Computer Organization and Design Course with FPGA Cloud

Ke Zhang, Yisong Chang, Mingyu Chen, Yungang Bao, Zhiwei Xu
State Key Laboratory of Computer Architecture, ICT, CAS
University of Chinese Academy of Sciences (UCAS)
Beijing, China
{zhangke,changyisong,cmy,baoyg,zxu}@ict.ac.cn

ABSTRACT

Computer Organization and Design (COD) is a fundamentally required early-stage undergraduate course in most computer science and engineering curricula. During the two sessions (lecture and project part) of one COD course, educational platforms play an important role in cultivating students' computational thinking, especially the ability of viewing the hardware and software in a computer system as a whole (computer system thinking ability for short in this paper). In order to improve teaching quality, in this paper, we discuss the deployment of an inexpensive in-house Field Programmable Gate Array (FPGA) cloud platform, which can provide students with hardware-software co-design methodology and practice. The platform includes 32 FPGA nodes and the scale can be dynamically changed. Each cloud node is heterogeneously composed of an ARM processor and a tightly-coupled reconfigurable fabric to provide students with hands-on hardware and software programming experiences. We illustrate our efforts to make the FPGA cloud as an easy-to-use resource pool to elastically support a class with 92 undergrads via Internet access and to monitor students' experimental behaviors. We also present key insights in our teaching activities that indicate such appliance is feasible to provide practice of both basic principles and emerging co-design techniques for students. We believe that our cost-effective FPGA cloud is of significant interests to educators looking forward to improving computer system-related courses.

CCS CONCEPTS

• **Applied computing → Education**; • **Computer systems organization**; • **Hardware → Reconfigurable logic and FPGAs**; **Hardware-software codesign**; • **Networks → Cloud computing**;

KEYWORDS

FPGA Cloud, Computer Organization and Design, Computer System Education, Hardware-Software Co-design

ACM Reference Format:
Ke Zhang, Yisong Chang, Mingyu Chen, Yungang Bao, Zhiwei Xu. 2019. Computer Organization and Design Course with FPGA Cloud. In *SIGCSE'19: 50th ACM Technical Symposium on Computer Science Education, February 27–March 2, 2019, Minneapolis, MN, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3287324.3287475

SIGCSE'19, February 27–March 2, 2019, Minneapolis, MN, USA

ACM ISBN 978-1-4503-5890-3/19/2…$15.00
https://doi.org/10.1145/3287324.3287475

1 INTRODUCTION

Computer Organization and Design (COD) is an early-taught undergraduate course that typically covers fundamental CPU design principles, instruction set architecture (ISA), assembly language and input/output interfacing in both lecture and project sessions. As a key component in ACM computer engineering curriculum [1], COD is essentially required to teach students the concepts of hardware and software as well as the synergy of both parts in a computer system. Moreover, COD gradually attracts attention in computer science undergraduate programs (e.g., ACM's 2013 curriculum guidelines [2]) as either a required course or an elective to show an abstraction of hardware from a programmer's perspective.

In the COD lab course, students are required to fully understand and carefully apply abstract concepts they learnt in lectures to practicing ordinary low-level software programming in assembly and C language or even advanced hardware logic design. Therefore, compared with other traditional lecture-centric courses, the COD course largely depends on educational platforms to provide real and interactive engineering environments for students to conduct hands-on experiments. As a result, it is crucial for educators and instructors to elaborately design and massively use educational platforms in both lecture and project sessions of a COD course in order to cultivate students' computational thinking [34], especially ability of viewing the hardware and software in a computer system as a whole (*computer system thinking ability* for short).

A) Using new hardware in COD platforms

The evolution of modern computer systems has been widely inspired by emerging heterogeneous acceleration and domain-specific architecture [14]. This trend emphasizes the key importance of *hardware-software co-design* to all participants in the field of computer system [3]. Therefore, besides basic software programming, students are strongly recommended to study hardware design principles and the co-design concepts as early as possible in computer system-related courses and especially in the early-stage COD class, improving their *computer system thinking ability*. Given the state-of-the-art heterogeneous computing systems, we argue that the key challenge to further improve teaching effects by manifesting system-level *hardware-software co-design* in COD courses lies in the exploitation of an appropriate educational platform that provides heterogeneous co-design capability for students.

In traditional COD courses, software ISA simulators [7, 17, 22, 32] are widely used as experimental environments for students' writing

[1]First authors Zhang and Chang have made equal contributions to this work.

and executing low-level programs. However, such virtual platform not only inherently restricts execution time and flexibility, but also exhibits an incomplete system view to students due to the shortage of real hardware components. The avant-grade single-board computers based on System-on-Chip (SoC) [28, 31] provide software execution environment on real hardware platforms but lack capability to integrate additional hardware accelerators. Similarly, despite suitability for hardware-related experiments, Field Programmable Gate Array (FPGA)-based development boards [9, 26] fail to support complicated software development.

Fortunately, a heterogeneous FPGA (e.g., Xilinx's Zynq UltraScale+ MPSoC [36]) which has been just commercially available can overcome all the above-mentioned shortcomings. This hybrid device tightly couples a high-performance, software-programmable hard core SoC (commonly based on ARM processor) with a reconfigurable logic fabric on the same chip die. Such configuration has been confirmed in setting up a feasible experimental platform for embedded system-related courses [25, 33]. We believe that heterogeneous FPGAs are more appropriate and cost-effective as educational platforms to demonstrate the concept of hardware-software co-design and deliver related experimental environment in COD courses.

B) Introducing cloud computing to COD platforms

Instructors of COD courses used to hand out each student one hardware experimental kit that would be connected to his or her laptop during projects. Due to different progresses of students' experimental projects, 1) status of dozens of kits are diverse, and 2) the number of simultaneously in-use kits is frequently changed. These two factors lead to a low utilization rate of the experimental kits. In addition, each student conducts projects privately in the laptop with an attached hardware kit, making his or her experimental behaviors entirely invisible to instructors. Motivated by emerging cloud computing technology, we believe that these situations would be improved if we could make experimental kits as a pool of remotely accessed resources that can be unilaterally provisioned and automatically monitored. Some preliminary cloud-based experimental platforms [10, 24, 37, 38] were established to provide remotely accessed development kits for students.

C) Why not commercial public FPGA cloud?

Cloud vendors have provided FPGAs as public acceleration resources for users to rent (e.g., Amazon's AWS F1 instance [4]). From our perspective, such public FPGA cloud exhibits several drawbacks in COD teaching:

1) Expensive economic costs to deploy the same number of FPGAs as in our in-house FPGA cloud within a semester. Note that the purpose of public FPGA cloud is mostly for accelerating web services, so the FPGA devices used in public cloud are always high-end and costly products,

2) Complicated FPGA development and configuration flow in public cloud that requires students to learn additional skills non-related to the COD course,

3) The execution environment of FPGA cloud is mainly based on virtual machines (VMs) running on the remote server, introducing unpredictable penalties and inter-VM contentions for various users.

D) Putting it all together

In this paper, we propose an in-house heterogeneous FPGA-enabled cloud platform with multiple nodes (32 for our current implementation) connected via Gigabit Ethernet for teaching COD courses in both lecture and project. Each node is an inexpensive custom circuit board based on Xilinx Zynq UltraScale+ MPSoC which provides both hardware and software programmability in a single board for students with the cost of only $950. We make each physical node with a fully-fledged Linux kernel and related environment atop the ARM SoC as a general cloud resource in which students can configure FPGA fabric, interact with logic modules in FPGA and launch software applications. All available resources are managed and scheduled via Ethernet by a commercial-grade cloud computing framework on a front-end x86 server. A pre-configured user-end VM running on each student's laptop provides standard toolchains of both hardware and software development. Unlike the server-end VMs for execution in public FPGA cloud, in such user-end VM, students use a simple script-based design flow to compile their FPGA designs and software programs offline, request an experimental cloud resource, and transparently upload the generated FPGA configuration files as well as executable binaries to the requested resource for execution via campus network. In order to fully achieve teaching objectives in both computer science and engineering curricula and comprehensively demonstrate the concepts of *hardware-software co-design* on our FPGA cloud, we propose an exploratory assignment on ISAs and a live demo about details of network packets processing in our COD lectures, as well as experimental projects that cover both fundamental principles and emerging techniques. Some key advantages our FPGA cloud provides for teaching COD course include:

Ease of use and maintenances. The cloud platform allows students to conduct projects as using local experimental kits without resource contention between each other, making it possible to carry out performance-related studies on remote hardware. Moreover, our FPGA cloud platform can be simply managed by teaching assistants of the COD course, without additional engineering efforts.

Elasticity and low cost. Resource pooling dynamically adjusts the number of available resources as per students' demands to afford a class of 92 students to conduct projects within the scope of only 32 nodes, significantly saving the infrastructural budget.

Teaching Assistance. Detailed execution behaviors of each student are monitored and logged in the cloud. Such information indicates learning outcomes of one student, helping instructors make individual teaching policies for different students.

2 BACKGROUND

Conventional COD courses mainly concentrate on how to teach students hardware abstraction from a software programmer's point of view, thus emphasizing the importance of ISAs and relevant assembly programming in experimental projects. With the minimum economic cost and educators' efforts, the software ISA simulator is a good choice for students to use, which is even leveraged today in some representative COD-like courses [29, 30]. As the rise of inexpensive hardware single-board computers such as Raspberry Pi, some educators have asked their students to conduct experiments on such platforms [21, 27].

On the other hand, emphasizing hardware design in COD course offers a new bottom-up perspective to students to fully understand computer systems instead of pure ISAs. In experimental projects of such class, students are required to implement their preliminary

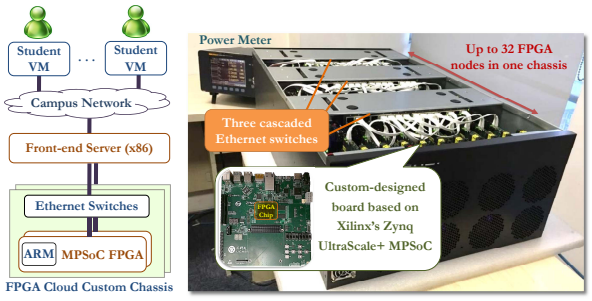

Figure 1: Our proposed FPGA cloud architecture and one prototype chassis with 32 individual physical FPGA nodes that is large enough to support our COD class with 92 students.

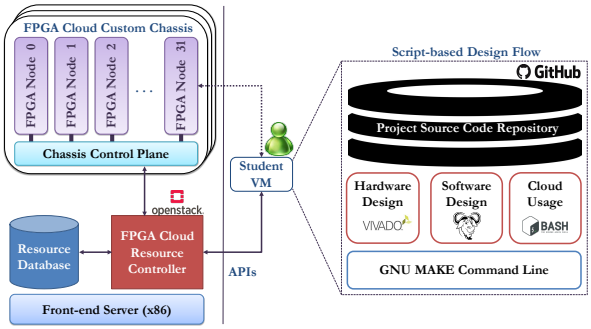

Figure 2: Efforts to provide an easy-to-use FPGA cloud service. An OpenStack-enabled resource manager (left part) allows students' remote access to the allocated FPGA node (arrowed dash line). A scripted design flow on each student's VM (right part) simplifies FPGA development and usage.

CPU cores using a hardware description language (HDL) such as Verilog HDL. However, these CPU cores are always verified just in HDL behavioral simulation and logic synthesis rather than implemented to run on real hardware [18, 29, 30]. Although real-world FPGAs are involved in some classes [5, 6, 8, 11, 13, 15, 16, 23], it is still extremely difficult for early undergrads to independently afford the task of organizing a minimum runnable system that wraps their CPU cores to various I/O components (e.g., memory, UART, etc.) with standard interfaces. This task always requires a significant amount of FPGA-specific knowledge and sufficient hardware design experiences that are out of the range of COD course.

In summary, existing experimental platforms in COD-related courses rarely offer students a real-world executable system-level environment with an easy-to-use interface to help them fully understand computer systems.

3 FPGA CLOUD IN COD COURSE

In this section, we introduce our proposed FPGA cloud architecture as well as one prototype chassis, and related methods to provide an easy-to-use interface for students. Based on the FPGA cloud, we discuss improvements in both lectures and projects of our COD course to emphasize *hardware-software co-design*.

3.1 Architectural and Technical Overview

Figure. 1 depicts our proposed FPGA cloud architecture and one in-house chassis in which at maximum 32 nodes based on custom circuit boards using Xilinx's Zynq UltraScale+ MPSoC FPGAs are interconnected via three cascaded Ethernet switches. More chassis can be cascaded for larger class. The design objective of our FPGA cloud is to effectively abstract platform details and agilely help students to leverage the FPGA cloud in campus via an user-friendly Ethernet-accessible interface, avoiding unnecessary efforts that negate students' concentration on the COD course.

First, we leverage and modify a commercial-grade cloud resource management framework—OpenStack [20] to provision each physical node in the custom chassis as a general cloud resource. As shown in the left part of Figure. 2, the modified OpenStack controller runs on the front-end x86 server to communicate with hardware control planes in FPGA chassis and provides APIs to allow students to request and release resources on-demand using either webpage or command line via Ethernet. OpenStack maintains a database of information of all physical nodes in the FPGA cloud, both in use and available for elastic allocation and release.

Second, in order to remove numerous FPGA design complexities for students, we implement a bunch of pre-built scripts that runs inside the commercial FPGA design tool (i.e., Xilinx's Vivado [35] in our current COD course) in the VM to automatically setup an FPGA project with static logic of complex I/O components surrounding students' synthesizable HDL designs and to strictly execute key design flow steps, making students purely concentrate on their HDL design that are strongly related to the contents of COD course. We further spread such script-based design flow via standard GNU `MAKE` command line to uniformly cover FPGA hardware development, software compilation and FPGA cloud usage (right part of Figure. 2). By this means, complicated experimental procedures are abstracted into one single and simple `MAKE` command, saving a large amount of efforts for students to carry out projects via the FPGA cloud. Moreover, all relevant source files and scripts in an experimental project are organized into a repository hosted on GitHub Classroom [12]. Therefore, all students' behaviors are committed in the repository to trace variations of one student's ability in different phase of COD course and make just-in-time guidance.

3.2 Usage in Lectures

Rigid oral explanations in conventional COD lectures leave students far away from real systems, making it difficult to understand some knowledge units that requires hardware-software collaboration. Although hands-on experiments relieve such situation to some extent, they are still unable to cover all relevant key points mentioned in the lecture class. In order to fill such gap, we attempt to propose an exploratory assignment as well as a live demo based on our FPGA cloud as supplementaries in lectures of our COD course.

Exploratory Assignment. As a fundamental interface between hardware and software, ISA is a strongly required knowledge unit in COD courses. This point is especially important to computer science students to understand hardware and computer system, as they have rare opportunities to engage in hardware-related courses.

Consequently, we push students to make a comprehensive comparison among ISAs of x86, ARM, MIPS and the emerging RISC-V

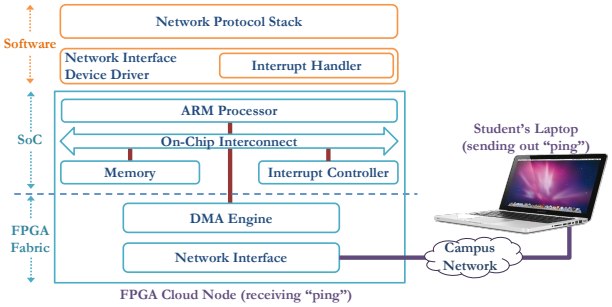

Figure 3: Full system stack on an FPGA cloud node to demonstrate system-level details when processing a "ping" network packet as a live demo in lectures of our COD course.

in basic assembly blocks, segmentations in binary object and executable files, endianness, application binary interface (ABI) and stack management based on practical compilation as well as analysis of binary files. Students use the standard `GCC` compilers and related `binutils` for x86 and ARM ISAs in the operating systems on each student's VM and the requested cloud resource respectively, as well as `GCC` cross-compiling environments for MIPS and RISC-V pre-deployed in the VMs. We provide a series of standard micro-benchmarks written in C language and encourage students to custom-design some other programs for comparison.

Live Demo. Advanced input/output interfacing, mainly including direct memory access (DMA) and interrupt, is an essential system concept that deeply relies on hardware-software collaboration. Due to shortage in real application scenario, it is very difficult for students to understand this key concept just according to descriptions in COD textbook. Moreover, such knowledge unit is rarely involved in a COD experimental project since students are required to learn a huge amount of additional knowledge out of the range of COD course. As a result, in our lecture in COD course, we build a live demo about details of network packet processing on real-world hardware-software full system stack inside our FPGA cloud platform, which is interactively visible to students for observation of system-level I/O processing behaviors. To make the demo easily to understand, we select `ping`, a common networking utility students are familiar to use in daily life as an illustrative example. We also abstractly introduce some easy-to-understand concepts of network protocol stack and device driver in our COD course in advance.

Figure. 3 briefly describes the full system stack in one FPGA cloud node and the hardware network processing path in a student's laptop with the VM. For students' hardware visibility, we implement a network interface and its related DMA engine in the FPGA fabric of each physical cloud node. System-level information of this newly-involved network interface is configured during resource allocation. Meanwhile, students are enabled to observe operations in ARM-end operating system via traces logged by pre-modifications in kernel-level network device driver and protocol stack.

After obtaining the IP address of an allocated FPGA cloud resource, one student executes a GNU `MAKE` command in the VM to launch the FPGA development tool GUI that automatically prepares the environment for remote signal probing of key hardware ports from network interface and DMA engine. The student then sends

Table 1: FPGA cloud-inspired COD lab projects

#	Project name and brief description
1	**Basis of Assembly Language:** Function call routine and stack management
2	**Basic CPU Design:** Register file, ALU and single-cycle CPU
3	**Advanced CPU Design:** Multi-cycle CPU with interface to DDR memory and UART
4	**Performance Evaluation:** Hardware performance counter design
5	**Memory Hierarchy (Optional):** Instruction and data cache
6	**Domain-Specific Architecture (Optional):** Custom deep convolutional neural network accelerator with DMA interface

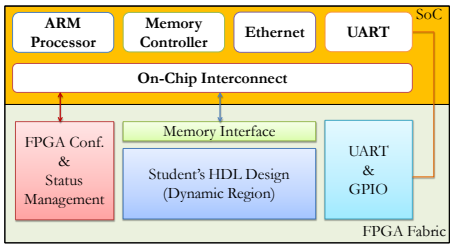

Figure 4: A template architecture in FPGA cloud node to facilitate easy integration of students' HDL design into a minimum runnable system.

out a "ping" packet to the obtained IP address. When the "ping" packet is injected into the network interface of the cloud node, variations of hardware signals are displayed in the GUI in time, allowing students to observe packet movements in hardware data path. Traces in the operating system allow students to clearly understand life cycles of network packets and motivate students to obtain insights in the approaches of hardware-software interactions.

3.3 Usage in Labs

Based on our FPGA cloud, we redesign the projects in our COD labs as shown in Table. 1. The main target is to push students to make their outcomes in these projects executable in an actual hardware-software environment, instead of a pure simulation.

In the first experimental project, students are disciplined in software programming abilities within hybrid assembly and C language on the ARM processors of cloud nodes, which emphasizes the most fundamental concepts of function call routine and runtime stack management. In this manner, students are driven to deeply understand the hardware interface from a programmer's perspective. If educators would like to extend such project arrangement with more software-oriented experiments, existing projects based on conventional platforms like Raspberry Pi or x86 are also inherently supported in our FPGA cloud without modifications to run atop the high-performance ARM processor. Moreover, exclusive occupancy of one physical cloud node guarantees students to conduct further software performance studies.

In the following projects, students are guided to implement synthesizable hardware components in the FPGA fabrics and make it executable with software collaborations. In order to simplify students' design efforts in such hardware-centric projects, we propose a template architecture in our FPGA cloud node as shown in Figure. 4. Based on such template, students are encouraged to

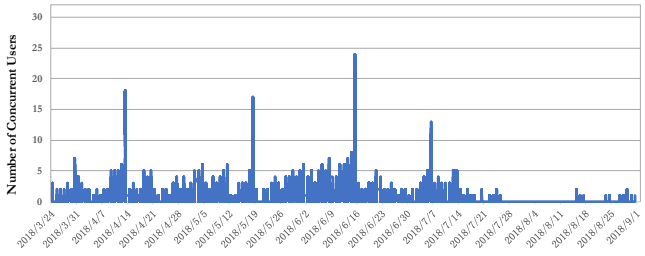

Figure 5: The number of in-use nodes in the FPGA cloud measured every 20 seconds during the 2018 Spring semester in which our COD course carried on, and short Summer semester to conduct students' research projects, respectively. The X-axis is cataloged into each day during both semester.

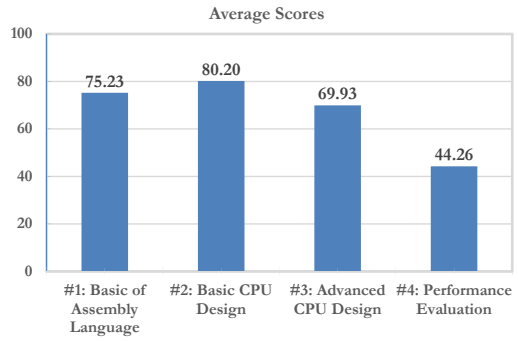

Figure 6: Average score (100-point grading system) of each required project in our class as declared in Table. 1.

implement custom CPU cores in the dynamic region based on either conventional MIPS or emerging RISC-V ISA to practice fundamental CPU design principles. Custom cores share the same address space of DDR4 memory populated on the FPGA circuit board with the ARM processor. In this manner, ARM processor is leveraged as an auxiliary engine that is easily driven by students in their VM-based development environment, taking charge of FPGA configuration, data and program loading, UART communication, status management and hardware debugging of students' cores. Undergrads are also required to implement a C language-based software framework running atop custom CPU cores to allow applications access memory-mapped hardware UART and GPIO interfaces.

Students are cultivated with the quantitative approach via a project towards design of hardware performance counters. Based on performance evaluation results, students optionally optimize the memory hierarchy of their CPU cores via instruction and data cache. Students also electively involve in design of deep convolutional neural network accelerators in the dynamic region of template architecture as well as implement an ARM-end full software stack incorporating with accelerators. By this means, students acquire preliminary skills in heterogeneous acceleration and domain-specific architecture design.

After disciplined with these experimental projects in a full semester, outstanding students in our COD course afford to further conduct research projects independently on our FPGA cloud in the short terms during summer vacations, some of which were highlighted in one tutorial [19] we held in ISCA 2018 (one of the flagship annual academia conferences in computer architecture).

4 TEACHING PRACTICE AND DISCUSSION

In this section, we mainly discuss some key observations we acquire from teaching practice in our COD course with the FPGA cloud in the Spring and Summer semester of 2018 to illustrate the efficiency and feasibility of our FPGA cloud as an easy-to-use infrastructure in motivating students to study computer system-related courses.

The number of concurrent users (Y-axis) monitored in our cloud appliance is chronologically plotted in Figure. 5. We used Linux shell scripts persistently running on the front-end server with our OpenStack-based resource controller to acquire how many students were using this platform every 20 seconds and log the data for the whole 2018 Spring and Summer semester (160 days, 690K log entries in total). In this figure, several spikes can be recognized around 2018/3/31, 4/14, 5/19, 6/16 and 7/7, which are due dates for each project. The maximum number in this plot is 24, which means 24 FPGAs in the cloud were simultaneously occupied and utilized by 24 students. Therefore, in the case of a COD class with 92 students, one 32-node FPGA cloud chassis can meet our course requirement. One more thing to be noticed is that several single-digit numbers were recorded from 8/11 to 8/31 in the Summer semester. The reason is that after the COD course, three outstanding students volunteered to join our research group and were actively involved in one of our research projects. Therefore, utilizing the FPGA cloud to teach in our computer system course can successfully help some early undergrads engage in computer system-related research.

Figure. 6 shows the average score of each required experimental project in the class of 2018 Spring semester. With the help of our easy-to-use design flow and user interface of the FPGA cloud, all experimental projects are feasibly carried out. In the beginning of this course, students started to be acquainted with this cloud platform in Project #1 and #2 that target programming in assembly as well as basic CPU design respectively and got good scores. In Project #3, students' average grade decreased as the integration of their custom CPUs into real-world environment increases difficulties in CPU design and HDL implementation compared with the previous projects. This situation was even worse in Project #4 since more sophisticated benchmarks were involved for performance evaluation exposing more design defects in students' CPU design and implementations. Another reason causes the lowest average grade in Project #4 lies in that existing debugging environment in our FPGA cloud is not efficient enough for students to fix complex bugs of their CPU cores in a four-week period. Moreover, grades of optional projects are ignored as there were few students to engage in such projects as important dates of these two projects were much close to final exams of other courses.

We asked our students to complete two surveys respectively in the middle and end of spring semester. The survey results from Table. 2 show that the students' satisfaction with the usage of remote FPGA cloud platform (72/82=87.80%) was much higher than using FPGA cards located in our computer lab (22/82=26.83%). As demonstrated in Figure. 7, most of students preferred to use cloud rather than local hardware kits due to the ease-of-use of our FPGA cloud

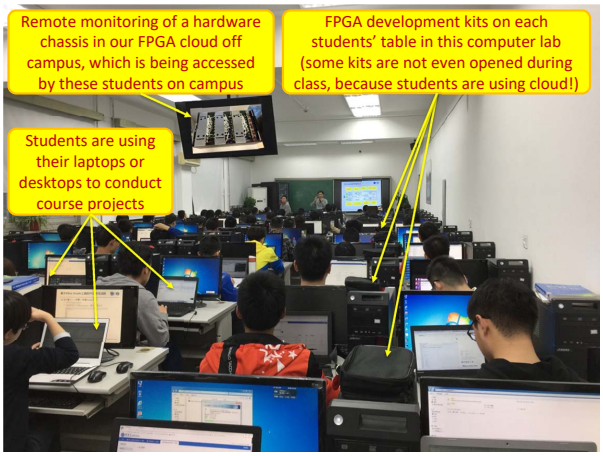

Figure 7: An instructor and a teaching assistant are leading the 92 students in our COD course to conduct experimental projects in a computer lab.

Table 2: Survey results of satisfaction to the FPGA cloud in our COD course. Note that 10 students who are minor in computer science are not required to conduct lab projects.

# of students	Remote FPGA Cloud Platform	Local FPGA Development Kits
who were fond of using	72	22
who never used	2	19
who completed lab projects	82	82
in the class (lectures + labs)	92	92

Table 3: Student engagement in three academic fields represented by student counts, which shows that how the degree of learners' interest in these areas changes during the semester ("+": favor, "++": strong interest, "−": dislike).

	Computer Science Major	Computer Architecture and System	Research in Computer Science
+ → ++	21	24	16
++ → ++	51	18	36
− → +	8	32	16
+ → −	10	10	11
− → −	2	8	13

platform. In terms of teaching effectiveness for both lecture and lab part of our COD course, as shown in Table. 3 we can notice that 55.43% (51/92) of the students have been sustainably interested in the major of computer science and 45.66% ((24+18)/92) for computer architecture & system throughout this course. More importantly, by leveraging the FPGA cloud platform in both lectures and labs, our teaching approach gradually changes additional 34.78% (32/92) students' attitude towards computer system-related knowledge from dislike to favor. Therefore, over 80% sophomore undergrads would like to take more computer system-related courses in the following semesters. Last but not least, in the third column of this table, an increasing number (16+16) of these early undergraduates are willing to conduct computer system-related research in the future and 3 people joined our research lab during this summer. As instructors, we should also pay attention to the 10 or 11 students who became unfriendly to CS in the end of our course. One student replied "Coding in HDL was terrible and hardware debugging was also tedious". Although this phenomenon exists for certain learners, we need to mitigate this side effect in following terms in order to improve the teaching effectiveness for all students. There are several possible solutions, such as adding HDL introduction and practice in the beginning or before the class, improving our FPGA cloud with an advanced debugging framework, and so on.

5 LIMITATIONS AND FUTURE WORK

Debugging and Simulation. As discussed in Section 4, debugging FPGA design in a remote manner currently exhibits inefficiency, especially for hardware-software co-debugging to find design defects of a custom CPU core. We plan to further boost our simulation framework which would be nearly the same as the hardware-software execution environment in our cloud FPGA node to allow students conduct local co-debugging in their VMs.

Project Schedule. We would like to rearrange the time slot of each project and coordinate with other courses to meet learning diversity of various students, making each of them possible to participate in all experimental projects.

Grading Method. Currently, the FPGA cloud platform can only return a result log file to each student after he or she completes a test case online. Therefore, the student could not see his or her score for this project instantly. Moreover, this also increases instructors' and teaching assistants' workloads due to the tedious and manual updates of project scores for each student. To tackle this problem, we would like to integrate an advanced auto-grading feature in our next-version FPGA cloud.

6 CONCLUSIONS

In this paper, we describe an in-house FPGA cloud platform for a COD course which can be effectively used to introduce the essential concept of hardware-software co-design for next-generation computer system design to early undergrads. Our proposed FPGA cloud provides an easy-to-use interface as well as a simple script-based design flow to cultivate students' *computer system thinking ability* with the minimum additional cost. We believe that our FPGA cloud and relevant ecosystem would be also appropriate to service students and instructors in many other computer system-related courses such as operating system, computer architecture, embedded system design, and so on.

7 ACKNOWLEDGMENTS

This work is supported by the National Key Technologies Research and Development Program of China under Grant No. 2017YFB1001602, by the National Natural Science Foundation of China under Grant No. 61702485, 61420106013, 61521092 and 61702480, and by the Youth Innovation Promotion Association of CAS under Grant No. 2017143 and 2013073. We appreciate Prof. Xiufeng Sui for his teaching in one of the two lecture classes of COD course at UCAS. We thank Zihao Yu, Bowen Huang, Sha Liu, Huizhe Wang, Xu Zhang, Lei Yu, Ran Zhao and Zhiwei Lai for their teaching and engineering assistance in our COD lab projects. We also thank anonymous reviewers' valuable feedbacks.

REFERENCES

[1] ACM. 2004. Computer Engineering Curricula 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering. https://www.acm.org/binaries/content/assets/education/curricula-recommendations/ce-final-report.pdf.

[2] ACM. 2013. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf.

[3] ACM. 2016. Computer Engineering Curricula 2016: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering. https://www.acm.org/binaries/content/assets/education/ce2016_web_final.pdf.

[4] Amazon Web Services. 2017. Amazon EC2 F1 Instances. https://aws.amazon.com/ec2/instance-types/f1/.

[5] Arizona State University. 2018. Progressive Learning Platform. https://plp.asu.edu/.

[6] D. Baldwin, P. Sanderson, R. McCartney, S. Ludi, N. T. Ramachandran, and C. Taylor. 2011. SIGCSE Special Project Showcase. In *Proc. 42nd SIGCSE Technical Symposium on Computer Science Education*. 5–6.

[7] Bochs. 2018. The Cross Platform IA-32 Emulator. https://bochs.sourceforge.net/.

[8] E. Brunvand. 2011. Games as Motivation in Computer Design Courses: I/O is the Key. In *Proc. 42nd SIGCSE Technical Symposium on Computer Science Education*. 33–38.

[9] P. Bulic, V. Gustin, D. Sonc, and A. Strancar. 2013. An FPGA-based Integrated Environment for Computer Architecture. *Computer Applications in Engineering Education* 21, 1 (March 2013), 26–35.

[10] Z. Chai, Z. Wang, W. Yang, S. Ding, and Y. Zhang. 2014. OpenHEC: A Framework for Application Programmers to Design FPGA-based Systems. In *Proc. 1st International Workshop on FPGAs for Software Programmers*. 59–64.

[11] M. L. Corliss and M. Melara. 2011. VIREOS: An Integrated, Bottom-up, Educational Operating Systems Project with FPGA Support. In *Proc. 42nd SIGCSE Technical Symposium on Computer Science Education*. 39–44.

[12] Github. 2018. Github Classroom: Your Course Assignments on GitHub. https://classroom.github.com/.

[13] M. Gschwind. 1994. Reprogrammable Hardware for Educational Purpose. In *Proc. 25th SIGCSE Technical Symposium on Computer Science Education*. 183–187.

[14] J. Hennessy and D. Patterson. 2018. A New Golden Age for Computer Architecture: Domain-Specific Hardware/Software Co-Design, Enanced Security, Open Instruction Sets, and Agile Chip Development. In *Turing Lecture of ACM/IEEE International Symposium on Computer Architecture (ISCA)*.

[15] M. J. Jipping, S. Henry, K. Ludewig, and L. Tableman. 2006. How To Integrate FPGAs Into a Computer Organization Course. In *Proc. 37th SIGCSE Technical Symposium on Computer Science Education*. 234–238.

[16] D. B. Larkins, W. M. Jones, and H. E. Rickard. 2013. Using FPGAs as a Reconfigurable Teaching Tool Throughout CS Systems Curriculum. In *Proc. 44th SIGCSE Technical Symposium on Computer Science Education*. 397–402.

[17] LC3Help.com. 2018. LC-3 Simulator. http://www.lc3help.com/.

[18] MIT Computation Structure Group. 2017. 6.175: Constructive Computer Architecture. https://csg.csail.mit.edu/6.175/.

[19] ICT of CAS. 2018. The Case for Labeled Von Noeumann Architecture (LvNA). https://sdc.ict.ac.cn/isca2018-tutorial/.

[20] OpenStack. 2018. OpenStack. http://www.openstack.com/.

[21] R. Ord. 2018. CSE 30: Computer Organization and System Programming. https://cseweb.ucsd.edu/~ricko/CSE30/.

[22] D. Patti, A. Spadaccini, M. Palesi, F. Fazzino, and V. Catania. 2012. Supporting Undergraduate Computer Architecture Students Using a Visual MIPS64 CPU Simulator. *IEEE Transactions on Education* 55, 3 (Aug. 2012), 406–411.

[23] Penn Engineering. 2016. Computer Organization (Course: ESE534). https://www.seas.upenn.edu/~ese534/.

[24] C. Quan, Y. Chen, S. Li, and Y. Zhao. 2016. Exploration of the Computer Hardware Experiment Teaching Method based on the Cloud Platform. In *Proc. IEEE Frontiers in Education Conference (FIE)*. 1–5.

[25] D. Roggow, P. Uhing, P. Jones, and J. Zambreno. 2015. A Project-based Embedded Systems Design Course Using a Reconfigurable SoC Platform. In *Proc. IEEE International Conference on Microelectronics Systems Education (MSE)*. 9–12.

[26] S. Sohoni, S. D. Craig, and S. Lu. 2017. Impact of Prior Exposure to the PLP Instruction Set Architecture in a Computer Architecture Course. In *Proc. 48th SIGCSE Technical Symposium on Computer Science Education*. 555–560.

[27] Stanford University. 2017. CS107E Computer Systems From the Ground Up. https://web.stanford.edu/class/cs107e/.

[28] D. Tarnoff. 2015. Integrating the ARM-based Raspberry Pi into an Architectural Course. *Journal of Computing Sciences in Colleges* 30, 5 (May 2015), 67–73.

[29] UC Berkeley EECS. 2018. CS61C: Machine Structures. https://inst.eecs.berkeley.edu/~cs617.

[30] UIUC ECE. 2018. ECE 411: Computer Organization and Design. https://courses.engr.illinois.edu/ece411/fa2018/.

[31] E. Upton, J. Duntemann, R. Roters, T. Mamtora, and B. Everard. 2016. *Learning Computer Architecture with Raspberry Pi* (1st ed.). Wiley Publishing.

[32] K. Vollmar and P. Sanderson. 2006. MARS: An Education-Oriented MIPS Assembly Language Simulator. In *Proc. 37th SIGCSE Technical Symposium on Computer Science Education*. 239–243.

[33] K. L. Wang, C. S. Cole, T. Wang, and J. Harris. 2013. An Effective Project-based Embedded System Design Teaching Method. In *the 120th American Society for Engineering Education (ASEE) Annual Conference and Exposition*. 6849:1–6849:9.

[34] J. M. Wing. 2006. Computational Thinking. *Communication of the ACM* 49, 3 (March 2006), 33–35.

[35] Xilinx. 2018. Vivado Design Suite — HLx Editions. https://www.xilinx.com/products/design-tools/vivado.html/.

[36] Xilinx. 2018. Zynq UltraScale+ MPSoC. https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html/.

[37] K. Zhang, M. Chen, and Y Bao. 2018. ZyForce: An FPGA-based Cloud Platform for Experimental Curriculum of Computer System in University of Chinese Academy of Sciences (Abstract Only). In *Proc. 49th SIGCSE Technical Symposium on Computer Science Education*. 1091.

[38] Y. Zhang, Y. Chen, X. Ma, Y. Tang, Y. Niu, S. Li, and W. Liu. 2017. Remote FPGA Lab Platform for Computer System Curriculum. In *Proc. ACM Turing 50th Celebration Conference - China (TUR-C)*. 3:1–3:6.