The Systematic Thinking Ability of Hardware/Software Co-design using FPGA

Ying Li[1] liying@buaa.edu.cn
Jiong Zhang[1] Corresponding Author
Hritik Mitra[1]
Shicheng Yu[1]
[1]School of Computer Science and Engineering, Beihang University, Beijing, China

Abstract—**This Research-to-Practice Work-In-Progress paper proposes a state-of-the-art method of "hardware-software co-design" (HSC) based on FPGA. The main contributions were: (1) Optimizing Curriculum System from hierarchical structure to vertical structure. The traditional computer courses are taught horizontally and independently which ignores the connection between software and hardware. Therefore, we adopt a coherent curriculum and teaching is vertically structured and logically sequenced to reconstruct the contents from loose coupling to tight coupling; (2) Optimizing teaching process from Software-Hardware to Hardware-Interface-Software. We establish a closed-loop teaching framework by designing some tightly coupled projects to integrate hardware, interface and software together; (3) Optimizing teaching method from complex to simple. Complex teaching tries to entirely develop a real system in one time but it is too difficult to implement. Based upon the theories of Occam's razor and Separation of Concern, simple teaching eliminates unnecessary knowledge and decouples the complex system into single and simple modules; (4) Optimizing teaching objectives from solving basic academic problems to solving complex engineering problems. To train engineering talents, we use industrial methods to solve industrial problems which meet industry standards. Finally, evaluation based on a capability-maturity model like CDIO-CMM (CDIO Capability-Maturity Model) was done by means survey questionnaire and the results indicate hardware-software co-design can effectively improve students' ability of system design and the proportion of students at advanced level is increased from 13% to 37%.**

Keywords—Hardware/Software Co-design, Vertical Teaching, Lightweight Teaching, Complex Engineering Problems

I. INTRODUCTION

With the emergence of large-scale data and the popularization of personal mobile devices, we entered the post PC era which involves more complex problems and more extensive application. Three main features of PC era are systematicness, comprehensiveness and integratedness. Therefore, computer teaching should place more emphasis on system design rather than program design.

Curriculum Guidelines for Undergraduate Programs in Computer Science 2013 (CS2013, ACM&IEEE) said computer science graduates should have an ability to design a system, component, or process to meet desired needs within realistic constraints. And the curriculum must prepare graduates to analyze, design, verify, validate, implement, apply, and maintain computer system. CS2013 attaches great importance to cultivating students' system design and engineering practice ability.[1] We thought system design ability is a kind of engineering practice ability to design architecture, write application programs and define software/hardware interface according to the specific functions in system. And systems design skills can be improved from three aspects, system knowledge, classroom experiment and engineering practice. System knowledge is the basis and it refers to the working principle, design process, software and hardware cooperation, and abstraction layers of computer system; classroom experiment is the means and it helps students' understanding of theory and modifying of the existing tacit knowledge; engineering practice is the carrier and it aims to develop systems engineering talents who can use engineering technologies to solving complex engineering problems.

Through the investigation and research analysis, at present, the main problems of system design courses (SDC) are: ① SDC are set independently, which leads to redundancy and disconnection of knowledge; ② SDC pay more attention to software than hardware, which leads to failing to understand the computer system from a system perspective; ③ SDC includes a great deal of validation experiments, which leads to a lack of large-scale system design training.

A lot of researches on software-hardware co-design (SHC) have been done and paper [2][3] thought SHC is a very important computer technology at both academic and industry. Paper [4] introduced a teaching method of "C to hardware" and paper [5] described a lab session-based course to develop hands-on SHC skills. But they focused more on software development and rarely introduced hardware design. Paper [6] designed a RISC processor and a high-level language together and paper [7] presented the techniques for design of embedded systems consisting of analog, hardware and software components. But most of their experiments are theoretical analysis and numerical simulation which are separate from the actual.

The rest of the paper is structured as follows: Section 2 introduced some optimized methods of software/hardware codesign in three stages: Basic stage is to optimize curriculum system from hierarchical structure to vertical structure; Professional stage is to optimize teaching method from complex to simple; Practical stage is to optimize teaching objectives from solving basic academic problems to solving complex engineering problems. Section 3 analyzed some statistical data from a course which used the software/hardware codesign to indicate the advantages in terms of the proportion of students who complete the course experiments. Finally, section 4 showed some conclusions and outlines future work.

II. OPTIMIZATIONS OF SOFTWARE/HARDWARE CODESIGN

A. Optimizing Curriculum System from horizontal structure to vertical structure

Traditional computer teaching (TCT) arranges its contents in a hierarchical way and the knowledge is decomposed into different courses, shown in Figure 1. Each course focuses primarily on one subsystem of the complex computer system. TCT is regarded as a kind of modular teaching (MT) and the contents between courses are mostly designed independently and they are loose coupling from each other. Teaching knowledge in a modular way can reduce learning difficulty and it accords with the cognitive law of students, more accurately, students are better at local cognitive than global cognitive. But it ignores the relevance between contents of each course, especially the close relationship between software courses and hardware courses, and it leads to students cannot see the wood for the trees. E.g., software development courses just teach students how to write programs and but they do not know what happened to the underlying hardware.

Therefore, we proposes a systematic teaching (ST) to teach knowledge in a vertical way and the contents is reconstructed from loose coupling to tight coupling. We use some hardware/software co-design projects to run through each abstraction layer of computer system, shown in Figure 1. Vertical integration in teaching and learning can develop students' systems thinking and teach them to solve problems with a systems perspective. ST pays more attention to the vertical relationship between different abstraction layers and aims to train students to have both complete computer system knowledge and practical ability to solve complex engineering problems.

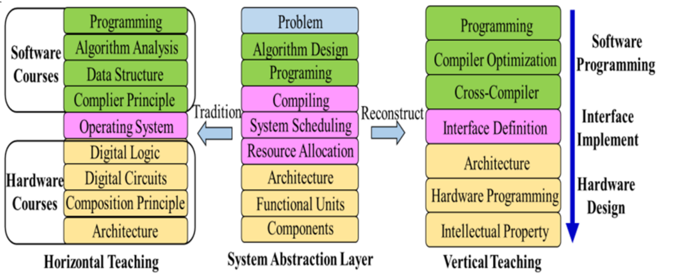

Figure 1 From Horizontal Teaching to Vertical Teaching
(Note: Green, pink and orange parts refer to software, interface and hardware, respectively)

How to develop some special hardware/software co-design projects to connect the software knowledge and hardware knowledge in a vertical way. Through research and practice, we find FPGA effectively becomes the system interconnection and construct a closed-loop teaching framework, which covers all the knowledge and the whole process of system development, from software development, to communication interface implement, to hardware structure design. A complete computer system involves in numerous unorganized knowledge of software and hardware, and in order to reduce the difficulty of teaching and learning, we propose some innovative methods, they are:

1) Separation of Concerns

We use the idea of "Separation of Concerns" (SOC) proposed by CS2013 to isolate the key problems in the overall system. [8] SOC is a powerful rule to remove one software or hardware component after another until the problem is separate to a small and independent task of the whole system. And each task, as a stage of system development life cycle, completes specific functions to support the task before it and offers services to the task after it. We take a well-defined Travel Sale Problem (TSP) as the hardware/software co-design project and the students only need learn how to solve TSP problem from the beginning to the end. The main feature of TSP is its hardware/software co-design and it can not only be solved by software but also is very suitable for the hardware implementation based on FPGA. The solution of TSP is divided into six parts and from upper software layer to lower hardware layer, they are algorithm design, programming, system scheduling, resource allocation, computer architecture and functional units.

2) Comparative Teaching Methodology

In order to make students understand the technology of software-hardware co-design more easily, we show multiple solution methods of TSP to students and analyze their differences by adopting a comparative method to know how well they perform. The comparisons include: ① Two algorithms: by running Branch-and-Bound (BB) and Dynamic Programming (DP) on the same single-core platform on FPGA (one core of Virtex-5 FPGAs), students can gradually understand the influence of solving quality based on different optimization algorithms and learn how to translate the real problem into its best algorithm, and into a computer program. ; ② Two programming patterns: each of the two algorithms, BB and DP, is designed in two types, serial pattern (SP) and parallel pattern (PP) to design. Compared with SP, students are unfamiliar with PP and PP involves a lot of hardware knowledge (e.g., task allocation, load balancing and dynamic scheduling) which is hard to understand. Therefore, we establish a closed-loop framework to realize the most efficient cooperation in both worlds—software and hardware. The framework uses hardware to guide software and develops hardware from software. In this way, BB and DP are transformed from serial pattern into parallel pattern which is implemented on a multicore FPGAs platform (16 cores of Virtex-5 FPGAs), respectively. By comparison both the performance and the implementation of SP and PP, students can learn more their advantages and disadvantages and understand how to write parallel programs to accelerate application.

3) Theory in Practice

As we know, computer system has strong characteristics of comprehensiveness and practice, which brings great difficulty to teaching. And practice is an indispensable part and it can help students understand theoretical knowledge better by learning by doing. [5] But the system development involves many operations, from software programming to interface communication, to hardware design and in order to fix the weakness of one operation-to-one experiment platform, we have developed a general online hardware/software co-design environment based on FPGA. It can effectively reduce the difficulty of hardware teaching and learning from following aspects. ① The code in the platform is open-source. It is both easy for teachers to extract code fragments and for students to build their own experimental environment and they are not affected by time and space; ② The code in the platform is widely used in academia and industry, which makes students to understand some standard and industrial coding style; ③ The code in the platform can be used to design a complete system, which makes students to solve complex engineering problems through interface calling.

B. Optimizing teaching method from Complex to Simple

Complex style tries to entirely develop a real system in one time but it is too difficult to implement. Based upon Occam's Razor theory, simple style eliminates unnecessary knowledge and decouples the complex system into single and simple modules. [9] One module corresponds to a single-problem and has clear boundary between hardware and software and modules can be integrated automatically through standardized interface. Progressive teaching method, in particular to “hardware/software co-design, software-to-hardware and hardware-accelerate algorithm” is adopted to design a small-but-complete system through lightweight hardware-structure, lightweight cross-compiler and lightweight software development. It can effectively facilitate the internalization both from knowledge to ability and from theory to practice.

1) Lightweight Hardware

Hardware experiments are difficult to put into practice, one reason is hardware programming language has a lower level of abstraction and another reason is hardware implementation is very difficult. After extensive survey and study and analysis, most of hardware courses generally choose some commercial processors (e.g., GPU) as the experiment platform for its high performance and full function. And they thought students can understand the actual engineering design. However, in practice, it turns out very difficult to teach this kind of sophisticated processor for undergraduate students, who have to spend a lot of time and energy to learn the complex hardware descriptive language which often play the role as noise and it leads to undesirable phenomenon like "hardware imagined hardly, software coded softly".

Accounting to the problems mentioned above, we improve teaching methods and contents and try to help students to clarify the major points from the noisy contradiction and, let them pay most attention on understanding the "hardware/software co-design" method as the first priority. We use the FPGA-based processor (Virtex-5 FPGA, Beehive) instead of GPU, which has a lightweight multi-core architecture and small monitor system to run programs. The lightweight architecture advantage of FPGA can simply the design process and make learning hardware easier. Then, students can focus on the core knowledge of “data consistency, shared memory, load balancing”, which can be taught with very simple programs.

2) Lightweight Experiments

The hardware experiment is very comprehensive. Even with the help of teachers, it is difficult for students to achieve the teaching goal. Therefore, in experimenting, the complex engineering problems are artificially divided and simplified into several sub problems of software-independent-hardware by applying modular programming idea. The method of decoupling software and hardware is used to define the boundary or interface between software and hardware which follows the Single Responsibility Principle. By this way, the process of system design is simplified and formulized, because it's daunting to try and tackle several problems at once and the difficulty of experiment is reduced. Specifically, we divide the "multi-core based TSP" into several small problems which are both independent and interrelated, and each of them can be solved directly. Through the process of problem decomposition, sub-problem solution, and system synthesis, students learn the "hardware/software co-design" method, and understand the sequential dependencies between hardware and software.

3) Lightweight Develop Process

Cross compilation is a common method in embedded system development. The high complexity of hardware architecture causes serious problems with compiling efficiency and implementation difficulty. Choosing a lightweight FPGA-based multi-core processor can simplify the development process and reduce the interference caused by the complete cross-compiler toolchain and the tedious procedure of hardware-software debugging. The advantage of FPGA-based processor lies in its simple cross compilation environment and further, it uses a reduced monitor system instead of a complex system, and it makes students give only 4 commands as well as 6 parameters to quickly migrate functionality from software to hardware, as "C source code - > assembly file - > target file - > executable file - > bootable file". In order to further simplify the compilation process, we provide a small but easy way to use toolchain (encapsulated configuration files) to meet the learning needs of the students with different ability levels.

C. Optimizing teaching objectives from solving basic academic problems to solving complex engineering problems

The goal of system design is to cultivate the students' ability to solve complex engineering problems, master the method of "hardware/software co-design", and establish a knowledge set based on the actual system. Therefore, students must do experiments to understand the relationship between hardware and software and improve their system development ability and engineering skills. However, system design is generally to solve large-scale problems with high complexity and many uncertain factors, which brings great obstacles for the students to complete the experiments. In this paper, we try to make it easier for students with novel teaching approaches, as follows:

1) Establishing progressive experimental content model as "Verification-Design-Research"

In order to reduce the difficulty of learning, we divide the experiments into three types: verification, designing and research. The level of these three types of experiments progresses in difficulty levels, and the requirements of students is expected to gradually increase.

- Verification type: Mainly focuses to test the theory, deepen the understanding of classroom knowledge, and cultivate the basic professional skills and practical operational ability of students. This kind of experiment adopts the "traditional teaching mode". So the relationship between teachers and students is one-way teaching, which reflects the directive position of teachers in teaching.
- Design type: Mainly student oriented designing and teacher tutoring as supplementary. Only experimental subjects and experimental purposes are given. During the process, students should independently complete the experiments. Design experiment is important to cultivate students' ability of comprehensive analysis and independent problem-solving. Using the "Discussion based mode" to teach the experiment contents and it is a two-way discussion relationship between teachers and students. Teachers inspire the students to extend their thinking and participation, through positive interaction and equal communication.
- Research Type: it is one of the main novelties of this paper. It effectively enables students to internalize the explicit knowledge they have learned into the tacit knowledge, and it guides the students how to design system with the method of "hardware/software co-design".

2) Establishing the progressive goal of the experiment as "simple engineering - well-defined engineering - broadly-defined engineering - complex-defined engineering"

According to the Washington protocol and cs2013, we believe that system design capabilities mainly include well-defined problems, broad-defined problems and complex-defined problems, which are respectively reflected in the early, middle and later stages of system development, and corresponding to the different types experiments (verification, design and research described in * * section). The specific meaning is as follows:

(1) Ability to solve well-defined engineering problems: students understand the realization principle of some specific problems under the guidance of teachers. Verification experiment aims to test students' understanding of the essential knowledge by providing the source code. At this stage, students should submit a code analysis report to explain how the source code realizes the working principle of the core knowledge points.

(2) Ability to solve broadly-defined engineering problems: students use their knowledge to solve some common problems and they should independently carry out the tasks of scheme design, code realization, debugging equipment and result verification. Design experiment only gives a brief description of experiments. At this stage, not only the students' understanding of knowledge is further deepened, but also the students' problem-solving skills are improved.

(3) Ability to solve complex engineering problems: students solve the complex engineering problems through teamwork. In order to cultivate students' practice ability, we adopt CDIO engineering teaching method. During the whole process, CDIO includes four stages: Conceive, Design, Implementation and Operation. [10]

- Conceive: teachers need to design some complex engineering problems, which covers all stages of the system development life cycle and involves the main hardware and software knowledge points. Complex engineering problems have the following characteristics: (1) they can only be solved by using in-depth engineering principles and theories; (2) the requirements of the problems involve many aspects related to engineering technology and theory knowledge, and may also conflict with each other; (3) problems can only be solved by establishing appropriate abstract models, and the solution embodies innovation; (4) they cannot be solved completely by common methods alone; (5) the factors involved in the problems may not be fully included in the curriculum; (6) the objectives of all parties in the problem are not completely consistent; (7) the problem possesses the characters of comprehensive, systematic, technical and economic.
- Design: it includes two parts: task assignment and interface design. Task assignment is implemented based on Separation of Concerns (SOC) theory. SOC divides the complex engineering problems into several sub-parts and one student undertakes one task. Interface design is used to exchange info between modules, and finally each well-designed sub-system is integrated into a complete system. It effectively reduces the correlation between sub-systems, and realizes the loose coupling mode of system development. The design stage embodies the strong characteristics of engineering thinking.
- Implementation: students are placed at the center and they do the experiments through teamwork under teachers' guidance. Teachers serve many great roles and they mentor students, track the progress of experiments, listen and look for signs of trouble, provide some valuable instructions during the key phases, correct students' mistake in time. This is the stage of embodying explicit knowledge into tacit knowledge and
- Operation: the results are displayed and tested. It mainly includes the evaluation of experiment completion (EC), assignment quality (AQ) and student performance (SP). Specifically, EC is examined in the form of presentation and live demo, and it reflects students' ability of system development; AQ, including code source, reports, is judged by teachers mainly based on their professional experience and it reflects students' understanding of software/hardware co-design; SP is graded by other students in the group and it reflects students' emotion in the learning process.

III. Experiment

CDIO-CMM (CDIO Capability-Maturity Model) evaluation system was used to evaluate the maturity of students' system development ability, which was proposed by Carnegie Mellon University. [11] According to the actual teaching situation, the experiment model divides the maturity of "hardware/software co-design" ability into three scales as primary, intermediate and advanced (see Figure 2). Among them, the primary level refers to students mastering basic software and hardware knowledge; the intermediate level refers to students being able to use the knowledge they have learned to solve medium-sized problems independently; the advanced level refers to students having certain scientific research ability and being able to solve complex engineering problems through team cooperation.

In the experiment, the questionnaire was distributed online (see Table 1 with some real subjects) to 100 students who were randomly selected. The questionnaire was completed (100 copies were collected) by the same participants at the beginning, middle and end of the course. SPSS software was used to analyze the reliability of the questionnaire (the reliability value of the two questionnaires was set to 0.85). Then, CDIO-CMM data analysis was carried out for the questionnaire whose reliability reached the standard. The results are shown in the Figure 2.

A. Analyze the maturity of core competencies that affect system development

CMM respectively evaluates the maturity of the four kinds of core competencies that affect the system development: basic knowledge, software and hardware collaboration, scientific research and system capabilities. The statistical results show that in the middle of the course, the students at the primary level are the largest, about 69%. With the development of the course, the proportion of students at the intermediate level or even at the advanced level is gradually expanding. At the end of the course, the proportion of students who answered the questionnaire, at the advanced level is increased from 13% to 37%.

B. Analyze the correlation scales between the core competencies that affect system development

Table 2 shows the statistics of the correlation scales between the core competencies (called index variables) affecting the system development. According to the statistical analysis theory, there is an obvious phenomena relationship between the four core competencies, that is, with the advancement of the curriculum, the maturity of students' system development ability is gradually improving, and there is a high linear correlation between the "hardware/software co-design" ability and the system design ability, indicating that there is a direct correlation between them.

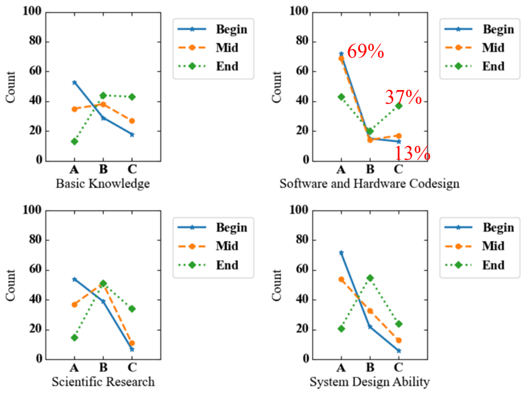

Figure 2 System Design Abilities based on CDIO-CMM
(Note: A, B, C represents the proportion of students who completed the primary, intermediate and advanced experiments, respectively; Begin, Mid, End represents different stages of the course)

IV. Conclusion

After more than 8 years of research, we found hardware-software co-design based on FPGA is an effective teaching method to improve students' systematic thinking skills. It includes three stages: Basic stage is to understand the fundamentals of computer system and establish a preliminary system concept by optimizing curriculum system from hierarchical structure to vertical structure; Professional stage is to form a system view of software-hardware co-design by optimizing teaching method from complex to simple; Practical stage is to develop system design capability by optimizing teaching objectives from solving basic academic problems to solving complex engineering problems. We will plan to design an integrated virtual simulation platform for hardware-software co-design.

Table 1 Some Questionnaire Questions and Results

No.	Closed Questions	Before	After
1	Have you read about thousands of lines of FPGA code?	35%	82%
2	Have you did some Hardware and Software Codesign projects?	13%	78%
3	Can you design a simple system?	8%	57%
4	Can you solve complex engineering problems?	24%	65%

(Note: Before indicates before course; After indicates after course)

Table 2 Correlation Scales between System Design Abilities

	BK	**SHC**	**SR**	**SD**
BK	1.00	0.57	0.28	0.41
SHC	0.57	1.00	0.36	0.91
SR	0.28	0.36	1.00	0.39
SD	0.41	0.91	0.39	1.00

(Note: Basic Knowledge-BK, Software/hardware Codesign-SHC, Scientific Research-SR, System Design-SD)

References

[1] Uhsadel, Leif, et al. "Teaching HW/SW co-design with a public key cryptography application." IEEE Transactions on education (2013): 478-483.
[2] Wolf, Wayne H. "Hardware-software co-design of embedded systems." Proceedings of the IEEE (1994): 967-989.
[3] Zhang, Ke, et al. "Computer Organization and Design Course with FPGA Cloud." Proceedings of the 50th ACM Technical Symposium on Computer Science Education. 2019.

[4] Weinhardt, Markus. "Teaching hardware/software codesign on a reconfigurable computing platform." International Symposium on Applied Reconfigurable Computing. Springer, Berlin, Heidelberg, 2012.
[5] Balasch, Josep, et al. "Teaching HW/SW codesign with a Zynq ARM/FPGA SoC." 2018 12th European Workshop on Microelectronics Education (EWME). IEEE, 2018.
[6] Joseph, Diya, Geetika Kaur, and Pinaki Chakraborty. "An exercise on hardware/software codesign following the RISC model." Computer Applications in Engineering Education (2016): 305-312.
[7] Ha, Soonhoi, et al. "Introduction to hardware/software codesign." Handbook of Hardware/Software Codesign. Springer Netherlands, 2017. 3-26.
[8] Ooms J. The OpenCPU system: Towards a universal interface for scientific computing through separation of concerns[J]. arXiv preprint arXiv:1406.4806, 2014.
[9] Lewis T, Perkowski M, Jozwiak L. Learning in hardware: architecture and implementation of an FPGA-based rough set machine[C] Proceedings 25th EUROMICRO Conference. Informatics: Theory and Practice for the New Millennium. IEEE, 1999, 1: 326-334.
[10] Bankel J, Berggren K F, Engström M, et al. Benchmarking engineering curricula with the CDIO syllabus[J]. International journal of engineering education, 2005, 21(1): 121-133.
[11] Solar M, Sabattin J, Parada V. A maturity model for assessing the use of ICT in school education[J]. Journal of Educational Technology & Society, 2013, 16(1): 206-218.