Fig. 1. This forms the testbed for Lab 4 and the project. The homework task was to use the co-processor to multiply two 32-bit numbers.

Lab 4: Lab 4 introduced students to the Vivado HLS tool. Students learnt the HLS design flow, and optimization options available in the tool. The homework task was to multiply two numbers as in Lab 3, and integrate the generated design as a drop-in replacement for the co-processor in Lab 3. A wiki was used as a convenient platform to allow easy collaboration [12]. The wiki is accessible at https://wiki.nus.edu.sg/display/ee4218.

III. Project Specifications and Details

The goal of this project is to create a hardware accelerator for a multilayer perceptron (MLP) neural network. An MLP is a ML technique inspired by the working of the biological neuron. Each neuron computes a weighted sum of the inputs to it, and subjects it to a non-linear activation function. A network of such neurons can be trained using an algorithm such as backpropagation, which involves optimizing the weights of the connections in the network to minimize prediction error. A trained network can give predictions regarding the class an input belongs to.

Students are required to implement a hardware accelerator just for prediction, not for training. The training process can be carried out purely in software. The designed system should be able to do prediction in 3 different ways
SOFT: A software implementation on ARM Cortex A9.
HARD_HDL : An AXIS co-processor implementing the neural network in hardware, written in HDL.
HARD_HLS : An AXIS co-processor implementing the neural network in hardware, which is at least partly created using HLS tool.

Students are encouraged to write the C code from scratch, but are free to use an implementation from elsewhere. However, students were expected write the HDL code for the co-processor themselves. The process of converting the C code to hardware using HLS also had to be done entirely on their own. To allow some configurability, hard coding of weights in the hardware was not permitted. Students should send the training data from the PC via the serial console. The C program running on the ARM Cortex A9 processor should receive the training dataset, train a neural network, and prompt the user to send the test data. Upon receiving the test data, the C program should get the prediction done through SOFT or HARD_HDL or HARD_HLS, and send the predictions back to the console. This had to be captured in a file, which could be compared against the actual labels to compute the prediction accuracy using a spreadsheet or a custom program.

There were no restrictions on the exact neural network architecture. The emphasis was not on raw classification accuracy, but on following systematic design and optimizations. The exact way computations were done, as well as precision, was left as flexible. It was recommended, but not mandatory, to have some sort of a timer/timing mechanism for comparing time taken by the three predictors. Students were allowed to choose a dataset. A modified Wine dataset [13], where values are limited to 0-255, was given to get them started.

Feedback Survey Question	Score (Out of 5)
Overall opinion of the course	4.0
Expected grade	4.2
Difficulty Level of the course	4.0

TABLE I: Results of the feedback survey

IV. Student Performance and Results

A. Student Performance

This new project was run in the current form described in the paper for one semester. It was attempted in parts in two semesters before, and the requirements were fine-tuned based on feedback. Performing training on hardware proved to be difficult for students within a semester, so the requirement with respect to hardware acceleration was changed to prediction only. Labs were also streamlined such that the activities in labs contributed directly to the project.

Majority of the students managed to do the C as well as HARD_HLS part of the project. Some used multiple co-processors, with load distribution done either in hardware or software. Some had the ability to use either of the three processing methods based on a user input on a console or through a hardware input. Some students had difficulties with the HARD_HDL, as they did not follow the systematic hardware design practices and implemented a design which was too high in the critical path delay, resulting in a non-functional hardware. Some implementations included mechanisms where the weights need not be sent again and again for each test data, unless and until there is a re-training/reset. There were also designs using AXI interface directly instead of AXIS.

B. Survey Results

Quantitative and qualitative results for the course indicate that most students perceived the course to be useful, in spite of many finding it difficult. The results of the end of the semester survey for the course is given in Table I, which indicates that most students had a good opinion of the course. The learning outcome survey in Table II indicates most students felt the learning outcomes were achieved. The challenging nature of the project could be why many students were not confident in choosing ‘Able all the time’ for the outcomes. These survey results are based on 18 respondents from a class of 43 students. Some of the positive and negative qualitative feedback from students is given below.

“The exposure to hardware and software and their links”, “practical lab”, “Very informative”, “The project

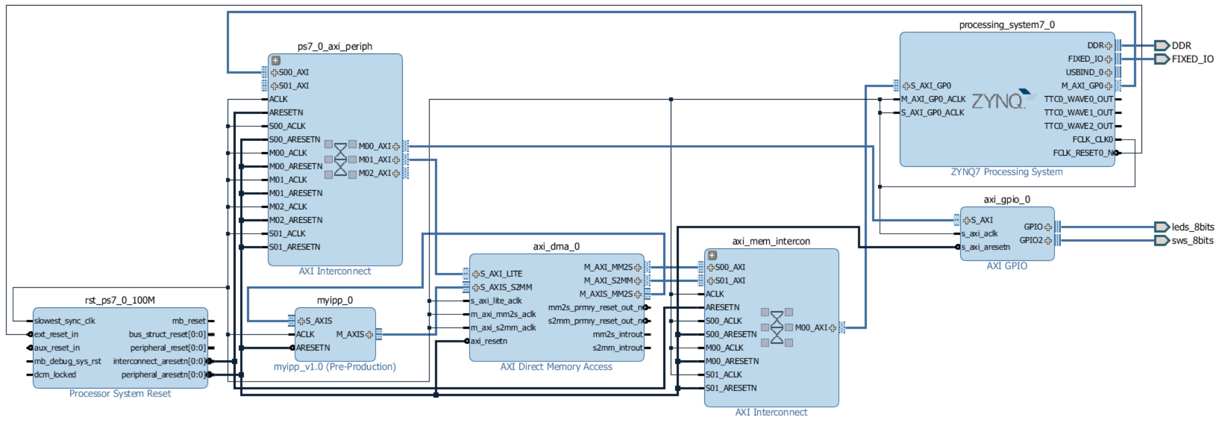

Fig. 1: Typical block diagram of the platform

Learning Outcome Survey Question	All the time (%)	Most of the time (%)	Sometimes (%)	Never (%)	Ave. Score (Out of 4)
Familiar with the design methodology for embedded systems	6	83	11	0	2.9
Translate system specifications into executable computation models using a high level language	17	67	17	0	3.0
Map these formal specifications into a register-transfer level HDL that can be implemented on an FPGA	22	67	11	0	3.1

TABLE II: Course Learning Outcome Survey

means that a lot is learned", "Labs and lectures had very little connection", "It took more time to debug the tool rather than learning a new concept in lab".

Some students felt that the knowledge of how various optimizations are done is not important, perhaps because modern tools hide much of the complexity from the designer. However, a good knowledge of the tools and optimizations is required to create an efficient system.

C. Challenges and Limitations

Dealing with state of the art tools can be frustrating for students at times, as the specific issue they face might not have a ready solution. Many students at undergraduate level might not have the experience to interpret the various error messages from synthesis tools. Another common issue students faced is missing out certain important connections, such as those to memory, which does not cause the tool to generate error messages. In this scenario where their own hardware design interacts with their own software, debugging can be tricky. Introducing advanced debugging tools such as the Integrated Logic Analyzer (ILA) of Vivado could be useful. Introducing the importance of timing analysis in the earlier part of the semester could be of help in solving timing issues. Using a cloud-based shared FPGA resource could address issues associated with the sharing of a boards among the two teammates.

V. Conclusions and Future Directions

A project for introducing students to hardware acceleration for ML on the edge for embedded implementations was explored. A neural network was accelerated using a Zynq FPGA over the course of a semester. Many students who were new to ML managed to pick up the basics. Students also gained experience in the state of the art synthesis tools. They could compare the results of their hardware implementation with pure software and HLS-generated accelerators. It gave them insights into computer system architectures as well as the interaction between hardware and software. Results from surveys indicate that most students felt they learnt the concepts well and achieved the learning outcomes.

A more thorough survey and evaluation of effectiveness will be conducted for the future versions of this project, especially with respect to the improvement in student understanding of ML basics. Interesting datasets and better defined specifications which allow for a fairer evaluation of projects, while retaining some flexibility and scope for creativity and visualization could be incorporated. The project could be extended further to incorporate reading of sensors as well as internet connectivity, thereby making it a true edge computing for IoT project.

References

[1] H. A. Hadim and S. K. Esche, “Enhancing the engineering curriculum through project-based learning,” in *32nd Annual Frontiers in Education*, vol. 2, 2002, pp. F3F–F3F.

[2] B. S. Bloom, M. Englehart, E. J. Furst, W. H. Hill, and D. R. Krathwohl, “Taxonomy of educational objectives: Handbook i,” *Cognitive domain. New York: David McKay*, 1956.

[3] S. Senay, “On the impacts of project based learning for workplace preparedness of engineering graduates,” in *2015 10th System of Systems Engineering Conference (SoSE)*, 2015, pp. 364–367.

[4] L. Huang and K. Ma, “Introducing machine learning to first-year undergraduate engineering students through an authentic and active learning labware,” in *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1–4.

[5] C. Wang, A. Dixit, A. Spanias, and S. Rao, “Introducing machine learning in a sophomore signals and systems course,” in *2019 IEEE Frontiers in Education Conference (FIE)*, 2019, pp. 1–4.

[6] D. Steinkraus, I. Buck, and P. Simard, “Using gpus for machine learning algorithms,” in *Eighth International Conference on Document Analysis and Recognition (ICDAR’05)*. IEEE, 2005, pp. 1115–1120.

[7] A. De La Piedra, A. Braeken, and A. Touhafi, “Sensor systems based on FPGAs and their applications: A survey,” *Sensors*, vol. 12, no. 9, pp. 12 235–12 264, 2012.

[8] Q. Shi, L. Xiang, T. Chen, and W. Hu, “FPGA-based embedded system education,” in *2009 First International Workshop on Education Technology and Computer Science*, vol. 1. IEEE, 2009, pp. 123–127.

[9] V. Kiray, S. Demir, and M. Zhaparov, “Improving digital electronics education with FPGA technology, pbl and micro learning methods,” in *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, 2013, pp. 445–448.

[10] S. Brown, “Leveraging FPGA technology for digital logic and embedded systems education,” in *2007 IEEE International Conference on Microelectronic Systems Education (MSE’07)*, 2007, pp. 159–159.

[11] D. Navarro, O. Lucía, L. A. Barragan, 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, pp. 43–47.

[12] P. D. Duffy and A. Bruns, “The use of blogs, wikis and RSS in education: A conversation of possibilities,” in *Online Learning and Teaching Conference, Brisbane*, 2006.

[13] A. Asuncion and D. Newman, “UCI machine learning repository,” 2007.