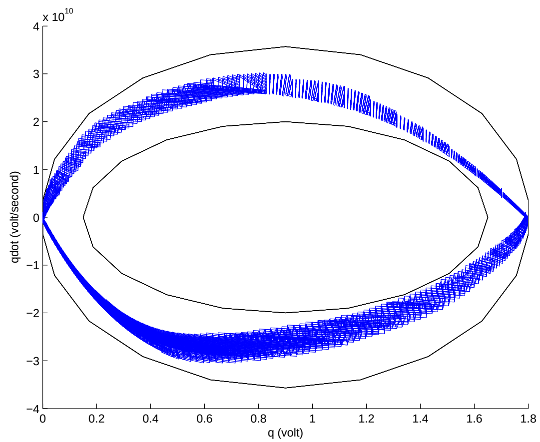
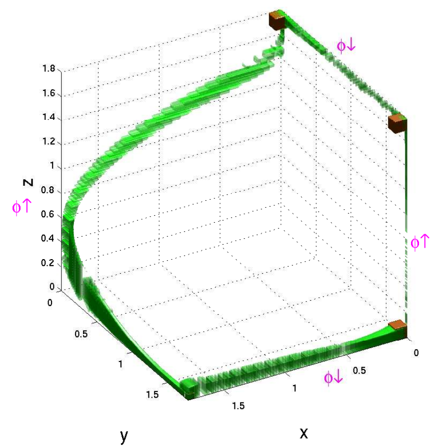
Faster Projection Based Methods for Circuit Level Verification

Chao Yan & Mark Greenstreet

University of British Columbia





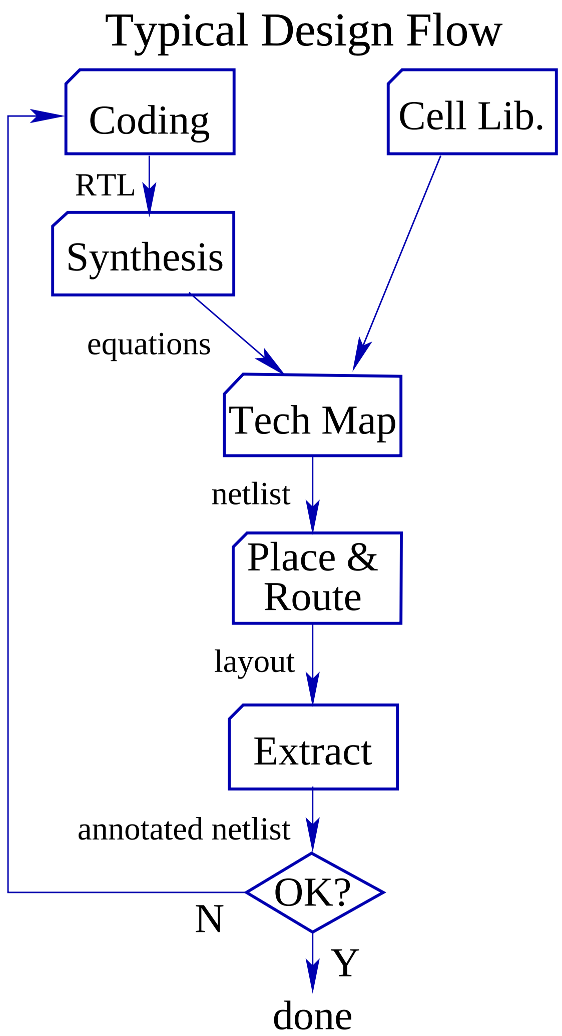
Overview

- Motivation
- Coho: Verifying Digital Circuits with SPICE-level models
- Verification Example: A Toggle Flip-Flop
- Performance Improvement: 15X speed-up in Coho Algorithm

Formal Verification of Digital Circuits Using SPICE-Level Models is Possible and Practical.

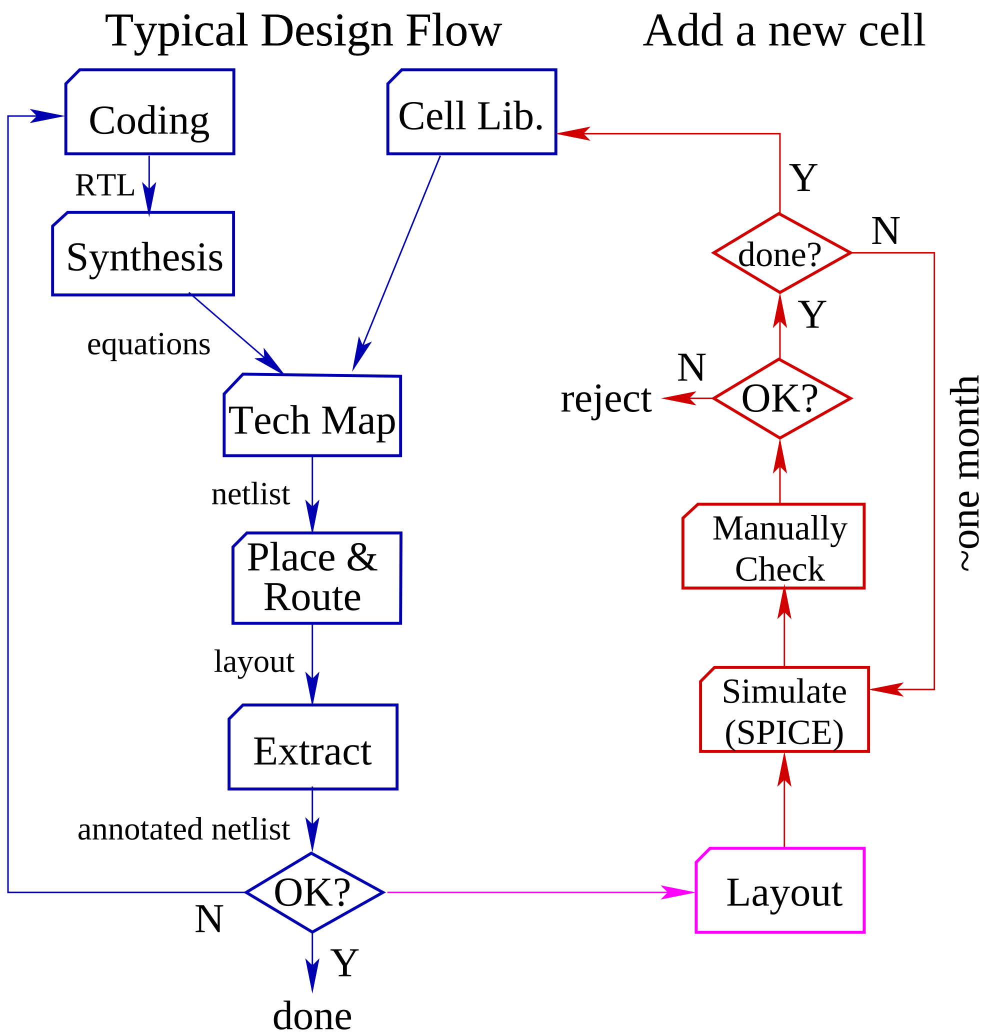
Motivation

- Design Flow



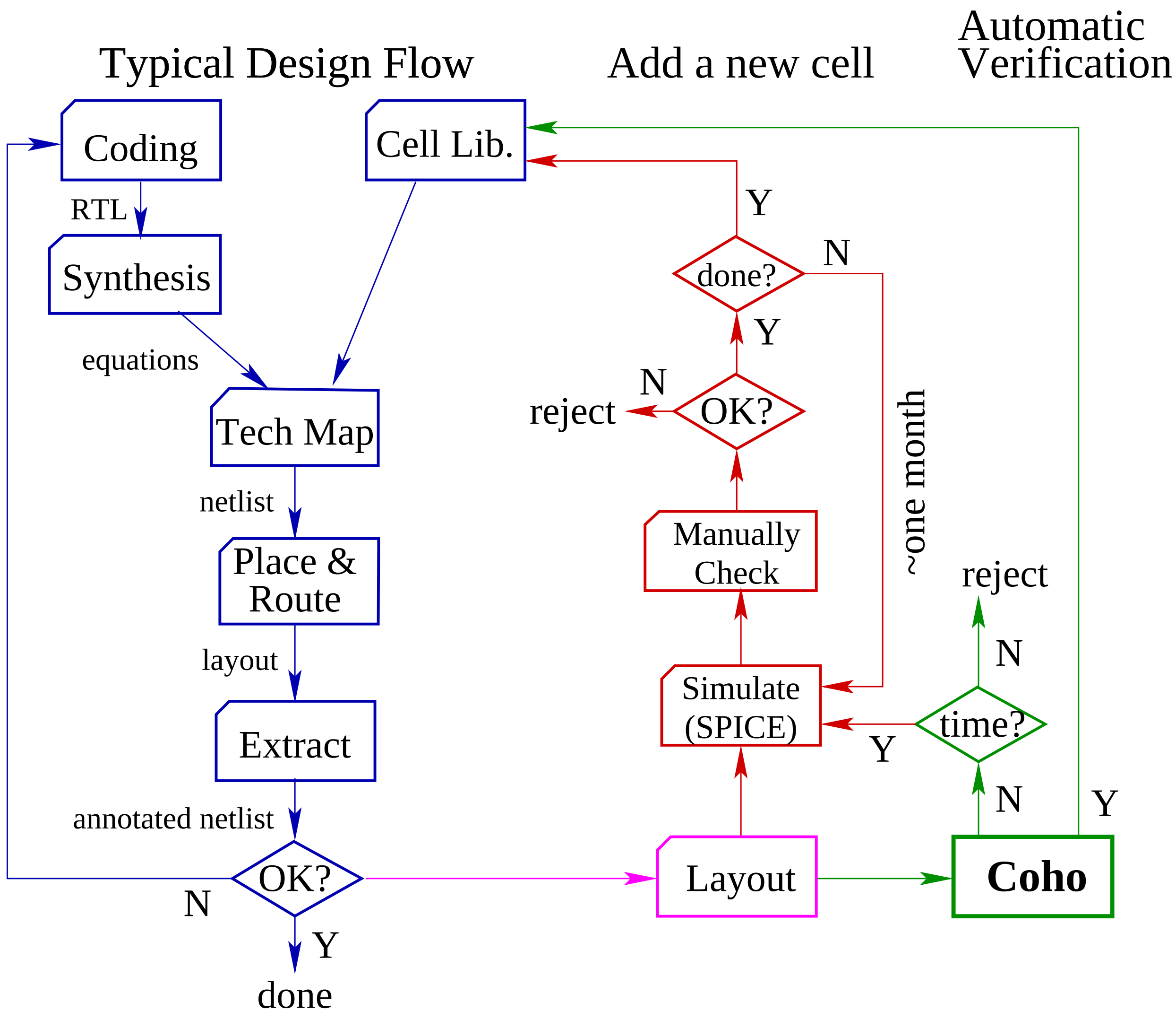
Motivation

- Design Flow



Motivation

- Design Flow



Coho

- A verification tool using reachability method
  - Compute the all possible states of the system
  - Check the specification over all states
- Projection based representation of reachable space
- Model the system by *non-linear ordinary differential equations* (ODEs)

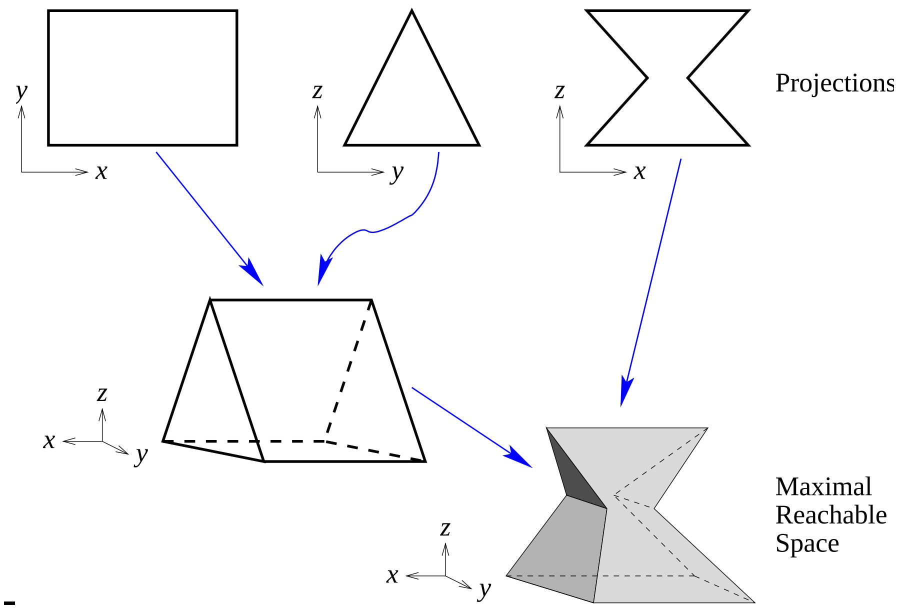
$$\dot{x} = f(x, in)$$

- Approximate the ODEs in small neighborhoods 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.

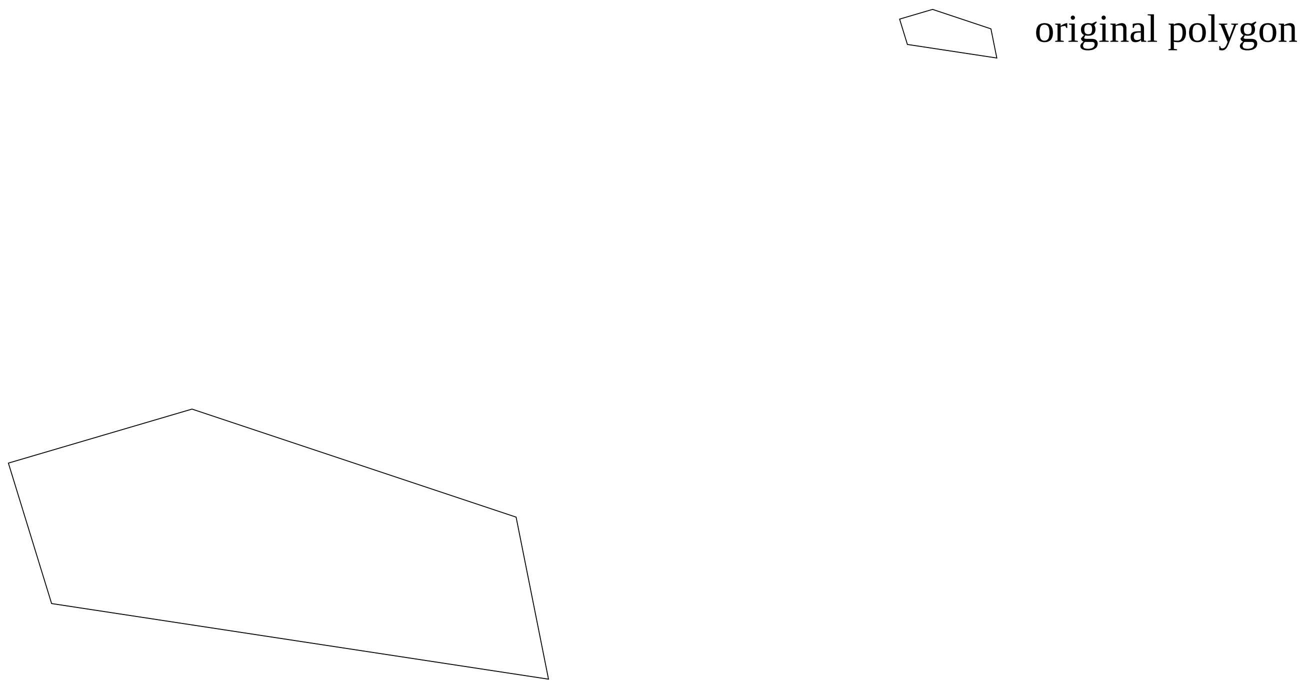


Reachability for Projectagons

- Compute the reachable space contains all trajectories from the projectagon
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.

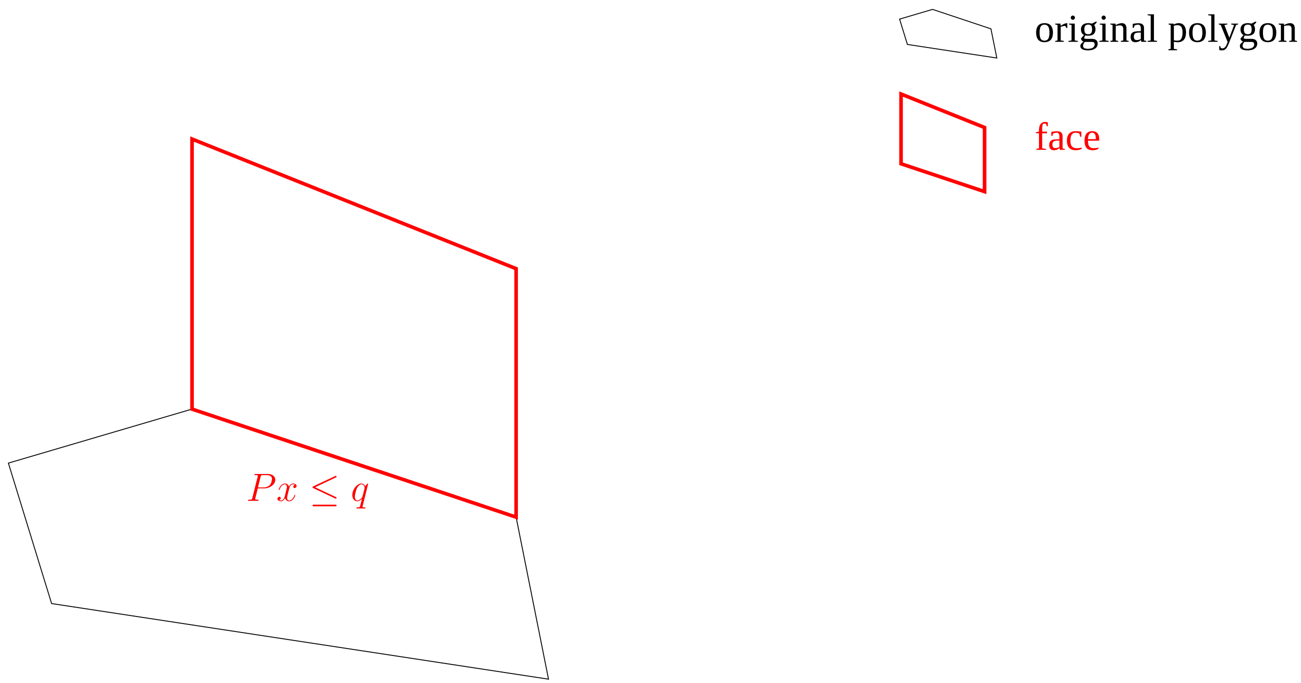
Reachability for Projectagons

- Compute the reachable space contains all trajectories from the projectagon
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.



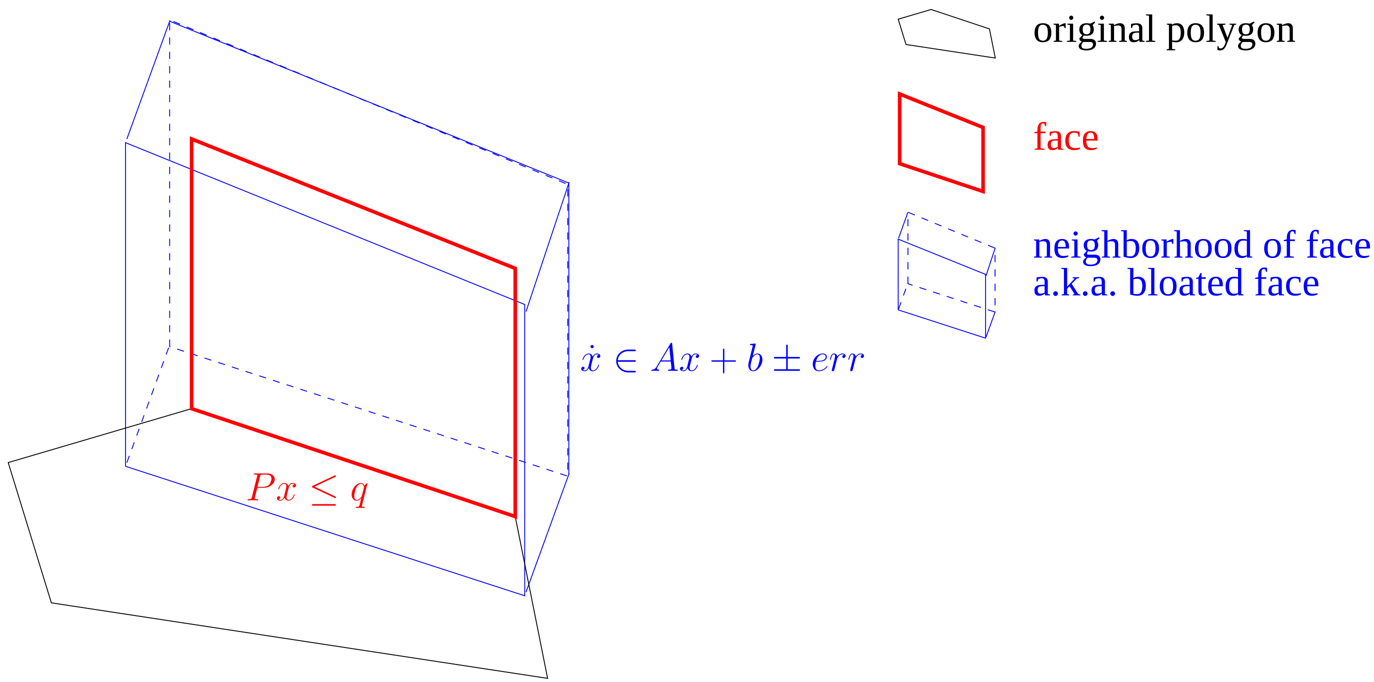
Reachability for Projectagons

- Compute the reachable space contains all trajectories from the projectagon
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.



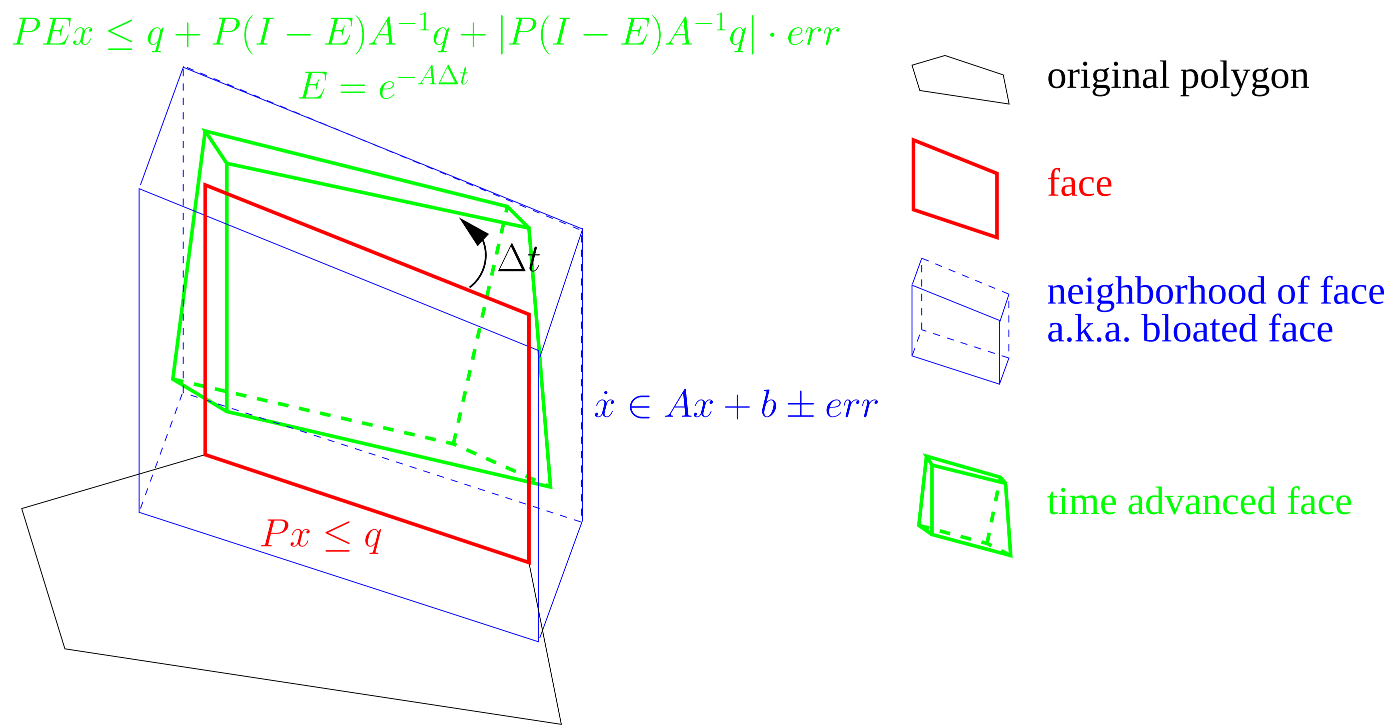
Reachability for Projectagons

- Compute the reachable space contains all trajectories from the projectagon
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.



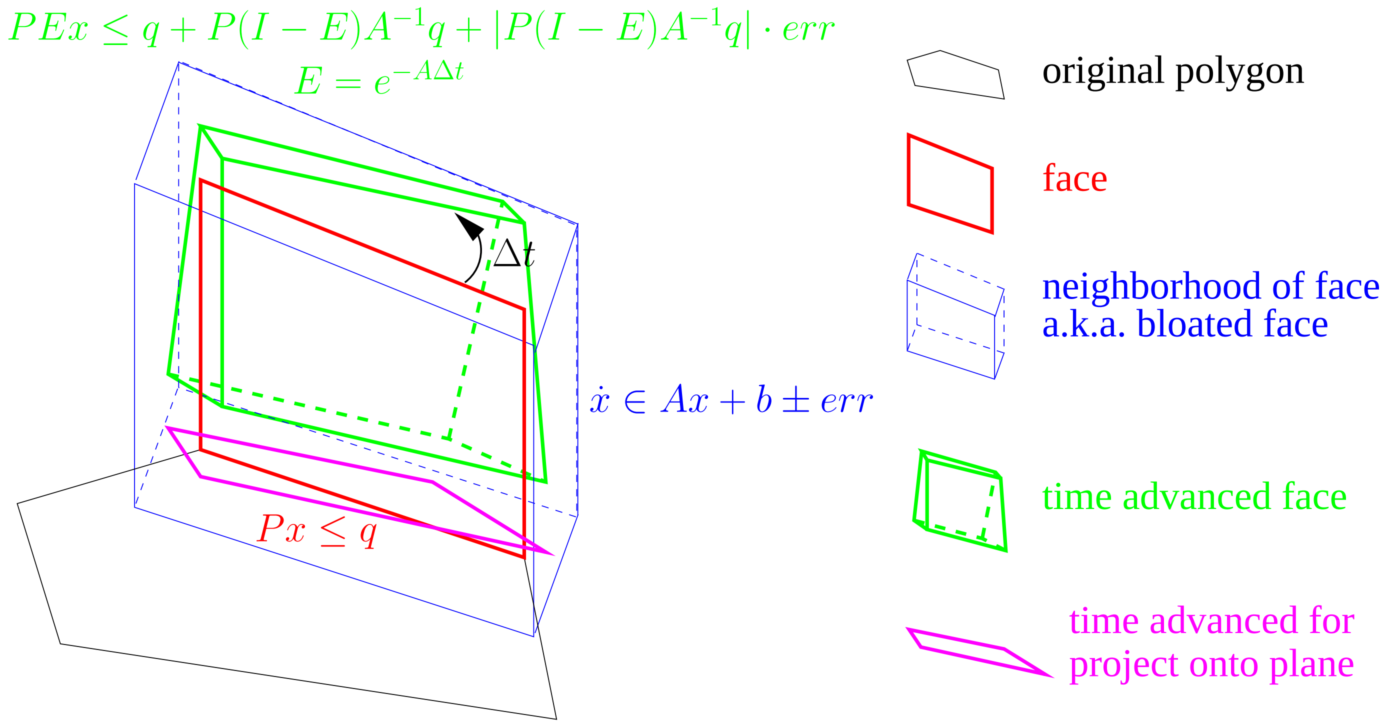
Reachability for Projectagons

- Compute the reachable space contains all trajectories from the projectagon
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.

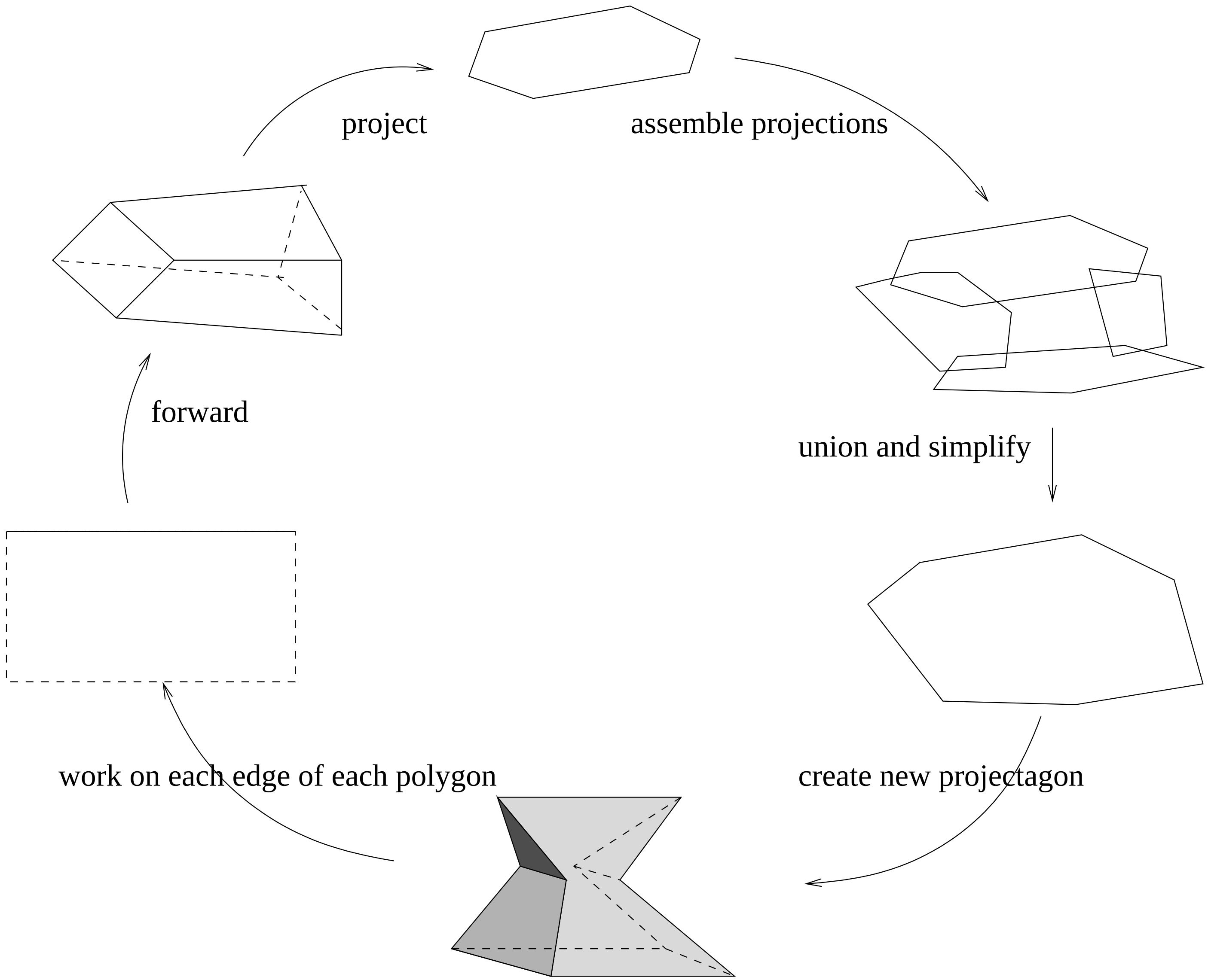


Reachability for Projectagons

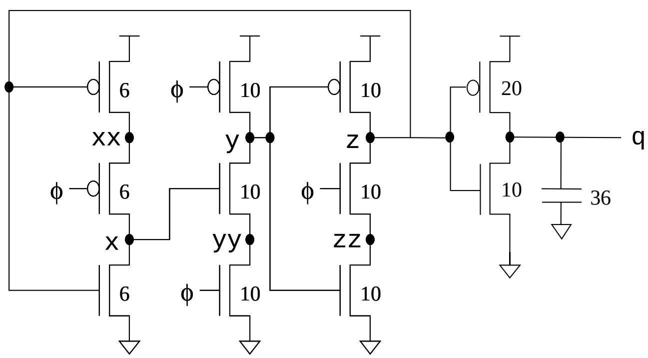
- Compute the reachable space contains all trajectories from the projectagon
- Extremal trajectories originate from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.

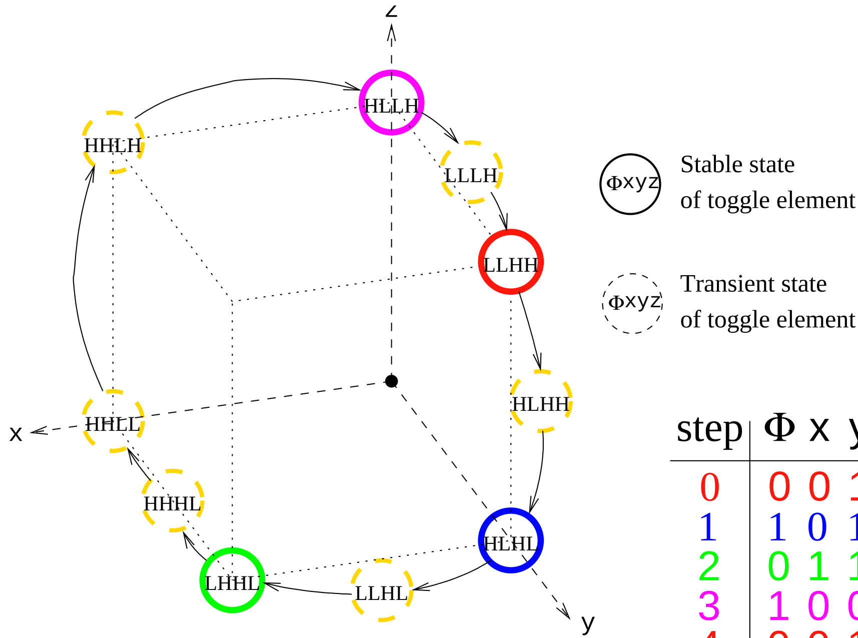


Basic Step of Coho

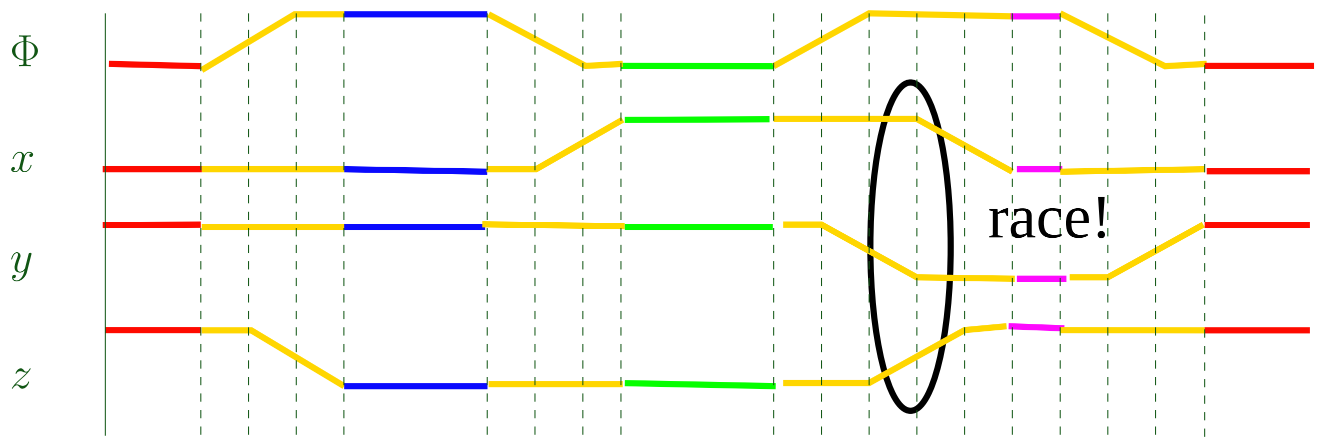


Toggle Circuit

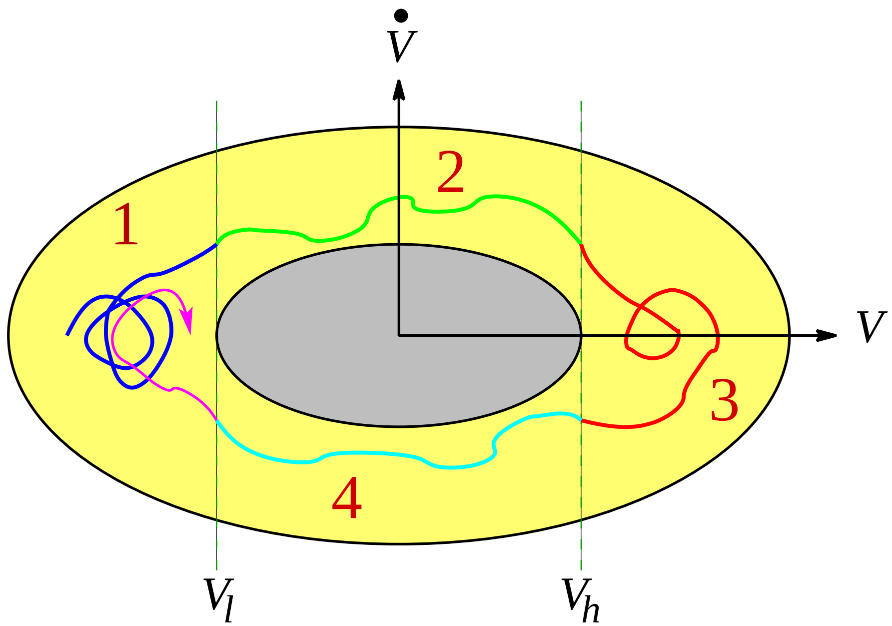


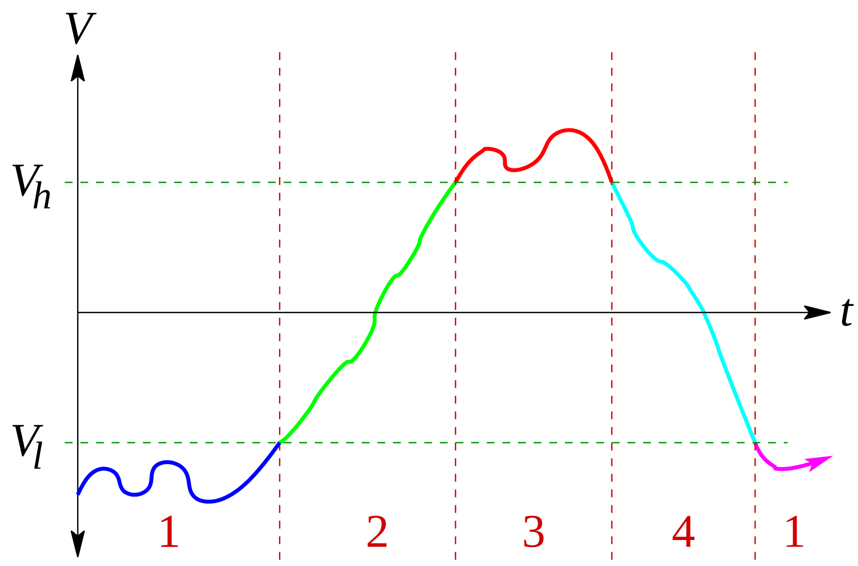


step	Φ	x	y	z
0	0	0	1	1
1	1	0	1	0
2	0	1	1	0
3	1	0	0	1
4	0	0	1	1



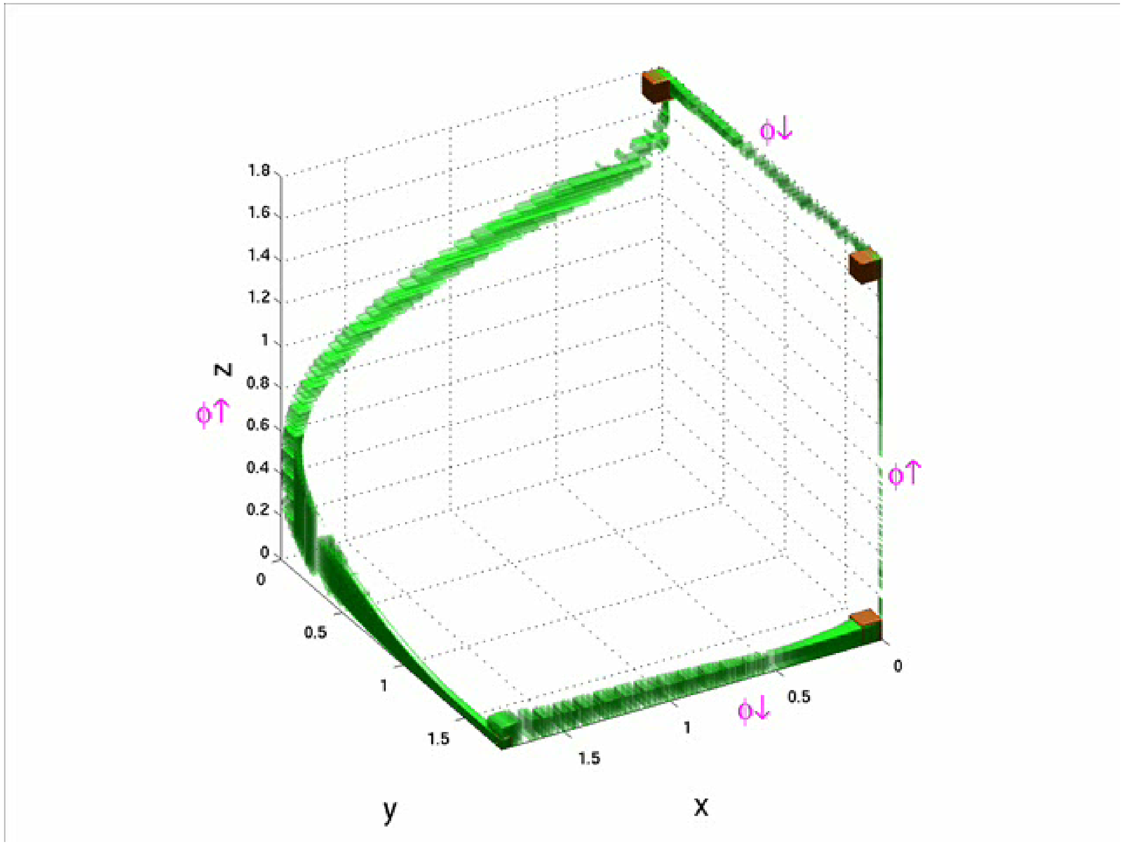
Brockett's Annulus





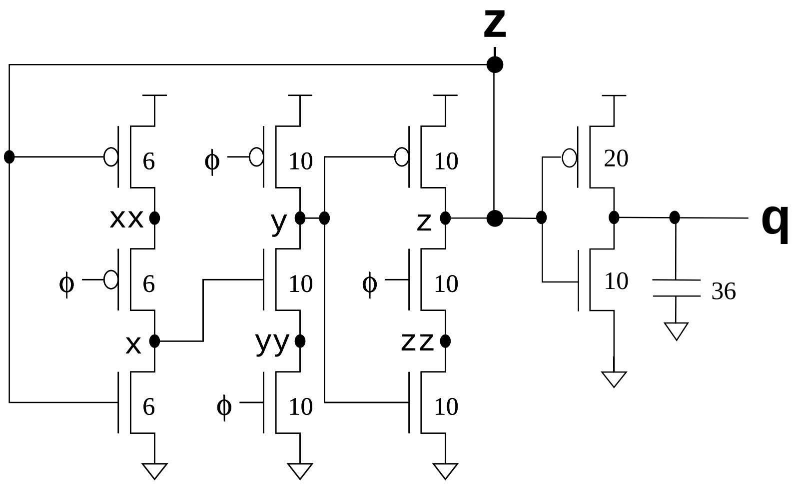
- Use Brockett's annulus to specify entire families of signals for verification
- Region 1 represents a logical low signal. The signal may wander in a small interval.
- Region 2 represents a monotonically rising signal.
- Region 3 represents a logical high signal.
- Region 4 represents a monotonically falling signal.

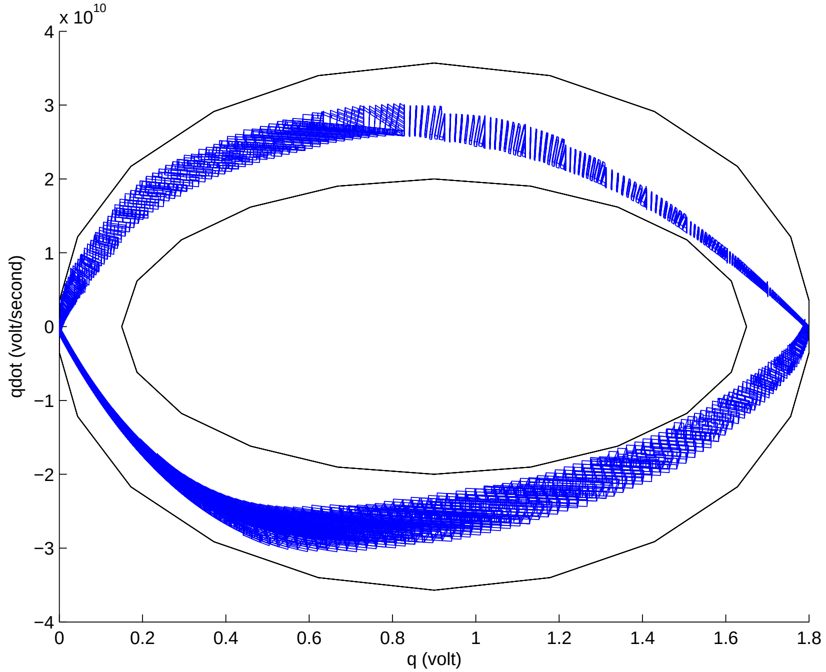
The Invariant Set



- Separate the verification into four phase
- Projectagon at the end is contained by the initial one of next phase
- An invariant set with twice the period of the clock has been established.

Brockett Ring at q



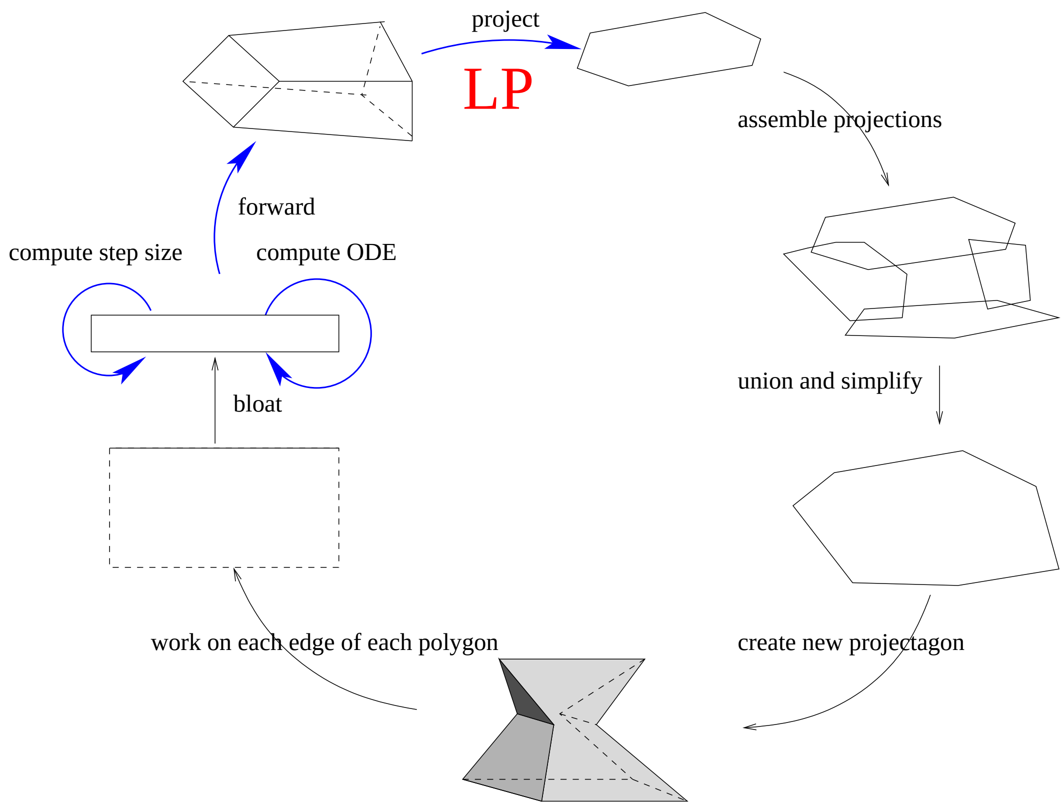


- The output satisfies the same brockett ring of the input clock
- Projectagon is efficient for a seven dimensional system
- Small approximation error to handle the race condition problem

Summary of Coho

- Coho is Sound
  - Works for moderate dimensional systems
  - All approximations overestimate the reachable space
  - Topological properties provide a mathematically rigorous abstraction from continuous to discrete models.
- Coho was Slow
  - Four CPU days to verify the toggle circuit
  - Several thousands of steps for two clock periods
  - Involves substantial manual effort

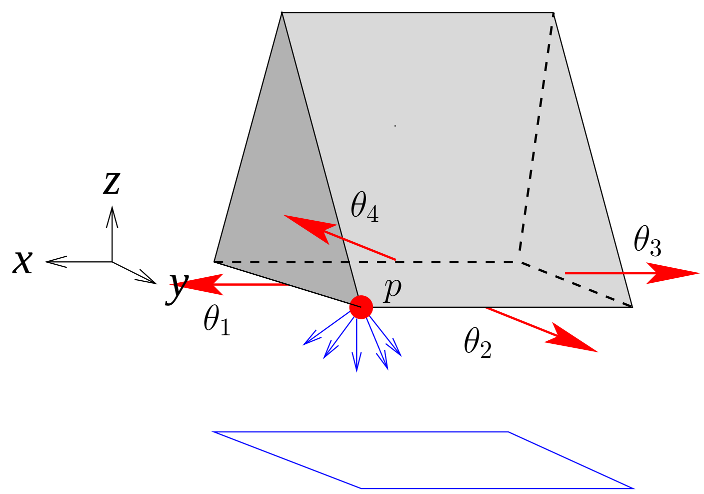
Where does the time go?



- Computing linear model is slow
- Extensive use of linear programming in project algorithm
- Efficient polygon operations
- The number of iterations is determined by the time step

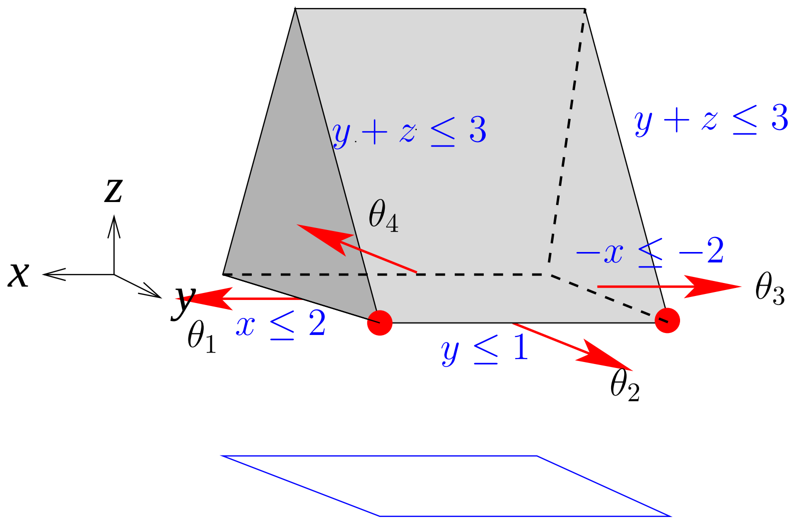
Original Projection Algorithm

- Problem: Project a LP $Ax \leq b$ down onto $(\hat{x}, \hat{y})$ subspace
- Solution: Sweep the optimization direction around the plane
  - Start with a optimization direction with angle of θ_c
  - Find the optimal vertex p by linear programming
  - Compute the optimization direction for which p is no longer optimal
  - Use the critical value θ_n as the angle of next optimization direction



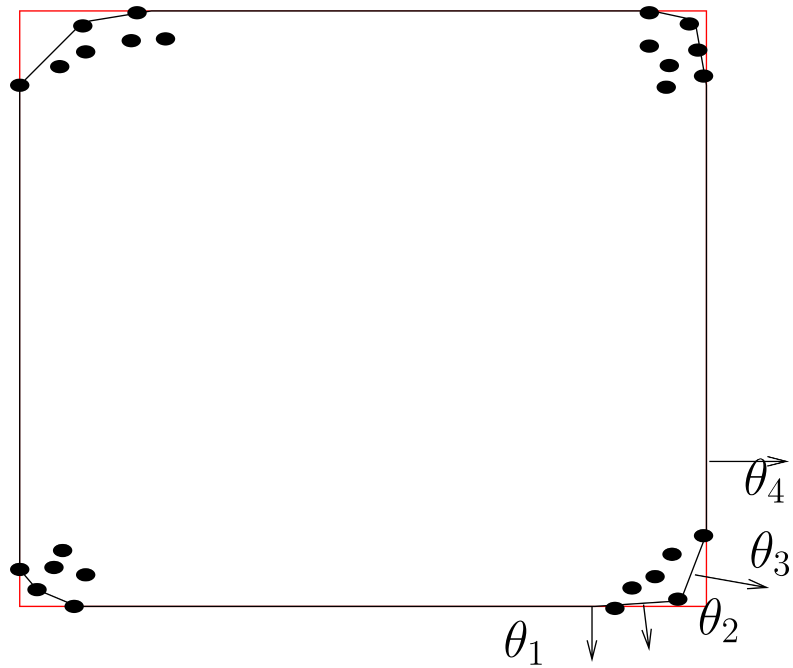
Faster Projection Algorithm

- Single pivot linear program solver
  - Adjacent optimal basis $(\mathcal{B}_c, \mathcal{B}_n)$ are similar: a single pivot distance for most of time
  - Remove infeasible column from $\mathcal{B}_c$
  - Add a new column
  - Check the optimality of the new basis
  - Works about 80% of the time
- Approximated projection algorithm



Faster Projection Algorithm

- Single pivot linear program solver
- Approximated projection algorithm
  - Projection of a face has clusters of very closely spaced vertices because of near degeneracies in the LP.
  - These clusters are discarded by the simplification process.
  - Combine projection and simplification by enforcing a lower bound on the change of θ
  - The number of LPs to solve decreases by 50%

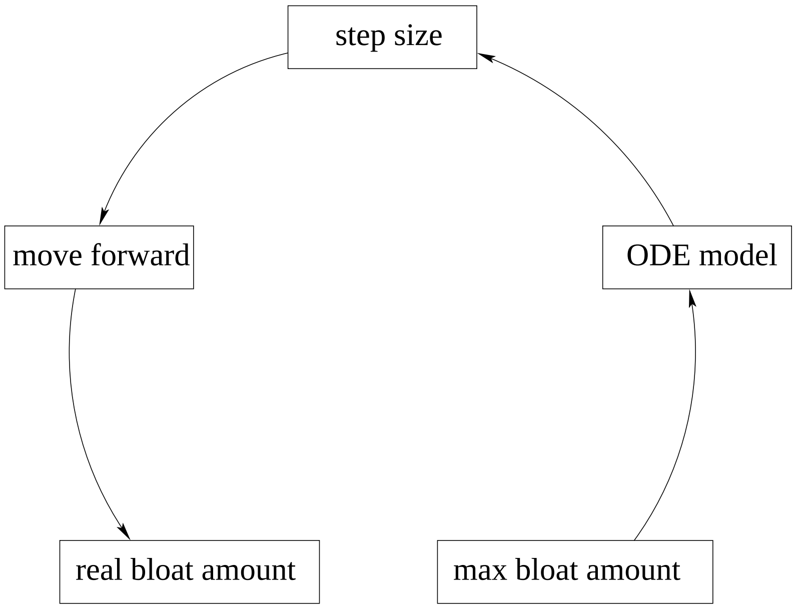


Faster Projection Algorithm

- Single pivot linear program solver
- Approximated projection algorithm
- 2.4x speed-up

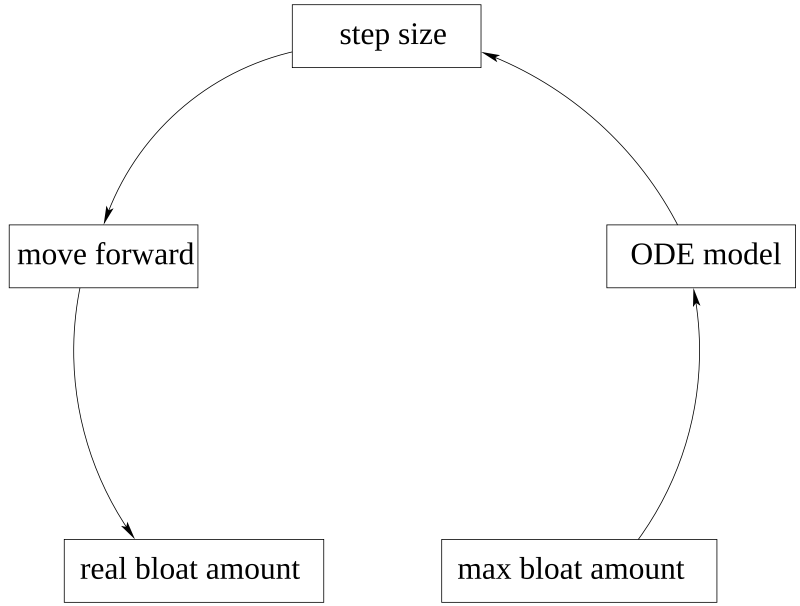
Improved Bloating and Time-Step

- Original algorithm
  - All variables are bloated equally in both positive and negative direction
  - Step size is much smaller than what would actually be safe for given bloat amount
  - Computed reachable space is much smaller than the bloat used to compute model
- Asymmetric and Anisotropic bloating
- Guess-Verify method for larger timestep



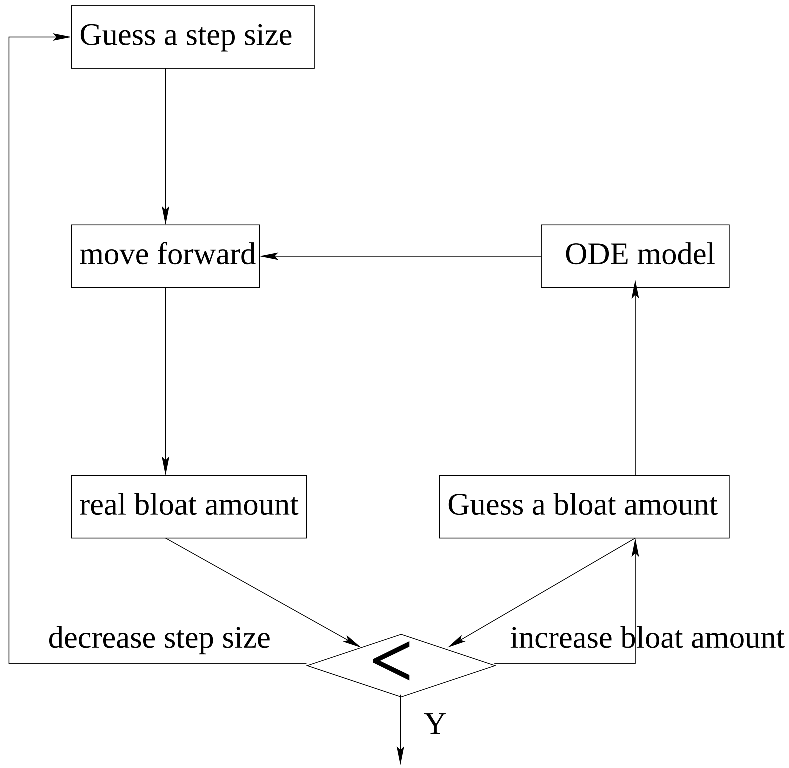
Improved Bloating and Time-Step

- Asymmetric and Anisotropic bloating
  - Asymmetric bloating : positive and negative bloats are different
  - Anisotropic bloating : each variable has its own bloat amount
  - Reduce linearization error by 48% and increase step size
- Guess-Verify method for larger timestep



Improved Bloating and Time-Step

- Asymmetric and Anisotropic bloating
- Guess-Verify method for larger timestep
  - Discard the phase of computing the time step
  - Use the time step and bloat amount of previous step
  - Check that the estimated bloat is sufficient for the estimated time step at the end
  - 2.8x larger time step



Improved Bloating and Time-Step

- Asymmetric and Anisotropic bloating
- Guess-Verify method for larger timestep
- 6x speed-up

Conclusion and Future Work

- Conclusion
  - Demonstrate a new reachability method to verify a real circuit
  - Model the circuit with SPICE-level, non-linear differential equations.
  - Projection based representation of reachable space
  - 15x (4 days vs. 400 minutes) reduction in computation time and significant reductions in the approximation errors
- Future Work
  - Develop more accurate circuit model
  - Parallel computing
  - Verify more circuits
  - Apply Coho to hybrid systems
  - Compare with other tools, checkMate, d/dt, HyTech, etc.