Brandenburg
University of Technology
Cottbus - Senftenberg

DAAD

DCPS

DCPS Workshop / Tutorial Announcement

for a 2-Day Tutorial on

"Formal Verification in Hardware and Software"

We announce a tutorial and workshop within the frame of the DCPS Project, supported by the German Academic Exchange Service (DAAD). The event will take place on Thursday, March 26th and Friday, March 27th, 2015, at BTU Cottbus-Senftenberg (main Campus).

Schedule:

March 26th:
10:00 Opening and Welcome

10:10 - 12:40 Tutorial on "Automated correction of bugs in designs and programs: from fiction to reality" by Prof. Jaan Raik, TUT Tallinn, Deptm. of Computer Engineering

12:40 – 13:30 Lunch break

13:30 – 16:00 Tutorial on "Formal verification in FPU design, deadlock detection and redundancy recognition" by Udo Krautz, IBM Deutschland Development GmbH, Böblingen

March 27th:

9:00-11:30 Tutorial on "Time predictable architecture solution of a multitask real time system" by Prof. Andrzej Pulka, Silesian University of Technology, Gliwice

11:30 – 12:30 Lunch Break

12:30 – 15:00 Tutorial on "A Practical Introduction into Software Model Checking for Embedded Systems" by Thilo Vörtler, Fraunhofer-Institute of Integrated Circuits (IIS), Deptm. EAS (Electronic Design Automation), Dresden.

For external participants / visitors:

Guests from partner universities and associated partners are, as always, welcome. Please, register with Kathleen Galke at BTI-CS:
kl@informatik.tu-cottbus.de

Please, note: Travel can be supported by funds from DAAD for students and scientists from Tallinn, Liberec, Poznan and Gliwice with 200 € (travel only) and 50 € per day.

Outline of Tutorials:

Thilo Voertler: A Practical Introduction into Software Model Checking for Embedded Systems

This tutorial will give an introduction into Software Model Checking based on the model checker CBMC. Model Checking has had great success for the verification of digital hardware. Due to the large state space of software it is often not directly applied to software but rather to simplified models of the actual software. In recent years however model checking tools, which directly verify the source code of software applications, have been developed. CBMC is such a model checking tool, which supports the verification of applications written in ANSI-C. The main focus of these model checking tools are embedded system application which are often used in safety critical applications.
In this tutorial an introduction into formal software verification and especially model checking will be given. The bounded model checking algorithm will be presented and the principal translation from source code into a model checking problem will be shown. Using example programs the CBMC tool suite will be introduced and practical problems when applying model checking will be discussed.
Requirements for the course:
- Basic knowledge of ANSI-C and Boolean algebra
- Duration 2 hours

Jaan Raik: Automated correction of bugs in designs and programs: from fiction to reality

In this talk Prof. Jaan Raik from Tallinn University of Technology shares his experiences in automated error localization and correction techniques and tools, also known as automated debug.
Prof. Raik acted as the coordinator for the EU's FP7 DIAMOND collaborative research project, where several European companies and universities were attempting to solve many of the problems that were remaining in automated debug. The talk will cover different techniques for error localization, including statistical and SAT-based ones. For automated correction of, re-synthesis and mutation-based approaches will be considered. Particular stress will be put on readable and small (localized) correction techniques.
The talk will also present some of the open source tools developed during the DIAMOND project. This includes a Formal Repair Engine for Simple C programs (FoREnSiC) as well as the zamiaCAD framework for bug localization in VHDL designs. Finally, a case study of a bug localization on a real commercial processor using zamiaCAD will be presented.

Udo Krautz: Formal verification in FPU design, deadlock detection and redundancy recognition

Floating point units (FPUs) have always been difficult to verify due to their vast state space and large amount of data path corner cases. For designs running at >5GHz such as IBM's System Z this is augmented by the increasing amount of design 'tricks' which have to be used in the implementation due to circuit limitations. Formal verification of such designs is crucial since no other method can provide enough confidence about their correctness. In the first part of the tutorial we'll discuss basic FPU data path concepts and give an insight how these are formally verified in IBM's verification process. In the second part we discuss pipeline control aspects of FPUs and introduce examples for liveness checks. Furthermore will the tutorial present other applications for formal methods that are used within IBM on hardware designs that are not strictly verification related. We'll show how a formal analysis can be used to determine redundancy in designs to identify potential optimisation.

Andrzej Pułka: **Time predictable architecture solution of a multitask real time system**

The tutorial discuss the problem of time predictability and repeatability of modern electronic embedded systems. The precision time machine (PRET) paradigm formulated in UC Berkeley in 2007 briefly recalled as a starting point. Main problems of multitasking and multi-threading and are discussed and research objectives of the time predictable real-time systems is identified. A brief survey of the related works is given. The Idea of thread interleaving is described and the architecture of the proposed time predictable embedded system is presented and the original UC Berkeley solution with the memory wheel is emphasized. Two PRET models developed in Silesian University of Technology are presented: HDL synthesized model and system level SystemC-simulation model. Both models are based on the original (SUT) solution of the dynamic interleave controller of threads (DICT) with dynamically modified (reconfigured) priorities of tasks (threads). We will investigate the model of the system and simulation results. In HDL model we will analyze the synthesized hardware structure.
In SystemC – high level modeling environment we will focus on scheduling algorithms that enable optimal utilization of the processing units. We will also show the main memory access control unit and the original memory system organization. Many simulation experiments conducted with single and multi-core system architecture implementation, will show results proving advantages of the presented system-level model.

Requirements for the course:
- Pipeline processing fundamentals, fundamentals of digital system design
- Duration 2 hours