Verifying an Arbiter Circuit

Chao Yan & Mark Greenstreet

University of British Columbia

Outline

- Circuit-Level Verification
  - Why verify at the circuit level?
  - Why verify an arbiter?
- Coho
  - Overview
  - Enhancements
- Verifying the Arbiter Circuit
  - Arbiter Circuit
  - Properties Verified
- Conclusion and Future Work

Circuit-Level Verification

- What is circuit-level verification?
  - Analog circuit verification: verify stability, gain, noise-figure, etc.
  - Mixed-signal circuit verifcation: verify interactions between analog and digital circuits.
  - Digital circuit verification: Show that a circuit in an analog-model implements the desired discrete behavior.
- Properties that we verify:
  - Show correct operation for all valid input waveforms.
  - Use real, industry standard circuit and device models.
- Properties that we would like to verify:
  - Show correct operation for all process parameters.
  - Include crosstalk, power supply noise, etc. in our circuit models.

Why do Circuit-Level Verification?

- Digital design has become relatively low error:
  - Systematic design flows.
  - Lots of simulation.
  - Equivalence checking.
  - Model checking.
- 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.
- Properties that we would like to verify:

An Arbiter Circuit

- Specification
- Why Verify an Arbiter?
  - Exercise in modeling concurrent events from the environment: requests can be asserted concurrently.
  - Requires handling a non-trivial circuit behavior: metastability.

Overview

- Circuit-Level Verification
  - Why verify at the circuit level?
  - Why verify an arbiter?
- Coho
  - Overview
  - Enhancements
- Verifying the Arbiter Circuit
  - Arbiter Circuit
  - Properties Verified
- Conclusion and Future Work

Circuit-Level Verification

- What is circuit-level verification?
  - Analog circuit verification: verify stability, gain, noise-figure, etc.
  - Mixed-signal circuit verifcation: verify interactions between analog and digital circuits.
  - Digital circuit verification: Show that a circuit in an analog-model implements the desired discrete behavior.
- Properties that we verify:
  - Show correct operation for all valid input waveforms.
  - Use real, industry standard circuit and device models.
- Properties that we would like to verify:
  - Show correct operation for all process parameters.
  - Include crosstalk, power supply noise, etc. in our circuit models.

Why do Circuit-Level Verification?

- Digital design has become relatively low error:
  - Systematic design flows.
  - Lots of simulation.
  - Equivalence checking.
  - Model checking.
- 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.
- Properties that we would like to verify:

Verification Flow

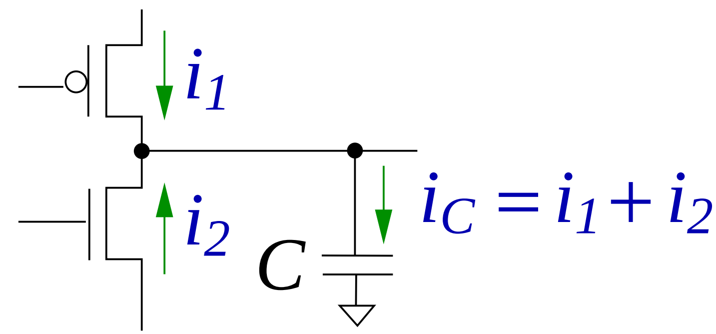
- Model the system by *non-linear ordinary differential equations* (ODEs)

$$\dot{x} = f(x, in)$$

- Compute all possible states of the system
  - Projection based representation of high dimensional space
  - Forward reachable space
- Check the specification over all states

Circuit Model

- Model MOS circuits as a collection of voltage controlled current sources
- Current function is obtained by simulating TSMC 180nm, 1.8 volt, bulk CMOS process
- Time derivative of voltage



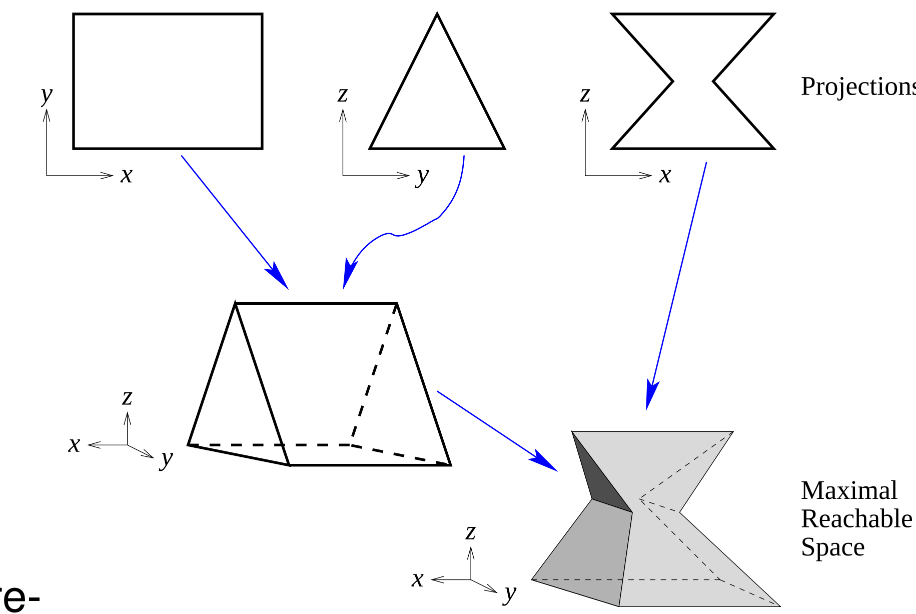
$$\dot{V} \quad = \quad C^{-1} \cdot I_c$$

- Approximate the ODEs by *linear differential inclusions*:

$$A \begin{bmatrix} x \\ in \end{bmatrix} + b - err \quad \leq \quad \dot{x} \quad \leq \quad A \begin{bmatrix} x \\ in \end{bmatrix} + b + err$$

Representing the Reachable Space

- Coho: Projectagons
  - Project high dimensional polyhedron onto two-dimensional subspaces.
  - A projecatgon is the intersection of a collection of prisms, back-projected from projection polygons.
  - Projectagons are efficiently manipulated using two-dimensional geometry computation algorithms.
  - Each edge of the polygon corresponds to a face of the projectagon.

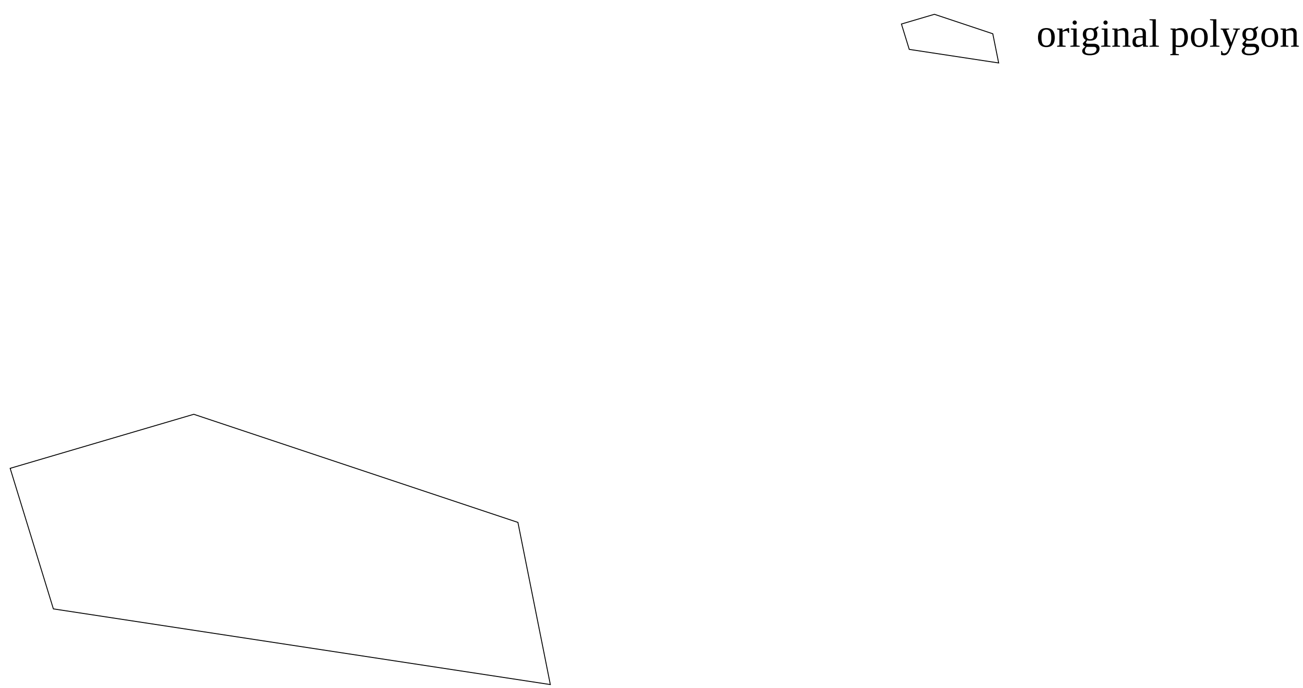


Reachability for Projectagons

- Given a projectagon and model, what is the reachable space after ΔT?
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.

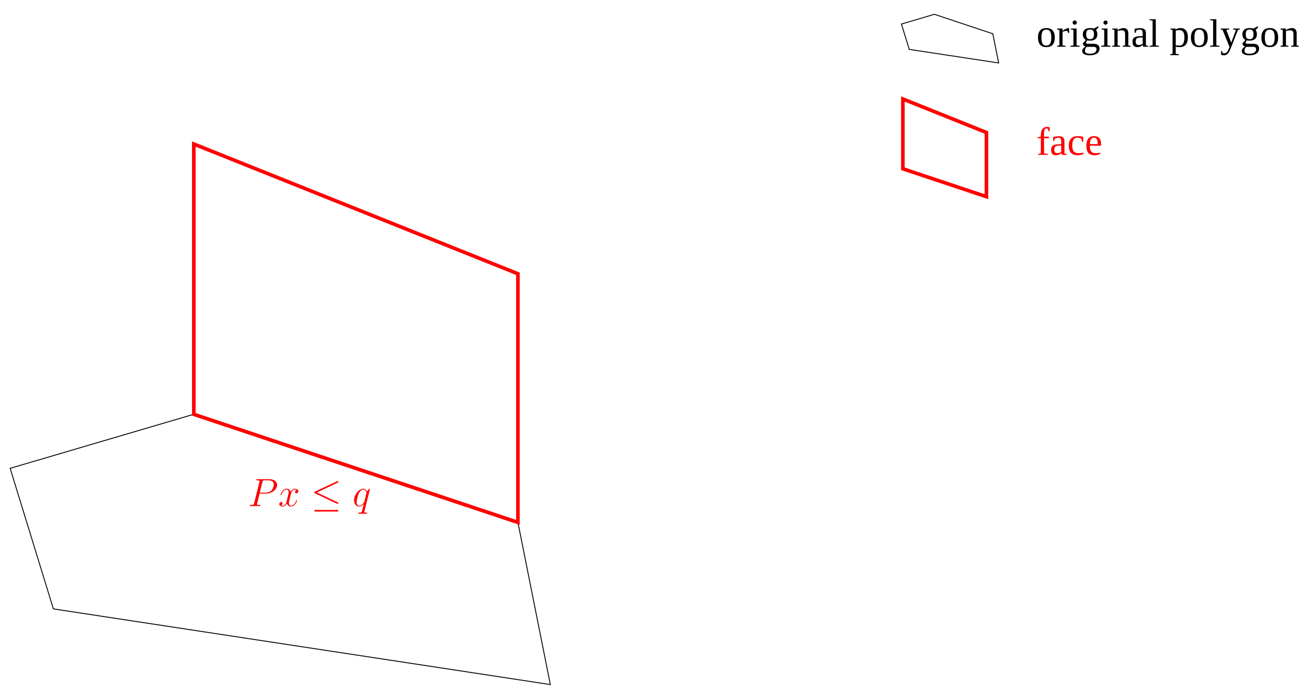
Reachability for Projectagons

- Given a projectagon and model, what is the reachable space after ΔT?
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.



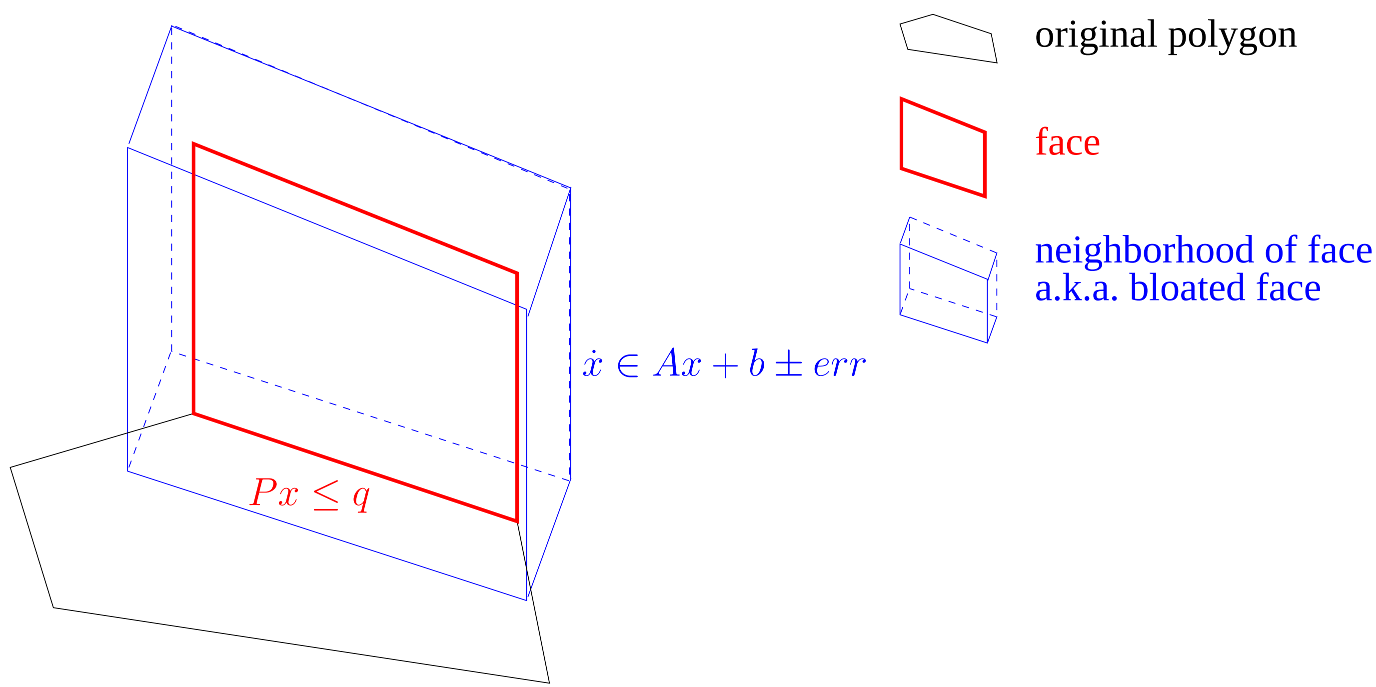
Reachability for Projectagons

- Given a projectagon and model, what is the reachable space after ΔT?
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.



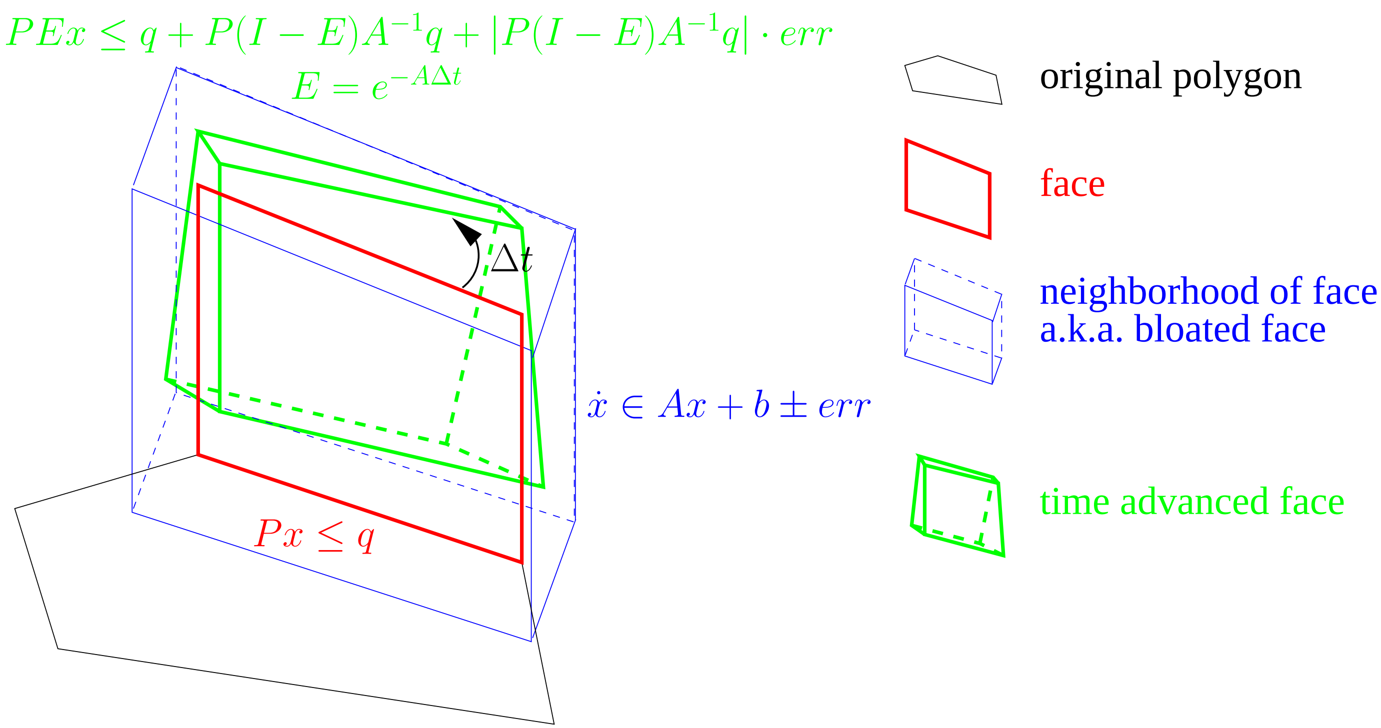
Reachability for Projectagons

- Given a projectagon and model, what is the reachable space after ΔT?
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.



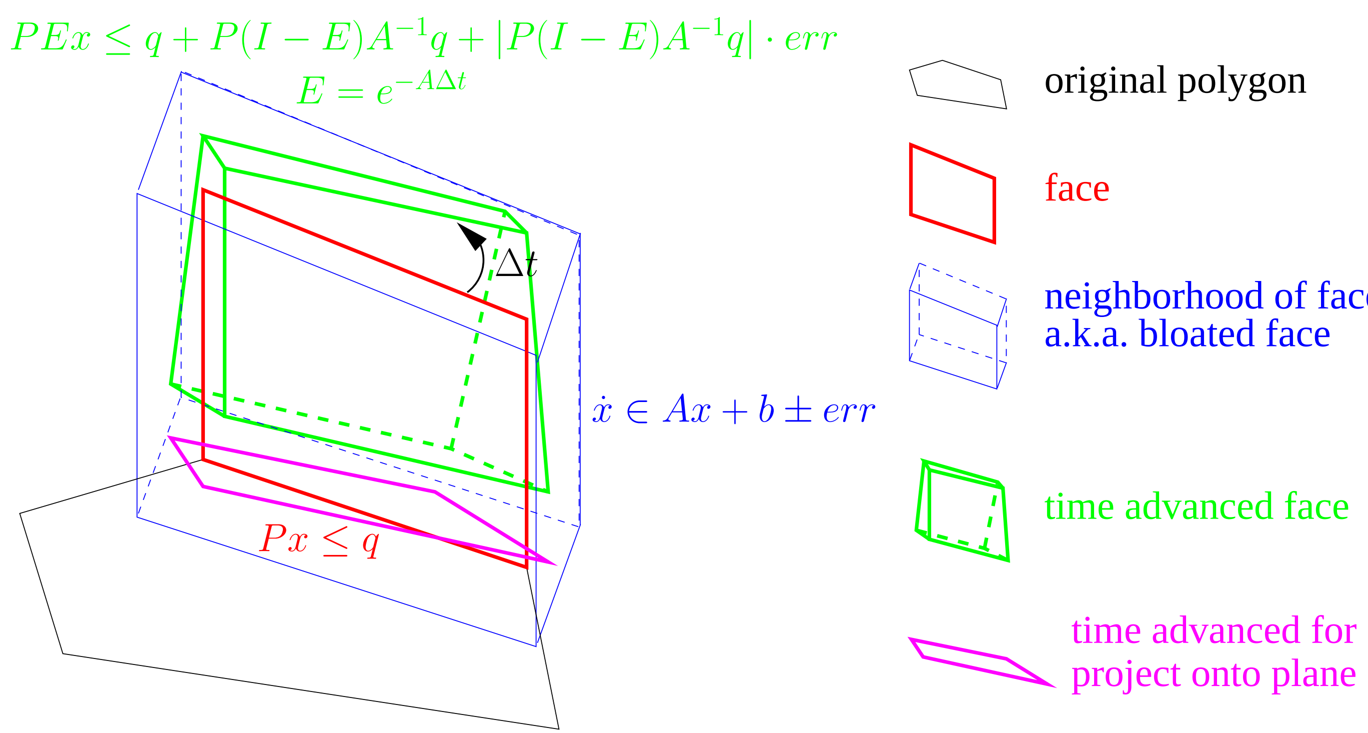
Reachability for Projectagons

- Given a projectagon and model, what is the reachable space after ΔT?
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.

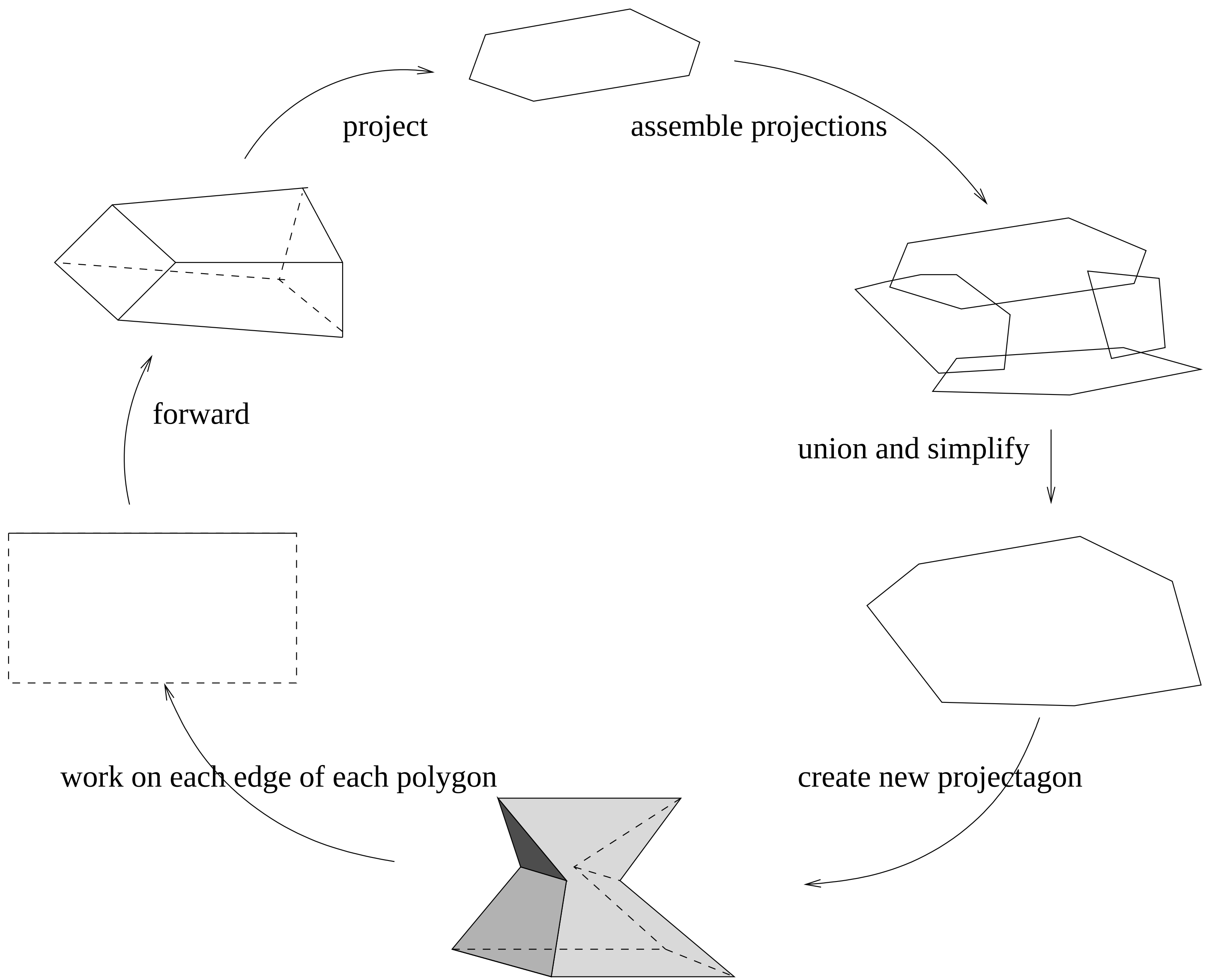


Reachability for Projectagons

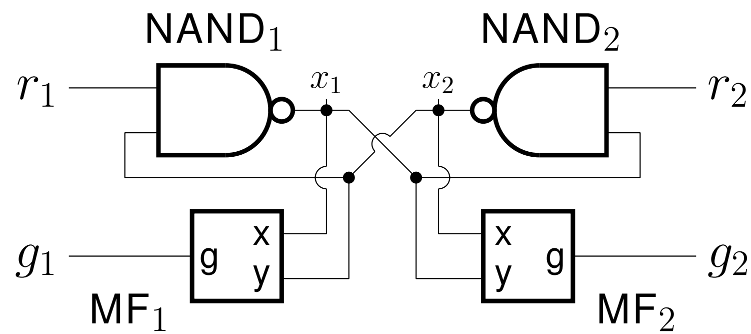
- Given a projectagon and model, what is the reachable space after ΔT?
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.



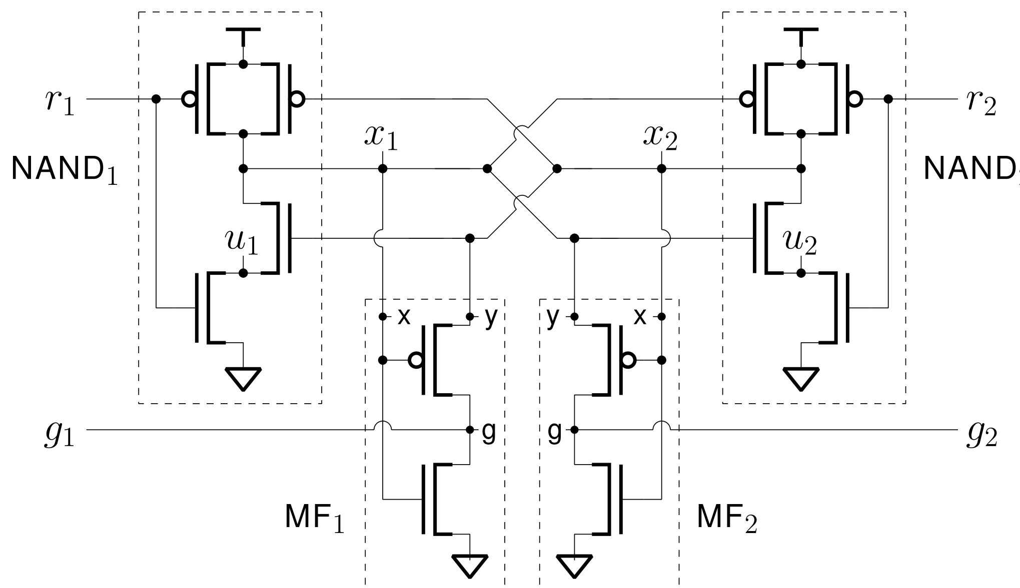
Basic Step of Coho



Arbiter Circuit



Gate-level schematic of the arbiter circuit.



Transistor-level schematic of the arbiter circuit.

Coho Improvements

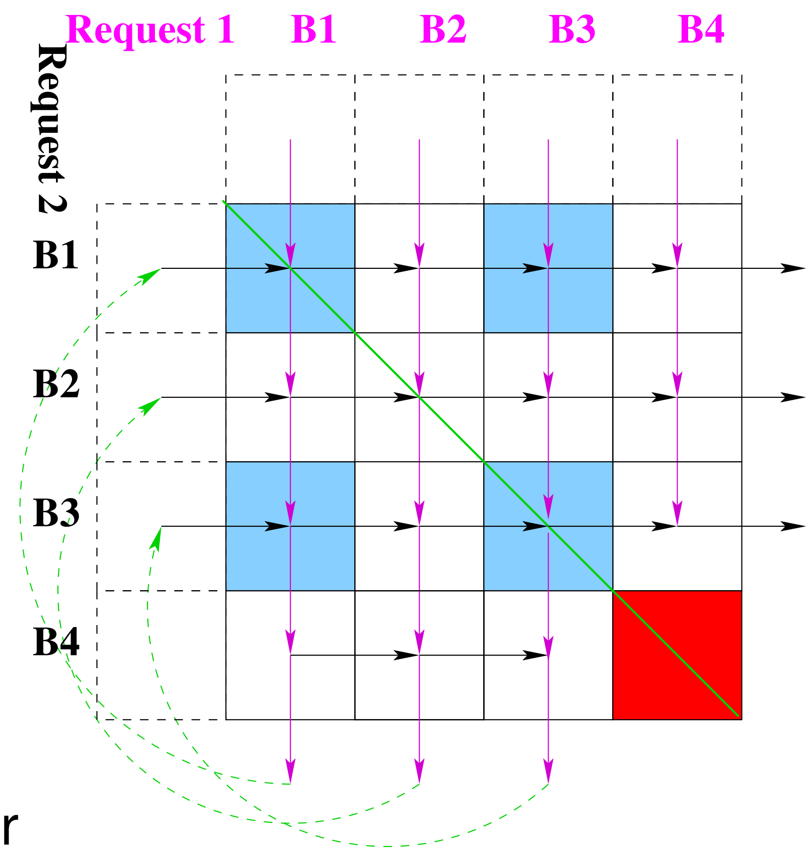
- Polygon Simplification
  - Two methods: 1) delete a concave vertex 2) replace a pair of convex vertices with a single one
  - Simplify polygon and convex hull separately
  - Evolve convex region into concave ones
- Interval Closure

Coho Improvements

- Polygon Simplification
- Interval Closure
  - Projection polygon may not be feasible to others
  - Compute tigher variable bounds to reduce infeasible region
  - Soundness is proved

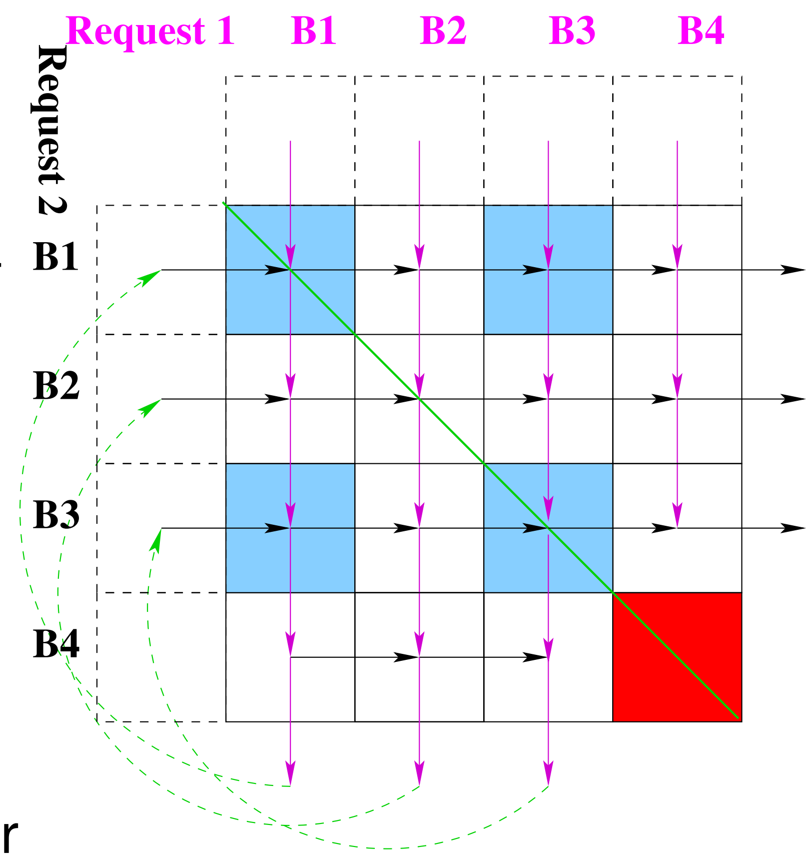
Verifying the Arbiter

- Modeling Concurrent Input Transitions
  - Rising transistion of two request signals can occur at the same time
  - Requests can start at different times and have different rise-time
  - Must account all allowed transitions
- Use Brocket annulus to represent request signal
- Divide the verfication into three phases
- Assume-guarantee approach for verification



Verifying the Arbiter

- Modeling Concurrent Input Transitions
- Use Brocket annulus to represent request signal
  - Sub-divide the annulus to reduce over approximation
  - Reduce the number of reachability problems by exploiting the symmetry of arbiter
- Divide the verfication into three phases
- Assume-guarantee approach for verification

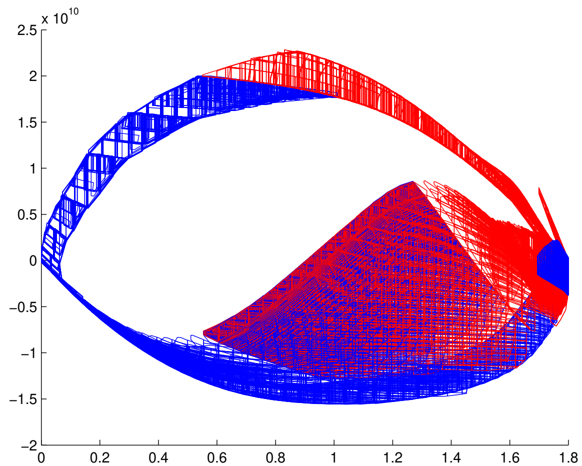


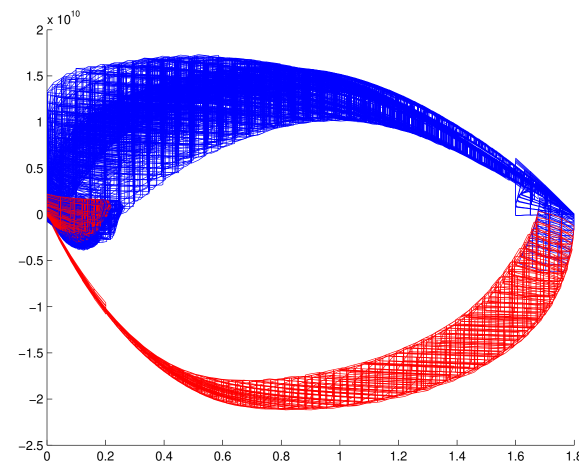
Avoiding the Stiffness

- Internal nodes u_1 and u_2 have much smaller capacitances
- Produce a stiff system
  - Large time steps results in large linearization over approximation
  - Small time steps lead to large number of projection operations
- Ignore the capacitance
  - Create a model for the nMos tetrode
  - Use similar method to compute linear differential inclusion

Results

- Safety Properties
  - Mutual Exclusion
  - Handshake Protocol
  - 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

$$\begin{array}{lll} r_1 \in B_3 & x_1 \in [0.55, 1.3] & g_1 \in B_1 \\ r_2 \in B_3 & x_2 \in [0.55, 1.3] & g_2 \in B_1 \end{array}$$

  - Reset: withdraw grants within 270ps
  - Fairness: grant the other client within 420ps

Conclusion and Future Work

- Conclusion
  - Verify safely and liveness properties of an arbiter circuit with under allowed request signals
  - Present improvements to Coho tool to verify more complicated circuit
  - The metastabbility filter transfers Brocket annulus
- Future Work
  - Solve the stiffness problem
  - Formally describes the specification and translate it automatically
  - Verify more properties and more circuits
  - Combine simulation and formal verification

- Questions?

Thank You!

Appendix

- appendix 1