Muralidaran Vijayaraghavan

CONTACT INFORMATION

32 Vassar Street 32-G822, Cambridge MA 02139 — Mobile: +1 408 839 3356
Homepage: http://people.csail.mit.edu/vmurali — Email: vmurali@csail.mit.edu

EDUCATION

Massachusetts Institute of Technology, Cambridge, MA
Ph.D., Electrical Engineering and Computer Science, **GPA 5.0/5.0** ***February 2016***

- Thesis topic: *Modular Verification of Hardware Systems*
- Advisors: Prof. Arvind, Prof. Adam Chlipala

S.M., Electrical Engineering and Computer Science, **GPA 5.0/5.0** ***February 2009***

- Thesis topic: *Theory of composable latency-insensitive refinements*
- Advisor: Prof. Arvind

Indian Institute of Technology, Madras, Chennai, India
B.Tech., Computer Science and Engineering, **GPA 9.53/10.0** ***June 2006***

WORK EXPERIENCE

Research Intern, IBM T.J. Watson Research Center, Yorktown Heights, NY ***Summer 2010***
Supervisor: Dr. Kattamuri Ekanadham

- Developed StructuralSpec, a high-level language for writing synchronous hardware modules with a very powerful static elaborator
- Developed a tool that converts a synchronous module into a latency-tolerant module, thus enabling modular refinements that even affect timing behaviors, without redesigning the circuit

Research Intern, VSSAD group, Intel Corporation, Hudson, MA ***Summers 2007, 2008***
Supervisor: Prof. Joel Emer

- Helped develop the *HASim* framework for designing FPGA-based simulators which have several orders of magnitude higher performance than conventional software simulators
- Helped establish a functional/timing partition for HASim, which enabled reuse of the same functional partition for different micro-architectures
- Implemented an out-of-order superscalar processor based on MIPS R10000 in HASim

INTERESTS

Programming Languages, Proof Assistants, Formal Verification, Language Specification, DSL, Computer Architecture, Hardware Synthesis

PROGRAMMING LANGUAGES

- *Functional languages:* Haskell
- *Imperative languages:* C, C++
- *Hardware description languages:* Verilog, Bluespec
- *Scripting:* Perl, Bash
- *Theorem provers:* Coq

RESEARCH EXPERIENCE

Formal Verification of Cache Coherence Protocols

Cache coherence protocols are notoriously difficult to design correctly, leading to widespread adoptation of formal verification techniques for proving their correctness. Most of the techniques used for verifying cache coherence falls under the realm of model-checking, which is known to have the problems of scalability. In my PhD dissertation, I explore using proof-assistant based techniques for verifying cache coherence protocol implementations. I *designed* a very realistic directory-based MSI cache coherence protocol that is parameterized over the shape of the hierarchy. I proved its correctness, i.e., I proved that it is indistinguishable from an atomic memory. The proof was carried

out using the Coq theorem prover. This is the first machine verification of a cache-coherence protocol parameterized over an arbitrary cache hierarchy, and is made feasible because of using theorem proving techniques as opposed to model-checking. The proof is also modular, making it possible to adapt it to other coherence protocols like MESI, MOSI, etc.

Kami: Modular Formal Verification of Hardware Designs

I am the lead developer of *Kami*, a framework for high-level specification and formal verification of hardware systems. Hardware systems are specified as a composition of several Kami modules, which communicate with each other via method calls. We formalize the semantics of Kami modules using labeled transition systems. We embed Kami and define its semantics inside the Coq theorem prover. We also developed a lot of tools inside Coq (called *tactics*) to help us prove invariants about Kami modules. The Kami language is inspired by the Bluespec hardware specification language, which is a commercial language for high-level hardware design and synthesis, and the semantics of Kami matches that of Bluespec. This is the first time the formal modular semantics of Bluespec has been specified. Kami grew organically from my thesis work of proving cache coherence protocols. While my thesis *modeled* hardware inside Coq, Kami embeds a DSL for hardware specification inside Coq, allowing us to extract hardware circuits from Kami modules. Kami modules are currently extracted into Bluespec specifications, which in turn get synthesized into hardware netlist using the Bluespec compiler. The next step is to implement a full verified compiler inside Coq to go directly from Kami modules into hardware netlists.

Formal Verification of multicore RISC-V Processor

I am involved in the formal verification of a multicore RISC-V processor. The system is parameterized over several cores, each connected to an L1 cache implementing the MSI protocol from my thesis. Specifically, I designed the memory system in Kami in this project and verified its correctness, essentially translating the Coq proof in my thesis to a proof based on Kami. Overall, the multicore system implements sequential consistency, which was formally proven inside the Kami framework. I am also designing the hardware support for machine-mode and supervisor-mode instructions of RISC-V processor and verifying its correctness with respect to the specification.

Weak Memory Models

Sequential consistency model for parallel execution of multi-threaded programs proposed by Lamport, in which each instruction in each thread executes atomically with arbitrary interleavings between threads, while simple, is not implemented in most modern processors. Instead, they implement weaker consistency models which relaxes the strict instruction ordering constraints imposed by sequential consistency. These relaxations enables designing simpler hardware while not affecting performance, and is used in all current commercial processors like Intel x86 and ARM. One of the main problems of these weaker consistency models is that their specification is always derived from implementations. This makes it hard for software engineers to check the correctness of their programs since they have to a) understand the microarchitecture completely and b) keep changing the programs everytime the microarchitecture changes. In this work, we specify a generalized weak memory model using atomic components (instantaneous execution core and atomic memory), and show how most widely used memory models (like TSO, sequential consistency, etc.) are instantiations of this weak memory model. This weak memory model can be used as a stable, future-proof specification, permitting a wide variety of microarchitectural optimizations without changing the memory model.

Hardware Synthesis from High-Level Synchronous Descriptions

I developed a high-level language, StructuralSpec, for designing synchronous modules, *i.e.* modules containing a list of actions, and each module communicates with other modules via hierarchical ports. An action can be a local state update or a transmission into an output port, while reading a register or receiving from an input port. All the actions in every module of a system execute every clock cycle. This language borrows a lot of features from Bluespec, such as implicit guards for actions and advanced static elaboration. At the same time, it deviates from Bluespec's guarded atomic action semantics, instead providing the user with a more familiar synchronous semantics, where every action takes place every clock cycle.

Automatic Synthesis of latency-tolerant designs

I developed a theory of composition of latency-tolerant modules which enables a wide range of modular refinement including changing timing characteristics. I proved that, if such a refinement obeys three properties, even if the actual timing behavior of the module changes, it can still be used to replace the original module in the setting of the whole system, without affecting the correctness of the overall system. These properties also ensure that it doesn't introduce any new deadlocks which were absent in the original system. This technique is especially useful for porting designs into FPGAs because typical ASIC designs are very expensive to implement on FPGAs, and this technique allows one to refine specific modules of the design to make it efficient on FPGAs, without having to redesign or reverify the whole system since it automatically guarantees correctness of the whole system. This is the topic of my master's thesis. This culminated in the design of StructuralSpec and I also developed a compiler to synthesize latency-tolerant designs from specifications in StructuralSpec.

Modular compilation of guarded atomic actions

I also developed an algorithm for modular compilation of Bluespec modules in the presence of module parameters, references to other methods that can be called from the current module. In current Bluespec, such a module cannot be compiled separately, and has to be inlined in the place of its instantiation. In order to enable separate compilation, some analysis about the conflict properties of the module parameters has to be done. This in turn results in a distributed scheduler, one for each module, which is a more scalable alternative to Bluespec's current scheduler. Closely associated with modular compilation is the synthesis of atomic actions spanning multiple clock cycles. I was involved in this work also which resulted in a patent.

Multicore simulator on FPGA

I took part in developing several FPGA-based multicore simulators, whose performances are orders of magnitude higher than traditional software-based multicore simulators. One of these simulators is being used as part of the HASim framework in Intel, and there we developed an out-of-order core for the Alpha ISA. The other was a simulator for an inorder PowerPC core, developed using the technique for automatic synthesis from latency-tolerant designs, and was synthesized from a high-level synchronous description using StructuralSpec.

Teaching Experience

Constructive Computer Architecture

I helped in the development of the course on Constructive Computer Architecture taught by Prof. Arvind for undergraduates. In this course, students are taught to design processors starting from a simple 1-cycle toy processor, to a complex 6-stage in-order pipelined processor with aggressive 3-level control speculation. These processors are designed in Bluespec. I helped develop both the course material, as well as the assignments and projects for this course.

Advanced Computer Architecture

I taught the course on Advanced Computer Architecture for graduate students, taught by Prof. Arvind and Prof. Joel Emer. In this course, students are taught the evolution of computer architecture and the factors influencing the design of hardware and software elements of computer systems. Topics include instruction set design, processor microarchitecture and pipelining, cache and virtual memory organizations, protection and sharing, I/O and interrupts, in-order and out-of-order superscalar architectures, VLIW machines, vector supercomputers, multithreaded architectures, symmetric multiprocessors, cache coherence, memory models and synchronization. I was involved in designing quizzes for the course and grading them.

PUBLICATIONS

1. Sizhuo Zhang, Arvind, **Muralidaran Vijayaraghavan**: *Taming Weak Memory Models*. CoRR abs/1606.05416 (2016)
2. **Muralidaran Vijayaraghavan**, Adam Chlipala, Arvind, Nirav Dave: *Modular Deductive Verification of Multiprocessor Hardware Designs*. CAV (2) 2015: 109-127
3. Xiangyao Yu, **Muralidaran Vijayaraghavan**, Srinivas Devadas: *A Proof of Correctness for the Tardis Cache Coherence Protocol*. CoRR abs/1505.06459 (2015)
4. Michal Karczmarek, Arvind, **Muralidaran Vijayaraghavan**: *A new synthesis procedure for atomic rules containing multi-cycle function blocks*. MEMOCODE 2014: 22-31
5. **Muralidaran Vijayaraghavan**, Nirav Dave, Arvind: *Modular compilation of guarded atomic actions*. MEMOCODE 2013: 177-188
6. Asif Khan, **Muralidaran Vijayaraghavan**, Silas Boyd-Wickizer, Arvind: *Fast and cycle-accurate modeling of a multicore processor*. ISPASS 2012: 178-187
7. Asif Khan, **Muralidaran Vijayaraghavan**, Arvind: *A general technique for deterministic model-cycle-level debugging*. MEMOCODE 2012: 109-118
8. Michael Pellauer, Abhinav Agarwal, Asif Khan, Man Cheuk Ng, **Muralidaran Vijayaraghavan**, Forrest Brewer, Joel S. Emer: *Design contest overview: Combined architecture for network stream categorization and intrusion detection (CANSCID)*. MEMOCODE 2010: 69-72
9. Michael Pellauer, **Muralidaran Vijayaraghavan**, Michael Adler, Arvind, Joel S. Emer: *A-Port Networks: Preserving the Timed Behavior of Synchronous Systems for Modeling on FPGAs*. TRETS 2(3): 16:1-16:26 (2009)
10. Abhinav Agarwal, Nirav Dave, Kermin Fleming, Asif Khan, Myron King, Man Cheuk Ng, **Muralidaran Vijayaraghavan**: *Implementing a fast cartesian-polar matrix interpolator*. MEMOCODE 2009: 73-76
11. **Muralidaran Vijayaraghavan**, Arvind: *Bounded Dataflow Networks and Latency-Insensitive circuits. MEMOCODE 2009*: 171-180
12. Michael Pellauer, **Muralidaran Vijayaraghavan**, Michael Adler, Arvind, Joel S. Emer: *A-Ports: an efficient abstraction for cycle-accurate performance models on FPGAs*. FPGA 2008: 87-96
13. Michael Pellauer, **Muralidaran Vijayaraghavan**, Michael Adler, Arvind, Joel S. Emer: *Quick Performance Models Quickly: Closely-Coupled Partitioned Simulation on FPGAs*. ISPASS 2008: 1-10
14. Kermin Fleming, Myron King, Man Cheuk Ng, Asif Khan, **Muralidaran Vijayaraghavan**: *High-throughput Pipelined Mergesort*. MEMOCODE 2008: 155-158
15. Man Cheuk Ng, **Muralidaran Vijayaraghavan**, Nirav Dave, Arvind, Gopal Raghavan, Jamey Hicks: *From WiFi to WiMAX: Techniques for High-Level IP Reuse across Different OFDM Protocols*. MEMOCODE 2007: 71-80
16. Nirav Dave, Kermin Fleming, Myron King, Michael Pellauer, **Muralidaran Vijayaraghavan**: *Hardware Acceleration of Matrix Multiplication on a Xilinx FPGA*. MEMOCODE 2007: 97-100

PATENTS

Michal Karczmarek, Arvind Mithal, **Muralidaran Vijayaraghavan**: *Hardware synthesis from multicycle rules*, Patent number: 8350594, Filed: November 9, 2009, Date of Patent: January 8, 2013, Assignee: MIT

Honours and Awards

- MEMOCODE design contest winner 2007, 2008, 2009, 2010
- All India Rank 15 in the Joint Entrance Examination for admission into IITs, June 2002
- Highest GPA in the Department of Computer Science at IIT Madras, June 2003