Documents on Teaching submitted by Abhik Roychoudhury

Email: abhik@comp.nus.edu.sg
http://www.comp.nus.edu.sg/~abhik

CONTENTS

I. **Teaching Statement** – Page 2
Teaching Experience, Teaching Philosophy.

II. **Modules Taught** - Page 5

III. **Pedagogical Innovations and Reflections** - Page 8

IV. **Supervision** – Page 11
Graduate students, Undergraduate Student projects,
Student thesis committee participation.

V. **Authoring of textbook** - Page 16

I. Teaching Statement – articulated by Abhik Roychoudhury

Email: abhik@comp.nus.edu.sg
http://www.comp.nus.edu.sg/~abhik

"*In learning you will teach, and in teaching you will learn" – Phil Collins.*

A. Teaching Experience

I have taught modules in programming languages / software engineering as well as modules in embedded system design at NUS, since 2001. Overall, I have taught a wide variety of modules starting from first year undergraduate courses (1000 level), to modules for PhD students (6000 level).

B. Teaching Philosophy

Effective teaching has often been associated with effective dissemination of knowledge leading to advanced understanding of a topic by the students. One can also extend this definition to include the notion of ``conceptualization'' -- where the students not only understand a specific topic of a discipline, but also can extrapolate this understanding in other topics or in real-life problems. As an educator, I feel that effective teaching goes far beyond effective understanding and conceptualization of the students as learners. It involves *integration of the spirit of research into teaching and vice-versa*. What we mean by integration of research into teaching is a situation where the course instructor becomes a *serious learner* himself/herself and the teaching facilitates this process. So, by teaching a course, the instructor is spurred to conduct research on certain cutting edge topics which are later integrated into module(s). At the same time, the students become more of researchers or investigators themselves, rather than learners.

As an example, let me refer to a graduate level module I have offered ``Automated Software Validation''. While teaching some traditional formal techniques for software validation, some of the difficulties in using them for software debugging became clearer to me. I came to fully realize that the user does not often have a clear specification of the ``error'' being checked -- rather he/she only has an understanding that the behavior of a program for a particular input is unexpected. This inspired me to do research in semi-formal techniques on software fault localization with one of my Ph.D. students. Some of the experience gained from this effort has been integrated into a lecture in later offerings of the course.

Bringing in the spirit of research into teaching, is, by no means, restricted to graduate courses. In the following, I outline some teaching methods which I employed in my undergraduate teaching for this purpose. It is worthwhile to mention here that I have often taught courses on formal verification or logic --- topics which are relatively mathematical in nature. Of course an important task in the teaching of such courses is to relate these mathematical topics to the actual practice of computer system design. However, this relationship is often brought out in an extreme way (in textbooks or other teaching materials) --- dramatic historical incidents which led to the spectacular failure of computer systems due to the lack of formal verification are presented to the students. This clearly attracts the students' attention, but this interest becomes difficult to retain, possibly because the historical incidents seem far removed. Instead, to motivate more mathematical topics like formal verification, I felt that the students need to *understand* the impact of verification in system design; they do not need to be surprised.

So, in my undergraduate course on validation of embedded systems (CS4271 Critical Systems and their Verification) -- I start with existing practices of verification/validation and how they are intrusive to the design process. I then discuss how a model-based formal approach can help the design cycle, without referring to historic disasters that occurred due to lack of formal verification. Thus, I have developed a rather different pedagogical style for teaching validation/verification, where the aim is to discuss the system design cycle with the students rather than surprising the students with the power of formal verification and logic. Once the learners appreciate the differences in the techniques, they develop a more long-lasting interest. This results in a learning process with more active participation from teacher/students --- a goal which I always try to achieve in my classes. Furthermore, such classroom discussions have spurred my own research, the results of which have later been integrated into teaching.

In particular, realizing the problems with the current materials on embedded systems validation (existing books either focus on embedded systems or verification in a generic sense but do not combine the two), *I also wrote an entire textbook on the topic, which has been published by Morgan Kaufmann (Elsevier) and has been adopted in modules at universities of different regions including US, South Korea, New Zealand and Europe*. *A Chinese translation of the book has been used for teaching at universities in China*. Writing the textbook has been influenced by my past teaching and research --- in particular it contains valuable intuitions gained from my past research on timing analysis of embedded software, and my past research on dynamic slicing / software debugging. Apart from being adopted for teaching at various universities – the book has also attracted attention from practitioners. Excerpts from the book were highlighted as *Editor's Top Picks* in EE Times (edition published on August 6th 2012). EEtimes is a popular electronics industry magazine which has been the electronics industry news source for design and development engineers for more than thirty years. Featuring of my textbook in such a practitioner's magazine, clearly shows the relevance of my teaching materials in industrial practice. This goes towards one of the goals for quality teaching – rendering the students with a teaching experience which makes them fully ready for the workplace. The overall experience here clearly shows a confluence of teaching / research / practice, namely

- valuable intuitions gained from my research on software debugging and analysis feed into the writing of a comprehensive textbook
- the textbook is adopted for teaching in my module, as well as in many other universities
- the textbook is highlighted in a popular practitioner's magazine, generating interest and comments from developers – which further refines future teaching of the material.

As another instance of merging teaching / research / practice – I can mention my upcoming module on Software Testing and Debugging. As is well-known, software testing is the most commonly used method validating software. It is widely used by programmers on a daily basis. Once failed tests are found by software testing, software debugging tries to localize the root cause of failure. However, developers may often use ad-hoc techniques for testing and debugging. My module on software testing and debugging is planned to introduce the students to cutting-edge systematic techniques for testing and debugging, while simulating industrial practices such as a group of students testing code written by another group of students. This simulates the reality in the software industry where "developer" and "tester" often correspond to different job descriptions. Furthermore due to the mobility of personnel across companies, a developer may be asked to take over the code written by a previous developer. I have attempted to simulate such real-life scenarios in curriculum itself – thereby integrating teaching, research, and practice.

In summary, my teaching methods are driven by an effort to cross-fertilize teaching, research and practice. I have adopted these methods in classroom teaching as well as in other forms of teaching such as student supervision and textbook development. As educators, I feel that we should strive to *break the walls between ``teaching", ``research" and "practice"* in our own minds. This can enhance a new spirit of excellence, allowing us to try and seek the best for our learners.

II. Modules Taught by Abhik Roychoudhury

Email: abhik@comp.nus.edu.sg
http://www.comp.nus.edu.sg/~abhik

In the following, I describe the eight modules taught by me during my employment at NUS since 2001. Some of these modules have also been designed by me. These modules are highlighted first.

A. Modules designed and taught

CS4218 Software Testing and Debugging I am currently developing an undergraduate module on Software testing and debugging, which will be offered in Jan 2014. This proposed module will be a primary module in the Software Engineering focus area in the CS department. This is also the first time that a module on the fundamentally important area of Software Testing will be taught at NUS. At this point, I have finished a survey of existing courses and teaching methods in software testing through my personal contacts among software testing and software engineering researchers. The course will be fully project based rather than having a collection of assignments. During the duration of the course, every attempt will be made to simulate industrial setups such as one project group fixing bugs in another project group's code. My discussions with software engineering researchers at industrial labs, such as Microsoft Research, has confirmed that bringing in such industrial practices into curriculum makes the students much more suited for the work-place. Finally, I also plan to include discussion of cutting-edge testing research in the course. At this point, I am actively involved in research on testing and debugging, and the students usually appreciate getting a feel of the most cutting edge technologies in the field.

CS6880 Advanced Topics in Software Engineering I offered a new module Advanced Topics in Software Engineering in 2011-12. This module involved paper presentations (with peer review by other students), projects and discussions, apart from lecturing. A broad based offering covering advanced theoretical concepts (such as link between logics and requirements) and the advanced state-of-the-practice (such as the testing methods used in industry) was worked out. In terms of the contents of the module CS 6880, I offered a mix of advanced theoretical concepts, as well as advanced concepts in the state-of-the-practice in software engineering. In terms of advanced theoretical concepts, we studied the underlying foundations of temporal logics and their linkage in terms of representing behavior as captured in software requirements. In terms of advanced practical topics, I covered testing methods studied in the industry for safety-critical software, such as MC/DC testing. Apart from conventional lecturing, this module also involved independent projects and paper presentations. The students in this module did projects in cutting edge software engineering issues such as: Software out-sourcing, software-as-a-service (SaaS) and software taint analysis. The paper presentations were done by the students and also involved peer-review from other students.

CS5219 Automated Software Validation In this module, the students are exposed to a wide variety of methods for formal verification of software. These methods are particularly important for software deployed in safety-critical settings. The techniques employed range from search based techniques such as model checking to more powerful automated theorem proving methods. The students undertake a substantial project where they take informal English language requirements, develop a formal model and employ state-of-the-art model checking tools to perform formal verification of properties of the system model. More advanced students are also encouraged to perform automated code generation from the formally verified model. One can refer to http://www.comp.nus.edu.sg/~abhik/5219/lesson-plan.html to get a feel for the structure of the module. In this offering, we gave one of the challenge problems in software verification to the students in the module. This involved modeling, verifying and developing the software for a pacemaker http://sqrl.mcmaster.ca/pacemaker.htm based on real-life requirements specification of a pacemaker. This gives the students exposure to the challenges in developing correct software from informal requirements.

CS4271 Critical Systems and their Verification [Module folder included in Supplementary Dossier] This module seeks to familiarize the students with the basic concepts of building reliable embedded systems. Various kinds of validation methods are studied, ranging from validation of models, to validation of actual system implementations. We also study validation of various criteria (functionality, performance) for embedded systems. There is focus for formal verification as well as other methods such as static and dynamic analysis. This module is one of the Depth (level 4) technical electives in the CEG program, jointly offered by Departments of CS and ECE. More details about the module content development also appear in my Teaching port-folio, in the Supplementary Dossier. A web-site of an earlier offering of the module can also be accessed from http://www.comp.nus.edu.sg/~abhik/CS4271/lesson-plan.html

B. Other Modules taught

CS3211 Parallel and Concurrent Programming [Module folder included in Supplementary Dossier] A concurrent system consists of set of processes which execute simultaneously and may collaborate by communicating and synchronizing with each other. This leads to subtle program behaviors which the students need to learn how to grasp, via this module. This is a module offered to third year undergraduate students; some of the students attending the module might also be in their second year of study. The module provides a second exposure to the powerful concept of concurrency – with the students having received a very initial exposure to concurrency earlier in a module on operating systems. The students learn concurrent programming primitives such as locks, barriers and also study the realization of such primitives in a mainstream multi-threaded programming language such as Java. In the later part of the module, the students appreciate the nuances separating concurrency and parallelism, and take part in hands-on parallel programming on top of a popular programming language. The module folder of CS 3211 is being made available with the Teaching portfolio, in the Supplementary Dossier. A web-site of the module is also available from http://www.comp.nus.edu.sg/~abhik/CS3211/2013/index.html

CS4272 Hardware Software Co-design Till recently embedded computing systems were designed by specifying and realizing hardware and software separately. This often leads to incompatibilities across the hardware-software boundary making system integration a very difficult task. The goal of co-design methodologies is to overcome these difficulties by using unified hardware-software representations and creating the means for exploring various ways of partitioning the system into hardware and software components. This module will expose the fundamental issues in hardware-software co-designs by exploring topics such as: High level system descriptions, design partitioning, scheduling, hardware platforms, software analysis and power-aware system design.

CS2104 Programming Language Concepts A small number of concepts underline the hundreds of programming languages that have been designed and implemented. This second year undergraduate module introduces the concepts that serve as a basis for programming languages. It aims to provide the students with a basic understanding and appreciation of the various essential programming language constructs, programming paradigms, evaluation criteria and language implementation issues. The module covers concepts from imperative, object-oriented, functional and logic programming. These concepts are illustrated by examples from varieties of languages such as C, Java, Smalltalk, Scheme, Prolog, Perl etc.

CS1102 Data Structures and Algorithms This module provided a first exposure to data structures and algorithms to first-year undergraduate students. The students get hands-on experience with basic data structures like linked list, stacks, queues, heaps, trees and graphs. They also get exposure to sorting algorithms, searching algorithms, and asymptotic analysis of algorithms.

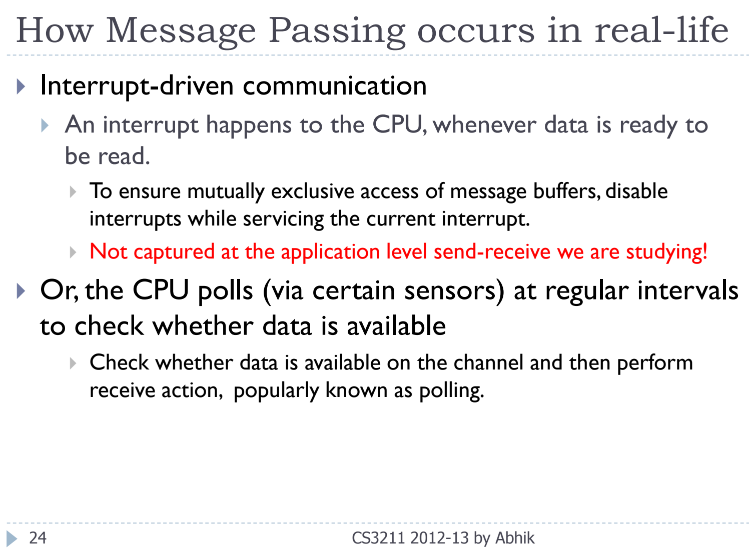
III. Pedagogical Innovations and Reflections by Abhik Roychoudhury

Email: abhik@comp.nus.edu.sg
http://www.comp.nus.edu.sg/~abhik

In the following, I highlight some relatively novel teaching practices I have adopted to enhance the students' learning experience.

A. Additional Reflection Sessions on Saturdays

Over the years, I have taken several steps to improve on my teaching in terms of the content, delivery, exposition and inspiration. In the previous semester, I was teaching a third year undergraduate module - Parallel and Concurrent Programming. This is the students' first serious foray into concurrency – they get an initial understanding earlier in a module on Operating Systems. As a result, I felt the need for reflection sessions – beyond the lecture and tutorials. Thus, the lectures were used to introduce the teaching materials, the tutorials were conducted as problem solving sessions, and the additional reflection sessions were conducted as open-ended discussion sessions to clarify the students' understanding of concurrency and parallelism. Note that the reflection sessions were very different from a group consultation session – where only the students ask questions and the instructor answers them, possibly by sharing the answer with all the students present. Instead, the reflection sessions served two purposes. First of all, exercises were given out impromptu during the reflection sessions, the students did not know these exercises in advance. From the working-out of these exercises during the session, the students were reminded of important concepts. Secondly, the students were encouraged and convinced to go beyond stereotypical understanding of concepts. In the context of concurrency and parallelism, a typical example of such stereotypical understanding would be --- the lack of understanding of how message passing happens in real-life, and the inability to separate out the message passing abstraction from its implementation. Students often think that message passing has no connection with shared variables.



How Message Passing occurs in real-life

- Interrupt-driven communication
  - An interrupt happens to the CPU, whenever data is ready to be read.
    - To ensure mutually exclusive access of message buffers, disable interrupts while servicing the current interrupt.
    - Not captured at the application level send-receive we are studying!
- Or, the CPU polls (via certain sensors) at regular intervals to check whether data is available
    - Check whether data is available on the channel and then perform receive action, popularly known as polling.

24 CS3211 2012-13 by Abhik

As an illustration of how such stereotypical understanding may be stamped out, we would delve into how the message passing abstraction may be implemented. They also realize that implementing the message passing abstraction may also involve managing shared variables – such as the shared message buffers. This separation between the abstraction and the implementation (and acquiring an understanding of both), brings a level of comfort in the students' minds about the topics – as opposed to understanding only the programming abstraction.

Sample student feedback: Overall, the students have appreciated the dedication and effort I have put in to encourage and inspire them to delve into the difficult concepts on concurrency and parallelism. Following are some of the feedback I have received.

2012-13 semester 2 feedback for CS3211

"Very very sincere and responsible. Very student focused approach which makes one develop an instant liking for the Professor and the subject. Explains in an amazing manner, humorous and very very good. On the whole, great job!"

"Serious overdosage of awesomeness inside him :D Ready to answer queries, however insignificant they might be to my class mates. Holds Saturday sessions for our sake (how cool is a Professor who is ready to spend time away from his family for our sake?). Swift e-mail replies. Understanding. Knowledgeible. You, sir, are a legend!"

2009-10 semester 2 feedback for CS3211

"Very nice person who is willing to go beyong to pass on knowledge to students. I will always remember the day when his father passed away but still came to lecture with a stable mind and a strong heart to complete the lecture and only releasing the news of his father at the end. I trully appreciate this dedicated nature of this lecturer. Not only that but he is always willing to answer students quesions with a smile and an open-mind. He welcomes student feedback every lecture and doesn't mind going over some sections of the lecture just in order to make students understand. I wish I could've performed better in his term test give the fact that he is such a dedicated and enthusiastic lecturer."

"One of the best professors I have had the opportunity to learn from. Is very knowledgeable in his area of expertise. Always positive and encouraging.. has a great sense of humour. Is very dedicated and sincere. Can spot and encourage the better students as well as help weaker students. Can be firm as well as caring. REally liked the module"

B. Gaining feedback in each lecture via post-it notes as an additional measure

I have put in measures to gain more and more technical feedback on which topics the students may find difficult. This is not only done for my modules, but for every single lecture. Prior to every single lecture, I go around and place one post-it note in each of the student's seat. This is done before the lecture starts, so that none of the lecture time is wasted. Subsequently, the students can put in any questions they may have in the post-it note and simply leave it at my desk, or at the door of the lecture theater.

It should be clarified that this measure is not put in to address any lack of interactivity in my lectures. As is evident in many of my student feedback over the semester, the students find my lectures to be very interactive with lot of discussions. However, at the end of the lecture, sometimes there are students who line up to ask additional questions, and sometimes when I am answering the questions for one student – the other students may have to wait which they may not prefer. For this reason, the post-it notes are provided.

At the end of every lecture – I collect the post-it notes and send out the answers to the questions in the post-it notes to the whole class via email. This gives the students an additional avenue for asking questions. Here is a sample feedback from a student on this facility of taking feedback from students in every lecture via post-it notes

Sample feedback for CS4271

```
Really gives his best in making the students understand. His idea of gaining
feedback from students after each class is impressive. Keeps asking again and
again to make sure if everyone understands.
```

C. Changes made to my teaching based on student feedback

Over the years, I have incorporated many changes in my teaching based on student feedback. I only mention a few examples here. In some of my modules, I have initiated a scheme of peer-review for certain components in the modules. For example in CS 6880 Advanced Topics in Software Engineering – the students need to make a paper presentation and this goes through an assessment by me, as well as their peers. This is particularly possible since this is a graduate module, and the students are expected to be more mature.

As mentioned earlier, I have initiated the scheme of putting in post-it notes on students' desks prior to start of the lecture. Subsequently I collect them at the end of the lecture and collate any post-it notes containing questions from students. The answers to these questions are broadcast to the whole class by email, immediately after the lecture. This provides an additional avenue of interaction to the students, apart from consultation sessions, and interactions during the lecture.

I have also initiated warm-up questions at the beginning of lectures – particularly when we continue a topic from the past lecture. This is because students may not fully recall the main concepts covered in the last lecture – so simply asking some warm-up questions and exercises at the beginning of the lecture, helps the students to glide into the topic with full attention.

Finally and most importantly, I have integrated more of recent research developments either by my own research, or by other research groups worldwide – into teaching. Examples on this front include recent progresses in automated debugging, and on symbolic execution based testing into my upcoming module CS 4218 Software Testing and Debugging. By indicating the upcoming research developments in the classroom teaching, not only are the students inspired further, it will also make them ready for the future decades since the technology of tomorrow comes from the important research results of today.

IV. Supervision by Abhik Roychoudhury

Email: abhik@comp.nus.edu.sg
http://www.comp.nus.edu.sg/~abhik

In the following, I elaborate on my research student supervision record. The section is divided into two parts – PhD student supervision, Master student supervision and Post-doc supervision. For each of these parts, I mention the achievements by the advisees, and their placement information.

A. PhD student supervision

Overall data for graduated students: So far, I have supervised the PhD thesis of eight (8) students from diverse nationalities. Out of these graduated students, three have moved to academia, four have moved to industry and one has moved to industrial research. In terms of post-PhD employment, three of the eight students are employed in Singapore, while the remaining are employed outside Singapore. Following are the eight PhD thesis supervised by me.

	From	To	Name	Thesis Title
1	2008/2009	2012/2013	QI DAWEI	Semantic Analyses to Detect and Localize Software Regression Errors
2	2008/2009	2012/2013	SUDIPTA CHATTOPADHYAY	Time-predictable execution of embedded software on multi-core platforms
3	2007/2008	2012/2013	SANDEEP KUMAR	Mining behavioral specifications of distributed systems
4	2006/2007	2010/2011	JU LEI	Model-driven timing analysis of embedded software
5	2004/2005	2009/2010	VIVY SUHENDRA	Memory Optimizations for Time-Predictable Embedded Software
6	2003/2004	2009/2010	ANKIT GOEL	Parameterized validation of UML-Like models for reactive embedded systems
7	2002/2003	2007/2008	WANG TAO	Post-Mortem Dynamic analysis for software debugging
8	2000/2001	2005/2006	LI XIANFENG	Microarchitecture modeling for timing analysis of embedded software

Best PhD thesis awards: Two of my graduated PhD students have won the Best PhD thesis award from NUS School of Computing. This award is a recognition of the potential industrial impact of their research on software testing. The two PhD theses which received the Best PhD thesis award were:

- Wang Tao, Post-mortem dynamic analysis for software debugging, graduated 2008
- Qi Dawei, Semantic analyses to detect and localize software regression errors, co-supervised with Liang Zhenkai, graduated 2013

Fellowships won by PhD students: Out of my eight graduated PhD students, three of them won the NUS Presidential Graduate Fellowship during their PhD studies – Wang Tao, Qi Dawei, Sudipta Chattopadhyay. Two of my graduated PhD students won the prestigious Microsoft Research Fellowship during their PhD studies – Vivy Suhendra and Wang Tao. Several of my graduated PhD students received the Dean's Graduate Award for Research Excellence (given out by NUS School of Computing), during their PhD studies.

Placement of graduated students in research positions: Several of my PhD students have moved to research positions in Singapore and abroad. These include

- Dr. Li Xianfeng, graduated 2006, moved to Peking University, currently Associate Professor
- Dr. Vivy Suhendra, graduated 2010, moved to I2R Singapore as Research Scientist.
- Dr. Ju Lei, graduated 2010, moved to Shandong University directly as Associate Professor.

Existing PhD students: Apart from the eight PhD students I have supervised – I am currently supervising a group of five PhD students from diverse nationalities. They are

- Marcel Bohme, Dipl. From T U Dresden (Germany), expected graduation 2014.
- Abhijeet Banerjee, Bachelors from Bengal Science and Engineering University (India), expected graduation 2016.
- Shin Hwei Tan, Masters from UIUC (USA), expected graduation 2016.
- Sergey Mechtaev, Masters from St. Petersburg State Univ. (Russia), expected graduation 2017.
- Thuan Pham Van, Masters Ho Chi Minh City Univ of Tech (Vietnam), expected graduation 2017.

B. **Supervision of masters students and research assistants**

So far, I have supervised the thesis of six (6) Masters by research students. Out of these six students, five (5) are employed in Singapore after their studies and one is employed abroad. Several of my graduated Masters students have moved to industrial positions in Creditsuisse, Mentor Graphics and Creative Technologies. The list of masters students graduated by me are provided in the following.

	From	To	Name	Thesis Title	Examiner Category	Degree	Year Conferred	Student Status
1	2010/2011	2010/2011	HUYNH BACH KHOA	Scope-aware data cache analysis for WCET estimation.	Sole	MASTER OF SCIENCE	2011	GRADUATED
2	2009/2010	2009/2010	LIU SHANSHAN	Model checking parameterized process classes	Sole	MASTER OF SCIENCE	2010	GRADUATED

	From	To	Name	Thesis Title	Examiner Category	Degree	Year Conferred	Student Status
3	2003/2004	2005/2006	TRAN TUAN ANH	Automatic generation of protocol converters from scenario-based specifications	Co-Supervisor	MASTER OF SCIENCE	2006	GRADUATED
4	2002/2003	2004/2005	SHEN QINGHUA	Impact of Java memory model on out-of-order multiprocessors	Main	MASTER OF SCIENCE	2004	GRADUATED
5	2002/2003	2004/2005	HEMENDRA SINGH NEGI	Two concrete problems in timing analysis of embedded software	Co-Supervisor	MASTER OF SCIENCE	2004	GRADUATED
6	2001/2002	2003/2004	XIE LEI	Performance impact of multithreaded Java semantics on multiprocessor memory consistency models	Sole	MASTER OF SCIENCE	2003	GRADUATED

Apart from the six Masters students, I also supervised two research assistants – who have finished their PhD studies at other universities – University of Wisconsin and University of Maryland respectively.

C. **Supervision of Post-doctoral Research Fellows**

So far, I have worked with five (5) post-doctoral research fellows. Three of them have finished their employment in my research group, and two research fellows are currently working with me.

The three post-docs who have finished their employment have moved to academic positions in Peking University, Indian Statistical Institute and Hong Kong University. Out of these, most recently Dr. Bruno C.d.S. Oliveira (PhD Oxford), finished his stint in my group and joined Hong Kong University as Assistant Professor from 1st September 2013.

Currently, I am working with two post-doctoral research fellows in my group. They are Dr. Jooyong Lee (PhD Aarhus Univ. Denmark) and Dr. Clement Ballabriga (PhD Univ of Toulouse, France).

D. **Supervision of undergraduate students for final year project**

Following are the undergraduate students whom I have supervised for their final year project. Two of these projects have been published, and the published papers have a good number citations as well.

(2001-02) (S.R. Karri) (Verification of AMBA bus protocol) This was written as a paper “Using formal techniques to debug the AMBA system-on-chip protocol” which was published in DATE 2003. The paper has *74 citations* in Google scholar as of September 2013.

(2002-03) (Zhan Jia) (Multi-threaded Java from Multi-processor perspective)

(2003-04) (K.K. Subramanian) (Extending algorithmic searches for Design Space Exploration of Embedded Systems)

(2003-04) (Shishir K. Choudhary) (Symbolic Execution of Behavioral Requirements). This was written as a paper with the same name, which was published in PADL 2004. The paper has *22 citations* in Google Scholar as of September 2013.

(2004-05) (Ju Lei) (Tracing methods to help multi-threaded program debugging)

(2004-05) (Luo Xue) (A Play-in front-end to a Live Sequence Chart symbolic simulator)

(2005-06) (Mustafa Yucefabdali) (Search optimizations for model checking of C# programs)

(2005-06)(Chua Chong Tat) (Improved instrumentation methods for software fault localization)

(2006-07)(Low Shek Chian) (Verification of Interacting Process Classes using PVS prover)

(2006-07)(Tan Kelly)(Verification of Live Sequence Charts using PVS prover)

(2007-08)(Choo Wei Chern)(Explanation of counter-examples in SPIN for education purposes)

(2008-09)(Samuel Risandy)(Integrating hierarchical dynamic slicing to JSlice)

(2012-13)(S. Subhasree)(Combining Software Debugging and Program Repair)

(2012-13)(Lei Dong)
(Use of software version debugging methods for programming education – user studies)

(2012-13)(Srestha Vijayaraghavan)
(Use of software version debugging methods for programming education – tool construction)

E. **Thesis committees participated in**

I have participated in ten thesis committees. Out of them six are PhD thesis committees, and four are Masters thesis committees. The full details appear in the following.

	From	To	Name	Degree	Role	Year
1	2011/2012	2011/2012	SURAJ PATHAK	Master of Science	INTERNAL EXAMINER	2012
2	2003/2004	2009/2010	CHEN CHUNQING	DOCTOR OF PHILOSOPHY	INTERNAL EXAMINER	2009
3	2002/2003	2006/2007	SUN JUN	DOCTOR OF PHILOSOPHY	INTERNAL EXAMINER	2006
4	2002/2003	2010/2011	SIM JOON, EDWARD	Doctor of Philosophy	INTERNAL AND ORAL PANEL MEMBER	2010

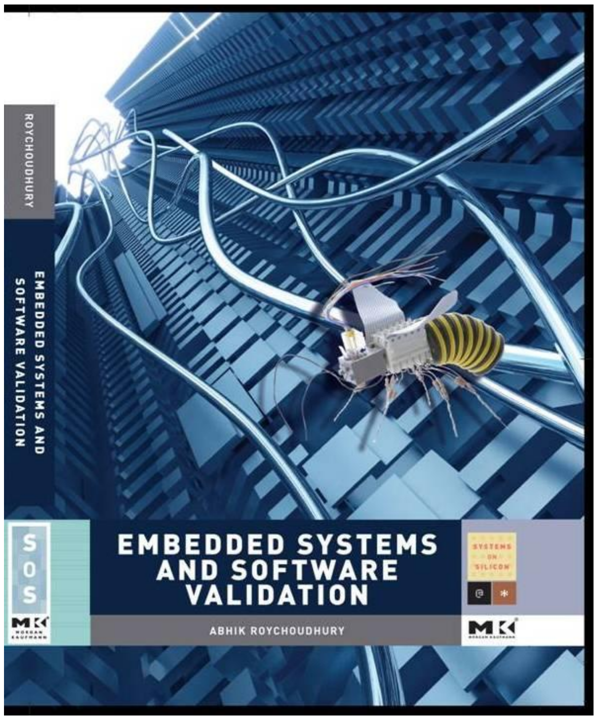
5	2002/2003	2006/2007	HAMID ABDUL BASIT	DOCTOR OF PHILOSOPHY	INTERNAL EXAMINER	2007	GRADUATED
6	2002/2003	2008/2009	KATHY NGUYEN DANG	DOCTOR OF PHILOSOPHY	THESIS COMMITTEE MEMBER	2009	GRADUATED
7	2001/2002	2003/2004	XU NA	MASTER OF SCIENCE	INTERNAL EXAMINER	2003	GRADUATED
8	2001/2002	2003/2004	KAMRUL HASAN TALUKDER	MASTER OF SCIENCE	INTERNAL EXAMINER	2004	GRADUATED
9	2001/2002	2008/2009	POPEEA CORNELIU CHRISTIAN	DOCTOR OF PHILOSOPHY	INTERNAL EXAMINER	2008	GRADUATED
10	1999/2000	2007/2008	ANDREW EDWARD SANTOSA	DOCTOR OF PHILOSOPHY	INTERNAL EXAMINER	2008	GRADUATED

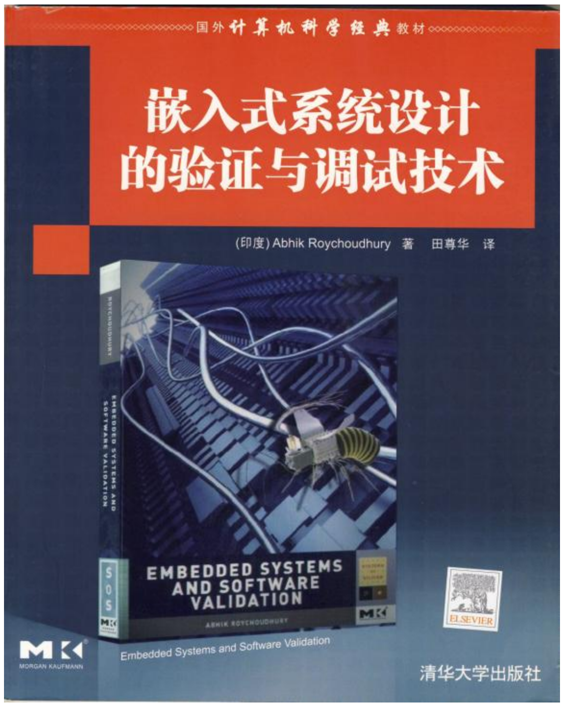
V. Other Educational Activities by Abhik Roychoudhury

Email: abhik@comp.nus.edu.sg
http://www.comp.nus.edu.sg/~abhik

In this section, I highlight one significant educational activity I have undertaken – authoring of a textbook on embedded system design and validation.

A. Information about the book




嵌入式系统设计的验证与调试技术
(印度) Abhik Roychoudhury 著 田尊华 译
Embedded Systems and Software Validation
清华大学出版社

Book cover	Cover of the Chinese translation

While teaching the module CS4271 Critical Systems and their Verification for several years – I became acutely aware of the need for a text-book in this area. There were indeed several books on formal verification or model checking – which I used to consult in my earlier offerings of the module. However, these books completely lacked the system design perspective and described the material on model checking from a theoretical perspective. On the other hand, textbooks on embedded systems did not contain a rigorous design and verification perspective. Around 2007, I was approached by Morgan Kaufmann (Elsevier) to write a textbook on this topic under their prestigious Systems-on-Silicon series. I undertook the effort, and the textbook has subsequently been adopted in various universities in Europe, USA and Asia. The book was published in January 2009

A Chinese translation of the book was initiated by Tsinghua University Press in 2011. The Chinese translation has been adopted for teaching in Chinese universities.

Excerpts from the book were featured as “Editors Top Picks” in EE times, a popular industry magazine with embedded developers. This appeared in the edition on 6th August 2012.

B. From the book cover

Modern embedded systems are a part of every modern electronic device, ranging from toys to traffic lights to nuclear power plant controllers. These systems help run factories, manage weapon systems and enable the worldwide flow of information, products and people. Unlike other computer systems such as those that operate personal computers, embedded systems must typically run error-free for years or even decades with little or no opportunity to reboot the system or fix problems. Such systems typically consist of a heterogeneous collection of processors, specialized memory subsystems and partially programmable or fixed-function components. This heterogeneity, coupled with issues such as hardware/software partitioning, mapping and scheduling leads to a large number of design possibilities, making both functionality and performance validation of such systems a difficult problem and an imperative issue.

Roychoudhury guides readers through a host of debugging and verification methods critical to providing reliable software and systems applications. Readers will find practical information and guidance including:

- Complete coverage of the major abstraction levels of embedded systems design, starting from software analysis and micro-architectural modeling, to modeling of resource sharing and communication at the system level;
- Integrates formal techniques for validation for hardware/software with debugging methods for embedded systems;
- Includes practical case studies to answer the questions: does a design meet its requirements, if not, then which parts of the system are responsible for the violation, and once they are identified, then how should the design be suitably modified?

C. Preface of the book – explaining its differentiation with other books

This book attempts cover the issues in validation of embedded software and systems. There are lot of books on “embedded software and systems" as a web search with the appropriate search terms will reveal. So, why this book?

There are several ways to answer the question. The first, most direct, answer is that the current books mostly deal with the programming and/or co-design of embedded systems. Validation is often discussed almost as an after-thought. In this book, we treat validation as a first class citizen in the design process, weaving it into the design process itself.

The focus of our book is on validation, but from a embedded software and systems perspective. The methods we have covered (testing/model-checking) can also be covered from a completely general perspective, focusing only on the techniques, rather than how they fit into the system design process. But we have not done so. Even though the focus of the book is on validation methods, we clearly show how it fits into system design. As an example, we present and discuss the model checking method twice in two different ways - once at the level of system model (Chapter 2) and again at the level of system implementation (Chapter 5). Finally, being rooted in embedded software and systems, the focus of our book is not restricted to functionality validation. We have covered at least two other aspects - debugging of performance and communication behavior. As a result, this book contains analysis methods which are rarely found in a single book - testing (informal validation), model checking (formal validation), worst-case execution time analysis (static analysis for program performance), schedulability analysis (system level performance analysis) and so on - all blended under one cover, with the goal of reliable embedded system design.

As for the chapters of the book, Chapter 1 gives a general introduction to the issues in embedded system validation. Differences between functionality and performance validation are discussed at a general level.

Chapter 2 discusses model-level validation. It starts with a generic discussion on system structure and behavior and zooms into behavioral modeling notations such as Finite-state machines (FSMs) and Message Sequence Charts (MSCs). Simulation, testing and formal verification of these models are discussed. We discuss model-based testing, where test cases generated from the model are tried out on the system implementation. We also discuss property verification, and the well-known model checking method. The chapter wraps with a nice hands-on discussion on practical validation tools such as SPIN and SMV. Thus, this chapter corresponds to model-level debugging.

Chapter 3 discusses the issues in resolving communication incompatibilities between embedded system components. We discuss different strategies for resolving such incompatibilities, such as endowing the components with appropriate interfaces, and/or constructing a centralized communication protocol converter. Thus, this chapter corresponds to communication debugging.

Chapter 4 discusses system level performance validation. We start with software timing analysis, in particular Worst-case Execution Time (WCET) analysis. This is followed by the estimation of time spent due to different interferences in a program execution from the external environment, or due to other executing programs on same/different processing elements. Suitable analysis methods to estimate the time due to such interferences are discussed. We then discuss mechanisms to combat execution time unpredictability via system level support. In particular, we discuss compiler controlled memories or scratchpad memories. The chapter concludes with a discussion on time predictability issues in emerging applications. Thus, this chapter corresponds to performance debugging.

Chapter 5 discusses functionality debugging of embedded software. We discuss both formal and informal approaches, with almost equal emphasis on testing and formal verification. The first half of the chapter involves validation methods built on testing or dynamic analysis. The second half of the chapter concentrates on formal verification, in particular, software model checking. The chapter concludes with a discussion on combining formal verification with testing. Thus, this chapter corresponds to software debugging.

Apart from some debugging/validation methods being common to Chapters 2 and 5, the readers may try to read the chapters independently. A senior undergraduate or graduate course on this topic may however read the chapters in sequence, that is, chapters 2, 3, 4, 5.

D. Invitation e-mail from Morgan Kaufmann (Elsevier) for writing the book

Following is the e-mail invitation I received from Morgan Kaufmann for writing a book on debugging and validation of embedded systems.

-------- Original Message --------

Subject: Debugging Embedded Systems...the BOOK?

Date: Wed, 17 Jan 2007 10:17:27 -0600

From: Glaser, Chuck (ELS-BUR) <C.Glaser@elsevier.com>

To:<abhik@comp.nus.edu.sg>

Dear Professor Roychoudhury:
I have taken the liberty of writing you today in my capacity as Senior Editor with Elsevier. I work closely with Wayne Wolf on the publication of our Morgan Kaufmann Series in Systems on Silicon. I noticed with keen interest that you gave one of the tutorials at VLSI 2007 on "Performance Debugging of Complex Embedded Systems" and I wonder if you have given any thought to working on a book-length treatment of this topic, either as author or as editor, with various, contributing authors?

If you would be interested in working on a Debugging Embedded Systems book, or if there are other book projects growing out of your current research, I would be very pleased to discuss the details with you.

Thank you in advance for your consideration.

Sincerely,
Chuck Glaser

PS: FYI, I have attached two files describing the series and listing titles…. Also, fyi, we pay our authors competitive royalties, we price our books to market (the most expensive in the series is $79.95), and we produce our books to the highest possible standards. I hope we can talk!

****
Charles B. Glaser
Senior Acquisitions Editor
Elsevier

E. Comments from instructors who adopted the book in their modules

Finally, following are some sample comments from instructors who have adopted my book in their teaching (as communicated to me via e-mail).

-----Original Message-----
From: Abhik Roychoudhury [mailto:abhik@comp.nus.edu.sg]
Sent: Tuesday, September 15, 2009 10:41 AM
To: J. Park
Subject: Re: Materials on your book

J. Park wrote:

Dear Prof. Roychoudhury:

I've chosen your book as a textbook of my lecture on the developing an embedded software.
I think this book is good for the concept and practical issues on embedded software verification.
I'm just wondering whether you have a kind of teaching materials such as powerpoint file or figures used in your book?

If you could provide any kind of teaching material, it would be greatly helpful for me to prepare my lecture.
Thank you very much for such a great book.
Regards,

Jaehyun Park
Professor
Inha University, Korea

-------- Original Message --------

Subject: About your book

Date: Wed, 29 Jun 2011 09:11:57 +0530

From: Pallab Dasgupta <pallab@cse.iitkgp.ernet.in>

To:<abhik@comp.nus.edu.sg>

Dear Abhik,

This is to congratulate you for your book "Embedded Systems and Software Validation", which is a wonderful contribution in an area where very few texts are available. You work is a great contribution to academia in this area.

I shall be teaching a course component on Embedded Systems and Software Validation in the forthcoming Autumn semester, for which I intend to use your book as the main reference. I got the link to this book from Dr Ansuman Banerjee and have been able to procure this book from our bookstore.

Ansuman told me that you share your powerpoint slides for this book -- may I get these?

Kind regards,

Pallab

==
Dr. Pallab Dasgupta,
Professor, Dept. of Computer Sc & Engg,
Indian Institute of Technology Kharagpur, INDIA 721302.

Phone: +91-3222-283470 (Off) 283471 (Res)
Fax: +91-3222-278985
Email: pallab@cse.iitkgp.ernet.in
Web: http://www.facweb.iitkgp.ernet.in/~pallab
==