**LISHA Research Colloquium 2013/2**

The Colloquium will be held at the Digital Systems Laboratory located in the Electrical Engineering building.

- Linear Temporal Separation Logic 3, by Mateus Krepsky Ludwich

Abstract: This talk introduces the Linear Temporal Separation Logic 3 (LTSL3), a logic that combines Linear Temporal Logic, and Separation Logic, and is able to express changes in pointer-based data structures along linear time. LTSL3 is suitable for both static formal verification, and runtime verification as it defines satisfaction relation over two, and three values of true. By introducing artificial states, representing pre, and post conditions of commands, and by using them in combination with temporal operators, LTSL_3 is able to express properties that must hold in a system, only at specific states. We have demonstrate the use of LTSL_3 by one example regarding the thread dispatching in an operating system, and by examples regarding a red-black tree data structure.

- *A secure key bootstrapping protocol for Wireless Sensor Networks*, by Davi Resner

Abstract: Most of encryption and authentication techniques that were developed and validated for the original internet that we use today were not designed with the concepts of Wireless Sensor Networks in mind. This is a scenario where time and energy efficiency is very important, because the microcontrollers used in smart objects will seldom be powerful enough to afford wasting any of these resources. With this in mind, we propose a protocol for establishing cryptographic keys specially designed for wireless sensor networks, aiming at enabling sensors to communicate with a gateway with the principles of authenticity, integrity and confidentiality granted at a low time, memory and energy cost.

- *An Implementation of the AES cipher using HLS*, by Rodrigo Meurer

Abstract: The Advanced Encryption Standard (AES) is the main algorithm used to ensure security and privacy in several different applications ranging from massive data servers to small low-power embedded systems. Such embedded systems often rely on dedicated hardware implementations of AES in order to meet tight power budgets. In this scenario, C/C++ High-Level Synthesis (HLS) solutions are gaining acceptance as traditional hardware design methodologies can no longer match the strict time-to-market requirements of current applications. In our work, we describe a C++ implementation of the AES algorithms and explore different hardware micro-architectures by using HLS solutions. We focus on describing the process of obtaining an efficient synthesizable C++ description from plain software code.

- *Porting EPOS to the Zynq-7000 ARM Cortex A9 platform*, by Bruno Farias de Loreto

Abstract: As the demand of computational power in embedded systems increases, so the demand for basic software for those hardwares increases. There is a big pressure for the software development of embedded systems, as there is an increasing number of smart devices that performs a wide variety of tasks. In order to meet this demand, the operating system EPOS is used, so that developers can focus their energy on their applications instead of developing new system for each new hardware. Zynq-7000 is a SoC platform with a huge processing power for a embedded system, as it features a multi-core ARM Cortex A9 processor and a fairly big RAM. Because EPOS has to be flexible and able to give a transparent environment to the developer, the porting of EPOS to the Zynq platform is needed, and this is what will be covered in this paper. The possibility to work with a multi-processor will also enable a variety of works that could only be accomplished in a CISC processor.

- *An FPGA based hardware implementation of the IEEE 802.15.4 protocol*, by João Gabriel Reis

Abstract: The IEEE 802.15.4 is the de-facto wireless communication standard for the Internet of Things (IoT). Although quite big and complex, it falls short on key issues such as trickle and low-power listening. In order to explore new low-level protocols, we propose an all-programmable-logic 802.15.4 radio. The architecture will be easily expandable and tunable driving the radio developers to incorporate sensitive components of cross-layer, secure, location-aware, time-aware communication protocols directly into hardware.

- *Adaptive Scheduling for Energy-Aware Real-Time Systems*, by Arliones Hoeller

Abstract: In embedded computing systems, real-time and energy requirements build antagonistic operation constraints. Actions to reduce energy consumption either reconfigure or disable system components, degrading components' performance or creating delays to switch components on and off, ultimately affecting the response time of real-time tasks. In this scenario, this work explores adaptive scheduling techniques to adapt tasks to constrained environments.

- *Real-Time Operating System Support for Multicore Applications*, by Giovani Gracioli

Abstract: Shared cache partitioning is a well-known technique used in multicore real-time systems to isolate task workloads and improve system predictability. Presently, the state-of-the-art studies that evaluate shared cache partitioning on multicore processors lack two key issues. First, the cache partitioning mechanism is typically implemented either in a simulation environment or in a general-purpose OS, and so the impact of kernel activities, such as interrupt handlers and context switching, on the task partitions tend to be overlooked. Second, the evaluation is typically restricted to either a global or partitioned scheduler, thereby by falling to compare the performance of cache partitioning when tasks are scheduled by different schedulers.

In this work, we design and implement a shared cache partitioning mechanism in a multicore component-based RTOS capable of assigning partitions to internal OS data structures, including task and system stacks and interrupt handlers data. We evaluate our shared cache partitioning mechanism running task sets under global (G-EDF), partitioned (P-EDF), and Clustered-EDF (C-EDF) multicore real-time scheduling algorithms. Our results indicate that a lightweight RTOS does not impact real-time tasks, and shared cache partitioning has different behavior depending on the scheduler and the task's working set size. Moreover, we present a task partitioning algorithm that is aware of tasks' cache partitions and is able to provide real-time garantees when those tasks share cache lines.