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.

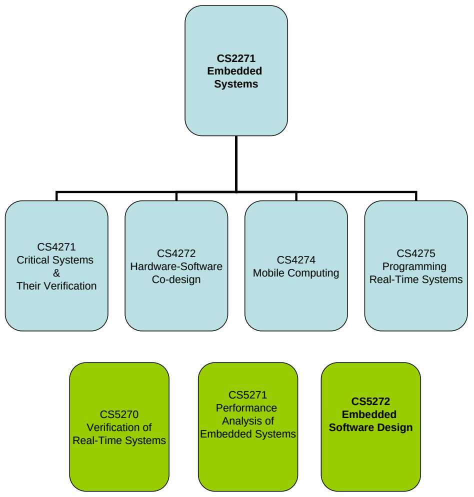
CS2271 Embedded Systems
CS4271 Critical Systems & Their Verification
CS4272 Hardware-Software Co-design
CS4274 Mobile Computing
CS4275 Programming Real-Time Systems
CS5270 Verification of Real-Time Systems
CS5271 Performance Analysis of Embedded Systems
CS5272 Embedded Software Design

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 universities 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.