A Pedagogical Framework to Teach HW-SW Co-Design

Ishaan Biswas, Mark Johnson
School of Electrical and Computer Engineering, Purdue University
{biswas6, mcjohnso}@purdue.edu

Abstract
Modern embedded system designs have evolved to a codesign approach of designing the hardware and software for the system. It is important therefore to introduce the concepts of HW/SW (Hardware/Software) co design in the undergraduate curriculum to better prepare future engineers. This paper looks at a framework developed at Purdue University to provide a practical introduction to HW/SW codesign in the computer engineering curriculum. The hardware platform used is the DE2i-150 board from Terasic which links an Atom N2600 processor to an Altera Cyclone IV FPGA via dual PCIe channels. The framework has been developed as a lab series and a lab toolkit to enable students to prototype their applications on a real hardware platform. It is also meant as an enabling and a learning tool for enthusiasts and researchers to may want to use this platform to prototype their ideas and do not have the relevant technical depth. We mention our successes and challenges in this endeavor so far in enabling undergraduate and graduate students to prototype custom logic designs in a larger hardware platform. The work complements existing courses in which students learn register-transfer-level (RTL) design, verification, and integration of custom logic, existing IP, and a soft core processor on an FPGA. This framework was piloted in the fall 2014 semester in the ASIC design course where students were exposed to this framework for their course projects. Two student projects successfully prototyped on the DE2i-150 platform: a Laplacian Edge Detection algorithm and a Bitcoin mining algorithm.

Keywords - Computer Architecture, Embedded Systems, HW/SW, computer engineering, Register Transfer Level (RTL), SoC Design, DE2i-150

I. Introduction

Modern computer systems have evolved to incorporate a complex amalgamation of hardware and software to exploit the benefits of both approaches. In order to achieve maximum performance and optimal power a close HW-SW integration is essential to design systems. System design has moved from single chip CPU cores attached to off chip peripherals to a system on chip approach often with heterogeneous cores within the same chip. Such an evolution warrants a perspective that is beyond the boundaries of hardware and software and needs a more integrated approach. In fact, the need for this integration can be dated back to the early 1990's as illustrated by Wolf's survey of HW/SW co-design in embedded systems [2]. A more recent need for this approach has also been highlighted in the context of Integrated Circuit design by Wolf in [5].

The requirements Wolf describes continue to play out in the creation of large scale System-on-Chip (SoC) designs which are so prevalent in mobile devices. More recently, the exascale computing program from the Department of Energy (DOE) has given further thrust to HW/SW codesign in the field of supercomputing [6]. Given the pervasiveness of co-design in the current technology era, computer engineers will continue to be needed to work at the boundary of

hardware and software. It is imperative therefore to introduce the design principles and concepts of hardware software codesign in the undergraduate computer engineering curriculum, to prepare future engineers with the knowledge and skills required to build modern systems.

The computer engineering curriculum especially for undergraduates has evolved relatively slowly in comparison to the trends in the industry in terms of skills and knowledge required to build current day embedded and computing systems. Current day applications have moved beyond the traditional separation between hardware and software, and have evolved to a use of Multi-core, Soft Core FPGA's, ASIPs (Application Specific Instruction Processor) and FPGAs. The Computer Engineering undergraduate curriculum in most universities focus either completely on hardware with introductory courses focusing on basic digital design like Karnaugh maps, digital logic, Finite State Machines etc. Higher level courses focus on HDL, RTL synthesis, ASIC design and Computer Architecture. On the other hand courses focusing on embedded systems tend to be more software focused ranging from introductory topics like assembly programming and peripheral interfacing to interrupt processing, thread based control, process scheduling etc. In lieu of the present day needs of the industry it is essential to make students think of hardware and software as a complete design space instead of being disparate discrete solutions. Giovanni and Gupta in [1] and Wolf in [2] elaborate more on the premise, principles and challenges of HW/SW Co design.

This paper looks at a pedagogical framework to initiate students to the design principles of hardware software co design by providing a hands on approach to develop working prototypes of embedded applications while also exposing them to the use of a heterogeneous platform. The material developed and presented in this paper uses Altera's DE2i-150 as the hardware platform although the ideas and design principles are largely platform agnostic. Section II describes the background of this work and relevant courses developed at other universities focusing on hardware-software design. Section III illustrates a typical computer engineering course curriculum and how a course like this fits into it. Section IV describes our proposal and implementation of a lab series. We analyze our results in Section V. Section VI describes the road ahead and Section VII summarizes this work.

II. Relevant Work

A key challenge with teaching a hardware software approach especially in the undergraduate curriculum is the mental gap between the two approaches.

1. Concept of time. in RTL and C. While software requires a sequential approach to decomposing a problem (especially low level software), hardware requires a spatial decomposition. In other words, C is sequential, RTL is parallel.
2. Modeling and Implementation. In C what you write is what you get. In RTL what you write may not behave correctly in hardware due to synthesis artifacts. Difficult for beginners to develop a mindset towards understanding this

This fundamental difference causes a dramatic shift in how a designer approaches a problem and writes code in a hardware description language or in a language like C. Because of this inherent difference in writing code for hardware and software, most courses take a more focused approach either in HW or in SW. This is elaborated in greater detail in the next section.

Given the complexity of designing a course that involves system design with a HW/SW codesign approach, it is found that most courses take a theoretical approach and expose students to architectural exploration and focus on system modeling. While system modeling (in a language like System C) is essential in system design it gives little feel for the practical considerations trade-offs in hardware and software. We feel that courses which provide a hands on approach of designing systems with a HW/SW co-design approach are very effective in skill building. A beginner graduate level course at Purdue University by Prof. Anand Raghunathan (SoC Design)[7] is a good example of a course intended to introduce architectural choices and ideas in HW-SW Co-design while also providing a practical experience of designing a system with trade-offs in hardware and software. Providing complete architectural freedom in an undergraduate course is difficult both due to time and complexity involved in building a real system within a semester. Patrick Schaumont's course on HW-SW Co-design at Virginia Tech [3] stands out as a course in HW-SW co-design in the undergraduate curriculum where students learn and build working prototypes of systems using HW-SW partitioning.

III. Bridging the Gap

The undergraduate curriculum in computer engineering across most universities are strictly partitioned into hardware and software. The figure below shows the current computer engineering curriculum at Purdue University. This is also typical of the curriculum in most universities..

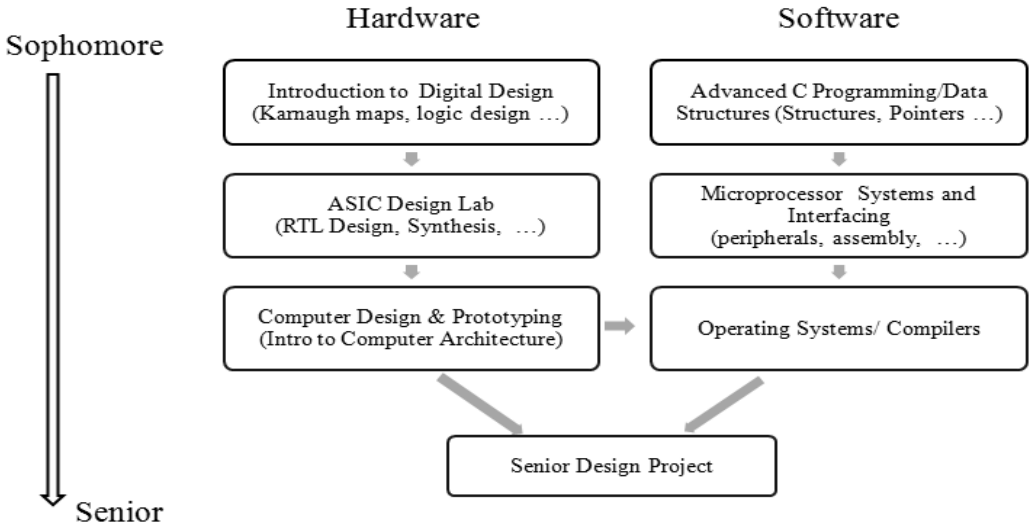

Figure 1: Computer Engineering Curriculum at Purdue University. Typical of most computer engineering curricula.

As is evident, courses focus strictly on either hardware or software which gives an illusion of strict demarcation to students towards their career choices. Students end up being biased towards either hardware or software not fully realizing their co-existence in real systems. This hampers

their development as rounded computer engineers who have a solid understanding of both. The fact that very few new college graduates are recruited into system design/architecture positions is representative of this lack of roundedness in the curriculum.

Our endeavor in ECE at Purdue University is to provide an inclusive background to our computer engineers by providing them the opportunity and infrastructure to design real system design by exploring both hardware and software. We feel the right time to introduce students to the ideas of HW/SW codesign is in the junior year while they still have enough time to explore this field before graduating. Therefore, the work presented here is meant to augment the ASIC Design Lab course. The major part of this course is an 8 week project where students design an application in RTL. They simulate and test their RTL using an RTL simulation tool and demonstrate their results at the end of the course.

Since the fall 2014 offering of the ECE 337 - ASIC Design course we have given students the option to prototype their designs on the DE2i-150 board. This involved the students interfacing their designs with I/O peripherals, system interconnect bus and other IP's. To enable students to achieve this we provide students with the necessary background and infrastructure which may fall slightly outside the scope of the course syllabus. However, it provides them an invaluable experience of designing and prototyping a real system. The approach and the infrastructure we provide is described in the following section.

IV. Designing a HW/SW Co-design framework for courses

Following the design principles of a similar course developed by Patrick Schaumont [3] we are designing a "lab series" with a supporting "lab toolkit" at Purdue. The hardware platform is fixed but is one which allows a lot of architectural freedom. We use the DE2i-150 [4] boards provided by Intel. This board features an Atom N2600 processor and a Cyclone IV FPGA on the same board. The Atom and the FPGA are connected by a dual channel PCIe bus. The DE2i-150 board is positioned as an educational platform and is perfectly suited for purpose such as this.
The lab series has been structured in an incremental manner. Students may either choose an application themselves or may be given a specific application to prototype. In any case, the application chosen should in general have the following characteristics.

1. Reasonably high computational complexity and inherent parallelism to take full advantage of the FPGA's resources.
2. Computation on the FPGA should be significantly higher than the communication overhead between the Atom and the FPGA via the PCIe lanes.

The lab series we propose builds on itself and takes a student through the entire flow of HW-SW co-design. The end product is a working prototype on the DE2i-150 board. The description of the lab series is as follows.

1. *Lab 1 - Profiling:*
    a. Students implement and run the application in SW in its entirety.
    b. Profile the code using gprof and get results.
    c. Analyze the results of profiling and understand parallelization opportunities

    The application software should be preferably written in C to make the translation of specific functions into HW manageable.
2. *Lab 2 - HW Design:*

a. Build a hardware accelerator for chosen function(s) from Lab 1 in Verilog
b. Hardware modules should be built with interface logic to the Avalon (Altera's System Interconnect protocol) bus.
c. Simulate the design in Modelsim using the available BFM models for the Avalon bus.

3. *Lab 3 - System Integration:*
   a. Introduce Qsys - System Integration tool in Quartus.
   b. Integrate the hardware accelerator with other peripheral interfacing IP's as a Qsys Subsystem.
4. *Lab 4 - Kernel programming.*
   a. Introduce the basics of kernel programming and drivers in linux.
   b. Students build selected kernel modules for the PCIe interface to the FPGA. Completed modules for other required functions and skeleton for the selected kernel modules for the FPGA will be provided in the lab toolkit
5. *Lab 5 - HW/SW Integration.*
   a. Use the hardware and software built in previous labs to prototype the entire application.
   b. Full system integration and application demo.

The lab toolkit mentioned above will comprise of the following:

1. Tutorial to introduce hardware software design on the DE2i-150 board. Tutorial uses a sample design to demonstrate a simple functionality which uses the Atom and the FPGA.
2. Tutorial to use the BFM model of the system interconnect (Avalon) in a simulation environment (Modelsim)
3. Sample hardware modules to illustrate the interfacing logic with the Avalon bus.
4. Tutorial to introduce the Qsys tool. Tutorial builds an entire system in Qsys.
5. Tutorial on loadable kernel modules and drivers in Linux. The kernel functions required to communicate with the FPGA will be provided as a library.
6. A guide for debugging on the FPGA. The guide will provide debugging tips on the FPGA hardware.
7. Commands cheat-sheet for beginners to linux for the relevant commands and utilities that may be needed through the course of the project.

The goal of this lab series and lab toolkit is twofold.

1. It will enable universities to develop a lab series or a project based course that provide a more hands on approach towards building a system with a HW/SW approach.
2. It will enable enthusiasts/researchers to use this platform to design and develop their own ideas. This may be extremely valuable to researchers in other related fields like biomedical engineering, mechanical engineering etc who may have some background but not enough technical depth to implement their ideas on such a platform.

V. Results

As mentioned in Section III, the lab toolkit was initiated in the fall 2014 semester. An initial version of the toolkit was provided to the students to enable them to use the DE2i-150 boards for their projects. The bare essentials of the lab toolkit described in the previous section was part of this release. This initial lab toolkit comprised of:

1. Sample reference design implementing basic communication between the Atom and FPGA. The reference design included both the hardware and the software framework.
2. Verilog modules to interface with the System Interconnect bus.
3. Detailed tutorial to describe concepts, tool chain and debugging tips.

Two teams went on to successfully prototype and demonstrate their application on the board. A Laplacian Edge detection Image processing core and a Bitcoin mining calculator were the two applications that students prototyped on this platform. While the Laplacian Edge detection project implemented a convolution of a 3x3 Laplacian filter with an input image, the bitcoin mining project implemented as many as 16 SHA-256 hashing cores on the FPGA. The students implemented a layer of software in C as the frontend interface and used functions provided as a library to offload the computations to the FPGA.

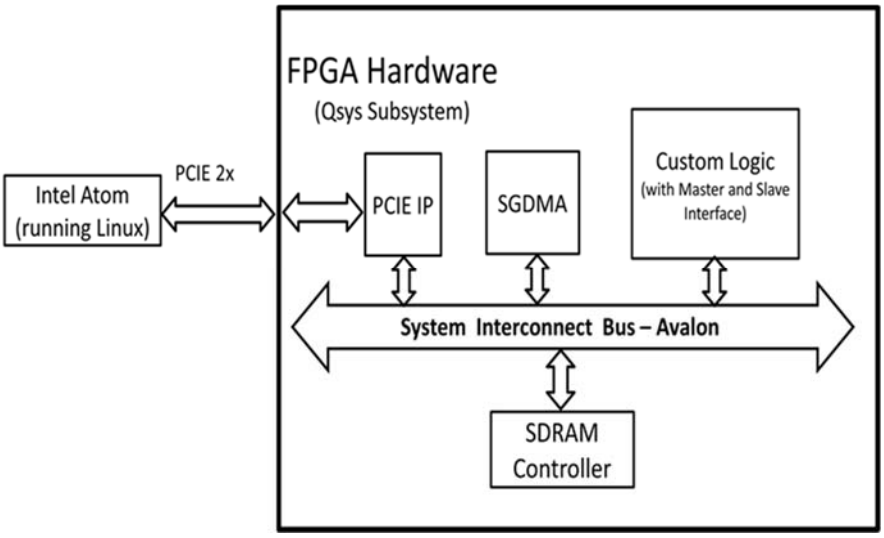

Figure 2: Base hardware architecture on the FPGA provided to students.

As part of the toolkit we provided a base hardware architecture as illustrated in Figure 2. The Base architecture consists of
1) A PCIe interface IP core (available as a soft IP)
2) An SDRAM controller (available as a soft IP)
3) A Scatter Gather DMA for large memory transfers (available as a soft IP)
4) A Custom Logic block with a Master and Slave interface. Student integrated their specific design into this block. The custom block also has a set of registers which can be used as status and control registers.

Students were free to use this base platform as was best suited for their specific application. For instance, for the Laplacian edge detection algorithm, the students used the slave port of the "custom logic" module to write control bytes in software to initiate the computation and another status register (that the software polled from) to inform completion of a task by their module. A similar approach was also used for the bitcoin mining project to initiate and indicate completion of transactions.

Providing this hardware software co-design platform allowed students greater flexibility in the front end. The software for this platform consisted of a C program that runs on the Atom processor. The Atom processor runs a version of Linux Yocto and provides a POSIX interface. Students were provided with a library of functions that implemented transfer of data to the FPGA via the PCIe driver. For instance, the bitcoin mining project had a front end (in C) that connected to the bitcoin network to receive and relay packets to the FPGA and in turn from the FPGA to the network to check for a correctly mined input set. The Laplacian edge detection team used a simpler layer of software to write a start byte into one of the registers to initiate a task and polled on another register to indicate completion. The software layer was also used to write an input image into the SDRAM such that their Edge detection core in the Custom Logic module could operate on the image data.

Given the scope of the projects and the timelines in the previous offering of this course, the software interface used by students was more to provide flexibility rather than explore complex hardware-software performance tradeoffs. As this platform and infrastructure evolves through this work (and hopefully elsewhere as well) we envision more advanced projects and ideas being successfully implemented on this platform thereby pushing the boundaries of this platform.

VI. Future Work:

We are currently designing the specifics of the labs mentioned in the Lab Series. The Lab series will be made available to the public by May 2015. This will include lab handouts, a sample design and working prototypes that may serve as solutions to course administrators. We plan to make the entire lab toolkit available to students for the spring 2015 offering of the ECE 337 course at Purdue where this was first piloted. Based on the feedback and necessary modifications the Lab toolkit will be released for the public along with the Lab Series. This is also planned to be used for the next course offering of ECE 695r - System on Chip Design - a graduate level course at Purdue.

VII. Summary

Incorporating HW/SW codesign in the undergraduate curriculum is rapidly becoming necessary considering the trends in the industry and even in research. However the challenges involved in developing a course structure which incorporates this in a practical way is often overwhelming and a big roadblock. Through the work presented in this paper we hope to enable more universities/professors involved in this domain to benefit and expand or create new courses or projects to introduce students into HW/SW codesign especially in the undergraduate curriculum. Through our experience we have found that a step by step approach to introduce students to system design with a co-design approach makes it easy for students to grasp concepts and skills. The lab series that is being developed as part of this work is based on this principle. Figure 3 below illustrates the different components of the lab toolkit and how it fits into the system design effort.

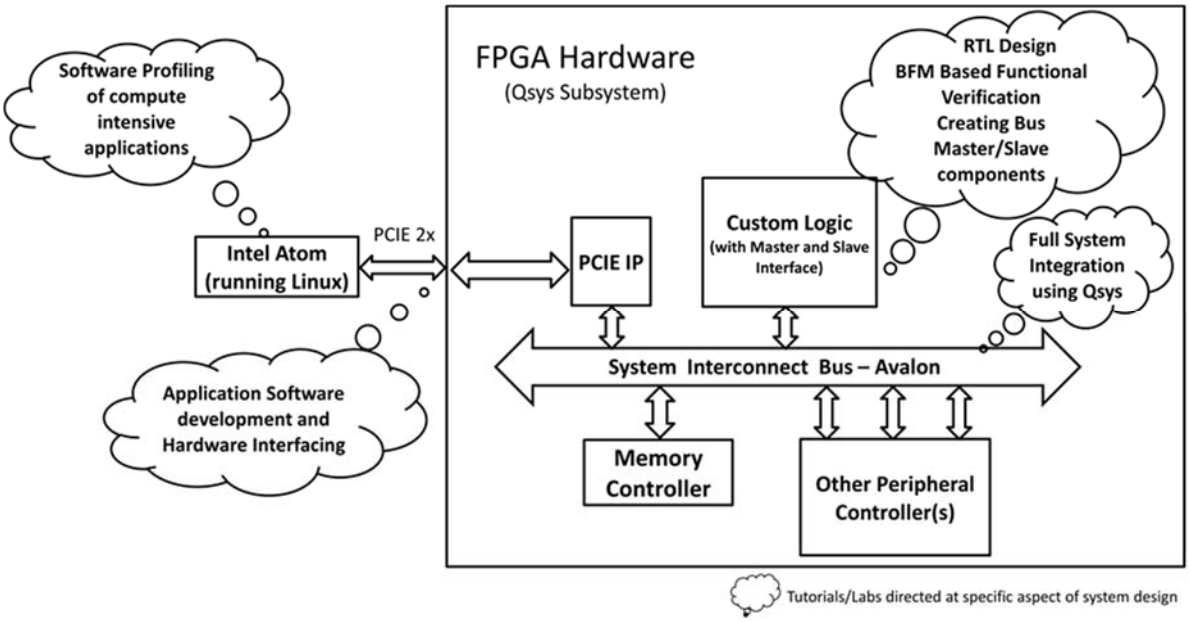

Figure 3: Summary of the lab toolkit and how it aids the design effort.

While it is challenging to build a hands on course in HW-SW co-design it is extremely valuable for students to prototype their designs on real hardware. As one student pointed out in his feedback "Putting our project on the FPGA was a difficult, frustrating, but ultimately rewarding experience when it actually worked". Through this work we hope to reduce the pain while making system design a fun and enriching experience.

Acknowledgements

The authors are grateful for support provided by Intel Corporation for this work. We are also grateful to the students of the ECE 337 course at Purdue University who provided us with valuable feedback regarding the initial version of the lab toolkit that they used for their projects. Special thanks also go out to James Chen and Zaiwei Zhang for their contribution in developing this framework.

References

1. Giovanni de Micheli and Rajesh K. Gupta, “Hardware/Software Co-design”, in Proc. IEEE, Vol. 85, No. 3, March 1997.
2. Wayne Wolf, "Hardware-software co-design of embedded systems", Proc. IEEE, Vol. 82, No. 7, July 1994.
3. P. Schaumont, "A Senior Level Course in Hardware/Software Codesign," IEEE Transactions on Education, Special Issue on Micro-Electronic Systems Education, 51(3):306-311, August 2008.
4. DE2i-150 homepage (2015, Jan 30). Retrieved from http://www.terasic.com.
5. Wolf, W, "A decade of hardware/software codesign, Computer", Volume 36, Issue 4. pp. 38-43. 2003.
6. J. Shalf, D. Quinlan, and C. Janssen, "Rethinking hardware-software codesign for exascale systems," IEEE Computer, vol. 44, no. 11, pp. 22-30, 2011.
7. ECE 695- System on Chip Design by Prof. Anand Raghunathan at Purdue University (2015, Jan 30). Retrieved from https://engineering.purdue.edu/ECE/Academics/