**Generalizing the ISA to the ILA: A Software/Hardware Interface for**

**Accelerator-rich Platforms**

Sharad Malik (Princeton University)

*Abstract*

The Instruction-Set Architecture (ISA) has long served as the software/hardware interface for programmable processors. The ISA simultaneously serves as a specification for the hardware implementation, and as an abstraction of the hardware for software development. With the advent of multiprocessors, the memory consistency model (MCM) provided the software/hardware interface for processor interactions through shared memory. We are now in an era where accelerator-rich platforms are widely used to deliver the power-performance requirements of emerging applications. Unfortunately, there is no widely accepted software/hardware interface for these platforms - this has implications for both hardware and software development.

My group, in collaboration with others, has been working on the Instruction-Level Abstraction (ILA) as a software-hardware interface that generalizes the notion of ISAs to accelerators. The ILA model of an accelerator is a functional model that defines the response of the accelerator to commands at its interface. These commands serve as “instructions” for the accelerator. Further, we have developed the ILA-MCM model for how the operational ILA model can be integrated with an axiomatic memory consistency model for a detailed functional specification that includes accelerator-processor interactions through shared memory.

In this talk I will describe the ILA model, and its application to different use cases for accelerator-rich platforms:

* Simulation and co-simulation: Show how this model enables automated generation of executable functional models (C/C++/SystemC) that can be used in hardware simulation and hardware-software co-simulation
* Formal hardware verification: Show how the formal ILA model can be used for formal verification of the hardware implementation and hardware-software co-verification
* Memory consistency: Show how the ILA-MCM model can be used to reason about correctness of code executing across processors and accelerators for a given MCM
* Compilation to accelerators: Show how the ILA instructions can be used in a compiler flow targeting specialized accelerators - in particular a compiler flow for deep-learning accelerators using the TVM compiler framework.

(This work was supported in part by the Center for Future Architectures Research (C-FAR) and the Applications Driving Architectures (ADA) Research Center, one of the STARnet Centers and JUMP Centers co-sponsored by SRC and DARPA, the DARPA POSH program, an NSF XPS grant and a research gift from Intel Corporation.)

*Speaker Bio*

Sharad Malik is the George Van Ness Lothrop Professor of Engineering at Princeton University. He has served as the Director of the multi-university MARCO Gigascale Systems Research Center (GSRC, 2009-2012), and as the Associate Director of the Center for Future Architectures Research (C-FAR, 2013-2016). His current research focuses on design methodology for formal functional and security verification of hardware and hardware-software systems. His research in functional timing analysis and propositional satisfiability has been widely used in industrial electronic design automation tools. He has received the IEEE/ACM Design Automation Conference (DAC) Award for the most cited paper in the 50-year history of the conference (2013), the Computer-Aided Verification (CAV) Award for fundamental contributions to the development of high-performance Boolean satisfiability solvers (2009), the IEEE CEDA A. Richard Newton Technical Impact Award in Electronic Design Automation (2017), the Princeton University President’s Award for Distinguished Teaching (2009), as well as several other research and teaching awards. He has also received the UC Berkeley Electrical Engineering and Computer Science Distinguished Alumni Award (2019) and the IIT Delhi Distinguished Alumni Award (2009). He is a fellow of the IEEE and ACM.