SRC Research Needs in Computer-Aided Design and Test

January 2008

CADTS Research Needs: Verification

This member-generated document describes needs and problems SRC member companies see in the verification of hardware and system designs. It is intended to point potential university investigators to the challenges members face and to encourage innovative thinking in the broad area of verification, including both formal and dynamic methods. Successful proposals in response to these needs will advance the state of the art, open new opportunities to SRC members, and provide new ways to assure error-free products. Projects are particularly encouraged which result in algorithms and implementations which are easily transferred to member companies, and which involve significant interaction between the principal investigator, students, and industrial liaisons.

2008 Verification Needs Categories	
V1	**Verification Core Technologies**
V1.1	**Bit-level solvers**
V1.2	**Word-level solvers**
V1.3	**Constraint satisfaction and dynamic validation**
V2	**RTL Verification**
V2.1	**Specification and coverage**
V2.2	**IP block verification**
V2.3	**Levels of abstraction**
V3	**System-Level Verification including SoC**
V3.1	**Hierarchical and compositional verification methods**
V3.2	**Protocol verification**
V3.3	**Verification of parameterized systems**
V3.4	**Verification or refinement across levels of abstraction**
V3.5	**System-level specification models**
V3.6	**Linking pre-silicon verification to post-silicon validation**
V4	**Microcode and Software**
V4.1	**Formal specification of instruction set architectures and memory models**
V4.2	**Verifying properties of embedded software**
V4.3	**Hardware/firmware/software co-verification**
V4.4	**Microcode verification**
V5	**Analog/Mixed Signal**
V5.1	**AMS verification**
V5.2	**Mixed signal integration**

V1: Verification Core Technologies

Advances in verification core technologies have been crucial for the practical application of formal methods to industrial designs, and significant further improvements are needed. Dynamic validation techniques continue to have a significant industrial role due to their scalability, and further improvements to these technologies are also of industrial importance. Leveraging multi-core platforms to advance verification technology can have advantages to both formal and dynamic methods. These core technologies have a variety of applications from functional verification to synthesis to equivalence checking and beyond. We identify several categories of needs:

- Advances to core bit-level solvers.

  Improvements in bit-precise reasoning have an immediate and broad impact on the practical industrial application of formal verification. Examples include:

  - Improved algorithms for model checking

- Continued improvements to both general-purpose (CNF and circuit) SAT-solvers, and SAT-solvers tuned for specific applications
- Novel and improved algorithms for optimizing design and specification logic
- Improved algorithms for automatic abstraction and abstraction-refinement
- Advances to hybrid methods which effectively combine techniques (e.g.,SAT-solvers with BDDs, simulation with symbolic simulation, theorem proving with model checking, ...)
- Improvements to symbolic trajectory evaluation and symbolic simulation algorithms
- Novel algorithms and techniques to assist in root-cause analysis and resolution of design errors

- Advances to word-level solvers.

  When available, the ability to exploit word-level and bit-vector level information from a design may yield dramatic speedups to the overall verification process.
  - Advances in techniques for satisfiability modulo theories (SMT)
  - Abstraction and abstraction-refinement methods which operate on word-level designs (e.g., predicate abstraction approaches)
  - Improvements to core word-level reasoning frameworks
- Constraint satisfaction and dynamic validation

  Modern test case generators for functional verification rely heavily on constraint satisfaction technologies; vendors and hardware manufacturers rely on state-of-the-art constraint solvers to solve the resulting constraint satisfaction problems (CSPs). Of interest are adaptations of traditional algorithms and new methods to overcome such challenges as:
  - Partitioning strategies for large and complex CSPs
  - Representation methods and operations (set, logic, arithmetic) on exponentially large variable domains
  - Generic propagators, propagation over large domains, approximation strategies and algorithms
  - Systematic search techniques such as non-chronological backtracking and no-good learning
  - Conditional CSPs, repeating (periodic) constraint networks with global constraints
  - Core improvements to simulation engines

V2: RTL Verification

Member companies are experiencing the “pain” of verification for which they seek a cure. Examples of these pains are:

- No hands on specification language for RTL designers that has a formal semantics, enables what/if reasoning, is succinct, enables design, is automatically mapable to industry standard languages and is not a complete model.
- No well defined notion of specification completeness and no effective way to measure/indicate the completeness of a given specification.
- No well defined way to specify and measure the quality of the functional coverage goals of a given RTL design.
- No effective, well defined way for measuring the correctness of a given specification for a given RTL design.
- No effective way, given a set of RTL functional coverage goals, to automatically generate a test case suite that covers these goals.
- No effective way for mapping signal level behavior to transaction behaviors.
- No effective way for doing assume-guarantee with environments that are efficiently specified using multiple abstraction layers.
- No effective way to detect deadlock in a given environment specification of the RTL design.
- No effective flow generation and coverage techniques for higher level testing.
- No effective way to verify power and performance requirements within the RTL environments.

- No efficient way for verifying a multi-parameter RTL representation.

V3: System-Level Verification including SoC

System-level (ESL) designs may be difficult to verify because of either size or heterogeneity. Heterogeneity exists, for example, in designs that contain both hardware and software. Verification methods that are highly automated are preferred in industry to those that require intensive manual effort. For research on specification methods, it is preferred if the same specification can be used in formal verification, simulation, stimulus generation, and emulation. Proposals should seek to advance the state of the art in one or more of the areas described below.

- Hierarchical and compositional verification methods
  Proposals should seek to advance the application in industry of methods for hierarchical and compositional verification, such as assume-guarantee reasoning. Assume-guarantee methods that aid both formal verification and simulation are desirable.
- Protocol verification
  Research in protocol verification includes both methods for verifying the design of new protocols and methods for checking that implementations of protocols conform to protocol standards. Multi-core protocols are an emerging area of interest.
- Verification of parameterized systems
  Many systems are difficult to verify because they contain replicated elements. These range from individual replicated cells to homogeneous and heterogeneous multi-core designs. Proposals in the area of verifying parameterized systems should seek to advance the verification of such systems in industry.
- Verification or refinement across levels of abstraction
  Industry is seeking methods that allow early versions of designs to be defined and verified at a high level of abstraction, followed by development of RTL implementations. It would be valuable to have practical methods for verifying that RTL implementations are correct with respect to system-level specifications. Other valuable approaches are to correctly refine system-level specifications into implementations and to reuse lower level stimulus at the higher level of abstractions.
- System-level specification models
  Research in this area should improve the ability of industrial groups to define abstract, system-level models with a well-defined semantics. The models should aid system-level verification and simulation tasks of the types described above.
- Linking pre-silicon verification to post-silicon validation
  A vast amount of effort is expended in post-silicon validation. Efforts in pre-silicon verification to alleviate post-silicon pain could include exploiting design modularity in simulation-based verification and in post-silicon functional verification and leveraging commonality between pre-silicon and post-silicon functional verification (including coverage and assertion checking).

V4: Microcode and Software

Software and firmware IP make up a substantial part of today's system-on-chip designs. These SoCs contain a range of programmable ingredients, such as microcontrollers, crypto, graphics, and DSP processors. The amount of software and firmware content is increasing dramatically in high-performance CPU designs as well. While hardware verification remains the major focus of the SRC, system verification will necessarily involve verification of embedded software components. Challenges include:

- Formal specification of instruction set architectures and memory models; generation of high-performance simulation/emulation models from these specifications; use of these specifications to support software development, as RTL checkers, and to generate validation tests
- Verifying properties of embedded software, such as correctness of locking protocols, pointer safety, termination, and freedom from deadlock; programming language features

for embedded software that aid specification and verification (for example, type systems and assertions)

- Co-verification of systems containing hardware and software/firmware components; stimulus generation for SoCs that incorporate embedded and end-user SW; power/performance modeling of software and combined hardware/software systems
- Microcode verification

V5: Analog/Mixed Signal Verification

Analog/mixed signal presents two challenges for verification:

- Verification of the analog-mixed-signal component

  In the past analog verification has taken one of two well understood tracks -- (a) using detailed differential equations from solid state physics or (b) very high level abstract modeling of analog elements. The latter has found limited use in industry, the former is hard to use in industrial verification practice. For realistic analog verification solutions that will address our current pain in analog verification, we need to re-focus on reasoning at the "right" level of abstraction. We need proposals that provide with abstract models that are not only able to deal with traditional CMOS-type structures, but also newer analog artifacts such as PLLs, MRAM components, Flash components, etc. We also need verification algorithms built on top of those models. Going forward, the concept of abstraction-refinement in analog verification needs to be addressed.
- Verification of AMS components when integrated in large, mainly digital SoCs

  For integration onto large SoCs, we need accurate, abstract models that will cleanly integrate into existing digital tools and flows for simulation, formal verification, coverage metrics, constraint checking and timing verification.

SRC Task Force in Verification 2008:

John Murphy	AMD	John O'Leary	Intel
Ken Albin	AMD	Robert Nguyen	Intel
Hillel Miller	Freescale	Michael Chen	Mentor
Jayanta Bhadra	Freescale	Jeremy Levitt	Mentor
Alper Sen	Freescale	Aseem Maheshwari	TI
Jason Baumgartner	IBM	Michael Thorn	TI
Steven German	IBM	William Joyner	SRC
Sharon Keidar-Barner	IBM	Dale Edwards	SRC
		David Yeh	SRC