Agenda: Memocode 2008, June 5-7

Thu 7:30-9:00am	Breakfast
Thu 9:00-10:00	**Keynote: Bart Kienhuis (LIACS)** "Programming multicores with Kahn Process Networks; a smart choice?"
Thu 10:00-10:30	Break
Thu 10:30-12:00	**Formal Verification** • Omid Sarbishei, Bijan Alizadeh and Masahiro Fujita "Arithmetic Circuits Verification without Looking for Internal Equivalences" • Christophe Jacquet, Frédéric Boulanger and Dominique Marcadet "From Data to Events: Checking Properties on the Control of a System" • Graziano Pravadelli, Luigi Di Guglielmo and Franco Fummi "Vacuity Analysis by Fault Simulation"
Thu 12:00-1:30	Lunch
Thu 1:30-3:00	**Semantics of System Description Languages** • Subash Shankar and Masahiro Fujita "Rule-Based Approaches for Equivalence Checking of SpecC programs" • Nalini Vasudevan and Stephen A. Edwards "Static Deadlock Detection for the SHIM Concurrent Language" • Claude Helmstetter and Olivier Ponsini "A Comparison of Two SystemC/TLM Semantics for Formal Verification"
Thu 3:00-3:30	Break
Thu 3:30-4:30	**Poster Session** • G. Hoover, F. Brewer and C. Gill "Latency-Insensitive Implementation of Hardware/Software Interfaces" • R. Mateescu and E. Oudot "Efficient On-the-Fly Equivalence Checking using Boolean Equation Systems" • Katell Morin-Allory, Yann Oddos and Dominique Borrione "Horus: A tool for Assertion-Based Verification and on-line testing"
Thu 4:30-6:00	**Tools for Processor Design** • Steve Haynal, Timothy Kam, Michael Kishinevsky, Emily Shriver and Xinning Wang "A SystemVerilog Rewriting System for RTL Abstraction" • Michael Katelman, Jose Meseguer and Santiago Escobar "Directed-Logical Testing for Functional Verification of Microprocessors" • Daniel Grund and Jan Reineke. "Estimating the Performance of Cache Replacement Policies"
Fri 7:30-9:00am	Breakfast
Fri 9:00-10:00	**Keynote: Tevfik Bultan (UCSB)** "Infinite State Model Checking with Arithmetic Constraints"
Fri 10:00-10:30am	Break
Fri 10:30-12:00pm	**Models of Computation** • Christian Zebelein, Joachim Falk and Christian Haubelt "Classification of General Data Flow Actors into known Models of Computation" • Bijoy A. Jose, Sandeep K. Shukla, Hiren D. Patel and Jean-Pierre Talpin "On the Automatic Inference of Synchronization Logic for Multi-threaded Software Synthesis from Polychronous Specifications" • Yue Ma, Jean-Pierre Talpin and Thierry Gautier "Virtual prototyping AADL architectures in a polychronous model of computation"

Fri 12:00-1:30	Lunch
Fri 1:30-2:30	**Co-Design Contest** Over 30 teams competed during the month long contest with working results posted from 9 teams- the top teams will present their design and findings.
Fri 2:30-3:00	Break
Fri 3:00-4:30	Design Case Studies • Kermin Fleming, Chun-Chieh Lin, Nirav Dave, Jamey Hicks, Gopal Raghavan and Arvind. "H.264 Decoding: A Case Study in Late Design-Cycle Changes" • Eyad Alkassar, Peter Boehm and Steffen Knapp "Correctness of a Fault-Tolerant Real-Time Scheduler Algorithm and its Hardware Implementation" • Venkatram Vishwanath, Lenore Zuck and Jason Leigh "Specification and Verification of LambdaRAM: A Wide-area Distributed Cache for High Performance Computing"
Fri 4:30-6:00pm	**Panel: Methodologies for On-Chip Communication Design: Trends and Challenges** • Organizer: Luca Carloni Panelists: Anthony Chun(Intel) Marcello Coppola (STMicro) Nikil Dutt (UCI) Radu Marculescu (CMU) Drew Wingard (Sonics)
Fri 6:30-8:00pm	Banquet (on-site) Presentations and Concluding Remarks
Sat7:30-9:00	Breakfast
Sat9:00-5:00pm	**Tutorial: Hands-on Introduction to BSV: (Bluespec SystemVerilog)** • Speakers: Arvind (MIT) Co-founder Bluespec Inc. Rishiyur Nikhil, CTO, Co-founder Bluespec Several Volunteer MIT Students to assist

BSV is a modern, fully synthesizable design language in which all behavior is expressed with Guarded Atomic Actions (rewrite rules). Rules can be systematically composed from fragments across module boundaries using atomic transactional interfaces. BSV has powerful abstraction mechanisms such as expressive and polymorphic types with overloading and strong static type-checking, full orthogonality (all types are first-class), and Turing-complete static elaboration. Thus, BSV is scalable to large, industrial-strength SoCs even while designs remain highly parameterized and succinct.

In this tutorial, you will get a solid technical introduction to BSV and learn how it improves many aspects of modern SoC development: modeling, early SW development, architecture exploration, design, verification, and long-term evolution and maintenance. The lectures will be organized around a few serious examples and we will examine and analyze excerpts of their actual source code.

The tutorial is also hands-on: Participants who bring their laptops will receive a non-commercial but full-featured short-term installation of the latest release of BSV (native under Linux, and via a VMWare image for other OSs). During the tutorial you will work with lab exercises tied to the lecture content. After the tutorial you will be able to continue your own exploration with plenty of other examples and lab exercises.