Formal Verification of an Arbiter

Chao Yan & Mark Greenstreet & Jochen Eisinger

The University of British Columbia
Outline

- Circuit-Level Verification
- Arbiter
  - Specification
  - Previous work
  - An arbiter circuit
- Verification as Reachability
  - Coho
  - Stiffness
- Almost-Surely Verification
- Conclusion and Future Work
Circuit-Level Verification

- Digital circuit verification
  - Show that a circuit in an analog-model implements the desired discrete behavior.

- Why do Circuit-Level Verification?
  - Digital design has become relatively low error:
  - Circuit-level bugs remain a problem:
    - SPICE is still the main validation tool, and it doesn’t scale.
    - Deep-submicron circuit effects undermine digital abstractions.
    - Hard/impossible to simulate bugs.

- Example
  - Synchronizer $\tau$ doesn’t scale with $FO4$.
  - Deep metastability beyond resolution of HSPICE.
  - How can we make sure our designs don’t have such “scaling bugs”?
Arbiters

● A Black-Box View

Previous Work

  ● Assumed inputs make instantaneous transitions.
  ● This assumption greatly reduces the size of the reachable space.

● No perfect arbiter can be built

Challenges

● Formal specification
● Realistic device and input models
● Stiffness
● Metastable behaviour
Discrete Specification

- LTL logic

  - \( p \): \( p \) holds in the current state.
  - \( \square p \): \( p \) holds this and all subsequent states.
  - \( p \overset{U}\rightarrow q \): if \( p \) holds in the current state, \( p \) will continue to hold until a state in which \( q \) holds.
  - \( p \overset{W}\rightarrow q \): if \( p \) holds in the current state, \( p \) will continue to hold forever or until a state in which \( q \) holds.

- Specification
Discrete Specification

- LTL logic
- Specification

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

Assume (environment controls \(r_1\) and \(r_2\)):
\[ \forall i \in \{1, 2\}. \quad \square (r_i \mathcal{W} g_i) \land \square (\neg r_i \mathcal{W} \neg g_i) \land \square (g_i \mathcal{U} \neg r_i) \]

\(r\) holds until \(g \uparrow\)  
\(g \uparrow \rightarrow r \downarrow\)  
\(r\) holds until \(g \downarrow\)

ASYNC10 — May 6, 2010 — Formal Verification of an Arbiter – p.5/17
Discrete Specification

- LTL logic
- Specification

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

Assume (environment controls \( r_1 \) and \( r_2 \)):
\[ \forall i \in \{1, 2\}. \square (r_i \bigwedge g_i) \land \square (\neg r_i \bigwedge \neg g_i) \land \square (g_i \bigcup \neg r_i) \]

Guarantee (arbiter controls \( g_1 \) and \( g_2 \)):
- Handshake:
  \[ \forall i \in \{1, 2\}. \square (\neg g_i \bigwedge r_i) \land \square (g_i \bigwedge \neg r_i) \]
- Mutual Exclusion:
  \[ \square \neg (g_1 \land g_2) \]
- Liveness:
  \[ \forall i \in \{1, 2\}. (\square (r_i \bigcup g_i)) \land (\square (\neg r_i \bigcup \neg g_i)) \]
Continuous-Time Logic

- Brockett’s Annulus

Map continuous trajectories to discrete sequences.
- 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.
- Brockett’s annulus allows entire families of signals to be specified.
Continuous-Time Logic

- Brockett’s Annulus

- Map continuous trajectories to discrete sequences.
- 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.
- Brockett’s annulus allows entire families of signals to be specified.
Continuous-Time Logic

- Brockett’s Annulus

- Map continuous trajectories to discrete sequences.
- 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.
- Brockett’s annulus allows entire families of signals to be specified.
Continuous-Time Logic

- Brockett’s Annulus

- Map continuous trajectories to discrete sequences.
- 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.
- Brockett’s annulus allows entire families of signals to be specified.
Continuous-Time Logic

- Brockett’s Annulus

- Map continuous trajectories to discrete sequences.
- 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.
- Brockett’s annulus allows entire families of signals to be specified.
Continuous-Time Logic

- Brockett’s Annulus

- Map continuous trajectories to discrete sequences.
- 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.

→ Brockett’s annulus allows entire families of signals to be specified.
Continuous-Time Logic

- Brockett’s Annulus

Initially:

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

\[ \forall i \in \{1, 2\}. B_1(r_i) \land B_1(g_i) \]
Continuous-Time Logic

- Brockett’s Annulus

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

Assume (environment controls \( r_1 \) and \( r_2 \)):
\[ \forall i \in \{1, 2\}. \Box (r_i \overline{\Box} g_i) \land \Box (\neg r_i \overline{\Box} \neg g_i) \]
\[ \land \Box (g_i \overline{\Box} \neg r_i) \]
\[ \forall i \in \{1, 2\}. \Box (B_3(r_i) \overline{\Box} B_2,3(g_i)) \land \Box (B_1(r_i) \overline{\Box} B_{4,1}(g_i)) \]
\[ \land \Box (B_3(g_i) \overline{\Box} B_4(r_i)) \]
Continuous-Time Logic

- Brockett’s Annulus

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

Assume (environment controls \( r_1 \) and \( r_2 \)):
\[ \forall i \in \{1, 2\}. \ \Box (B_3(r_i) \bigwedge B_2,3(g_i)) \land \Box (B_1(r_i) \bigwedge B_4,1(g_i)) \land \Box (B_3(g_i) \bigcup B_4(r_i)) \]

Guarantee (arbiter controls \( g_1 \) and \( g_2 \)):
- Handshake:
  \[ \forall i \in \{1, 2\}. \ \Box (B_1(g_i) \bigwedge B_2,3(r_i)) \land \Box (B_3(g_i) \bigwedge B_4,1(r_i)) \]
- Mutual Exclusion:
  \[ \Box \neg (B_2,3(g_1) \land B_2,3(g_2)) \]
- Liveness:
  \[ \forall i \in \{1, 2\}. \ (\Box (B_3(r_i) \bigcup B_2,3(g_i))) \land (\Box (B_1(r_i) \bigcup B_4,1(g_i))) \]
Continuous-Time Logic

- Brockett’s Annulus
- Specify Metastable Behaviours
  - *Almost-surely* version of LTL “always” operator
    - $\phi \models \Box_Z S \equiv (\phi \models (\Box S) \lor ((\phi \in Z) \land (\mu(Z) = 0)))$
    - A trajectory $\phi$ satisfies $\Box_Z S$ 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.
  - $\alpha$ – insensitivity
    - arbiter’s clients do not act as feedback controllers
    - $\alpha$-ins $\Rightarrow (\Box_Z (B_3(r_i) \bigcup B_{2,3}(g_i)))$
Continuous Specification

Initially:

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

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

\[ \forall i \in \{1, 2\}. \ \Box (B_3(r_i) \rightarrow B_2,3(g_i)) \land \Box (B_1(r_i) \rightarrow B_4,1(g_i)) \land \Box (B_3(g_i) \rightarrow B_4(r_i)) \]

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:

\[ \forall i \in \{1, 2\}. \]

\[ \alpha\text{-ins} \Rightarrow (\Box Z(B_3(r_i) \rightarrow B_2,3(g_i))) \land (B_3(r_i) \rightarrow B_2,3(g_i) \lor B_3(r_{\sim i})) \land (\Box (B_1(r_i) \rightarrow B_4,1(g_i))) \]
Based on a SR-latch

The \textit{metastability filters} ensure that no grant is asserted until metastability has resolved.
Verification as Reachability

- Phase-space view of circuit behaviour
  - Waveforms $\rightarrow$ phase-space (reachable regions)
  - An inverter example

- Formal verification by reachability analysis
  - Specify continuous signals by Brockett annuli
  - Compute the reachable region that contains all reachable trajectories
  - Verify each signal satisfies its Brockett annulus
  - Verify LTL properties are satisfied in the reachable region
Coho: Reachability Computation Tool

- Solving dynamic systems: linear differential inclusions.
  - Inclusions computed for neighborhood of a region.
  - Each inclusion is of the form: $\dot{v} = Av + b \pm u$, where $u$ is an error term.

- Representing and manipulating high dimensional space: projectagon
  - Provides a tractable representation.
  - Exploits extensive algorithms for 2D computational geometry.

- All approximations overapproximate the reachable space:
  - Coho is sound for verifying safety properties.
  - False negatives are possible.

http://pond.dnr.cornell.edu/nyfish/Salmonidae/coho_salmon.jpg
Circuit Model

- Transistors modeled as voltage controlled current sources.

- The $I_{ds}$ function is obtained by tabulated data from HSPICE simulations.

- At each time step, and for each projection polygon edge, Coho:
  - computes a bounding box for node voltages of each transistor.
  - computes a model of the form $i_1 = A_1 v + b_1 \pm u_1$ where $u_1$ is an error bound. Likewise for $i_2$.
  - bounds $i_c = (A_2 + A_2) v + (b_1 + b_2) \pm (u_1 + u_2)$.
    This produces a worst-case error bound.

- Approximate the ODEs by linear differential inclusions:

$$\begin{bmatrix} v \\ \text{in} \end{bmatrix} + b - u \leq \dot{v} \leq \begin{bmatrix} v \\ \text{in} \end{bmatrix} + b + u$$
Projectagons

- Coho projects high dimensional polyhedron onto two-dimensional subspaces.
- Projectagons are efficiently manipulated using two-dimensional geometry computation algorithms.
- Projectagon faces correspond to projection polygon edges.
- 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.
Stiffness

Why stiffness is a problem for Coho

Solution

Additional invariants
Result

- **Mutual Exclusion**
- **Brockett Annuli**

\[ \text{a. } g_1 \text{ vs. } g_2 \text{ (including } B_4) \quad \text{b. } g_1 \text{ vs. } g_2 \text{ (excluding } B_4) \]
Result

- Mutual Exclusion
- Brockett Annuli
  - Metastability filters work as Brockett's Annulus transformers

\[ \begin{align*}
  \dot{x}_1 & \text{ vs. } x_1 \\
  \dot{\bar{g}}_1 & \text{ vs. } \bar{g}_1
\end{align*} \]
**Liveness**

- Metastable behaviours
  - Both requests are asserted concurrently
  - Fail to show a client is eventually granted

- Bound the metastable region by Coho
Liveness

- Metastable behaviours
- Bound the metastable region by Coho
  - Stay in region $M$ forever because of over-approximation.
  - How to show region $M$ is exited with probability one?
Almost-surely Approach

- **Intuition**
  - The distance of any two points on a line increases
  - There is at most one point stay in the metastable region
  - The set of trapped trajectories is of maximum dimension $d - 1$

- **Double Cone**
Almost-surely Approach

- Intuition
  - The distance of any two points on a line increases
  - There is at most one point stay in the metastable region
  - The set of trapped trajectories is of maximum dimension $d - 1$

- Double Cone
Almost-surely Approach

- Intuition
  - The distance of any two points on a line increases
  - There is at most one point stay in the metastable region
  - The set of trapped trajectories is of maximum dimension $d - 1$

- Double Cone
Almost-surely Approach

- **Intuition**
  - The distance of any two points on a line increases
  - There is at most one point stay in the metastable region
  - The set of trapped trajectories is of maximum dimension $d - 1$

- **Double Cone**
Almost-surely Approach

- Intuition
  - The distance of any two points on a line increases
  - There is at most one point stay in the metastable region
  - The set of trapped trajectories is of maximum dimension $d - 1$

- Double Cone
Almost-surely Approach

- Intuition
- Double Cone
  - Construct a double cone with vertex on a trajectory
  - Trajectories on boundaries flow inward
  - Trajectories in the cone diverge on the axis direction

Sufficient condition
Almost-surely Approach

- Intuition
- Double Cone
- Sufficient condition
  - diverge property relate to the Jacobian matrix
  - details formulated in paper
- Almost Surely Verification
Almost-surely Approach

- Intuition
- Double Cone
- Sufficient condition
- Almost Surely Verification
  - use Coho to find metastable region
  - compute Jacobian matrix
  - use interval arithmetic to show divergence holds everywhere in metastable region
Conclusion and Future Work

Conclusion
- Specification
- Solutions to stiffness problem
- Almost-surely verification of metastable region

Future Work
- General solution for stiff system and metastable behaviours
  - An oscillator start-up verification problem
- Formal specification for more circuits
- Combine static (symbolic) techniques with reachability computation
Conclusion and Future Work

- Conclusion
  - Specification
  - Solutions to stiffness problem
  - Almost-surely verification of metastable region

- Future Work
  - General solution for stiff system and metastable behaviours
    - An oscillator start-up verification problem
  - Formal specification for more circuits
  - Combine static (symbolic) techniques with reachability computation

- Questions?

Thank You!