SRC Research Needs in Computer-Aided Design and Test

Verification

2011-2012

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.

2011-2012 Verification Needs Categories	
V1	**Verification Core Technologies**
V1.1	Bit-level solvers, word-level solvers, constraint solvers, and higher-level reasoning frameworks
V1.2	Dynamic validation and hardware acceleration
V2	**RTL Verification**
V2.1	Specification and coverage
V2.2	IP block verification
V2.3	Levels of abstraction; hierarchical and compositional methods
V2.4	Debug
V2.5	Efficiency and cost
V3	**System-Level Verification including SoC**
V3.1	Protocol and parameterized verification
V3.2	Verification or refinement across levels of abstraction
V3.3	System-level specification models
V3.4	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	Software induced partitioning methods to scale co-verification
V4.5	Microcode verification
V5	**Analog/Mixed Signal**
V5.1	AMS modeling
V5.2	AMS Verification
V6	**Other Applications of Formal Methods**
V6.1	Power modeling and verification
V6.2	Formal verification of design rules against layout, circuit functionality, and performance

V1: Verification Core Technologies

Bit-level and higher-level solvers: Advances in verification core technologies have been crucial for the practical application of formal methods to industrial designs, and significant further improvements are needed. Improvements in bit-precise solvers have an immediate and broad impact on the practical industrial application of formal methods. When available, the ability to exploit word-level and higher-level information from a design may yield dramatic speedups to the overall verification process. Constraint solvers used for test case generation share many of the challenges of efficient SMT solvers. These core technologies have a variety

of applications including functional verification, equivalence checking, synthesis, automated debugging, etc. Parallelism may enhance the effectiveness both of formal and dynamic methods. We identify several categories of needs:

- Improved algorithms for model checking and invariant generation.
- 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.
- Hybrid methods which effectively combine techniques: SAT-solvers with BDDs, simulation with symbolic simulation, theorem proving with model checking, ...
- Improvements to symbolic trajectory evaluation and symbolic simulation.
- Advances in techniques for satisfiability modulo theories (SMT).
- Improved techniques for computing small finite models of quantified formulas.
- Advances in constraint solving techniques, including those for partitioning and approximation, efficient operation upon exponentially-large variable domains, efficient search techniques and propagators, conditional and repetitive constraint networks.
- Scalable techniques to generate a good diversity of solutions for constraint solving from the various core verification frameworks.
- Novel techniques to assist in root-cause analysis and resolution of design errors.

Dynamic validation and hardware acceleration: In addition to the performance and distribution quality of testcase generators, the effectiveness of dynamic validation is heavily dependent upon the speed and scalability of the underlying simulation platform. Hardware accelerators enable substantial gains in simulation throughput, though their performance and applicability is critically dependent upon efficient programming of their instruction memories. The increasing size of modern designs - approaching billions of gates - exacerbates runtime and scalability concerns. We identify several categories of needs:

- Core improvements to traditional simulation engines.
- Techniques to efficiently migrate simulation testbenches and debugging infrastructures to acceleration platforms.
- Scalable techniques for compilation onto accelerators, including netlist partitioning to minimization off-chip traffic and maximize interconnect locality, allocation of logic to processing units considering concurrency and communication cost, temporal assignment of gates to stages to mitigate critical paths and minimize routing, and tight integration between scheduling and synthesis considering structures that commonly occur with result-collection logic.

V2: RTL Verification

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

Specification and coverage

- An effective way, given a set of RTL functional coverage goals, to automatically generate a test case suite that covers these goals.
- Formal techniques to check correctness that parameterization at the IP level is not faulty.
- Methods to determine, given the functional coverage, the coverage holes in the design that can be targeted using assertions and to provide a trace to cover them
- Given an executable specification language, techniques to define assertions and coverage for verification

IP block verification

- An effective method for measuring the correctness of a given specification for a given RTL design (i.e., heuristics to determine verification quality and “completeness”).

- An effective way for mapping signal level behavior to transaction behaviors.
- An efficient way for verifying a multi-parameter RTL representation.
- Effective assertion synthesis methods.

Levels of abstraction

- An effective way for doing assume-guarantee with environments that are efficiently specified using multiple abstraction layers.
- Hierarchical and compositional verification methods that aid both formal verification and simulation.

More efficient debug and repair of assertion failures

- An effective way to generate multiple counter-examples to a set of failing assertions
- Techniques to repair failing assertions assuming a correct design and faulty assertion implementation.
- Generation of focused counterexamples for specified temporal scenarios ("show me counterexamples where request is asserted right around the time the buffer fills up")
- Automation for RTL error diagnosis
- Efficient re-verification of repaired designs

Efficiency and cost: Verification infrastructure includes tools and methodologies for test case generation, simulation, coverage evaluation, debug and bug fix. The cost of executing these components continues to grow. We are seeking novel approaches that explore commonality and redundancy (temporal and/or spatial) in complex verification processes, identify potential cost saving areas and improve overall verification efficiency. Examples may include:

- Techniques to automatically prioritize test cases at a given point of verification stage or based on a given verification objective
- Techniques to estimate simulation cycles required to achieve a certain quality level based on a given verification plan to facilitate resource allocation
- Techniques for early bug detection and to minimize the number of bug fixing iterations

These considerations apply to all areas of verification needs and are included here for convenience.

V3: System-Level Verification including SoC

System-level (ESL) designs are difficult to verify because of both size and 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.

Protocol and parameterized 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. Of particular interest:

- Methods for validating multi-core system protocols
- Parameterized methods that address systems with an arbitrary number of agents
- Methods for micro-architectural verification
- Verification of communication fabrics, with focus on liveness and non-functional properties such as quality-of-service

Verification or refinement across levels of abstraction: Industry is seeking methods to describe and verify system designs at a high level of abstraction. These system level models will co-evolve with RTL through the product design cycle and across product generations. The following are key sub-problems:

- Formal verification of system level models
- Practical methods for verifying that an RTL realization (obtained manually or via high level synthesis) is faithful to the system design.
- Re-use of verification collateral – formal properties, simulation test-benches and stimuli, etc – between system-level and RTL models
- Incremental techniques for re-verifying system-level vs RTL correspondence in the face of continuously evolving system-level and RTL models

System-level specification and modeling: Industry requires languages or language extensions that make it easier to design and implement concurrent systems at a high-level of abstraction, and that are suitable for describing and implementing both hardware and software components and their interaction. 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. Ideally, all or part of these models could be synthesized into highly efficient parallel hardware. Languages that go beyond System Verilog and SystemC are of interest.

Linking pre-silicon design and verification to post-silicon validation: The challenges of post-silicon validation come from high complexity and integration on-die, sophisticated performance/power management, an increasing amount of embedded software, and limited observability and controllability. Research directions include:

- Leveraging commonality between pre-silicon and post-silicon functional verification including coverage and assertion checking (e.g., deriving post-silicon observation hooks from assertions in pre-silicon)
- Evaluation of observation points, coverage monitor design
- Early validation content readiness: content development and content validation
- Debug automation: triage (post-Si validation issue classification), erroneous trace analysis, bug isolation and root cause analysis (especially abstracting post-silicon fails to RTL/ESL), bug fix validation
- Modular and hierarchical DFX design and validation

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 specifications
- Use of specifications to support software development, such as RTL checkers, and to generate validation tests

Verifying properties of embedded software

- Verifying properties 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)
- Verifying multithreaded, distributed, and large scale systems)
- Power/performance modeling of software and combined hardware/software systems

Hardware/firmware/software co-verification

- Co-verification of systems containing hardware and software/firmware components

- Stimulus generation for SoCs that incorporate embedded and end-user SW
- Bug classification for SoCs with embedded and end-user SW

Software induced partitioning methods to scale co-verification

- Use of SW-induced methods, such as software application scenarios, trace-partitioning, to automatically decompose and scale co-verification
- Harvesting design-knowledge/intent from SW and algorithmic models to automatically decompose system-level co-verification

Microcode verification

V5: Analog/Mixed Signal Verification

Analog/mixed signal presents two challenges for verification:

AMS Modeling

To successfully fabricate a complex mixed-signal SoC, AMS models at several levels of abstraction are needed:

- Methods to demonstrate the equivalence or correlation between the models at various levels of abstraction are needed.
- Methods to help designers create models at various levels of abstraction are needed. Attention should be paid to readability, maintainability, and particularly debugability of any automatically generated models.
- Models for analog and mixed-signal circuits that can be used on emulation and other hardware accelerators

AMS Verification

Metric driven-verification methods for AMS circuits are needed that address these questions:

- Do constraint specification languages and solvers need to be extended for specifying/solving mixed-signal constraints?
- What extensions are needed to current assertion specification languages to efficiently write mixed-signal properties?
- Can extensions to assertion languages be checked efficiently in both continuous and discrete simulation engines?
- What does an AMS functional coverage model contain and how can it be specified concisely?
- Can these techniques be applied in both simulation and formal verification?

Constraints can be used for both modeling and verification challenges. For example, a constraint set can be used to bound the equivalence checking space as well as detect inputs that exceed input boundary conditions during simulation. Methods that leverage the constraint space to solve both modeling and verification challenges are desirable.

V6: Other Applications of Formal Methods

Verification and power: Power management verification and properties in power modeling:

Power management is the biggest lever to reduce power consumption in today's SoC designs. Verification of power management algorithms within IPs, not to mention across IPs, is a complex problem. Power modeling and estimation in large scale SoC designs is another continuing challenge. Research is needed into

- Efficient verification of SoC, platform, and system level power management algorithms and the specification of such algorithms.
- Verifying the quantitative aspect of power management algorithms (i.e. PM algorithm saves/uses K milliwatts).

- Generating and using verification properties to improve power estimation; which properties are important, and how do they interact with early power estimation.
- Generation/verification of parameterized component power models
- Generation of "relevant" use-case (scenario) traces for power modeling
- Effective methods to verify power intent at the RTL level

Formal verification of design rules against layout, circuit functionality, and performance: The increase in number and complexity of design rules, and the interactions between design rules, continue to increase. Even though the rules are met, it is not guaranteed that the original circuit functionality will meet its performance metrics.

Some manufacturing schemes also apply pattern recognition to collect the largest subset of common features for mask fabrication and identify the minimum set of patterns needed by the design. Restrictions to use only these shapes need to be verified against original intended circuit function and performance.

Research is needed in this area to improve design rule formalism for IC manufacturing, improve verification of design rules against layout, and relate design rules to circuit functionality and performance using formal methods.

SRC Task Force in Verification 2011-2012

AMD	Farhan Rahman
Freescale	Jay Bhadra, Scott Little, Vivek Vedula, Shaun Feng, Brian Kahne
GLOBALFOUNDRIES	Nicholas Eib, Rasit Topaloglu
IBM	Jason Baumgartner, Steven German, Michael Moffitt
Intel	Jim Grundy, Robert Nguyen, John O'Leary, Emily Shriver, Jin Yang
Mentor Graphics	Jeremy Levitt, Christian Stangier
Texas Instruments	Mandayam Srivas, Noam Teutsch
SRC	William Joyner, David Yeh