Proceedings of the 2006 Workshop on Embedded Systems Education

Editor

Dr. Jeff Jackson

The University of Alabama

October 26, 2006

Seoul, South Korea

ARTIST2

Proceedings

2006 Workshop on Embedded Systems Education

WESE2006

Editor

Jeff Jackson

Seoul, South Korea

October 26, 2006

Organizer's Message

I am pleased to present this second workshop on embedded system education associated with the 2006 EMSOFT embedded software conference.

The workshop consists of multiple sessions focusing on embedded systems education programs and consortia; embedded systems courses and curricula issues; embedded systems hardware and methodologies; and embedded systems curricula, programs and projects. I expect the sessions to give matter to lively, animated and enriching discussions as they did at the first workshop held last year.

The organizing and program committees and additional reviewers selected sixteen papers for inclusion in this workshop from international academic and industrial authors. I would like to thank the organizing committee and the program committee members for their time and efforts in inviting papers, organizing reviewers, and for providing general assistance to the workshop organization. Without their efforts the workshop would not have been possible. I would like to thank the authors for their manuscript submissions and their timely response to implement improvements suggested by the reviewers. Finally, I would like to thank IEEE, ACM and ARTIST2 organizations for their support.

As last year, I feel this first workshop is a great success. However, there exist many opportunities for international cooperation in the community of embedded systems researchers and educators and I look forward to those efforts.

Jeff Jackson

Organizing Committee Members

Jeff Jackson, The University of Alabama, USA
Paul Caspi, Verimag-CNRS, France
Jogesh Muppala, The Hong Kong University of Science and Technology, Hong Kong
Wayne Wolf, Princeton University, USA
John K. Zao, National Chiao Tung University, Taiwan

Program Committee Members

Tom Conte, North Carolina State University, USA
Mats Daniels, Uppsala University, Sweden
Jen Davoren, The University of Melbourne, Australia
Kenji Hisazumi, Kyushu University, Japan
Jin Hyung Kim, KAIST, South Korea
Yann-Hang Lee, Arizona State University, USA
Kenneth G. Ricks, The University of Alabama, USA
Chi-Sheng (Daniel) Shih, National Taiwan University

2006 Workshop on Embedded Systems Education

October 26th, 2006, Seoul, South Korea

Program

8:30 Opening

8:45 Education Programs and Embedded Systems Consortia

- **Stylianos Mamagkakis**: *Research Network for System Level Design of Embedded Systems: Dynamic Memory Allocation Design Flow Case Study*
- **Kenji Hisazumi**: *QUBE: A Practical Education Program for System LSI Designers*
- **Tai-Yi Huang**: *The Embedded Software Consortium of Taiwan - A Progress Report of Educational Activities*

10:00 Coffee Break

10:20 Embedded Systems Courses and Curricula Issues

- **Tulika Mitra**: *Challenges in Designing Embedded Systems Courses*
- **Kenneth Ricks**: *Addressing Embedded Programming Needs within an ECE Curriculum*
- **Jogesh K. Muppala**: *Bringing Embedded Software Closer to Computer Science Students*
- **Shiao-Li Tsao**: *The Development and Deployment of Embedded Software Curricula in Taiwan*

12:00 Lunch

13:20 Embedded Systems Hardware and Methodologies

- **Shekhar Sharad**: *Methodologies to Bring Embedded Systems to Non-EE Students*
- **Shanq-Jang Ruan**: *Development and Analysis of Power Behavior for Embedded System Laboratory*
- **Chi-Sheng Shih**: *Toward HW/SW Integration: Networked Embedded System Design*

- **Falk Salewski**: *Hardware Platform Design Decisions in Embedded Systems - A Systematic Teaching Approach*

15:00 Coffee Break

15:20 Embedded Systems Curricula, Programs and Projects

- **M. Balakrishnan**: *Experiences of a Summer Workshop in Embedded Systems*
- **Hans-Gerhard Gross**: *The Delft MS Curriculum on Embedded Systems*
- **Niklas Adamsson**: *Experiences from large embedded systems development projects in education, involving industry and research*
- **Lindsay T. Kane**: *The Windows Embedded Academic Program – Retrospective & Directions, 2002-2006*
- **Masaki Yamamoto**: *An Extension Course for Training Trainers of Embedded Software*

17:30 Conclusions and Future Considerations

Table of Contents

Research Network for System Level Design of Embedded Systems: Dynamic Memory Allocation Design Flow Case Study

Stylianos Mamagkakis[1, 3], David Atienza[2], Francky Catthoor[3], Dimitrios Soudris[1] and Jose M. Mendias[2]

[1] *VLSI Design Center- Democritus Univ. of Thrace 67100 Xanthi, Greece {smamagka,dsoudris} @ee.duth.gr*

[2] *DACYA, Univ. Complutense de Madrid 28040 Madrid, Spain {datienza,mendias} @dacya.ucm.es*

[3] *Interuniversity Micro-Electronics Centre (IMEC vzw) 3001 Heverlee, Belgium {mamagka,catthoor} @imec.be*

Abstract

The challenges encountered in system level design require a global approach and considerable manpower, which are not likely to be found in a single organization. Cooperation and sophisticated communication between partners from both academia and industry are required. This paper describes the formal relationship and collaboration between IMEC and several Universities for research and training in the domain of system level design in embedded systems. IMEC (Leuven, Belgium) provides a meeting point between multiple academic institutions and a gateway to relevant information and drivers from the industry, unifying the common research goals of several academic groups working in this area in a consistent flow, sensitive to the market needs. In this paper we focus in the general concept of the cooperation network, which can be illustrated shortly by one part of it, namely the dynamic memory allocation issues of the design flow and the related universities in this effort: DUTH (Xanthi, Greece) and DACYA UCM (Madrid, Spain).

1. Introduction

The collaboration illustrated in this paper addresses the needs for a unified, consistent global approach to the various problems encountered in embedded system design. These problems are so closely interconnected that it is common to worsen one aspect of the embedded system design, while trying to improve another [1]. In fact, a small research group with limited resources will ignore the global view in order to maximize its potential within its limited research focus. In order to realize a global approach, without sacrificing quality, a network of research groups is needed. It should be guided by a coordinator that has sufficient critical mass and has access to the necessary information to work on practically relevant end goals.

IMEC with its unique balance between academic and industrial R&D can lead this effort of a star-shaped training and research network. Its geographical location in Belgium offers a unique meeting point for researchers of universities all across Europe, where they can be trained and establish a collaboration towards interconnected goals in embedded system design. The context of the collaboration focuses mostly on the mapping of concurrent, dynamic applications, which are increasingly important in multimedia and network applications. Several design steps are required to cost-effectively map the data and tasks in such applications to a multi-processor platform.

2. Dynamic Memory Allocation Design Flow Case Study

Modern embedded applications have become increasingly dynamic and consist of multiple tasks, which run in multi-processor embedded systems. Design optimizations at the system level are required to optimize the energy consumption, memory footprint, memory accesses and execution time of the final design [1]. A complete optimization design framework is proposed by IMEC in collaboration with its university partners at the system level (see Figure 1), which can be followed in a step-by-step procedure. Each stage consists of systematic design methodologies (or steps), which are closely interconnected and the outputs of the higher ones pass their constraints to the lower ones (more details about the optimization steps and the way they are derived can be found in [2]).

These systematic methodologies require many person-years of research to be developed and perfected and different research networks are assembled in order to take advantage of the expertise of the individual researchers that comprise them. In fact, the proposed system level methodology comprises the combined effort of 14 PhD students and 7 professors from IMEC, the Democritus

University of Thrace (DUTH located in Xanthi, Greece) and the Department Arquitectura de Computadores y Automatica (DACYA UCM located in Madrid, Spain), as shown in Figure 2. Earlier on, also ESAT (K.U. Leuven, Belgium) and recently also U. Bologna have become evolved (see Sect. 4).

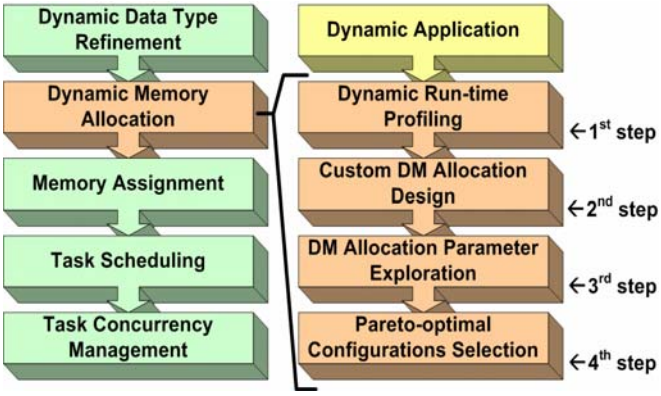


Fig.1. System level design optimization steps and dynamic memory allocation sub-steps

As shown in Fig. 1, the transformation and refinement of the original dynamic data types in the specification are addressed. The transformation and refinement of the original dynamic data types in the specification are addressed. IMEC focuses its efforts both on dynamic multimedia applications, e.g. scalable video coding, dynamic image processing, video games, graphical user interface with animation of rendered multimedia objects, as on wireless network applications. In both cases, the target is to embed them on a cost/power-sensitive platform. In these applications, the conventional compile-time techniques only allow to come up with static solutions that assume worst-case and hence much too pessimistic requirements on memory footprint and performance. Hence, the data type transformation methodology was further developed and extended to generate a low cost implementation in terms of performance, memory footprint and energy consumption, which is especially suited for portable appliances, involving dynamic data types. Trade-offs up to several orders of magnitude have been obtained for realistic applications. Also an effective prototype tool has been completed to automatically profile the evolving dynamic memory behavior of the application and adding a systematic exploration technique was started. This research has happened in cooperation with the Univ. Compl. of Madrid and Democr. Univ. Thrace. It builds further on the past experience in the Matisse research project for wired telecom protocols.

The refined dynamic data types still have to be (de)allocated in the virtual memory space and this typically happens with a dynamic memory manager. The dynamic memory manager is analyzed separately on the right side of Fig. 1. In this case, standard library based solutions lead to much access and footprint overhead compared to the minimal achievable costs. Hence, we have further extended our techniques to fully automatically explore cost-efficient custom dynamic-memory manager solutions exploring still the complete search space in a systematic way, including now also the total energy consumption as an overall objective on top of the ones for memory accesses and footprint. All this is based on a detailed analysis of the (de)allocated patterns of the application. Compared with conventional (dynamic) memory management techniques for embedded systems, IMEC has obtained significant gains, both for multi-media applications and for wireless networks. A tool has been developed that supports the complete trajectory, starting from C++ source code.

The performance and energy consumption of the shared memory hierarchy on a platform depends largely on how the data access of a dynamic application is ordered and assigned to the physically available memories. Therefore, IMEC has developed new techniques to solve the data-to-memory assignment problem for multi-tasked applications where data of dynamically created/deleted tasks is allocated at runtime. In contrast to the previous years where focus was mainly on memory access scheduling and assignment, work now started on a complete script for task-level data transfer and storage exploration. For this purpose a representative application and target platform were selected, and the different steps were evaluated that are required to bridge the gap between the high-level object-oriented specification of data in dynamic concurrent types, together with their realization on a distributed hierarchical memory organization. We are also investigating the use of a hardware emulation platform to evaluate the mappings. This research was executed in cooperation with the Univ. Complutense Madrid, Univ. of Bologna, and Univ. Barcelona.

To show the productivity of such combined approach, just in the dynamic memory allocation step, the cooperative work of 3 to 4 PhD candidates from each research group working full-time during the last 3 years for the development of the dynamic memory allocation methodologies have accomplished the development of 4 national and European funded research projects and the publication of 4 internal technical reports, 21 conference papers, 4 journal papers and 1 book chapter.

3. IMEC – Universities Network

The network between the research groups in IMEC and in the universities was star-shaped up to now. This means that the formal relationship is between any given university and IMEC and not yet between the universities themselves. The typical research work plan involves regular visits of at least one researcher of each group to IMEC for a period of 3 to 6 months per year. Researchers working in the

same optimization step have to overlap for at least 3 months each year to synchronize with each other properly.

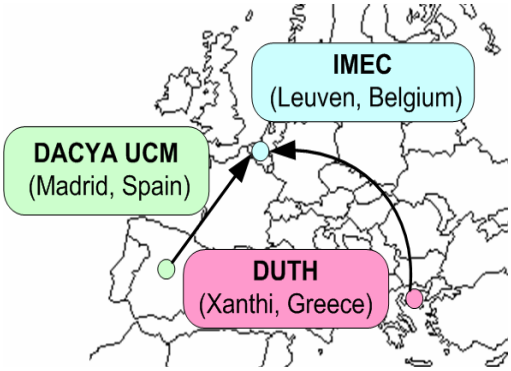


Fig. 2. IMEC-Universities network for embedded system design

In the case of the research and training required for the optimization step of dynamic memory allocation, one PhD candidate from DUTH and one from DACYA UCM have spent 20 and 21 months respectively in IMEC in the last 3 years. The economic support for the travel and cost of living abroad at this stage is largely provided by EU-funded Marie Curie Fellowships [4].

The EU-funded Marie Curie Fellowships are not tied to other specific projects (e.g., IST or national projects) but are a project on itself. The money that the PhD candidates receive are on addition to their existing funds and are provided in order to pay for their travel costs, their accommodation and living expenses in the country where their host institution is (i.e., in this case IMEC, Leuven).

During their stay in IMEC, the researchers (mainly PhD students) follow general training courses (e.g. "Speaking in public" and "Writing technical reports") and more specialized courses on the unified meta-design flow concepts [1], which are a basis for the global approach of system design. Additionally, they follow specialized courses on the latest system design flows, design toolkits and improvement methodologies.

Finally, they enjoy a high quality research environment (both in facilities and research personnel) and are given the attractive opportunity to interact with both the industrial and the academic international world. Also, this includes seminars once every week from prominent industry and academic figures in the IMEC auditorium.

During each visit, the researchers from the universities collaborate also with the corresponding researchers in IMEC (which work on dynamic memory allocation in this particular case study) and define the interactions between the steps with researchers working in other optimization steps. Most importantly, in each visit in IMEC they kick start the research of one systematic methodology (step) of the dynamic memory allocation and then they continue their research on this step, when they return in their research group back in their university. For the rest of the year, they retain their strong cooperation by e-mail, web conferencing and phone-conference calls.

The PhD candidates also coordinate the effort within their research groups devoted to the same research goal. The research done locally in each university is supported by funds from the corresponding laboratory acquired through associated research projects or research grants. These research projects are not directly related to the Marie-Curie fellowships but are related to the type of research that is done during the fellowships in the host institution (e.g. see AMDREL project [5]).

4. Conclusions and Future Prospects

The research network addressed in this paper, has been very successful in the field of dynamic memory allocation and system level design for embedded systems and already 2 students (i.e. M. Leeman and D. Atienza) have received their PhDs, while working in the network environment. This collaboration is and will be continued and extended in the future to other research topics. Finally, a polyhedron-shaped network between the current groups and additional excellent institutions in Europe, like DEIS University of Bologna and CS 12 of Dortmund, is under construction to ensure the best interaction and mobility between the partners. Further actions under the ARTIST2 [6] and HIPEAC [7] umbrella are also envisioned.

Acknowledgements

This work is partially supported by the E.C. funded program AMDREL IST-2001-34379, the Spanish Government Research Grant TIN2005-05619 and E.C. Marie Curie Fellowship contract HPMT-CT-2000-00031.

REFERENCES

[1] F. Catthoor et al., *"Unified meta-flow summary for low-power data-dominated applications"*, Kluwer Publications, 2000.

[2] *IMEC Design Technology Program*, http://www.imec.be/ovinter/static_research/designtechnology.shtml

[3] S. Mamagkakis et al., *"Energy-efficient dynamic memory allocators at the middleware level of embedded systems"*, in Proc. of EMSOFT, S. Korea, 2006

[4] *Marie Curie Actions - Human Resources and Mobility Activity*, http://europa.eu.int/comm/research/fp6/mariecurie-actions/action/fellow_en.html

[5] AMDREL project "Architectures and Methodologies for Dynamic REconfigurable Logic", http://vlsi.ee.duth.gr/amdrel/

[6] ARTIST 2, Network of Excellence on embedded system design, http://www.artist-embedded.org/artist/

[7] European Network of Excellence on High-Performance Embedded Architecture and Compilation, http://www.hipeac.net/

QUBE: A Practical Education Program for System LSI Designers

Akira Tsukizoe
System LSI Research Center,
Kyushu University
305-3-8-33 Momochihama,
Sawara-ku, Fukuoka 814-0001 JAPAN
+81-92-847-5190
tsukizoe@slrc.kyushu-u.ac.jp

Kenji Hisazumi
System LSI Research Center,
Kyushu University
305-3-8-33 Momochihama,
Sawara-ku, Fukuoka 814-0001 JAPAN
+81-92-847-5190
nel@slrc.kyushu-u.ac.jp

Takanori Hayashida
System LSI Research Center,
Kyushu University
305-3-8-33 Momochihama,
Sawara-ku, Fukuoka 814-0001 JAPAN
+81-92-847-5190
hayasida@slrc.kyushu-u.ac.jp

Hiroto Yasuura
System LSI Research Center,
Kyushu University
305-3-8-33 Momochihama,
Sawara-ku, Fukuoka 814-0001 JAPAN
+81-92-847-5190
yasuura@slrc.kyushu-u.ac.jp

Akira Fukuda
The Department of Computer Science and Communication Engineering,
Kyushu University
744 Motooka Nishi-ku, Fukuoka
819-0395, JAPAN
+81-92-802-3644
fukuda@f.csce.kyushu-u.ac.jp

Tsuneo Nakanishi
The Department of Computer Science and Communication Engineering,
Kyushu University
744 Motooka Nishi-ku, Fukuoka
819-0395, JAPAN
+81-92-802-3644
tun@f.csce.kyushu-u.ac.jp

ABSTRACT

The System LSI Research Center (SLRC), Kyushu University, has launched a project educating System LSI designers referred to as QUBE since 2005. The intended applicants for the QUBE are senior engineers who are required to have wide knowledge and practical skills on hardware, embedded software, and system design. QUBE made a curriculum that students can learn advanced hot topics in each technical region and exercise to design and implement System LSI which consists of hardware and embedded software. QUBE originally developed an exercise centric course material for latter program in the curriculum. QUBE provides 16 classes (35 days) in 18 weeks according to the plan in 2005 school year. 106 applicants ware submitted to classes. 89% students ware satisfied with QUBE classes according to questionnaire.

Keywords

Education, SoC, System LSI, Architect, Co-Design

1. INTRODUCTION

The importance of System LSI is increasing. System LSI are embedded to various products such as cellar phones, TV, *etc*. It is, however, difficult to design appropriately System LSI considering aspects of hardware, software, and also business. It requires wide knowledge and know-how of technology in such multiple areas for engineers to design and develop System LSI. In Japan, universities provide classes in which professors lecture individual technologies only. The class integrating various technologies to learn designing System LSI was not provided. There is no other education organization to educate engineers required to develop System LSI. Industry under heavy cost competition cannot use budgets for educating engineers.

The System LSI Research Center (SLRC), Kyushu University, has launched a project educating System LSI designers referred to as QUBE since 2005. The intended applicants for the QUBE are senior engineers who are required to have wide knowledge and practical skills on both hardware and embedded software design. The QUBE also intends researchers who want to know technologies used in the field as its target.

This paper is organized as follows: Section 2 introduces the QUBE project and Section 3 describes related works and QUBE properties. Section 4 describes its missions and Section 5 shows the curriculum of the QUBE. Section 6 explains newly developed course referred to as System LSI Design Training Course. Section 7 reports the activities of the QUBE in 2005. Finally, section 8 concludes this paper.

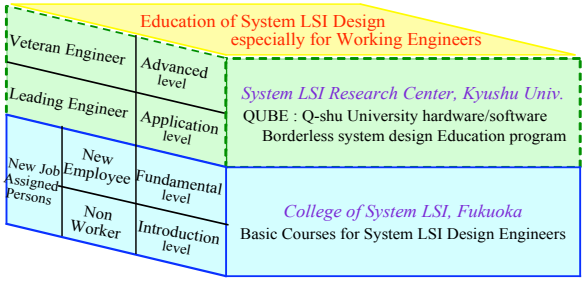


Figure 1 The consistent curriculum of the QUBE and College of System LSI, Fukuoka

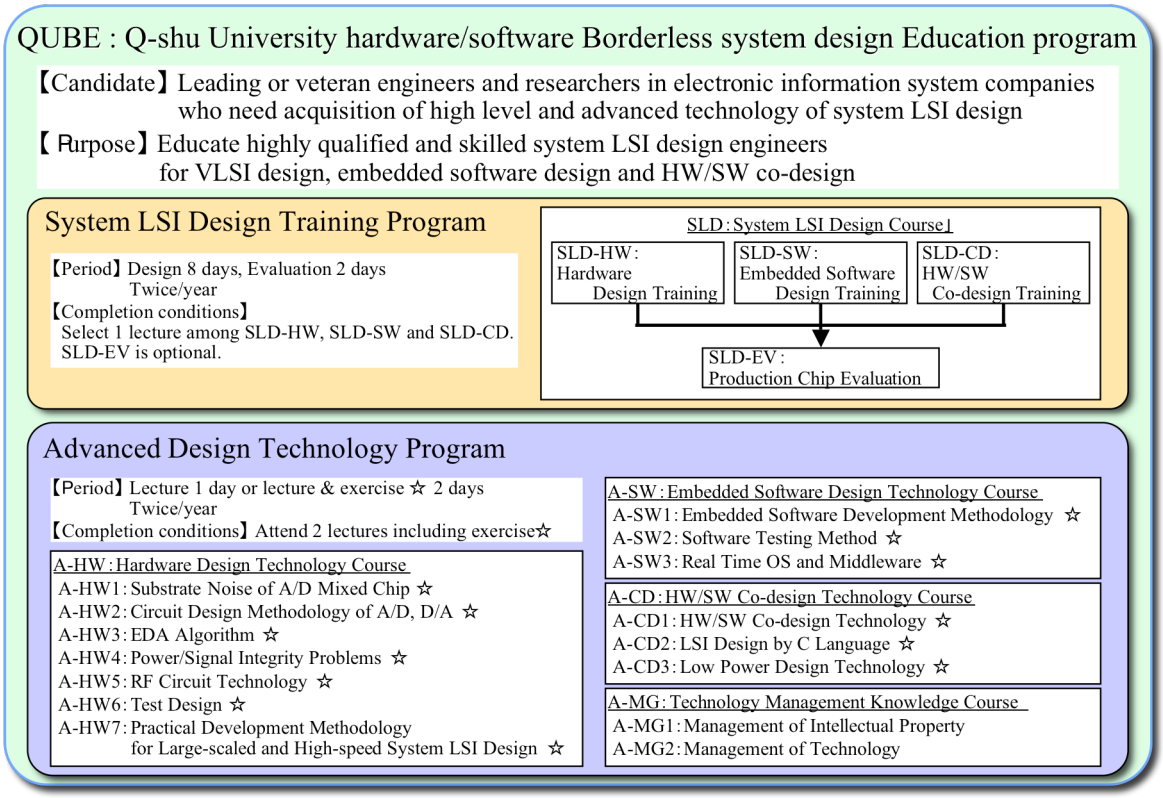


Figure 2 The Curriculum run in 2005

2. PROJECT OVERVIEW

This section explains the QUBE project overview.

The QUBE has begun as a one of a unit in System LSI Research Center, Kyushu University funded by Ministry of Education, Culture, Sports, Science and Technology, Japan. The QUBE is a five year project from July, 2005 to March, 2010. The QUBE is administered by the QUBE project members consisted of four full-time staffs and three concurrent staffs. The QUBE has a advisory committee, which consists of the QUBE project members, professors invited from Kyushu University, other universities, and industries for the QUBE classes, and members Teaching Staff Meeting of College of System LSI, Fukuoka[1].

Before QUBE, Prof. Yasuura who is a leader of the QUBE began College of System LSI, Fukuoka for educating entry-level hardware designer since 2001[1]. The QUBE has begun to provide advanced-level education for System LSI designers, hardware engineers, and embedded software engineers. To educate those engineers from entry-level to advanced level QUBE cooperates with College of System LSI, Fukuoka and make a consistent curriculum as shown in Figure 1. College of System LSI, Fukuoka provides entry and basic level classes and QUBE provides advanced-level classes.

3. RELATED WORKS

This section introduces some related works that educates SoC design, LSI design and embedded software. There are many education courses to educate these topics in the world. The SoC Consortium of Taiwan[2] is developing curriculum and course materials to educate LSI design and embedded software for master course students. NEXCESS[3] is developing them to educate embedded software for industrial engineers.

The specialty of the QUBE is that the targets of the QUBE are industrial engineers. Most targets of education courses like SoC consortium of Taiwan are students who belong to universities. It is required to design QUBE courses especially for industrial engineers. For example, each lecture for them should keep short since they are too busy to take long lecture. One of merits to set a target to industrial engineers is that the QUBE can obtain industrial needs directly. In feature, curriculum and course materials developed by the QUBE according to industrial needs will be used as one for educating master course and/or undergraduate students.

The next specialty is that QUBE has a integrated curriculum including hardware design, embedded software, hardware / software co-design, and also managements skills for engineers who develop System LSI.

4. MISSIONS

The mission of QUBE is to educate System LSI designers who can design value added System LSI considering with both of hardware and embedded software. System LSI designers consists of hardware engineers, embedded software engineers, and hardware / software co-design architects. The targets of our program is senior engineers who would like to learn advanced technical topics. These engineers are basically employed by electrical and/or information related industry.

The goal of QUBE is to educate 180 engineers after 3 years, 360 engineers after 5 years who have skill levels as follows:

a) Hardware Engineers:

Hardware engineers are not only able to design a part of hardware of the System LSI, but they also can understand requirement specifications and solve problems related with hardware / software interfaces.

b) Embedded Software Engineers:

Embedded Software Engineers are not only able to design and implement a part of software of the System LSI, but they also can understand requirement specification and solve problems related with hardware / software interfaces.

c) Hardware / Software Co-Design Architects:

Hardware / Software Co-Design Architects can design System LSI and write requirement specifications both of hardware and embedded software appropriately. Architects can understand tradeoffs between hardware and embedded software implementation and design System LSI considering with these tradeoffs.

QUBE provides practical exercise centric classes to educate these engineers and architects as follows: a) QUBE invites top professors those who research and develop something related to hot topics. b) QUBE develops exercise centric classes in which students can use EDA tools and development environments employed at fields in industries. QUBE employs tools provided by VDEC (VLSI Design and Education Center, Tokyo University) [4] and design and verification lab serviced by Industry Science Technology Foundation, Fukuoka to compress costs using these tools[5].

5. CURRICULUM

In this section, we describe the curriculum of QUBE project. QUBE curriculum aims at educating three technical domain of System LSI; hardware engineer, embedded software engineer, and hardware / software co-design architect as mentioned in previous section. The curriculum run in 2005 is shown as Figure 2.

The curriculum consists of two parts as follows:

(1) System LSI Designer Training Program

System LSI Designer Training Program aims at educating engineers who can design systems considering both aspects of hardware and embedded software. This program consists of two classes: a) System LSI Design Training and b) Trial Chip Evaluation Lab. b) will be run in 2006.

System LSI Design Training class aims at educating all of hardware design, embedded software design and implementations, and co-design. This class also aims that students can be learned practical design and implementation skills through exercise. In this exercise, students implement something using hardware and embedded software on the processor embedded FPGA board.

In this class, students form some teams and each team implement a system. A team member consists of hardware design engineers, embedded software engineers, and hardware/software co-design architects. Engineers can understand other technical domains and get a chance to learn how to work with engineers in other technical domains. Architects can learn co-design skills and try them in a team.

This class is originally developed by QUBE. We describe detail of this class and its material in section 4.

(2) Advanced Design Technology Program

Advanced Design Technology Program aims at educating deeply professional design technology in each technical domain. In this program, QUBE also provides System LSI related management classes. Classes in this program are provided by professors engaged in System LSI related advanced research and development. These professors are invited from Kyushu University, other universities, and industry. In this program,

This program consists of four courses:

1) Hardware design technology course

   This course provides hot topics related to hardware design technologies for System LSI such as noise, power/signal integrity, RF, large-scaled design, *etc*.

2) Embedded software design technology course

   This course provides topics about development methodologies, test, RTOS, and middleware for software embedded into System LSI.

3) Hardware / software (HW/SW) co-design technology course

   This course provides topics for co-design technologies such as development methodology using ASIP, C-based design and low-power design for System LSI.

4) Technology management knowledge course

   This course provides topics for managing System LSI projects such as intellectual properties and management of technology.

We make session time of classes in this program short and provide multiple classes in one year to ease taking classes for engineers who have business. Session time of classes are between one day and a week.

6. SYSTEM LSI DESIGN TRAINING CLASS

This section explains the System LSI Design Training Class.

6.1 Lecture Plan

This class consists of four parts; lecture, tutorial, exercise (analysis, design, implementation), and presentation. QUBE plans to maximize exercise time to make this course practical. QUBE provides tutorials to make students familiar with the target board and development environment like [6].

AM of 1st , 2nd day: The Lecture

At AM of 1st and 2nd day, QUBE lectures knowledge to design and implement System LSI as follows:

1) About System LSI

2) Viewpoint of System Design

3) Basics of Hardware Design

4) Basics of Software Design and Implementation

5) Cost model of System LSI

6) Low Power Design for System LSI

7) HW / SW Co-Design

PM of 1st, 2nd day: Tutorial

At PM of 1st, 2nd day, QUBE provides tutorials to make students familiar with development environment and to learn skills to develop System LSI.

Professors lecture to use development environment step by step. Professors incrementally give students exercises to connect a processor and original hardware both of designing hardware and developing device drivers as follows:

1) Write a device driver for given LED controller

2) Design a original LED controller

3) Design a push button switch controller and Write device driver for it

3rd, 4th day: Analysis / Design

At first of the 3rd day, professors explain the exercise target. Students analyze and design it and make documents at the 3rd and 4th day.

4th – 7th day: Implementation / Verification / Test

At the 4th to 7th day, students design hardware and/or implement embedded software. Hardware and software components such as OS, middleware, *etc.,* are provided in order to decrease developing time. Students develop an exercise target using these components.

8th day: Presentation

At the last day, each team gives presentations including the results of analysis and design. They also demonstrate a system implemented by them. Professors request a peer review to their documents. Students discuss about their design based on their documents.

6.2 Multimedia Phone

QUBE select Multimedia Phone (MMP), which is a multi functional phone, as exercise target. Overview of MMP is shown as Figure 2. MMP has call / hook function, voice communication function, MMP discovery function, and white board sharing function. MMP equips microphone and speaker for voice communication. MMP also equips a LCD and a mouse for white board sharing. There are multiple MMPs in a network, and MMPs are connected via Ethernet.

To make exercise target scalable, QUBE provide two function profiles: basic profile and extended profile. First, students implement MMP according to the basic profile. If exercise time remains, they extend it according to the extended profile. Thus, the exercise target can be applicable to students who have various skill levels.

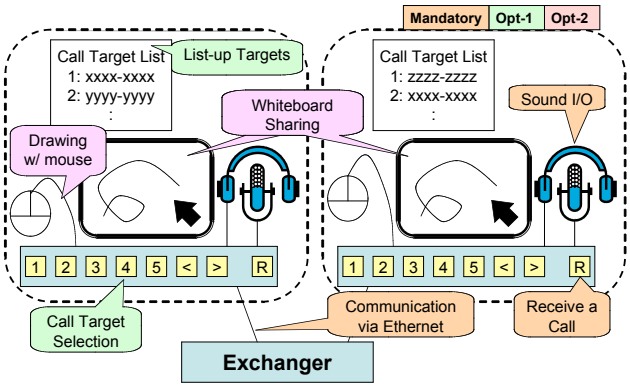


Figure 3 Overview of the Multi Media Phone

For the exercise environment, in which students can implement MMP, QUBE provides Xilinx XUP Virtex II Pro board and its development environment. The board equips a FPGA (Virtex II Pro) embedded two PowerPC 405 processors, an ethernet interface, an audio codec based on AC97, two serial interfaces, some buttons, a VGA interface, a keyboard interface, and a PS/2 mouse interface. QUBE employs ISE and EDK as a development environment. QUBE also provides some debug tools such as Ethereal which is a packet capturing tools, WinPcap which is a library to develop software dealing with ethernet frame, and MMP emulator executed on PCs.

6.3 1st run Report

This subsection reports briefly the 1st run of System LSI Design Training Class at March 1 to 10, 2006. Five applicants from industries and two applicants from a university subscribed to this course. QUBE made two teams from them.

Basically, professors lectured according to the plan mentioned in section 5.1. QUBE also provide short lectures in which professor lecture basic topics requested by students who want to know such topics. In this time, mechanism of interrupt, OS, and drivers are requested. The students want the short lecture due to engineers are not educated enough about basic topics at fields in industries in Japan.

After lectures, each team designed and implemented the MMP respectively. Development speed was slower than supposed one at course designing phase.

Finally, each team gave the presentation of the design of MMP and demonstrated their MMP. They designed and wrote

Table 1 Statistics of the applicants

	organizations		registered		total applicants						Diploma		taken classes /a student	pass ratio
					applicants		cancelled	absenced	passed					
Total	40		106		149		15	2	98		20		1.25	74%
Non students	37	93%	87	82 %	124	83%	14	2	89	91%	20		1.24	82%
Industries	32	80%	76	72 %	101	68%	12	2	76	78%	16	80%	1.14	87%
Non industries	5	13%	11	10 %	23	15%	2	0	13	13%	4	20%	1.91	62%
Students	3	8%	19	18 %	25	17%	1	0	9	9%			1.26	38%

documents satisfactorily. The team of students from industries designed according to eUML[7], which is a development methodology for embedded system based on UML. Another team designed according to PLUS[8], which is a one of the product lines development methodology. Each team implemented the MMP based on basic profile only. They could not implement functions as hardware logics.

To run this 1st class, QUBE developed course materials such as lecture slides, software and hardware components for learning knowledge and know-how to lead projects in which engineers should design both of hardware and software.

QUBE found a problem through running this 1st class that the exercise time is too short to implement functions using both of hardware logics and software. The reason is that skill levels of students are lower than supposed one. QUBE has a plan to extend exercise target to cover students who are in entry level.

7. PROJECT ACTIVITIES in 2005

In this section, we report results of QUBE classes run in 2005. Its statistics are shown as Table 1.

We provided 16 classes (35 days) in 18 weeks according to the plan. 106 applicants ware registered to QUBE classes. Most of registered applicants ware belong to Fukuoka area as same prefecture as QUBE. Total 132 applicants took these classes and the average applicants in each class are 8.3.

98 students passed these classes. We supposed the pass ratio is over 90% at designing the QUBE curriculum. But real pass ratio was 74%. Its reason is that levels of each class ware higher than students' skills. The pass condition is that taken point is over or equal 70 points. The points consist of 40 attendance points, 10 report submission points, and 50 lecturer points.

Diploma students are 20, through the set points of diploma students are 360 in 5 years. These reasons are numbers of students is less than supposed one and the supposed pass ratio is too high as mentioned. The diploma condition is that a student should pass one class in System LSI Designer Education Program or two classes in Advanced Design Technology Program.

QUBE requested writing questionnaires to the students. The result of the questionnaires is as shown in Table 2. Most of the motivation to take QUBE class was applicants' wish for learning knowledge and skills supposed to use in near feature. Engineers in fields in industries have motivation to learn new knowledge and skills themselves. Satisfaction for their prospect is 74%, which is sum of percentage of very good and good. Their feelings of the levels of the classes, however, are distributed by class. Total points of their satisfaction are 89%, which is sum of percents of very good and good.

QUBE can provide classes satisfied by students according to this questionnaire. However, QUBE should tackles to improve pass ratio. One of the cause of such low pass ratio is applicants don't know required skill level to understand classes enough. QUBE have a plan to improve it to clarify target skill level that applicant can understand lectures and to clarify the goal skill level that applicants can understand to take lectures according to the Embedded Technology Skill Standard[9]. QUBE also tackles to adjust levels of classes to applicants needs. QUBE also make publicity to industries in Kyushu area to increase applicants.

8. CONCLUSIONS

This paper introduced the Education Program for System LSI Designers referred to as QUBE. QUBE aims at educating system LSI designers who have wide knowledge and practical skills both of hardware and embedded software.

QUBE made a curriculum that students can learn advanced hot topics in each technical region and exercise to design and implement System LSI which consists of hardware and embedded software. QUBE originally developed an exercise centric course material for latter program in the curriculum. QUBE provides 16 classes (35 days) in 18 weeks according to the plan in 2005 school year. 106 applicants ware submitted to classes. 89% students ware satisfied with QUBE classes according to questionnaire.

In the feature, QUBE will improve course materials constantly to adapt new technologies. QUBE also will tackle to improve pass ratio adjusting a level of classes to applicants needs. QUBE make publicity to industries in Kyushu area to increase applicants.

Table 2 The result of questionnaire to the students

1. Motivations	1.1 What occasion did you take this class.	a) As an induction course	7	6%	2. Remarks to the class (cont'd)	2.3-1 How did you understand topics in the lecture?	a) Very good	28	26%
		b) Advice from your boss	24	22%			b) Good	38	35%
		c) Your wish	75	69%			c) Average	32	29%
		d) Others	4	4%			d) Bad	11	10%
	1.2 Why did you take this class?	a) for current bussiness	36	33%			e) Very bad	0	0%
		b) for business in near feature	45	41%		2.5 How did you feel about the text?	Describe good points	67	61%
		c) not for your business	31	28%			Describe bad points	47	43%
		d) Others	7	6%		2.6 What did you find something while the lecture?	Describe good points	66	61%
2. Remarks to the class	2.1-2 How was you satisfied with your prospect?	a) Very good	37	34%			Describe bad points	40	37%
		b) Good	48	44%		2.7-1 Time of the lecture	b) not appropreately	12	11%
		c) Average	19	17%		2.8 Total point	a) Very good	47	43%
		d) Bad	5	5%			b) Good	50	46%
		e) Very bad	0	0%			c) Average	12	11%
	2.2 How did you feel about level of the lecture?	a) Easy	57	52%			d) Bad	0	0%
		b) Difficult	52	48%			e) Very bad	0	0%

REFERENCES

[1] Yasuura, H., Tsukizoe, A., Hirakawa, K., Ito, N., and Nakano, S., ``College of System LSI, Fukuoka: A Callenge to Sillicon Sea Belt (in japanese)", Journal of IEICE, Vol.86, No.11, pp.857-863, 2003.

[2] Huang, T., King, C., Lin, S. Y., and Hwang, Y.,: The Embedded Software Consortium of Taiwan, ACM Transactions on Embedded Computing Systems, Volume 4, Issue 3, pp. 612 – 632, 2005.

[3] Yamamoto, M., Tomiyama, H., Takada, H., Agusa, K., Masa, K., Kawaguchi, N., Honda, S., Kaneko, N.: NEXCESS: Nagoya University Extension Courses for Embedded Software Specialists, Proc. 2005 Workshop on Embedded System Education (WESE2005), pp.16-20, 2005.

[4] VLSI Design and Education Center: About VDEC, http://www.vdec.u-tokyo.ac.jp/English/index.html.

[5] Institute of System LSI Design Industry, Fukuoka: The design and verification lab (in Japanese), http://www.ist.or.jp/lsi/.

[6] Edwards, S. A.: Experiences Teaching an FPGA-based Embedded System Class, Proc. 2005 Workshop on Embedded Systems Education (WESE2005), 2005.

[7] Watanabe, H., *et al* : Embedded UML: Object Oriented Development Using eUML(in Japanese), Syoueisya, 2002.

[8] Gomma, H.: Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures, Addison-Wesley, 2004

[9] IPA, Japan: Embedded Technology Skill Standard (ETSS), http://sec.ipa.go.jp/std/skillex_eb.php (in Japanese).

The Embedded Software Consortium of Taiwan - A Progress Report of Educational Activities

Tai-Yi Huang and Chung-Ta King
Department of Computer Science
National Tsing Hua University
{tyhuang, king}@cs.nthu.edu.tw

Shih-Hao Hung
Department of Computer Science and Information Engineering
National Taiwan University
{hungsh}@csie.ntu.edu.tw

Abstract

To meet the challenges brought by recent advancement on semiconductor manufacturing technology to develop system-on-chip (SoC) techniques, the Ministry of Education (MOE) of Taiwan has been running the VLSI Circuits and Systems Education Program since 1996. This program is currently at its third phase of revision, which starts at 2006 and ends at 2010. The Embedded Software (ESW) Consortium is one of the 8 domain-specific, inter-collegiate consortia funded by the MOE under this program. The goal of ESW is to address the challenges of embedded software development for SoC systems. This paper first introduces the VLSI Circuits and Systems Education Program. The organization and activities of ESW is described next. Finally, we present an execution summary of ESW in the past three years.

Keywords: embedded software, integrated circuit design, educational programs

1 Introduction

The continuous advancement in nanometer semiconductor manufacturing, integrated circuit (IC) design, and system-on-chip (SoC) techniques has made practical to place on a single chip a complete computing system. The key to the success in the development of a SoC industry depends on a large number of well-educated engineers on IC design, embedded software, and system integration, in addition to supporting personnel in business, management, and law. The Ministry of Education (MOE) of Taiwan has been running the VLSI Circuits and Systems Education (CSE) Program since 1996. The program is to provide our local semiconductor industry with its needed technology and engineers, both in quality and quantity. Phase I of this program began in 1996 and ended at 2000. The objective of this phase was primarily to establish fundamental IC design environments at universities in Taiwan to promote IC design research and new educational curricula. Phase II of this program continued at 2001 and ended at 2005. To make its decision process and execution efficiently, Phase II adopted a top-down approach by forming six domain-specific, inter-collegiate consortia. This consortium architecture is further expanded into eight consortia in Phase III that begins at 2006 and will end at 2010.

The ESW Consortium, established in February 2004, is the latest consortium founded by the MOE under the VLSI CSE Program to address the challenges of embedded software development for SoC systems. A SoC system, different from a traditional board-level system, places on a single chip all system components, including one or more microprocessors, cache, memory, and communication modules. Traditional board-level system software cannot fully utilize the resources of a SoC system if directly ported. In addition, these software are often developed to support a wide variety of platforms. In contrast, a SoC system software is mostly designed for a specific platform. The ESW Consortium intends to achieve its goal by integrating resources from the government of Taiwan, domestic and foreign academic societies, and local semiconductor industry. The programs of ESW include academic and industrial forums, international

collaborations, student activities and promotion, and a new embedded software curriculum.

The VLSI CSE program of Taiwan and the ESW Consortium was formally introduced in [1]. We described the Phase II of the CSE program, the supply-demand analysis of the local semiconductor industry, the six-consortia architecture, and an execution summary of ESW in 2004. In this paper, we update the recent progress of the CSE program and the ESW Consortium. We first briefly describe the Phase III program and the eight-consortia architecture. We next provide an ESW execution summary in 2006. The execution report focuses on the 2006 Embedded Software Design Contest (ESDC) and the ESW hands-on lab map. The ESDC, hosted by the MOE of Taiwan, is an annual event in which students compete with their constructed embedded systems in both creativity and advanced technology. This event has been used by ESW to evaluate and demonstrate the performance of their reformed ESW curriculum. A number of hands-on labs are produced during the development of the ESW curriculum. We started a new project in 2005 to classify and integrate all these labs into a comprehensive map of labs. This map serves both as a handbook for professors to select labs suitable to their courses and as a guide for ESW to advance its curriculum. Finally, due to the space limit, we skip the update on the ESW curriculum that includes more courses than its shape in 2004.

The rest of the paper is structured as follows. Section 2 introduces the Phase III of the CSE program. Section 3 presents the latest results of ESDC. Section 4 describes our efforts in constructing a comprehensive map of hands-on labs. An execution summary of other ESW activities is reported in Section 5. Finally, Section 6 concludes this paper.

2 The VLSI CSE Program

The VLSI CSE program was initiated in 1996 by a group of senior professors and industry leaders who envisioned that high-quality IC design engineers will be essential to its local semiconductor industry. The execution of the VLSI CSE program can be divided into three phases. The Phase I executed from July 1996 to December 2000. Its primary goal was to establish advanced VLSI design curricula in around 20 research-oriented universities. The Phase II executed from January 2001 to December 2005. Recognizing that SoC is a must-have technology, Phase II took an aggressive top-down approach to form a six-consortia architecture. A top-level office named the SOC Consortium [6] hosts all cross-consortia activities while

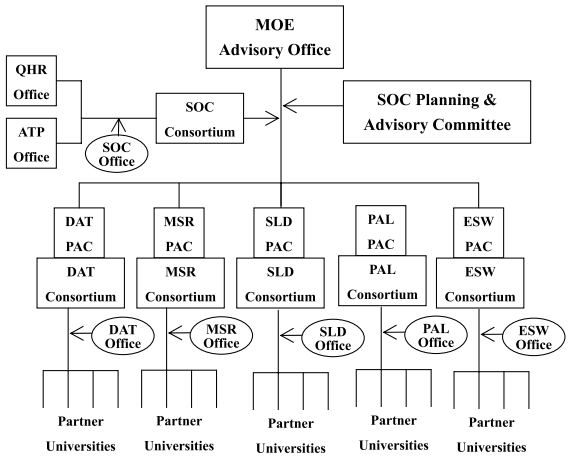


Figure 1: The eight-consortium architecture

each sub-level consortium manages its own domain-specific events. Along with the execution of the CSE program, the MOE founded the Chip Implementation Center [3] that offers its service of manufacturing chips designed by students. Total funding for Phase I and Phase II are US$3M and US$15M, respectively. In addition, there are around 1,000 chips fabricated and tested by the CIC each year.

Phase III starts at January 2006 and will end at December 2010. Its purposes are two folds: to continue what we have achieved and to make our results visible to international communities. To adapt to its fast growth in scope and impact, Phase III expands its consortium architecture to an eight-consortia one, as shown in Figure 1. There are four levels of administrations. The advisory office of the MOE defines general guidelines and approves budget planning. The SOC Consortium coordinates and monitors activities among all consortia. The DAT (Design Automation & Testing), MSR (Mixed Signal & Radio Frequency), SLD (System-Level Design), PAL (Prototyping Application & Layout), and ESW (Embedded Software) are domain-specific consortia, each of which has a group of partner universities to define its curriculum and activities. Both the QHR (Quality Human Resources) and ATP (Advanced Teaching Platform) consortia are added in Phase III to coordinate cross-consortia activities. QHR hosts all programming contests, invites international speakers, and sponsors cross-consortia activities. These contests are the IC Design Contest, the Silicon IP Authoring Contest, the EDA Programming Contest, and the Embedded Software Design Contest. Each contest is held annually to provide a platform for student competition and performance evaluation. ATP pro-

motes new curricula developed in all consortia by providing the financial support needed to adopt a new curriculum. The total funding for Phase III is increased to US$20M proximately.

The organization of the ESW Consortium remains much the same as we reported in [1]. The ESW office is currently based at National Tsing Hua University and chaired by Professor Chung-Ta King. Professor Tai-Yi Huang serves as the executive secretary since ESW was first founded in February 2004. Its planning and execution is supervised by an independent Planning and Advisory Committee (PAC) consisting of one senior researcher and four industry leaders. Its programs are classified into five main categories: embedded software forums, student activities and promotion, international cooperation programs, industrial strategic forums, and the embedded software curriculum. There are totally 22 partner universities of which 46 professors are involved in planning and executing ESW activities. All information and promotions are published and accessible at its official web site [5].

3 The Embedded Software Design Contest

The annual ESDC event is sponsored by the MOE and organized by the QHR Consortium. The first ESDC was held in 2003 for the purpose of demonstrating the capability of embedded software design and development. Since 2006, ESDC is organized by Professors Tei-Wei Kuo and Shih-Hao Hung at National Taiwan University. A full ESDC schedule spans over four months. Each team can have no more than three students, with no restriction on undergraduate and graduate students. Each team registers by submitting a proposal describing its project scope and working list. Each proposal is reviewed by the program committee to provide early feedback and suggestions. There is a period of two months for students to complete their working list. A final report is due two weeks prior to the final on-site demonstration. The score of a project is based on its final report (30%), its oral presentation (20%), and the system demonstration (50%). We evaluate a project by its software originality and creativity, soundness of software engineering, and system performance, depending on which category of competition it is registered for. A winning team is financed as an award to attend one international conference related to embedded system and software.

Figure 2 shows the total number of registered teams from 2003 to 2006. There were only 30 teams

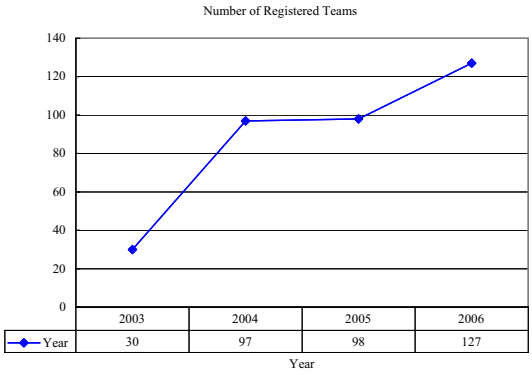


Figure 2: The number of registered teams

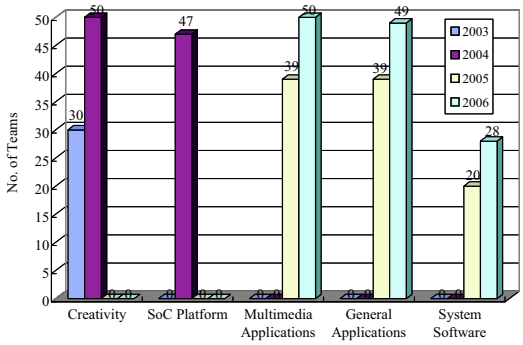


Figure 3: The number of teams in each category

in 2003 when ESDC first took place. This number increases significantly to 97 in 2004 when the ESW Consortium was founded. As the ESW curriculum continues to develop and be deployed in more universities and colleges of technology, this number reaches a record-high of 127 in 2006.

In 2003, there is only one contest category of Creativity. In 2004, ESDC added a new category of SoC Platforms to attract more hardware-centric students. Each team is allowed to register only in one category. Figure 3 shows the number of registered teams in each category. A team registered for Creativity has an open choice of embedded platforms, and their work was evaluated on its software originality and creativity. A team registered for SoC Platforms can only work on a list of designated platforms and was evaluated on its software efforts in design tradeoffs and resource optimizations. Starting in 2005, we made ESDC as a thee-category contest based on areas of applications. The category of Multimedia Applications is there in response to the ever-growing inter-

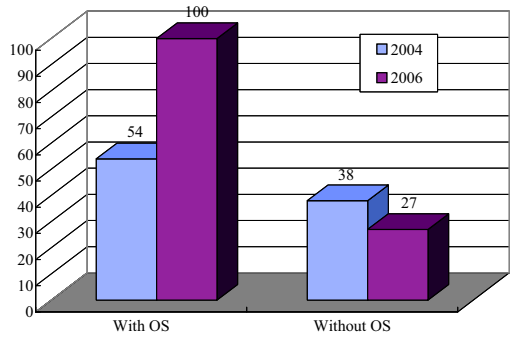


Figure 4: The types of embedded systems

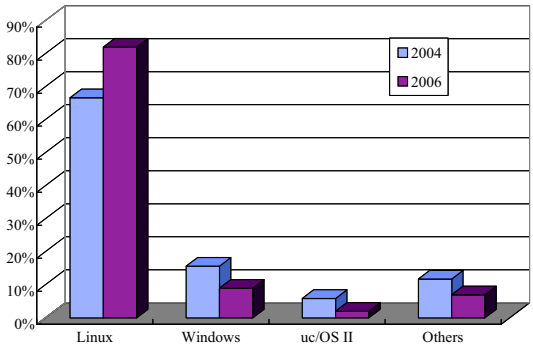


Figure 5: Classification by operating systems

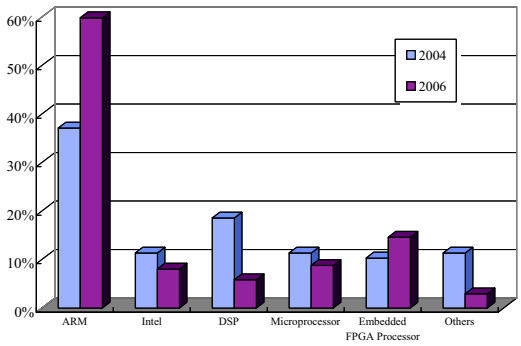


Figure 6: Classification by processors

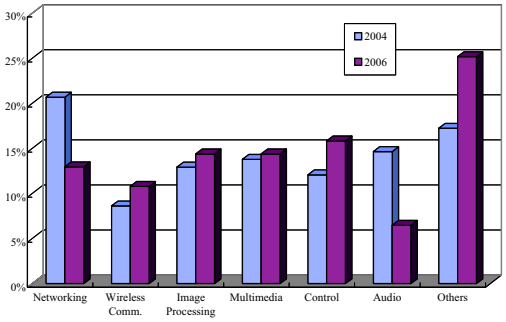


Figure 7: Classification by topics

ests and ideas in multimedia. The category of General Applications provides the arena for students to showcase their creativeness and skills without limit of topics and platforms. Finally, the category of System Software is designed to encourage the research and development efforts in embedded system software such as the operating systems, middleware, compilers, and device drivers.

Figure 4 shows the number of projects that are managed by an operating system or by a mere loop scheduler in 2004 and 2006. It is an interesting observation that more students choose to work with an operating system to implement complex applications, manage real-time tasks, and handle sophisticated embedded platforms. It is also an indication of stronger competition and more advanced skills of participated students from 2004 to 2006. Furthermore, Figure 5 breaks down the types of operating systems adopted by students. In 2006, 82% of participated teams chose Linux, higher than the 67% adoption in 2004. This shows the dominance of Linux in our academic society of embedded software. This dominance results from the flexibility and availability of Linux-based development kits and strong educational and industrial support behind Linux.

Figure 6 shows the classification by processors. The percentage of ARM-based processors increases from 37% in 2004 to 60% in 2006, well demonstrating the dominance of ARM-based processors in the local market. There are more network and multimedia applications developed with FPGA prototyping in 2006 while there are less projects utilizing DSP processors. If we classify projects by user-mode and kernel-mode, about 80% of projects are user-mode applications while 20% of them are kernel-based modules including operating systems, device drivers, and system-related tools. We noticed this distribution since 2004 and have made a strong effort to encourage and promote the development of kernel-level software. In fact, the category of System Software is created to address this need. The improvement of technology and skills demonstrated by projects in this category is considered superior to what we observed in the other two categories.

Figure 7 further classifies user-mode projects by domains of applications. Generally speaking, net-

Year	Category	Project Topics
2005	Multimedia Applications	An integrated design of OSGi gateway and DVB interactive platform
		An intelligent digital camera with adjustable power and performance
	General Applications	e-Housekeeper
		Realization of human-robot interaction via vision capturing
	System Software	Software-less SCAN sensor
		A memory protection mechanism in an embedded OS
2006	Multimedia Applications	An intelligent guide for emergency fire exit
		Embedded H.264 video streams
	General Applications	3G entertainment - Good, Great, General
	System Software	A low-power real-time operating system
		A fast multi-core simulation framework with synchronization IP

Table 1: The topics of winners in each contest category

working (including wired and wireless) applications and multimedia (including audio and image processing) applications are favored. The category of Others includes projects in the domain of security, health, and kernel-level topics. Finally, Table 1 lists the winners in each contest category of 2005 and 2006. Most winners of 2003 and 2004, available in [1], use multimedia applications to demonstrate their embedded software. On the other hand, many winners in 2005 and 2006 developed embedded systems that efficiently integrate system resources of hardware and software. The latter winner certainly requires more understanding and training on embedded software. This trend is again considered as an indication to the success of the ESW curriculum.

In summary, after four years of strong competition, our experience from ESDC confirms that this government-sponsored annual event has become a great force in promoting embedded software research and educational programs. In addition, it also helps to reveal the strength and weakness in our ESW curriculum. The program committee of ESDC continues to observe increased quality and complexity in the proposed projects. An active item of future work is to encourage more kernel-level embedded software and increase the number of teams in System Software.

4 The Hands-on Lab Map

The development of the ESW curriculum is to provide a set of lab-centric courses. Each course was first designed by a group of four professors and was later adopted by interested universities. An adoption of a course receives financial support by the MOE to setup a lab for its purpose. In return, each adoption is required to provide at least 4 sets of hands-on labs at its final review. The ESW office has collected 80 labs from 8 different courses. We label each lab with an index of *CourseNum-UnivNum-LabNum*. For example, a lab module with a label of 002-01-03 indicates that it is the third lab module developed by the first university for the second course.

For ease of reference, we classify these 80 modules by three categories: platforms, operating systems, and tool chains. There are currently 14 embedded platforms and 5 different operating systems. Due to the space limit, we only give out partial information of each classification table in the following. The complete tables are accessible at the official web site of ESW [5].

4.1 Classification by Platforms

Table 2 gives part of our classification by platforms. The first column lists the platforms used in each set of labs. The second column lists the operating system used by these labs. An empty slot of this column indicates that no operating system is installed. In contrast, if an operating system is ported as a lab, we provide its label for reference. The third column gives the topic of this lab and the fourth column gives its label. This table helps ESW professors to easily identify the platform suitable to their teaching needs. In addition, it serves as a map to develop unavailable modules for a selected platform, avoiding redundant efforts.

4.2 Classification by OS

There are 5 different operating systems used among all lab modules. Figure 8 shows the name of these operating systems and the number of labs installed with each operating system. Linux-based operating systems are strongly favored by these hands-on labs. Table 3 gives part of our classification by operating systems. The first column shows the name of the

Platform	OS	Topic	Label
Creator S3C2410	Linux 2.4 [001-02-02]	LCD Display	[001-02-01]
		MP3 Player	[001-02-03]
		RT-OS module FireLinux	[001-02-04]
NET-Start!W3001 ARM S3C4510B	uClinux 2.0 [001-03-04]	Bootloader	[001-03-03]
		Switch and 7-segment	[001-03-05]
		Interrupt Handlers and Time Delay	[001-03-06]
		DMA	[001-03-07]
	uC/OS [001-03-08]	Scheduler: Priority-inversion Issue	[001-03-09]

Table 2: The classification of labs by platforms

OS	Platform	Topic	Label
Linux	Creator S3C2410	Porting Linux 2.4	[001-02-02]
		MP3 Player	[001-02-03]
		RT-OS Module FireLinux	[001-02-04]
	TI OMAP 5910	Porting Linux 2.4	[001-04-01]
		IPC (Share Memory & Semaphore)	[001-04-01]
		PThread Programming	[001-04-01]
		Embedded GUI Programming	[001-04-01]
		Networked Digital Camera	[001-04-01]
uClinux	NET-Start!W3001 ARM S3C4510B	Poring uClinux 2.0	[001-03-04]
		Bootloader	[001-03-03]
		Switch and 7-segment	[001-03-05]
		Interrupt Handlers and Time Delay	[001-03-06]
		DMA	[001-03-07]

Table 3: The classification of labs by operating systems

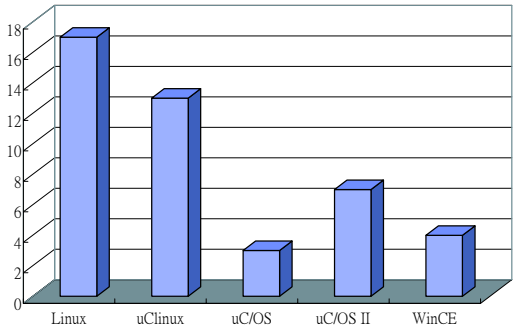


Figure 8: The number of labs with each OS

Topic	Label
ARM Development Suite	[001-03-01]
	[001-03-02]
	[001-07]
	[001-08]
ARM SDT	[001-05]
Hitool	[001-10-02]
	[003-01-01]
ARM GCC Cross-compiler	[002-01-01]
	[003-01-01]

Table 4: The classification by tool chains

operating system and the second column shows the platform. The third and the fourth columns show the topic of a lab and its label. By this table, an interested professor can easily find out the topics of labs and platforms for her favored operating system. Similarly, a new course on embedded operating systems or related topics can enhance this table by developing unavailable lab modules.

4.3 Classification by Tool Chains

One major category of courses in the ESW curriculum is to develop tool chains for embedded platforms. These tool chains are mainly compilers and debuggers. There are totally 16 labs related to the development of tool chains. We classify these labs by its developed tool. Table 4 gives partial information about this classification.

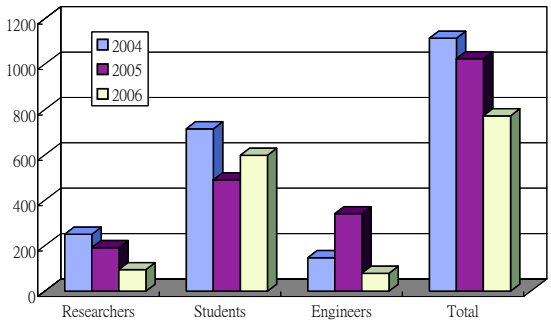


Figure 9: The attendance summary report

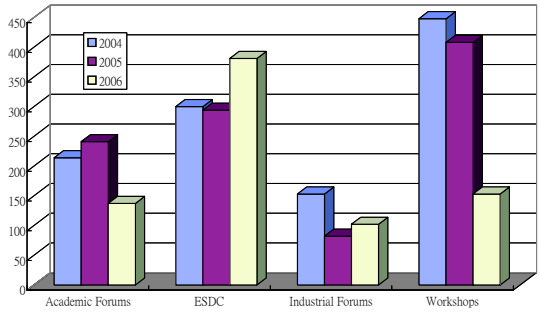


Figure 10: The attendance report by events

5 Execution Summary

Figure 9 shows the attendance report of ESW activities in the past three years. There are totally 1115, 1025, and 773 persons in 2004, 2005, and 2006, respectively. We further break down this number by two different categories. Figure 9 also lists the attendance by roles: academic researchers, students, and industrial engineers. On the other hand, Figure 10 breaks down this number by events: academic forums, ESDC, industrial forums, short courses or workshops. In the following, we analyze this data by events and show some interesting observations. Our comparison mainly focuses between 2004 and 2005. Because the calendar of 2006 is still in its early July dates, no direct comparison will be made.

5.1 Academic Forums

We held four academic forms in both 2004 and 2005. The number of academic forums will be reduced to 3 in 2006 and each forum will have a limit of 40 persons to foster more focused discussions. The attendance increases from 214 in 2004 to 241 in 2005. There are two reasons behind this increase. First, we were only exploring topics when ESW was first founded in 2004. When entering 2005, we had better ideas about what will interest academic researchers. The selection of topics is therefore more customer-oriented. The topics of the forums in 2005 are "Quality of Services and Care for the Elderly", "Building Structurally Stable Embedded Software", "Compilers and Software Development Toolchains for Embedded Systems", and "The Promotion of Embedded Software Research and Education in Taiwan". The other reason of having a better attendance is to co-locate a forum with a short course or a workshop. This strategy proves to be successful for both forums and workshops since an interested audience can attend two events in one trip.

5.2 International Programs

In 2004, we executed the international cooperation program by inviting renowned international scholars to Taiwan to give a short course or a tutorial and exchange research ideas with local researchers. In 2005, we added one more item by sponsoring researchers to attend foreign events such as an international conference or an education program held by an academic institute or a company. We extend this vision in 2006 to actively participate in international activities of embedded software research or education. In fact, this submission, along with other submissions of ESW, to this workshop is part of this effort. At the invitation of international scholars, we invited Professor Lui Sha [4] from the University of Illinois at Urbana-Champaign to talk about the construction of reliable and stable embedded software in September 2005. We also invited Dr. Stephen Kent [2] from BBN Technologies to give a short course about the issues, practices, challenges, and opportunities of VoIP security in July 2006.

5.3 Industrial Forums

The industrial forums of 2004 addressed on subjects that interest both academic researchers and industrial engineers. Starting in 2005, we executed industrial forums to discuss more industrial-focused subjects. As a result, less researchers attended these forums while more engineers were attracted to participate. We considered it a positive change and continue to execute likely in 2006. The forums of 2005 featured two panel discussions, one on the subject of providing the embedded software engineers needed by industry and another one on the subject of constructing profit-oriented business models. We held one industrial forum so far in 2006. This forum featured a panel discussion of research and development

of turn-key technology in embedded software.

5.4 Short Courses and Workshops

To promote ESW events and attract more audience, we co-located academic forums with a short course in 2005. This short course presents a 6-hour content describing the bootstrapping process of Linux, the $O(1)$ scheduler introduced in 2.6, and a couple of hands-on lab to construct a multi-tasking embedded operating system. This short course was held three times, each of which drew more than 100 students and engineers to attend. The success of this short course and its high attendance encourages us to bring together multiple events of similar audience in 2006. In addition, we also held one workshop during the period of ESDC to provide the training required for platform adoption as well as share some successful stories from previous ESDC winners.

The ESW curriculum now consists of 12 courses, 9 of which completed its design in 2005 and another 3 of which were newly added in 2006. The courses that are completed by 2005 have been adopted by 11 universities already. To introduce and promote these courses, we hold at least 2 workshops each year to briefly go over the content and structure of each course. These workshops also provide a platform for interested professors to exchange their teaching experience. In 2006, we further expand this effort to have an overview workshop of all courses and a lab-centric workshop for each course. All these workshops are planned and will be held in August 2006. Finally, the ESW curriculum for technical colleges is currently under development. It is expected to be ready for adoption by September 2006.

6 Conclusions

The MOE of Taiwan has been running this VLSI CSE program since 1996, in order to provide well-trained students both in quality and quantity for its local semiconductor industry. Over the past 10 years, the previous two phases of the CSE program have successfully achieved its educational goal. This program has made Taiwan stayed in a leading role in many sectors of the semiconductor industry. The ESW Consortium, the last consortium founded by the MOE during Phase II, is to address challenges and opportunities in embedded software for SoC systems. Starting 2006, the CSE program enters its third phase of revision. The importance of cultivating talents for embedded software is commonly recognized by senior researchers and industrial leaders in a number of meetings. Accordingly, more resources will be placed on the ESW Consortium during Phase III.

ESW has created a series of events to promote both research and educational reforms of embedded software. These events include academic forums, international collaboration, industrial forms, and student activities. The annual ESDC event has been used to demonstrate and evaluate the execution of ESW. The number of participated teams increases each year and reaches a record-high in 2006. Furthermore, the quality of code developed by the winners is also better than ever. To organize all hands-on labs and make easy future planning, we create a map of labs by classifying all labs by platforms, operating systems, and tool chains. The future work is to continue our efforts in providing high-quality education for more students. Also, we will promote our results in international communities and share our experience.

Acknowledgments

The authors thank Man-Chi Lan for her continuous support in the advisory office of the MOE of Taiwan. We are grateful to have Yu-Hui Lin in the ESW office. Her administrative work is essential to the success of the ESW Consortium.

References

[1] Tai-Yi Huang, Chung-Ta King, Yin-Tsung Hwang, and Youn-Long Steve Lin. The Embedded Software Consortium of Taiwan. *ACM Transactions on Embedded Computing Systems Special Issue on Embedded Systems Education*, 4(3):612–632, August 2005.

[2] Stephen Kent. Chief Scientiest. BBN Technologies. http://www.bbn.com.

[3] National Chip Implementation Center. National Applied Research Laboratories. http://www.cic.org.tw/cicv13/.

[4] Lui Sha. Professor. Department of Computer Science, University of Illinois at Urbana-Champaign. http://www-rtsl.cs.uiuc.edu/ lrs/.

[5] The ESW Consortium. The VLSI Circuits and Systems Education Program. Ministry of Education, Taiwan. http://esw.cs.nthu.edu.tw/.

[6] The SOC Consortium. The VLSI Circuits and Systems Education Program. Ministry of Education, Taiwan. http://www2.ee.ntu.edu.tw/∼vlsi/english/introduction.htm.

Challenges in Designing Embedded Systems Courses

Tulika Mitra
Department of Computer Science
School of Computing
National University of Singapore
tulika@comp.nus.edu.sg

ABSTRACT

This article describes my experience in designing and teaching both undergraduate and graduate level embedded systems modules in School of Computing at National University of Singapore. Designing embedded systems modules for an audience with pre-dominantly computer science background poses some unique challenges. However, once the barrier to entry is crossed, the benefits more than outweigh the difficulties.

1. INTRODUCTION

The Department of Computer Science under the School of Computing at National University of Singapore offers a four-year undergraduate degree program called Bachelor of Computing in Computer Engineering since July 2000 [6]. This program is quite unique in the sense that the focus is exclusively on graduating students with keen appreciation and knowledge in designing complex embedded systems. This program was triggered in response to the growing need of the embedded systems industry in Singapore for graduates with an integrated view of hardware-software design. In addition to the essential computer science related modules (e.g., algorithms, data structures, software engineering, operating systems, databases, etc.), the students under the computer engineering program choose embedded systems related modules specifically designed for this program.

Figure 1 shows the embedded systems modules currently offered under this program and their dependencies. The modules in blue (2000-4000 level) are the undergraduate level modules and the modules in green (5000-level) are the graduate level modules. However, our undergraduate students may choose some graduate level modules to fulfill their degree requirements. Similarly, our graduate students may choose limited number of 4000-level undergraduate modules. Undergraduate modules are offered on a demand driven (essential) basis whereas graduate module offerings primarily depend on the teaching/research interests of the faculty members.

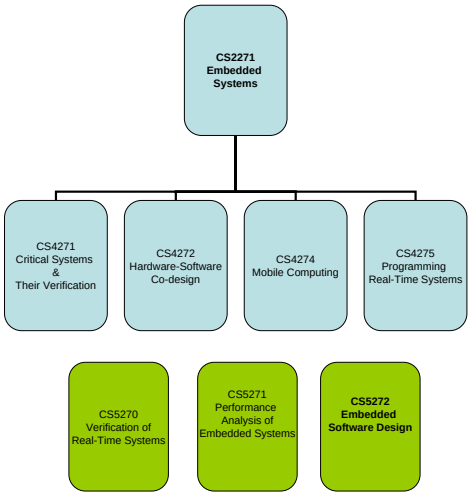


Figure 1: Embedded Systems related modules under the Computer Engineering program.

I designed and taught the undergraduate module **CS2271: Embedded Systems** for three consecutive academic years (2001/2, 2002/3, 2003/4). Then I moved on to design and teach graduate level module **CS5272: Embedded Software Design**. I have taught CS5272 for two consecutive years (2004/5, 2005/6) and I am teaching it again this academic year.

In the rest of the article, I will elaborate on my experience in designing and teaching these two modules from computer science perspective.

2. UNDERGRADUATE MODULE

CS2271: Embedded Systems is an essential module for Computer Engineering (CE) program typically taken by the second year undergraduate students. This module is a gentle introduction to the embedded systems technology and is a pre-requisite for the rest of the modules in this area.

The main challenges that I faced in designing this module was the lack of embedded systems courses to use as a reference model and the fear of hardware among the computer science undergraduates.

2.1 Breadth versus Depth

When I started teaching CS2271 back in 2001, there was only one textbook available on embedded systems: Computers as Components by Wayne Wolf [7]. Very few uni-

versities were offering embedded systems modules at that point. Moreover, our computer engineering (CE) curriculum implied that CS2271 should be a foundation module that would set the stage for more advanced modules. Therefore, it was difficult to strike the right balance between breadth and depth in designing the course content for CS2271. After careful considerations, we settled for the following major topics.

- Hardware design with FPGAs
- Processor, Peripherals, and Interfacing
- Programming with ARM
- Real-time systems
- System-level design
- Case study

Real-time systems and system-level design are elaborated later on in CS4275/CS5270 and CS4272, respectively. Embedded software design aspects are covered in more detail in CS5272 (which I will describe later).

Most of the students felt (according to formal and informal student feedback) that they got reasonable exposure to embedded systems through this module. However, 90% of the students did not even have clear idea of what is an embedded system when they started. Therefore, student feedback is perhaps not that effective a measure to judge the suitability of the course content. The downside is that CS2271 is considered as one of the most difficult modules in the CE program. This is mainly due to the integrated hardware-software focus (almost all the other CS modules are completely software based), in particular, for the lab exercises. I will elaborate more on this point in the next subsection.

2.2 Overcoming Steep Learning Curve

The primary challenge in designing an introductory course in embedded systems such as CS2271 is to bridge the lack of background knowledge. Notice that we are offering this module to primarily computer science audience, albeit in computer engineering program. These students are quite comfortable with software programming (in C/Java) but have very little background in digital design. Due to various constraints in our curriculum, the students only go through a quick introduction to digital design and computer organization through a semester-long module before they take up CS2271. I believe this problem is not unique to our school. Any computer science department that wants to offer undergraduate modules with specific focus on embedded systems will face this dilemma.

A quick fix might be to offer CS2271 in senior years. This is an acceptable solution if this is the only module offered in embedded systems. In our case, we would like to develop a curriculum that graduates students with comprehensive background in embedded systems. Therefore, a series of other modules (see Figure 1) has CS2271 as a pre-requisite. Pushing CS2271 to senior years will adversely affect all these other modules. So how do we go about solving this problem?

Fortunately for us, software accounts for 80% of the development cost for embedded systems. Even though our graduates are expected to have solid background in understanding hardware aspects, they may never design an ASIC. Indeed, their proficiency in software programming is the key asset, which should be harnessed with the knowledge and appreciation of the hardware interface. Keeping this in mind, we expose the students to hardware aspects with the help of (a) Field Programmable Gate Arrays (FPGAs) and (b) Handel-C behavioral language for FPGA design [3].

FPGAs are ideal for introducing hardware design aspects due to various reasons. First, it is exciting for students to experiment with their design in real silicon rather than through simulation. This excitement is important to hold their interest in the subject throughout the module. Second, the major aspects in which hardware design differs from software programming — parallelism, communication, clock cycle delay, and resource/area requirements — can all be easily explained with the help of FPGAs. Finally, many FPGA vendors provide access to low-cost development boards through university program making it easy to set up the lab infrastructure.

For the hardware description language, we could not afford to use Verilog or VHDL due to the time constraints. Introducing these languages would have constituted a module by itself. Instead, we decided to use Handel-C — a fully synthesizable, behavioral description language from Celoxica for FPGA design. Handel-C uses much of the conventional syntax of ANSI-C with the addition of inherent parallelism. Therefore, we simply had to introduce the additional constructs (parallelism, communication, clock cycles), which are anyway essential in understanding the tradeoff between implementing the same algorithm in hardware or software. These aspects could be covered in a single two-hour long lecture and the students typically took two weeks to get familiar with the language and the design environment.

The shorter learning curve also implied that we could develop comparatively complex and interesting lab exercises such as designing stack-based processor and simple video games. We used the DK Design Suite from Celoxica [2] to compile and synthesize Handel-C descriptions to RC100 development boards containing 200K-gate Xilinx Spartan II FPGA. The RC100 I/O includes 24bit DAC VGA output and video decoder, two seven-segment LED displays, PS2 mouse and PS2 keyboard.

2.3 Inclusion in CS Curriculum

Currently, we offer CS2271 *exclusively* to the undergraduate students in the computer engineering (CE) program as a compulsory module. In general, it provides a satisfactory learning experience for these students. However, I feel that an introductory module on embedded systems, such as CS2271, as an elective can be quite exciting, interesting, and beneficial for any computer science (CS) graduate due to the following reasons.

- First, embedded systems represent 95-98% of the total market share for computing devices. Therefore, anyone with a CS degree may end up programming for these devices at some point in their career.
- Second, current CS curriculum covers various aspects of computer systems as independent stand alone modules (e.g., computer organization, computer architecture, operating systems, compiler, networking etc.). An embedded systems module provides a unique opportunity to put these concepts together and expose the big picture.

- Moreover, the hands-on nature of the lab exercises ensures that the students will not forget easily what they learnt.
- Last but not the least, the novelty factor of embedded systems labs (playing around with development boards as opposed to all software labs) can provide some fun and excitement in the students' learning experience. Given the declining interest in CS programs worldwide, this factor can be quite important.

3. GRADUATE MODULE

CS5272: Embedded Software Design is a graduate-level elective module offered for the PhD (degree by research) as well as the Masters (degree by course work) students. The Masters students can be either full-time or part-time. PhD students are typically full-time students. In addition, a significant number of undergraduate students also opt for this module. For example, in the 2005/6 offering of the module, 9 out of 42 students were in the undergraduate program.

I designed and offered this module for the first time in the fall of 2004. The motivation was two-fold. First, the embedded systems research group in our school grew considerably to about 20 graduate students in 2004. Most of our research focuses on software aspects of embedded systems starting from high-level models of computation to system level design. An introductory module on the unique aspects of embedded software design would benefit the students embarking on research. Secondly, a significant fraction of our Masters students (both part-time and full-time) were interested in picking up the background knowledge about this exciting area. 45 students registered for CS5272 in its first offering indicating that there was indeed a need for such a module.

As opposed to CS2271, which is an introductory module on embedded systems, CS5272 is an advanced module focused on only embedded software aspects. The goal is to develop a comprehensive understanding of the unique design issues in building highly optimized and customized software for different hardware platforms, satisfying design constraints related to size, power, and performance. We expect the students to have solid background in general software engineering. Therefore, the focus is more on the distinguishing characteristics of embedded systems that takes software development beyond traditional programming approaches. More concretely, the major topics covered in this module are

- Embedded software development with ARM
- Optimizations to meet area constraints
- Optimizations to meet power constraints
- Compilers for hardware acceleration
- Case study

In an attempt to re-use the lab infrastructure as much as possible, we use the same ARM-based development boards for both undergraduate and graduate program. However, while the undergraduate module focuses more on basic programming for embedded systems with ARM (simple device driver programming), the graduate level module is much more intensive. The first lab serves as a warm-up exercise to familiarize the students with the basics of embedded software development. The rest of the lab exercises go in sync with the topics covered such as area and power optimizations. These lab exercises are quite popular as the students can apply first-hand the knowledge they acquire in the lectures.

The major difficulty is designing this module stems from the diverse background of the student population. In the following, I elaborate on these issues.

3.1 Research versus Industry

As mentioned before, CS5272 caters to research students, course work students as well as part-time students. This makes is quite challenging to decide on the course content. On the one hand, the part-time students coming from the industry and the course work students planning to join the industry are more interested in the practical (lab) aspects of the module. On the other hand, research students are presumably interested in areas that are still in the realm of academic research so as to get an exposure to possible future thesis topics. Catering to both these group of students requires a delicate balancing act.

However, research students benefit quite a lot from hands-on programming for embedded systems that develop awareness about the low-level systems issues. At the same time, introducing state-of-the-art research ideas to the future industry folks is a positive means towards technology transfer. Therefore, I decided to have a mix of both practical and research issues covered in the module. The end of the semester student feedback suggested that all the students were quite happy with the syllabus.

3.2 CS versus EE background

The advantage of designing embedded systems courses for junior undergraduate students is that you expect quite a homogeneous student body with roughly the same background knowledge. But given the varied background of graduate students, it is extremely difficult, if not impossible, to come up with an appropriate pre-requisite for a graduate-level module. We expect that the students will have some background in computer architecture and compiler. However, the inter-disciplinary nature of embedded systems implies that both computer science and electrical engineering graduate enroll for the module.

As mentioned before, compared to the undergraduate-level module, the focus of the graduate-level module is more on embedded software. Therefore, we introduce a number of compiler optimization techniques. Unfortunately, the students with electrical engineering background have little or no knowledge about the compiler issues. So we are faced with the opposite problem of what we faced in CS2271 while teaching hardware synthesis. I provide supplementary material about the basic concepts of compiler design to bring these students up to speed. But there is no denying of the fact that it somewhat slows down the schedule. I am not aware of any satisfactory strategy to resolve this issue.

3.3 Tradeoff between Projects and Exercises

Ideally, in a graduate level module one would prefer to have some projects that span across the entire semester. In the first offering of the module in 2004, I opted for projects. Keeping in mind the diverse interests of the students, the

suggested project ideas spanned from implementation of complex embedded applications to research-oriented projects. Given the time constraints, research-oriented projects typically involved incremental modification and evaluation of existing research ideas. Unfortunately, the projects were not as successful as I expected. The main reasons behind this are (a) the learning curve was too steep for the students to complete meaningful projects, and (b) grading was difficult given the wide disparity between the nature of the different projects (practical versus research-oriented).

Based on this experience, in the next academic year, I decided to include lab exercises along with a term paper. Lab exercises were designed to be more difficult compared to the previous year. As an example, the students developed a simple embedded operating system from scratch with real-time scheduling option. For the term paper, the students surveyed a topic in detail that was not covered in class. Overall, this model worked better compared to the project-based model. However, the ideal option will be to offer a follow-up module to CS5272 that is completely project based.

3.4 Labs on Laptops

An important practical issue that I did not consider before offering CS5272 is the lab hours. For CS5272, I kept the lab open 24 hours, 7 days a week. Unfortunately, this was not a satisfactory option for the part-time students. Even though Singapore is a small country, our students need to commute on an average 30 minutes to one hour each way. Long commute over the weekend to come to the lab was not feasible for most part-time students. Most of them preferred to have some form of evaluation version of the development software installed on their laptop and complete most of the lab exercises at home/office. Even though I used ARM ADS [1] development kit for the undergraduate module, I switched to GNU development tools [4] for CS5272. The missing piece was a virtual development platform that simulates all the peripherals of the real development platform. Then the students can test/debug their program on the virtual platform and only need the real hardware platform in the final stage of the design. From the next academic year, I am planning to use the new RealView Microcontroller Development Kit for ARM-Powered microcontrollers from Keil [5]. Keil offers a free evaluation version for professors and students. But more importantly it offers virtual platforms for a range of ARM-based development boards.

4. CONCLUSIONS

This article summarizes my observations in designing and teaching embedded systems modules to both undergraduate and graduate students over the past five years. Establishing the computer engineering curriculum with a focus on embedded systems was a struggle when we started back in 2001. But it was a worthwhile experience. We have produced close to 300 graduates armed with the appreciation and knowledge of unique computing devices that work at the boundary of hardware and software. This teaching focus was instrumental in helping us to establish a strong research group in embedded systems in School of Computing at National University of Singapore. We currently have 5 faculty members, 2 post-doctoral fellows, and 26 graduate students in the embedded systems research group and a total of about Singapore 2.5 million dollars in current research funding.

Dedicated embedded systems modules are gaining increasing importance in both computer science and electrical engineering curriculum. However, the inter-disciplinary nature of embedded systems, which spans across hardware and software domain, poses unique challenges in designing the curriculum. Fortunately, these problems are not insurmountable. It is essential to exploit the technological advances (such as C-to-hardware synthesis tools) that raise the design complexities to the higher abstraction layers. This allows us to build on the strength of the students' background rather than starting from scratch.

It is important, though, to mobilize an effort in standardizing embedded systems curriculum across universities. These include textbooks, development platforms as well as simulators. As an example, educators worldwide benefit quite a lot from the classic textbooks by David Patterson and John Hennessy as well as SPIM and SimpleScalar simulators while teaching computer organization and computer architecture courses. An equivalent of this effort in embedded systems would make the life much easier for anybody embarking on an effort to design a new module.

Finally, lab exercises constitute an essential component of any embedded systems module. In any computer science department, setting up the lab infrastructure is quite involved as we need to acquire hardware development platforms that most system administrators are not familiar with. It is important to decide on a common platform infrastructure for all the modules under embedded systems area in order to exploit the knowledge base and amortize the cost of hardware acquisition. A common platform also helps the students to have a smooth transition from one module to another.

5. REFERENCES

[1] ARM. Arm developer suite. `http://www.arm.com/products/DevTools/ADS.html`.

[2] Celoxica. Dk design suite. `http://www.celoxica.com/products/dk/default.asp`.

[3] Celoxica. Handel-c: C-based design and behavioral synthesis. `http://www.celoxica.com/technology/c_design/handel-c.asp`.

[4] CodeSourcery. Gnu toolchain for arm processors. `http://www.codesourcery.com/gnu_toolchains/arm/`.

[5] KEIL: An ARM Company. Arm evaluation software. `http://www.keil.com/demo/`.

[6] National University of Singapore School of Computing. Bachelor of computing in computer engineering, 2006. `http://www.comp.nus.edu.sg/~cmcurric/AY2006_7/soc67_CEprogram.pdf`.

[7] Wayne Wolf. *Computers as Components: Principles of Embedded Computing System Design.* Morgan Kaufmann Publishers, 2001.

Addressing Embedded Programming Needs within an ECE Curriculum

Kenneth G. Ricks
The University of Alabama
Electrical and Computer Engineering
Tuscaloosa, Alabama 35487-0286
(205)-348-9777
kricks@eng.ua.edu

David J. Jackson
The University of Alabama
Electrical and Computer Engineering
Tuscaloosa, Alabama 35487-0286
(205)-348-2919
jjackson@eng.ua.edu

William A. Stapleton
The University of Alabama
Electrical and Computer Engineering
Tuscaloosa, Alabama 35487-0286
(205)-348-1436
wstapleton@eng.ua.edu

ABSTRACT
In this paper, the typical electrical and computer engineering (ECE) curriculum is examined to determine its effectiveness at presenting embedded programming skills. The software concepts and programming techniques necessary for embedded systems are somewhat different than those seen in other engineering domains. Thus, it makes sense to specifically address embedded programming needs within the formal programming education ECE students receive. Several topical areas of concern are identified, and two possible ways to incorporate these areas into an ECE curriculum are presented. The experiences gained within the ECE curriculum at The University of Alabama are presented and are used to develop recommendations for incorporating these topics into typical ECE curricula.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *curriculum, computer science education.*

General Terms
Languages

Keywords
Embedded systems education, embedded systems programming, C programming language, engineering curriculum

1. INTRODUCTION
The need for embedded systems engineers is well documented and supported by the recent attention the field has gained within academia. Embedded systems education presents some interesting problems however. One of the most important problems is the breadth problem [6], i.e. how to feasibly incorporate the broad spectrum of embedded systems topics into the curriculum. To gain a perspective on exactly how broad this scope is, one can look at the IEEE/ACM model computer engineering curriculum report describing recommended topical coverage for embedded systems [8]. In this report, there are 11 knowledge areas covering 59 different topics assessing 34 learning outcomes.

In this paper, we address a small part of the breadth problem, specifically focusing on the programming skills needed by embedded systems engineers. Since programming is already a fundamental component of nearly every electrical and computer engineering (ECE) curriculum, one might think students are well exposed to the necessary skills in this area. However, this paper will address concerns that the general programming skills being taught in typical ECE curricula are not addressing all embedded programming skills needed.

Embedded software programming is a critical aspect of embedded systems education. In [5], it is estimated that the amount of embedded software created doubles every ten months, and will reach 90% of all software being written by the year 2010. To address this demand, embedded software programming skills must be incorporated into embedded systems education curricula immediately.

The rest of this paper is organized as follows. In Section 2, we describe a typical ECE approach to programming and point out several pitfalls to such an approach. Section 3 describes several need areas for embedded programming skills not usually addressed in this typical approach. Section 4 presents some possible solutions for correcting the problem, and Section 5 presents the conclusions.

2. TYPICAL ECE APPROACH
A typical ECE curriculum includes a two-level approach to teaching programming skills. First, one or more introductory general programming courses are typically offered early in the curriculum. The goals of these courses vary considerably between programs, but usually a high-level language (HLL) is presented. The motivation for offering this material at an early point in the program is to provide engineering students a tool with which to solve problems as they progress through the curriculum.

Later in the curriculum, students are exposed to assembly language usually in the context of a microprocessors or microcontrollers course. At this time, students write assembly language programs to control and interface with various hardware devices thus acquiring low-level hardware interfacing skills. In many cases, students are shown how a HLL program is decomposed into the assembly and machine language equivalents, thereby tying the two programming levels together.

By and large this two-level approach to programming in ECE curricula is effective and provides students with many useful skills. However, from an embedded programming perspective, there are pitfalls in this approach. First, general programming courses often incorporate many different concepts that do not map directly to the skill set needed by embedded systems engineers. This is especially the case for general programming courses supporting students from many different engineering disciplines. It is quite common for these courses to serve a multipurpose role within the curriculum covering all sorts of topics including

teaching programming language syntax [1, 3, 12], problem solving [4, 11], teaming [4], communication skills [4], program design [4], algorithm design, and object-oriented (OO) programming techniques. While these concepts are valuable, they do not replace the fundamental programming skills needed by embedded systems engineers, and they tend to dilute the programming aspects of the course from the embedded systems perspective. There are other effective techniques to teach problem solving using the computer that can be incorporated into other parts of the curriculum [1, 4]. Also, OO approaches are not applicable to all engineering disciplines including some embedded applications [11]. In some cases, the general programming courses are given an engineering flavor by focusing on engineering applications and problems [11]. This makes the applications more appropriate to embedded programming but does not necessarily improve the programming fundamentals presented.

The assembly language courses also have drawbacks. Since these languages are tied very closely to the hardware, class time is required to address syntax, basic assembly software design, particular hardware interfacing issues for specific devices, system integration, debugging, and concepts such as pulse width modulation and analog-to-digital conversion. These are skills and concepts needed in many embedded applications, but this is a large amount of information to present in one course and tends to overwhelm the basic programming skills aspects of the course. Also, assembly language is not the most popular language used for embedded applications. In 2000, 80% of all embedded systems applications were written in a HLL, specifically the C programming language [16], and this number increases as more capable tools introduce more abstraction.

Embedded systems engineers are migrating to a higher level of abstraction, using tools and development environments to handle the lower level details, such as the hardware/software interfacing aspects of embedded systems. But, the typical ECE curriculum depends on general purpose programming courses and low-level assembly language courses to present the programming skills required of embedded systems engineers. This approach to programming does not prepare students to address low-level details from higher levels of abstraction. It is not uncommon for students in such curricula to encounter problems in upper-level ECE courses, like a senior-level Embedded Systems course or a Capstone Design course, where they are expected to use higher levels of abstraction to address the embedded system. A higher level view of embedded programming is needed in the curriculum to address this concern.

3. EMBEDDED PROGRAMMING NEEDS

This section describes four specific areas where ECE students in typical curricula are likely to encounter problems with high-level programming skills needed for embedded applications. These areas, summarized in Table 1, are derived from research, personal communications with educational and industry professionals, and personal observations made within the ECE curriculum at The University of Alabama (UA). UA follows the traditional two-layer approach to programming, as described in the previous section, within its ECE curriculum.

Table 1. Summary of areas of concern for embedded programming skills.

Areas of Concern for Embedded Programming Skills
Choice of HLL
Peripheral Interfacing Using Registers
Program Structure
Resource Constraints

3.1 Choice of HLL

Since we are addressing programming skills needed by embedded systems engineers, let's begin with the most basic idea, the choice of HLL. While there are compelling arguments for the use of many different programming languages within education, at this time there is little argument over the need for embedded systems engineers to know the C programming language. As previously mentioned, C is the language of choice for a large majority of embedded applications. It is our experience that students without a working knowledge of C are at a significant disadvantage when competing for jobs related to embedded systems. While having experience with a different HLL certainly flattens the learning curve for those desiring to learn C, there is no reason that C should not be the language of choice for ECE students.

To expand on this idea a bit more, it is our belief that students should begin with ANSI C. This provides a portable programming foundation upon which additional skills can be built. Introductory programming courses that mix object-oriented concepts and other concepts with ANSI C tend to dilute the basic understanding of C and undermine its portability across hardware platforms. Portability is an important concept to embedded programmers because it promotes code reuse thereby shortening time-to-market and impacting design decisions. Software development tools are ranked as the number one factor for microprocessor choice for embedded systems [17], and portability is a key evaluation criterion for these tools and the code they produce. In the following sections, the C programming language is used for all examples.

3.2 Peripheral Interfacing Using Registers

Embedded systems often contain a set of memory-mapped locations, called device registers, used for communication with the device. The ability to access and manipulate registers is critical for embedded systems. Device registers are often located at specific locations in memory. Variables in the HLL must be declared and initialized to represent data located at those specific locations. One cannot declare a variable in ANSI C so that it resides at a specified memory address [14]. Thus, pointers are required, and pointer arithmetic becomes critical to maintain access to the registers [2, 7, 9, 10]. An understanding of the differences between memory-mapped registers and port-based input/output (I/O) registers is needed since the type of registers determines how they can be accessed from the HLL. For example, memory-mapped registers can be treated like any other memory location and read and written using standard assignment and dereferencing operators as shown in Figure 1. But, port-based I/O registers may require special instructions for reading and writing because the underlying processor instruction set differs for accessing these types of registers. Also, bitwise manipulation of register contents is required since in many cases

registers contain unrelated bitfields. Thus, it is essential to present the bitwise operators in C, including the bitwise-AND operator "&", bitwise-OR operator "|", logical-NOT operator "~", and the bitfield structure operator ":" [2, 7, 9, 10].

```
int * CSR_ptr = 0xFFAA;
*CSR_ptr = 1;
```

Figure 1. Example C code that accesses a memory-mapped Control/Status register and assigns it a value.

Another important aspect of registers is that in many cases, register values change outside the scope of the program. For example, a register might contain a bitfield used as a status indicator. The user program may initialize this register to a certain value, and the device writes a new value into this bitfield to represent a change in the device's status. If appropriate variable type qualifiers are not used, compiler optimizations can result in erroneous code. For example, consider the unsafe code fragment in Figure 2A. This fragment initializes the CSR device register and then later reinitializes the same register using the same value. The compiler might conclude the second write operation to this location is redundant since the same value is being written twice and no other assignment operation separates the two initialization assignments. Thus, the compiler removes the second write unaware that an external device modified the CSR register contents between the writes. To prevent this, the type qualifier "volatile" can be used, shown in Figure 2B, to inform the compiler that the variable associated with this register may change outside the scope of the program [2, 9, 10, 14]. The resulting code segment is safe from these types of erroneous optimizations.

```
int * CSR_ptr = 0xFFAA;
*CSR_ptr = 1;      /* initialize register contents */
... (During this part of the program, the device's
    status changes and the register value is
    overwritten by the device hardware.)
*CSR_ptr = 1;      /* re-initialize register contents */
```

A. Unsafe code fragment subject to erroneous compiler optimization.

```
volatile int * CSR_ptr = 0xFFAA;
*CSR_ptr = 1;      /* initialize register contents */
... (During this part of the program, the device's
    status changes and the register value is
    overwritten by the device hardware.)
*CSR_ptr = 1;      /* re-initialize register contents */
```

B. Safe code fragment not subject to erroneous compiler optimization.

Figure 2. Example of the importance of variable type qualifiers.

Students can succeed in the two-layer programming paradigm without these register manipulation skills. Without an emphasis on embedded programming, a general-purpose C programming class has little motivation to present pointer arithmetic outside the context of particular data structures. Similarly, pointer assignment, variable type qualifiers, and bitwise operators are topics that are often not covered in depth. Instead, variables represent data needed by the program and are stored "somewhere in memory." The translation tools including the compiler, assembler, linker, and loader abstract the details of exactly where in memory the variables reside. The programmer must use an explicit prefix, "&", to determine this information, if it is needed.

The information presented within an assembly language course does not fill in the gaps left by the general-purpose programming course. The term "register" takes on a different meaning within the context of an assembly language course. In this context, the register set of the processor executing the code is usually what is meant and care must be taken to prevent confusion between the processor registers and the device registers. Also, students are faced with using obscure addressing modes to access the device registers instead of using pointers which is needed at higher-levels of abstraction. While it is true that bitwise operators can be more commonly seen in assembly language, this does not give students the skill to perform these operations in a HLL.

3.3 Program Structure

While basic program structures do not differ significantly between general-purpose programming and embedded programming, the motivations for using certain structures need to be understood by embedded software developers. For example, students need to understand that using subroutines and a modular programming approach may offer "divide and conquer" benefits such that a large problem may be debugged in smaller, more manageable parts. Similarly, a modular, subroutine-based approach may lead to reusable code that reduces "reinventing the wheel" over the long term and shortens time-to-market. On the other hand, utilizing in-line code eliminates the overhead associated with calling subroutines. This can reduce execution time requirements for real-time applications, but may require significantly greater storage requirements since otherwise reused code must be replicated in-line.

Other structural nuances such as global vs. local variables, subroutine parameter forms, and the impacts of these concepts on embedded system performance are critical. Variables that are global in scope may reduce storage requirements and subroutine overhead as they are only stored once and do not need to be copied to the subroutine's context. Unfortunately, global variables may be problematic in a system where multiple routines may all access the same data as additional mechanisms must be put into place to assure data coherency [9, 10]. Similarly, passing data to subroutines by reference rather than by value may reduce the amount of data that must be copied as subroutine overhead when large data structures are involved.

In most cases, students are presented many of the HLL implementation details associated with these concepts in a general programming course. The observed shortcomings in student ability involve understanding the motivations for choosing one alternative over another. For example, most students have seen subroutines in a HLL. The problem is getting students to

understand their value and to choose a modular programming approach when appropriate. The goal must be to educate the students to make the best choice for the given application instead of defaulting to what is considered to be easier for the programmer.

In some cases, high-level programming tools abstract structural details of software development that embedded programmers need to understand. For example, embedded software developers in a team-based development environment often use a modular programming approach. The ability to compile HLL code to object code and to link with existing object code to create an executable are basic concepts often encountered in such development environments. But, students in general programming courses typically use high-level tools that abstract the translation process and automatically build executables. Thus, students are seldom aware of the different aspects of program translation required to support different program structures.

3.4 Resource Restrictions

Because embedded systems typically are more resource-limited than general-purpose computing systems, the need to manage system resources carefully is much more urgent for embedded systems than is typically taught in general-purpose programming. This is particularly true when the general-purpose computing platforms incorporate object-oriented programs where the programming paradigm purposefully hides the resource implementation details from the programmer. Large memory-footprint objects such as linked lists or arrays of structures that are easily accommodated on a general-purpose PC may not be possible on an embedded system with limited memory. Even a factor as simple as choosing the most efficient data type can be very important to an embedded system with limited resources. For example, reducing Boolean variables to single bits from "int" size can greatly reduce the memory footprint of a program but require the use of bit-manipulation operators that may not be considered in a general-purpose programming class. Alternatively, the bitfield structure operator ":" may be used to create variables of various sizes that may be treated functionally much like integers. But, using this operator requires the understanding of how the compiler allocates bits, either starting from the most-significant or the least-significant bit position [2, 9, 10, 14]. Such details are rarely addressed in general-purpose programming courses.

Memory is not the only resource with limitations that must be considered for embedded systems that is generally ignored in general-purpose computing. Time is also a limited commodity. Many embedded systems programs must operate under significant time constraints. General-purpose programming rarely considers hard or soft real-time constraints. Consequently, coding efficiency is typically not given as much emphasis as is required for embedded systems.

Power is another interrelated limited commodity. Many embedded systems operate from batteries or other limited power sources and must consequently operate as efficiently as possible. Limiting power usage and excess heat production often requires embedded system microprocessor clock rates to be limited. This can increase the difficulty of meeting timing constraints.

Embedded systems programmers must be able to balance the tradeoffs involved with execution speed, memory space, and available power. Pre-calculated values in tables can reduce the execution time and power used at the expense of larger, more expensive memories. Clock rates may be reduced to save power at the expense of performance and responsiveness. Smaller, cheaper memories may be used if coding efficiency is increased.

4. CURRICULA REFORM

Now that several need areas have been identified, we must address the problem of adding the embedded programming specific need areas to the typical ECE curriculum. We address two possibilities of how this can be done. First, the embedded programming topics can be integrated into the existing programming courses. Second, additional courses designed specifically to present embedded programming concepts can be added to the curriculum. Each of these options is discussed in the following sections along with specific results from the UA efforts.

4.1 Integration into Existing Curricula

Integration of embedded programming concepts into an existing curriculum begins with the general-purpose programming course. If the HLL presented is C, then the first desired skill is automatically addressed. Integrating the other embedded programming concepts into an existing C programming course does not require a complete re-design of the course. Many of the necessary C constructs are already being presented. Assignments can be modified to address register interfacing, program structure alternatives, or resource limitations. Hardware platforms are usually required for embedded software development. These platforms are typically not used for general-purpose programming courses. In that case, embedded hardware can be simulated using artificial memory constraints, timing deadlines, and register addresses.

In cases where the introductory course is organized as one lecture section with smaller laboratory sections, one laboratory section can be dedicated as an embedded programming section for ECE students. In this section, assignments can be tailored for embedded applications and supplemental material can be presented to address embedded programming concepts. Also, at this level it is more plausible to integrate embedded hardware into the course. Such an organization introduces the embedded programming concepts for ECE students while not impacting students from other disciplines and not requiring additional courses be added to the curriculum.

The embedded programming concepts introduced in the general-purpose programming course can be revisited and reinforced in the microprocessors/microcontrollers course where assembly language is presented. In this course, a higher level of abstraction must be incorporated and hardware manipulation from the HLL must be included. Based on the typical content of these courses, this is a natural place within the curriculum to present embedded programming skills. Care must be taken, however, not to substitute HLL programming skills for the low-level skills typically presented in these courses. While it is true that more abstraction is being used in embedded systems, it is still important that students learn an assembly language and the programming skills associated with it. These skills are critical as indicated by their inclusion in the IEEE/ACM model curriculum and are not presented elsewhere in the typical curriculum [8].

4.2 Adding Courses to the ECE Curriculum

If integration into existing programming courses is not possible, then another option is to address these concepts in dedicated coursework. The one main positive to this approach is that existing courses are not affected and widespread coordination within the curriculum is not required to achieve the educational goals. Of course, there are difficulties associated with adding hours to any curriculum if courses cannot be identified for replacement.

For ECE curricula that rely on other departments to teach the general-purpose programming courses, one possibility is to replace these courses with courses that include embedded programming skills and are designed specifically for ECE students. The obvious merits of this include having a course taught by ECE faculty designed for ECE students. The drawbacks include additional teaching loads on ECE faculty and what appears to administration to be redundancy and waste in the curriculum.

Another possibility is to add a higher-level course to the curriculum such as an Embedded Systems course. This course would be taken after the assembly language course and would focus specifically on all aspects of embedded systems. Presenting embedded programming skills in such a course is possible, but does it make sense? Addressing basic programming skills is not usually part of the syllabus for a junior-level or senior-level Embedded Systems course. These courses are loaded with the more advanced knowledge areas, topics, and learning outcomes discussed in the IEEE/ACM model computer engineering curriculum for embedded systems [8]. For example, real-time concepts, interprocess communication, effects of caching on program performance, and scheduling are software concepts that must be covered, not to mention the hardware, design, and interfacing aspects of embedded systems. Taking time to review basic programming skills and teach proper program structure are topics that clearly do not belong at this level.

4.3 The UA Experience

The UA ECE curriculum uses a typical two-level approach to programming. The prerequisite relationships among the programming courses and the core computer engineering courses are shown in Figure 3. Introductory programming skills and problem solving skills are taught in CS 114 and CS 116, where the C programming language, although not strictly ANSI C, is used as the basis. Object-oriented programming is presented in CS 124 by specifically introducing students to C++. These programming courses are taught by the Department of Computer Science within the UA College of Engineering and serve as prerequisites to ECE 380 and ECE 383. ECE 380 is a typical digital logic course. The ECE 383 Microcomputers course covers the traditional topics associated with such a course including assembly language programming and peripheral interfacing. ECE 383 serves as a prerequisite to several traditional computer engineering courses including ECE 480/481, ECE 484, and ECE 486/487. ECE 480/481 is a digital systems design course using VHDL. ECE 484 is a typical microprocessor architecture course. ECE 486/487 is a senior-level embedded systems course recently added to the curriculum. The course sequence culminates with ECE 494 Capstone Design where students must design and implement a complete project within a one-semester timeframe. All the core computer engineering courses within the UA ECE curriculum have an integrated embedded systems component, creating a curriculum with an overall focus on embedded systems [13, 15].

As part of this effort, the authors collected assessment data from UA ECE students at the junior level to specifically evaluate overall programming skills and embedded programming skills. Figure 4 shows some representative questions asked in the assessment process and a summary of the results from each question. The results represent the percentage of students who provided a reasonably correct answer to each question.

1) "Describe the difference between the ANSI C bitwise operators (e.g. "&") and the logical-test operators (e.g. "&&"). When is each appropriate?"
   Results = 19%

2) "Describe the difference between the ANSI C bitwise-AND operator "&" used like "X = Y & 0x1f" and the ANSI C address operator "&" used like "ptr = &var;". How are these two uses for the same character ("&") distinguished?
   Results = 22%

3) "In ANSI C, what is a pointer? How is a pointer specified? How is a pointer used? How is the pointer value related to the physical memory system or computer?"
   Results = 31%

4) "How are parameters passed between C subroutines? (i.e. in what format and in what order)"
   Results = 9.4%

5) What is the difference between global and local variables in ANSI C? How are each type specified? What are the advantages of each type?
   Results = 16%

Figure 4. Representative assessment questions and the corresponding results.

The assessment data collected corroborate several observations made by the authors related to students' performance within the UA ECE program and generalized in earlier sections of this paper. First, ECE students show a general lack of overall programming skills (questions 3, 4, 5). Second, there seems to be little retention by upperclassmen of the HLL programming concepts presented early in the curriculum. Third, students demonstrate a complete lack of understanding of the HLL constructs necessary for embedded systems programming (questions 1, 2, and 3). This is due to lack of retention and the material not being covered in the introductory programming courses.

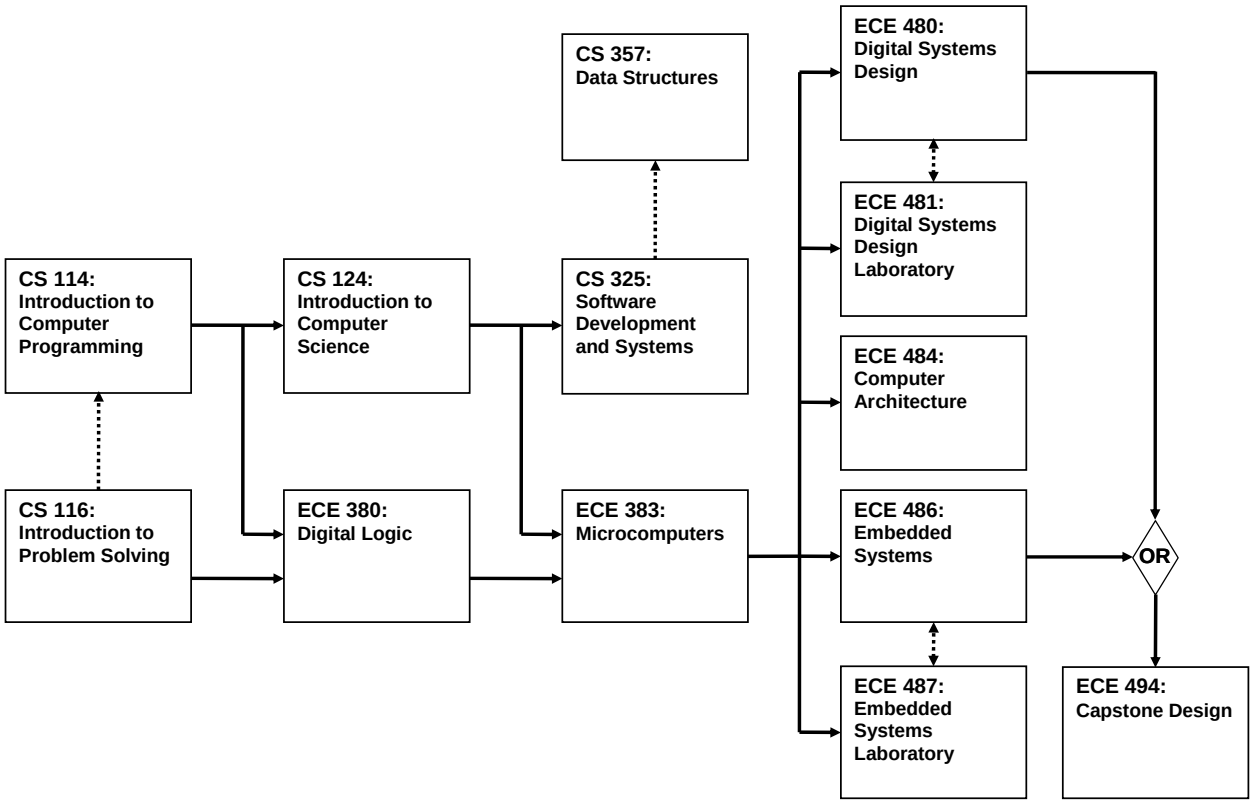


Figure 3. Prerequisite relationships among programming courses and core computer engineering courses within the UA ECE curriculum.

To integrate the desired embedded programming skills into the UA ECE curriculum, the senior-level ECE 486/487 Embedded Systems course is used. Supplemental material is presented in this course to fill in the gaps resulting from the typical two-layer programming approach. To avoid the loss of significant lecture time within the Embedded Systems course, informal presentations of embedded programming skills are incorporated into the laboratory portion of the course. The specific embedded programming skills presented in this course closely follow the need areas presented in Table 1. Specifically, the strict use of ANSI C is required as the programming language. Device register interfacing using ANSI C constructs is required including the use of pointers, bitwise operators, and the address operator ("&"). Program structure is presented and students are required to write modular code using ANSI C subroutines. This program structure includes emphasis on passing parameters by reference and by values as well as the advantages and disadvantages of global and local variables. Finally, resource constraints are presented by analyzing variable declarations and the memory requirements associated with those declarations.

There are several lessons learned from the UA embedded systems programming approach. First, based upon the assessment data presented earlier showing the limited programming skills of our students, any exposure to software development within the curriculum is important. Thus, adding a significant software development component to the ECE 486/487 course is viewed as a positive. Second, incorporating this material into the curriculum tends to break down the stereotypes associated with "software" engineers and "hardware" engineers. Although these distinctions exist in other disciplines, in the embedded systems world, engineers must be proficient in both the hardware and the software domains to be successful. Third, it is difficult to integrate introductory programming concepts into a senior-level course. Although attempts were made to present this material in the laboratory portion of the course to reduce lecture time dedicated to these concepts, the students cannot be expected to conquer more advanced concepts when such fundamental skills are lacking. For example, it is difficult to introduce real-time scheduling concepts to students that do not understand modular program structure and multitasking. Finally, programming skills are more important in an embedded systems course than the hardware platform used. Hardware platforms are plentiful and in many cases, design of embedded systems courses centers around the selection of an appropriate hardware platform. However, it is the programming skills that should take priority. As long as the hardware platform provides access to basic peripherals, device registers, and memory, the focus of the equipment should be the software development environment. Students without sound embedded programming skills will struggle regardless of the platform used.

Based upon the experiences at UA, the following recommendations can be made. First, presenting embedded programming concepts in an introductory programming course does not appear to be a good solution. The retention problem would persist when students are asked to recall the information for the first time years later. Also, early in the curriculum,

students do not have an appreciation for the skill sets needed for embedded applications. Thus, presentation of these concepts later in the curriculum is recommended. A senior-level Embedded Systems course was used at UA for this purpose and the results were mostly positive. But, not every ECE curriculum has such a course, and there are problems associated with adding courses to a curriculum. Also, there is no debating the fact that introductory programming skills do not really belong in senior-level courses. Incorporating these concepts into ECE 486/487 replaces more advanced embedded systems topics that need to be included. Therefore, it is recommended that incorporation of embedded programming concepts should first occur in the microprocessors or microcontrollers course where assembly programming is presented. This course appears to be the best fit for the embedded programming topics. The interfacing aspects already present in such a course make it easy to include interfacing to device registers. Also, this course serves to introduce assembly programming. So, there is already a programming component to which embedded programming concepts can be attached. Presenting C and linking it to the underlying assembly is an especially attractive possibility in such a course.

5. CONCLUSIONS

As embedded systems continue to increase in number and complexity, ECE curricula must address the embedded programming skills needed by their graduates. The typical ECE programming experience does not address these embedded programming concepts, often leaving graduates to acquire these skills on-the-job. This paper presents four areas of concern regarding embedded programming concepts not addressed in typical ECE curricula. These areas include a lack of C programming skills, inability to interface to registers, incomplete knowledge of appropriate program structures, and the inability to address resource constraints common in embedded systems.

To address these problems, ECE curricula must incorporate these concepts into existing programming courses or introduce new dedicated courses to address these specific topics. Each of these options presents its own set of concerns. The ECE curriculum at UA introduced these topics in a dedicated embedded systems course introduced at the senior level. The lessons learned from this experience indicate that these topics belong in the assembly language programming course found in typical ECE curricula. UA is currently investigating implementing this change in its curriculum and developing a plan to assess its effectiveness.

6. REFERENCES

[1] Bjedov, G., Andersen, P.K., "Should Freshman Engineering Students Be Taught a Programming language?", Proceedings of the 26th Annual Frontiers in Education Conference, Volume 1, Nov. 6-9, 1996, pp. 90-92.

[2] Bramer, B., Bramer, S., *C for Engineers, 2nd Edition*, John Wiley & Sons, New York, New York, 1997.

[3] Budny, D., Lund, L., Vipperman, J., Patzer, J.L.I.I.I., "Four Steps to Teaching C Programming", Proceedings of the 32nd Annual Frontiers in Education Conference, Volume 2, November 6-9, 2002, pp. F1G-18 - F1G-22.

[4] Davenport, D., "Experience Using a Project-Based Approach in an Introductory Programming Course", *IEEE Transactions on Education*, Volume 43, Issue 4, November 2000, pp. 443 – 448.

[5] Ganssle, J., "The Demise of the Embedded Generalist", Embedded.com, Available: http://www.embedded.com/showArticle.jhtml?articleID=51202213, November 2, 2004.

[6] Haberman, B., Trakhtenbrot, M., "An Undergraduate Program in Embedded Systems Engineering", Proceedings of the 18th Conference of Software Engineering Education and Training (CSEET'05), April 18-20, 2005, pp. 103-110.

[7] Harbison III, S. P., Steele Jr., G. L., *C: A Reference Manual, 5th Edition*, Prentice Hall, Upper Saddle River, New Jersey, 2002.

[8] Joint Task Force on Computer Engineering Curricula, IEEE Computer Society, Association for Computing Machinery, "Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering", December 12, 2004, pp. A.43 – A.45, Available: http://www.computer.org/education/cc2001/CCCE-FinalReport-2004Dec12-Final.pdf.

[9] Kernighan, B. W., Ritchie, D. M., *The C Programming Language, 2nd Edition*, Prentice Hall, 1988.

[10] Kochan, S. G., *Programming in ANSI C*, Prentice Hall, Indianapolis, Indiana, 1994.

[11] Nagurney, L.S., "Teaching Introductory Programming for Engineers in an Interactive Classroom", Proceedings of the 31st Annual Frontiers in Education Conference, Volume 3, October 10-13, 2001, Reno, Nevada, pp. S2C - 1-5.

[12] Parrish, A., Borie, R., Cordes, D., Dixon, B., Jackson, J., Pimmel, R., "An Integrated Introductory Course for Computer Science and Engineering", Proceedings of the 29th Annual Frontiers in Education Conference, Volume 1, November 10-13, 1999, pp. 11A3/12 - 11A3/17.

[13] Ricks, K. G., Stapleton, W. A., Jackson, D. J., "An Embedded Systems Course and Course Sequence", in Proc. of the 2005 Workshop on Computer Architecture Education (WCAE), Madison Wisconsin, June 5, 2005, pp. 46-52.

[14] Saks, D., "Representing and Manipulating Hardware in Standard C and C++", Embedded Systems Conference, Session ESC-243, San Francisco, California, March 6-10, 2005, Available: http://newit.gsu.unibel.by/resources/conferences%5Cesc_2004%5CSan_Francisco%5Cesc_243.pdf.

[15] Stapleton, W. A., Ricks, K. G., Jackson, D. J., "Implementation of an Embedded Systems Curriculum", in Proc. of the 20th International Conference on Computers and Their Applications (CATA'05), New Orleans, Louisiana, March 16-18, 2005, pp. 302-307.

[16] 1999/2000 TRON Association Survey, Available: http://www.ncsu.edu/wcae/ISCA2005/submissions/ricks.ppt.

[17] Turley, J., "Survey says: Software Tools More Important Than Chips", Embedded Systems Design, Available: http://www.embedded.com/showArticle.jhtml?articleID=160700620, April 11, 2005.

Bringing Embedded Software Closer to Computer Science Students

Jogesh K. Muppala
Dept. of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clearwater Bay, Kowloon, Hong Kong
Tel: +852 2358 6978
muppala@cse.ust.hk

ABSTRACT
Computer Science students have often shied away from the field of embedded systems owing to their perception of this area as "hardware" oriented, not without reason. But recent trends in embedded systems, with the growing importance of the software component, has brought about new opportunities for computer science students to participate and contribute to embedded system development. In this paper we present our views and experience gained by teaching Computer Science students, on how we can bring embedded systems closer to them, to provide them the opportunity to fully participate in this growing field.

Categories and Subject Descriptors
K.3.2 [Computer and Information Science Education]: *computer science education, embedded software education.*

General Terms
Computer Science Education, Curriculum.

Keywords
Embedded software, embedded systems education.

1. INTRODUCTION
Embedded systems design as a discipline has long been pursued in the industry in various application domains including avionics, aerospace, automobile, and industrial control etc. As pointed out by Lee [11], this area has largely been ignored by academics because it has not thrown up sufficient complex research challenges. However, all this has changed recently with increasing interest being paid by the academic community towards embedded systems education, as evidenced by the papers published in the recent special issue of ACM Transactions on Embedded Computing Systems dedicated to embedded systems education [1] and the successful organization of the present workshop last year dedicated to embedded systems education [7].

Embedded systems as a field, is often perceived as a conglomeration of different areas, and is truly interdisciplinary [4][7]. Thus teaching "embedded systems" as a unified topic is viewed as a difficult task [6]. This field is often perceived by undergraduate Computer Science (CS) students as "hardware" oriented, not without reason. In our experience at our university, this perception hinders CS students from considering this dynamic field as an option for their future career, despite the fact that embedded systems offers exciting new opportunities for them. This is truer these days with the growing importance of the "software" component of embedded systems. In this situation it is imperative that CS students' misconceptions about this field be removed and they be introduced to the exciting new possibilities.

As we reported earlier [12], most university courses tend to emphasize the "systems" in embedded systems. The emphasis on the "software" side is not that prevalent. The recent papers on embedded education in [1] have mostly emphasized about a comprehensive embedded systems curriculum. These have often originated in universities with established embedded systems research groups. Many universities may not have either the expertise, resources or flexibility to introduce comprehensive curricula dedicated to embedded systems. Instead, a more feasible alternative would be to offer elective courses on embedded systems and software. It is from this perspective that we introduced two new courses into our curriculum at the Hong Kong University of Science and Technology (HKUST), one aimed at embedded systems and the other at embedded software, to give our undergraduate students the flexibility to take these as electives in order to gain knowledge and familiarity with embedded systems.

In this paper, we describe our experience in designing and offering an embedded software course specifically concentrating on the software requirements and design for embedded systems. We briefly revisit the course that we described earlier [12] including the list of topics, the hands-on laboratory exercises and some reflections on the experience with the second offering of the course. We also give a brief account of the students' perception and opinions on the course, after some modifications were introduced into the course based on feedback gathered in the earlier offering of the course [12].

The paper is organized as follows. Section 2 reviews some of the background for the course. Section 3 provides the details of the course. Section 4 reflects on the students' opinions. Finally we give conclusions in Section 5.

2. BACKGROUND
Many universities have been actively engaged in designing new embedded systems curricula as evidenced by the papers published in the recent special issue of ACM Transactions on Embedded Computing Systems (TECS) dedicated to embedded systems education [1]. Similarly a number of papers describing faculties' experiences with designing and offering embedded systems related courses were published in the previous workshop on embedded systems education [7].

It is interesting to note that most of the papers in the special issue of TECS [1] were dedicated to describing embedded systems courses offered at the graduate level or about comprehensive undergraduate curricula. The emphasis has been on comprehensive curricula but with systems perspective. Two papers [2][16] did put emphasis on embedded software.

The observations by Sztipanovits et al. [16] about computer science students' strength is worth noting. They mention that CS students tend to be trained well on abstractions, especially about creating formal abstractions of computational processes and to relate layers of abstractions to each other. However they lack training on the relationship to "physical" processes and systems. These students tend to be trained in formal automata-based and algebraic models, but lack training in the continuous-time domain. Thus designing embedded courses for CS students should keep in mind their strengths and weaknesses.

Given the lack of flexibility in introducing embedded systems concepts into existing curricula without overtly disturbing the curricula, we need to find creative approaches to bring about the modifications. One possible approach that we suggested [12] was to introduce the embedded software concepts into various related CS courses like operating systems, compilers, architecture, programming languages etc. It is interesting to note that such an approach at least in the context of operating systems was also considered by Prof. Nutt [13]. In particular, he suggests that a traditional OS course could be reorganized to emphasize the growing importance of small computer systems. He views the computer systems space as consisting of three types based on the use of interrupts and the CPU mode bit: dedicated systems (DS), Truasted Process (TP) systems and Managed Process (MP) systems, each with increasing capability and complexity. In particular, the DS and TP systems covers what we normally view as the embedded systems space.

Prof. Lee [10] advocates a drastic rethinking on the way embedded software is developed, putting emphasis on the timing aspects of software. He emphasizes that the traditional approach to embedded software has paid more attention to optimizing the software for the resource limitations imposed by the platforms, rather than on timing issues. This overemphasis on efficiency with the consequent neglect of functionality and reliability, ill-serves the embedded software domain. Similarly he mentions that the thread model adopted for concurrent programming is not the best suited approach [9]. Threads introduce uncontrolled non-determinism which is detrimental to the functional and reliability requirements that most embedded software need to exhibit.

The model-driven approach [15][16] to embedded software development has also been advocated. Most of the existing courses that follow this approach are targeted at the graduate level because of the inherent complexity and the need to have sufficient background to appreciate this approach. As mentioned in [16] such concepts are often difficult to teach at the undergraduate level.

Grimheden and Törngren [4] present didactic analysis of embedded systems with detailed discussions to establish the legitimacy and identity of embedded systems. They establish that embedded systems have a functional legitimacy, implying emphasis on skills, and a thematic identity. They advocate an example driven approach with interactive communication.

3. OUR COURSE

In this section we give the details of our course, including the list of course topics, the hands-on laboratory exercises, and the student projects. We also reflect on our experience with the second offering of the course where modifications were introduced based on student feedback from the first offering of the course.

3.1 What Should Be Taught?

After having seen the review of various approaches and opinions presented in the earlier section, the situation begs the question what should be taught to computer science students? More generally, what is feasible to be taught to CS students, given their background knowledge by the time the students reach their Junior/Senior year in their undergraduate education? Indeed, we were faced with this dilemma when we first set out to design and introduce embedded systems "elective" courses into our computer science/computer engineering curriculum. In this section, we reflect upon the guiding principles, thought processes and reasoning behind selecting various topics that we chose to include in the course on embedded software. While we do not claim to answer all the questions, we believe that this sharing of ideas might help elicit similar reflections on part of other faculty facing similar circumstances.

Through our own deliberations on this issue we came up with a set of topics that we felt will add sufficiently to the CS students' knowledge from the perspective of embedded systems. We felt that three broad categories of topics as listed below can be taught to CS students, leveraging on the students' background and the topics that are already part of the curricula. The emphasis was on embedded software development, with hardware concepts being only lightly touched where appropriate.

- Embedded Software Development: CS students are quite well-versed with software development. The main emphasis of this section therefore is how embedded software development differs from traditional software development. We found that the increasing use of integrated development environments like Microsoft Visual Studio™ isolates the students from the entire process of compilation. Thus we felt it was important to emphasize this point, and especially make clear the concept of cross-platform development and issues related to cross-compilation. The students were introduced to the ideas of cross-platform development including host based embedded software development and embedded target environments, integrated development environments, interrupts and interrupt handling, and embedded software architectures.
- Real-Time Operating Systems (RTOS): CS students take operating systems as a core course during their study. The emphasis in such courses is usually placed on efficiency and fairness of resource sharing among processes, and the use of techniques like resource abstraction, virtual memory management and file systems. In the embedded world, the emphasis is more on reliability and timeliness. Thus in an embedded software course, real-time scheduling techniques with deadline constraints need to be emphasized. In our course, task scheduling topics including rate monotonic scheduling, the priority

inversion problem and its solution were covered. Task synchronization issues including the use of semaphores and events were covered. Inter-task communication mechanisms including message queues, mailboxes and pipes were covered. Memory management issues including dynamic memory management, memory leak and dangling pointer problems were covered. Several example RTOS were also introduced in the course. As a simple and efficient real-time kernel, the µc/OS-II was first introduced. Two full-fledged real-time OS viz., Windows CE and Embedded Linux were introduced.

- Embedded Software Engineering: Trying to find a suitable set of material to cover under the embedded software engineering heading was the most difficult, because of the diffuse nature of the area. We leveraged on the existing techniques from software engineering and briefly covered several topics including embedded software development, software development lifecycle, software development models, including the waterfall model, the spiral model, rapid application development model, object-oriented approaches. Sufficient coverage was also given to software testing. Universal modelling language (UML) was introduced as an important formal method for software engineering, and its use in the different parts of the software development lifecycle was illustrated. We also concentrated on specific techniques for testing, verification and validation for embedded systems. It is interesting to note the observations made in Josesfsson [8] that industrial software engineering approach taught by universities are adequate from the technical perspective, but need more emphasis on business, teamwork and practical skills.

While these three are the broad categories of topics that we decided to cover in our course, we do acknowledge that other topics can be considered for inclusion. In particular we decided not to include topics on interfacing and device drivers in this course, leaving it to a companion course on embedded systems. Application programming interfaces like the approach adopted by Phidgets Inc. [14] makes it feasible to do programming with a conceptual view of the hardware, rather than being concerned with the details.

We do notice that the detailed curricula proposed in several papers in [1] at the undergraduate level typically include courses on real-time systems and software engineering. Our course is designed more as an umbrella course covering these topics within a single course, which within a semester offers a reasonable coverage of the aforementioned topics.

3.2 Course Topics and Structure

The list of topics covered in the course is given in Table 1.

Table 1: List of Course Topics

1. Introduction
   - Introduction to Embedded Systems
   - Examples of Embedded Systems
   - Embedded System Characteristics
2. Embedded Systems Architecture
   - Hardware Fundamentals: Processors, Memory, Bus, etc.
   - Software: OS, Application Software
3. Embedded Software Development
   - Hosts and Targets
4. Interrupts
   - Introduction to Interrupts
   - Interrupt Handlers and Interrupt Service Routines
5. Embedded Software Architectures
6. Real-Time Operating Systems (RTOS)
   - Review of Operating Systems Basics
     - Tasks, Processes and Threads
     - Task Scheduling: Rate Monotonic Scheduling, Priority Inversion
     - Task Synchronization and Coordination
     - Intertask Communication
     - Memory Management
   - Example RTOS: µC/OS-II, Windows CE, Embedded Linux
7. Embedded Software Engineering
   - Basics of Software Engineering
   - Software Engineering Models
   - Unified Modeling Language (UML)
   - Software Testing
8. Testing and Debugging Embedded Systems

3.3 Hands-on Laboratory Exercises

The hands-on laboratory component concentrated mainly on introducing the students to embedded software development using Windows CE. A general purpose teaching laboratory equipped with standard PCs was used for the laboratory exercises. The majority of the labs were organized around the Microsoft Windows Embedded software including Platform Builder 4.2 and Windows CE. Students were introduced to the Platform Builder IDE, Visual Studio environment including the Embedded Visual C++ and ".NET" compact framework. Then the students did several laboratory exercises which were aimed at illustrating several RTOS concepts including threads, task scheduling, task synchronization, and memory management, including memory leaks. The laboratory exercises were mainly aimed at preparing the students for designing and implementing the course projects.

The students also had access to Ebox-II Windows CE 5.0 jump-start embedded software development kits from ICOP Technology Inc. [5]. This system is built around a Vortex X86 system on a chip technology with 128 MB system memory and 64 MB IDE bootable flash storage. Also additional Intel PXA-255 based embedded development kits from Emdoor Inc. [3] were also available.

3.4 Course Projects

The course project was an important part of the student assessment, in addition to quizzes and examinations. Students

formed teams of up to 3 students per team and proposed their own project based on their interest. The students had about 2 months to develop their idea, propose the project, design and implement it. Students were encouraged to apply the software engineering principles learnt in the course during the design and implementation of the project, starting from requirements analysis to final implementation and testing.

The students were very enthusiastic and proposed and implemented interesting projects. While there were the typical projects implementing games on handheld devices, several unique projects were also designed. A list of some of the most interesting project topics with a brief description are presented below:

- Automatic "Dim Sum" Ordering System: the students designed a restaurant menu based ordering terminal for ordering Chinese "Dim Sum" which can be deployed at each table enabling customers to enter their orders online. The terminals are connected to a back-end order tracking system in the kitchen and pantry to schedule and supply the ordered dishes to the tables.
- On-line Retail Management System (RMS): This system enabled retail merchants to keep track of their inventories. The system was designed with a Pocket PC handheld connected to a barcode scanner, with the backend database support.
- Stock Manager: This system was implemented for a Pocket PC handheld with the provision for online stock information display and trading.
- Podcast CE: A Windows CE based device was designed to support podcast reception and playback.
- Video Surveillance System: This system enabled surveillance video being streamed to a Pocket PC.
- "Bomberlady" – an embedded linux game: This project implemented the classic "bomberman" video game on an embedded linux platform
- NewStation: This project designed a news kiosk based on Windows CE to display news headlines subscribed from various sources through RSS subscription
- Real Time Operating System – USTOS: This project aimed at designing a very compact real-time operating system with basic functionality from the scratch.

The final project reports of these projects is available online at http://www.cse.ust.hk/~muppala/comp355/project/reports.html.

As can be seen from the long list of topics, several interesting ideas were developed by the students. The students felt that this experience illustrated to them that a whole new arena of small device programming was easily accessible to them and provided them with alternate avenues for future career, compared to the traditional data processing field which is often the biggest employer of CS students in Hong Kong.

4. STUDENTS

As already mentioned the course was designed as a senior undergraduate course. In the second offering of the course, most of the students were in their third year (final year) of their undergraduate education at the university with a small number of students from the second year of study. It must be noted that the Hong Kong higher education system is based on a three year bachelor's degree program. Students entering the university are at the sophomore level of a typical US university. Thus students in the second and third year at the university here are equivalent to students in their junior and senior year of a 4-year bachelor's degree at a typical US university.

4.1 Background

Unlike the first offering of the course in Spring 2005 where most of the students taking the course (almost 99%) were doing their bachelor's degree in computer engineering, the second offering of the course in Spring 2006 attracted a balanced mix of students both from the Computer Science and the Computer Engineering stream. This was more heartening to note because the course was designed to attract students with both computer science and computer engineering background. Almost all the students were in their final year of study.

Two undergraduate courses were listed as prerequisites for our course, viz., Computer Architecture, and Principles of Systems Software (Operating Systems). Most students enrolled in the bachelor's programs in computer science or computer engineering at our university typically take these two courses by the time they have completed the first semester of the second year of their study. Since these two courses provide the necessary background required for introducing embedded systems, we had to devote only a short time at the beginning of our course to review the relevant materials before delving into the topics of our course.

4.2 Student Feedback

The course was very well received by the students in the second offering, surpassing the evaluations from the first offering of the course. At the end of the semester, a comprehensive survey of the students' opinions was carried out in order to assess how well the course met the students' expectations. The results of the survey indicated that the students were satisfied with the course. Some suggestions for improvement were given which are noted below:

1. Most students expressed interest in having more hands-on labs that currently available. The students expressed a desire to see at least some of the labs organized in a step-by-step manner to illustrate the development of an example embedded system from conception to completion. Currently the labs are more focused on providing the students the skills in illustrating the RTOS concepts and illustrating the capabilities of Windows CE.
2. The topics that attracted the most interest among the students were embedded development, and RTOS. The least popular topic was software engineering, perhaps because of lack of demonstrative case studies. This was similar to the opinions expressed by the students in the first offering of the course.
3. In the earlier offering, the students expressed the opinion that the coverage of the RTOS should be limited to an in-depth coverage of only two or three major ones, rather than an overview of several RTOS. Thus in the second offering we restricted ourselves to only Windows CE and Embedded Linux. This was much better received by the students.
4. Some students expressed the opinion that some of the topics had overlap with other related courses that they had taken, and hence suggested that the overlap should be minimized. This was especially true for software engineering which is covered in detail in another course dedicated to the topic. We

tried to address this problem to some extent in the second offering of the course, but could not completely avoid the overlap.

5. CONCLUSIONS

In this paper we discussed our effort at introducing an embedded software course, with the principal aim of bring embedded systems closer to computer science students. We described the structure of the course, the list of topics, the labs and the course projects implemented by the students. We also discussed some of the findings of the survey on students' opinions of the course. Through this effort we wished to illustrate that it is feasible to introduce computer science students to this exciting new field without getting trapped into the intricacies of the underlying hardware.

6. ACKNOWLEDGMENTS

The authors wishes to thank the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology for providing him the opportunity to explore the issues presented in this paper.

7. REFERENCES

[1] Burns, A., and Sangiovanni-Vincentelli, A. Editorial for the Special Issue on Embedded Systems Education, *ACM Trans. Embedded Computing Systems, 4*, 3 (Aug. 2005), 469-471.

[2] Caspi, P. et al. Guidelines for a Graduate Curriculum on Embedded Software and Systems. *ACM Trans. Embedded Computing Systems, 4*, 3 (Aug. 2005), 587-611.

[3] Emdoor Inc. http://www.emdoor.com.

[4] Grimheden, M. and Törngren, M. What is Embedded Systems and How Should It Be Taught? – Results from a Didactic Analysis. *ACM Trans. Embedded Computing Systems, 4*, 3 (Aug. 2005), 633-651.

[5] ICOP Technology Inc., http://www.icop.com.tw.

[6] Koopman, P., et al. Undergraduate Embedded System Education at Carnegie Mellon. *ACM Trans. Embedded Computing Systems, 4*, 3 (Aug. 2005), 500-528.

[7] Jackson, D. J. and Caspi, P. Embedded Systems Education: Future Directions, Initiatives, and Cooperation. *ACM SIGBED Review Special Issue on the First Workshop on Embedded Systems Education (WESE 2005), 2*, 5 (Oct. 2005), 1-4.

[8] Josefsson, M., Ed. 2003. *Industriell Programvaruutveckling*. ITQ Nordic Institute.

[9] Lee, E. A. The Problem with Threads. *IEEE Computer*, *39*, 5 (May 2006), 33-42.

[10] Lee, E. A. Absolutely Positively On Time: What Would It Take? *IEEE Computer*, *38*, 7 (July 2005), 85-87.

[11] Lee, E. A. What's Ahead for Embedded Software? *IEEE Computer*, *33*, 9 (Sep 2000), 18-26.

[12] Muppala, J. K. Experience with an Embedded Systems Software Course, *ACM SIGBED Review Special Issue on the First Workshop on Embedded Systems Education (WESE 2005), 2*, 5 (Oct. 2005), 29-33.

[13] Nutt, G. An OS Course for Small Computer Systems, submitted for publication, http://www.cs.colorado.edu/~nutt/SCC-OS-paper.pdf.

[14] Phidgets, Inc. http://www.phidgets.com/.

[15] Sangiovanni-Vincentelli, A. L., and Pinto, A. An Overview of Embedded System Design Education at Berkeley. *ACM Trans. Embedded Computing Systems, 4*, 3 (Aug. 2005), 472-499.

[16] Sztipanovits, J. et al. Introducing Embedded Software and Systems Education and Advanced Learning Technology in an Engineering Curriculum. *ACM Trans. Embedded Computing Systems, 4*, 3 (Aug. 2005), 549-568.

The Development and Deployment of Embedded Software Curricula in Taiwan

Shiao-Li Tsao
Dept. of Computer Science
National Chiao Tung University,
Hsinchu, Taiwan
sltsao@cs.nctu.edu.tw

Tai-Yi Huang
Dept. of Computer Science
National Tsing Hua University,
Hsinchu, Taiwan
tyhuang@cs.nthu.edu.tw

Chung-Ta King
Dept. of Computer Science
National Tsing Hua University,
Hsinchu, Taiwan
king@cs.nthu.edu.tw

ABSTRACT
The Embedded Software (ESW) consortium under the VLSI Circuits and Systems Education Program which is supervised by the Ministry of Education (MOE) of Taiwan was initiated in 2004 to develop and promote embedded software education. One of the major missions for this consortium is to develop and deploy embedded software curricula which are not well established in traditional computer science (CS) and electrical engineering (EE) education programs in Taiwan. Therefore, the ESW consortium has spent two and half years in developing embedded software curricula with total 12 new courses. In this paper, our strategies, implementation, and experiences for developing and deploying the embedded software curricula are presented.

Keywords
Embedded Software (ESW), Educational Curricula, System-on-Chip (SoC) Design

1. INTRODUCTION
Taiwan is one of the leading countries in the world in manufacturing information technology (IT) and integrated circuit (IC) products. To improve global competitiveness of local IT/IC industries, Taiwan government is seeking strategies to help local companies to migrate their businesses from manufacture-oriented products to design-oriented products. Embedded software (ESW) that provides high add-on values for IT/IC hardware is regarded as one of the key strategies [3]. Therefore, the Ministry of Education (MOE) of Taiwan has initiated the Embedded Software consortium under the VLSI Circuits and Systems Education Program since 2004 to promote the ESW education and build up the fundamental research and development energies for embedded software technologies [1][2]. The missions of the ESW consortium are to develop the embedded software curricula for universities, promote the embedded software education and activities, and bridge the connections between industries and universities to exchanges ideas and new technologies. Hence, several task groups under the ESW consortium are formed to handle the specific working items and missions for the consortium [4].

The development and deployment of the ESW curricula are the major missions for the consortium and are handled by the curricula development task group. First, the task group investigates curricula for embedded software, embedded system, computer science, and computer engineering developed by ACM, IEEE-CS and leading universities in the world [5][6][7][8][9][10][11][12][13][14]. Then, the reference ESW curricula were developed after a number of meetings with the ESW advisory board which are formed by distinguished professors in the world and industrial executives. The curricula consider the needs from local industries, cooperate with the existing courses and curricula for computer science (CS) and electrical engineering (EE) programs and offer both theoretical and practical trainings for students. The reference curricula also provide a tailoring guideline for universities who adopt the curricula in their education programs. Based on the curricula, course modules that are missing or not well established in the current programs are then developed. Cooperating with the development of the curricula and course modules, a deployment program which helps universities to develop their own ESW curricula in the undergraduate and graduated programs and sponsors universities to establish laboratories for the ESW education is also initiated. With the investment of efforts for the past two and half years, the curricula and infrastructures for the ESW education have been successfully established in many universities in Taiwan. Now, the curricula development task group moves to the next stage to establish a database and guideline for hands-on labs which are extremely important to the ESW education, and keep developing course modules to address new ESW technologies such as hardware/software co-design, embedded multi-core programming, simulator, emulator, and debugger design for SoCs/embedded processors.

The rest of the paper is organized as follows. Section 2 presents the strategies to develop and deploy the ESW curricula. Section 3 describes the curricula and their development processes. Section 4 discusses the results and experiences, and finally, Section 5 summarizes our future work.

2. OUR STRATEGIES
The curricula development task group of the ESW consortium develops the plans and strategies in the very beginning phase. These strategies have been modified and enhanced according to the feedbacks from professors and inputs from MOE representatives and members in the ESW advisory board. Figure 1 and Figure 2 illustrate the development and deployment flow charts of the ESW curricula, respectively. During the development phase, the reference ESW curricula are first developed based on the curricula suggested by ACM, IEEE-computer, and other universities. The ESW advisory board members review the ESW curricula and provide comments and suggestions to improve the curricula to fulfill the needs. According to the reference ESW curricula, missing courses and

the courses without mature lecture and hands-on lab materials are determined. Then, the projects which develop the lecture materials together with hands-on labs for these courses are initiated. The ESW office is responsible for seeking candidates for the project leaders that must be senior or experienced professors in the fields related to the courses. The project leader further needs to form a development team which has at least four domain professors. The MOE then sponsors the team for three years with a total budget about USD 80,000. During the first year of the project, the professors in the team develop course and hands-on lab materials. The deliverables are lecture presentation slides, and notes for hands-on labs. Professors who develop the course are also encouraged to write a text book cooperating with the slides and lab materials. They will receive additional funding for writing a text book. During the second and third year, the team receives the feedbacks and inputs from other professors adopting the course materials, and maintains the slides, notes and the text book. Besides the development of course materials, they need to have a trial run of the course in their universities. Also, they have to introduce the course to other universities in the curricula promotion workshop. The curricula promotion workshop is an important and public event to introduce and promote the curricula, and it is held twice a year and hosted by the ESW consortium. This is also an important occasion for the development team to gather inputs from other professors and to exchange the ideas and teaching experiences with them. Besides curricula promotion workshops, all project leaders of the courses have to attend the regular consortium meetings which are held quarterly. In the meeting, project leaders have to update their status and may request other supports from the consortium or the MOE.

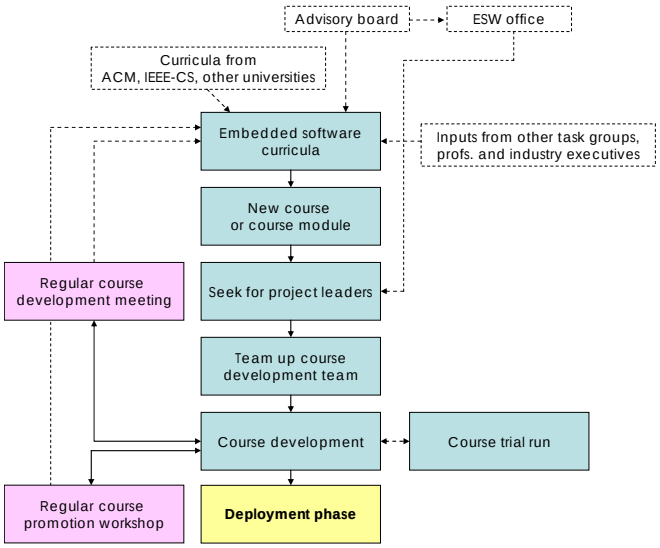


Figure 1. Development strategies and flowchart

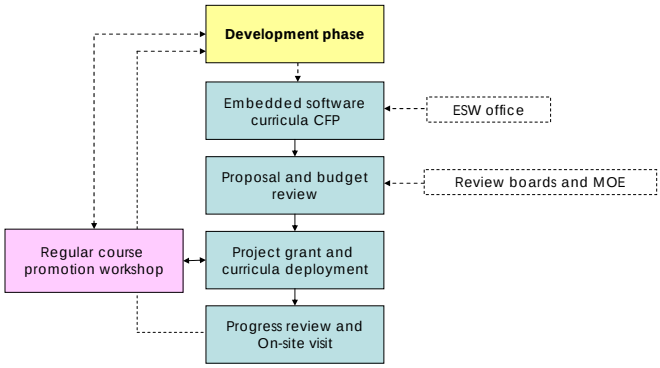


Figure 2. Deployment strategies and flowchart

Once new course materials become stable, the new courses move to the second stage, i.e. deployment stage, which is to promote the course to other universities and encourage other professors to adopt the course. In other to encourage the universities to use the curricula and materials developed by the ESW consortium, a deployment program which is also sponsored by the MOE of Taiwan is offered. The deployment program helps universities to develop their own ESW curricula for undergraduate and graduated programs and build up the infrastructure such as a dedicated laboratory for the ESW education. The program normally sponsors universities for two years, and the total budget for two years is about USD 100,000 to USD 150,000 per university. The budget is mainly used for equipment purchase and to setup the ESW education laboratory. Besides the funding from the government, the universities who apply the program have to commit a dedicated space for the laboratory and at least 20% matching fund of the total budget.

According to experiences learned by other sponsorship programs for curricula deployment, course-based sponsorship programs have several drawbacks. For example, universities receive budgets and have this course in their programs in the first one or two years, but the course may be removed from their programs if there are no continuative funding and project support. Also, the course might overlap with the other existing courses and cannot be integrated into the existing curricula of the educational programs. To overcome this problem and also consider that the ESW curricula normally require equipment support, the sponsorship program for the ESW curricula changes to curricula-based and laboratory-oriented. To achieve this goal, the reference ESW curricula and a list of new course modules developed by the ESW consortium are provided in the call for proposal (CFP). Universities who would like to submit a proposal has to provide their own ESW curricula which could be complete new one, tailored version from the reference curricula, or an enhancement based on their existing curricula. They must commit to have a dedicated laboratory for the ESW education and also to have some new courses or courses with enhancements in their curricula. In order to deliver the new courses or enhance their existing course materials, universities should propose their needs for the equipment which must be installed in the dedicated ESW education laboratory. The review committee evaluates the proposals and decides the amount of budget based on their curricula and their plans to establish the ESW laboratories. This

approach makes the sponsorship program more effective and efficient than the previous course-based sponsorships.

Besides establishing the ESW curricula and laboratory, the professors must participate the semi-annual curricula promotion workshops. They have to present their experiences, and demonstrate their results such as hands-on labs from students. The sponsorship program is for two years but has an annual review to evaluate the progress and determine the budget for the second year. The statistical information such as the number of equipment purchased, the numbers of new courses which are delivered or enhanced, the quality of the course and lab materials, and the number of students enrolled these courses are collected and evaluated. The review consists of on-site review which checks the laboratory, equipment, course modules and interviews with professors and students, and paper documents review.

3. EMBEDDED SOFTWARE CURRICULA

Universities in Taiwan are categorized into universities and technical universities which provide quite different training to students and have different education objectives. Universities train students with research, design and development abilities. Technical universities offer trainings to students with technical skills and implementation abilities. To address the different needs, the curricula deployment task group has developed two reference ESW curricula for universities and technical universities. Also, the courses and materials are different in the two curricula. The sponsorship programs for the course deployment for universities and technical universities are also separated. Figure 3 depicts the curricula for universities. The courses are classified into embedded hardware and System-on-Chip (SoC) courses which focus on hardware design, embedded system software courses which introduce system software for embedded systems, and embedded application courses which provide the training for developing embedded applications and services. Also, the courses can be categorized into three levels, i.e. fundamental courses for undergraduates in the third and fourth year, intermediate courses for senior undergraduate students or graduated students, and advanced courses for graduated students such as master or PhD students. According to the proposed curricula, eight courses are either newly developed or enhanced. Except the embedded middleware course which is still being developed, other seven courses have been developed. Brief descriptions of the eight courses are:

- **Embedded software programming**: Programming languages and design considerations for applications running on embedded systems and SoCs are quite different from these on general purpose PCs. The course introduces several well-know programming languages for embedded systems such as embedded C/C++, Java for mobile edition (J2ME), and C#. Moreover, the course introduces optimization technologies such as down-sizing, power consumption and etc. for embedded software development.
- **Embedded tool chains**: This course consists of five modules and a case study on an open-source tool chain. The modules are tool chain introduction, basic compilation techniques, assembler and linker, optimizations for high-performance and code density, and energy-saving methodologies. The case study gives students hands-on experiences on porting an open-source tool chain to both a RISC and a DSP platform.
- **Implementation of embedded operating system (EOS)**: EOS which is different from OS has to deal with many hardware-specific functions and/or customized optimizations. A new course focusing on bootloader design, EOS design, porting and optimizations is thus developed.
- **Embedded middleware**: Traditionally, electronics vendors are very adept at providing embedded hardware platforms and low system-level software at competitive cost; however, they often lack necessary software skills to develop a complete, complex embedded applications and systems that have much higher profit margins. Developing these high-margin embedded applications require good understanding of the various embedded middleware platforms and knowledge in emerging international middleware standards that provide interoperability among different embedded systems and H/W vendors. The goal of this course is to educate students about fundamental concepts in embedded middleware as well as hands-on programming skills on these emerging middleware platforms and standards. The lab of this course contains two parts. The first part includes individual programming assignments which each student will practice programming APIs of different embedded middleware systems (such as .Net Compact, J2ME, Jini, RFID middleware, etc.). The second part includes a course project where students will form teams to complete complex embedded applications by drawing components from different embedded middleware systems.
- **I/O and device driver**: This course is developed to answer strong demand by system companies for well-trained engineers in porting and writing device drivers. To satisfy this need, the section of I/O management in the fundamental course of Operating System Concepts with emphasis on device drivers and extensive implementations is elaborated. Students are required to implement drivers for devices such as communication interfaces, and audio and video cards. In addition, for hardware-accelerated capabilities, students are trained to implement a device in a FPGA extension and to access it through a device driver.
- **Embedded real-time system**: This course extends the one on Embedded OS Design and Implementations to deal with real-time characteristics required by an embedded real-time system and real-time operating system. Modules such as real-time scheduling, resources management, issues on priority inversion, and real-time driver architecture are essential. Students will acquire knowledge through both in-class lectures and implementation projects on open-source real-time operating systems.
- **Embedded compiler design**: This course starts with general introduction and background information on embedded compilers. It next shifts its focus to optimizing skills for developing efficient and energy-saving object code for power aware embedded systems. The course consists of four dependent modules: ILP compiler introduction, DSP compiler introduction, compilers for

embedded parallel processors, and data dependence analysis for embedded heterogeneous processors.

- **ESW for networked SoC devices**: The course is the project-oriented course that aims to give students a system level practice on embedded system and software. Instead of focusing on a stand-alone system, this course addresses more on the networked SoC system. Heavy hands-on experiments are required for students to build an intelligent transportation coordination system on robot cars.

Different from universities, technical universities train students to have more programming and engineering skills. The ESW Consortium thus invites professors in the technical universities to develop the suitable courses and curricula for technical universities. Figure 4 depicts the reference ESW curricula for the technical universities. The courses for technical universities are classified into embedded hardware/SoC courses and embedded software courses. Similar to the curricula for the universities, courses are categorized into fundamental, intermediate and advanced levels. Some of courses are specially designed and developed for technical universities. Some of courses are tailored from the course modules developed for universities but hands-on labs are enhanced. The course development and curricula promotion process are identical to the universities, but faculties from technical university are invited to develop the courses. Currently, there are four new courses which are already developed. The objectives of the development courses are briefed below.

- **Introduction to embedded system**: Embedded systems are not only applied to IC or IT industries, but also widely employed on control, mechanical, aerospace, biological, automobile industries. A general introductory course on embedded system could be very useful to these students who are not major in CS or EE but need embedded system skills or knowledge on their domains. Therefore, the course is designed and developed. This course is an entry level course and provides general introduction and hands-on practices on all aspects of an embedded system.

- **Embedded system labs**: 8-bits micro-controllers such as 8051 are normally used as the education platform for micro-computer system training. Advances in IC technologies, the usages of 16-bits, 32-bits embedded processors right now become the market trends. To help technical university students to step into the high-end embedded system development, the course uses ARM as the reference architecture to introduce the embedded system design. The course provides complete and step-by-step hands-on trainings for students to practice the development of ARM-based embedded systems.

- **Interface design**: Interfaces such as RS-232, parallel ports for computer I/O systems are frequently used in the technical universities. However, these interfaces are upgraded to faster and more complicated interfaces such as I2C, I2S, USB, P1394, PCMCIA, and etc. To help students to learn such new interfaces and have hands-on experiences on these interfaces are the major goals of the course.

- **Implementation of USB devices/drivers**: As USB quickly becomes a primary choice for the interfacing protocol, it is critical for a portable embedded system to provide USB ports alone with USB device drivers and firmware. To meet such a demand, this course that is specifically designed for technical universities is proposed. This course provides a comprehensive hands-on training on USB technology including device driver, firmware, and physical-layer IP. Each student needs to develop software on a host to recognize and control a device through USB interface. Students also learn to use a set of tools such as a USB analyzer, oscilloscope, and function generator to facilitate debugging their software.

4. RESULTS AND EXPERIENCES

The ESW consortium has initiated 12 course development projects since 2004. Eight courses are for universities and the other four courses are designed for technical universities. The 12 projects received about total USD 600,000 for the course development, and total USD 150,000 per year for maintenance. 43 professors from more than 20 universities involve the course development. Among 43 professors, 18 professors are from EE and related departments, and the others are from the CS department. More than five curricula promotion workshops which introduce, promote the course and curricula, and demonstrate the results were held. Also, more than 400 attendees including professors, students, and engineers from industries participated the events.

As for the sponsorship program to deploy the ESW curricula, the MOE of Taiwan received 24 proposals and approved 11 proposals last year. A total USD 600,000 is funded for 11 universities for the first year. It is expected to have some amount of budget to support the 11 universities for the second year. The statistical data for the first year shows that the ESW education infrastructures have been established in the 11 universities. More than 30 new courses or courses with enhancements are lectured in the 11 universities in the first year. A total of 1000 students enrolled these courses under the ESW curricula deployment program. In this year, the ESW consortium also announced the call for proposal for technical universities and the ESW curricula will be deployed over technical universities.

Several new issues and challenges are raised in the past two years. The first one is about the common education platform. The curricula developed by the ESW consortium require heavy hands-on practices but the hands-on platforms are usually different in the course modules. It is very expensive for universities to purchase several lab platforms and also very difficult for professors, teaching assistants (TAs) and students to learn different lab platforms. To resolve this problem, to develop different sets of hands-on labs over various platforms or to require all hands-on labs to be developed over one or two common hardware platforms are two possible approaches. The first approach requires the investment of time and teaching assistants' resources in developing reference labs. The second approach requires defining one or two hardware platforms which meet the needs by all courses. Since the courses normally require open sources and rich technical supports from vendors or open source community, the selection of platform is quite difficult at this stage. It is believed after running the curricula for more years, the platforms will be harmonized and the reference hands-on labs over different platforms become rich. The second challenge is

about the development of hands-on labs. ESW course usually requires heavy hands-on practices but the development of hands-on labs needs significant efforts to make them useful and complete. To maintain the hands-on labs including TA notes, reference source codes or reports, and knowledge and experiences learned from the labs are extremely important. In the last year, the ESW consortium starts to establish a database for hands-on labs to resolve this issue. The ESW consortium requests each course development project to turn in at least four hands-on labs which include a step-by-step TA notes, and a hardware and software platform for the lab. These notes and platforms were sent to other professors for peer reviews. The review teams need to reproduce the labs based on the lab notes and platforms. The review comments are sent to the development team to improve the note qualifies. Once the lab notes are approved and these hands-on labs can be checked in the database. Currently, more than 40 hands-on labs have been established in the lab database which is shared by all professors who adopt the course modules or lab modules.

5. CONCLUSIONS AND FUTURE WORK

The paper presented the strategies to develop and deploy ESW curricula in Taiwan. First, the reference ESW curricula that provide a comprehensive and modern training to students were proposed for different needs of universities and technical universities. New courses and hands-on labs were developed to support the curricula. Then, the deployment program which encourages universities to establish ESW curricula and helps universities to establish the ESW education laboratory is also offered. Our future working items are to seek a common teaching platform, construct a complete database for the hands-on labs and hands-on experiences, and develop more advanced courses to support ESW research and education.

ACKNOWLEDGEMENT

The authors would like to thank the Ministry of Education of the Republic of China for financially supporting this program and the ESW consortium.

6. REFERENCES

[1] The SOC Consortium, VLSI Circuits and Systems Education Program, Ministry of Education, Taiwan, http://moesoc.ee.ntu.edu.tw.

[2] The ESW Consortium, VLSI Circuits and Systems Education Program, Ministry of Education, Taiwan, http://esw.cs.nthu.edu.tw/.

[3] Industrial Economics and Knowledge Center, "The Supply-Demand Analysis of High-Tech Engineers in Key Industries and Deficit-Recovery Solutions," Technical report, Industrial Technology Research Institute, 2003.

[4] Tai-Yi Huang, Chung-Ta King, Yin-Tsung Hwang, and Youn-Long Steve Lin, "The Embedded Software Consortium of Taiwan," ACM Transactions on Embedded Computing Systems Special Issue on Embedded Systems Education, 4(3):612-632, August 2005.

[5] ACM Curricula Recommendations, http://www.acm.org/education/curricula.html.

[6] David Jeff Jackson and Paul Caspi, "Embedded Systems Education: Future Directions, Initiatives, and Cooperation," First Workshop on Embedded System Education (WESE), 2005.

[7] Alberto Luigi, Sangiovanni-Vincentelli, and Alessandro Pinto, "Embedded System Education: A New Paradigm for Engineering Schools?" First Workshop on Embedded System Education (WESE), 2005.

[8] Suehee Pak, Eunha Rho, Juno Chang, and Moon Hae Kim, "Demand-Driven Curriculum for Embedded System Software in Korea," First Workshop on Embedded System Education (WESE), 2005.

[9] Peter Marwedel, "Towards laying common grounds for embedded system design education," First Workshop on Embedded System Education (WESE), 2005.

[10] Alberto L. Sangiovanni-Vincentelli, and Alessandro Pinto, "An overview of embedded system design education at berkeley," ACM Transactions on Embedded Computing Systems Special Issue on Embedded Systems Education, 4(3): 472 - 499, August 2005.

[11] Philip Koopman et al., "Undergraduate embedded system education at Carnegie Mellon," ACM Transactions on Embedded Computing Systems Special Issue on Embedded Systems Education, 4(3): 500 - 528, August 2005.

[12] Janos Sztipanovits et al., "Introducing embedded software and systems education and advanced learning technology in an engineering curriculum," ACM Transactions on Embedded Computing Systems Special Issue on Embedded Systems Education, 4(3): 549 - 568, August 2005.

[13] Rudolph E. Seviora, "A curriculum for embedded system engineering," ACM Transactions on Embedded Computing Systems Special Issue on Embedded Systems Education, 4(3): 569 - 586, August 2005.

[14] P. Caspi et al., "Guidelines for a graduate curriculum on embedded software and systems," ACM Transactions on Embedded Computing Systems Special Issue on Embedded Systems Education, 4(3): 587 - 611, August 2005.

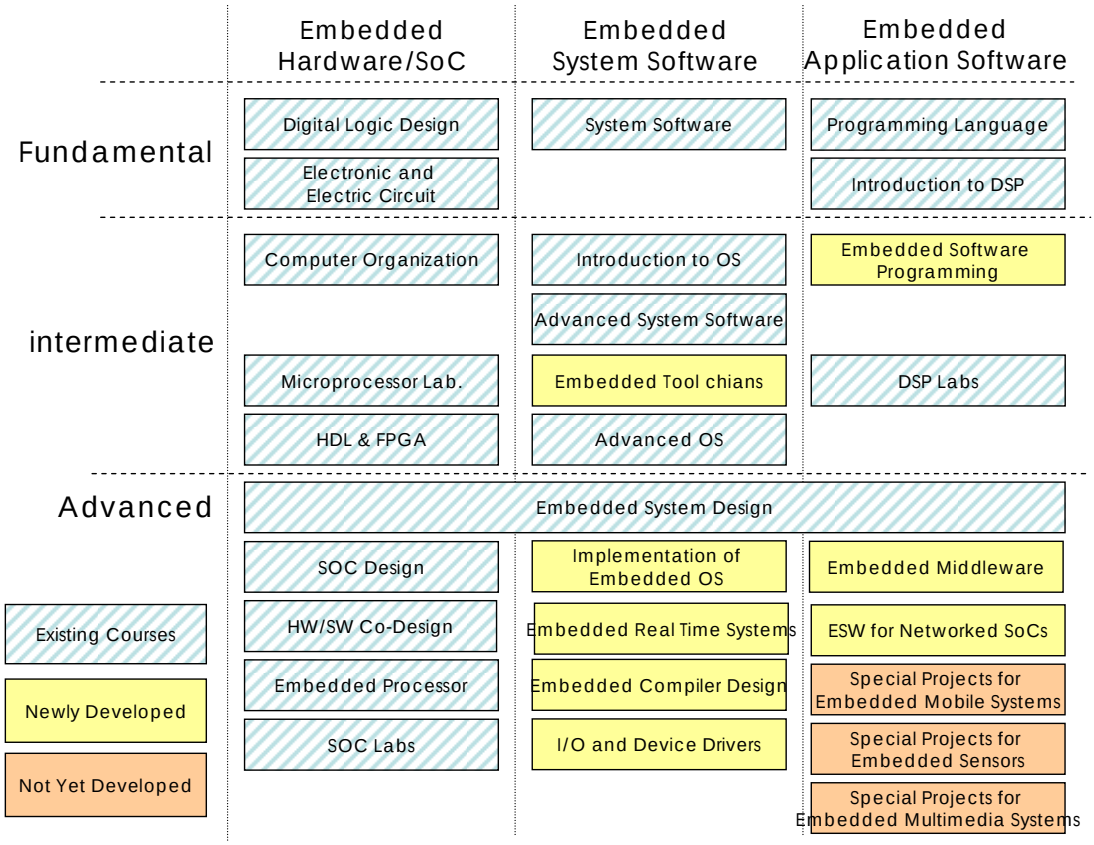


Figure 3. Embedded software curricula for universities

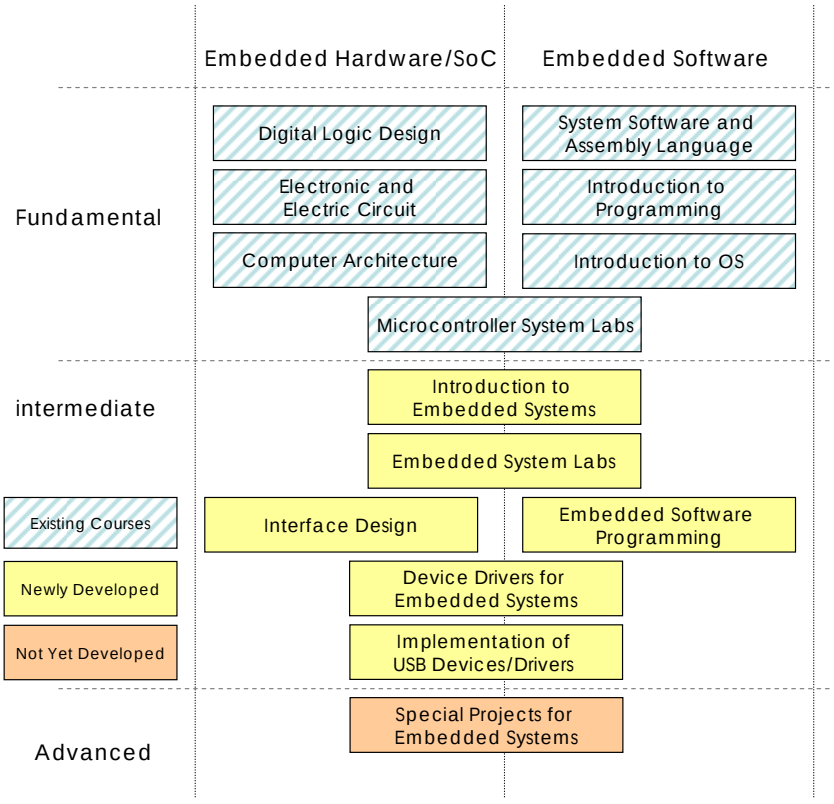


Figure 4. Embedded software curricula for technical universities

Methodologies to Bring Embedded Systems to Non-EE Students

Shekhar Sharad
National Instruments
11500 N Mopac Expwy, Austin, TX
78759, USA
+1-512-683-5219
shekhar.sharad@ni.com

ABSTRACT
With embedded systems being used in every industry, it is important to empower domain experts who are not embedded design engineers to be able to design, prototype and deploy them in their applications. In this paper, we explore some graphical methodologies available today that provide a higher level of abstraction that can help professors teach embedded systems to non-EE majors. We will identify the advantages such methodologies present and explain with some marquee examples such as ChallengeX and LEGO.

Categories and Subject Descriptors
Dataflow programming, event structures, abstraction

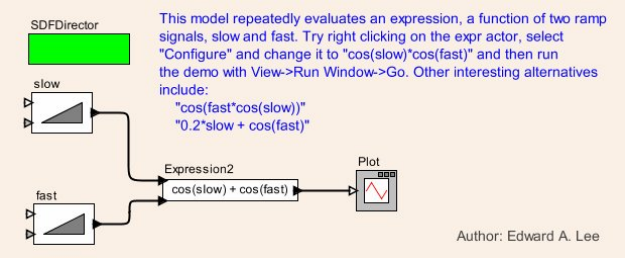
Keywords
Embedded, innovative teaching methodologies, hands-on learning, non-EE majors

1. INTRODUCTION
Embedded systems are becoming increasingly pervasive, from space stations to everyday objects such as cell phones and microwaves. With technologies converging and silicon shrinking, embedded systems now present viable opportunities for individuals and teams that are not embedded design engineers but domain experts in their field of choice. For example, a mechanical engineer building the next generation vehicle may not be an embedded design engineer but is an expert on the dynamics of a vehicle including the design of the engine and the drive-train. Hence there is a need for tools that provide an increasing level of abstraction while providing the power and flexibility of embedded systems. One of the biggest hurdles that traditional programming methodologies present to teaching embedded systems to non-EE majors is that of complexity. Non-EE majors may have little to no textual programming experience and hence there is a need for ways that present an intuitive interface to designing embedded systems so that Professors can explain the concepts and students can understand and use embedded platforms for their projects.

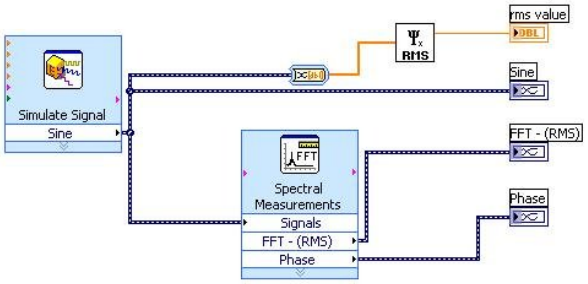
One such effective methodology is using Graphical tools and techniques to teach embedded systems. In this paper, we will present how graphical programming present a viable platform to teach embedded design. We will elucidate on some of the advantages that such graphical programming methodologies present. We will also show how graphical programming methodologies provide a flexible platform that can be used to target embedded hardware such as FPGAs, DSPs or Microcontrollers giving users the option to completely design, prototype and deploy their system. We will also list some of the marquee programs around the world that have been using such techniques to teach embedded systems.

2. GRAPHICAL PROGRAMMING
It is common in embedded and signal processing communities to use block diagrams to represent systems. The blocks typically represent computations and the arrows connecting the blocks represent flow of data. Graphical programming is based on the "dataflow" paradigm which in mathematical terms is a directed graph whose nodes represent computations and the arcs represent streams of data. The nodes may consist of subsystems that consist of a series of nodes and arcs or contain algorithms implemented in a primitive language such as assembly or C. As it can be observed, graphical programming languages lend themselves naturally to embedded and DSP programming which has resulted in a lot of research and development in graphical programming languages.

There are a number of tools that allow a mixture of visual and textual programming such as Signal[1], Lustre[2] and Silage[3]. Completely graphical programming environments such as Ptolemy[4] from University of California, Berkeley and National Instruments LabVIEW[5] have evolved to present powerful tools that professors can use to teach embedded systems without compromising on the flexibility and control that the traditional methodologies provide. Figure 1 shows a typical screenshot from both Ptolemy and LabVIEW.



(a) [4]



(b)

Figure 1. Block Diagrams from Ptolemy (a) and NI LabVIEW (b)

As it can be seen from figure 1, both of these block diagrams have several common features. First, both languages use a dataflow approach that comes naturally to engineers designing systems, embedded or otherwise. Second, the interface provides a high level of abstraction, thereby resulting in a clean interface that supports debugging. Third and most important for professors, both of these representations lend themselves naturally to teaching concepts using a step-by-step approach.

3. EMPOWERING DOMAIN EXPERTS

As noted in a previous section, the designers are not necessarily DSP-design experts. However, they are experts in their application area and want to use embedded platforms such as DSPs and FPGAs for their application because of the advantages that these platforms provide. Conventional programming methods increase the learning curve making it difficult to use DSPs or FPGAs for application development, especially for domain experts who are not embedded design engineers. Graphical programming languages alleviate this problem by providing a simple, easy-to-use interface that can be used to program DSPs, FPGAs and microcontrollers empowering the domain experts to quickly design, prototype and deploy systems. Figure 2 compares the steps that are required when using conventional and graphical programming techniques for DSPs. A similar chart can be drawn for other hardware platforms such as FPGAs or Microcontrollers.

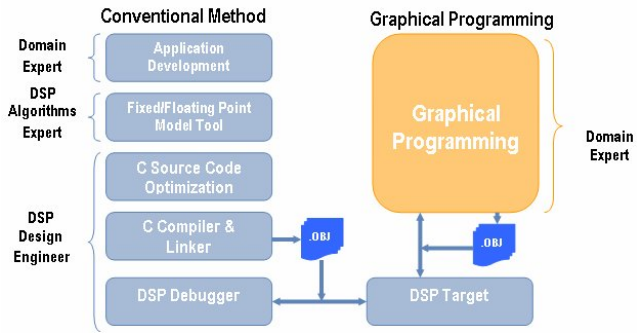


Figure 2. Abstracting complexities using graphical programming

The comparison between the two approaches in figure 2 present a very interesting perspective to teaching embedded systems to non-EE majors. The domain expert, in this case, the non-EE students are concerned with their design only. By providing a higher level of abstraction we are able to abstract the complexity in designing the embedded system helping the domain expert focus on the design at the same time teaching them how to use embedded systems.

4. A MARQUEE EXAMPLE - ChallengeX

ChallengeX[6] is a multi-year student competition in which seventeen teams have been challenged to re-engineer a GM Equinox, a crossover sport utility vehicle to minimize energy consumption, emissions, and greenhouse gases while maintaining or exceeding the vehicle's utility and performance.

The team that won ChallengeX in 2006, Virginia Polytechnic Institute and State University in the United States consisted primarily of mechanical engineers who are experts in the design of vehicles. In this competition, there was a need to use embedded systems that could be used to build complex Control Units and other regulatory parts for the Equinox. Figure 3 shows the overall architecture that Virginia Tech presented.

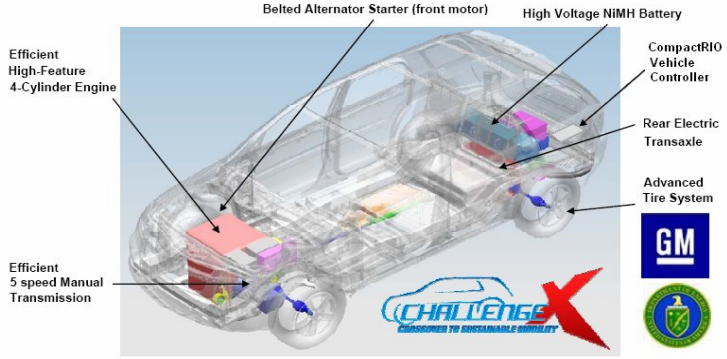


Figure 3. ChallengeX Vehicle design from Virginia Tech

Virginia Tech used graphical system design techniques to design their system. They used NI LabVIEW to program an FPGA-based platform, NI CompactRIO. The key point here is that the team was not comprised of embedded design experts, but because of they learnt embedded design using graphical techniques, they were able to design a highly efficient embedded control system.

5. ANOTHER EXAMPLE – LEGO

LEGO Mindstorms have been used by several professors and educators to demonstrate engineering concepts [7][8][9][10]. While it may seem as a toy, it is also a powerful robotics platform. For example, the new LEGO Mindstorms NXT brick is a 32-bit ARM processor that interfaces with sensors and motors and communicates with the host via Bluetooth. The key pedagogical element here is that the individuals for whom this "toy" is aimed at are typically between the 5-15 years of age group! The reason for this "toy" being successful is that it uses a completely graphical interface. Figure 4 shows the LEGO Mindstorms NXT software.

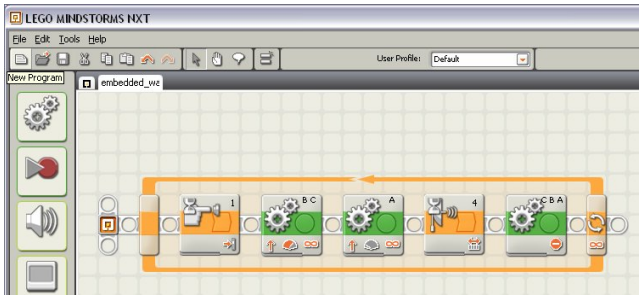


Figure 4. The next generation LEGO Mindstorms NXT software

Not only is this tool being used by high-school kids, but professors from engineering disciplines such as mechanical and aerospace systems are using this tool to help students learn about the nuances of embedded systems such as limitations on memory without having to worry about the implementation details such as memory allocation, pointers and netlist generation. The software takes care of all these details allowing the student to focus on the design of his or her system.

6. BENEFITS OF GRAPHICAL PROGRAMMING

6.1 Drag-and-drop interface

Graphical programming languages provide the user with a true drag-and-drop interface that reduces the learning curve drastically. Figure 5 shows an example of a block diagram from NI LabVIEW. Ptolemy has similar diagrams as well. From a pedagogical standpoint, the benefit for professors is that they can now teach the students using block diagrams using a chalk and a whiteboard and create the same block diagram in software that automatically translates and runs on hardware.

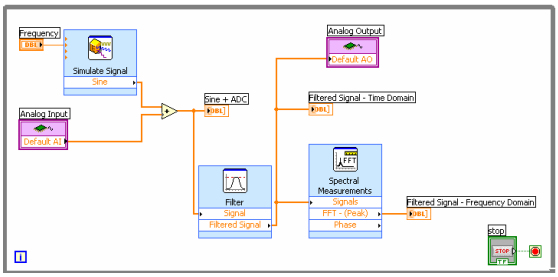


Figure 5. Drag-and-drop interface helps teach concepts easily

6.2 Hundreds of Pre-built Function Blocks

Designing embedded systems is incomplete without signal processing and control functions. As it was shown in figure 5, graphical programming environments aim to minimize textual code needed to design algorithms and provide users with a library of basic functions that can be used to develop embedded systems quickly. Graphical environments achieve this goal by providing pre-built function blocks that can be wired together to design a system. The advantage that graphical environments provide is that the same program that is used for simulation purposes can also be downloaded to supported targets and hence give the students access to real-world signals and enhance hands-on learning. Figure 6 shows some examples of time-domain function blocks.

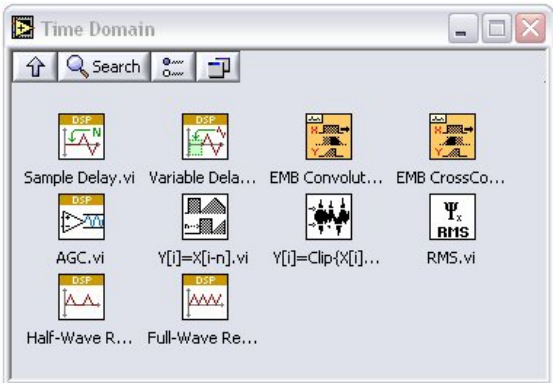


Figure 6. Some pre-built function blocks for time-domain

Professors can also teach students to build complex functions using the basic function blocks and hence promote the concept of reuse in embedded systems.

6.3 Ability to Re-use Existing C Code

Embedded design has traditionally involved a lot of C and assembly code. Professors and students may also have a lot of existing C code that they may wish to reuse. Graphical environments present a ways to reuse existing C code. The underlying implementation may vary for the different environments. One of the simplest techniques is shown in Figure 7. This technique is used in NI LabVIEW. Professors and students can insert their C code and create the input and output terminals and the compiler will compile any C code in the inline-C node into a header file and include it in the embedded project.

This also provides an opportunity for professors to introduce the finer aspects of embedded programming to non-EE majors. An example scenario may involve designing the application using the pre-built function blocks and then redesign the system by replacing the function blocks with C code that may involve custom optimization that enhance the performance of that particular application. In this way, students get to experience the intricacies of embedded programming while still being able to learn about designing the system by using the pre-built function blocks

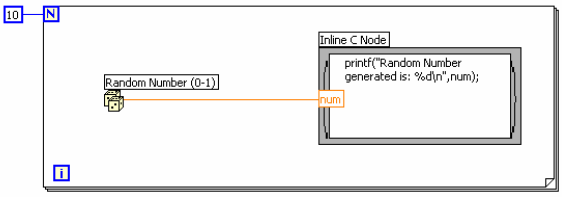


Figure 7. Reusing Existing C code in Graphical Environments

6.4 Inherent Concurrency and Parallelism

In text-based languages, implementation of parallelism often requires a complex balance of library calls to operating system functions, resource management, memory protection, and locking mechanisms. As a result, the compiler needs extensive code to be written to ensure shared sections of code are properly protected, making it cumbersome to build parallel programs. However, parallel programming forms the cornerstone for embedded architectures such as FPGAs that are used extensively in Academia. The same is true for several other applications that needs some form of parallelism. Because graphical programming is based on the dataflow approach, concurrency and parallelism are inherent making it easy to design such systems. Figure 8 shows an example of parallel programming where in input is acquired and filtered in the top loop and a triangle waveform is generated and output in the second loop.

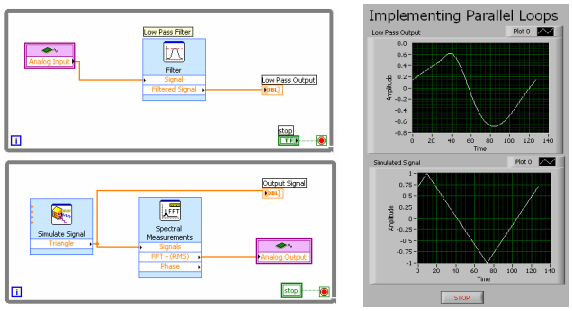


Figure 8. Implementing Parallelism in Graphical Programming

7. CONCLUSION

Embedded systems are pervasive and have moved away from being used only by embedded design experts. Because of the general adoption by "domain" experts in various areas, there is a need to teach the basics of embedded design to non-EE majors. Graphical programming helps address this challenge by providing a simple, easy-to-use interface that provides a platform with a higher level of abstraction allowing the domain experts to focus on their design and hence present a viable alternative for professors and students to learn to design embedded systems. In this paper, we presented an overview of graphical programming systems and explained their benefits. We have also outlined two use cases - ChallengeX and LEGO Mindstorms NXT – the next generation LEGO system that is targeted for the kindergarten and middle-school children that take advantage of the graphical programming approach.

8. REFERENCES

[1] A. Benveniste and P. Le Guernic, "Hybrid Dynamical Systems Theory and the SIGNAL Language," *IEEE Tr. on Automatic Control*, Vol. 35, No. 5, pp. 525-546, May 1990

[2] N. Halbwachs, P. Caspi, P. Raymond, D. Pilaud, "The Synchronous Data Flow Programming Language LUSTRE,"*Proceedings of the IEEE*, Vol. 79, No. 9, 1991, pp. 1305-1319

[3] P. Hilfinger, "A High-Level Language and Silicon Compiler for Digital Signal Processing", Proceedings of the *Custom Integrated Circuits Conference*, IEEE Computer Society Press, Los Alamitos, CA 1985, pp 213-216

[4] The Ptolemy Project http://ptolemy.eecs.berkeley.edu/

[5] Overview of LabVIEW
http://zone.ni.com/devzone/conceptd.nsf/webmain/F34045D2CC5357F486256D3400648C0F?OpenDocument&node=200067_us

[6] ChallengeX tournament official website

http://www.challengex.org/

[7] Teaching Engineering through LEGO mindstorms
http://www.areeonline.org/?id=5193

[8] LEGO Robotics in Engineering
www.asee.org/acPapers/00638_2001.pdf

[9] ROBOLAB @ CEEO website
http://www.ceeo.tufts.edu/robolabatceeo

[10] LEGO Mindstorms NXT System
http://mindstorms.lego.com/

Development and Analysis of Power Behavior for Embedded System Laboratory

Shanq-Jang Ruan
Dept. of Electronic Engineering,
National Taiwan University of Sci. & Tech.
No. 43, Sec. 4, Keelung Rd., Taipei 106, Taiwan
+886-2-27376411
sjruan@mail.ntust.edu.tw

Yi-Ruei Lai
Dept. of Electronic Engineering,
National Taiwan University of Sci. & Tech.
No.43, Sec. 4, Keelung Rd., Taipei 106, Taiwan
+886-2-27333141 ext. 7204
M9302131@mail.ntust.edu.tw

ABSTRACT
With the widespread use of portable systems, power dissipation has become a major concern of battery-driven embedded system design. This paper introduces several fundamental techniques to measure power consumption and analyze the power behavior of the embedded system. Some experiments based on the theories and concepts introduced in this class are provided to help students to build their own power measurement system for embedded systems. The target of this work is to develop an undergraduate laboratory course for teaching power analysis of embedded system to the students in electrical and computer engineering. They will find it very helpful to understand system-level low power design before they enter the field.

Keywords
Power Analysis, Power Behavior, Embedded System.

1. INTRODUCTION
In recent years, embedded systems are already used in a diverse range of products, including PDAs, cell phones, GPS, and smart cards, etc. Because of the tight energy constraints, power aware design is always a popular research topic. Therefore, energy efficiency must be considered in all phases of system design. In addition to constant improvements in battery technology, there are also rapid developments in processors and LCDs in terms of power consumption. Although [4] proposed a low power electronic design course that introduces several fundamental low power circuit design techniques, it is only for graduate students and VLSI hardware design. As the electronic low power techniques are comprehensively applied to more and more embedded system, an understanding of the power behavior is necessary for these students. Although most textbooks on embedded system include some discussions of power-aware design, there is a clear need for complete system-level low-power design skills within undergraduate education in electrical and computer engineering.

Recently, system-level power reduction has become recognized as the most efficient way to achieve low power [1], [2], [3]. Our laboratory course is devised to provide students with exploration of the effect of different system-level design decisions on energy consumption. This integrated laboratory introduces students to learn the basic knowledge and skills of low power design and system power measurement. Then the advanced concepts can help them to apply these techniques to practical electronic system design in terms of power consumption. By leading students to build their own power measurement systems and find out the power behaviors of the real embedded systems, this laboratory teaches students how to measure the power consumption accurately, store the data for analyzing, and the meaning of obtained value.

Even though high-level power aware design does not need to consider tiny power changes, we still take care in abstraction of low-level behaviors for the derivation of a power saving policy. The power behaviors can differentiate the quality and effectiveness of high-level power reduction practices for embedded systems. Furthermore, energy-efficient software design requires developers to understand the characteristics of power consumption and energy impact of software design decisions. Our course focus on software, with emphasis on the power behavior and the difference of energy cost among each kind of algorithm. In addition, the ability to efficiently analyze battery life time under different design choices of application software is an important aid in designing battery-efficient systems.

This course covers experimental techniques in the measurement of power consumption, statistical analysis, errors in measurements, and signal processing. LabVIEW enables data acquisition and control system parameters. We provide some experiments, such as power evaluation of the PDA, power analysis of computer games for handheld computer, and wireless communication on the Internet phone, to introduce students to understand the measurement flow by the computer-based data acquisition (DAQ) system. With a series of experiments, students learn basic experimental techniques to use sensor for various battery-powered embedded systems. Then students can easily build their own power module by writing LabVIEW programs.

In brief, the target of our work is to develop an undergraduate laboratory course for power analysis of embedded system. This course provides students in electrical and computer engineering with the power measurement of embedded system and the basic concepts of low power system design through several experiments including application software, various power modes of PDA and wireless Internet telephony. Students will have the basic knowledge and skills for further research topics in low power embedded system design after taking this course.

2. POWER MEASUREMENT SYSTEM
System-level power/performance optimization is a crucial element of embedded system design. To understand the energy dissipation of the embedded system, we have to build a power measurement system. A handheld data acquisition system was designed and built for use in an undergraduate engineering course [5]. [6] proposed an instrumentation and experimental methods

for mechanical engineering program at the University of the Pacific. Our work is different from that course. We focus on teaching students step by step to build their own measurement system for system level power analysis. Figure 1 shows the structure of the power measurement system that we use in this course. We can observe that Data acquisition (DAQ) system is the key component in the overall flow. In the following subsections, we will detail some LabVIEW based exercises and experiences.

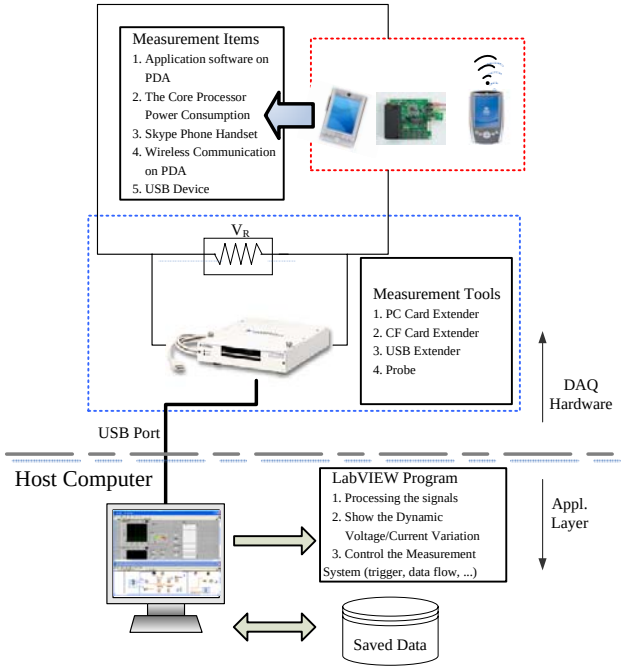


Figure 1. The power measurement flow in our course

2.1 DAQ Hardware Device

With respect to the power consumption measurement of the target system, we used a data acquisition device - National Instruments USB Series DAQ Pad 6016 - in our laboratory. A resistance is placed in series with the current source and then the DAQ device samples the voltage drop across the resistance at a custom defined rate. The acquisition data can be further processed by means of LabVIEW [20]. We will detail the treatment in the next section. It should be noted that the selection of the appropriate resistance is very important for the experiment. If a large resistance is used, it will lead to much voltage drop and crash the target device. On the contrary, if a small resistance is used, the measured value may not be accurate because of the noise. As a result, how to choose the proper resistance plays a significant role in this experiment. The instantaneous power consumption (P_{Inst}) corresponding to each sample and the average power can be determined by the following equations:

$$\boldsymbol{P_{Inst}} = \frac{\boldsymbol{V_R}}{\boldsymbol{R}} \times \boldsymbol{V_{PowerSupply}} \quad (1)$$

$$\boldsymbol{P_{avg.}} = \sum_{\Delta t} \boldsymbol{P_{Inst}} / \boldsymbol{T_{total}}, \quad (2)$$

where the average power consumption ($P_{avg.}$) in this experiment is the time-weighted mean of all the time periods.

The total power consumption (P_{total}) of the device is acquired from the main battery power rail. However, in order to obtain the power composition of the whole system, we need to get the power consumption of each module in the target device by the power breakdown measurements [14]. For example, we measure the current of each power rail of a Dell AXIM X50 PDA to explore the power consumption in different modes. The PDA used in the experiment contains an Intel® PXA27x Processor Family (PXA27x processor). It is a highly integrated system-on-chip and provides several low-power operating modes that can temporarily suspend or power down the core and peripherals to reduce the power consumption. As shown in Fig. 2, the six main power rails of AXIM X50 are CPU_Core (P_{CPU}), 2.5V ($P_{2.5V}$), 3.3V ($P_{3.3V}$), Audio ($P_{Aud\_3.3}$), LCD back light (P_{LCD}) and the rest of power consumption (P_{rest}). We acquire the total power consumption under three conditions: play a WMV file, idle, and system off by measuring the power consumption of each module individually. The total power of the PDA can be determined by:

$$\boldsymbol{P_{total}} = \boldsymbol{P_{CPU}} + \boldsymbol{P_{2.5V}} + \boldsymbol{P_{3.3V}} + \boldsymbol{P_{Aud\_3.3}} + \boldsymbol{P_{LCD}} + \boldsymbol{P_{rest}} \quad (3)$$

With Equation (3), we can illustrate the average power consumption breakdown of each mode such as the idle condition shown in Figure 3. From the experiment, students can investigate the effects of battery lifetime by placing CPU into different modes.

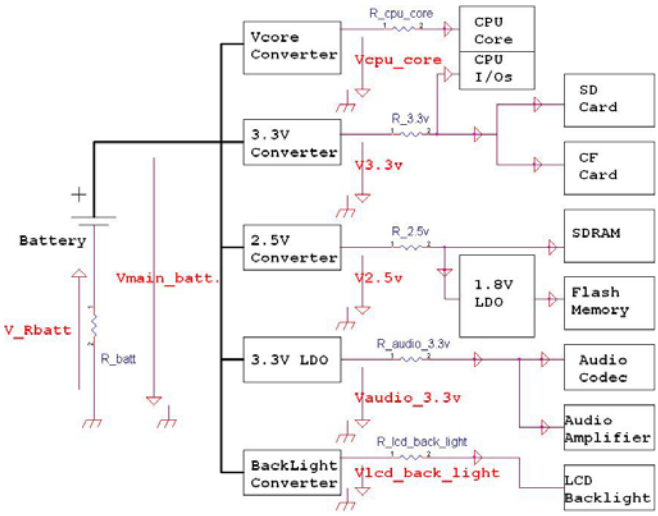


Figure 2. Power rails of Dell AXIM X50 PDA system

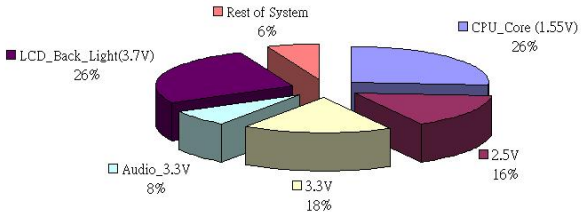


Figure 3. Average power consumption breakdown in Idle mode.

The findings of this study suggest that improvement of power consumption can be achieved by switching CPU into deep sleep mode instead of sleep mode while system is off. With the discussion students can understand how to estimate the power consumption of the embedded system by power-breakdown experiment and extend the battery lifetime by improving the power management algorithms of CPU.

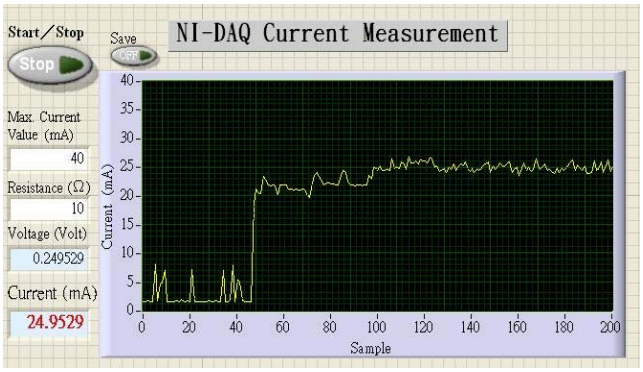


Figure 4. An example of the front panel

2.2 Using DAQ Software

One of the targets in this paper is to incorporate fundamental experiment techniques with modern computer-based data acquisition using LabVIEW. LabVIEW is a GUI-based data acquisition, measurement analysis, and presentation software which is developed from National Instruments Inc. The data acquisition software runs on a personal computer as a power measurement system and directly connects to the PDA through the USB port. Virtual Instruments (VIs) are the program of LabVIEW and which are made up of two parts: the front panel and the block diagram.

In the beginning, students should learn how to write VI's source code in the block diagram and then add needed instruments and indicators in the front panel. The front panel is the user interface of the VI. It looks similar to Figure 4. With the strong library functions in LabVIEW, students are encouraged to further exploit advanced function such as combining C/C++ language with LabVIEW, event structure, signal processing, and data acquisition. The power measurement system can be well established by the following steps:

(1) Setup the DAQ Device and use the test panel in DAQ-mx driver to check whether the device is ready.

(2) Connect the device to the DAQ system with probe and check whether the channel is correct or not.

(3) Issue commands to control DAQ. With the recent versions of LabVIEW, we can easily control and communicate with the DAQ device by using the DAQ Assistant. Otherwise we could use the functions in block diagram with older version.

(4) Write the LabVIEW programs to process and analyze the signals. LabVIEW supports various loop, mathematical, and conditional functions to make a complex program which is similar to a C program.

(5) Store the data and then analyze the record. Make sure the measurement time is long enough to obtain the reasonable average result.

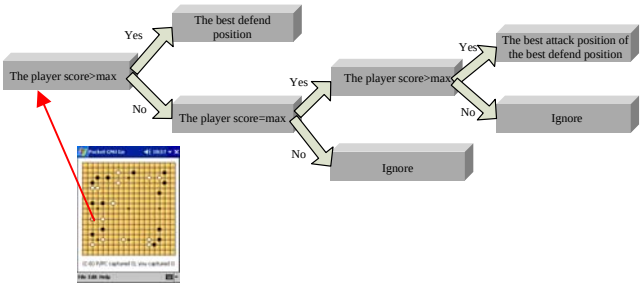


Figure 5. AI decision tree for a typical board game

A key point of the measurement system is the precision and resolution. LabVIEW provides various kinds of timer and random generator for users to perform measurement accurately. For example, if the voltage drop is 0.01V and the precision is only 0.1V. The result of this experiment is meaningless because the lack of precision. The time resolution is another factor that should be noted. The experiment will be subjected to a critical distortion if the resolution is not accurate enough to get the true value. In most cases the time resolution is closely related to the data acquisition rate. In addition, in order to eliminate the personal equation students should try to utilize triggers and conditions to start the measurement, turn on/off the program, and change the state of the device, and so on.

3. USING DAQ SYSTEM TO BUILD POWER MEASUREMENT MODULES

The DAQ hardware and software described in the previous section are used to support our laboratory for power measurement. Furthermore, we provide students the methodology for finding the characteristics of the power consumption and the power behavior of a system. The methodology together with DAQ usage taught in this laboratory course is very helpful for students to engage on future research projects related to low power system design. In this paper, two examples are given: power behavior of computer game algorithms and power analysis of wireless Internet (Skype) telephony.

3.1 Power Behavior of Software Algorithms

Communication and computation are two major sources of power consumption in a mobile computer [3] [15] [16] [19]. According to the applications running on the system, CPU power can be varied from 14% to 52% [22]. The algorithm design for developing application software will result in different computation complexities. This section covers a brief overview of the low power software design and guides students to play a part in the power analysis procedure of software algorithms by means

of a laboratory project – power analysis of computer games algorithms for embedded system [18].

Algorithms for games can be classified into some parts of field, such as graphic operation, AI, memory access, and message process. It should be noted that the same algorithm used in different game types may incur opposite power behavior in the same PDA. This is due to that any particular algorithm is closely tied to how it is used within a game. So we consider the algorithms used in different types of game. For the sake of inspiring students to apply the measurement system introduced in this course, we discuss the motivation for characterizing game energy with students and propose a systematic methodology for their further study.

First of all, for analyzing the power consumption we have to select several games and some of them have been modified to conform to our test conditions. Most computer games used in our experiments are quite simple and concise. On one hand the limited resource and hardware of PDAs can not implement or execute the complex games effectively, and on the other hand simple games have purer power behavior than complex ones. Hence students can easily observe the energy effect by different algorithms running on a PDA.

By means of measuring the total power consumption of the PDA main battery, we can obtain a preliminary analysis of the power behaviors between different game types. In order to further understand the power consumption of games, we undertake the analysis of the CPU current. Depending on different game types the core processor of the PDA consumes power in specific regular behavior. Figure 5 shows the AI decision tree for a typical board game such as the game called "Go" [12]. The game is a kind of board game that expends a great deal of time in evaluating the next position. As can be seen in Figure 6, the power is much higher than wait when the game AI calculating score and comparison the game rules. In addition, the examples Blackout1 and Blackout2 that are designed in different methods are given. As listed in Table 1, the different design approaches lead to much difference in power consumption. The game library provides the designer to develop the games easily but it produces redundant code and results in more energy consumption.

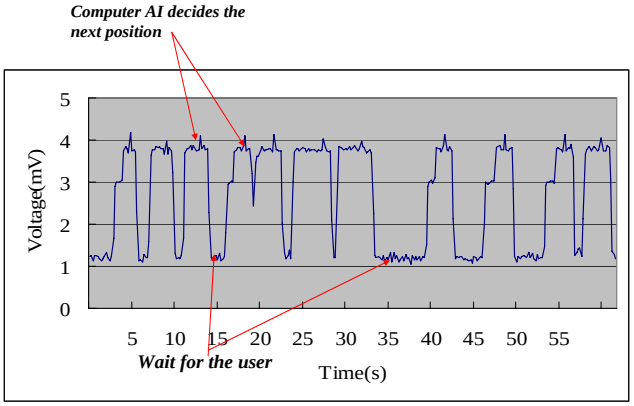


Figure 6. The power consumption behavior of the game of "Go".

We conclude and discuss various opportunities for realizing energy-efficient implementations of game algorithms. We do believe that such investigations are an important first step towards addressing the challenges of energy efficient algorithms for battery-constrained systems to students.

Table 1. Different methods to design the game

Game	Design Method	Avg. CPU power(mW)	Avg. Battery Power(mW)
Blackout1	Game Library	241.00	802.22
Blackou2	WinCE API	152.20	662.80

3.2 Power Analysis of Wireless Communication

The other major source of power consumption of a mobile computer is the wireless communication. To study the energy usage of the wireless network, we make use of a Skype phone which is designed and implemented by our research group for education. The Skype phone employed 2.4 GHz wireless network to extend the work range and avoid the radio interference by hopping technology that is similar to Bluetooth. However, unlike Bluetooth, our wireless Skype phone can also dial out by its hand-set device.

The Skype phone consists of a mixed implementation of software and hardware. It is composed of two parts: the base station and the handset. The system hardware design consists of baseband circuit, RF circuit and MCU circuit. The speech coprocessor used in our design is a low-power, small size and highly integrated (on-chip ADC, DAC and 25 MIPS peak CPU) 8051 micro-controller. We adopted Nordic nRF2401 as the RF transceiver because its output power and frequency channels are easily programmable and especially suit to this laboratory. The third party application on the host computer communicates with the device and transfers the human interface control parameters to Skype by the USB HID driver. The MCU controls the data transmission, enables the RF transceiver, ADC/DAC, and the communication with the host computer and the handset. This system is very convenient for students to understand some power aware factors in the wireless environment. Figure 7 is the prototype of the handset system.

Due to the tight power consumption of the handset, we design the power saving mode for it. When the user presses the OFF button (see Figure 7), the micro-controller switches off all the external current and then enters into sleep mode. Then the micro-controller sets the RF transceiver to receive mode in order to synchronize with the basestation. At last, the micro-controller

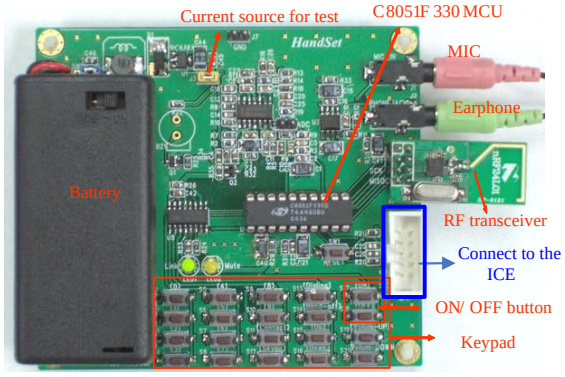


Figure 7. The system prototype of the handset

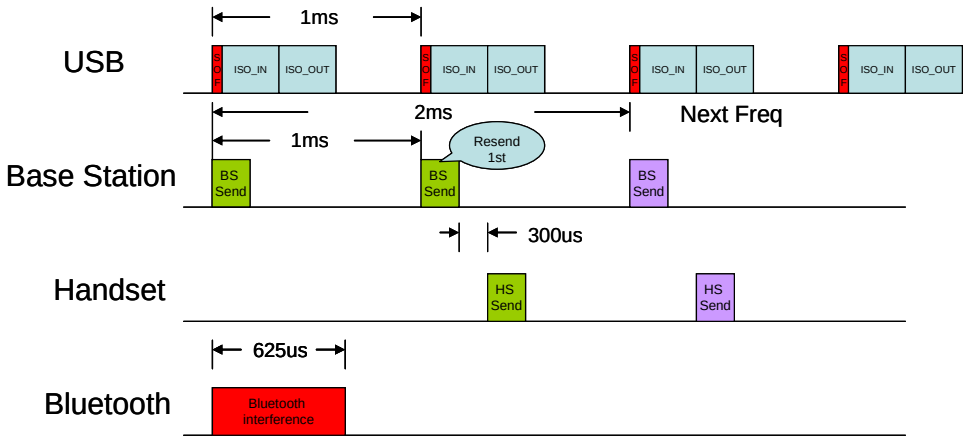


Figure 8. The communication protocols for our wireless Skype system

examines the control field of the received data and decides the next state. In this phase student can observe the micro-controller wakes up periodically by external RC charging and detects if any button has been pressed. In follow-up advanced course students can try to change the firmware of the device by writing their specific 8051 programs. The example program was written by Keil C. As shown in Figure 7, we have designed a port to connect with the 8051 ICE to download the program and a test connection for power measurement.

For energy-efficient design, one of the most important factors is to design the power saving mode. How to find the idle time and turn off the useless units is a major function for power management to switch system into power saving mode. The concept of power management is an important issue and has been employed in most battery-driven embedded system in recent years [7] [8]. Students should learn about how to improve the example program and compare with the Bluetooth products in the market. A great deal of research also discusses the power management concept and applies to various aspects [9] [10] [11]. For the first step student should have the general idea of power management in this experiment.

Wireless communication protocol is another issue worth to discuss. Many researches about wireless network have been proposed for optimizing mobile computer in system level design [3] [13] [17]. [21] presented a new energy efficient and secure communication protocol that exploits the locality of data transfer in distributed sensor network. In our example program, the core technology of the Skype phone employed 2.4GHz wireless network. There is a problem should be noted that 2.4GHz band is the same as industry, science and medicine (ISM) channel shared by a variety of other applications such as wireless network, bluetooth product and so on. Therefore, frequency hopping technology is used to avoid the radio interference in our design. Students can start with a non-hopping version to know how the protocol work, and then compare the differences of power behavior between hopping and non-hopping versions.

The detail of the protocol is shown in Figure 8. The original design of the Skype phone operates in half-duplex mode to avoid interference between transmitter and receiver. We design the hopping mode as Bluetooth with 80 different channels. Due to the switch time of RF transfer/receive is 130us, we set the hopping frequency to 500Hz. A timer is exploited to count time in both sides to synchronize between handset and basestation. Note that we also provide data retransmission mechanism to keep the speech data in certain quality. We recommend students to begin with adjusting the parameters such as the scale of data frame, time interval, and so on. When students are gradually familiar with the communication system, they can try to improve the framework of the example protocol and apply other possible techniques to reduce the power consumption. The speech compression, fault-tolerance, advanced data transmission mechanism or other functions about power reduction can be the project topics.

4. Conclusion

Power dissipation has become an important consideration in modern embedded system design. The primary objective of the laboratory is to deliver the understanding of system-level power analysis to undergraduate students in electrical and computer engineering, and eventually to make a promotion to low power system design experts. At the first, this paper conducts a number of measurements about power consumption. According to the measurement results, students can observe the power behavior of an embedded system and find out the key factors of the power consumption. Students need to pay attention to some factors, such as the precision and resolution, in the measurement procedure. Then the students have a chance to experience low power schemes in both software and hardware. We provide some experiments for example to help students with understanding the methodological approach to analyze power behavior. For the power behavior of software algorithms, we can rely on analyzing of the power behavior of the core processor to realize how the algorithms affect the energy. In addition, different algorithm designs of games lead to very different power consumptions. For the analysis of wireless communication, students can understand the effect of each power-aware factor and try to improve the battery life of the mobile handset.

We developed an undergraduate laboratory course for power analysis of embedded system. This course provides students in electrical and computer engineering with the power measurement of embedded system and the basic concepts of low power system design through several experiments

5. REFERENCES

[1] N. Chang, H. Lim, K. Lee, Y. Cho, H. G. Lee and H. Shim, "Graduate Class For System-Level Low-Power Design". *In Proceedings of the MSE*, pp. 31-32, 2005.

[2] Y. W. Bai and Y. C. Lin, "Measurement and Improvement of Power Consumption for Portable Computers," *IEEE International Symposium on Consumer Electronics – (ISCE2005)*, Macau China, pp. 122-127, June 14-16, 2005.

[3] J. Warren, T. Martin, A. Smailagic, D. P. Sieworek, "System Design Approach To Power Aware Mobile Computers," *IEEE Computer Society Annual Symposium on VLSI (ISVLSI'03)*, p. 101-106, February 2003.

[4] J. S. Yuan, Senior Member, IEEE, and Jia Di, "Teaching Low-Power Electronic Design in Electrical and Computer Engineering," *IEEE transactions on education*, vol. 48, no. 1, pp.169-182, February 2005.

[5] G. Mason, "A Handheld Data Acquisition System for Use in an Undergraduate Data Acquisition Course," *IEEE transactions on education*, vol. 45, no. 4, pp. 388-393, November 2002.

[6] C. Lee, "LabVIEW Based Instrumentation and Experimental Methods Course," *Proceedings of the ASEE Annual Conference*, July 2001.

[7] T. Simunic, G. D. Micheli, and L. Benini, "Event-Driven Power Management of Portable Systems," *Proceedings of the 12th international symposium on System synthesis*, p.18, November 01-04, 1999

[8] Y. Chen, N. Smavatkul and S. Emeott, "Power Management for VoIP over IEEE 802.11 WLAN," *IEEE Wireless Communications and Networking Conference*, Vol3, pp.1648-1653, March 2004.

[9] C. Schurgers, V. Raghunathan, and MB Srivastava, "Power Management for Energy Aware communication systems," *ACM Transactions on Embedded Computing Systems*, August 2003.

[10] S. Banerjee and S. Khuller, "A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks," *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, vol. 2, 1028-1037, April 2001.

[11] C. Chen, "Energy Efficient Routing for Clustered Wireless Sensors Network," *In Proceedings of the 29th Annual IEEE Conference on Industrial Electronics*, Vol. 2, pp. 1437–1440, November 2003.

[12] Vieka Technology Inc, Pocket GNU Go is a GNU Go player for Windows mobile devices, including Pocket PC and Smartphone. http://www.vieka.com/gnugo/

[13] Havinga PJM, Smit GJM, "Low Power System Design Techniques for Mobile Computers", internal report University. of Twente, 1997.

[14] M. A. Viredaz, D.A. Wallach, "Power evaluation of a handheld computer," in *Proceeding of IEEE Computer Society*, Vol. 23, Issue 1, pp.66--74, 2003

[15] K. Farkas, J. Flinn, G. Back, D. Grunwald, and J. Anderson, "Quantifying the Energy Consumption of a Pocket Computer and a Java Virtual Machine," *In Proceedings of the ACM SIGMETRICS '00 International Conference on Measurement and Modeling of Computer Systems*, Santa Clara, CA, June 2000.

[16] G. Chen, B. T. Kang, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, R. Chandramouli, "Studying Energy Trade Offs in Offloading Computation/Compilation in Java-Enabled Mobile Devices," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 9, pp. 795-809, Sept., 2004.

[17] A. Smailagic, M Ettus, "System design and power optimization for mobile computers," *VLSI on Annual Symposium, IEEE Computer Society ISVLSI 2002*.

[18] Y. R. Lai and S. J. Ruan, "Power Analysis of Computer Game Algorithms for Handheld Embedded System," *In Proceeding of 1st IEEE International Conference on Industrial Electronics and Applications (ICIEA 2006)*, Singapore May 24-26 2006.

[19] D. Bertozzi, A. Raghunathan, L. Benini, and S. Ravi, "Transport Protocol Optimization for Energy Efficient Wireless Embedded Systems," *IEEE Design, Automation and Test in Europe Conference and Exhibition, 2003*, pp. 706-711.

[20] National Instruments Corp. http://www.ni.com.

[21] H. Saputra, N. Vijaykrishnan, M. Kandemir, R. Brooks, MJ Irwin, "Exploiting Value Locality for Secure-Energy Aware Communication," *In Proceedings of SiPS'03*, pp. 116-121, Aug. 2003.

[22] A. Mahesri and V. Vardhan, "Power Consumption Breakdown on a Modern Laptop," *Workshop on Power Aware Computing Systems, 37th IEEE International Symposium on Microarchitecture*, Dec. 2004.

Toward HW/SW Integration: Networked Embedded System Design

Chi-Sheng Shih
Graduate Institute of Networking and Multimedia
National Taiwan University
cshih@csie.ntu.edu.tw

Shiao-Li Tsao
Department of Computer Science
National Chiao Tung University
sltsao@csie.nctu.edu.tw

Yeh-Ching Chung
Department of Computer Science
National Tsing Hua University
ychung@cs.nthu.edu.tw

Shyh-In Hwang
Department of Computer Science and Engineering
Yuan Ze University
shyhin@saturn.yzu.edu.tw

***Abstract* — Traditional Computer Science curricula focuse on the training for logic reasoning and programming skills. System integration is often not covered in computer science curricula. As the embedded platforms migrate from 8-bit microprocessors to 32-bit microprocessors, the engineers require different skills to design modern embedded systems. The Computer Science faculties at several universities in Taiwan have collaborated to design a new course to meet such needs. In this article, we report the design rationale and current status of this course.**

I. INTRODUCTION

The embedded industries in Taiwan and other countries have been a booming industry session in the last few years. Nevertheless, the needs of embedded system engineers greatly increase. The industry have complained that neither computer science curricula nor electrical engineering curricula provide sufficient skills for junior engineers to design networked embedded systems.

Current computer science curricula focus on the training for logic reasoning and programming skills. Examples are algorithm design, programming languages, computer architecture, micro-electronics, logic design, and object-oriented design. Such curricula train the students to design the software systems to be more efficient and effective. The students have learned how the software and hardware components in a computer interact with each other. For instance, the students learn how the file system stores and retrieves data from the storage device on behalf of the user programs and kernel, and how the memory management unit in operating system allocates new memory regions for the user programs and collects the unused memory regions. However, it has been ignored to teach the Computer Science students how a computer-based system or an embedded system interacts with the real world. Specifically, how the software program and hardware collaborate to interact with the users and environment.

Starting in 2004, we, the Computer Science faculties with several universities in Taiwan, collaborate to design a new course to provide the skills for designing networked embedded systems. The missing ingredient in current curricula is the connection between low level hardware design and high level software development. Our students have the knowledge for VLSI design and high performance computation. But, they are lack of the knowledge to put them to work together. The purpose of this course is to teach the students how to design a system in which the hardware and software collaborate to complete the system's mission. In particular, the trend of embedded platforms has been migrated from 8-bit micro-processors (or micro-controllers) to computationally powerful micro-processors (or micro-controllers), or special designed processors. Examples are Intel Xscale and TI OMAP processors. Hence, multi-thread and multi-task programming are now suitable for embedded systems. The course has been offered at more than four universities in Taiwan and received the requests for the course materials from other countries and regions in Asia. In this paper, we report our curriculum design and reflections from students and faculty.

The target students for this class are Computer Science major seniors and first-year graduate students. The students shall have taken the fundamental courses for operating systems, computer architecture, C/C++ programming, and x86 Assembly programming. The course materials are available at our course web site [1].

The remainder of the paper is organized as following. Section II presents the efforts at different universities to promote embedded system education. Section III presents the curriculum design including in-class lectures and hands-on laboratories. Section IV presents the reflections from faculty and students, and the lessons we learned. Section V summarizes the paper.

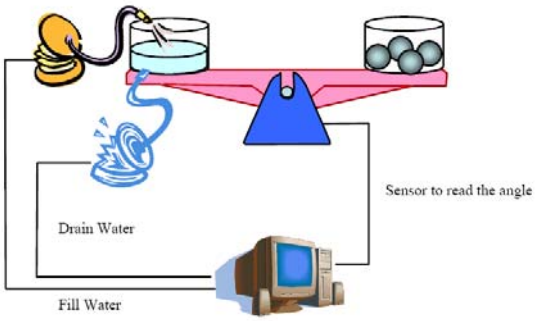


Fig. 1. Water See-Saw: An example of real-time monitor and control system

II. BACKGROUND

Recent years have seen much discussion about the appropriate curricula for embedded systems design. Many universities have offered courses related to embedded systems design [2]. We can categorize the courses in three categories: hardware oriented, software oriented, and HW/SW integration design.

Many of the embedded system design courses are hardware oriented and offered in Electrical Engineering department. Most of them cover topics in System on Chip (SoC) design, FPGA, and VLSI aspects of embedded hardware design. The courses do not cover the software development for embedded systems and are not suitable for Computer Science major students.

Several of the embedded systems design courses focus on software design such as [3], [4], and [5]. Dr. Muppala's course [3] focuses on programming on resource limited platform such as Windows CE and μC/OS II. The students learn the characteristics for embedded real-time operating systems, embedded software development process, and software development on such platforms. Dr. Evans's course [4] focuses on formal modeling methods, simulating systems, and system-level design tools. Dr. Caccamo's course [5] focuses on sensor sampling, serial communication, and hands-on laboratory. The courses provide the fundamental concepts for embedded software development, real-time systems, and real-time scheduling. However, most of them do not cover the materials for the integration of hardware and software components in the systems. The integration needs to take into account the different characteristics of hardware and software components so as to avoid the jitters and drifts, which are critical for embedded real-time systems. In addition, networking communication and power consumption are not covered in the courses.

III. CURRICULUM DESIGN

The course is designed with helping the students to design networked embedded systems in mind. Hence, we design the class to have equal weights on in-class lectures and hands-on laboratories. In every week, the students will have one two hours in-class lecture and one two hours hands-on laboratory. In the lectures, the students learn the general concepts and theoretic background for the networked embedded systems; in the hands-on labs, the students apply the theories learned in the classroom to the pre-designed lab modules.

The course materials including lectures and laboratories consist of four modules: *embedded real-time systems*, *networking*, *power management*, and *embedded real-time programming*. The four modules are closely related and one sample application is used to illustrate the lecture materials. The sample application we used is a water see-saw, which is a simple real-time monitor and control system [5]. Figure 1 illustrates the water see-saw. In a water see-saw, containers are mounted at the two ends of a level beam: one is a water tank and the other keeps marble balls. The micro-controller commands the pumps to pump in and drain out water from the water tank. In the middle of the beam, an angle encoder, which is connected to a micro-controller, is installed to read the level angle of the beam. By periodically sampling the angle, a real-time monitor and control program executed on the micro-controller computes the amount of water to be pumped in or drained out, and controls the pumps.

In the lab modules, the students are guided to complete another real-time embedded application. The application we used is an Automatic Vehicle Navigation (AVN) System for vehicles in Intelligent Transportation Systems (ITSs). In ITSs, we assume that there are no traffic lights on the street. All the vehicles have to negotiate with each other or the traffic controller for the right of way. On busy intersections, traffic controller pole will be installed to avoid frequently pair-wise communication. The traffic controller is responsible for monitoring the traffic, arbitrating the right of way, and clear the way for emergency vehicle.

The AVN system on each vehicle is responsible for driving the vehicle to the destination with the following

constraints:
1) Safety: Safety always has the highest priority for the vehicle. The navigation system should avoid collision with other vehicles on the street and keep the vehicle on the right lane.
2) Efficiency: the second priority of the AVN system is to guide the vehicle to the destination as soon as possible. In this course, we are not concerned with route planning and, hence, all the routes are pre-determined. However, the vehicle should pass the intersections as soon as possible when it is safe to do so.
3) Low Power Consumption: The last constraint is the power management. The AVN system should adjust its clock frequency (and processor voltage) when the first two constraints are met.

A. In-Class Lectures

At the beginning of the semester, we spend four hours to introduce real-time embedded systems. Most students are familiar with the general purpose computer systems including workstations, personal computers and high performance computation systems. Not all of them have the correct or same definition of embedded systems and real-time systems although lots of students use and own embedded systems. In this lecture, we focus on the difference between real-time embedded systems and general purpose computer systems. Several example embedded systems are used to formally define the characteristics and features for real-time embedded systems, e.g., anti-lock brake systems and avionic systems. At the end of the introduction, the students should be able to tell if a system is an embedded system and whether a system is a non-real-time system, soft real-time system, or a hard real-time system.

Embedded Real-Time Systems Module: The first module is *embedded real-tim*e *systems*. In this module, we cover three topics: embedded real-time operating systems, introduction to real-time scheduling algorithms, and embedded software development process.

The purpose of the first topic is not to detail the features of one or several embedded real-time operating systems. The goals are to illustrate the popular system architectures for embedded real-time operating systems and to focus on how to select the suitable operating systems for different embedded real-time systems. In addition, the general concept of operating systems should be covered on the introduction-level course for operating systems. Hence, we will not cover that part too. We introduce three embedded real-time operating systems: eCos [6], [7], μC/OS II [8], and RTLinux[9]/RTAI [10]. We select the three operating systems because they have different architecture designs and different levels for real-time supports. Some of them such as RTLinux and RTAI are designed to meet hard real-time constraint; some of them such as eCos and μC/OS II are better suitable for limited resource embedded systems. We focus on the difference of their performance metric, footprint, memory management supports, and real-time supports.

The second topic covers the two classes of real-time scheduling algorithms: dynamic priority scheduling algorithm and static priority scheduling algorithms. The purpose of this topic is to illustrate the concepts of priority-driven scheduling algorithms. Hence, we discuss the general concepts of the two classes of algorithms and avoid teaching all different kind of real-time scheduling algorithms at this point, which should be covered by another course. The third topic covers the cross-development process of embedded software. The hands-on laboratory will guide the student to write their '*Hello Embedded World*' program for the development board and upload their program to the board.

Upon the completion of this module, we expect the students to learn the characteristics of embedded real-time systems, general concepts of real-time scheduling algorithms, and how to select a suitable operating system while designing an embedded real-time system. In addition, the students now have the same mindsets for embedded real-time systems and are ready to dig into other issues for designing networked embedded real-time systems.

Networking Module: The second module focuses on the various networking protocols including serial communication and wireless networks for embedded systems. Specifically, it covers the serial communication, personal area networks (PAN), wireless local area networks (WLAN), and wireless wild area networks (WAN). For serial communication, we cover I^2C, CAN bus architecture [11], and universal asynchronous receiver and transmitter (UART). In particular, we focus on UART because the students will use RS-232 serial communication in the laboratory. For wireless network protocols, we focus on PAN and WLAN such as IEEE 802.11, IEEE 802.16, HiperLAN, Zigbee, Bluetooth, and HomeRF.

In this module, several networking protocols are covered. Note that the purpose of this module is not to have a sound knowledge for the network protocols. We focus on the timing and performance issues for the communication protocols. For instance, the students will learn the coding policy for serial communication, which policies are clock friendly, how to write a low level program to receive and send data over serial communication protocols. We spend several hours on serial communication because, thus far, serial communication is the most reliable communication

protocol for embedded systems and has low cost. In addition, several modern communication protocols such as USB are designed base on the serial communication protocol. While illustrating the modern wireless communication protocols, we will focus on the power consumption and their propagation delay. At the end of this module, we expect the students to know how to select a suitable communication protocol when power consumption and timing issue play important roles in the design specification.

Power Management: The third module focuses on the power consumption issues for embedded real-time systems. Specifically, the course materials cover lower embedded processor design, power-aware scheduling, and low power kernel design.

Embedded Real-time Programming: The fourth module covers most of the programming materials in the class. In this module, a water see-saw, which is a real-time monitor and control application, is used to illustrate all the theories and programming practices in this module.

The module starts with correcting a simple but bogus monitor and control program. A sample pseudo-code, listed in Algorithm 1, is shown to the students to illustrate how to monitor and control, using a simple loop. To most of the CS-major students, the pseudo-code seems to work well to control the water see-saw. However, several lines in the example may cause jitters and drifts during the execution and lead to unpredictable performance. For instance, when the program is preempted during its execution at Line 1 and 5, it may cause drift, and jitter may occur at Line 9. In this module, the students learn how to correct the program so there is no jitter and drift for the program.

Algorithm 1 Monitor and Control Using Single Task

```
1:  current_time = read_clock()
2:  if START_TIME - current_time < 10 msec then
3:     //report too late and exit
4:  else
5:     sleep(START_TIME - current_time)
6:     loop
7:        current_time = read_clock()
8:        wake_up_time = current_time + 20 msec
9:        // read sensor data from the device, it takes <<
          20 msec
10:       // do work, assuming that it takes << 20 msec
11:       current_time = read_clock()
12:       // send control data to the device
13:       sleep(wake_up_time - current_time)
```

The first part of this module illustrates how to write a real-time program to control an external device and periodically monitor the sensor data. We start with the POSIX-RT ([12], [13], [14]) standard as the fundamental skills such as timer signals, signal handlers, and data acquisition, to program periodic real-time tasks. In particular, we focus the reentrant functions for signal handlers. Although the students learn how to write signal handlers in other courses, most of them are not aware that there could be multiple instances executing at the same time.

The second part of this module is related to real-time scheduling theories and resource sharing in embedded real-time systems. This part addresses how to conduct the schedulability analysis during the design time. The course materials cover General Rate Monotonic Scheduling (GRMS) algorithm, schedulability analysis including utilization bound approach and exact analysis, pre-period deadlines, high priority I/O, and interrupts in GRMS. The learning path starts from assuming that all the tasks in the systems are periodic and independent and ends at the case that the tasks may share resources such as I/O and memory. The water see-saw example is again used to illustrate the materials.

The last topic is the multi-thread programming. The simple loop program shown in Algorithm 1 is revised to use timers in the first part, and is further revised to a multi-thread program. In the last version, one thread is designed to conduct the computation and the other is designed to read the sensor data and output the control command. The last version provides a jitter-free and drift-free program for monitoring and control. In addition, the shared memory and message box mechanism for inter-process communication are illustrated in this part. The covered topics of the in-class lectures are listed in Table I.

B. Hands-on Laboratory

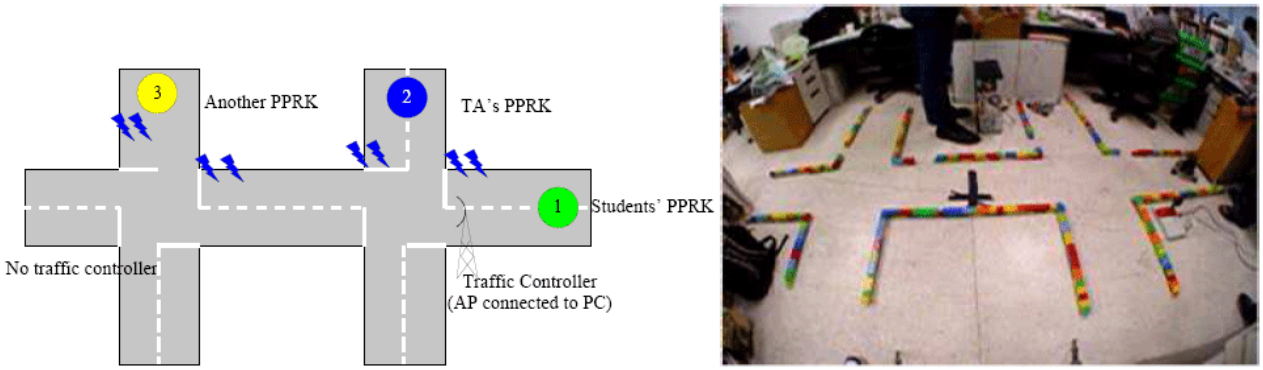
The hands-on laboratories are as important as the in-class lectures in this course. We plan an eighteen weeks hands-on laboratory program to complete an AVN system. In every week, the students complete a part of the AVN systems and demo their projects at the end of the semester. Figure 2 shows the scenario for the lab project. On the streets, there are two kinds of intersection: one without traffic controller and one with traffic controller. When there is not much traffic on the intersection, no traffic controller is installed and the AVN systems negotiate with each other via ad hoc wireless network. The traffic controller, which arbitrates the right of way for all the nearby vehicles, is installed on busy intersections to reduce communication overhead.

TABLE I
LECTURE SCHEDULE

Unit	Covered Topics
Unit 1	Introduction: • Networked embedded system design trend • Networked system on chip • Networked embedded system examples • Real-Time issues • Power consumption issue • Security issue
Unit 2	Real-Time Operating Systems: • eCos and RTLinux • Real-Time Scheduling
Unit 3	Real-Time Programming: • Real-time periodic tasks in POSIX RT • Real-time periodic tasks using timer signals and signal handlers. • Signals and data acquisition • General Rate Monotonic Scheduling (GRMS) • Schedulability Analysis: utilization bound and exact analysis • Pre-period deadlines, high priority I/O, and interrupts in GRMS
Unit 4	Power Management and Low Power Design of an Embedded OS: • Power management and low power support for an embedded processor • EOS design to support power management • Energy-aware EOS scheduling, Low power kernel design
Unit 5	Low power I/O and device driver: • Overview of power consumption of a Networked SoC system • Device driver design and implementation • Low power I/O and device driver design
Unit 6	Networks for Embedded Systems: • I2C, CAN, SHARC, UART, Ethernet • IEEE 802.11, IEEE 802.16, and HiperLAN • Zigbee, Bluetooth, and HomeRF
Unit 7	Real-Time Networking: • Real-Time Networking • Low power technologies for wire-line and wireless network protocols • Power management and power control of radio access systems and protocol • Low power design technologies for application programs • Design example - A case study based on Wi-Fi phone
Unit 8	Security in Wireless Network Systems: • Encryption, Authentication • Key management, Design example - A case study

Lab Setting: Each set of lab equipments consist of (1) Palm Pilot Robot Kit (PPRK) [15], (2) Intel Xscale Development board, (3) power supply, (4) digital data acquisition card, and (5) an Intel-based host computer. Legos are used to construct the street curb. PPRK has a unique holonomic drive system and a roomy deck to mount PDAs or single board computers. It has a BrainStem module for general purpose use whether running code stand-alone, connected to a host computer, or enabling reflexive actions. The BrainStem module has one 40 MHz RISC processor and RS-232 serial port. The PPRK can be controlled by console application or through C, C++, and JAVA. We select PPRK as the target system for several reasons. First of all, the programming interface is easy to deploy. The development kit provides C APIs to control the motors and read the data from the sensors. The three infrared sensors mounted on the robot can sense the distance between the robot and other objects. It allows the robot to detect nearby robot or the street curb. Second, PPRK moves at low speed, which reduces the choice of breaking the robots.

PPRK is designed to work with Palm-based and Windows-CE based PDA. The PDA acts as the brain of the robot. In our lab setting, rather than PDAs, we use Intel Xscale development board as the brain. The development board has an Intel PXA-255 400 MHz microprocessor, IEEE 802.11 adapter, comprehensive



(a) Scenario (b) Students' Final Demo

Fig. 3. Final Project

I/O interfaces, VGA Controller (Tvia 5202G graphic chip), resolution up to XGA (1024 x 768) is supported along with 24-bit graphics performance. Embedded Linux is installed on the platform. The board connects to the PPRK via RS-232. The host computer is an Intel-based PC on which Fedora Core Linux is installed.

Lab Modules*:* The lab modules are closely related to the in-class lectures. It consists of the modules for embedded real-time programming, embedded network programming, and low power network protocol. In each lab, the students are given an instruction sheet and should submit their lab reports at the end of the lab. The students can practice the lab individually or in a group for up to four students.

TABLE II

LAB SESSION SCHEDULE

Lab	Session
Lab 0	Meet the friends and form the group.
Lab 1	Prepare for the embedded programming
Lab 2	Real-Time Monitoring
Lab 3	Real-Time Control
Lab 4	Sensing and Control
Lab 5	Real-Time Scheduling
Lab 6	Multi-Thread for Sensing and Control
Lab 7-1	Power Saving Technologies for Embedded Processor and OS (I)
Lab 7-2	Power Saving Technologies for Embedded Processor and OS (II)
Lab 8-1	Networking via 802.11 infrastructure mode
Lab 8-2	Networking via 802.11 infrastructure mode
Lab 9	Networking via 802.11 ad hoc mode
Lab 10-1	Low power network protocol design (I)
Lab 10-2	Low power network protocol design (II)
Lab 11-1	Algorithms for the car to yield (I)
Lab 11-2	Algorithms for the car to yield (II)

Table II lists the lab modules. The first six labs are designed for the students to program a real-time monitor and control system. The students learn how to read the sensor data and convert the data into meaningful information. They also learn how to control the motors to move the robot. Thus far, the students can now program the robot to move along the designated route and, according to certain rules, to avoid bumping into other objects. For instance, in Lab 6, the students program their robots to go after but not to bump into TA's robot, which is controlled by TA from the console and may move toward any direction at different speed.

The second part is related to the power management on embedded processors. In Lab 7, the students are guided to learn how to change the voltage level on the voltage scaling processors and measure the change the voltage on the processor. The third part of the lab modules is related to wireless communication, which allows the robot to communicate with each other on the intersection and to negotiate with the traffic controller if exists. When the robot is close to an intersection, it starts its communication program. The students practice the lab for 802.11 infrastructure mode and ad hoc mode.

In the last part, the students design the yield protocols and prepare for the final demo. In the final demo, the students' robot starts at the start line, follows a designated route, and stops at the destination line. The scenario is shown in Figure 3(a). On the route, there are two intersections: one of them has a traffic controller and the other one does not. When traffic controller is presented, the robot has to obey traffic controller's commands. When there is no traffic controller, higher priority robot such as emergency vehicle has the right of way. The picture shown in Figure 3(b) was taken during the final demo.

IV. REFLECTION

Prof. Shiao-Li Tsao of National Chiao Tung University (NCTU) adopted the course and lab materials developed by this project for teaching networked embedded system design in the department of computer science. Since there are already courses which introduce wireless network protocols, embedded

operating system, and real-time system in NCTU, the revised course focuses on the knowledge, technologies, and hands-on practices for system-level design and system integration. The syllabus tailoring from the course materials developed by this project becomes.

- Introduction to Networked Embedded System (Lab 1)
- Basics of Real-Time Control (Lab 2)
- Embedded CPU and Embedded Hardware Design
- Bootloader and BSP Design (Lab 3)
- Introduction to Embedded Operating System
- Device Drivers, specifically on network interface driver and protocols stack design (Lab 4)
- Integration, Testing, Verification and Validation Technologies for a Networked Embedded System. (Term Project)

Different from the original plan proposed by the project which suggests 2-hour in-class lecture and 2 hour hands-on labs, the networked embedded system design course in NCTU has three hours lecture and all hands-on labs are homework for students. Therefore, only four hands-on labs among the total 11 lab modules developed by the project are adopted in order to reduce the loads for students. The four labs are:

- Understand and practices on the Intel PXA-255-based embedded system board
- Control of PPRK robot car
- A simple bootloader development
- Porting camera and WLAN interface drivers and protocol stacks

The final project integrates the above four hands-on labs together to develop a robot car navigation system which can be controlled from a remote console over wireless networks. Students (two persons per team) are requested to develop their own robot car navigation system together with a remote controller which is running on the remote PC. Students' robot car navigation systems are placed in room with a maze and the students controlling the car are in the room next to the room with the maze. Students have to use the information such as distances to walls which are sampled by infrared sensors, and live camera images to control the car. These distance and image information are transferred over the WLAN in real-time. The students are very interested in the hands-on labs and final projects. It is believe that a step-by-step arrangement for the hands-on labs and goal-driven final project greatly help the students to concentrate on their homework and practices. This philosophy for designing hands-on labs is especially helpful for teaching an embedded system course.

One important lesson we have learned was that the TAs played an extremely important role to the success of this course. In this course, the hands-on lab and in-class lectures are equally important. To guide the students to practice the lab, the TAs should be well-trained in embedded systems design. An experienced TA can answer the most questions from the students and assist the students to debug their designs in the lab. Otherwise, the students will be very frustrated and may drop the class.

V. SUMMARY

Traditional computer science curricula are concerned with the knowledge and training for software development; traditional electrical engineering curricula are concerned with the hardware design. To design embedded systems, it requires the skills to integrate software and hardware components in the system to accomplish the work. We design a new course to be offered in several universities in Taiwan to provide such trainings. The course provides a complete training for students to understand the design of a networked embedded system from a system point of view. Its materials cover several topics such as real-time systems, embedded operating systems, device driver and wireless protocols which might have some overlaps with other existing courses. The course also provides a tailoring guideline for lecturers who want to use the course materials in their universities. As the complexity of embedded system continues to increase, we will continue to review the course materials. For instance, multi-thread real-time programming and multi-thread debugging should be added in the near future to meet the evolving needs for embedded system design.

ACKNOWLEDGEMENT

We would like to thank for the generous financial supports from the Ministry of Education at Taiwan and Microsoft Research Asia. Parts of the course materials are inspired by and collected from Prof. Lui Sha's classes at the University of Illinois. We thank for his generous supports. We are also extremely grateful to the graduate students for previewing the course materials, providing valuable comments, and practicing the lab materials before the course was offered. In particular, they are Yi-An Chen, Chen-Min Lien, and Chuan-Yue Yang at National Taiwan University. In addition, Mr. E-Cheng Cheng, Jin Chang Chou, Pang-Hsiang Lo and Miss Ya-Lian Cheng at National Chiao Tung University in Taiwan also contribute their valuable time to this project.

REFERENCES

[1] "Embedded software for networked soc systems," at http://sslab.cs.nthu.edu.tw/course/ESW94NSOC/, July 2006.

[2] A. Sangiovanni-Vincentelli and Alessandro Pinto, "Embedded system education: a new paradigm for engineering schools?" *SIGBED Rev.*, vol. 2, no. 4, pp. 5–14, 2005.

[3] J. Muppala, "Experience with an embedded systems software course," in *Proceedings of the 2005 Workshop on Embedded Systems Education*, September 22 2005.

[4] B. L. Evans, "EE382C-9 embedded software systems," at http://www.ece.utexas.edu/~bevans/courses/ee382c/, last accessed at August 2006.

[5] M. Caccamo, "CS431 embedded systems architecture and software," at http://www.cs.uiuc.edu/graduate/courses.php?course=cs431, last accessed at August 2006.

[6] "eCos 2.0 documentation," at http://ecos.sourceware.org/docs-2. 0/, July 2006.

[7] *Embedded Software Development with eCos*, 1st ed., ser. Bruce Perens' Open Source Series. Prentice Hall, November 2002.

[8] *MicroC/OS-II*: *The Real Time Kernel*, 2nd ed. CMP Books, June 2002.

[9] V.Yodaiken, "The rtlinux manifesto," in *Proc. of The 5th Linux Expo*, Raleigh, NC, March 1999.

[10] P. Mantegazza, E. L. Dozio, and S. Papacharalambous, "RTAI: Real time application interface," *Linux J.*, vol. 2000, no. 72es, 2000.

[11] O. M. Group, "CAN specification version 2.0," at http://www.omg.org/, July 2003.

[12] IEEE/ANSI Std 1003.1: Information Technology-(POSIX)-Part 1: System Application: Program Interface (API) [C Language], includes (1003.1a, 1003.1b, and 1003.1c), 1996.

[13] 1003.1d Information Technology-(POSIX)-Part 1: System Application Program Interface (API)-Amendment: Additional Real-time Extensions, 1999.

[14] 1003.1j-2000: Information Technology-(POSIX)-Advanced Real-time Extensions, 1999.

[15] "Palm pilot robot project," at http://www.cs.cmu.edu/~ pprk, last accessed at July 2006.

Hardware Platform Design Decisions in Embedded Systems - A Systematic Teaching Approach

Falk Salewski
Embedded Software Laboratory
RWTH Aachen University
Ahornstr. 55
52074 Aachen, Germany
salewski@informatik.rwth-aachen.de

Stefan Kowalewski
Embedded Software Laboratory
RWTH Aachen University
Ahornstr. 55
52074 Aachen, Germany
kowalewski@informatik.rwth-aachen.de

ABSTRACT

Designers of embedded systems can choose between a large variety of different hardware platforms. The question often arising is which hardware platform is suited best for a certain application. This decision is usually made by an expert in industry being familiar with a variety of hardware platforms. Therefore, it is of major interest how this expert knowledge and the skills necessary for such a selection process can be taught to students. In this paper a systematic hardware platform selection process based on hardware attributes is presented. Moreover, it is proposed how to integrate this approach in the embedded systems education.

Categories and Subject Descriptors

K.3.2 [**COMPUTERS AND EDUCATION**]: Computer and Information Science Education

Keywords

Embedded System Education, Hardware Platform Selection, CPU vs. PLD, MCU vs. FPGA, Microcontroller, Reconfigurable Hardware

1. INTRODUCTION

Embedded systems integrate hardware and software components and require developers with skills in both subjects. Hardware skills should include the capability of adopting a systematic approach to making design decisions in choosing between various hardware platforms for a given embedded systems application. These decisions are nontrivial because there is a large number of quite different hardware platforms available. These are CPU based systems such as microcontrollers (MCU) and digital signal processors (DSP), as well as Programmable Logic Devices (PLD)[1] as Complex Programmable Logic Devices (CPLD) and Field Programmable Gate Arrays (FPGA). For some applications also combinations of different discrete devices or an integration of a microprocessor core in an FPGA is preferable. Moreover, the available hardware devices are constantly changing over time. Thus, embedded systems education needs to include, in addition to knowledge of basic functional properties of current hardware platforms, the capability to systematically analyze both functional and non-functional features of hardware devices and the skills to map these to the requirements of a given specified application to be designed.

One way to impart some of these experiences to students is by conducting exercises and lab courses dealing with different hardware platforms and architectures, as for example done in [2] or [8]. This hands on learning is very useful to teach students the fundamental properties of different platforms, mostly with respect to functionality. However, we experienced in our own embedded lab course, presented in [8], that not all influencing factors can be taught this way. Additionally, a comparing overview of different hardware platforms is needed. Promising approaches can be found for example in [9] or [10]. Both give a good comparative introduction into CPU and PLD based embedded systems. Furthermore, an example illustrating the trade off among different hardware architectures is given in [10]. However, a systematic approach for hardware platform selection is missing. In this paper we are going to propose such an systematic approach for hardware platform selection and discuss how it could be used successfully in embedded system education.

The remainder of this paper is organized as follows. In section 2 the selection process is analyzed in order to gain the information necessary for further work. The steps of a proposed hardware platform selection process are presented in section 3. An integration of this approach in the embedded systems education is proposed in section 4, followed by a discussion in section 5. Finally, a conclusion is given in section 6.

Note: All physical parts of an embedded system are referred to as hardware (e.g. MCU, FPGA), while the languages which determine the behavior of these systems are referred to as software (e.g. assembly, C, VHDL).

2. AN ANALYSIS OF THE SELECTION PROCESS

In industry, the selection of hardware platforms for individual applications is a common task. The decision is made on the basis of experiences from former projects and the

[1]PLDs are also known as *reconfigurable hardware*

WESE '06, October 26th, Seoul, South Korea

knowledge of domain experts and is influenced by many factors. First of all, the hardware platform must allow to realize the desired *functionality* given by functional requirements. Hardware attributes that define functional requirements are the *functional range* and, in case of real-time systems also the *performance*. Functional range defines the amount of functionality (e.g. number of functions, size of data structures) that could be realized on a certain hardware platform. The attribute performance defines how fast the functions are executed.

Besides functional requirements non-functional requirements are of interest for the selection process. They do not represent the required functionality itself but additional qualities the later system must have. Typical non-functional requirements range from reliability, maintainability and testability to power consumption and marketability.

A lot of publications are dealing with non-functional requirements in software design, e.g. [6] or [7]. The strong interaction between software and hardware components in embedded systems demands that designers deal with the combination of both components. The selection of hardware platforms is usually prior or parallel to software design. In order to build an overall system fulfilling the requirements, it is not only important to consider hardware properties, but also their influence on certain software properties.

Accordingly, the selection process demands knowledge of hardware properties, including properties influencing the software, and skills to map these properties on the requirements of the system to be designed. In the following, we will present an approach that can support this selection process.

3. SYSTEMATIC HARDWARE PLATFORM SELECTION PROCESS

3.1 Connecting System Qualities with Hardware Properties

As a result of the former section, functional and non-functional requirements are important for the hardware platform selection process. As a structure representing how much a system fulfills non-functional requirements, a tree structure as the *quality attribute utility tree* used in the ATAM technique [6] developed at the SEI for software qualities, seems appropriate. Since we are interested in support for hardware platform selection, a structure representing the influencing hardware properties on the corresponding system qualities (utility to fulfill non-functional requirements) and functionalities (utility to fulfill functional requirements) is needed. According to these considerations the quality attribute utility tree has been modified to a *hardware attribute tree* (see Fig. 1) representing hardware attributes which influence functional and non-functional properties of the corresponding system.

At this stage it is necessary to find out how these hardware attributes are influenced by hardware properties. This question is quite obvious for some properties while other require the knowledge of experts.

Another aspect is the way these dependencies are presented. We decided to provide them in a graphical way. This presentation, which can be found for several attributes in the figures 2 to 9, is done on an abstract level which allows us to remain mostly unspecific with respect to concrete hardware platforms.

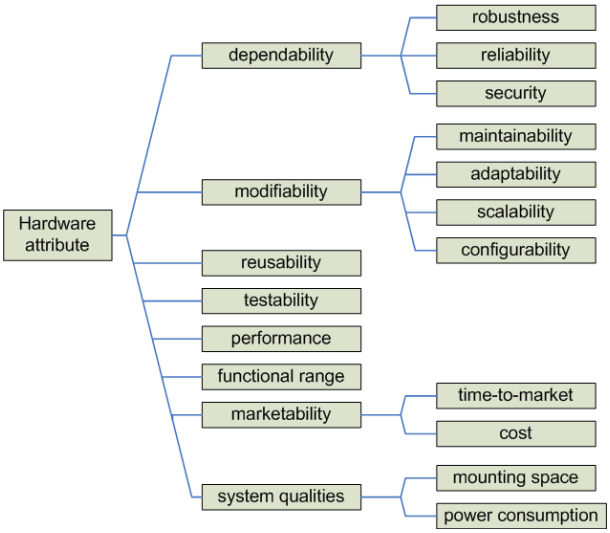


Figure 1: hardware attribute tree

As an example, the attributes *performance*, *functional range* and *marketability* and their influencing hardware properties are looked at closer in the following.

The attribute *performance* is influenced by the architecture of the processing unit (what can be done in one cycle?) and the clock frequency (how fast is one cycle?), as depicted in Fig. 2. The architecture of the processing unit itself is influenced by the bit width (how many bits can be changed at a time?), by the available instructions (what can be processed within one instruction?), by the grade of parallelism (how many things could be done in parallel?), and the integrated peripherals (Which operations have dedicated hardware modules?). Especially the last aspect is of major interest in embedded systems.

As can be seen in Fig. 3, the *functional range* of a device is determined by the available memory (e.g. how much program code could be stored) and/or the available chip area (e.g. how much functionality can be synthesized?), depending on the type of hardware platform. The functional range is also determined by the integrated peripherals (e.g. allow dedicated hardware modules parallel execution of certain functionalities?), the I/O capabilities (can all required external devices be connected?) and abstraction capabilities (how is complexity handled?). The latter factor might be considered controversial, since lack of abstraction capabilities does not limit functional range in principle. However, possibilities to use hardware abstraction or hierarchies are useful to improve the efficiency of many design processes [9].

In contrast to the former two attributes, *marketability* is influenced by factors which are mostly outside the actual target hardware. One of the influencing factors is *time-to-market*. The development time of a device should be kept short in order to save money and to achieve advantages over competing companies. Hardware properties influencing *time-to-market* are the general development effort of a functionality on a certain type of hardware platform (how long does it take to implement this functionality on this type of hardware?), the expertise of the development team (does the team have experience in this type or a comparable type of hardware?) and the external design support. The

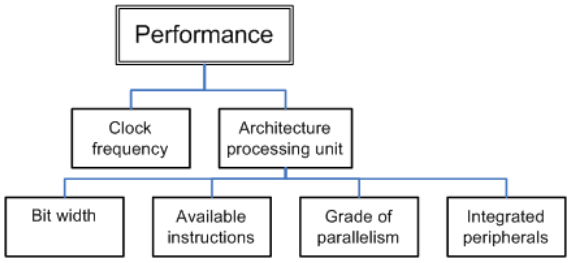


Figure 2: Hardware attribute performance

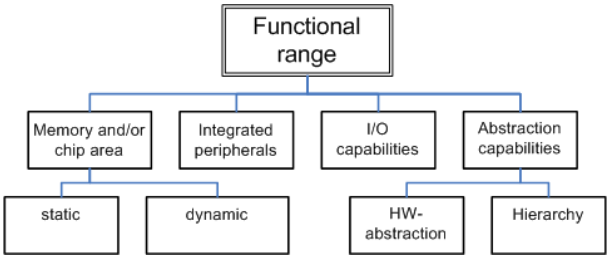


Figure 3: Hardware attribute functional range

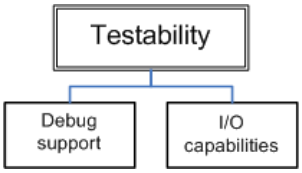


Figure 4: Hardware attribute testability

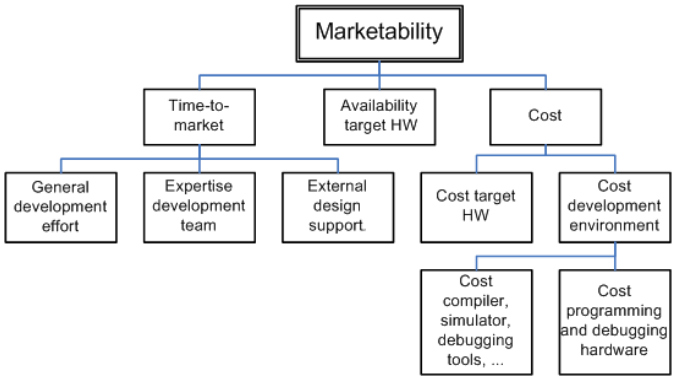


Figure 5: Hardware attribute marketability

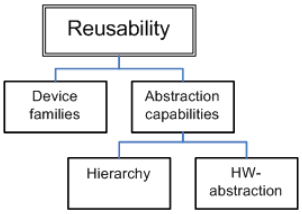


Figure 6: Hardware attribute reusability

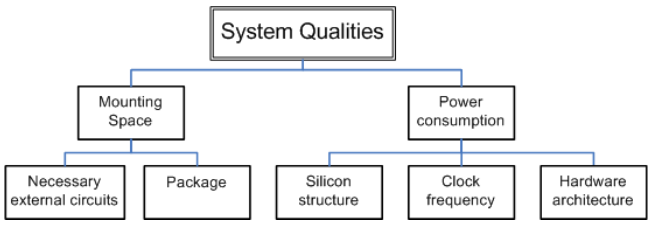


Figure 7: Hardware attribute system qualities

support can be further divided in hardware and software design support. Both include factors as the quality of available tools and the quality of available hotlines, newsgroups, and implementation examples. Another important factor influencing the marketability are the *costs* resulting from a certain hardware platform decision. First of all, these costs include the according target hardware platform itself and the costs to integrate this device physically in the system (e.g. special layout requirements). The degree of influence to the selection process depends mainly on the number of units which should be produced as well as on the target market. The costs for a hardware platform in a high volume consumer product (e.g. mobile phone) are probably of higher influence than the corresponding costs in an unique industrial plant. Another factor are the costs of the development environment needed for a successful implementation of the desired functionalities. This environment includes software as compilers, simulators and certain debugging tools, and hardware as programming and debugging devices. Finally, the availability of a certain hardware platform is an important factor for the marketability. If a design is developed for a certain hardware and the production of this device is stopped, a lot of design effort will be lost if the system cannot be migrated to another hardware platform easily.

The hardware influences presented in the structures above can be divided in *direct* and *indirect* influences. The hardware attribute *robustness*, for example, is influenced directly by hardware properties as the silicon structure or the I/O capabilities. This is different for the hardware attribute *time-to-market*. This attribute is influenced by factors as for example the expertise available in the design team. This expertise includes knowledge about the hardware (direct hardware influence), but also abilities of developing the software for this device (indirect hardware influence). These indirect influences integrate software design issues in the hardware design process, as for example postulated in [3].

Due to the abstraction, applied for hardware platform independence, this representation lacks of concrete information regarding individual hardware platforms. Since this information is important for the selection process, additional information might be provided in form of tables which will be presented in the following section.

3.2 Detailed Hardware Properties

For a successful selection of the optimal hardware platform, it is important to have detailed information about the individual hardware properties (as e.g. available instructions). This information could be taught, for example, on

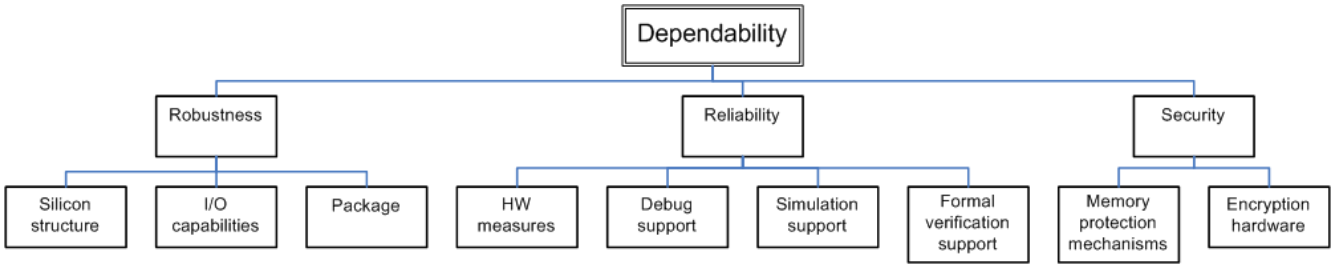


Figure 8: Hardware attribute dependability

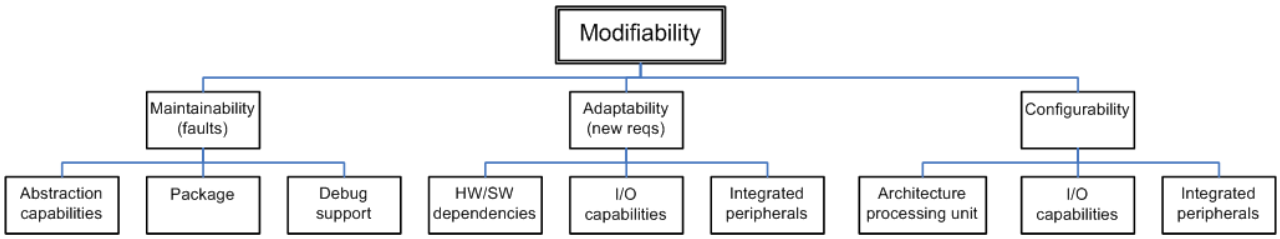


Figure 9: Hardware attribute modifiability

basis of subject-specific literature as [9] or [10]. In our opinion, a useful way to provide a survey of these information is in form of tables as this structure allows to emphasize similarities and differences of different types of hardware platforms. As an example, table 1 represents information regarding the hardware properties influencing the attribute *adaptability*. In the table structure, information of each hardware property is divided into three subcategories. A first category contains general information applicable for all hardware platforms. A second and third category each contain information specific for one of the two main groups of embedded systems, CPU based systems and programmable logic devices respectively. This presentation allows to get information about similarities and differences of these two groups very quickly. Further subcategories can be added, if necessary, by starting a description with the according device name (e.g. DSP: special algorithms for data streaming). Only general information should be included in theses tables in order to maintain clarity. For further information it should be referred to relevant publications (e.g. paper, book chapter, data sheet).

An alternative to the table structure could be a HTML based structure which would also allow the integration of additional levels of hierarchy. To be able to print a readable paper on basis of this structure (e.g. for a lecture script) an according print option has to be integrated to arrange the different levels of hierarchy in a paper readable format.

3.3 Hardware Platform Selection

Since functional and non-functional requirements are important for the hardware platform decision process, a thorough requirements engineering process should be performed in advance (see e.g. [1]). After this process the important qualities and functionalities for the system under consideration are known. In case of programmable systems these requirements have to be met by the combination of hard- and software. For hardware, the according hardware attributes could be found in the graphical structure presented in section 3.1. For each attribute, influencing hardware properties could be found with help of this structure. More detailed information of these properties with respect to different hardware platforms could be found in a table structure as presented in section 3.2 which should be supplemented by information from specific data sheet.

On basis of this information a structured hardware platform selection is possible. Trade-offs usually will be necessary since the different hardware attributes often compete with one another. It is important to mention that the structure presented should not provide the optimal solution but should form a basis to systematically compare the available options.

3.4 Example: 4 channel frequency generator

In order to illustrate the proposed approach, we present the following example. The desired system is a programmable 4 channel frequency generator that is to be programmed via an RS232 serial connection (12 byte message + CRC), with a minimum time between messages of 100ms.

The description of the task includes several functional requirements as performance and functional range. These have to be used for the selection process. In order to have complete requirements, an analysis of the non-functional requirements has been done whose results can be found in table 2.

In this example we analyze the following hardware platforms (with respect to requirements rated as medium or high):

1. low cost MCU
2. medium size MCU
3. low cost MCU + small CPLD
4. medium size FPGA

Table 1: Hardware properties influencing adaptability

Factors influencing adaptability	CPU	PLD	Describtion
HW/SW dependencies	X	X	Determines the effort necessary to transfer software from one device to another
	X	X	Transfer is necessary if requirements cannot be fulfilled with the actual device
	X		Microcontroller families ease migration from one MCU to another of the same family (microcontroller family = same CPU, different peripherals/packages/memory)
	X		Special application notes give help for migration process
	X		Hardware abstraction/operation systems could decrease dependencies between hardware and higher software layers
		X	If the functionality of a module is described in a hardware description language (**behavioral** description), the module can be transferred easily to any PLD suitable for this function In this case, transfer involves only a new assignment of package pins
		X	If the functionality of a module is described in **structural** description, the module can be transferred easily only to hardware platforms with a similar structure (e.g. the same basic elements used must be available)
I/O capabilities	X	X	Determines how easy new requirements with respect to I/O pins could be realized
	X		Usually, certain I/O functionality (e.g. serial input)) is mapped to a particular I/O pin
	X		Some I/O functionalities can be mapped to different I/O pins (e.g. analogue input)
	X		Few MCUs offer a free mapping of functionalities to I/O pins
	X		External buses (system bus, SPI, C2I,...) ease the integration of additional external peripherals/memory in the system.
		X	Usually, all I/O pins have the same properties/options (in, out, pull-up,...)
		X	Clock signals should be fed into the device via dedicated I/O pins
		X	Mapping of functionalities to I/O pins is done by software which allows maximum flexibility
		X	Pin assignment could influence chip area used for the design
Integrated peripherals	X	X	Multi purpose integrated peripherals increase adaptability
	X		Functionality of integrated peripherals can be determined via dedicated registers
	X		Integrated peripherals with a high number of options increase adaptability (and complexity)
		X	Almost all (digital) functionality is determined via software
		X	Predesigned modules are available for common functions, written in HDLs (*soft cores*, [4])
		X	Hard wired peripherals (e.g. clock divider) can be used if available
		X	Integrated peripherals usually do not include analog-to-digital converter
Additional influencing hardware properties can be derived from the hardware attributes *performance* and *functional range*			

Table 2: Example: non-functional requirements

requirement	priority
robustness	high
reliability	high
security	low
maintainability	low
adaptability	medium
scalability	medium
configurability	low
reusability	low
testability	high
time-to-market	high
cost	high
mounting space	medium
power consumption	low

3.4.1 Implementation 1: low cost MCU

First of all, it has to be checked if the functional requirements can be fulfilled with this hardware platform. We assume for this example that a suitable RS232 controller is integrated in this device. Accordingly, serial communication could be realized with minimum implementation effort and CPU burden (Read 12 byte from receive buffer, maximum every 100ms). Only the checking of the CRC would need some calculation time. The generation of the frequency signals is more challenging since no suitable on-chip peripheral is available. If it was the only job for the CPU, it could have been realized in software. However, the RS232 receive buffer has to be read and the CRC has to be checked each time a new message is arriving. The clock frequency of the MCU would have to be very high in order to generate four different signals doing this job concurrently. Programming would have to be realized on a very hardware specific level (probably assembly). Since the maximum clock frequency possible for this device is 16MHz, the task could not be realized on this device.

3.4.2 Implementation 2: medium size MCU

A faster MCU might be a solution for the problem stated above, or an MCU with an integrated peripheral suitable for the 4 channel frequency generation. In this second implementation we are analyzing the latter case. The MCU has to handle the incoming RS232 messages, the according CRC and the update of the peripheral for the frequency generation. Since most of the real-time tasks are done by on-chip peripherals, the MCU is able to handle the required functionality. As can be seen from Fig. 8, the system's *robustness* depends on the MCU hardware (check according data sheet) while the *reliability* depends on several factors. *Hardware measures* to improve the reliability could be a simple watchdog or more advanced built-in CPU- or memory-monitors. The *debug support* stands for the built-in debug hardware (JTAG, Trace, etc.) and the available software tools. They allow us to look inside the embedded system for verification purposes and are available for this device. *Simulation* and *formal verification support* permit the validation and verification of various functionalities before the software is executed on the target hardware platform. Simulation support is available for this MCU, but no real-time simulation. Since most of the real-time functionality is realized with on-chip peripherals this is not any drawback. Formal verification support is not available for this MCU. The *testability* is mostly determined by the debug support, which has been described above.

According to Fig. 9, *adaptability* is influenced by some hardware issues directly (How flexible are my internal peripherals) and by some indirectly (HW/SW dependencies, as how easy is it to migrate to another hardware platform if necessary?). Concerning the latter aspect, adaptability is probably improved if the hardware platform allows the use of higher programming languages. Further information can be found in table 1. In this case, *scalability* is similar to *adaptability* with an emphasis on direct hardware issues. Additional functionality is mostly limited by factors influencing *performance* and *functional range* (see Fig.. 2 and 3).

The influences of the hardware platform on the *time-to-market* are of indirect nature (Fig.. 5). The general software development effort is probably low, since the program has to initialize and coordinate on-chip peripherals mostly. The hardware development effort depends of the number and complexity of the external circuits needed to operate the hardware platform. In both cases, an influencing factor is the usability of the according data sheets and external design support as mentioned in section 3.1. As in all designs, the expertise of the design team has to be considered. The *costs* are determined by the cost for the individual target hardware platform and the costs for the according development environment. The planned production volume determines the importance of these two costs.

Finally, the *mounting space* is determined by the package of the target hardware platform and the amount of external circuits needed for operation (Fig.. 7).

3.4.3 Implementation 3: low cost MCU + small CPLD

In this third implementation, required functionality is realized on a combination of an MCU and a CPLD. The MCU handles the RS232 communication and the CRC check only while the performance-critical part of the frequency generation is done by the CPLD. The MCU and the CPLD are connected via 8bit address/data bus and 3 command lines. The frequency generator can be realized in the CPLD on basis of clock dividers. Functional requirements can be fulfilled by this implementation. The *robustness* of this implementation is depended on the MCU hardware, the CPLD hardware and their interconnection. The CPLD used in this example has damageable input circuits, according to limited protective measures (I/O capabilities). Accordingly, protective circuits for the four frequency signal inputs are necessary. The communication between the devices is an additional point of failure. The *reliability* of the application is now dependent on several hardware devices which decreases reliability in comparison to a single chip solution. On the other hand, the different functionalities (communication/frequency generation) are strictly separated from each other, which could reduce the amount of side effects in the according software. Simulation and debugging is more complicated, since there are two devices which have to be analyzed together. The same is applicable for *testability*. However, since the MCU-CPLD interface is simple and well defined, it should allow simulation and testing of both parts separately.

The *scalability* with respect to a number of frequency signals mostly depends on the chip area available on the CPLD.

Scalability with respect to the maximum frequency depends on the maximum clock frequencies of the CPLD. *Adaptability* with respect to additional functionality concerning signal generation depends mostly on the CPLD chip area available (see table 1). New requirements regarding the communication depend on the MCU. Changing, e.g. from RS232 to USB would only affect this part.

Time-to-market depends a lot on the expertise in the design team and additional support. The implementation involves the MCU part, the CPLD part, the communication part between the two devices and a board design integrating both devices. The costs consist of the costs for both hardware platforms, both development environments, a more complex printed circuit board (PCB), and protective circuits. The mounting space is probably larger than for a solution using only a single device.

3.4.4 Implementation 4: medium size FPGA

Since PLDs seem suitable for frequency generation, it would be interesting to integrate the whole functionality in a larger PLD, e.g. a medium size FPGA. For the RS232 communication, a soft core controller[2] is available which could be integrated into the design. An algorithm performing the CRC has to be integrated. The frequency generation is done as described in 3.4.3. The only control structure would be reading from the RS232 controller, perform the CRC and feed the message to the frequency generator. The *robustness* depends on the FPGA hardware, especially the *I/O capabilities*. To improve robustness protective circuits for the four frequency signals might be necessary. *Reliability* is better than in the previous version since no external communication path is present. Real-time simulation is available for this device. *Testability* is improved since all intermediate signal can be routed to FPGA pins or to internal test modules without influencing the main functionality. *Scalability*, with respect to number of signals and *adaptability* with respect to additional functionality concerning signal generation depends mostly on the FPGA chip area. Changing from RS232 to USB would mean replacing the RS232 soft core by an USB soft core (if available and small enough to fit in device). As in all cases, *time-to-market* does involve a lot the expertise of the design team. In contrast to 3.4.3, this approach only involves one hardware device and no interconnections are necessary. However, the realization of RS232 communication and CRC is probably faster on an MCU than on an PLD. The costs for the hardware platform is probably similar or slightly higher than in case of the second implementation, the costs of the development environment range from none to very high. As in section 3.4.2, the *mounting space* is determined by the package of the target hardware platform and the amount of external circuits needed for operation.

3.4.5 Hardware Selection

For the hardware selection, a survey concerning the suitability of the different hardware platforms is useful, e.g. in form of a table. This table should at least include functional and non-functional requirements which have been considered as important. For the simple example presented above this survey is done in table 3 with "++" representing *very good* to "-" representing *not suitable*. The functional requirements are fulfilled only by the implementations 2, 3 and 4. The platforms 2 and 4 seem to be suited best. Platform 4 seems to be very suitable if the focus is on adaptability and scalability, while platform 2 seems to be preferable if the focus is on fast development (assuming that the development team is familiar with all variations). As in this example the importance of fast development is rated higher than adaptability and scalability, platform 2 should be chosen.

[2]A core is a predesigned, preverified circuit block; a soft core consists of a synthesizable HDL (Hardware Description Language) description that can be retargeted to different devices (see e.g. [4]).

Table 3: Example: comparison of alternatives

requirement	1	2	3	4
performance	-	+	+	+
functional range	-	++	+	++
robustness	+	+	-	+/-
reliability	+/-	++	+	++
adaptability	-	+/-	+	+
scalability	-	+/-	+	++
testability	+/-	+	+/-	++
time-to-market	+/-	++	+/-	+/-
cost	++	+	+/-	+
mounting space	++	+	-	+

4. APPLICATION OF THE APPROACH IN EDUCATION

In accordance with [5, 9, 10] we believe that teaching embedded systems should include CPU and PLD based systems. In our opinion, one minor aspect is whether these different systems are taught together or one after another. However, it is of great importance to provide a comprehensive comparison of the alternatives available. Lab courses, as already mentioned, are well suited for this comparison. Since not all properties of the different hardware platforms could be taught in a lab course, they should be complemented by lecture contents. One possibility could base upon the approach presented in this paper. We will integrate this approach in one of our courses called *Introduction to Embedded Systems*. This course is intended for computer science students, typically in their 5th semester, and comprises the basic properties of embedded systems. Hardware platforms introduced in this course are MCUs, Programmable Logic Controllers (PLC) and FPGAs. In the corresponding exercises the students work with development boards in case of MCUs (Atmel ATmega16) and FPGAs (Xilinx Spartan3) and a simulation environment for PLCs (CoDeSys).

We plan an integration of our approach at the end of the course, as soon as students are familiar with the different hardware platforms. With this background the structure presented in this paper can be developed together with the students. First, system functionalities and qualities and their correlation to hardware properties could be discussed in a lecture. The second step, developing the more detailed tables, could be done by students in an exercise. Proposals could be discussed. It has to become clear that only general information could be included in theses tables. Information, e.g. specific for a certain MCU, could be included as an example, but usually has to be taken from the according data sheets. Later on, students could be provided with a set of tables including general information for embedded hardware platforms. This way, the approach presented could be used

as a comprehensive survey of the material taught.

This approach covers two issues important for embedded systems education. Firstly, it can give a survey of different hardware platforms existent in embedded systems and their influences concerning functional and non-functional requirements. This survey eases the understanding of similarities and differences of these hardware platforms. Secondly, the approach offers a systematic methodology for hardware platform selection based on functional and non-functional requirements. The approach itself is flexible to include additional properties of existent and new future devices and thus is not dependent on any device technology. This methodology should provide students with the skills necessary to use their knowledge in their own future selection processes in a systematic way.

5. DISCUSSION

The aim of this paper is to present a systematic approach for hardware platform design decisions which could be integrated into the embedded system education. The structures and tables presented are based on experiences gained and research done at our institute and are not claimed to be complete.

According to the high number of different and special functions in modern embedded hardware platforms and current changes in techniques and devices, it would be useful to provide a system in which "experts", as domain experts from academia and industry, and application engineers of the according hardware and software companies, could include their knowledge. A HTML-based web system in which participants could give feedback and propose additions to the structure and its contents could be one way to realize this integration of expert knowledge[3].

Due to limited space, the feasibility of the selection method has been illustrated with a comparatively simple application only. More complex applications would not change the selection method (hardware attribute tree, hardware properties influencing these hardware attributes, platform selection) itself, but would complicate the analysis process in case of some system requirements (e.g. reliability for platform A vs. platform B). The analysis of other hardware attributes are less affected by the application's complexity, as for example the marketability. This analysis is part of every selection process and does not represent any particular limitation to the approach presented in this paper.

We are going to integrate this approach into one of our next lectures, called *Introduction to Embedded Systems* which will allow us further evaluation of our approach.

6. CONCLUSIONS

In this paper we discussed the need for education in the field of hardware platform selection. Even if students learn about different hardware platforms used in embedded systems, a comprehensive comparison of these systems is often missing. To overcome this problem, a structured approach for teaching a systematic hardware platform selection process has been presented. In the first step of this approach, functional and non-functional requirements are mapped to hardware properties with the help of a graphical representation. This mapping is done on an abstract level which allows staying mostly hardware independent. In a second step, additional information concerning the hardware properties presented before, are provided in form of tables. These tables allow a separation between general information, specific information for CPU based hardware, and specific information for PLD based systems. Possibilities to integrate up-to-date information in this structure have been discussed. Our plans to integrate this approach in an embedded systems course have been presented. We believe that this two step approach provides a structure which could be readily integrated into embedded system education.

[3]planned web system:
www-i11.informatik.rwth-aachen.de/index.php?id=shps

7. REFERENCES

[1] M. Broy. Requirements engineering for embedded systems. In *Proceedings of FemSys'97*, 1997.

[2] M. Delvai and A. Steininger. Teaching hardware software codesign to software engineers. *1st International Workshop on Reconfigurable Computing Education*, 2006.

[3] B. Graaf, M. Lormans, and H. Toetenel. Embedded software engineering: The state of the practice. *IEEE Software*, volume 20:pages 61 – 69, 2003.

[4] R. K. Gupta and Y. Zorian. Introducing core-based system design. *IEEE Design & Test of Computers*, 1997.

[5] R. Hartenstein. The changing role of computer architecture education within cs curricula. Invited talk, Workshop on Computer Architecture Education (WCAE'04) at 31st International Symposium on Computer Architecture., 2004.

[6] R. Kazman, M. Klein, and P.Clements. Atam: Method for architecture evaluation (cmu/sei-2000-tr-004). Technical report, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.

[7] J. Mylopoulos, L. Chung, and B. A. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *Software Engineering*, 18:483–497, 1992.

[8] F. Salewski, D. Wilking, and S. Kowalewski. Diverse hardware platforms in embedded systems lab courses: A way to teach the differences. In *First Workshop on Embedded System Education (WESE)*, volume 2. SIGBED Review, 2005.

[9] A. Sikora and R. Drechsler. *Software-Engineering und Hardware-Design*. Hanser Verlag, 2002.

[10] F. Vahid and T. Givargis. *Embedded System Design - A unified Hardware/Software Introduction*. Wiley, 2002.

Experiences of a Summer Workshop in Embedded Systems

Kolin Paul **M Balakrishnan**
Department of Computer Science & Engineering
IIT Delhi, India
email: {mbala,kolin}@cse.iitd.ac.in

1. MOTIVATION

In today's scenario, the teaching of courses in embedded systems has become very important. The ever decreasing feature sizes (keeping in tune with Moore's law) has led to the integration of ever increasing number of transistors in the die. Moore's law also has the additional cascade effect of more devices (or functional units) being integrated in the chip. This clearly has led to the chip becoming more "powerful". These chips when assembled on a PCB now have enough capabilities to build large ("embedded") applications. Apart from this, even single chips offer a lot of processing power. For example, a Virtex II Pro [1] chip contains two embedded PowerPC processors (operating at 400 MHz), 100+ single cycle multipliers, resources to build a memory hierarchy, clock managers and high speed serial transreceivers. Clearly using all these features effectively falls into the embedded systems design paradigm. Today hardware designers are compelled to think of at the level of systems rather than at the level of individual components. Similarly software developers for embedded systems requires proficiency in exploiting available hardware resources and they also may have to guide the other system developers in choosing resources appropriately. In the embedded systems domain, system level design is of crucial importance --- this domain fundamentally depends on how effectively we utilize the available resources in an optimum manner. Although embedded systems design is truly interdisciplinary in nature, for purposes of discussion in this paper, we only look at processor based design.

The other reason which motivates us to embark on teaching courses in embedded systems is the trend in industry. Today a systems developer is in more demand than a specialist hardware designer. Also the availability of embedded development boards and kits has made the teaching of such courses feasible.

In this paper, we try to describe our efforts, in teaching embedded systems and systems design in this scenario. In the next section we try to enumerate what we perceive to be the major constraints in teaching such a course. In section 3, we try to list the key enablers which we believe will help in the success of courses in the embedded systems area. We conducted an experiment in the summer (2006) by organizing a Summer Workshop [2][2]. Section 4 describes the structure of the workshop and the results of the experiment. In section 5, we try to summarize our findings.

2. CONSTRAINTS

The structure of the curriculum in most countries, and in particular India, has often not encouraged interdisciplinary work. This has been influenced by almost watertight compartmentalization of engineering disciplines. For example, in most of the engineering institutions in India the students graduating in Computer Science (CS)/ Electronics and Communication Engineering (ECE)/ Electrical Engineering (EE) disciplines are exclusively trained in their own discipline. Our experience with these students shows that the training given to a CS student rarely enables him/her to go beyond building trivial circuits with a number of discrete components. Similarly,

students with an ECE or EE background are often found lacking in basics programming skills. Of course, there has been an effort now to introduce a course in Information Technology which is expected to build bridges across multiple disciplines --- however, in most cases this is at a very nascent state.

In general, there is a huge gap in the teaching of a course in embedded systems --- the subject by its very nature straddles and requires knowledge and skills across various disciplines. At IIT Delhi, we allow our students to work on an embedded systems laboratory. This is a 5 credit laboratory oriented course spread over a period of 4 months where the students work in groups of 4/5 and are expected to develop a system. The focus of the laboratory is to build systems which require cross disciplinary skills from CS, EE and ME (Mechanical Engineering. This course is subscribed to by students from CS/EE and ME. Some of the projects that have been done include

1. RFID based personnel tracking system
2. Theatre Lighting System
3. Campus Cycle System

As is evident, the focus is to enable students to develop interdisciplinary skills. The graduating students from most of our engineering Institutions tend to be very good in their parent disciplines. This is mainly due to the fact that our curriculum is designed to train our students in the core subjects of a particular discipline. However, the world is increasingly becoming interdisciplinary in nature as is evidenced by the growing interest in programs like bioinformatics, biocomputing, sensor networks and embedded computing. In the next section we try to identify a few of the key enablers which have helped us to develop a course in embedded systems.

3. ENABLERS

In the last few years, a lot of manufacturers have made available low cost hardware development boards. Most of these have a small FPGA along with a host of peripherals. The advent of high capacity devices from the two major FPGA vendors has been largely responsible for the growth of interest in developing complete applications around FPGAs. These platform FPGAs have large amounts of RAM (both block RAM as well as distributed RAM). These boards also come with Software Development Packages as well as have some development environment for hardware development.

At IIT Delhi, we have in the past, designed and developed an educational board for Co-Design called INCODE. This has been effectively used in the curriculum of this Institute as well as other educational Institutions for teaching courses in digital systems design. The system based on Xilinx 4010 series FPGAs and 8051 microcontrollers allows students to do

a) Hardware development using both schematics as well as HDL
b) Microcontroller + accelerator to implement codesign projects
c) Build complete systems using keyboard, LED, LCD macros

The availability of low cost development boards which have Spartan3 or Virtex FPGAs from Digilent Inc [3] has encouraged people to develop courses ([5][6]) around these boards. The Xilinx University Program has encouraged the use of these boards by providing technical help as also helping many Institutes with donations. These are ideal in developing small and medium size hardware (and many cases embedded systems) projects. The company also makes available the complete synthesis package freely available off the web (WebPACK). The development of systems which have both hardware and software components has been facilitated by the availability of system development tools like EDK and Quartus SOPC [4]. These tools help the student to quickly use a processor (NIOS/MicroBlaze/PowerPC) along with some custom IPs attached to the bus to build solutions. As a case study, we asked students to port baseline JPEG to MicroBlaze and then build a system which had the DCT component running in

hardware. A C implementation for baseline JPEG was provided. An implementation of the DCT core was also provided. It took a few days of effort to get the system running on the Digilent boards.

The availability of well documented and properly constructed Board Support Packages (BSP) helped in a fast ramp up as the students did not have to struggle through many of the often annoying (and frustrating) details of understanding all pin connections as well as getting UCFs correct. The availability of such tools well integrated into the development framework (EDK) helps to focus students with diverse skill sets to work cohesively --- in a group of two, we found that one student would look at the hardware details whereas the other concentrated on building systems software for the application. Therefore, the key enablers for developing short courses in embedded systems or hardware software codesign are

a) Boards with a lot of peripheral support and a reasonably sized FPGA
b) BSP package which abstracts away the details of the board
c) Good Integrated development environments which allow software and hardware development and simulation to be performed within the same environment.

Today we do see a lot of development in this area in the open source community. This has helped in the effective dissemination of knowledge as many "amateur" developers have joined the bandwagon to develop IP "cores". The availability of such cores (some of them are clearly sub optimal designs) from www.opencores.org and such sites helps maintain and grow the interest in platform based design. Once we were satisfied that we do have the basics in place, we decided to offer a summer workshop in embedded systems for undergraduate students based on these new platform FPGAs. In the next section, we outline the schedule of the summer workshop that we conducted earlier in the year.

4. TRAINING SCHEDULE

The engineering curriculum in India requires that all students undergo a compulsory internship program (typically of 50 to 60 days) at the end of their 3rd year of the four year engineering degree program. A sponsor's help was enlisted and summer workshop program was announced. This internship program in Digital Systems design was run successfully in the previous year also. The summer workshop (SWDSD2006) was planned as a 45 day program. The schedule was formulated keeping in mind the fact that the students had completed 3 out of 4 years of their engineering program. Also the selected students were from different backgrounds --- we picked 6 CS students, 8 ECE/EE students and 2 students who were doing their degree in IT. This ensured that the program had the necessary interdisciplinary flavor. The students were asked to work in groups of two.

The workshop was structured in such a manner that the students were exposed to the hardware and software platforms and crucial design aspects in the first two weeks of the 6 week program. There were lecture sessions on digital design techniques, VHDL Programming, Hardware Software Partitioning and behavioral synthesis. They were complemented with detailed lecture and lab sessions on modern FPGAs and development boards on which they would work. Initially the students worked on the INCODE platform --- this exposed the students to digital hardware design as well as familiarized them with the lab environment. The lectures were primarily held in the mornings and the afternoons were left exclusively for hands on experiments. The focus of the experiments was on system design and correct datapath /control partitioning. It also introduced them to debugging hardware. The INCODE environment as mentioned earlier, allows the development of complete systems including integration of IO devices. Among the experiments that the students did was a GCD Computation – the input was taken

from the keypad and the output was displayed in the LED or LCD displays. All the components were available on the development board. This portion of the workshop was, in our opinion, crucial as this helped students build up the necessary confidence to work with hardware as also tinker with the environment to sort out problems like applying service packs, assembling and checking null modems and other cables etc.

We decided that we would like to expose the students to these platform FPGAs and ask them to develop systems based on a microprocessor (available as soft core or hard core) and some IPs. The idea was to get the students to appreciate systems building using some of the features available on the development board (like SDRAM interface, VGA Controller etc). The lecture sessions introduced the concepts of platform FPGAs along with demos to show how a MicroBlaze or a PowerPC based design is done. The students learnt how to instantiate such cores, attach IPs (like UARTs) to the bus in the Embedded Development Kit (EDK) environment as well simulate the entire system using Modelsim. Synthesis steps to get the design running on the target board were clearly explained in the demos. The availability of well documented examples (of "hello world" type programs) whose design included a soft core (Microblaze), UART, hyper terminal etc was a great help in the ramping process. These tutorials are available in the course pages of many universities as well as in the Xilinx website. The experiments were done on the Digilent Boards which contained a Virtex II Pro FPGA and was made available by Xilinx under their University Program. The students were able to complete their preparatory labs at the end of the third week and were in a position to clearly articulate what "major" project they would undertake. The workshop schedule mandated that the participants complete a project and two best project awards of considerable financial value were at stake. The coordinators of the workshop worked with the students to define the parameters of the projects taking care to ensure that pragmatic design choices were made. We give below short descriptions of 3 out of the 7 projects that were done in the workshop to illustrate the complexity of the work done.

a) Implementation of 2-Player PAC-MAN Game On FPGA

The objective of the project was to implement a 2-player Pac-man game on the XUPV2P board, so that part of the game resides in MicroBlaze and part of it is implemented on the CLBs of the FPGA. Pac-man is an age old game, in which a small figure (Pac-man) moves around a maze eating small round pellets and saving itself from monsters rushing towards it. The task was to design the entire game on the XUPV2P board. It was decided to design the game controls on software, i.e. controlled by MicroBlaze, and a Graphics Accelerator on the hardware, i.e. on the CLBs. The human player communicates with the system by way of keystrokes from a standard QWERTY keyboard; the keystroke information is translated into certain data and signals by the MicroBlaze and this information is then transferred to a shared memory (which is part of our custom logic) through the OPB(On-chip Peripheral Bus). After this, the Graphics Accelerator part of our custom logic extracts the information from the shared memory and uses this information to generate proper pixel-information for the VDU. The Graphics Accelerator contains a VGA-Timing module which generates synchronizing and blank signals for the proper display on the VDU. The pixel information is actually passed onto the DAC, which passes them onto the VDU port. The VDU is refreshed multiple times within a second by the Graphics Accelerator. The keystrokes are reflected to the player as congruous movement of

his or her Pac-man. The monsters are controlled by the game controller program; their movement is effected by an AI logic that takes as inputs the keystrokes given by the user.

b) Implementation of 256 point Radix-4 FFT module

The system designed calculates and displays Fast Fourier Transform of the streaming audio input. The audio input is fed through the MICIN port of the AC'97 of the DIGILENT board and the output can be taken from its VGA port. The displayed output shows the power spectrum of the input signal. Due to presence of noise the output gets slightly distorted. The system used LM4550 (AC'97 standard) chip for taking the sampled inputs. The EDK library (available for the Digilent board) controller for the AC 97 was used. The Microblaze then starts taking audio samples, which is then fed to the FFT module. The pipelined FFT module processes the input to produce the output, which is then fed to the Video RAM by the Microblaze after calculation of its power. The VRAM is BRAM, which is shared by both the Microblaze and the VGA controller. Its locations are mapped to the locations of the monitor.

The FFT module is divided into four stages of almost identical computations. This has helped in achieving pipelining and streaming although the amount of memory used is larger than what was required. In the design standard IPIF interface provided by EDK was removed and a custom interface between OPB and the FFT module was added.

c) Multiprocessor Emulator

The objective of this project was to develop an emulator for a multiprocessor system using 4 soft core microblaze processors and running an application on it. These microblazes were attached with a common OPB bus through which they communicated with the peripheral devices attached with the bus. Each microblaze had its local memory in which the application for that processor resides along with its local data. The communication between the microprocessors was done through a shared memory which was also attached to the OPB bus. The shared data was stored in the shared memory. The synchronization of the use of the peripheral by the microprocessors was controlled by a Custom IP called SYNC IP which was attached to the OPB bus. Each microblaze had an ID which was to used to synchronize its access to the shared resources.

5. ANALYSIS

Clearly, the focus of all these projects was on the building of a complete self contained system. We were satisfied with the level achieved by the students keeping in view, the constraints that they faced. All of these projects clearly required more that 3-4 weeks time allotted in the workshop. Despite this primary constraint, all the groups did manage to get some (in many cases, all) aspects of the design working. We believe that the following reasons can be attributed for this apparent success:

- Defining a structure for the workshop which enabled the students to initially work on a system which was simpler and well documented. Seeing their non trivial hardware designs work gave them the confidence to aim higher.
- Availability of good instructional material which allowed a fast ramp up. The

easy access to tutorials as well as complete project zips (with HDL code as well C Code for the application along with the project make files) was paramount in students getting interested and working very hard. These tutorials worked out almost off the box and helped the student to gain in confidence.

- The availability of (inexpensive) hardware (FPGA chips and boards) which were completely supported in EDK. This has often turned out to be a major bottleneck as getting to understand board specifics and getting UCFs right often takes a lot of time. The associated BSPs helped the students in not spending too much time in the understanding of pin connections and other board details and they could work with the abstraction provided by the BSP.
- Availability of drivers (HDL as well as Software) for the hardware modules available on the board was very significant. This can be appreciated from this use case scenario. The base installation of EDK did not have the correct driver for SDRAM attached to the board. After connecting the module in EDK and building the system, incorrect results were obtained. This took 3 crucial days to fix with help from the online community. Once the correct drivers were put in the pcores directory, the application worked smoothly.
- The growing number of newsgroups and online communities interested in hardware and embedded systems development. Many of the initial teething problems faced by the students in using the boards or getting MicroBlaze /PowerPC to behave correctly were resolved with 12-24 hours of posting the problem in the web.
- Availability of design templates which the students could import into their designs.

Availability of low cost development boards, good co design software and a wealth of designs available in the web are the salient drivers which could make the students do this much in the limited time span.

6. CONCLUSION

The experience with the summer workshop shows that in a period of six weeks, it is possible to introduce processor based design on Platform FPGAs to a mixed audience of CS and EE students. The structure requires introductory design classes, ramp up with simpler designs followed with projects involving platform FPGAs on embedded development boards. The use of online material and help is crucial in the process.

7. REFERENCES

[1] http://www.xilinx.com
[2] http://www.cse.iitd.ernet.in/~kolin/Training/FPGA/Summer%20Workshop%202005.html
[3] http://www.digilentinc.com/
[4] www.altera.com
[5] http://www.eece.unm.edu/xup/index.htm
[6] http://www.cse.cuhk.edu.hk/~phwl/mt/public/archives/old/ceg5010-2005/

The Delft MS Curriculum on Embedded Systems

Hans-Gerhard Gross
Delft University of Technology
Mekelweg 4, 2628 CD Delft, NL
+31 15 27 87750
H.G.Gross@tudelft.nl

Arjan van Gemund
Delft University of Technology
Mekelweg 4, 2628 CD Delft, NL
+31 15 27 87750
A.J.C.vanGemund@tudelft.nl

ABSTRACT
Embedded Systems (ES) is the fastest growing sector in ICT technology [1, 15]. Also in the Dutch economy, this sector is believed to become an important generator of added value. In this paper we describe a new MS program on ES that will be offered in Delft as of the academic year 2006. The MS program is a joint offering by the three Universities of Technology within The Netherlands (at Delft, Eindhoven, and Twente). We describe the program, its rationale, and two examples of already existing courses (Embedded Systems and Real-Time Systems), from which the new ES curriculum has emerged.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education; *Curriculum*

Keywords
Embedded systems, Curriculum

1. INTRODUCTION
Amongst the 13 universities present in The Netherlands, three of them have a distinct focus on scientific engineering, namely (i) Delft University of Technology [5], (ii) Eindhoven University of Technology [7], and (iii) University of Twente [23]. For many years each of these three universities offer BS and MS programs in Computer Science (CS) and Electrical Engineering (EE), and in the case of Delft, also an MS program Computer Engineering (CE).

In sync with the international trend, within The Netherlands there is a growing understanding that ES can no longer be designed in terms of two separate threads of hardware and software that are merged at a later stage [1]. A systems approach is required that mixes functional and non-functional requirements from the start. Central to this approach is the need to understand the interaction of the system with its physical and network environments. These changes require engineering teams that possess skills in a wide range of disciplines such as computer science, electrical engineering, real-time computing, systems architecture, control engineering, signal processing, security and privacy, computer networking, mathematics, hardware, sensors and actuators. Engineering teams are currently unable to effectively consider fundamental design issues from all these perspectives simultaneously, because they lack the common background and technical language to interact efficiently. Creating these multidisciplinary skills requires fundamental changes in engineering education, and, since a number of years, many courses and curricula on ES have emerged [6, 16, 17, 20].

Motivated by (1) the need for ES engineers at MS level, that master, or at least are comfortable with the above range of multidisciplinary engineering subjects, and (2) given the ever increasing disparity between the current, discrete-domain driven CS curricula and the predominantly continuous-domain driven EE curricula, the Dutch Ministry of Education has approved the implementation of a new MS program on ES, slated for the academic year 2006. Note, that it is not the intention (nor possible) to 'retrain' CS bachelors to EE masters, nor EE bachelors to CS masters. An important motivation of an MS program on ES is that the bachelor becomes comfortable with the complementary domain. For instance, a graduated ES master with a BS CS will much better understand the language and tools of EE engineers, and has better feeling with the EE problems, and vice versa. In this way, they can more effectively work together on ES.

Unlike traditional programs, for reasons of efficiency[1], the Ministry has stipulated that in order to receive accreditation, the ES program be offered jointly by the three Dutch technical universities, which have recently been federated under the joint brand name 3TU (TU = technical university). While this joint offering implies that all Dutch students learn a national MS degree ES (issued by 3TU), the current implementation still allows some room for a local Delft, Eindhoven, and Twente differentiation.

In this paper we describe the MS program ES as offered by Delft University of Technology. Although some 80 percent of the two-year program2 is identical for all three universities, the Delft program has a particular focus on (1) embedded software, (2) bridging the gap between the discrete and continuous domain, and (3) system-level engineering, rather than component-level engineering which is typical for the Dutch EE/CE programs. Apart from describing the curriculum and its rationale, we describe two courses IN4073 and IN4024 (Embedded Systems and Real-Time Systems, respectively) that have been offered for a couple of years as electives within the CS master program, and which are now part of the mandatory core of the new ES program. In particular, the IN4073 offering can be seen as typical for the Delft approach to ES education. The paper is organized as follows. In Section 2, we present the new ES curriculum as offered by Delft, and motivate the specific focus on embedded software and multi-

[1] No more than some 100 students are expected to register nationwide.

[2] Dutch MS programs are typically two-year programs, comprising 120 EC. 1 EC (European Credit, defined through the European Credit Transfer System, ECTS) stands for 28 hours of nominal study load.

disciplinarity. Section 3 describes the course offering IN4073 Embedded

Systems, while Section 4 describes the Real-Time Systems course. In Section 5, we provide some additional details on the general didactic context in which our education is currently performed. Section 6 concludes the paper.

2. DELFT ES PROGRAM

The Delft version of the 3TU masters program ES [12] is offered by the Faculty of Electrical Engineering, Mathematics, and Computer Science (EEMCS [9]), which covers the entire embedded systems engineering spectrum from embedded software engineering to sub-micron engineering. The overall, high-level learning goals of the program aim to bring students into the position to develop and apply new research ideas in a multidisciplinary working context, integrate their knowledge to solve complex problems, and make judgments based on limited or incomplete information, reflect on the socio-ethical impact of their judgments, and communicate their decisions, solutions, and reflections to other people even outside of their field. The ES program has the following overall structure:

- Mandatory courses (40 EC). This is the common core of the ES program. Up to 10 EC of the mandatory part consists of so-called homologation courses to equalize the differences in previous educational background.
- Elective courses (40 EC). The electives span a broad range from control theory to sub-micron Si realization. Up to 20 EC of the elective program can be taken in terms of a traineeship, preferably carried out with an international company or research institute.
- Thesis project (40 EC). The thesis project includes an introductory (10 EC) individual project in preparation. This individual project is tailor-made and may contain such elements as literature surveys related to the final project's subject, preparatory research studies, or additional specialist courses, whatever is needed to make the student well-prepared for the project.

Whereas the scope of the electives and thesis project is equal for all three universities, the differentiation between the universities lies in the mandatory part. Consequently, in the following, we will focus on the mandatory part of the curriculum.

2.1 Delft Focus

With regard to the mandatory part of the program, Delft has a particular inclination towards embedded software, dependable systems, and multidisciplinarity.

2.1.1 Embedded Software

As an increasing amount of functionality shifts from hardware to software, embedded software engineering cost is becoming the bottleneck. A contributing factor is the ever decreasing cost of (field) programmable hardware (microcontrollers, FPGAs), which often outweigh the advantages of devising application-specific silicon solutions. This applies in particular to the Dutch context[3] where, on a whole, more valorization is likely to occur in the software systems (specification, integration) domain, than on the processor domain[4].

Another important aspect of the above Delft orientation towards systems and software with respect to the ES program, is the fact that Delft, unlike Twente and Eindhoven, already has a MS program CE in place for a number of years. Although the Delft CE program does include compilers, system software, and operating systems, the main aim of the program is to produce experts in computer architecture and the development and implementation of computer hardware (for example, more than 25 percent of its mandatory core is entirely devoted to computer arithmetic). Although there is overlap between the CE and ES programs with respect to embedded systems, the Delft CE program effectively focuses on hardware issues[5]. In a sense the Delft ES program seeks to complement the existing CE program, by focusing on embedded software, multidisciplinarity, and system-level engineering, rather than embedded hardware, and component-level engineering.

2.1.2 Dependable Systems

Furthermore, the increasing level of connectivity that comes with "ambient intelligence" is leading to increased availability of data and information, anywhere and at any time. This offers a huge potential, but also presents tough challenges in terms of interoperability, efficiency, complexity and vulnerability. Embedded systems must also be safe and reliable: it is important that increased complexity does not compromise their dependability. In line with this emphasis on embedded software and dependability, Delft has established a new (part-time) chair Embedded Software that performs research and education on embedded software and dependability [8].

2.1.3 Homologation

Similar to Twente, a specific feature of the Delft approach is that a part of the mandatory program is devoted to what is called homologation. Homologation is a an equalization process aimed at achieving homogeneity with respect to the student's starting level, in view of the fact that the MS program is typically entered by CS and EE bachelors which have quite a different background. For example, Dutch students with an EE BS will lack knowledge and experience in, e.g., programming, operating systems, and software engineering, whereas Dutch students with a CS background will

[3] In a hardware sense The Netherlands is essentially an import economy.

[4] In this view, hardware synthesis based on HDLs is considered a software approach. Actually, in the CS bachelor program, a sophomore course on HDL programming is envisaged for the academic year 2007.

[5] It is worth mentioning that the Dutch situation with respect to CE slightly differs from that in the US. The CE Body of Knowledge (BOK) in the US [14] has naturally included embedded systems (denoted ES-ESY) from the outset, even though it has recently been acknowledged that an ES BOK should somewhat extend this CE BOK [21]. With only one (Delft) CE program in The Netherlands, which, compared to the CE BOK, is more hardware-oriented, there is even more point to start a new ES program than in the US.

generally lack knowledge and experience with, e.g., logic design, control theory, and signal processing. As indicated earlier, the purpose of the ES program is to essentially integrate the CS and EE disciplines, rather than just providing an ES flavor to CS and EE bachelors, who would then maintain their respective path through the electives and final project without meaningful cross-fertilization. Hence, the first 10 EC of the master program, called the homologation phase, provide specific subjects to remedy the differences in background. Depending on a student's background, some of these subjects will be mandatory. Homologation is seen as a crucial instrument to acquire a common background to achieve early integration between the various disciplines. It provides the students with the broadening that is required to effectively absorb the multidisciplinary program that lies ahead.

2.2 Mandatory Courses

As mentioned earlier, the 40 EC mandatory part includes a 10 EC homologation phase and a 30 EC common core. For a student with a Delft CS bachelor degree, the mandatory homologation subjects are Systems and Signals, Control Theory, Logic Design, and Digital Signal Processing. For a student with a Delft EE bachelor degree, the mandatory subjects are Operating Systems, Systems Programming (using C), and Software Engineering (using Java). While the homologation phase establishes a common background, the following 30 EC offerings essentially constitutes the common core of the curriculum:

- IN4087 System Validation (formal methods, model checking)
- IN4088 Software Testing and Quality Engineering (testing large embedded software systems)
- IN4024 Real-Time Systems (detailed in Section 4)
- ET4282 Performance Analysis (performance modeling of computation and communication)
- IN4073 Embedded Systems (detailed in Section 3)
- ET4165 Embedded Computer Architecture (contemporary embedded processors, microcontrollers)

Compared to the two sister universities in Eindhoven and Twente, the above program has a distinct focus on systems, software, and dependability. It should be kept in mind, however, that next to the 40 EC thesis work, the remaining 40 EC electives offer a very broad scope, ranging from control engineering to sub-micron Si realization, parts of which can be taken from any of the three universities.

Describing all elective courses (more than 50) would go well beyond the scope of this paper. We can therefore only list the range of courses offered. Electives comprise courses on system modeling, signal processing, software engineering, artificial intelligence, high performance computing, robotics, and microprocessor design. The full list of courses can be obtained from the study information system of TU Delft [18]. Elective courses are chosen by students based on consultation with the MS ES coordinator of the faculty and the supervising professor of the thesis project.

3. COURSE: EMBEDDED SYSTEMS

The course IN4073 Embedded Systems [10], now part of the mandatory ES curriculum, originates from an elective offering as of 2004 within the Delft CS master's curriculum (and is still being offered at CS). Based on the new guidelines for quality teaching at Delft described in more detail in Section 5, the course is primarily aimed at providing the students with hands-on experience with designing embedded systems that perform non-trivial tasks. The main ingredient of the 6 EC course is a lab project carried out by 12 competing teams of around 5 students per team, where each team must design an embedded system to control a model helicopter. Due to financial restrictions, each team has (supervised) access to the lab for only 7 slots of 4 hours per slot. This implies that most of the work must be prepared by the teams outside lab hours, while the actual testing is to be done during the 7 slots.

The course has been taken by some 55 MS students each year, of which some 50 pass the course. The students' background is mostly CS (60 percent) and CE (30 percent). Future editions will, of course, include the new ES students. As the lab work is essentially multidisciplinary, each team is composed of a mix of CS and CE students, which usually presents a first-time opportunity to get acquainted with different engineering backgrounds. Apart from the lab work, the course includes a number of supporting lectures, as to provide the necessary background knowledge to successfully perform the lab work. Given the large focus on lab work, a support web site is provided [11] that contains various resources, rather than a text book. In any case, embedded systems text books are usually very much oriented towards a particular hardware architecture, instruction set, and/or programming model. As indicated earlier, our approach towards embedded hardware synthesis is through software. Consequently, in the design lab, hardware is programmed in terms of VHDL and C virtual machines only, where hardware particulars are reduced to a few trivialities such as memory mapping registers of the (VHDL) devices in the C programming model, and setting interrupt routine function pointers. The goal of the 6 EC credits course is not to have the student master all multidisciplinary skills of embedded systems engineering, but rather to have the student understand the basic principles and problems, develop a systems view, and to become reasonably comfortable with the various disciplines involved in embedded systems design.

3.1 Lab Project

In order to describe the body of knowledge covered by the course, we will briefly detail the lab project. The project chosen for this course is to design embedded software to control and stabilize an electrical model helicopter, such that it can be flown by inexperienced users using a single joystick. This application has been chosen for a number of reasons:

- The application is typical for many embedded systems, i.e., it integrates aspects from many different disciplines (mechanics, control theory, sensor and actuator physics/electronics, signal processing, and last but not least, computing hardware and software).
- The application is contemporary. Today's low-cost RC model-helicopters are only rudimentary controlled (simple yaw control), which implies that only skilled hobby pilots are capable of flying these machines (i.e., simultaneously controlling motor speed, pitch, roll, and

yaw[6]) without crashing within a few seconds after lift-off. Although perceived as the true sporting challenge, many recreational users (the authors included) would prefer an aerial vehicle that is much easier to handle.

- The application is typical for many air, land, and naval vehicles that require extensive embedded control software to achieve stability where humans are no longer able to perform this complicated, real-time task.
- Last, but certainly not least, helicopters are great fun.

Although designing embedded software that would enable a heli to, e.g., autonomously hover at a given location, or move to a specified location (a so-called autopilot) would by far exceed the scope of a 6 EC course, the course project is indeed inspired by this very ambition. In fact, the project is presented to the students as a prototype study aimed to ultimately design such a control system, where, e.g., a pilot would remotely control the helicopter through a single joystick, up to the point where the helicopter is entirely flown by software.

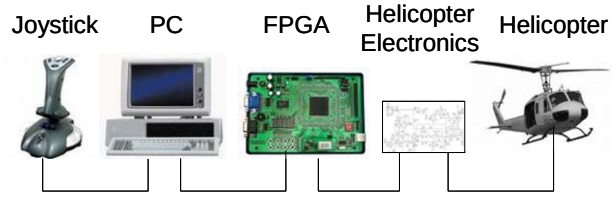


Fig. 1: Hardware Setup

3.1.1 Experimental Setup

While the ultimate embedded system would ideally be implemented in terms of one chip (SoC) which would easily fit within the electrical model helicopter's limited payload budget, the student's prototype embedded system comprises an external FPGA board, that is connected to a sensor/actuator electronics board that interfaces the FPGA board with the helicopter's actuators (rotors, servos) and sensors (gyros and accelerometers). The system (which is perceived as part of the helicopter) receives its commands from the so-called ground station, which consists of a Linux PC and a joystick. In our prototype approach, we refrain from implementing radio communication between the helicopter and the ground station. Rather, we conduct tethered flight, where the helicopter is connected to the ground system through wires. This, incidentally, also allows for prolonged testing, since low-cost helicopter batteries are usually depleted within 20 minutes. The system setup is shown in Figure 1. As shown, an additional, analog electronics board is required to interface the heli rotors, servos, and sensors to the FPGA board. The (low-cost) board, specifically designed for the used sensors, is realized by the authors, and features power FETs for PWM motor control, and DC-to-PWM converters for the 3 gyros. Although, the prototype embedded system therefore comprises two boards (FPGA +heli interface), it is perfectly possible to map all electronics within one chip onboard the helicopter[7].

3.1.2 Resources

Rather than resorting to expensive helicopters, sensors (e.g., entire IMUs), electronics, FPGAs, etc., we explicitly choose a low-level, low-cost approach, which opens up a real possibility for students inspired by this course to continue working with helicopters and/or ES on a relatively low budget. The helicopter[8] is a low-cost (O(100) Euros) version that essentially comprises a fuselage with only the main + tail motors, and the pitch + roll servos (i.e., 4 actuators). The gyro and accelerometer sensors required to derive the attitude (which is the minimal information for a simple autopilot application) are also low-cost (O(50) Euros) per sensor; at least 5 are required for proper attitude computation.

The ES hardware approach is also low-cost. Rather than using some high-performance microcontroller (and costly development tool suite) we use a low-cost FPGA board (O(100) Euros) featuring a 400K gates Xilinx Spartan-3 (XC3S400), that comes with free development tools. In order to allow less time-critical parts of the ES to be conveniently developed in C rather than VHDL, an experimental 32 bit soft core (VHDL processor component) is provided, that interfaces with high-speed VHDL components through shared memory. The soft core, called X32, has been developed at our Embedded Software Lab [24], and comprises the VHDL core and associated C programming tool chain (ANSI compiler based on lcc, assembler, linker, simulator, up-loader, debugger). A Delft development, the X32, is license-free, allowing the entire software to be used by students at home without cost.

The PC is a standard Linux PC, that acts as development and upload platform for the FPGA (VHDL) and the X32 soft core (C). The PC is also used as run-time user interface, in which capacity it reads joystick and keyboard commands, transmits setpoint commands to the embedded system, logs telemetry data from the embedded system, while visualizing and/or storing the data on file for off-line inspection. The PC has a parallel port (LP), used to upload FPGA designs to the FPGA board, and a 115,200 baud serial link (COM) to communicate to the FPGA board at run-time.

Of course, the above low-cost approach is not without consequences. A very small helicopter implies that it is highly unstable and therefore extremely difficult to control in comparison to real helicopters. Moreover, instead of controlling thrust using collective rotor pitch, thrust is currently controlled by motor speed, which also adds to the stability problem. Using low-cost sensors introduces larger measurement errors (drift) which degrade computed attitude accuracy and control performance. Using a low-cost FPGA (compared to a high-end FPGA with multiple hard processor cores) severely reduces the size and performance of the designs. Although the X32 occupies less than 50% FPGA space, this leaves limited space for the additional VHDL devices (timers, PWM converters, UART) that make up the total microcontroller architecture. Moreover, the FPGA designs typically run at 50 MHz, which implies limited soft core processing performance,

[6] pitch, roll, and yaw are the three angles that constitute the helicopter's attitude in 3D.

[7] In a future version, the interface electronics will be integrated within the heli.

[8] Currently a Piccolo V2, to be succeeded by a more sturdy TREX 450 as of next year.

compared to a high-end hard core. Nevertheless, practice has shown that the low-cost setup is more than capable to perform adequate helicopter stabilization, and provides ample opportunity for students to get fully acquainted with embedded systems programming in a real and challenging application context that includes resource limitations.

3.2 Design Challenges

The FPGA - PC design involves components such as signal filters, yaw, roll, and pitch controllers, serial-parallel communication transceivers, pulse-width modulators/demodulators (motors, servos), joystick/keyboard handling, and a UI (OpenGL), including a main FSM controlling the various mode of heli operation (safe mode, calibration mode, yaw stabilizer-only, full roll/pitch/yaw stabilization). The design involves a number of challenges, including:

- concurrent real-time programming and debugging at both FPGA and PC (e.g., scheduling/interaction of FPGA - PC communication tasks, the PID controllers, and signal filters),
- hardware/software co-design (determining which components must be implemented in VHDL or C in view of performance and space limitations),
- coping without floating point support: as many low-cost microcontrollers, the X32 has no floating point support. Hence, controlling and filtering must be done using fixed-point arithmetic, which introduces stability issues,
- teamwork: producing an overall system design, and partitioning the overall design in components, allowing parallelization of student effort, considering the various differences in educational, and cultural background,
- understanding the various disciplines (mechanics, electronics, control theory, software engineering), and understanding and coping with the various interfaces (sensors, motors, joystick, graphics, X32 memory and programming model).

The supporting lectures include lectures on helicopter mechanics, modeling, and simulation, control theory, digital filter theory, VHDL programming, X32 architecture and programming, and basic electronics.

3.3 Evaluation

Currently, the course has seen two editions (academic year 2005 and 2006). As to be expected with a new course, a number of issues surfaced. In the 2005 edition, the moderate goal of the lab was to design helicopter yaw control only (i.e., no roll and pitch control). Even then, students found themselves struggling with many side issues, mostly relating to insufficient prior knowledge of VHDL. Consequently, PC—FPGA communication, which involved designing UARTs at the FPGA side, as well as designing PWM generators to control the heli actuators, and the PW readers to read back sensor data (also in PWM format), generally took more than half of the lab time. As a result, only 50 percent of the teams was able to deliver a demonstrator that performed helicopter yaw control.

In the 2006 edition more functionality was added to the soft core, such as a UART, which somewhat took the load of the VHDL part, allowing more focus on the actual control problem (which is programmed in C on the X32). As this year's goal was to also deliver pitch and roll control (next to the easier yaw control), signal filtering became a more demanding issue. At this point, another deficiency became apparent, namely the fact that most students had no prior hands-on experience with designing fixed-point IIR filters, even though, at least half of the students had prior exposure to a DSP course. Although the course provides a supporting DSP lecture (a 4-hour crash course), immediately applying the fresh knowledge proved to be time-consuming. While, on average, the 2006 results were much better than 2005, only some 25 percent of the teams were able to demonstrate 3D attitude control.

Despite the fact that most teams didn't fully comply with the project goals, student satisfaction and retention was very high. In both editions of the course, less than 10 percent failed the course. Students consistently indicated a steep learning curve, especially with regard to the subjects outside of their curricular domain. Due to the low-cost setup, some 10 percent of the students actually purchased their own (FPGA) hardware to continue on private ES projects. The time spent on the course typically exceeded the nominal 168 study hours that stands for a 6 EC course.

One major complaint was the lack of lab time and resources. Although each team had access to an FPGA board outside lab hours (and access to two FPGA boards during lab hours plus a PC for each team member), lab time was usually devoted to testing and debugging (parts of) designs, which, of course, took much longer than anticipated. A complication was the fact that there is only one model helicopter available, leading to restrictions on helicopter test time. Although, students understood that this adds to engineering reality, we are inclined to investigate the (financial) possibility of increasing the level of lab support.

4. COURSE: REAL-TIME SYSTEMS

The course IN4024 Real-Time Systems [13] is also 6 EC. The course originates from the CS curriculum in which it is still taught as an elective module. It is obligatory in the new ES curriculum. Both courses, IN4073 and IN4024, are complementary. The real-time system course is intended to equip students with the basic concepts of real-time systems in the context of a standard PC, whereas the embedded system course goes beyond the boundaries of the PC, connecting it to an outside physical world, and thus adding another dimension. This is why, in the schedule of the curriculum, the real-time system course is placed before the embedded systems course.

IN4024, similar to IN4073, is split into 14 two-hour lectures and 7 four-hour lab sessions, with a strong emphasis on hands-on real-time system development experience. The lab sessions are the driving factor of the course, and this is in line with the new didactic requirements of the university (see Section 5). Assessment is carried out based on an 8 page research paper that the students have to submit at the end of the module. This focus on research is due to the fact that an MS degree requires scientific skills, in contrast to a BS which concentrates more on predefined implementation of tasks. Additional advantages of such an assessment are the students' improved research and writing skills when they enter their final year MS projects. The lectures are intended to equip them with the right terminology, and in the labs, they gain experience with doing their own research on a selection of subjects in

the real-time system domain. Typical number of students at registration is 100, going down to 60—70 in the first two lectures, dropping to some 35 in the last lecture with just about 40 paper submissions at the end of the module, all of which pass the course. The quality of the submitted papers is usually high[9].

4.1 Laboratories

The equipment of the labs comprises standard Linux PCs with the RTAI real-time extension [19] installed. We do not permit the students to use the full functionality of the real-time modules provided, which, in fact, would require the students to have administrative permission for loading and unloading their own real-time modules. Instead, the system provides the LXRT module which implements a user-level interface to RTAI's real-time services. This requires only normal user privileges with minor overall performance losses. RTAI is a freely available, easy to use real-time environment that most students install on their own PCs in order to be able to spend more time on the subject beyond the scope of the lab sessions, and perform more thorough experiments. We find this very positive. The real-time system lab is not one single larger project, but rather based on a number of different assignments that the students have to solve. They are supposed to implement parts of the assignments in C, do some experiments, e.g., timing measurements and devising schedules, reflect on the problems encountered, and the solutions found during the assignments. The assignments are designed in the form of scenarios that would be typical in a normal working environment in industry, such as "we need the worst-case execution time for this algorithm", "choose the best implementation of existing algorithms for the constraints of this system", or "which scheduling strategy would be optimal under such circumstances", etc. The assignments represent the framework in which the students carry out research. The following types of assignments are available:

- Comparison of standard Linux timing operations and the RTAI timing operations, accuracy of time measurements,
- Development of a high-resolution timer based on the processor clock and comparison of that timer with the standard Linux and RTAI timers,
- Development of a code framework under RTAI for worst-case execution time (WCET) analysis experiments and scheduling experiments
- Dynamic worst-case execution time analysis for standard algorithms, e.g., sorting, searching, computer graphics, etc., optimization of algorithms toward higher analyzability,
- Evaluation of dynamic timing analysis through code coverage analysis,
- Design of constant execution time algorithms (WCET-oriented programming),
- Search-based execution time analysis; comparison between manual testing, random testing, and application of a genetic algorithm as test case generator,
- Development of an instruction tracer for combined static and dynamic timing analysis.

The assignments are solved in groups of 3-4 students, and they provide a broad enough basis for many research topics. Each student chooses from the subjects of the module a research topic for the research paper, according to own preferences. Example submissions are:

- Priority-driven Algorithms for Safety-Critical Systems.
- Timing Analysis for Real-Time Systems.
- Using Advanced Genetic Algorithms to Improve Dynamic Testing.
- Implementation of a Constant Execution Time Sorting Algorithm.

The students are very eager to come up with original ideas, which makes reading the papers a pleasure.

4.2 Lectures

The lectures are intended to provide all the background knowledge, terminology and concepts for carrying out the assignments and writing the research paper. There is traditional style face-to-face presentation, but a large amount of lecturing time is devoted to discussions, questions, summaries, ad-hoc exercises and small assessments, so-called mini-tests. The primary topics covered in the lectures are about

- How to perform system domain analysis and derive high-level system timing requirements,
- How to decompose the system timing requirements and distribute them over the individual components,
- How to realize and implement software components with timing requirements,
- How to perform static and dynamic timing analysis in order to derive an execution schedule,
- How to implement a schedule with the means of the platform used,
- How to deploy a system or parts thereof on the platform used,
- How to test components and a system with timing requirements,
- How to evaluate the afore-mentioned steps by taking a development process view, and
- How to communicate this evaluation to others in terms of paper writing (scientific writing).

The students are supposed to discuss most of the topics among themselves and provide definitions and explanations. Their opinions are collected to form a general picture of the terms and concepts (supported through the lecturer). Some of the concepts are suitable for exercises, e.g., worst-case path analysis for static

[9] It seems that either students make the effort and perform well, or they drop out.

timing analysis, so in each lecture there is either an exercise of 20-30 minutes or a mini-test. In mini-tests, which is 5 questions to be answered on paper, students can assess for themselves whether they have understood the most important topics of previous lectures, or whether they have grasped the essentials of an article that they were supposed to read as homework. Each lecture closes with a summary provided by the students.

4.3 Evaluation

The real-time systems course is now also in its third edition, and it has been improved considerably compared to the first time, in particular, with respect to the diversity of the lab assignments and the amount of student/student interaction in the lectures. The difficulty of the lab assignments has been increased considerably, but also more support is being provided. Initially, students had to do a lot more own programming work which meant that there was not enough time for trying out different solutions and performing more measurements. Although now the assignments are more complex and more difficult, the students can use more existing code that they can incorporate into their own developments which gives them much more leeway for experimenting.

The lectures are now much more interactive, and much time is spent on exercises and discussions. According to course evaluation sheets, the students appreciate the high level of interaction and the fact that they are asked to "perform their own lectures". This, of course, costs a lot of time, so that coverage of the subject had to be sacrificed for in-depth analysis of individual topics. In our opinion, this is not necessarily a disadvantage, since the students are now a lot more confident on the fewer topics covered.

Demanding a research paper as the single assessment of the module is a bit ambivalent for two reasons. First, students like to be awarded for whatever they have done, so completing assignments and not getting course credits for them seems odd. Their assignments could be collected and marked, but it would impose an extremely high working load upon the lecturer. Not marking them is simply a matter of budget. Second, in particular the first participants feared that their provided papers would not be up to standards, and they would fail the course because of their low expected paper quality (another reason for having the assignments marked). However, the submitted papers were very good; only 2 out of 45 submissions had to be resubmitted in the first year. All subsequent students have now access to the best papers submitted (4-5 papers every year) which gives them a much better idea of what is required to pass the module, so that in the second round all 42 submissions were of high quality. Despite the fact that many are struggling, almost all students like the idea of submitting a research paper. Most of them have never written a scientific report or a paper, and they appreciate this opportunity to exercise for their coming research assignment and master thesis project. Every paper is going through a peer review by the students themselves, based on quality criteria provided, they can improve their work, and then make the final submission[10]. A final review by the lecturer provides them with concrete comments for improvement. Such final assessment is feasible only because of the low number of submissions (around 40). For larger courses, this would definitely not work.

[10] This is the version to be marked.

5. DIDACTICS

New rules in the didactics and tuition development in Delft require from academic staff to participate in a training program called the "Basiskwalifikatie Onderwijs" (BKO – basic qualification tuition), a quality scheme carried out or set up in most universities in The Netherlands. It has been implemented in order to impose a measurable base-line for teaching activities throughout Delft University, but also among the other universities. The main modules of the BKO program communicate skills for activating students to engage in learning processes, course design for interactive learning, project oriented and problem-based learning, and assessment, but also social skills such as teaching in English, and inter-cultural communication. In addition, the program offers further education in developing digital learning environments, and self-assessment and personal development.

The University's didactic vision, and, hence, the focus of the BKO is on communicating and training skills for active and collaborative learning (ACL) [3, 4]. ACL promotes instructional techniques that help students engage in higher-order thinking and problem-solving activities during class or seminar, and collaborative and communicative activities involving other fellow students. Traditional university teaching focuses on making students "understand" a subject through transmitting information via lecturing the students. "Pouring knowledge into the students" depends on what the teacher does and says. It is heavily teacher-centered. In contrast, ACL focuses on what the student does, and how the student engages in learning activities. It sees the teacher more in the role of the moderator and supporter [22]. Here, the primary task of the teacher is to help create a positive learning context for the students and moderate activities that encourage students to engage in deep, or higher-order learning approaches [3]. Higher-order (deep) learning approaches help students to apply knowledge, hypothesize and reflect upon a subject, in contrast to lower-order (shallow) learning approaches such as merely memorizing, identifying, or describing subjects [2].

Both courses outlined here implement ACL principles according to the new university guidelines. Both apply student centered learning principles. Both the embedded systems course and the real-time systems course do that by forming small groups, cooperating in a large project. Additionally, the real-time systems lectures have many self-directed activities such as exercises, group discussions, and tests. Both courses put their main emphasis on the performance of the lab projects/assignments and use the lectures more for supporting the actual practical work. Overall, there is a great emphasis put on students carrying out their own research work in an interactive and stimulating environment.

6. CONCLUSIONS

In this paper we have presented the new Delft MS program on ES that is slated for the academic year 2006. With respect to the 40 EC mandatory part of the 120 EC program, the program focuses on embedded software, ranging from UML to VHDL, which is a departure from the more hardware-oriented approach, such as offered by the Delft CE program. Another feature of the mandatory program is its equalization of the students' prior educational backgrounds during the first part of the program in order to optimize the multidisciplinary education that underlies the ES curriculum. Launched for the coming academic year, no results are yet available. Yet, the authors are convinced this is a step towards

a curriculum that will meet the industrial and academic requirements of the embedded systems field in The Netherlands.

7. ACKNOWLEDGMENTS

The national 3TU Embedded Systems MS curriculum, is a joint effort by the three Dutch technical universities. The joint curriculum has been authored by Jan-Friso Groote and Arlène Louiza (Eindhoven University of Technology), Gerard Smit and Gerrit van der Hoeve (University of Twente), and Arjan van Gemund and Hans Tonino (Delft University of Technology).

8. REFERENCES

[1] ARTEMIS. European Platform for Advanced research and Technology for Embedded Intelligence and Systems. www.cordis.lu/artemis.

[2] J. Biggs and P. Moore. The Process of Learning. Prentice Hall, 1993.

[3] J. Biggs and P. Moore. Teaching for Quality Learning at University. Prentice Hall, 2003.

[4] C. Bonwell and J. Eison. Active learning: Creating excitement in the classroom. ASHE-ERIC Higher Education Report No. 1, George Washington University, Washington, 1991.

[5] Delft University of Technology. www.tudelft.nl.

[6] S. Edwards. Experiences teaching an fpga-based embedded systems class. ACM SIGBED Review, 2(4):56–62, 2005. www.cs.virginia.edu/sigbed/vol2_num4.html.

[7] Eindhoven University of Technology. www.tue.nl.

[8] Embedded Software Lab. www.rtess.ewi.tudelft.nl.

[9] Faculty of Electrical Engineering, Mathematics, and Computer Science. www.ewi.tudelft.nl.

[10] A. v. Gemund. In4073 Course Web Site. www.st.ewi.tudelft.nl/~gemund/Courses/In4073/index.html.

[11] A. v. Gemund. In4073 Resource Web Site. www.st.ewi.tudelft.nl/~gemund/Courses/In4073/Resources/index.html.

[12] A. v. Gemund and J. Tonino. Master of Embedded Systems, Information Dossier in Support of the Application for the New Graduate Studies Test. Technical report, Faculty of Electrical Engineering, Mathematics, and Computer Science, Delft University of Technology, Delft, The Netherlands, Sept. 2005.

[13] H.-G. Gross. In4024 course web site. www.st.ewi.tudelft.nl/gross/index files/Page348.html.

[14] IEEE-CS/ACM. Computing curricula 2005: The overview report. www.computer.org/education/cc2001.

[15] ITEA. Technology Roadmap for Software Intensive Systems 2nd edition May 2004. www.itea-office.org/itea_roadmap_2.

[16] D. Jackson and P. Caspi. Embedded systems education: Future directions, initiatives, and cooperation. ACM SIGBED Review, 2(4):1–4, 2005. www.cs.virginia.edu/sigbed/vol2_num4.html.

[17] J. Muppala. Experience with an embedded systems software course. ACM SIGBED Review, 2(4):29–33, 2005. www.cs.virginia.edu/sigbed/vol2_num4.html.

[18] TU Delft. Ms embedded systems elective courses. www.sis.tudelft.nl.

[19] Politecnico di Milano. Rtai - the realtime application interface for linux. www.rtai.org.

[20] A. Sangiovanni-Vincentelli and A. Pinto. Embedded systems education: A new paradigm for engineering schools? ACM SIGBED Review, 2(4):5–14, 2005. www.cs.virginia.edu/sigbed/vol2_num4.html.

[21] R. Seviora. A curriculum for embedded system engineering. ACM Trans. Emb. Comp. Syst., 4(3):569–586, August 2005.

[22] T. Shuell. Cognitive conceptions of learning. Review of Educational Research, 56:411–436, 1986.

[23] University of Twente. www.utwente.nl.

[24] S. Woutersen and A. v. Gemund. X32 Web Site. Via www.st.ewi.tudelft.nl/~gemund/C6/index.html.

Experiences from large embedded systems development projects in education, involving industry and research

Martin Törngren
Dept. of Machine Design, Division of Mechatronics, Royal Institute of Technology - KTH
SE-100 44 Stockholm, Sweden
+46-8-790 6307
Martin@md.kth.se

Martin Grimheden
Same affiliation as 1st author
MartinG@md.kth.se

Niklas Adamsson
Same affiliation as 1st author
NiklasA@md.kth.se

ABSTRACT: We present experiences from a final year M.Sc. course. The overall aim of the course is to provide knowledge and skills to develop products in small or large development teams. The course is implemented in terms of large projects in cooperation with external partners, in which the students, based on a product specification, apply and integrate their accumulated knowledge in the development of a prototype. This course, which has been running and further elaborated for 20 years, has been proven successful in terms of being appreciated by the students and by the external partners. The course has during the recent years more frequently been carried out in close connection to research groups. Our experiences indicate benefits by carrying out these types of large projects in an educational setting, with external partners as project providers, and in close cooperation with research groups.

Having external partners as project providers feeds the course, students and faculty with many industrially relevant problems that are useful for motivational purposes, and in other courses for exemplification and for case studies in research. Carrying out the projects in close connection to research groups provides synergy between research and education, and can improve the academic level of the projects. A further interesting dimension is accomplished when the projects run in iterations, requiring new groups of students to take over an already, partly developed complex system, and work incrementally on this system. The students are then faced with a very typical industrial situation. We advocate that students should be exposed to a mixture of "build from scratch" and "incremental" projects during the education.

1. INTRODUCTION

1.1 Challenges for Embedded Systems Education and paper outline

The industrial globalization and the competitiveness made possible by embedded systems based products, makes it more important than ever for the educational systems to provide society with competent engineers in the area of embedded systems. However, accomplishing this is a highly challenging task.

The technology and products of the area are evolving rapidly, with products broadening from simple stand-alone measurement/-controlling devices to systems that are internally distributed and also increasingly connected to other devices and users over wired and wireless communication links. The services of the systems are also evolving to include advanced and autonomous functions. The resulting products and systems are as a consequence becoming more complex. The increasing complexity requires that a systems engineering approach is adopted for successful product development, an approach that emphasizes several dimensions including product architecture, enterprise organization and processes. Due to the relative novelty of embedded systems, there is also no established science for embedded systems nor documented pedagogical methods for educating embedded systems engineers. The wide variety of types of embedded systems, with different quality requirements, technologies, and multiple design dimensions and parameters, makes it difficult to agree on a suitable definition on what an embedded system is and how to teach it in a suitable way, (see for example [6]).

At the first Workshop on Embedded Systems Education in Jersey City, U.S., in 2005, two directions and ways to approach embedded systems education were identified:

- *Scientific foundation approach*. In this approach, presented by for example [11], it is suggested that the education has the aim to teach engineering knowledge that lies at the core of embedded systems, including issues such as models of computation and formal analysis.

- *Product development approach*. In this approach, presented by for example [4], the education has many ingredients that are corresponding to industrial product development including project oriented work, team work and problem solving manifesting in a functional prototype.

We consider these two approaches to be complementary, where the former emphasizes knowledge in science and technology, and where the latter provides knowledge and competence in actual system development where several processes and product development activities, as well as team work are central.

The product development approach clearly relies on knowledge acquired in earlier courses such as in mathematics, automatic control, computer science and mechanics.

In this paper we elaborate further on education following the product development approach and focus in particular on a larger project oriented course which is introduced in Section 1.2. In Section 2 we give examples of course instances and in Section 3 discuss experiences from this type of courses from several different viewpoints.. Finally, we conclude in Section 4 by discussing how we would like to further evolve these types of courses and exploit the concept throughout the engineering education.

1.2 Project courses in Mechatronics and the capstone course

In this paper the main emphasis is on a capstone course given by the Division of Mechatronics at the Department of Machine Design at KTH. This course, entitled "Advanced course in Mechatronics" (course no: 4F1161, 22,5 ECTS credits), involves a project with a nominal effort corresponding to 60% full time studies over one full semester. The division of Mechatronics at KTH has been conducting such courses since 1984. The overall aim of the course is as follows:

After the course the student will have knowledge and skills to develop mechatronic products in small or large development teams.

The underlying motivation for the course is based on the fact that the development of complex/advanced products including embedded systems requires an understanding of the problems involved in the corresponding complex development projects. Thus, apart from specific tasks such as design of mechanics, software, control systems and electronics, central ingredients of the course include requirements engineering, systems architecting and integration, risk management, time and resource planning, interactions within the project team, and with suppliers and the task provider.

The hypothesis behind the course goal is consequently that the corresponding knowledge and skills are best acquired by actually having the students be part of real development projects (before the capstone course, the students have participated in courses which involved smaller projects).

This hypothesis is supported by research on problem based education [2][4][10][13].

Another motivation is provided by Grimheden and Törngren, [6]. Based on the "didactical analysis" they conclude that the subject of embedded systems has a thematic identity and a functional legitimacy, which implies that an exemplifying educational method is preferable, in an interactive setting. An exemplifying selection is facilitated by a problem based setting; each course is built upon a project, the design of a product or a system, for example a control system of an autonomous robot. The functional legitimacy is facilitated by a project organization and problem based setting as well, each student is responsible for the design of a subsystem and will, during a number of projects, acquire a certain set of skills such as project work and administration, PCB design, implementation of control algorithms etc. The motivational factor is further increased by giving the students a high degree of economical responsibility in the projects and by carrying out the projects in collaboration with industry.

Embedded systems typically represent the core enabling technology in the products that are developed. The projects are multi-technological since they also most often involve mechanical design/packaging, sensing and actuating technologies. The main emphasis in the course, however, is on product development.

In the capstone course, the projects are organized as development projects typically resulting in a functional prototype.

The guiding principles for these projects are as follows:

- Students manage the project and take own responsibility as far as possible. Students also manage contacts with industrial partners, suppliers and external contacts.
- During the project, rotation of responsibilities among students takes place in terms of management and technical work roles.
- The educational goal has priority during the project, but has to be balanced with the project/prototype goal
- The project is coached by one to two persons from the academic staff. The role of the coaches is further discussed in Section 3.

It is considered preferable that the project provider is external to the University, since this by experience provides realistic problems and also separates the roles of task provisioning and task coaching/supervision (and examination).

Because of an increasing number of students over the years, starting with about 10 participants in 1984 and with about 40 in 2006, the students are divided into several projects. In 2006 for example, the course was divided into four projects, each coached by one to two faculty members, and each with an external project provider. Complementing the project, there are also parallel common activities which include joint teaching, seminars, and social activities. These activities take place across the projects.

The capstone course, as well as other courses given by the Division of Mechatronics, are currently taken primarily by students from the programs of Mechanical Engineering, Vehicle Engineering, and Industrial Engineering and Management.

Earlier experiences from these courses have been reported in a number of previous publications; see for example [4], [5], [6].

Over the years, more than 50 projects have been performed, see Table 1 for a list of example projects.

Table 1. Example of capstone course projects, 1984 - 2006

Project	Task	Provider (main)
SAINT1+2	Automotive software configuration and platforms	Scania / KTH
Mucca	Cow milking robot	De Laval
FAR	X-by-wire architectures and model based development	Volvo Car Corporation / KTH
WARP	Four-legged robot and its control system	KTH researchers
Xless	Wireless communication in train distributed control systems	Adtranz
Agilis	Fuel-efficient car	Shell-Eco Marathon
Balance	Prosthesis and aid for human balance control	Boston University / Harvard Medical School
PBLX	Reduction of wiring in cars	General Motors

2. Example course project instances

2.1 FAR

In 2002, 30 students specialized in Mechatronics at KTH. Three projects were defined as part of the capstone course, one of them being the FAR project.

The overall purpose of the project was to demonstrate a suitable tool-chain environment, based on adapted existing environments, that supported model based development of embedded control systems and in particular Function and ARrchitecture (FAR) integration. As part of the project, a model X-by-wire car demonstrator and its distributed control system were developed. The project was formulated in close connection to ongoing research projects at KTH, Chalmers and Volvo Car Corporation (VCC). The overall budget for the project was limited to approx. 40k€ which should cover all material and software development costs. KTH and VCC financed academic and industrial staff.

10 students were selected for the FAR project. The project was coached from KTH by one PhD student and one senior faculty member. One research engineer at VCC was responsible for the specifications and contacts with the project team at KTH. In addition, an M.Sc. project involving two students at Chalmers was connected to the project by developing an alternative human machine interface. The project started in November 2002 and ended in June 2003. At the project end the final demonstrator was delivered to VCC (the normal procedure in these courses is to develop a prototype that is handed over to the project provider.

The project adopted a four stage development process. In the initial phase the goals and constraints (e.g. budget and timing) of the project were identified and assessed. Available technology was studied to get input to feasible solutions. The choices of technology and tools were to a large extent limited by the budget and availability of technology. Project as well as product risk analysis were performed throughout the project. An example of the relevance of this is as follows. In the middle of the project, the team faced the challenge to change the networking technology because of an agreement problem with one technology provider. The project team managed in the short time available to change technology. The system specification phase addressed overall solution approaches and structures for functionality, mechanics and electronics hardware were developed. The final phases of the project focused on module development and step-wise integration.

Competition model cars were used as a basis for a mechanical re-design where the individual wheel steering, braking and driving required special attention. The car can be programmed into several modes of operation including manual driving, cruise control and with a simplified collision avoidance functionality. Figure 1 illustrates the hardware components of the demonstrator.

2.1.1 Experiences from the FAR project

The FAR project indicated the advantages of running this type of student project in close connection to research. The project was one of the first (if not the first) in the world to deploy a TT-CAN network inside a (model) car. The result provided a useful case study in complex systems development, for example adopted in the EAST-EAA project to evaluate modeling concepts for automotive embedded systems, [9].

The software architecture, illustrated in Figure 2, and development approach were seen as quite promising in that it combined component based and model based development with support for distributed systems through the global TT-CAN based clock and the off-line scheduling facility in the Rubus RTOS, [12]. The demonstrator vehicle is still in use at VCC for work on dependable embedded system platforms, see e.g. [7].

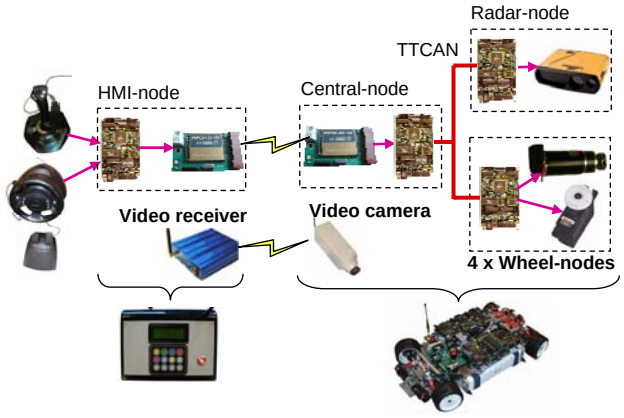


Figure 1. Hardware part of the FAR demonstrator, illustrating its distributed control system, where the nodes on the vehicle are connected through a TT-CAN network.

The students responded positively to the project and the experiences it conveyed. The students however also remarked that the workload and goals of the project were set too high. It became clear that the project had a too large scope to meet all the goals set out in the beginning. Although a model car was successfully developed, the desired application level functionality, e.g. collision avoidance could only be rudimentary completed within the duration of the student project. Also, the tool-chain to support distributed control systems development could only partly be tested.

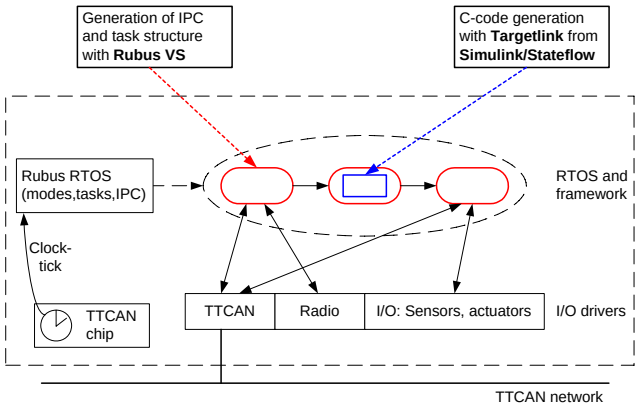


Figure 2. Software architecture of the FAR demonstrator illustrating separation of concerns between application functionality and platform, and functionality vs. timing..

These limitations were largely dependent on the limited duration and resources of this project. Given a larger student group, or increased participation of researchers, would have made it possible to push the results somewhat further. Another idea is that of working incrementally on the demonstrator, this gave rise to an idea forming the basis for the SAINT projects.

2.2 SAINT1 and SAINT2

The SAINT (Self Adaptive INTelligent truck) project was formulated with a basis in a joint research project between the industrial partner Scania and KTH, and with the experiences of the FAR project in mind. The development of a common demonstrator was of joint interest. It was early decided that the demonstrator and its environments would remain at KTH after development. The budget was limited to approx. 30k€.

The research topics initially considered for study with the demonstrator included function and software configuration, and software platforms/architecture. Whereas modular development of mechanical systems, such as trucks, is rather well studied and supported by established Product Data Management (PDM) tools, the same is not true for software. A current problem in the automotive industry is to fit software efficiently into the product structure and thereby deal with variants and configuration of the EE (Electrical/Electronic)-system. To better support design, reuse and maintenance, there is a need to manage not only hardware and binaries corresponding to software. Traceability of software code, design and analysis models, as well as to requirements becomes increasingly important along with the increasing complexity, [8].

The main purposes of the project included to:

- develop a truck model including its distributed control system
- evaluate the use of PDM tools for function and software configuration, mainly for the production process
- develop a prototype software platform supporting location transparent execution.

The intention was to continue the earlier development of the SAINT demonstrator. This also turned out to be possible, resulting so far in two subsequent student projects where the second one took over a partially completed demonstrator and finalized its development. The two subprojects are now briefly described.

2.2.1 The SAINT1 project

In 2004, there were 40 students specializing in Mechatronics at KTH. Four projects were defined as part of the capstone course, one of them being the SAINT1 project. 16 students were selected for SAINT1. The project was coached by two faculty members and one industrial PhD student with Scania/KTH.

Given the large scope of the project, the coaches spent quite some time in preparing and evaluating the PDM system such that it could easier be used by the students. The students were also helped in starting to use UML for function modeling, primarily using activity diagrams. Nevertheless the students were faced with the challenging task to develop the mechanics, electronics, software, functions and adapting the tool environment.

The results of the SAINT1 project were quite satisfactory. The truck and its control system, together with a middleware based on the ENEA OSE RTOS were developed together with basic functions. It was demonstrated that the chosen PDM system was adequate for software configuration purposes. In addition, the complete truck was made operational, [3]. See Figure 3 for an overview of the truck.

However, due to the time limitation only basic functionality and one simpler configuration scheme was possible to develop.

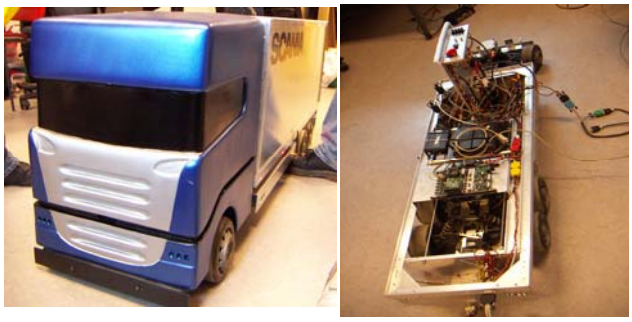


Figure 3. The SAINT demonstrator after the SAINT2 project. (left), built in scale 1:6, and under the hood (right), including eight microprocessors, CAN networks, electrical motors, laser and ultra-sonic sensors..

Given this situation and the willingness to continue the project, SAINT2 was formulated.

2.2.2 The SAINT2 project

The goals for the SAINT2 project were essentially to develop more advanced vehicle functions and a more elaborated function/software configuration scheme (essentially those tasks that were not finalized in SAINT1). It was decided to focus on longitudinal motion control and active safety functions including cruise control, adaptive cruise control, anti-spin, emergency brake and collision avoidance. For the configuration, the goal was to be able to perform a complete function/software configuration connected to the PDM database, upon which a software tool automatically would select the appropriate software, allocate it to control units, and then build and download the software for flashing to the appropriate units.

Six students, out of 24 students during the capstone course in 2005, were selected for the SAINT2 project which was coached by one faculty member and one PhD student, in cooperation with Scania. The relatively small number of students was due to the fact that the course that year had fewer students than usual, but still had four projects competing for students.

The results of the project were very encouraging. With the second iteration, a demonstrator including the vehicle and its configuration environment was achieved. Moreover, the educational goals were met and the demonstrator development gave rise to a multitude of new ideas for further exploration.

2.2.3 Experiences from the SAINT projects

In terms of education, the main novelty of the SAINT projects was to perform the projects in an incremental fashion.

When the SAINT2 project idea was first presented there was some initial hesitation. For some of the students the project appeared less interesting because the construction of a mechatronic device including mechanics, electronics and software was already to a large extent accomplished in SAINT1. An earlier negative experience in running a capstone project with the same external partner in two consecutive years resulted in some hesitation from other faculty members. The results of the SAINT2 project clearly indicate the potential of incremental projects. From an educational viewpoint the results were very satisfactory. The students at the beginning of the project faced huge amounts of

(not always up to date and consistent) documentation and quickly learnt the value of documentation. The students were faced with multiple subsystems, components, functions and development tools which they had to learn how to use. This was a great challenge and it was not until after approximately 50% of the project time that the students gained control of the project. This experience was in addition to welll known problems of stabilizing and agreeing on the requirements and goals of the project.

However, having crossed this "take-over" barrier, the project team could build upon the already developed platforms – many of which proved possible to build further upon. As a consequence, it was possible in a few months time to develop examples of advanced functionality and to greatly enhance the configuration environment. As a side effect, the complexity of the demonstrator quickly increased due to the more advanced distributed control functions. Although the coaches knew this in advance, the functionality given the support from the existing platforms including the middleware grew faster than they anticipated, resulting in difficulties in distributed systems debugging (feature interaction and multiple fault sources). This turned out to be a good educational experience although it somewhat hampered the completion of the demonstrator. These problems can be alleviated with better support tools for development and debugging.

The external partner acknowledged these experiences as a suitable introduction to real-world engineering. In addition, the project provided technical feedback to the external partner regarding the particular technologies (PDM tools, RTOS, networking tools etc.) used.

The SAINT projects were conduced in close connection to research at KTH. This had the benefits of strongly motivating research personnel at KTH to participate in the teaching activities. As compared to the FAR project, more effort was spent on preparing tool environments for the students. The experience is that this was necessary for the success of the project.

In the SAINT2 project, a PhD student was involved also as subsystem supplier for the project. From the point of view of the project, the involvement of a researcher in the development work was easy to motivate with the small project team. The integration of the subsystem was successful and the example indicates a way in which PhD students, and teams of students, usefully can be part of the same project.

A continuation, in terms of a third SAINT student project is being planned. SAINT is also considered as a strong candidate for case studies and demonstrators in several ongoing research projects.

2.3 Boston balance projects

The Boston balance projects are examples of a capstone course project performed in an international setting, with a large number of spin-off projects and products. The original project was formulated in 2001 between KTH and two partner universities in Boston; the Neuromuscular Research Center (NMRC) at Boston University and the Massachusetts Eye and Ear Infirmary (MEEI) at Harvard Medical School. The project attracted twelve KTH students, two KTH faculty members, or coaches, and two Boston faculty members plus a number of Boston students, associates and corporate liaisons from interested companies.

The goal of the Balance project was to develop a prosthesis for use with people with a balance disorder, either relating to a malfunctioning balance organ in the inner ear or loss of sensory impressions from the soles of the feet. Both NMRC and MEEI had rudimentary prototypes realizing aspects of the prosthesis, and the aim of the capstone course project was to integrate these ideas and produce a wearable prototype.

Both NMRC and MEEI perform research in the area of balance prosthesis but had not before collaborated in this sense, neither had access to development resources capable of developing advanced prototypes. The aim of the prosthesis was therefore mainly to enable research on algorithms for balance control as well as feedback signals, sensors and filtering techniques.

The students developed a prototype based on five distributed microcontrollers, 72 actuators, 18 pressure sensors, accelerometers, gyros and CAN communication. The project was organized in four phases, each ending with a cross-atlantic trip – either to discuss ideas or to present results.

After delivery of the prototype, both collaborating institutes continued to use and perform research on the new device.

2.3.1 Spin-off projects

During the four years that has passed since project delivery, a constant collaboration between KTH and NMRC and/or MEEI has been maintained. Usually in the form of KTH students performing M.Sc. projects at one of the partner institutes, but NMRC/MEEI has also hired a number of current and former KTH students for work for between a few weeks up to three years, and academic staff from KTH has also spent sabbaticals at NMRC/MEEI. The balance prosthesis is therefore constantly further developed, and the high level of competence is maintained at both NMRC/MEEI and KTH to enable supervision of new projects. Currently there are even discussions on establishing a KTH center in Boston, to provide a base and permanent accommodation for visiting KTH students and faculty.

2.3.2 Experiences from the balance projects

The five years of collaboration has provided a win-win setting for all partners. The M.Sc. thesis projects offered to the KTH students are highly attractive, both since those enable the students to study in the U.S., but primarily since all projects are highly product development oriented and mostly incremental, and since high competence in the area exists both at KTH and with the partners. Each new student develops an aspect, a new module or a new function for a system that is thoroughly documented.

The partner institutes benefits greatly from being able to both receive skilled students capable of directly contributing, but also from being able to hire professional staff skilled in the area of balance prostheses. Several students so far have been offered a full time job at one of the institutes.

3. Experiences from the capstone course

The capstone course has largely been a successful activity. Key factors in accomplishing a successful course and education scheme include the selection/specification of an appropriate project and to have well motivated stake-holders involved in the project (students, coaches and the project providers). Given a well motivated project provider that has sufficient resources and time for the project, often ensures that the students become motivated.

In general, the course is well received by the students. In one evaluation all students that graduated between 1984 and 1994

were asked to specify which courses in their engineering education that they considered most valuable based on their current work as (Mechatronics) engineers. The answer was that the capstone course was deemed as the most valuable course. This is probably related to the fact that the students in the capstone course are both expected to take a large responsibility themselves, but also to the idea that students are expected to utilize knowledge and skills from previous courses rather than being taught new areas – so the course cannot realistically be compared to previous courses in terms of usability.

The course is also well received by the external partners providing the projects. For the Swedish industry, the project courses are seen as a means for recruitment, connection with academic research, and as a means to obtain resources for (cheap) prototype development. Many of the prototypes developed in all the course instances are still in use, including the examples mentioned in Section 2.

However, some projects and course instances have also been less successful and there is still room for improving the course concepts. In this section we first present some general prerequisites and experiences with respect to successful project course implementation. Section 3.2 then discusses the connections to research and use of incremental projects.

3.1 General experiences and prerequisites

There are a number of issues that have a large impact on and influence the outcome of the mentioned type of project-oriented courses. We here discuss a selection of these (the reader is referred to [4] for more detail).

3.1.1 Conflicting goals: Prototype vs. education

In basically all projects the coaches and students experience conflicting goals. Typically at the end of the project, students are eager to put more emphasis on the project goals than the educational goals. A typical example would be that, in a critical moment, the task of finalizing the software is given to the student most experienced in programming – not the student most in need of programming practice. However, project progress also motivates the educational goals. A focus on project goals helps motivate students, which then become more easily subjected to learning. The key is therefore to manage the balance between keeping a constant development pace in the project, but also by having all students subjected to exposure from unfamiliar areas and making sure that all students get enough time to learn, reflect and practice new ideas. Team management is therefore crucial, which requires a skilled coach.

As in all projects, the balance between the project goals, the available time and project resources (personnel, budget and technology) is central.

3.1.2 Homogeneity/heterogeneity and sizes of groups

Embedded systems course projects are facilitated by, and benefit from, diversity and heterogeneity. This is due to the fact that the students both learn from each others experiences and mainly from the larger number of courses and areas covered by the students combined backgrounds. The KTH projects usually attract students from at least three different M.Sc. programs which together with exchange students usually provide expertise in most covered areas. When creating the teams, a large effort is also made on gender and cultural diversity.

The size of the groups has varied from six to sixteen during the last few years, depending on both the total number of students and the scope of the projects. A large number of students usually requires a more experienced and skilled coach/supervisor, but from experience a small team might also develop conflicts and could require large supervisory resources. A larger team though gives the possibility to introduce sub-teams which could be rotated and changed periodically, which from a supervisory point of view makes the larger team advantageous, even if a small team on some occasions might manage so well on it's own that supervision is superfluous.

3.1.3 Team work challenges

The objectives for the capstone course state that some team work related aspects should be covered within the content of the course. We have from our experiences found that, with our strong technical background and educational traditions, it is sometimes hard to provide sufficient support for the students to cope with such things as time and resource planning and effective management of the teams.

We therefore believe that one of the major challenges is to find a balance between providing the students with knowledge and capabilities about managing an R&D team at the same time as providing them with an environment which fosters sophisticated technical knowledge.

3.1.4 International projects

One of the three projects described in Section 2 was performed in an international setting. Currently, the aim is to have at least one project every year, either in collaboration with a foreign university of with a foreign corporate sponsor. The purpose is to educate Mechatronics engineers for the future market; as of today an increasing number of Swedish companies hiring Mechatronics engineers are working on a global market. Engineers thus have an advantage with project work experiences involving an international setting.

The usual method to reach these aims is to spend some time abroad: as exchange student, in an exchange project or to perform a Master's thesis project abroad. In this context however, the capstone course project aims at giving all participating students a similar experience, but without the need to travel.

The international projects have proven to be successful but simultaneously require considerable more resources in terms of coaching, equipment and financing [4]. The projects usually benefit from travels and international meetings, and equipment such as videoconference technology is necessary.

3.1.5 Coaching

Seen historically, faculty supervision has been replaced by team coaching. This is primarily an effect of a larger transformation of higher education: from faculty teaching to student learning, from university- to student responsibility, from lecturing to problem based learning [2]. Coaching means guidance rather than directing, helping rather than telling and basically making the individual student perform at his or her best.

The coach act as mediator between the faculty and the student team, provides resources and directs students towards appropriate faculty experts. A highly valuable property of the coach is his/her previous experience of product development projects and the ability to manage the team in the different phases of the project.

As described in Section 3.1.1, the most difficult task of the coach is to balance between the project aims and the educational aims.

At the Division of Mechatronics at KTH, most PhD students and faculty members have acted as coaches at one time or another, but the experience is that this coaching requires much more effort and consideration than what is apparent, and requires industrial experience and years of training and apprenticeship to master.

3.1.6 Grading

Grading is currently subjected to change at KTH, depending on the adaptation process toward the common European system. Previously, the students were basically only given grades of pass or fail, and only students dropping out failed. Today, grades on the ECTS scale, from A to F are given. A considerable amount of research has been performed in this area however, and the actual grading is not as difficult as to design educational aims for each grading level, and to communicate these to the students well in advance.

Some help can also be found in the continuous documentation and presentation processes as part of the project, which can be used as a platform for grading of the entire team. The most important aspect though is to have a continuous discussion between the students and the coach, about the intention of each student and the requirement asked by the faculty to reach the respective aims.

Usually, the process of grading is kept separate from the actual coaching. This has not been the case at KTH so far, but will most probably be adopted in the near future.

3.1.7 Course evaluation and evolution

The number of students applying for the course increased in a rather linear fashion from 1984, with 12 students, until year 2000 with 40 students, and since then the student interest has decreased slightly every year. The same decreasing trend has been identified in most engineering programs in Sweden. Regarding course evaluations and regular improvement conclusions are difficult to make since the course is heavily dependant on the industrial partner; the partner's area, engagement and the scope of the project. Our experience is also rather that course quality is more a matter of finding the right course coaches and assistants. In some cases, the projects have been closely related to research projects and successfully coached by doctoral students. This further motivates a stronger connection between research and capstone projects, thereby creating synergistic effects between researchers and student projects.

3.1.8 Industrial partner prerequisites

Our experience from industrial partners vary from foreign to local partners, from one-man-companies to large global companies and from companies who show slight interest for either the process or the results to companies who participates daily in the educational activities. Our experience is that neither the locality nor the size of the company matters, neither the field of the company. What matters most is the engagement and interest of the company and primarily the corporate liaison. In many cases this person is a former student of the capstone course, which usually guarantees a good relationship. Of high importance is also the ability of the partner to provide funding for prototypes, tools etc as well as access to the companies own resources.

3.1.9 Student preparation

As to student preparation and requirements, the discussion seems to be everlasting. On one side people advocate for more problem-based courses early on in the education, and on the other side people are considering that more focus on application and holistic courses will reduce the overall disciplinary content of the educational programs. It is obvious that a firm theoretical background is important and has to be established in the early university years. However, this does not exclude having problem-oriented courses as an important complement to provide a systems and synthesis perspective. Regarding the capstone course however, diversity is advantageous, advocating not for a single curriculum in embedded systems but rather that the capstone projects benefit greatly from accepting students with various backgrounds, courses and educational tracks.

3.2 Research connections and incremental projects

Over recent years we have noticed a synergetic effect when the projects are aligned or connected to research projects/themes. We have also noticed that incremental projects have a complementary nature as compared to projects that build from scratch.

3.2.1 Running the projects connected to academic research

Aligning the projects with research themes and topics at KTH makes it easier to motivate and involve research staff and doctoral students in the projects. For the researchers a student project can provide extra resources in developing a research prototype. This also motivates extra work in preparing and participating in the project. For example, the SAINT capstone projects described in Section 2 were part of a larger research project where the involved researchers coached the project, assisted in setting up supporting tools, and also in developing the solutions.

We believe that such synergetic efforts can be very important today for the academic staff in view of the overloaded (extremely efficient) University system and to achieve efficient economy and resource utilization.

Connecting the capstone projects to research projects also ensures a connection to state of the art, which can improve the motivation for the students. From a methodological point of view, the research connection has two additional strong benefits:

- Involving researchers in the projects can assist in the adoption of a systematic approach in the development of complex products by e.g. use of systems engineering knowledge and embedded systems theory.
- For the researchers, the projects are an excellent opportunity to learn, and to apply theories on system development and research results on realistic systems. The projects and products developed can also be reused later in research for case studies and demonstration purposes.

As partly exemplified in Section 2, we have had projects that have been connected to external research projects (the Boston projects), cooperative research projects (e.g. SAINT) and internal research projects (e.g. WARP). It has been easier to ensure motivation and a separation of responsibilities when the project/task has been provided by an external partner.

Is there a risk of involving researchers too much in the projects? Similar to industrial prototype development, it is essential that the researchers have sufficiently clear ideas and plans for the research prototype to be developed before a project is started; this to avoid too large project risks. A complete prototype failure may jeopardize the educational goals. There could potentially be a risk of having the academic staff too much involved, but in our experience it has rather been the opposite way – we probably could have included the researchers even more.

3.2.2 Incremental projects

In the examples presented in Section 2, advantages and disadvantages of both incremental and non-incremental projects are discussed. Non-incremental projects provide a strong incentive and satisfaction for the students where they, starting from scratch, end up with a working product. The incremental projects, on the other hand, provide the benefit of accomplishing a more complex and high quality prototype and meet the industrial need for this type of common development.

As with all development processes though, incremental product development fosters incremental innovation rather than radical innovation. One solution is to provide projects varying on the scale from radical to incremental and to clearly express the differences to the students prior to choosing projects.

4. CONCLUSIONS

This article puts forward a number of issues dealing with problem based learning in embedded systems education, and specifically in the setting of large capstone projects in collaboration with industry and academic research. The intention by the authors is to investigate and increase synergistic integration between academic research in collaboration with industry and engineering education, and to describe and analyze the difficulties and possibilities.

The capstone course projects performed at KTH have varied from radical product development to incremental projects spanning over several years, and among our conclusions are the fact that the incremental projects have proved greatly beneficial in terms of synergy with academic research and long-time collaboration with industrial partners. Some students express initial concern at joining an incremental project rather than starting from scratch, but in the end consider the incremental project advantageous since this project usually contains increased complexity and more advanced applications. In accomplishing incremental projects there is the need for long term arrangements; this can be a challenge if the projects involve external partners. Although our experiences indicate a benefit with external providers of projects, the benefits of incremental projects may still be valid given internal project providers – this is a topic for further investigation.

In a way the project oriented courses, when they take place at the end of the education, act as a kind of exam where the student in their work will reveal what knowledge they really have acquired and are able to use in a concrete development project. This has given us some feedback for earlier courses we are providing, and is one track which could be of interest for further investigation. We believe that achieving synergy between research, education and industry will be of increasing importance for the academic system – and that this topic also deserves further investigation.

Finally, we advocate that students should be exposed to a mixture of build from scratch and incremental project, and that this type of problem based education should be adopted as one of the educational forms throughout the engineering education.

5. ACKNOWLEDGMENTS

We would like to thank our students, colleagues and external partners who have contributed in making these courses a success.

6. REFERENCES

[1] ARTIST2: The European Network of Excellence on Embedded Systems Design. http://www.artist-embedded.org/FP6/ (accessed August 1, 2006).

[2] Barr, R. B., Tagg, J., A new paradigm for undergraduate education. Change, 27, **6**, 1995.

[3] Blixt D, Brikho S, Bråkenhielm E, Cedergren U, Cronebäck Ö, Edvinsson L, Eloranta T, Forséll S, Hallberg M, Karlsson N, Olsson A, Rödén M, Steiner A, Wängdahl J, Öhlund D, Öhrvall M. 2005. Project SAINT, Technical Report TRITA-MMK 2005:26 ISSN 1400-1179. Royal Institute of Technology, KTH, Stockholm, June 2005. (In Swedish)

[4] Grimheden, M. Mechatronics Engineering Education, Doctoral Thesis, Royal Institute of Technology, Stockholm, Sweden. TRITA – MMK 2006:1, ISSN 1400-1179, 2006.

[5] Grimheden, M., Hanson M. Collaborative Learning in Mechatronics with Globally Distributed Teams. International Journal of Engineering Education, 19, **4**, 569-574. 2003.

[6] Grimheden M., Törngren M. What is embedded systems and how should it be taught? – Results from a didactical analysis. ACM Transactions on Embedded Computing Systems; Special Issue on Education, Vol. 4, Issue 3, August 2005.

[7] Johannessen, P. On the Design of Electrical Architectures for Safety-Critical Automotive Systems. PhD thesis, 2004. Diss. CPL 2334, Dept. of Computer Eng. Chalmers.University.

[8] Larses Ola. PhD Thesis. Architecting and Modeling Automotive Embedded Systems. Dept. of Machine Design, KTH, Stockholm. TRITA – MMK 2005:31, ISSN 1400-1179, ISRN/KTH/MMK/R-05/31-SE, Nov. 2005.

[9] Lönn H., Tripti S., Törngren M., Nolin M.. FAR EAST: Modeling an Automotive Software Architecture Using the EAST ADL. Workshop on Software Engineering for Automotive Systems, 26th Int. Conf. on Software Engineering, May 2004.

[10] Leifer, L., Design Team Performance: Metrics and the impact of technology. In: Brown, S. and Seidner, C. (eds), Evaluating Organizational Thinking. Kluwer, 1998.

[11] Pinto A. and Sangiovanni-Vincentelli A.. An overview of embedded system design education at Berkeley. ACM Trans. on Embedded Computing Systems (TECS); Special Issue on Education, Volume 4, Issue 3, August 2005.

[12] Törngren M., Adamsson N. and Johannessen P.. Lessons Learned from Model Based Development of a Distributed Embedded Automotive Control System. SAE World Congress, Detroit, 2004. SAE paper no. 2004-01-0713

[13] Vernon, D., Blake, R., Does problem-based learning work? A meta-analysis of evaluative research. Academic Medicine, 68(7), 550-563, 1993.

The Windows Embedded Academic Program – Retrospective & Directions, 2002-2006

Lindsay T. Kane
Product Manager
Mobile & Embedded Devices Group
One Microsoft Way, Redmond
WA 98052, USA
+1 (425) 707 7800
lindkane@microsoft.com

D. Stewart W. Tansley, PhD
Program Manager
Microsoft Research
One Microsoft Way, Redmond
WA 98052, USA
+1 (425) 707 5686
stansley@microsoft.com

ABSTRACT
The Windows Embedded Academic Program (WEMAP) was founded in 2002 with the aim of bridging some of the gap between Microsoft's commercial training offerings for its embedded systems operating systems products, and the embedded systems teaching needs of higher education institutions. In addition, through partnering with Microsoft Research, it has explored how to make the Microsoft embedded systems products more accessible to the university research lab. This paper explores some of the history and lessons learned from the first four years of this program, summarizes the current and future directions of the program, and seeks to encourage a dialogue – through the vehicle of the WESE 2006 workshop and beyond – between professors active in teaching embedded systems and Microsoft embedded systems representatives, who are keen to shape the program to meet the needs of academia and industry alike. The paper presents the joint personal perspectives of the authors who have both been closely involved in the program, and does not necessarily represent an official position of Microsoft Corporation.

1. INTRODUCTION
Commercial embedded systems operating systems (OS) manufacturers such as Microsoft frequently make their products available for academic purposes, both in teaching and research contexts, often at little or no cost, for a variety of reasons. There is the obvious perceived commercial benefit of seeing students educated in the capabilities of their products, with the assumption that this will eventually transfer into the careers of graduates and perhaps their employers' potential purchasing decisions in due course. However, this is a long term benefit at best, and companies also value the many shorter term benefits, such as feedback on their products from a particularly eager and inquisitive community, experience with novel training techniques, and research opportunities. The latter deserves independent discussion and is largely deferred to be outside the scope of this paper. We instead focus on the pedagogical dimensions, and in particular, concentrate on the experiences of the Microsoft Windows Embedded Academic Program (WEMAP) [1] as an example of commercial embedded OS software that has been made especially accessible to university classrooms.

WESE'06, October 26, 2006, Seoul, South Korea.

2. BACKGROUND
This section explains the scope of the WEMAP program for those with little or no prior knowledge of the Windows Embedded product portfolio.

2.1 What is Windows Embedded?
The Windows Embedded product group [2] has been responsible for two main embedded OS products, Windows CE [3] and Windows XP Embedded [4]. A number of more specialized systems for particular vertical markets are also covered by this group, but the Windows Embedded Academic Program discussed in this paper has focused solely on these two general purpose OSs.

Windows CE (aka CE) is a 32-bit real-time OS for small footprint devices, such as battery-powered hand-held devices including PDAs (Personal Digital Assistant) [5] and Smartphones [6]. Windows CE is also used in industrial control devices such as robots [7]. The Windows CE kernel is unique to Windows CE, and contrary to widespread misconceptions, it is not based on either the MS-DOS (MicroSoft Disk Operating System) or Windows NT kernels used in Microsoft's desktop and server OSs. Windows CE is frequently seen embedded within the Windows Mobile platform (not currently covered by WEMAP), which is a special configuration of Windows CE purpose-built for PDAs and Smartphones. Windows CE is highly customizable by the embedded developer, allowing myriad special configurations of its hundreds of components appropriate to the specific target device and application. Windows CE runs in a number of CPU architectures beyond the familiar x86 desktop standard, including ARM, SuperH and MIPS. Windows CE is also notable for being supplied with many of its components visible in source code form.

Windows XP Embedded (aka XPE) is, simply put, a componentized version of Windows XP – Microsoft's flagship 32-bit x86 desktop OS. The componentization allows the embedded developer to leverage the vast PC ecosystem, together with its very attractive economics, while being able to tailor the

OS platform to the specific needs of the target device, removing unwanted components. Special additional components provide extra functionality of benefit to the embedded developer, allowing booting from memory for example and avoiding the need for a hard disk. Because of the common codebase with the desktop OS, XPE runs on x86 CPUs only and is not real-time – though extensions are available from third parties. Typical applications of XPE include kiosks, ATMs and gaming machines [8].

Figure 1 summarizes some of the contrasting key differences between CE and XPE (citing also for contrast, one of the special vertical application embedded OSs, Windows Embedded for Point of Sale). For a more application-oriented comparison, see [9].

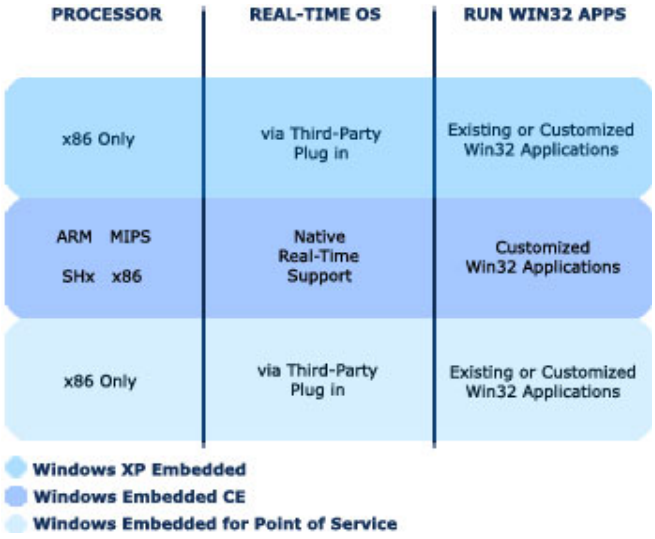


Figure 1. Windows Embedded OSs compared.

3. WEMAP Milestones

This section explains the history of the WEMAP program in broad strokes, identifying the most significant activities and events in order to provide an overview and introduction to the program.

3.1 Origins

Microsoft has had a long history of relationships with academia and there have been a rich variety of means of engagement, ranging from individual product groups and simple sales relationships with specific schools, to deep, rich, long-term partnerships in a research context. Microsoft has a number of academic-focused groups [10] with specific responsibilities to work with schools in various ways. In general, a product group (whose primary role is to design, manufacture, ship and market their products) will defer engagements with academia to these academic-focused groups, except in rare situations. The Windows Embedded product group was therefore relatively bold in its conception of the WEMAP. It arose both from the particular efforts of individuals championing the importance of academia to the product group, and positive feedback from direct early engagements such as with Lancaster University [11].

3.2 Partnership with Microsoft Research

From an early stage, Microsoft Research's (MSR) University Relations (UR) group (now called various names, but External Research & Programs in the US, see [12]) proved to be a valuable partner to the early WEMAP concept. MSR is well known to university researchers as a leading center for computer science research, and its UR programs such as the RFPs (Requests for Proposals) are popular, especially in a research context. MSR teamed with the nascent WEMAP from an early stage, and soon a major joint project was conceived to help launch the program.

3.3 The 2003-2004 Request For Proposals

WEMAP and MSR UR together designed and implemented a major world-wide RFP program focused on research and teaching with Windows Embedded (both CE and XPE). Overall, $1.7M was distributed to 76 projects in 26 countries, following a review of over 150 proposals. Most of the projects were research-oriented, but 14 were focused on teaching. Projects were scheduled to last 1 year from the time of funding, and a wrap-up workshop was held in September 2004 in Cambridge, UK. Results were published on a per-project basis through regular academic channels (various conferences and journals), and projects were also encouraged to submit their materials to the MSDNAA Curriculum Repository (see section 4.5). At the time, this was the largest academic RFP ever run by Microsoft [13].

3.4 Windows Embedded Student Challenge

During 2003, WEMAP teamed-up with the IEEE (Institute of Electric and Electronic Engineers) CSIDC (Computer Society International Design Competition) to form a new student team-based embedded system design competition called the WESC (Windows Embedded Student Challenge) [14]. Run each year since Spring 2004, 30 finalist teams from around the world have convened at Microsoft's headquarters in Redmond, WA, to determine best embedded software design team with projects using Windows CE and a common reference CPU component. Hundreds of teams enter the competition during a document-based elimination round. From 2007, this competition is being integrated with the much larger general Microsoft student competition, the "Imagine Cup" – see section 6. We hope this will enable even more students to participate.

3.5 A New Proposal for Academic Training

In the summer of 2006, a new approach to Windows CE training materials specifically designed for academics was decided to be based around the forthcoming next release of the product. Rather than periodically inviting professors to attend "Train The Trainer" (TTT) events (see section 4.6) which are based on commercial professional developer training materials, this approach would call for curriculum specifically designed by professors for the University classroom.

A group of established Windows Embedded professional trainers, embedded MVPs ("Most Valuable Professional"), and professors were invited to respond to an RFP in May 2006. Georgia Institute of Technology (principal investigator Professor James Hamblen) was selected to develop this curriculum based on its direct experience using Windows CE in the University classroom. The resulting curriculum materials arising from this contract will provide professors with the resources for learning about embedded development using Windows CE and will be made widely available to all academics in early 2007. [15]

4. KEY COMPONENTS OF WEMAP

We now turn to discussing the component parts of WEMAP that have not already been covered by the events described above.

4.1 Accessing the Products

As mentioned earlier, the WEMAP scope is Windows CE and Windows XP Embedded. The standard route for academics to actually get these products, both professors and students, is through Microsoft's regular academic software access channel, MSDNAA (Microsoft Developer Network Academic Alliance) [16]. This is a minimal cost program and provides very wide access to a large array of software (not just embedded), but it relies on the school's department to make annual subscription payments and there is a certain administrative overhead involved in downloading the software. Occasionally therefore, we have had to find workarounds to MSDNAA in certain situations, and this can be challenging as the Windows Embedded products are not sold through retail channels – i.e. they are not standard "shrink wrapped products". A temporary workaround is that full-featured evaluation editions of both CE and XPE are available as downloads on the public Internet [17].

4.2 Shared Source[1] – Curriculum License

Professors are asked to sign a special but simple licensing agreement if they wish to teach using the built-in source code in CE, called the Windows CE Academic Curriculum License [19]. To date, over 400 licensees have signed around the world. While this agreement is not onerous, and it does provide a means to start a conversation between WEMAP and professors who are interested in using CE in their classes, the requirement is a (small) barrier to overcome. We have periodically reviewed the necessity for this license and confirmed it must be continued. However, the current fax-back mechanism we use could be improved and we are seeking a more convenient process as soon as possible.

This attention reflects a trend towards increased accessibility of CE source code that reflects the very different requirements that embedded systems developers have for source code compared to other Microsoft products. Note that we have never required a curriculum license for teaching with XPE – though its source code is not available through standard channels for XPE, so such a license is moot.

4.3 Shared Source – Research License

Historically, a "research source code license" was available to support researchers using CE who needed access to certain software modules that were not available in source form in the standard distribution of CE. This license was based on the original source code license for the regular desktop Windows, which qualified researchers have been able to access for some years. Recently, the research license for CE has not been active but we are reviewing this situation based on demand from academia. One challenge with this license has been its restriction to certain countries [20].

4.4 The Hardware Empowerment Program

Originally conceived as a way to help researchers and educators get up to speed quickly in their RFP projects, the Hardware Empowerment Program (HEP) [21] is a way for embedded development board manufacturers to provide incentive discounts to academics, as well as the BSPs (Board Support Packages) to allow marrying of the Windows Embedded OSs to their hardware. At the time of writing, 10 such manufacturers are part of the program. The discounts are solely the responsibility of the manufacturer and vary between product and manufacturer. Microsoft's role is to provide a readily-accessible portal for its academic partners to see the various hardware and discounts on offer. Microsoft welcomes new vendors applying to join this program.

4.5 The Curriculum Repository

The Curriculum Repository, recently renamed "The Academic Alliance Repository" [22] has been a key feature of the MSDNAA software distribution portal for a number of years. Its purpose is to provide a freely-available large database of proven useful downloadable content materials produced primarily by other academics for academics. (There is some Microsoft-authored and third-party-company authored material too, but academic-authored material dominates). The Repository is fully searchable by keyword and browsable by topic area. At the time of writing, there are over 50 embedded systems data items across 10 types present, most of which are Windows Embedded related. We continue to encourage all of our sponsored or otherwise supported academic projects (e.g. RFPs) to submit their embedded systems and other contents to the Repository, in order to provide useful items for the academic community at large and further the reputation of the authors within their peer groups.

4.6 Train the Trainers

An important recognition in the commercial side of Windows Embedded is that Microsoft is better placed to be a "trainer of trainers" rather than of individual developers. This reflects the business model of Windows Embedded whereby Microsoft sells its embedded products through embedded systems software distributors rather than directly to developers or device manufacturers. Microsoft therefore works with training vendors to significantly multiply the reach of its training efforts. To support this approach, Microsoft develops in-house materials to train such trainers, and then the trained trainers will augment these materials to deliver individual developer training.

Microsoft has periodically invited professors interested in teaching using Windows Embedded products to join in these TTT ("Train The Trainer") events. We have also invited groups of professors, often also with their research students, to attend quasi-training events including Microsoft's professional developer conferences. An example is the annual MEDC (Mobile and Embedded Developer Conference), which is packed densely with detailed technical briefing sessions, hands-on labs, and demonstrations. We also try to make these materials available more widely on disk after the event itself.

However, Microsoft is not in a position of expertise when it comes to the pedagogical requirements of a teaching professor and their students in a university classroom, so it is not likely optimal for Microsoft to train such professors in the way that it trains its commercial training and developer partners. It is this recognition that has led to the development of authoritative curriculum materials through contracting with a respected professor and their team, as described in section 3.5, and an

[1] "Shared Source" is the collective program name for Microsoft's source code availability to customers, partners, developers, academics, and governments worldwide. See [18].

evolved outreach approach in the form of "Academic Days" described in section 5.2.

4.7 Other Components

Over the years, there have been a number of other aspects to WEMAP, but the ones above are dominant in terms of the program's structure. The other aspects include an academic association with the MVP (Most Valuable Professional) [23] program that is a common part of Microsoft's extensive developer community, sponsorship of conferences or academic attendance at conference (including Microsoft "developer conferences", aka DevCons, such as MEDC mentioned above [24]) and various awards to specific projects under special circumstances. We have also provided a default point of contact for academics seeking information or other questions about Windows Embedded for academic use. These items will be deferred from this paper for the sake of focus.

5. NEW DIRECTIONS

5.1 Software Licensing

We expect continual evolution in the software licensing aspects of WEMAP as Microsoft continues to evolve its Shared Source initiative. Today, it should be noted, there are minimal or no special software licensing requirements that a professor needs to consider when using Windows Embedded software, beyond the click-through EULAs (End User License Agreement) in the product themselves, and the fax-back one-page Shared Source license for teaching using the source code. Microsoft is particularly keen to hear from the academic community what source code is required for teaching and research purposes if it is not already available – the trend to date has been to make steadily increasing amounts available according to customer demand (including academic customers).

5.2 Academic Days

As described in sections 3.5 and 4.6, we recognize room for improvement in the ways we can help professors maintain their knowledge of Windows Embedded products and technologies. We do not feel this is a "solved problem" in any sense – the rapid pace of evolution of the embedded systems field, and the Windows Embedded portfolio in particular, makes this especially challenging.

A new approach we are evolving to is the use of dedicated intense training events for professors, based on a model that Microsoft's general developer education organization has found effective, but specifically tailored for academics and with academic-only attendance – we call these "Academic Days". This is a "crash course" style event held over 2-3 days for large numbers of professors (~50-100 is typical) who are invited to make their own way to a venue, usually a university campus. Microsoft provides a series of academic-oriented training and seminar material presented by a mix of professors, embedded MVPs and Microsoft staff, usually from the core product development team. We will continue to refine this approach over time based on feedback from attendees and welcome comments.

5.3 Application Development for Embedded

Expanding on the theme of application development further, it is notable that the scope of "embedded system" is broadening rapidly – today's everyday pocket devices like our smartphones are powerful computers in their own right, and often as powerful as the desktop computers of just a few years ago. As we discuss embedded systems development education therefore, the educator must consider the stretching of what it means to teach embedded systems to higher orders of software development.

For example, Microsoft's Compact Framework ("NETCF") [25] is a special version of the .NET Framework [26] managed code development environment, where the framework provides basic services such as memory management and security to the programmer, enabling greater programmer productivity. NETCF runs on Windows CE, enabling the embedded systems application developer to use managed code such as C# on their device rather than native languages without built-in memory management and security such as C or C++. The implications of such powerful applications development environments for the embedded systems programmer are still being realized. WEMAP today does not directly address such a scope – but perhaps it will need to consider this in the years to come. As with the other topics in this paper, we are keen to hear professors' views on this.

5.4 Hardware Availability

A perennial challenge in embedded software development in the university classroom is the availability of inexpensive, robust, but capable, hardware platforms. Unfortunately the economics of embedded systems hardware means that these boards have tended to be relatively expensive (at least hundreds of US dollars per board is typical).

Alternatives such as using a PC to emulate an embedded system are pragmatic and somewhat useful, but many professors (and students) feel this ultimately frustrates the learning experience for the student – "it's not truly embedded" is a common sentiment.

Much cheaper embedded systems boards are available (typically as evaluation kits for low-end processors), but they are generally far less capable, running 8-bit microcontrollers generally, and may not even run an OS – including not being able to run Windows Embedded.

Major systems trends such as Moore's Law [27] have had an important impact however, and we are starting to see 32-bit boards become available at prices much closer to US $100 – an attractive price point in the student context, since it is around the price of a good text book. So while we do not see this as a solved problem today, the direction looks promising. Examples (will) appear at [21].

5.5 Can I Re-flash My Smartphone for Class?

Related to the hardware question but in terms of more complete devices, a common question from academics who want to teach or research using Windows Embedded has long been, "can I change the OS on a Windows Mobile device?" – such as a retail PDA or Smartphone that they or a student might already have.

The answer in general is no – at least, not unless this scenario is specifically supported by the manufacturer of the device in question, which is rare. Figure 2 illustrates the general case, with the software stack shown as an application running on the OS running on the BSP running on the hardware. Note it is possible to patch parts of the OS and achieve some customization in the Windows Mobile case. It is also possible of course to fully customize your application in both cases.

The common misunderstanding here is of the precise role of Microsoft in the ecosystem that produces such devices. Microsoft is the OS developer, and does not develop or manufacture complete Windows Mobile devices today. Often the crucial IP (Intellectual Property) that lies in the BSP for such devices is simply not made available by the device manufacturer. We continue to explore ways to improve this situation for the academic developer.

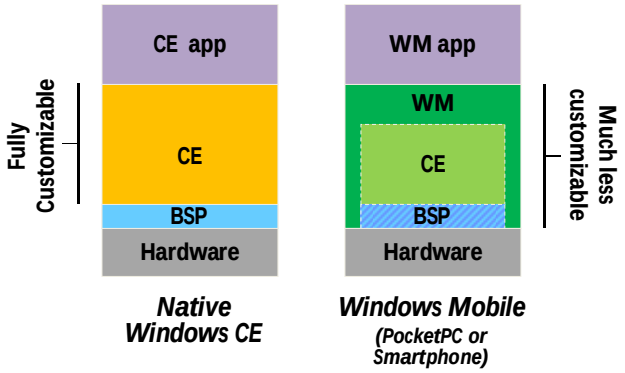


Figure 2. Windows CE & Windows Mobile.

6. WEMAP TODAY

What is the status of WEMAP today?

From the website [1], "The Windows Embedded Academic Program (WEMAP) helps provide a better understanding of the Windows CE and Windows XP Embedded operating systems to faculty and students. As a participant in this program, you can learn how to be part of the next generation of embedded developers or educators on Windows Embedded."

The website includes the following topics:

Getting Started with Windows Embedded in the Classroom

Learn more about how to get started using Windows Embedded as an academic. Learn how to access Windows Embedded software and find out about the best resources to help you get up and running.

Faculty

Academic Curriculum: Learn more about curriculum to help you implement Windows Embedded in the classroom.

Faculty Training: Learn more about opportunities to get training on Windows Embedded.

Research Projects: Find out about Microsoft Research projects.

Discounted Hardware: Learn how to get discounted hardware from leading hardware vendors through the Hardware Empowerment Program (HEP).

Students

The Imagine Cup 2007 Embedded Development Invitational: Are you ready to change the world? Formerly the Windows Embedded Student Challenge, this competition challenges you to build your own embedded device that will have an impact on solving some of our world's toughest problems.

Community Development Projects: Find out about ways you can get involved in Windows Embedded Community projects.

Windows Embedded Career Opportunities: Are you interested in exploring opportunities to work at Microsoft? Find out about ways you can apply for Microsoft internships of full-time opportunities.

Following these key topic areas, there is a news section, and some remarks about the difference between Windows Mobile and Windows Embedded (a common source of confusion and frequently asked question, per section 5.5). There is also a feedback section – again, we are keen to hear your comments and suggestions!

7. CONCLUSIONS

We have described the Windows Embedded Academic Program (WEMAP) from its founding in 2002 through its formative years, up to the present day in 2006. We have done this through illustrating some of the key interesting lessons learned, and provided pointers to some important future directions. We seek to encourage a dialogue through the paper's presentation at the WESE 2006 workshop, between professors active in teaching embedded systems and Microsoft embedded systems representatives, who are keen to shape the program to meet the needs of academia and industry alike.

8. ACKNOWLEDGMENTS

We wish to thank the numerous professors and students who have used the services of the Windows Embedded Academic Program since its inception and provided extremely valuable feedback, and also to our hardware partners in the Hardware Empowerment Program.

9. MORE INFORMATION

For more information on the current WEMAP program, please visit reference [1].

10. REFERENCES

[1] WEMAP home page, http://msdn.microsoft.com/embedded/community/wemap/

[2] Windows Embedded home page, http://www.microsoft.com/embedded/
See also the developer version of this page, http://msdn.microsoft.com/embedded/

[3] Windows CE home page, http://www.microsoft.com/windows/embedded/eval/wince/

[4] Windows XP Embedded home page, http://www.microsoft.com/windows/embedded/eval/xpe/

[5] E.g., Windows Mobile Pocket PC Phone Edition home page, http://www.microsoft.com/windowsmobile/pocketpc/phone/ (PDA is a device category available from other manufacturers).

[6] E.g., Windows Mobile Smartphone home page, http://www.microsoft.com/windowsmobile/smartphone/

(Smartphone is a device category available from other manufacturers).

[7] E.g., KUKA Controls GmbH product, http://www.kuka-controls.com/product/cewin

[8] Case studies of devices using XPE (and CE), http://www.microsoft.com/windows/embedded/testimonials/

[9] Choosing an Operating System by Device Category, http://www.microsoft.com/windows/embedded/eval/choosing.mspx (contains a useful table of device types).

[10] Microsoft home page for Education, http://www.microsoft.com/education/

[11] *Academic Collaboration With Lancaster University Leads to Implementation of IPv6*, Redmond, WA, July 30, 2002. http://www.microsoft.com/presspass/press/2002/jul02/07-30NextWindowsCEPR.mspx

[12] Microsoft Research University Relations world-wide groups, http://research.microsoft.com/ur/

[13] *Microsoft Awards Academic Grants for Cutting-Edge Device Research and Curricula*, Redmond, WA, June 25, 2003. http://www.microsoft.com/presspass/press/2003/Jun03/06-25AcademicGrantPR.mspx

[14] Windows Embedded Student Challenge home page, http://www.windowschallenge.com/

[15] Professor James Hamblen home page, http://users.ece.gatech.edu/~hamblen/ (See also: http://msdn.microsoft.com/embedded/community/wemap/Faculty/default.aspx#WEAC)

[16] MSDNAA home page, http://msdn.microsoft.com/academic/

[17] Windows Embedded evaluation editions, http://www.microsoft.com/windows/embedded/eval/trial.mspx

[18] Shared Source home page, http://www.microsoft.com/resources/sharedsource/default.mspx

[19] Windows CE Academic Curriculum License, http://msdn.microsoft.com/embedded/community/wemap/Faculty/default.aspx#WEAC

[20] Shared Source Licensing Programs: Availability by Geographic Market, http://www.microsoft.com/resources/sharedsource/Licensing/Availability.mspx

[21] Hardware Empowerment Program, http://msdn.microsoft.com/embedded/community/wemap/Faculty/default.aspx#HEP

[22] Academic Alliance Repository, http://www.msdnaacr.net/curriculum/facetmain.aspx

[23] Windows Embedded MVPs, http://msdn.microsoft.com/embedded/community/community/mvps/ (See also the general parent program: http://mvp.support.microsoft.com/mvpexecsum)

[24] E.g., Mobile & Embedded Developer Conference 2006, http://www.medc2006.com/

[25] Compact Framework home page, http://msdn.microsoft.com/netframework/programming/netcf

[26] .NET Framework home page, http://msdn.microsoft.com/netframework

[27] Moore, G. E, *Cramming more components onto integrated circuits*, Electronics Magazine, 19 April 1965.

An Extension Course for Training Trainers of Embedded Software

Masaki Yamamoto*, Shinya Honda*, Hiroaki Takada*, Kiyoshi Agusa*
Hiroyuki Tomiyama*, Kenji Mase**, Nobuo Kawaguchi*** and Nobuyuki Kaneko*

*Graduate School of Information Science, Nagoya University
**Information Technology Center, Nagoya University
***Graduate School of Engineering, Nagoya University
Furo-cho, Chikusa-ku
Nagoya, 464-8063, Japan

myamamoto@nces.is.nagoya-u.ac.jp, {honda, hiro}@ertl.jp,
{agusa, tomiyama}@ is.nagoya-u.ac.jp, {mase, kawaguti}@nagoya-u.jp,
kaneko@agusa.i.is.nagoya-ua.c.jp

ABSTRACT
The embedded system industry in Japan needs more and more embedded software engineers. In 2004, Nagoya University has started an extension course program for embedded software engineers, called NEXCESS (Nagoya university EXtension Courses for Embedded Software Specialists). In the first year of NEXCESS, we have got aware of the tremendous need for the introductory-level courses. With this demand, in the second year, we have started a new course to train future trainers of embedded software. After completion of the trainer course, they are expected to train introductory-level embedded software engineers at their companies. This paper reports the extension course for embedded software trainers.

Categories and Subject Descriptors
K.3.2 **[Computing Milieux**]: Computer and Information Science Education

General Terms
Human Factors

Keywords
extension course, embedded software, workshop, skill, trainer

1. INTRODUCTION
In the automotive and electronics industries over the world, Japan is one of the most competitive countries. A large number of embedded software engineers underpin the high competitiveness. It is reported in [1] that approximately 193,000 engineers are working on the development of embedded software in Japan. Nevertheless, the report also reveals a shortage of as many as 94,000 engineers.

In 2004, we have started an extension course program named NEXCESS (Nagoya university Extension Courses for Embedded Software Specialists) [2], supported by Ministry of Education, Culture, Sports, Science and Technology, Japan. NEXCESS is targeted towards embedded software engineers in industry. In the first year, NEXCESS offered eight courses at several levels of expertise, from the introductory level to the advanced level.

Among the eight courses, we have recognized that the strongest demand from industry is addressed towards the introductory-level course. The background of this fact is that few universities in Japan provide sufficient classes on embedded software development even for computer science students. Although NEXCESS opens the introductory course several times a year, its capacity is limited. Therefore, we have decided to develop a new class to train future trainers of embedded software. After completion of the trainer course, they are expected to teach introductory-level embedded software engineers at their companies.

This paper presents an outline of the introductory course and the trainer course in NEXCESS.

The rest of this paper is organized as follows. Section 2 describes the needs of embedded software engineers in industry. Section 3 presents the NEXCESS introductory courses. Section 4 presents the trainer course. Section 5 reports the results of the trainer course.

2. NEEDS
2.1 Courses
When we designed extension courses, we first classified technical skills into three levels as follows.

WESE2006, October 26, 2006, Seoul, South Korea

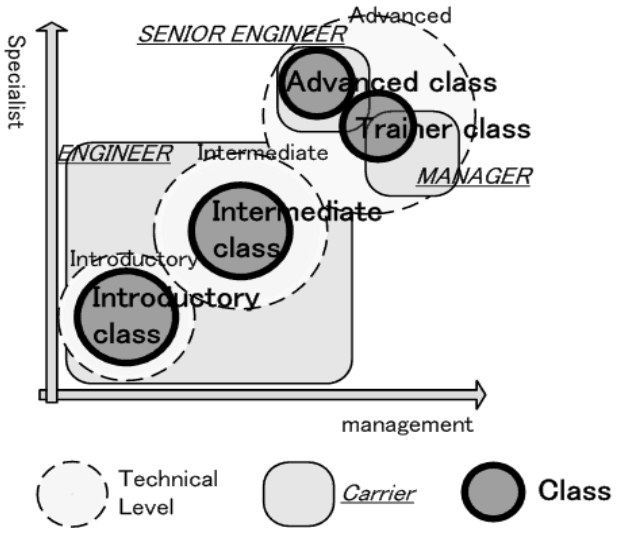


Figure 1. Course Map

(1) Introductory Level

(2) Intermediate Level

(3) Advanced Level

We also classified career paths for engineers in two directions as follows.

(1) Administrative Managers

(2) Technical Specialists

Then, we classified the engineers into four classes using the both sides of technical skill and management ability as shown in Figure 1. Engineers belonging to each class have the following responsibility.

(1) Introductory class

- Engineers in the introductory class are engaged in the programming process of software development, and in direct software production (see Figure 2).

(2) Intermediate class

- Engineers in the intermediate class are engaged in the design and high quality testing processes of software development, and in direct software production (see Figure 2).

(3) Advanced class

- Engineers in the advanced class work as a specialist of each advanced technology, and are engaged in indirect software production.

(4) Trainer class

- Engineers in the trainer class work as an educational specialist.

We developed nine extension courses, which educate engineers in these classes. Each course is only two to four days long so that industrial people can easily take the class. The titles of the nine courses are listed below.

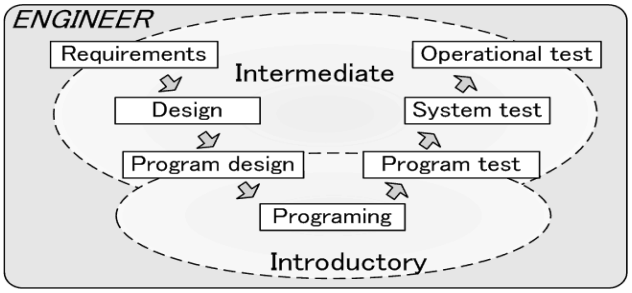


Figure 2. Software process and Introductory engineer

Table 1. Course list

Course	Total applicants	Applicants per course
Introductory class	273	54.6
Intermediate class	154	38.5
Advanced class	261	26.1
Trainer class	17	17

(1) Introductory class (one course):

(course-1) "Fundamentals of the embedded software development technology"

(2) Intermediate class (two courses):

(course-2) "Design methodology and management of embedded software"

(course-3) "Software design technology with a real-time OS"

(3) Advanced class (five courses):

(course-4) "Internal structure of a real-time OS"

(course-5) "C-based embedded hardware design"

(course-6) "System control middleware and application"

(course-7) "Software engineering for embedded systems"

(course-8) "Ubiquitous interface and embedded software programming for image processing"

(4) Trainer class (one course):

(course-9) "Training of future trainers who give a lecture on the introductory class"

The faculty members in Nagoya University offer most of the courses, but some courses are partially supported by engineers in industry, professors of the other universities and members of Specified Nonprofit Corporations (NPOs). Specifically, NPO TOPPERS Project [4] and NPO SESSAME [5] are primary contributors to NEXCESS.

2.2 The number of applicants

NEXCESS is to be carried out from 2004 to 2009. In 2004 and 2005, we held courses 20 times in total. The introductory course was held 5 times. The intermediate courses were held 4 times, and the advanced courses were held 10 times. The trainer class was held once.

Table 2. Introductory skill map

Technology component		
Information Processing	Information input	Read a switch using a PORT
	Information output	Turn on a LED using a PORT
	Data processing	Use the inside timer of CPU
Platform	Processor	Initialize CPU, Understand memory map
	Basic software	Use the real time operating system
Exploitation technology		
Design	Software design	Understand structured design
Programming and testing	Programming	Do C programming
	Testing	Generate boundary value
Integration	Integration testing	Do test design
Management skill		
Development process management	Process design	Understand development process

We accepted applications through the Web. Table 1 shows the number of applications to each class. The number of applications to every course exceeded the capacity. We selected the students suitable for the course.

The number of embedded software engineers at all the technical levels is insufficient [1]. For example, program managers are insufficient in most of large companies. Designers are insufficient in most companies that perform advanced research and development. Programmers are insufficient in all the companies. As such, most companies need much more embedded software programmers.

In many companies, engineers at the introductory level work on the programming and testing processes (see Figure 2). We assume that the introductory engineers must study structured programming, embedded C programming and testing. These skills are necessary to carry out the programming and testing processes. We have developed the introductory course titled “Fundamentals of the embedded software development technology”. We educate the introductory-level programmers in this course.

Table 1 clearly indicates the strong demand from the industry to education of introductory-level engineers. We open the introductory course several times a year, its capacity is limited due to the limitation of facilities. Therefore, we have decided to develop a new class to train future trainers of embedded software. After completion of the trainer course, they are expected to teach introductory-level embedded software engineers at their companies.

3. INTRODUCTORY COURSE

The trainers should be able to teach at least the introductory course, i.e., “Fundamentals of the embedded software development technology”. This section explains the outline of the introductory course.

3.1 Requirement for application

In extension courses including NEXCESS, students generally have very different backgrounds such as graduated schools, ages, current companies and responsibilities in the companies.

At the introductory course in NEXCESS, exercises in embedded software programming are based on the C language. Therefore, we require applicants to the introductory course to have more than one year of experience in C programming. Applicants who do not satisfy this requirement are not selected if the number of applications exceeds our capacity, and fortunately, this always held so far. After this selection, although students have various backgrounds, at least, we can assume that all the students can write programs in C.

3.2 What to teach

We have defined the skills that an introductory embedded software engineer has to acquire.

An engineer at the introductory level works on the programming and testing processes, not on the design and requirement analysis. An engineer at the introductory level has to do C programming and test design. Of course, it is preferable for him/her to be able to do software design, but we think that software design is one of responsibilities of intermediate-level engineers.

Introductory-level engineers must be able to write embedded software programs. Programming of embedded software is different from that of enterprise- and business-oriented software. They must be able to directly manipulate CPUs and other hardware devices such as UART, timers, ports and so on, without OS or library supports.

Introductory-level engineers write embedded software under a direction of project managers. It is not necessary for the introductory engineers to have the ability of project management,

but it is necessary for them to understand the overall software development process.

In Table 2, we have defined the skills that an introductory embedded software engineer has to acquire. The skills, which we have defined, are based on ETSS (Embedded Technology Skill Standards) which was defined by IPA [3]. For NEXCESS, we have modified ETSS.

3.3 How to teach

For engineers working in industry, teaching just theoretical knowledge is not sufficient. Not only knowledge but also practical skills need to be improved. They are expected to utilize the skills acquired during the course in their work in practice. They are not commentators but engineers. The introductory embedded software engineers should write embedded software programs.

In order to improve technical skills for introductory engineers, we have developed a set of exercises as follows.

(1) Documentation exercise

In software development, documentation is one of the most important processes. In this exercise, a student writes design specifications, test specifications, a review report, etc.

(2) Real machine exercise

In this exercise, a student uses a real microcomputer board. The student writes a program and runs it on the microcomputer board. It is effective for a programming exercise.

(3) Discussion exercise

Students discuss in a group. The exercise is effective for code review.

(4) Role playing exercise

In this exercise, a student plays a different role from his/her original role. For example, those who are always programming play a role of a manager. By acting in a different position, a student can obtain a wider scope

(5) Case study exercise

For example, if a student studies the example of the program bugs, he/she will not cause the same failure.

(6) PBL (Project Based Learning) exercise

A student undertakes a small project for education. The student will obtain a wider view of work by experiencing the project from the beginning to the end.

The introductory course is targeting to engineers who take charge of a part of software development processes. Their primary responsibility is programming. Therefore, we have decided to perform real machine exercise. Of course, all the software engineers need to write documentation. Therefore, we have decided to perform documentation exercise, too.

3.4 Real machine exercise

We use a microcomputer board with a 16-bit CPU (see Figure 3). This CPU has 2KB RAM and 64KB flash ROM. This board also has a UART port to connect to a host PC. We can develop a C program on the PC and download it to the target board. The development environment on the PC is an MS-Windows

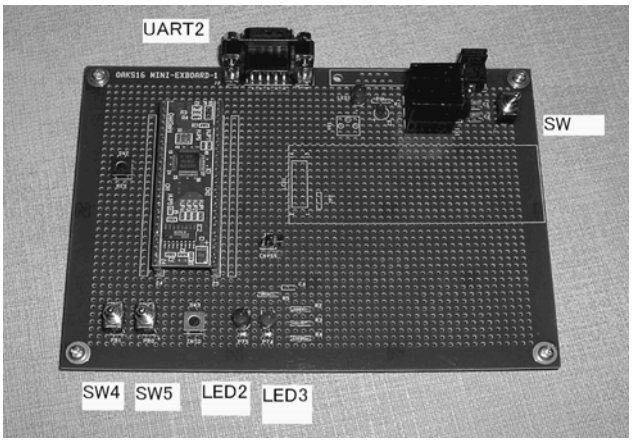


Figure 3. The exercise board

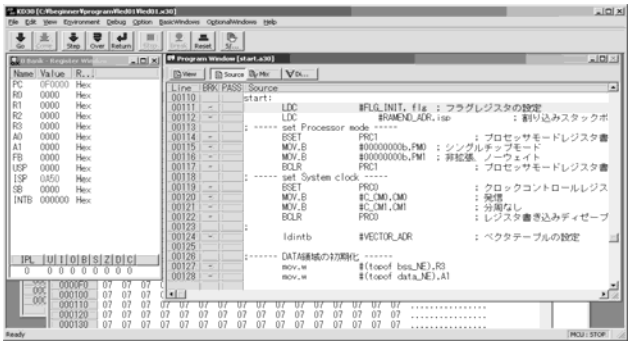


Figure 4. The development environment

application, which is easy to use (see Figure 4). Therefore, it is suitable for the education purpose.

This board has switches and LEDs, too. Through this exercise, a student develops a simple countdown timer. The specification of the timer is as follows.

- When switch 5 is turned on, a set period is set to 1 minute and countdown is started.
- After starting the countdown timer, LED2 blinks at intervals of 10 seconds.
- LED2 blinks twice at intervals of 0.25 second.
- If the set period is reached, LED3 blinks for 15 seconds at intervals of 0.25 second.
- When switch 4 is turned on, a set period will be extended for 30 seconds.

Although this is a simple application, it is very effective for students to study embedded programming.

For example, in order to read a switch, the students have to set up a port appropriately. It is necessary to set up the direction of input and output of the port by a special register in the CPU. The students have to read a CPU manual. By reading a circuit diagram, one understands whether a switch is on or off at the time of '0'. In

programming of enterprise- or business software, programmers do not have to be aware of such things, but they are very important in embedded software programming

A student also experiences characteristic coding by embedded C programming. For example, assume that in order to reflect the contents written in a port register, the port register needs to be read. An inexperienced engineer may write the following code.

```
void led_reflecte(void){
  unsigned char tmp;
  tmp = *((unsigned char *)(0x3ed));
}
```

However, an optimizing compiler will remove "`tmp`" because "`tmp`" is not referred to after it is written. Therefore, this program does not work correctly. For correct behavior, "`volatile`" needs to be specified as follows.

```
void led_reflecte(void){
  unsigned char tmp;
  tmp =
     *((volatile unsigned char *)(0x3ed));
}
```

In addition, many other skills are acquired through these exercises.

3.5 Documentation exercise

For an introductory engineer, programming is the main mission. However, he/she needs to study the design process, too. This is because a better program can be made if he/she has knowledge on design.

Moreover, he/she needs to consider test specification. If there is capability to test, he can make a better program.

Then, we provide a document exercise in the design and test processes. Students write a 10-page document in the design exercise. They also write a 10-page document in the test exercise.

3.6 Course schedule

A lecture is required before an exercise. If the students do not understand the reading of a CPU manual, they cannot do these exercises. Therefore, after giving a lecture first, they do these exercises.

In NEXCESS, a class starts at 9:30 and finishes at 17:00. An introductory course is four days long. The overall schedule for the four days is as follows where mark '*' indicates an exercise.

The 1st day

> Embedded system, Software development process, Structure design, Structure design exercise*, Check of development environment*

The 2nd day

> Hardware knowledge, Embedded C programming, How to use the board for an exercise*, Embedded C programming exercise*

Figure 5. Introductory course evaluation: availability

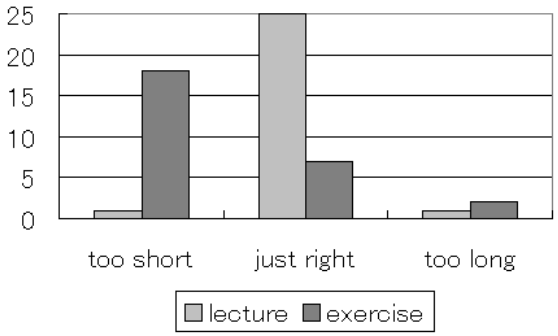


Figure 6. Introductory course evaluation: time

The 3rd day

> Real time OS, ITRON programming*

The 4th day

> Test process, Test specification*, Down count timer application*

3.7 Evaluation by the students

We ask students to evaluate the course. Figure 5 shows a part of the evaluation results where 5 is the highest. The score of the usefulness in practice is 4.4. This shows that the contents of the introductory course are directly applicable to their work.

Evaluation on time length is shown in Figure 6. Students think that the time of a lecture is just right. However, they think that exercise time is too short.

In most companies, embedded software engineers are too busy to spend many days for the education purpose. Therefore, at NEXCESS, the introductory course is carried out in four days. However, the evaluation results show that four days are not enough. The right length may be about 6 days.

4. THE TRAINER COURSE

In NEXCESS, the capacity of the introductory course is 30, and the course is held two or three times every year. This means that only about 100 introductory engineers per year can be trained, which is not sufficient at all

In 2005, we have developed a new course to train future trainers of introductory engineers. We train 10 trainers per year in the

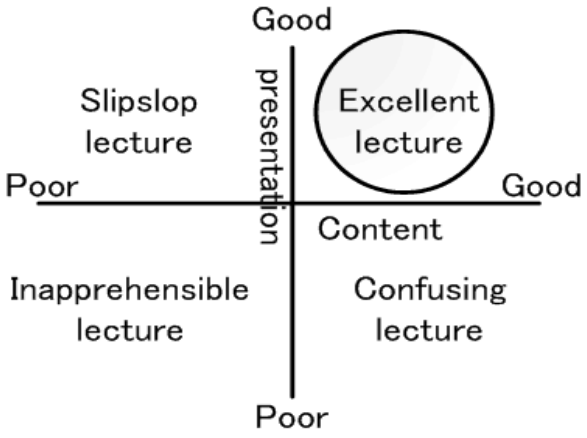


Figure 7. Teaching skill is useful

course. If each trainer trains 20 introductory engineers per year, then 200 introductory engineers are trained per year.

4.1 Course schedule

The trainer course is a little longer than the other courses. The course consists of four stages.

(1) First stage (2 days)

Teaching the education skill

(2) Second stage (4 days)

Introductory class observation

(3) Third stage (4 days)

Practice teaching of introductory class

(4) Fourth stage (1 days)

Review

Students of the trainer course have to understand the technical contents of the introductory course. In addition, they have to acquire educational skills.

4.2 Education skills

What are the education skills? METI (Ministry of Economy Trade and Industry, Japan) has defined education skills as follows [6]. Education skills are categorized to three types (a) to (c) as follows.

(a) Investigate a training trend skill, Project management skill, Leadership skill, Communication skill, Negotiation skill

(b) Planning skill of a training course, Course development skill

(c) Instruction skill

Type (a) skills are general and not specific to education.

Type (b) skills are the planning skills which are not necessary for the trainer course at NEXCESS since we have already developed the introductory course.

We put our focus onto type (c) skill. The instruction skill is the core skill for education. We have further classified the instruction skill into two, i.e., a teaching skill and a coaching skill.

Table 3. Teaching and Coaching

Teaching	Coaching
The educational method to clue up	The educational method to draw forth
It teaches that a student does not know.	It helps for a student to discover.
Required also of individual guidance or class room.	It is required when doing individual guidance.
Capability required as an educator.	Capability required also as a boss.

4.3 Teaching skill

Teaching skill is trainer's capability to give a lecture. The trainer cannot give a lecture without teaching skill, even if teaching materials are prepared. Good presentation as well as good contents is required for a good lecture (see Figure 7). In the trainer course at NEXCESS, we teach the following teaching skill.

- Preparation of a lecture
- How to talk
- How to use a whiteboard
- The method of a question to the student
- How to hear the opinion of a student

We prepare several exercises to train the teaching skill. For example, a student gives a short lecture for several minutes, followed by discussion on the teaching skill.

4.4 Coaching skill

What is the coaching skill? What is the difference between teaching and coaching?

Coaching is an educational method, which grow the capability of students. The difference between coaching and teaching is summarized in a table 3.

Recently, the importance of coaching has been recognized in the business field. In particular, in embedded software industry, it is very important due to the following reasons.

One reason is due to the technical contents to be educated. As for the embedded software industry, technology is rapidly progressing. In such domains, it is very difficult for even leaders or senior engineers to catch up with all of the technology. A leader of a group has to survey the technology together with the other members in his/her group. The leaders may not always have more knowledge on the technology than other members may. Even in that case, however, if the leader has the coaching skill, he/she can guide the other members.

The second reason is concerned with a mentality. In software industry, it is necessary for workers to be highly motivated. The software developed with a high motivation has high productivity and quality. For this reason, a leader needs to manage his/her group members so that they keep a high motivation. Moreover, their mental illness can also be prevented by raising their motivation.

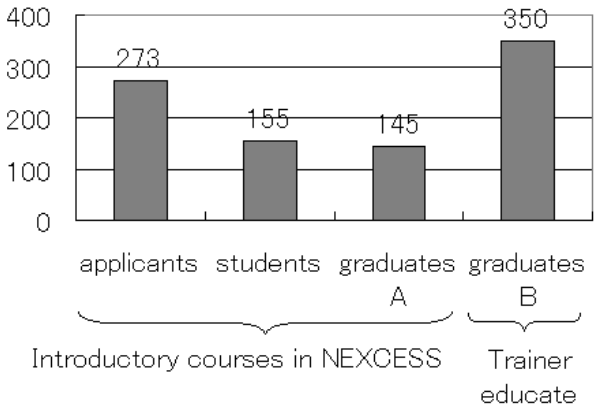


Figure 8. Introductory engineers

In the trainer course at NEXCESS, we educate the following coaching skills.

- How to adjust to the motion of a partner. (Mirroring)
- How to adjust to the way a partner speaks. (Pacing)
- How to listen closely.
- How dose a partner speak about real intention.
- How to make a good action plan.

We have developed several exercises to train the coaching skill. Many managers tend to teach without hearing subordinate's talk. Intentionally mirroring and pacing exercise improve the attitude to listen.

5. RESULT

We held the introductory courses five times in 2004 and 2005, and the trainer course once in 2005.

5.1 The number of introductory engineers

We held five the introductory courses in 2004 and 2005. The number of applicants is 273. Out of them, we selected 155 persons. Since 10 persons were absent, 145 persons successfully completed the course (see Figure 8).

17 persons took the trainer course in 2005. Many of them are working in the educational section of their companies. In their companies, they teach the introductory engineers in 2006. A total of 350 engineers are educated in their report (see Figure 8).

Only 145 persons have been educated through 5 times of introductory engineer courses. However, 350 introductory engineers' education was made by a single trainer course. We can conclude that the trainer course is very effective to educate introductory engineers.

5.2 Evaluation by the trainers

We asked the students of the trainer course to evaluate the course. Figure 9 shows a part of the evaluation results where 5 is the highest. The average score of usefulness in practice is 4.7. This shows that the contents of the trainer course are applicable to their work.

Figure 9. Trainer course Evaluation: availability

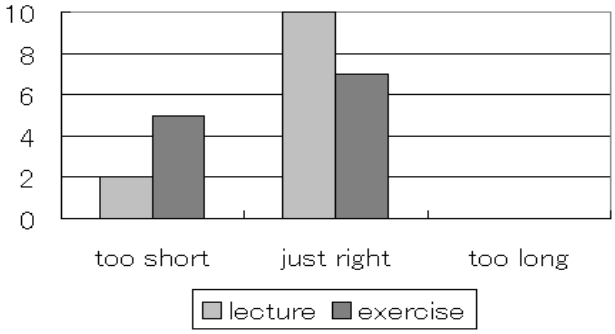


Figure 10. Trainer course Evaluation: time

Evaluation on time length is shown in Figure 10, indicating that the students need longer time.

5.3 Evaluation of the education skill

Especially the student in the trainer course got interested in the coaching skill. The students wrote in the questionnaire as follows.

- I understood the definition of coaching.
- I got to know that not only teaching skill but also coaching skill are required.
- When I tried it, I found that it was difficult.
- I learned many things that I had never experienced or investigated.

Many of the students of the trainer course are working in the educational section of the companies. However, many of them had not sufficiently studied the education skill. In order to raise an educational effect, it is required to improve the education skill. We understood that this course was important.

6. CONCLUSION

Education of embedded software engineers in industry is very important. Nagoya University has started an extension program on embedded software, named NEXCESS. Through the first year of NEXCESS, we have found that introductory engineer training is highly required.

We have developed a trainer course to educate future trainers. Students of the trainer course are expected to train introductory-

level embedded software engineers at their companies. Thereby, many introductory engineers can be educated efficiently.

NEXCESS is now in the third year. We are continuing to improve our courses.

ACKNOWLEDGMENTS

NEXCESS is supported in part by Ministry of Education, Culture, Sports, Science and Technology, Japan.

REFERENCES

[1] Ministry of Economy Trade and Industry. *2006 Embedded software actual condition survey*, https://sec.ipa.go.jp/download/200606es.php, 2006.

[2] M.Yamamoto, et. al., "NEXCESS: Nagoya University Extension Courses for Embedded Software Specialists", Workshop on Embedded Systems Education, Sep. 2005.

[3] Ministry of Economy Trade and Industry. *2006 Embedded Technology Skill Standards*, https://sec.ipa.go.jp/download/200606eb.php, 2006.

[4] NPO TOPPERS Project, http://www.toppers.jp/

[5] NPO SESSAME, http://www.sessame.jp/

[6] Ministry of Economy Trade and Industry. *IT skill standard Ver. 2*, http://www.ipa.go.jp/jinzai/itss/download.html, 2006.