Projectagon-Based Reachability Analysis for Circuit-Level Formal Verification

Chao Yan

The University of British Columbia
Motivation

- Circuit-Level Verification: verifying circuits using non-linear ODE models
  - Analog and mixed signal (AMS) circuits
  - Deep-submicron circuit effects undermine digital abstractions

- Analog Bugs
  - Account for large percentage of critical bugs
    - Analog design lacks systematic design flow and test methodologies
    - Analog design relies on designer intuition and expertise
  - Intel Sandy Bridge Chipset.
    - Lost: one billion $!
    - Passed all Intel’s internal tests and all of OEM tests.

- Simulations cannot find all critical bugs before fabrication.
  - Incomplete coverage: difficult to cover all corner cases.
  - Inaccurate models: ideal working conditions and input signals.

- Problem Statement

  We need formal methods to circuit-level design errors.
Contributions

This thesis demonstrates the feasibility of formally verifying behaviors for circuits modeled by non-linear, ordinary differential equations.

- Verification as Reachability
  - Methodology for specifying circuit properties
  - Table-based methods for modeling circuits
  - Circuit verification $\rightarrow$ reachability analysis

- Reachability Analysis
  - Projectagon-based reachability analysis
  - Solve nonlinear ODEs by maximum principle
  - Improvements: robustness, efficiency, accuracy, usability

- Practical Circuit Verification Examples
  - Yuan-Svensson toggle circuit,
  - A flip-flop circuit
  - An arbiter circuit
  - The Rambus ring oscillator
Circuit-Level Formal Verification

Limitations of Prior Work
- Small circuits: \textit{e.g.}, 2-3 variables
- Simplistic dynamics: \textit{e.g.}, (piecewise) linear
- Verify simple properties

Our Goal
- Real circuits (\textasciitilde 10-dim)
- Nonlinear models
- Verification need of designers
Outline

- Motivation and Contributions
- Verification as Reachability
- COHO: Reachability Analysis
- Circuit Verification Examples
- Conclusion
Formal Verification

- Goal: show Model |= Specification

- Model: nonlinear ordinary differential inclusion
  - Nonlinear
    - Real circuits are nonlinear
    - Linear dynamics can be analyzed by simulations or other methods
  - Inclusion
    - Deterministic models are approximations of reality
    - Non-deterministic behaviors: e.g., input uncertainty, PVT variation, noise

- Specification
  - Continuous extension of LTL
    - Continuous variables
    - Dense time
  - Brockett's annulus
    - Specifying analog signals
    - Providing propositions over continuous spaces
  - Probability
    - Specify analog properties
    - Metastability behaviors are common
Example: 2-input Arbiter

- A Black-Box View

Request signals: $r_1, r_2$; grant signals: $g_1, g_2$

- Mutually exclusive access
- Handshake protocol
- Liveness

Initially:

\[ \forall i \in \{1, 2\}. \neg r_i \land \neg g_i \]

Assume (environment controls $r_1$ and $r_2$):

\[ \forall i \in \{1, 2\}. \Box (r_i \implies g_i) \land \Box (\neg r_i \implies \neg g_i) \land \Box (g_i \implies \neg r_i) \]

Guarantee (arbiter controls $g_1$ and $g_2$):

Handshake:

\[ \forall i \in \{1, 2\}. \Box (\neg g_i \implies r_i) \land \Box (g_i \implies \neg r_i) \]

Mutual Exclusion:

\[ \Box \neg (g_1 \land g_2) \]

Liveness:

\[ \forall i \in \{1, 2\}. (\Box (r_i \implies g_i)) \land (\Box (\neg r_i \implies \neg g_i)) \]
Specification

- Continuous LTL
  - Continuous variables & trajectories
  - Dense time
    - $p$: $p$ holds for the current time.
    - $\square p$: $p$ holds for this and all subsequent time.
    - $p U q$: $p$ holds until a time for which $q$ holds.
    - $p W q$: $p$ holds forever or until a time for which $q$ holds.

- Brockett’s Annulus

- Continuous Specification
Brockett's annulus allows entire families of signals to be specified.

Region 1 represents a logical low signal.

Region 2 represents a monotonically rising signal.

Region 3 represents a logical high signal.

Region 4 represents a monotonically falling signal.

Map continuous trajectories to discrete sequences.
Specification

- Continuous LTL
- Brockett’s Annulus
- Continuous Specification

Initially:
\[ \forall i \in \{1, 2\}. B_1(r_i) \land B_1(g_i) \]

Assume (environment controls \( r_1 \) and \( r_2 \)):
\[ \forall i \in \{1, 2\}. \Box (B_3(r_i) \Rightarrow B_{2,3}(g_i)) \land \Box (B_1(r_i) \Rightarrow B_{4,1}(g_i)) \land \\
\Box (B_3(g_i) \Rightarrow B_{4,1}(r_i)) \]

Guarantee (arbiter controls \( g_1 \) and \( g_2 \)):

Handshake:
\[ \forall i \in \{1, 2\}. \Box (B_1(g_i) \Rightarrow B_{2,3}(r_i)) \land \Box (B_3(g_i) \Rightarrow B_{4,1}(r_i)) \]

Mutual Exclusion:
\[ \Box \neg (B_{2,3}(g_1) \land B_{2,3}(g_2)) \]

Liveness:
\[ \forall i \in \{1, 2\}. (\Box (B_3(r_i) \Rightarrow B_{2,3}(g_i)) \land (\Box (B_1(r_i) \Rightarrow B_{4,1}(g_i)))) \]
Liveness and Probability

- **Metastability**
  - There must exist trajectories where contested request are never granted
  - Liveness is not “always” satisfied
  - No ideal arbiter

- **Almost Surely**
  - Liveness is not always satisfied, but it may be satisfied with probability 1.
  - Solution: introduce probability theory to logic
  - *Almost-surely* version of LTL “always” operator
    - A trajectory $\phi$ satisfies $\square Z S$ iff $S$ holds everywhere along $\phi$, or if $\phi$ is in a negligible set, $Z$.
    - The probability of $S$ holding everywhere along $\phi$ is equal to 1.
    - $Z$ is the same for all trajectories.

- **Continuous Specification**
  \[
  \forall i \in \{1, 2\}. \quad \alpha\text{-ins} \Rightarrow (\square Z (B_3(r_i) \cup B_{2,3}(g_i)))
  \land (B_3(r_i) \cup (B_{2,3}(g_i) \lor B_3(r_{\sim i}))) \land (\square (B_1(r_i) \cup B_4(g_i)))
  \]
Outline

- Motivation and Contributions
- Verification as Reachability
- СОНО: Reachability Analysis
- Circuit Verification Examples
- Conclusion
Representing and manipulating high-dimensional regions: projectagons

Projectagons are efficiently manipulated using two-dimensional computational geometry algorithms.

COHO projects high-dimensional polyhedra onto two-dimensional subspaces.

COHO uses both geometrical and inequality representations.

Projectagon faces correspond to edges of the projection polygons.

Solving dynamic systems: linear differential inclusions.

Basic step of COHO
**COHO: Basic Ideas and Algorithms**

- Representing and manipulating high-dimensional regions: **projectagons**
- Solving dynamic systems: **linear differential inclusions**.
  - Approximate the ODEs by linear differential inclusions.
    - least squares method
    - linear programming
  - Compute forward reachable region using the **Maximum Principle**
  - Efficient: matrix computations
  - Accurate: work on each face rather than the whole projectagon.

- Basic step of **COHO**
**COHO: Basic Ideas and Algorithms**

- Representing and manipulating high-dimensional regions: **projectagons**
- Solving dynamic systems: **linear differential inclusions**.
- Basic step of COHO

A bounding projectagon is obtained by moving each face forward in time.

The advanced face is projected onto two-dimensional subspaces to maintain the structure of projectagon.

Computation continues until no new reachable region found.
**COHO: Basic Ideas and Algorithms**

- Representing and manipulating high-dimensional regions: *projectagons*
- Solving dynamic systems: *linear differential inclusions.*
- Basic step of COHO
- All approximations over-approximate the reachable space: COHO is sound.
- Available at http://coho.sourceforge.net
**Coho: Improvements**

- Robustness
  - Arbitrary precision rational (APR)
    - ill-conditioned problems
    - simple implementation
  - Remove infeasible regions
    - incomplete boundary
    - guarantee non-empty projectagon faces
- Efficiency
- Accuracy
- Usability
**COHO: Improvements**

- **Robustness**
- **Efficiency**
  - Guess-verify strategy
    - step-size is much smaller than necessary
    - guess a larger step-size based on previous step
    - verify the step-size and model is correct
  - Approximation algorithms
    - linear programming
    - projection algorithm
  - Hybrid computation
    - APR is expensive
    - floating point, interval, APR
- **Accuracy**
- **Usability**
COHO: Improvements

- Robustness
- Efficiency
- Accuracy
  - Interval closure
    - over-approximation by using convex hull
    - non-convex polygons are constraints of variables
    - interval propagation
  - Reduce model error
    - asymmetric bloating
    - anisotropic bloating
    - multiple models
  - Reduce projection error
- Usability
**CoHO: Improvements**

- **Robustness**
- **Efficiency**
- **Accuracy**
- **Usability**
  - Interface based on hybrid automata
  - Templates for reachability computations
  - Options for trade-off of performance and accuracy
Infeasible Regions

- Infeasible regions
  - Projection polygons are computed independently
  - Infeasible regions: the prism from this region does not intersect with other prisms
  - Leads to incomplete boundaries in the next step

- Clipping infeasible regions
  - The problem of determining if a projectagon is non-empty is NP-complete
  - Approximation techniques must be applied
  - Make a projectagon feasible to its convex hull
Infeasible Regions

Algorithm
- Construct the convex hull of the projectagon
- Project onto all planes to obtain an updated convex hull
- Compute the intersection of the projectagon and the updated convex hull
- Repeat until the result converges

Projectagon Faces
- Use interval closure to find an accurate projectagon face if it is feasible

\[ \text{prism}(e) \cap \text{convexhull}(P) \cap \text{intervalClosure}(e, P) \]

- Use convex hull to find an over-approximated projectagon face otherwise

\[ \text{prism}(e) \cap \text{convexhull}(P) \]
Hybrid Computation

- Arbitrary precision rational numbers (APR)
  - Expensive
  - Only necessary for ill-conditioned problems
- Hybrid computation
  - Use double-precision arithmetic for general computation
  - Use interval computation to validate the result
  - Use APR to repeat the computation if failed
- Applications
  - Linear programming
  - Geometric computations
  - Projection algorithms
Outline

- Motivation and Contributions
- Verification as Reachability
- \( \text{CNO} \): Reachability Analysis
- Circuit Verification Examples
- Conclusion
Example: Arbiter

Based on cross-couple NANDs

The *metastability filters* ensure that no grant is asserted until metastability has resolved.
Reachability Computation

- **Stiffness**
  - Vastly different node capacitances: $z_1, z_2$
  - Ill-conditioned Jacobian matrix for ODE $\dot{x} = f(x)$
  - Difficult to find a good time-step
    - Large: large model error
    - Small: large projection and simplification errors

- **Solution**
  - Changing variables: converge much more rapidly
  - Additional invariants: reduce over-approximation

- **Performance**
  - 6-dimensional non-convex regions
  - $\sim 90$ hours
    - large number of steps
    - circuit modeling, projection, linear programming
  - $1 \sim 2G$ memory
    - large tables for circuit models
Results

- Safety Properties
  - Mutual Exclusion
    - Handshake Protocol
    - Brockett Annuli

- Liveness Properties

Mutual Exclusion
Results

- Safety Properties
  - Mutual Exclusion
  - Handshake Protocol
  - Brockett Annuli

- Liveness Properties

Ph.D. Oral Examination  30\textsuperscript{th} August, 2011  Reachability Analysis for Circuit-Level Formal Verification – p.18/22
Results

- Safety Properties
  - Mutual Exclusion
  - Handshake Protocol
  - Brockett Annuli

\[ \dot{x} \text{ vs. } x \]

\[ \dot{y} \text{ vs. } y \]

Brockett Annuli

- Liveness Properties
Results

- Safety Properties
  - Mutual Exclusion
  - Handshake Protocol
  - Brockett Annuli

- Liveness Properties
  - Initialization: stable within 200ps
  - Uncontested Requests: grant the client within 350ps
  - Contested Requests: metastability within hyper-rectangle
  - Reset: withdraw grants within 270ps
  - Fairness: grant the other client within 420ps
Metastability and Liveness

- **Metastable Behaviours**
  - Fail to show a client is eventually granted when both requests asserted concurrently
  - Trajectories cannot leave the metastable region $M$ because of over-approximation
  - Cannot be solved by reachability analysis

- **Almost-Surely Verification**
  - Bound the metastable region $M$ by COHO
  - Compute interval Jacobian matrix in $M$
  - Show divergence holds everywhere in $M$ using dynamical system theory [Mitchell96]
Other Circuits

- The Yuan-Svensson Toggle Circuit
  - Revealed a leakage current bug
  - The period of output is twice that of the input
  - Verify that the output and input signals satisfy the same Brockett’s annulus

- A Flip-Flop Circuit
  - Showed the output satisfies the specification
  - Modeled circuit with multiple inputs with timing constraints

- The Rambus Ring Oscillator
  - Real problem from industry
  - Space reduction to improve performance of reachability analysis
  - Combine reachability analysis with other methods (e.g., static analysis) to solve practical problems
  - Find sufficient conditions that guarantee oscillation from all initial conditions except for a set of measure zero.
Conclusions

- Circuit Verification as Reachability Analysis
  - A systematic way of translating verification problems to reachability analysis problems
    - Modeling: modified nodal analysis, table-based models from simulations.
    - Specification: Brockett annulus, LTL and probability theory
  - Efficient reachability analysis
    - Projectagon-based reachability analysis
    - Improvements: robustness, efficiency, accuracy, etc.

- Correctness and Efficiency Demonstrated by Circuit Verification Examples
  - Synchronous: Flip-flop, toggle
  - Asynchronous: Arbiter
  - Analog: Rambus ring oscillator

- Verification of Practical Circuits
  - Stiffness: challenges of reachability analysis
  - Metastability: cannot be solved by reachability analysis alone

- Analog Properties of Practical Circuits Can be Formally Verified Based on Reachability Analysis.
Future Work

- Verify Larger, Practical AMS Circuits
  - Parameterized verification
  - Point verification: specialized tools for common circuit classes

- Combine Reachability Analysis with Other Methods
  - Small-signal analysis
  - Static invariant computations: HYSAT, HSOLVER
  - Almost-surely verification

- Specification
  - Check properties automatically
  - Integrate with other verification tools

- Improve Performance of COHO
  - Parallel computation
  - More efficient approximation algorithms

- More applications
  - Apply to other hybrid system problems
  - Compare with other reachability analysis tools