Multicore Programming
- an ICES Seminar

Wednesday 24th November 2010
12.00 – 17.15
KTH Kista Campus, Lecture Room C2, Electrum 1, Isafjordsgatan 20-26 (trapphus C, plan 2)

The trend to increase the number of cores on each chip is likely to continue during the forthcoming decade. As a consequence, the programming of multi-core and many-core processors has emerged as a main challenge. Although 40 years of research in parallel computing have shown that there are no easy solutions, the need to develop workable, efficient programming models, tools and methodologies for utilizing the massive parallelism available is becoming more urgent every year.

This seminar day will review the problem, that industry is facing, and give an overview of state of the art methodologies and tools to address it. Furthermore it will offer an analysis of current trends and what we can expect from the near future. Most importantly, it will provide a forum for exchange of thoughts, ideas and opinions on this timely topic.

The afternoon will wrap up with a Panel Debate “Embedded Multicore: Caches or local storage?” moderated by Associate professor Christian Schulte.

As computing becomes more parallel in many-core systems, access to memory and the memory architecture easily becomes a performance bottleneck and show-stopper for scaling. The trade-offs are hard and involve latency, bandwidth, power, predictability, ease of programming, portability, legacy code, and other important design objectives.

The panel will elaborate and discuss the architecture alternatives that we foresee:

- caches or local scratchpad memory
- central or distributed memory
- private or shared memory

A post event summary will be posted on the ICES homepage next week, and the presentations will be made available there as pdf documents. Please see www.ices.kth.se/

Thank you for taking part in today’s seminar!

We welcome your feedback and suggestions! Please don’t hesitate to get in touch: ices-admin@md.kth.se

Join the KTH Innovative Center for Embedded Systems Group on LinkedIn!
Multicore Programming  
- an ICES Seminar

Wednesday 24th November 2010  
12.00 – 17.15

KTH Kista Campus, Lecture Room C2, Electrum 1, Isafjordsgatan 20-26 (trapphus C, plan 2)

PROGRAM:

12-13 Registration, Lunch & Mngel

13.00 Welcome and Introduction  
Prof Mats Brossson, KTH ICT School

13.05 TERAC - Terascale Reconfigurable Architecture and Methodology  
Prof Ahmed Hemani, KTH

13.25 Manycore programming models: a case for Task Centric Programming  
Prof Mats Brossson, KTH ICT School

13.45 Developing Mobile Platforms  
Magnus Österholm, ST Ericsson

14.30 Coffee break and leg stretch

14.45 Programming Multicores in Erlang  
Joe Armstrong, Ericsson

15.30 Programming Tools for Embedded Multicore  
Jakob Engblom, Wind River

16.15 Coffee break and leg stretch

16.30 PANEL DISCUSSION  
Led by Associate Professor Christian Schulte and involving all of today’s speakers
SPEAKERS’ ABSTRACTS

TERAC - Terascale Reconfigurable Architecture and Methodology, Ahmed Hemani, (KTH)

TERAC is a coarse grain reconfigurable architecture concept. It achieves ultra high performance and computational and engineering efficiency. Massive parallelism, right granularity, domain specific resources, customization and a parallel distributed memory architecture integrated in 3D helps achieve the ultra high performance and computational efficiency. Regularity and hierarchical compile strategy together with annotated code is the basis for a a layered and pragmatic parallel programming model that will result in high engineering efficiency. The project is in initial stages of development and a proof of concept architecture and methodology for wireless applications is in advanced stage of development.

Manycore programming models: a case for Task Centric Programming, Mats Brorsson, (KTH)

Multicore processors are rapidly evolving to manycore. One definition of a manycore processor is when the number of cores have increased to the point where we no longer have effective programming models to fill the cores with useful work (source Wikipedia). At Kista Multicore Center, we investigate manycore programming models, their implementations, architectures and operating systems for industrial applications. We advocate to leave thread-centric programming models as the main parallel programming model in favour of task-centric models which may be combined with other models as needed.

In a thread-centric model, programmers explicitly create compute threads (e.g. pthreads) and manually assign work to these threads and manage their execution on parallel cores. This process is error-prone, difficult to analyze from a performance perspective, and not future-proof. With the rapid increase of the number of cores, a program written with threads easily becomes outdated as it is often tied to a specific number of cores. In contrast, in a task-centric programming model, programmers concentrate on exposing the available parallelism, expressing it in concurrent tasks. A run-time system maintains a pool of worker threads that helps out in executing the tasks in parallel. The worker threads can be managed manually in bare-metal platforms or by the operating system if such exists. Good implementations use private task-queues per worker thread and work-stealing is often employed to provide for good load balance. Task-centric models are rapidly gaining popularity since the incorporation of tasks in OpenMP 3.0. Other examples of task-centric models are Cilk++, Microsoft Task Parallel Library, Intel Threading Building Blocks and Apple Grand Central Dispatch. Current development consider expressing inter-task data dependencies for more efficient task and data scheduling and exposure of real-time and/or resource requirements.

In this presentation I will provide an overview of the trend of manycore architectures and programming models for embedded systems, and argue for how task-centric models will help developers get more out of these architectures. The presentation builds on experiences from industrial manycore projects and several EU research projects.

Developing Mobile platforms, Magnus Österholm, (ST Ericsson)

ST-Ericsson is developing platforms for the wireless industry. A number of challenges emerges during the development of a new product portfolio. First, systemization of hardware is important to get increased performance and scalability, at the same time as area and power dissipation is kept at a low level. Second, Multicore CPUs is one way to increase performance and flexibility. Third, methodologies for early software development and system validation is vital to reduce time to market.

Programming Multicores in Erlang, Joe Armstrong, (Ericsson)

Erlang programs are made up from large numbers of small parallel processes. These processes share no memory and communicate by pure message passing. Running an Erlang application on a multicore involves mapping these processes onto the individual cores in the multi processor. This can be done manually or automatically. Dividing the application into large numbers of small processes which share no data makes it relatively easy to build scalable fault-tolerant applications. This talk will touch on the following topics:

- Why "shared nothing" simplifies parallel programming
- How Erlang processes map onto multicores
- What kind of kind of multicore speedups can we expect
- What problems remain
- Who’s using Erlang and why

Programming Tools for Embedded Multicore, Jakob Engblom, (Wind River)

Multicore devices are entering the embedded realm just as fast as the general computing world. To keep up with the transition to multicore, embedded programming tools have to build on the traditions of embedded programming, while extending to handle multiple cores and concurrent targets. Multicore affects software architecture, runtimes, debuggers, and simulation tools. In this talk, we will mostly discuss simulation tools and multicore, but also shortly discuss other aspects of programming tools for multicore.